

HIGHLY ACCURATE LINEAR CLASSIFIER WITH APPLICATIONS IN HEALTH
INSURANCE COVERAGE

Songkomkrit Chaiyakan

A Dissertation Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy Program
in Business Analytics and Data Science
Graduate School of Applied Statistics
National Institute of Development Administration
Academic Year 2024

Copyright of National Institute of Development Administration

Thesis Title	HIGHLY ACCURATE LINEAR CLASSIFIER WITH APPLICATIONS IN HEALTH INSURANCE COVERAGE
By	Songkomkrit Chaiyakan
Field of Study	Business Analytics and Data Science
Thesis Advisor	Assistant Professor Preecha Vichitthamaros, Ph.D.

Accepted by Graduate School of Applied Statistics, National Institute of Development Administration in Partial Fulfillment of the Requirements for the Doctoral Degree

..... Dean of Graduate School of Applied Statistics
(Siwiga Dusadenoad, Ph.D.)

DISSERTATION COMMITTEE

..... Chairman
(Associate Professor Ohm Sornil, Ph.D.)

..... Thesis Advisor
(Assistant Professor Preecha Vichitthamaros, Ph.D.)

..... Examiner
(Associate Professor Surapong Auwatanamongkol, Ph.D.)

..... Examiner
(Associate Professor Pachitjanut Siripanich, Ph.D.)

..... External Examiner
(Assistant Professor Boonyarit Intiyot, Ph.D.)

6310432002: MAJOR BUSINESS ANALYTICS AND DATA SCIENCE

KEYWORDS: BOX CLASSIFICATION / OPTIMIZATION / 0-1 MIXED INTEGER PROGRAMMING / DIMENSIONALITY REDUCTION / CONTINUOUS DATA / CATEGORICAL DATA / HEALTH INSURANCE

SONGKOMKRIT CHAIYAKAN: HIGHLY ACCURATE LINEAR CLASSIFIER WITH APPLICATIONS IN HEALTH INSURANCE COVERAGE. ADVISOR: Assistant Professor Preecha Vichitthamaros, Ph.D., 484 pp.

This work proposes a multiclass box classifier both theoretically and empirically proven to produce the highest training accuracy through the rigorous formulation of 0-1 mixed integer programming problem. It can also determine significant factors. Unlike a decision tree classifier well-known for simplicity and fast execution, the proposed classifier has control over a maximal number of features of interest, whether continuous or categorical, and a number of splitting values on all features. The use of this method is illustrated on 2020 Current Population Survey (CPS) Annual Social and Economic Supplement (ASEC) health insurance dataset with, as a result of the exponential time complexity of the model, only three independent variables univariately preselected by the SelectKBest technique. Compared to decision tree classifiers of different depths, the proposed classification model can keep a balance between the number of total splitting values and the number of decision boxes, and it achieves a relatively high training accuracy at the expense of significantly high computational time and storage usage. Nonetheless, both give the same set of contributing factors. The fast algorithm of decision box merging is also suggested when the number of selected features can be further reduced after optimization.

Graduate School:	Applied Statistics	Student's Signature
Field of Study:	Business Analytics and Data Science	Advisor's Signature
Academic Year:	2024	

Acknowledgements

I would like to express my thankfulness to my dissertation advisor Assistant Professor Preecha Vichitthamaros for his consistent help throughout the entire dissertation. Further thanks extend to Jessica Barnett and Laryssa Mykyta, U.S. Census Bureau, for their advice on public use files and technical documentation of Current Population Survey (CPS) Annual Social and Economic Supplement (ASEC) health insurance datasets. I am especially indebted to my parents, Songsak and Kanjanee, and to my sister, Kanokkan, for their continual support, encouragement, and love. This dissertation is partially supported by National Institute of Development Administration for publication and Google Cloud research program for free access to a remote compute engine.

Contents

	Page
Abstract	iii
Acknowledgements	iv
Contents	v
List of Tables	vii
List of Figures	viii
List of Codes	ix
Nomenclature	x
Chapter	
1 Introduction	1
1.1 Objectives	2
1.2 Limitations	2
2 Literature Review	3
2.1 Health Insurance Coverage	3
2.2 Feature Selection	4
2.2.1 Decision Tree	4
2.2.2 SelectKBest	4
3 Research Methods	6
3.1 Overview	6
3.2 SSH Key Generation	6
3.3 Remote Virtual Machine Setup	9
3.3.1 Specifications	9
3.3.2 SSH Key-Based Authentication	9
3.3.3 Python Installation	9
3.3.3.1 Introduction to Compilation in C	10
3.3.3.2 Basic Object Types	10
3.3.3.3 String Interning	13
3.3.3.4 Configuration and Build	14
3.3.4 Backup to OCI Object Storage	17
3.3.4.1 Introduction to OCI	17
3.3.4.2 OCI Object Storage	19
3.4 GitHub Repository	20

Chapter	Page
3.5 Health Insurance Dataset	23
3.5.1 Background	23
3.5.2 Scope of Study	24
3.5.3 Metadata	24
3.5.4 Python Modules	106
3.5.5 Python Classes	114
3.5.6 Exploratory Data Analysis (EDA)	120
3.5.7 Data Encoding	235
3.5.8 Sampling using SelectKBest	238
3.5.9 Setting Number of Variable Splits	241
4 Proposed Classifier	244
4.1 Proposed Model for Selecting Continuous Factors	244
4.2 Selection of Mixed-Type Features	249
4.3 CPLEX OPL Modeling	253
4.4 Recalculation of Decision Boxes	277
5 Results on Health Insurance	314
5.1 Training Data	314
5.2 Decision Tree	317
5.3 Proposed Model	325
6 Concluding Remarks	376
Appendix	
Appendix A CPLEX Engine Log	379
Biography	484

List of Tables

Table	Page
3.1 Example of advanced NTFS permissions in Windows	8
3.2 Python options for third-party dependencies	16
3.3 Basic Git commands	20
3.4 Class codes of insurance coverage combination	24
3.5 Categories of health insurance factors	121
3.6 Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB)	125
4.1 Relevant CPLEX parameters	253
5.1 Cross tabulation of sample data by preselected variables and health insurance coverage types	315
5.2 Comparison between multiple decision tree and proposed classifier	327
5.3 Splitting values on features of multiple decision tree and proposed classifier	328
5.4 Training accuracy, execution time, minimum storage usage, relative MIP gap and number of inconsistent data across all iterations	329
5.5 Selected variables and groups of values across all iterations	330
5.6 Decision regions and predicted class labels across all iterations	340
5.7 Inconsistency between numerical CPLEX and true decision regions	352

List of Figures

Figure	Page
3.1 Directory tree structure of the template GitHub repository	22
5.1 Decision tree with depth 3 and 8 leaf nodes	317
5.2 Decision tree with depth 4 and 12 leaf nodes	318
5.3 Decision tree with depth 5 and 16 leaf nodes	319

List of Codes

Code	Page
3.1 Utility module	106
3.2 Encoding module	107
3.3 Dataset module	108
3.4 EDA module	110
3.5 ThesisExtension class	114
3.6 Data class	117
3.7 Info class	119
3.8 Exploratory data analysis (EDA)	233
3.9 Data encoding	235
3.10 SelectKBest	238
3.11 Setting number of variable splits	241
4.1 Main OPL model	254
4.2 Box classifier OPL model	272
4.3 Basic utility for recalculation of region	279
4.4 Typecasting	285
4.5 Recalculation of regions	287
4.6 Feature selection	291
4.7 Cuts or split values	292
4.8 True decision regions	295
4.9 CPLEX decision regions	297
4.10 Classification correctness	299
4.11 Final mixed box classifier	300
5.1 Gini-based decision tree classifier	320

Nomenclature

\tilde{d}	full dimension of given training instances
d	number of both continuous and categorical features of interest
d_{cat}	number of categorical features of interest
$\tilde{\mathcal{C}}_{\text{cont}}$	index set of given continuous features
$\tilde{\mathcal{C}}_{\text{cat}}$	index set of given categorical features
$\mathcal{C}_{\text{cont}}$	index set of new continuous features before optimization
\mathcal{C}_{cat}	index set of intermediate categorical features before optimization
\tilde{x}^i	given training instance i
x^i	training instance i as a classifier input of lower continuous and full categorical dimensions
x_j^i	value of feature j of instance x^i
y_k^i	whether a given instance \tilde{x}^i is in class k
$c_{j,\tilde{j}}$	whether a new continuous feature j comes from an original continuous feature \tilde{j}
f_j	whether categorical feature j is selected or, equivalently, significant
p_j	number of splitting values on feature j
$b_{j,q}$	q^{th} splitting value on continuous feature j
u_j	new group labels on categorical feature j
v_{j,x_j^i}	new group label of instance x_j^i on categorical feature j
B	number of total decision boxes
S_β	β^{th} decision box
$\alpha_{j,q}^i$	whether x_j^i is in open interval $(b_{j,q}, b_{j,q+1})$
M	sufficiently large positive number
m_j	sufficiently small positive number on feature j that can distinguish individual feature values of x_j^i
$l_{j,q}^i$	$\alpha_{j,q}^i(b_{j,q} + m_j)$
$r_{j,q}^i$	$\alpha_{j,q}^i(b_{j,q+1} - m_j)$
γ_β^i	whether instance x_j^i is in decision box S_β
Θ_β	set of most frequent classes in decision box S_β
h_β	negative value of number of correctly classified training instances

CHAPTER I

INTRODUCTION

Social science research heavily relies on the traditional use of logistic regression or structural equation modeling (SEM) to explore or confirm the linkage between multiple factors with the ultimate goal of causal explanation. In addition to the significance test of coefficients, the utilization of mediators, moderators, confounders and covariates provides the convincing magnitude and direction of estimated effects. On the rare occasion of classification with numerous independent variables measured on nominal scales, the excessive number of required dummy variables nevertheless imposes a limitation on these two approaches.

To address this problem, classification algorithms in machine learning are used to identify key characteristics of a separate group despite lack of important statistical tests. For example, a decision tree constructs a set of rules individually formed by minimal attributes to fully describe a training data, and a neural network employs a hidden layer to account for nonlinear interaction between attributes and therefore increases model accuracy. The first maximizes an information gain, whereas the latter minimizes a residual sum of square. Both objective functions are usually smooth and enable real-time data processing.

Despite their advantage, a decision tree and a neural network may provide undesirable inaccuracy, evidently because their performance metrics are not accuracy. As a result, a multiclass box classifier developed from conventional support vector machine (SVM) through the application of 0-1 mixed integer linear programming (MILP) by counting the number of misclassified instances through majority voting will be proposed in the dissertation to ensure maximum accuracy without overfitting simply due to its linearity. In this case, external testing seems redundant unless a training data contains an outlier. As early-stage research, the classifier will serve no purpose of real-time analytics. This modified approach will be adopted for illustrative purposes to examine without consideration of interrelationship contributing factors, including their groups of values, on coverage types of health insurance in the United States in 2019. The classification model is trained on the entire survey data because in this dissertation all responses collected from different participants are of equal importance and no prediction about future health insurance coverage is made.

1.1 Objectives

1. To propose a multiclass box classifier that yields highest training accuracy.
2. To apply the proposed classification method to investigate significant factors, whether continuous or categorical, influencing health insurance coverage.

1.2 Limitations

1. Nonlinear classification in addition to logistic regression are beyond the scope of the study because no interaction between health insurance factors is investigated and splitting values on any two factors should be independent.
2. The health insurance sample data only includes Americans. It was collected in 2020 to reflect health insurance coverage for entire calendar year 2019.
3. Despite its high training accuracy, the proposed classifier takes a significantly long training time and requires enormous space to store a branch-and-cut tree. Its approximation algorithm is not developed in this dissertation although mitigating both problems to some extent. Furthermore, only three factors are preselected and investigated with a sample size of 100. Even in this simple circumstance, the model training lasts longer than a day, but the early-exit classifiers are nonetheless more accurate and parsimonious than a Gini-based decision tree.

CHAPTER II

LITERATURE REVIEW

2.1 Health Insurance Coverage

A variety of statistical tools have long been used to study the factors related to health insurance coverage of multiple subpopulations across different countries. These analytical techniques include linear probability modeling ([Cebula, 2006](#)), probit regression analysis ([Mulenga et al., 2021](#)) and logistic regression analysis ([Jin et al., 2016](#); [Dolinsky and Caputo, 1997](#); [Markowitz et al., 1991](#)).

Generally, health insurance coverage across the U.S. states was positively associated with median family income, female labor force rate, the proportion of population aged 65 and over, and it was negatively linked with the percentages of household with husband absence and Hispanic household ([Cebula, 2006](#)). Psychological characteristics also greatly affected the influence of demographic factors among American women ([Dolinsky and Caputo, 1997](#)). After controlling for psychological variables, health status and employment were significant determinants only for married and unmarried women respectively. Income and education played important roles in both groups. Americans aged 18 to 24 with permanent, full-time employment were more likely to be insured than those with permanent, part-time employment ([Markowitz et al., 1991](#)). This trend became reverse specifically for the students. Low income, less education, rural residence, unmarried status, Hispanic ethnicity and Western residency were indicators of being uninsured in general.

Outside the United States, many research works on health insurance coverage have also been of interest. Income, education, health status and employment correlated with the coverage types among Chinese people aged 45 and over ([Jin et al., 2016](#)). Males dominated in both public and private health insurance. Migrants appeared to be covered by both rural and urban public insurance, private insurance or no insurance in comparison to local residents. Rural residents were more inclined to have public insurance coverage. Furthermore, private health insurance in Zambia tended to be purchased by males with service, skilled and unskilled occupations and rural residency as well as women in marital union and clerical duties ([Mulenga et al., 2021](#)).

2.2 Feature Selection

2.2.1 Decision Tree

Each parent node partitions a feature space by splitting a specific training variable into two intervals, left and right nodes (Scikit-learn, 2024a). A splitting value is chosen to minimize the weighted average of the impurities of both child nodes by their number of training instances. This dissertation uses as an impurity measure the Gini index defined by the probability of a sample at a node being wrongly classified.

A categorical feature can be handled by one-hot encoding. A multiway tree can be transformed into a binary tree by performing the following operation recursively. For a node having more than two successors, its new successor is created by negating the predicate of one of its preexisting successors and becomes the predecessor of the rest. This procedure maintains the decision regions.

2.2.2 SelectKBest

The SelectKBest technique (Scikit-learn, 2024b) serving as univariate feature selection finds top K features relating to a target variable based on a score function, for example the mutual information for a discrete target in this dissertation. The mutual information (Cover and Thomas, 2005) is a statistic for measuring relationship between two random variables or in practice two datasets.

Definiton 2.1. The *Kullback-Leibler distance* $D(f||g)$ between two densities f and g is defined by

$$D(f||g) = \int f \log \frac{f}{g}.$$

Definiton 2.2. The *mutual information* $I(X;Y)$ between two random variables with joint density $f(x,y)$ is defined as

$$I(X;Y) = D(f(x,y)||f(x)f(y)).$$

Two random variables share no mutual information, i.e. $I = 0$, only when both are independent. Suppose X is a training variable and Y a discrete target or class. A continuous feature requires an estimation of mutual information, for example by the k -nearest neighbor method (Ross, 2014), because its true probability remains practically unknown. Suppose the k -nearest neighbor of a training instance x^i of the same class has m_i instances of all classes and there are N_i out of N that share the same class with x^i . Compute

$$I_i = \psi(N) - \psi(N_i) + \psi(k) - \psi(m_i)$$

where the digamma function ψ is the logarithmic derivative of the gamma function. The mutual information $I(X; Y)$ is estimated by averaging I_i over all training instances.

Definiton 2.3. The *gamma function* Γ and *digamma function* ψ are defined on the set of positive real numbers by

$$\Gamma(z) = \int_0^{\infty} t^{z-1} e^{-t} dt$$

and

$$\psi(z) = \frac{d}{dz} \log \Gamma(z)$$

respectively.

CHAPTER III

RESEARCH METHODS

3.1 Overview

1. Propose a multiclass box classifier which is able to predict continuous contributing factors, produces disconnected decision regions and provides minimum misclassification.
2. Extend the classifier when certain features of training data are allowed to be categorical.
3. Connect to a cloud virtual machine using secure shell (SSH) and install Python from source as well as CPLEX.
4. Illustrate the use of the proposed classification method on the health insurance dataset.
5. Compare multiple facets of results with the use of a decision tree.
6. Back up the scripts and results to Oracle Cloud Infrastructure (OCI) Object Storage.
7. Publish the project to GitHub.

3.2 SSH Key Generation

The Secure Shell (SSH) protocol is employed for secure connection to a remote compute engine through one-way client authentication by a pair of asymmetric keys: private and public. SSH keys can be generated with the OpenSSH command `ssh-keygen` by using a native SSL/TLS library provided by an operating system: Secure Channel (Schannel) in Windows or OpenSSL in Linux. The latter keys are very specific to a currently active OpenSSL version especially when an alternative OpenSSL is manually built and installed. In this dissertation, the SSH keys are created on a local computer with the elliptic-curve Ed25519 algorithm ([Bernstein et al., 2012](#)), proven to be faster and more efficient than the RSA algorithm ([Rivest et al., 1978](#)).

```
cd ~/.ssh  
ssh-keygen -f <output_keyfile> -C <comment> -t ed25519
```

A Google Cloud virtual machine requires the comment at the end of a public key file to be a Google username.

Since the dissertation results are uploaded to a GitHub repository using SSH, an additional key pair specific to this purpose is suggested to tighten security. Unless the default private key `id_rsa` is used for authentication, a `Host` (named host) must be specified in the configuration file `~/.ssh/config` by, for example, `HostName`, `User` (username) and `IdentityFile` (path to a private key file). The `Host` field can have wildcard patterns to match multiple hosts. If it contains an IP address or a domain name, the `HostName` field becomes unnecessary.

```
Host <named_host>
    HostName <hostname>
    User <username>
    IdentityFile <private_keypath>
```

Unlike Windows, Linux has the `.ssh` directory hidden, directly by the use of a dot character at the beginning, and partially inheritable POSIX access control list (ACL). A Linux parent directory does not reapply its new ACL to existing descendants, and it simply acts as during path resolution a gate with its execute permission. An SSH connection may be refused in the case of loose private key permissions to prevent privilege escalation attacks. In order to resolve this issue, the principle of least privilege (PoLP) should always be applied to generated keys. Basically, only a key owner can read his/her private key, and the read-only permission on a public key can be granted up to everyone.

In Linux, there are three POSIX permission levels: owner, group and other. Each level is represented by three permission bits: read (r), write(w) and execute (x). They are usually rewritten in base 10, ranging from 0 to 7. The `chmod` command is used to set all three levels of permission with three numerical digits.

```
chmod 400 <private_key>
chmod 444 <public_key>
```

In Windows, the command `icacls` is used, and additional rights can be denied due to more fine-grained permission control as displayed in Table 3.1. An SSH key should be hidden and have no inherited NTFS permission. Its ownership should also be nontransferable. In this dissertation, a key pair is generated on a personal computer (PC) on which the key owner is the only administrator. Under this circumstance, the private key is not accessible to the SYSTEM account. Furthermore, the Administrators group can only read, but neither change nor delete, its content, regular and extended attributes, and permissions. This set of access privileges on the public key can also be granted up to the Everyone group and the SYSTEM account.

```

icaccls <key> /inheritancelevel:d
icaccls <key> /grant ${Env:USERNAME}:F Administrators:F SYSTEM:F `
    Everyone:F
attrib +h <key>
icaccls <key> /remove ${Env:USERNAME} Administrators SYSTEM `
    Everyone
icaccls <key> /deny "${Env:USERNAME}:(WD,AD,WA,WEA,DE,WDAC,WO)" `
    "Administrators:(WD,AD,WA,WEA,DE,WDAC,WO)"
icaccls <key> /grant ${Env:USERNAME}:R Administrators:R
icaccls <private_key> /deny SYSTEM:F
icaccls <public_key> /deny "SYSTEM:(WD,AD,WA,WEA,DE,WDAC,WO)" `
    "Everyone:(WD,AD,WA,WEA,DE,WDAC,WO)"
icaccls <public_key> /grant SYSTEM:R Everyone:R

```

Table 3.1: Example of advanced NTFS permissions in Windows

Permission	Description
WD	Write data or add file
AD	Append data or add subdirectory
WA	Write attributes
WEA	Write extended attributes
DE	Delete
WDAC	Write DAC (change permissions)
WO	Write owner (take ownership)

3.3 Remote Virtual Machine Setup

3.3.1 Specifications

All codes are executed on a Google Cloud compute engine with a 64-bit 8-vCPU 4-core CPU, 64 GB RAM and 250 GB SSD persistent disk running on Ubuntu Server 24.04 LTS. The instance locates in region `us-central1` (Iowa) and zone `us-central1-f`. The standard provisioning model, although noticeably more high-priced than the spot counterpart, is chosen to prevent VM preemption primarily because the proposed classifier has exponential time complexity, thereby requiring exceptionally high CPU utilization. The network traffic is routed in a premium tier to provide low latency. A static external IPv4 address is reserved and assigned to the instance for remote connection.

3.3.2 SSH Key-Based Authentication

Password authentication should be disabled by uncommenting the following line in the SSH configuration file `/etc/ssh/sshd_config`.

```
PasswordAuthentication no
```

SSH authentication requires adding a public key of a local computer to the key file `~/.ssh/authorized_keys`.

```
echo <public_keyfile> >> ~/.ssh/authorized_keys
```

3.3.3 Python Installation

Ubuntu Server 24.04 LTS is equipped with outdated Python 3.12.3. The installation of latest Python 3.13.0 at the current stage inevitably requires building from source. As opposed to Python 3.12, Python 3.13 experimentally supports multithreading without global interpreter lock (GIL). However, disabling GIL prevents the successful installation of `scikit-learn` package which is required to build a decision tree in Chapter 5. In this circumstance, the binary distribution, commonly known as wheel, of `scikit-learn` is unavailable. Its compilation by Rust and Cargo with the build system requirements specified in `pyproject.toml` also fails. Therefore, GIL remains in this dissertation as a default mechanism of mutual exclusion lock.

3.3.3.1 Introduction to Compilation in C

All Python source codes are written in C, and they require a C compiler such as GNU Compiler Collection (GCC) and Clang/Low Level Virtual Machine (LLVM). This dissertation chooses the first compiler. GCC 13 can be installed by using the Advanced Package Tool (APT), an interface to a packaging system on Debian and its derivatives such as Ubuntu.

```
sudo apt install build-essential
```

A newer version of GCC, currently GCC 14 release and GCC 15 experimental, can optionally be built from source by its previous version. The C/C++ compiler commands, including versions, and flags can be added to the environment variables `CC`, `CXX`, `CFLAGS` and `CXXFLAGS` respectively.

GNU Make is used as a build automation tool by reading instructions from `Makefile`. Parallelism is supported by utilizing multiple CPU threads with the `-j` or `--jobs` flag.

```
make -j<N>
make -j<N> install
```

The parameter `<N>` is the maximum allowable number of jobs executed in parallel which should not exceed the number of available CPU threads.

3.3.3.2 Basic Object Types

Python object structures are declared in the header file `Include/object.h`. A Python object is stored in memory, it has a C structure named `_object`, and it can be referenced as a `PyObject*` pointer. With GIL enabled by default, it declares a reference counter `ob_refcnt` of type `Py_ssize_t` and a pointer to the object type `*ob_type` of type `PyTypeObject`. When GIL is disabled by configuring Python with the `--disable-gil` option, a local reference counter is declared by `ob_ref_local` of type `uint32_t` is only adjusted by an owner thread, whereas a shared counterpart `ob_ref_shared` of type `Py_ssize_t` is adjusted by remaining threads. Its actual reference counter can be computed by merging both. When its reference counter is decremented to zero, it is deleted by a garbage collector (GC). If it only has a cyclic reference, a generational garbage collection is employed. A variable-size Python object can be cast further to `PyVarObject*` with an additional field `ob_size` of type `Py_ssize_t` which holds the number of its items.

```

#ifndef Py_GIL_DISABLED
struct _object {
    #if (defined(__GNUC__) || defined(__clang__)) \
    && !(defined __STDC_VERSION__ && __STDC_VERSION__ >= 201112L)
    // On C99 and older, anonymous union is a GCC and clang extension
    __extension__
    #endif

    #ifdef _MSC_VER
    // Ignore MSC warning C4201: "nonstandard extension used:
    // nameless struct/union"
    __pragma(warning(push))
    __pragma(warning(disable: 4201))
    #endif

    union {
        Py_ssize_t ob_refcnt;
        #if SIZEOF_VOID_P > 4
        PY_UINT32_T ob_refcnt_split[2];
        #endif
    };

    #ifdef _MSC_VER
    __pragma(warning(pop))
    #endif

    PyTypeObject *ob_type;
};

#else
// Objects that are not owned by any thread use a thread id (tid) of
// zero.
// This includes both immortal objects and objects whose reference
// count
// fields have been merged.
#define _Py_UNOWNED_TID 0

// The shared reference count uses the two least-significant bits to
// store
// flags. The remaining bits are used to store the reference count.

```

```

#define _Py_REF_SHARED_SHIFT 2
#define _Py_REF_SHARED_FLAG_MASK 0x3

// The shared flags are initialized to zero.
#define _Py_REF_SHARED_INIT 0x0
#define _Py_REF_MAYBE_WEAKREF 0x1
#define _Py_REF_QUEUED 0x2
#define _Py_REF_MERGED 0x3

// Create a shared field from a refcnt and desired flags
#define _Py_REF_SHARED(refcnt, flags) (((refcnt) <<
    _Py_REF_SHARED_SHIFT) + (flags))

struct _object {
    // ob_tid stores the thread id (or zero). It is also used by the
    // GC and the
    // trashcan mechanism as a linked list pointer and by the GC to
    // store the
    // computed "gc_refs" refcount.
    uintptr_t ob_tid;
    uint16_t _padding;
    PyMutex ob_mutex; // per-object lock
    uint8_t ob_gc_bits; // gc-related state
    uint32_t ob_ref_local; // local reference count
    Py_ssize_t ob_ref_shared; // shared (atomic) reference count
    PyTypeObject *ob_type;
};

#ifdef
/* Cast argument to PyObject* type. */
#define _PyObject_CAST(op) _Py_CAST(PyObject*, (op))

typedef struct {
    PyObject ob_base;
    Py_ssize_t ob_size; /* Number of items in variable part */
} PyVarObject;

```

3.3.3.3 String Interning

Python interns strings, which are immutable objects, of the same value mainly through the function `_PyUnicode_InternInPlace()` defined in the source file `Objects/unicodeobject.c` by retaining only one copy in memory. This reduces memory usage and speeds up certain operations, for example equality comparison. The reference to all interned strings is stored in the per-interpreter dictionary `interned` initialized during the first invocation. As opposed to a release build, a debug build denies with an assertion the addition of a process-global interned string into the existing dictionary to prevent the possibility of getting a duplicate.

```
static /* non-null */ PyObject*
intern_static(PyInterpreterState *interp, PyObject *s /* stolen */)
{
    // Note that this steals a reference to `s`, but in many cases
    // that
    // stolen ref is returned, requiring no decref/incref.

    assert(s != NULL);
    assert(_PyUnicode_CHECK(s));
    assert(_PyUnicode_STATE(s).statically_allocated);
    assert(!_PyUnicode_CHECK_INTERNED(s));

#ifdef Py_DEBUG
    /* We must not add process-global interned string if there's
       already a
       * per-interpreter interned_dict, which might contain duplicates.
       */
    PyObject *interned = get_interned_dict(interp);
    assert(interned == NULL);
#endif

    /* Look in the global cache first. */
    PyObject *r = (PyObject *)_Py_hashtable_get(INTERNED_STRINGS, s);
    /* We should only init each string once */
    assert(r == NULL);
}
```

```

/* but just in case (for the non-debug build), handle this */
if (r != NULL && r != s) {
    assert(_PyUnicode_STATE(r).interned ==
           SSTATE_INTERNED_IMMORTAL_STATIC);
    assert(_PyUnicode_CHECK(r));
    Py_DECREF(s);
    return Py_NewRef(r);
}

if (_Py_hashtable_set(INTERNED_STRINGS, s, s) < -1) {
    Py_FatalError("failed to intern static string");
}

_PyUnicode_STATE(s).interned = SSTATE_INTERNED_IMMORTAL_STATIC;
return s;
}

```

Soon after Python 3.13.0 had been released, JupyterLab could not be launched in the debug build despite its successful installation. This problem can be fixed by commenting the following assert statement, though discouraged, and rebuilding the Python.

```
//assert(interned == NULL);
```

This can also be done by using the `sed` command.

```

sed -i -e \
    's/assert(interned == NULL);/\n\nassert(interned == NULL);/g' \
    Objects/unicodeobject.c

```

However, the source code modification is not required for running the latest JupyterLab.

3.3.3.4 Configuration and Build

It is recommended to have three separate directories: source, build and install. In this dissertation, Python is built against OpenSSL whose runtime library directory `rpath` is automatically detected, and it respects the OpenSSL crypto policy `openssl.cnf` by overriding the default Python cipher list.


```

--with-openssl=<openssl_rootdir>
--with-openssl-rpath=auto
--with-ssl-default-suites=openssl

```

As opposed to the built-in Python, a static library (with `.a` extension) is built from source by default. This dissertation builds a dynamic library (with `.so` extension) by adding the `--enable-shared` flag to minimize disk footprint of several programs because Python 3.13.0 will intentionally be built as a new primary version, but inside a home directory. It is entirely separate from the latest system Python library, shared by multiple native applications, `/usr/lib/python3.12/config-3.12-x86_64-linux-gnu/libpython3.12.so` which currently points to another symbolic link `/usr/lib/x86_64-linux-gnu/libpython3.12.so.1` and finally to the actual shared library `/usr/lib/x86_64-linux-gnu/libpython3.12.so.1.0`, of which all interfaces remain unchanged (interface version 1) and the library source code is unmodified (revision 0).

Although a release build, default in Python, is more optimized but harder to debug, this dissertation chooses the Python debug build by passing the `--with-pydebug` flag. The source codes are compiled to intermediate object codes in an attempt to reduce the code size and execution time. A linker produces shared libraries and executables from objects without duplicate definitions. Both compilation and linking are optimized by turning on the `--enable-optimizations` and `--with-lto` flags. C assertions are enabled in debug mode by default. Python can be compiled with profiling turned on by using the `--enable-profiling` flag. The GNU profiler `gprof` collects data during Python execution and outputs the file `gmon.out` in a current working directory. Based on this information, the code performance can be analyzed in terms of execution time and memory consumption, and its bottleneck is identifiable. Nonetheless, this dissertation omits the profiling flag.

Python optimization, if specified, is profile-guided (PGO) based on collected data from sequential test runs. For the PGO generation task, Python by default uses the following arguments assigned to the environment variable `PROFILE_TASK`.

```

-m test --pgo --timeout=

```

The `-m` flag searches for all files matching a given pattern, in this case `test*` in the `Lib/test` subdirectory. The `--pgo` flag enables PGO training and selects 44 out of 478 test runs. Python 3.13 sets no timeout for an individual test, in contrast to Python 3.12 a default timeout of 20 minutes, and no longer ignores a test failure. Its build time is partly impacted by these test runs and can significantly improve by ignoring through the `-i` flag time-consuming tests which can be detected, for instance, by setting a custom timeout. This dissertation excludes the test for embedding APIs located at `Lib/test/test_embedded.py` and sets a timeout of 5 minutes.

```
export PROFILE_TASK="-m test --pgo --timeout=300 -i test_embedded"
```

No timeout error is raised, and all remaining 43 tests pass.

Furthermore, the `pyexpat` module can be built using an installed `expat` library by the `--with-system-expat` flag. `DTrace`, `Valgrind` and loadable extensions in the `_sqlite` extension module are supported by the `--with-dtrace`, `--with-valgrind` and `--enable-loadable-sqlite-extensions` flags. Address sanitizer (ASAN) and memory sanitizer (MSAN) are disabled by default. Certain flags requires additional dependencies. Their environment variables for C compiler and linker flags, required libraries, Python modules to be optionally built, and corresponding APT packages are given in Table 3.2.

Table 3.2: Python options for third-party dependencies

Environment Variables	Library	Module	APT Package
BZIP2_[LIBS CFLAGS]	libbz2	bz2	libbz2-dev
CURSES_[LIBS CFLAGS]	libncurses	curses	libncurses-dev
GDBM_[LIBS CFLAGS]	gdbm		libgdbm-compat-dev
LIBB2_[LIBS CFLAGS]	libb2	hashlib	libb2-dev
LIBEDIT_[LIBS CFLAGS]	libedit	readline	libreadline-dev
LIBFFI_[LIBS CFLAGS]	libffi	ctypes	libffi-dev
LIBMPDEC_[LIBS CFLAGS]	libmpdec	decimal	
LIBLZMA_[LIBS CFLAGS]	liblzma	lzma	liblzma-dev
LIBREADLINE_[LIBS CFLAGS]	libreadline	readline	libreadline-dev
LIBSQLITE3_[LIBS CFLAGS]	libsqlite3	sqlite3	libsqlite3-dev
LIBUUID_[LIBS CFLAGS]	libuuid	uuid	uuid-dev
PANEL_[LIBS CFLAGS]	libpanel	curses.panel	libpanel-dev
TCLTK_[LIBS CFLAGS]	TCLTK		tk-dev

Table 3.2: Python options for third-party dependencies (continued)

Environment Variables	Library	Module	APT Package
ZLIB_[LIBS CFLAGS]	libzlib	gzip	zlib1g-dev

After Python is completely installed in the destination directory, both source and build directories can be removed. The `bin` directory should be added to the `PATH` so that the executables are accessible from any location. The system environment variables `LD_LIBRARY_PATH` and `LDFLAGS` should include the `lib` directory so that the library code can be loaded into memory at runtime and compile time respectively. The recently built version must precede the system-wide version.

```
export PATH="<install_dir>/bin:$PATH"
export LD_LIBRARY_PATH="<install_dir>/lib:${LD_LIBRARY_PATH}"
export LDFLAGS="-L<install_dir>/lib $LDFLAGS"
```

This migration should be made to the Bash configuration file `~/.bashrc`. Deprecation warnings may be emitted during runtime, but they can be suppressed by setting the Python environment variable `PYTHONWARNINGS`.

```
export PYTHONWARNINGS="ignore::DeprecationWarning"
```

The changes are not applied until the configuration file is reread.

```
source ~/.bashrc
```

3.3.4 Backup to OCI Object Storage

3.3.4.1 Introduction to OCI

Oracle Cloud Infrastructure (OCI) basically has two logical concepts of organization management: *tenancy* and *compartment*. A *tenancy* is a root container for administering cloud resources. During the signup process, a parent tenancy is provisioned and tied to a specified, unchangeable home region which is `ap-singapore-1` in this dissertation. Multiple child tenancies can be created and managed by the parent tenancy. A *compartment* belongs to a tenancy, controls access to cloud resources, supports up to six levels, and brings clearer separation. It must be specified when a resource is created. A tenancy can be considered as a root compartment.

The OCI command line interface (CLI) can be installed by the `oci-cli` package in an isolated Python environment to prevent dependency conflicts. The `source` command is used to activate this environment. After the installation finishes, the executables including `oci` and its libraries are in the `bin` and `lib` directories. Only the first is additionally added to the `PATH` so that the `oci` command can be executed in the global environment, not limited to the virtual counterpart.

```
~$ python3 -m venv <env_dir>
~$ source <env_dir>/bin/activate
(env_dir)$ pip3 install oci-cli
(env_dir)$ deactivate
```

Before accessing an OCI resource or service, a basic OCI configuration must be made in an interactive mode from a terminal, for instance.

```
oci setup config
```

This can also be done from a custom configuration file by setting the environment variable `OCI_CLI_RC_FILE` to its full path. The file has two main components: section and key. A section except the default should be specified via the `--profile` option in the CLI.

```
[DEFAULT]
user=<user>
fingerprint=<fingerprint>
key_file=<key_file>
tenancy=<tenancy>
region=ap-singapore-1
```

3.3.4.2 OCI Object Storage

An Object Storage *namespace* serves as the top-level container for all buckets and objects, it is unique to a tenant, and it spans all compartments within a region. Although region-specific, its name remains the same across all regions. An *object* is any type of data along with its metadata stored in a logical container called *bucket* unique in a namespace. Object Storage is highly scalable, cost-effective and structurally flat, compared to block and file storage. There are two default tiers. A *standard tier* has a higher cost and no retention period. In a low-cost *archive tier*, an object must be retained for at least 90 days, and restoration takes very long time to retrieve all data bytes. OCI Object Storage supports auto-tiering, object versioning and multipart uploading which is greatly resilient for a very large object. Uncommitted or failed multipart uploads can be cleaned either manually or through a predefined lifecycle policy rule.

In this dissertation, only a full backup of scripts and results, not only due to its small size but also to avoid the possibility of a corrupted incremental or differential backup, is stored in OCI Object Storage. A total of 20 GB in all tenancies is always free, and no upgrade to a paid account is required. A bucket is created without auto-tiering and versioning. All buckets in a compartment can be listed along with their namespace.

```
oci os bucket list -c <compartment_id>
```

A backup is performed by a one-way synchronization, and each version is uniquely identified by an object prefix such as a timestamp. An object that exists in a destination but not in a source is deleted.

```
oci os object sync -ns <namespace> -bn <bucket> \
  --prefix <obj_prefix> --src-dir <src_dir> --delete
```

Furthermore, an object can be renamed and deleted where bulk deletion is also permitted.

```
oci os object rename -ns <namespace> -bn <bucket> \
  --name <obj_name> --new-name <obj_new_name>
oci os object delete -ns <namespace> -bn <bucket> \
  --name <obj_name>
oci os object bulk-delete -ns <namespace> -bn <bucket> \
  --prefix <obj_prefix>
```

3.4 GitHub Repository

The template GitHub repository for this dissertation is available at <https://github.com/songkomkrit/phd-template>. The basic Git commands are included in Table 3.3. The path to the Git global configuration file `.gitconfig` specific to a user is given by the environment variable `GIT_CONFIG_GLOBAL`. The username and the email address can be set up either by the `git config` command with the `--global` option or by editing the configuration file.

```
git config --global user.name <username>
git config --global user.email <email_address>
```

The following settings should appear in the file.

```
[user]
    name = <username>
    email = <email_address>
```

Table 3.3: Basic Git commands

Command	Description
<code>git clone</code>	Clean copy
<code>git pull</code>	Update with local changes kept
<code>git reset --hard</code>	Update with local changes discarded
<code>git clean -fdx</code>	Clean with untracked files and directories removed
<code>git push</code>	Remote update with local commits

The JSON-format metadata of both independent and dependent variables are at `Data/Original/metadata/meta-indep.json` and `Data/Original/metadata/meta-dep.json`. The health insurance in SAS7BDAT format is omitted, but its feather file of smaller size is already included in the directory `Data/Original/feature`. This dissertation further limits the number of participants and features to smaller size before fed to a classification model. Since data sampling is random, the sample is put in the directory `Samples/cplex`.

The box classifier proposed in Chapter 4 is located in the CPLEX Optimization Programming Language (OPL) project `Projects/box` where its `input` subdirectory contains a sample data including additional information and its `output` counterpart all relevant results such as splitting values and predicted class label per decision box. The model can be executed by the `oplrun` command and logged into file and on console by the `tee` command.

```
oplrun -p <project_dir> 2>&1 | tee <log_file>
```

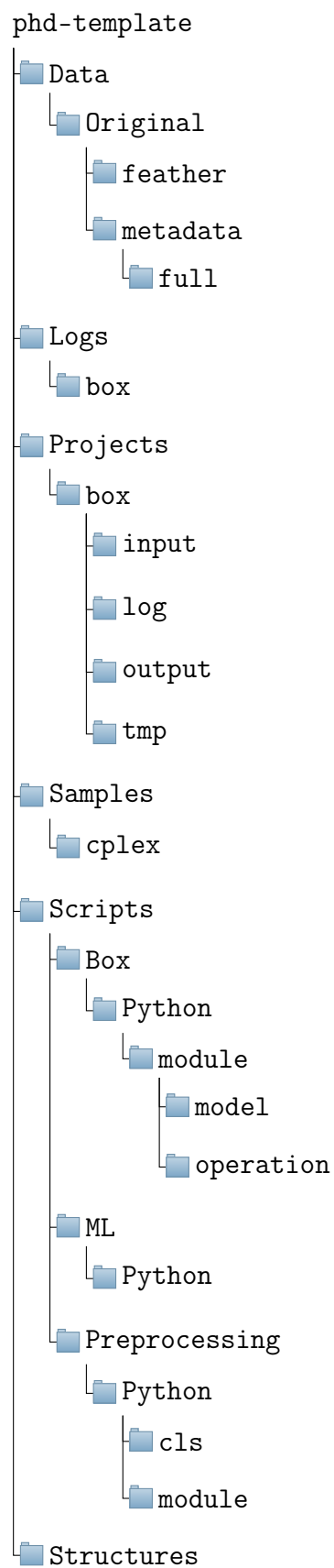
The `<project_dir>` is `Projects/box`. Thanks to its comparative low-resource consumption, using the `oplrun` executable in a terminal is preferred to starting the CPLEX Studio IDE by executing the `oplide` command. The manual backup of the CPLEX engine log is stored in the directory `Logs/box`. The Python scripts for data preprocessing, decision tree building and decision box merging can be found in `Scripts/Preprocessing/Python`, `Scripts/ML/Python` and `Scripts/Box/Python` respectively. The directory and file tree structures can be printed in terminal by using the `tree` command, and they are saved to `Structures/directory.txt` and `Structures/file.txt`.

```
tree -d . > Structures/directory.txt
tree -f . > Structures/file.txt
```

There are currently 29 directories and 60 files. The directory structure is displayed in Figure 3.1.

The template repository is very minimal with merely output files generated by a CPLEX optimizer. Its main purpose is to allow users to generate a new repository with the same structure before further Python execution such as exploratory data analysis (EDA). The up-to-date repository based on the template with additional outputs included is available at <https://github.com/songkomkrit/phd>.

Figure 3.1: Directory tree structure of the template GitHub repository



3.5 Health Insurance Dataset

3.5.1 Background

The 2020 U.S. Census Bureau’s Current Population Survey (CPS) Annual Social and Economic Supplement (ASEC) dataset will be used in the dissertation. Questions were asked for the information on a previous calendar year. Therefore, the person-level dataset provides the estimates of individual health insurance coverage for calendar year 2019.

An individual may simultaneously have different coverages. Private health insurance includes an employment-based plan and a direct-purchase plan. Public health insurance comprises Medicare, means-tested coverage (i.e., Medicaid, Peace Church Health Insurance or PCHIP and others), military healthcare (i.e., TRICARE formerly known as Civilian Health and Medical Program of the Uniformed Services or CHAMPUS, Civilian Health and Medical Program of the Department of Veterans Affairs or CHAMPVA and Veterans Affairs or VA) and the combination of Indian Health Service (IHS) and other coverages. Those who only have IHS are considered uninsured.

Since there are in total 10 subtypes of insurance coverage, quantitative data analysis may involve up to $2^{10} + 1 = 1,025$ possible classes. In fact, the maximum number of subtypes of an overall class can be determined by the total sum of the indicator variables of the first ten subtypes. Furthermore, the dataset has at least 150,000 records and 750 attributes which are mostly measured on nominal scales. In addition to their allocation and topcode flags, the dataset variables cover a broad spectrum of characteristics: demographics, work experience, income (i.e., earnings, other income, non-cash benefits and tax), poverty, health insurance (i.e., government, private, employment-based, direct-purchase, subsidized marketplace, unsubsidized marketplace, non-marketplace, Medicaid, other means-tested, PCHIP, Medicare, IHS, TRICARE, CHAMPVA, VA and employer-sponsored), health status and migration. They also include basic CPS items (i.e., labor force and earnings) and medical out-of-pocket (OOP) expenditures.

3.5.2 Scope of Study

Within existing conceptual frameworks, certain independent variables will be preselected in the dissertation before further investigation. A group of infant born after the calendar year is excluded in the analysis. The combination of three following coverages is merely considered: employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB). There are eight possible binary tuples (GRP, DIR, PUB) which are regrouped into five following classes in Table 3.4.

Table 3.4: Class codes of insurance coverage combination

Class	Code	Combination of insurance coverages		
		GRP	DIR	PUB
0	NNN	No	No	No
1	NNY	No	No	Yes
2	NY_	No	Yes	Yes
		No	Yes	No
3	YNN	Yes	No	No
4	Y1Y	Yes	No	Yes
		Yes	Yes	Yes
		Yes	Yes	No

3.5.3 Metadata

Metadata 3.1 and 3.2 contain related information on dependent and independent variables in JSON format with a variable symbol as a main key and all of the following as its informative value in dictionary format: label, universe, type (either continuous or categorical), topic, subtopic and possible values including NIU (not in universe).

Metadata 3.1: Dependent variables (data/original/metadata/meta-dep.json)

```

1  {
2    "NOW_COV": {
3      "label": "Currently covered by health insurance coverage",
4      "universe": "All Persons",
5      "type": "Categorical",
6      "role": "Dependent",
7      "topic": "Health insurance",
8      "subtopic": "Any health insurance coverage",
9      "values": {
10       "1": "Yes",
11       "2": "No"
12     }
13   },
14   "NOW_PUB": {
15     "label": "Current public coverage",
16     "universe": "All Persons",
17     "type": "Categorical",
18     "role": "Dependent",
19     "topic": "Health insurance",
20     "subtopic": "Public coverage",
21     "values": {
22       "1": "Yes",
23       "2": "No"
24     }
25   },
26   "NOW_PRIV": {
27     "label": "Current private coverage",
28     "universe": "All Persons",
29     "type": "Categorical",
30     "role": "Dependent",
31     "topic": "Health insurance",
32     "subtopic": "Private coverage",
33     "values": {
34       "1": "Yes",

```

```

35         "2": "No"
36     }
37 },
38 "NOW_GRP": {
39     "label": "Any current employment-based coverage",
40     "universe": "All Persons",
41     "type": "Categorical",
42     "role": "Dependent",
43     "topic": "Health insurance",
44     "subtopic": "Employment-based coverage",
45     "values": {
46         "1": "Yes",
47         "2": "No"
48     }
49 },
50 "NOW_DIR": {
51     "label": "Any current direct-purchase coverage",
52     "universe": "All Persons",
53     "type": "Categorical",
54     "role": "Dependent",
55     "topic": "Health insurance",
56     "subtopic": "Direct-purchase coverage",
57     "values": {
58         "1": "Yes",
59         "2": "No"
60     }
61 },
62 "NOW_MCARE": {
63     "label": "Current Medicare coverage",
64     "universe": "All Persons",
65     "type": "Categorical",
66     "role": "Dependent",
67     "topic": "Health insurance",
68     "subtopic": "Medicare coverage",
69     "values": {
70         "1": "Yes",

```

```

71         "2": "No"
72     }
73 },
74 "NOW_MCAID": {
75     "label": "Current Medicaid, PCHIP, or other means-tested coverage",
76     "universe": "All Persons",
77     "type": "Categorical",
78     "role": "Dependent",
79     "topic": "Health insurance",
80     "subtopic": "Medicaid or other means-tested coverage",
81     "values": {
82         "1": "Yes",
83         "2": "No"
84     }
85 },
86 "NOW_CAID": {
87     "label": "Current Medicaid coverage",
88     "universe": "All Persons",
89     "type": "Categorical",
90     "role": "Dependent",
91     "topic": "Health insurance",
92     "subtopic": "Medicaid coverage",
93     "values": {
94         "1": "Yes",
95         "2": "No"
96     }
97 },
98 "NOW_PCHIP": {
99     "label": "Current PCHIP coverage",
100    "universe": "All Persons",
101    "type": "Categorical",
102    "role": "Dependent",
103    "topic": "Health insurance",
104    "subtopic": "PCHIP coverage",
105    "values": {
106        "1": "Yes",

```

```

107         "2": "No"
108     }
109 },
110 "NOW_OTHMT": {
111     "label": "Current other means-tested coverage",
112     "universe": "All Persons",
113     "type": "Categorical",
114     "role": "Dependent",
115     "topic": "Health insurance",
116     "subtopic": "Other means-tested coverage",
117     "values": {
118         "1": "Yes",
119         "2": "No"
120     }
121 },
122 "NOW_MIL": {
123     "label": "Any current TRICARE coverage",
124     "universe": "All Persons",
125     "type": "Categorical",
126     "role": "Dependent",
127     "topic": "Health insurance",
128     "subtopic": "TRICARE coverage",
129     "values": {
130         "1": "Yes",
131         "2": "No"
132     }
133 },
134 "NOW_CHAMPVA": {
135     "label": "Current CHAMPVA coverage",
136     "universe": "All Persons",
137     "type": "Categorical",
138     "role": "Dependent",
139     "topic": "Health insurance",
140     "subtopic": "CHAMPVA coverage",
141     "values": {
142         "1": "Yes",

```

```

143         "2": "No"
144     }
145 },
146 "NOW_VACARE": {
147     "label": "Current VACARE coverage",
148     "universe": "All Persons",
149     "type": "Categorical",
150     "role": "Dependent",
151     "topic": "Health insurance",
152     "subtopic": "VACARE coverage",
153     "values": {
154         "1": "Yes",
155         "2": "No"
156     }
157 },
158 "NOW_IHSFLG": {
159     "label": "Current coverage through the Indian Health Service",
160     "universe": "All Persons",
161     "type": "Categorical",
162     "role": "Dependent",
163     "topic": "Health insurance",
164     "subtopic": "Indian Health Service coverage",
165     "values": {
166         "1": "Yes",
167         "2": "No"
168     }
169 }
170 }

```

Metadata 3.2: Independent variables (data/original/metadata/meta-indep.json)

```

1  {
2    "A_AGE": {
3      "label": "Age",
4      "universe": "All Persons",
5      "type": "Continuous",
6      "role": "Independent",
7      "topic": "Demographics",
8      "subtopic": "Individual characteristics",
9      "values": {
10       "00-79": "0-79 years of age",
11       "80": "80-84 years of age",
12       "85": "85+ years of age"
13     }
14   },
15   "A_EXPRRP": {
16     "label": "Expanded relationship code",
17     "universe": "All Persons",
18     "type": "Categorical",
19     "role": "Independent",
20     "topic": "Demographics",
21     "subtopic": "Individual characteristics",
22     "values": {
23       "1": "Reference person with relatives",
24       "2": "Reference person without relatives",
25       "3": "Husband",
26       "4": "Wife",
27       "5": "Own child",
28       "7": "Grandchild",
29       "8": "Parent",
30       "9": "Brother/sister",
31       "10": "Other relative",
32       "11": "Foster child",
33       "12": "Nonrelative with relatives",
34       "13": "Partner/roommate",

```



```

35         "14": "Nonrelative without relatives"
36     }
37 },
38 "A_FAMTYP": {
39     "label": "Family type",
40     "universe": "All Persons",
41     "type": "Categorical",
42     "role": "Independent",
43     "topic": "Demographics",
44     "subtopic": "Individual characteristics",
45     "values": {
46         "1": "Primary family",
47         "2": "Nonfamily householder",
48         "3": "Related subfamily",
49         "4": "Unrelated subfamily",
50         "5": "Secondary individual"
51     }
52 },
53 "A_HGA": {
54     "label": "Educational attainment",
55     "universe": "All Persons",
56     "type": "Categorical",
57     "role": "Independent",
58     "topic": "Demographics",
59     "subtopic": "Individual characteristics",
60     "values": {
61         "0": "Children",
62         "31": "Less than 1st grade",
63         "32": "1st,2nd,3rd,or 4th grade",
64         "33": "5th or 6th grade",
65         "34": "7th and 8th grade",
66         "35": "9th grade",
67         "36": "10th grade",
68         "37": "11th grade",
69         "38": "12th grade no diploma",

```

```

70         "39": "High school graduate - high school diploma or equivalent
71             ",
72         "40": "Some college but no degree",
73         "41": "Associate degree in college - occupation/vocation
74             program",
75         "42": "Associate degree in college - academic program",
76         "43": "Bachelor's degree (for example: BA,AB,BS)",
77         "44": "Master's degree (for example: MA,MS,MENG,MED,MSW, MBA)",
78         "45": "Professional school degree (for example: MD,DDS,DVM,LLB,
79             JD)",
80         "46": "Doctorate degree (for example: PHD,EDD)"
81     }
82 },
83 "A_MARITL": {
84     "label": "Marital status",
85     "universe": "All Persons",
86     "type": "Categorical",
87     "role": "Independent",
88     "topic": "Demographics",
89     "subtopic": "Individual characteristics",
90     "values": {
91         "1": "Married - civilian spouse present",
92         "2": "Married - AF spouse present",
93         "3": "Married - spouse absent (exc.separated)",
94         "4": "Widowed",
95         "5": "Divorced",
96         "6": "Separated",
97         "7": "Never married"
98     }
99 },
100 "A_PFREL": {
101     "label": "Primary family relationship",
102     "universe": "All Persons",
103     "type": "Categorical",
104     "role": "Independent",
105     "topic": "Demographics",

```

```

103     "subtopic": "Individual characteristics",
104     "values": {
105         "0": "Not in primary family",
106         "1": "Husband",
107         "2": "Wife",
108         "3": "Own child",
109         "4": "Other relative",
110         "5": "Unmarried reference person"
111     }
112 },
113 "A_SEX": {
114     "label": "Sex",
115     "universe": "All Persons",
116     "type": "Categorical",
117     "role": "Independent",
118     "topic": "Demographics",
119     "subtopic": "Individual characteristics",
120     "values": {
121         "1": "Male",
122         "2": "Female"
123     }
124 },
125 "P_STAT": {
126     "label": "Status of person identifier",
127     "universe": "All Persons",
128     "type": "Categorical",
129     "role": "Independent",
130     "topic": "Demographics",
131     "subtopic": "Individual characteristics",
132     "values": {
133         "1": "Civilian 15+",
134         "2": "Armed forces",
135         "3": "Children 0-14"
136     }
137 },
138 "PEAFEVER": {

```

```

139     "label": "Did you ever serve on active duty in the U.S. Armed
        Forces?",
140     "universe": "A_AGE greater than or equal to 17",
141     "type": "Categorical",
142     "role": "Independent",
143     "topic": "Demographics",
144     "subtopic": "Individual characteristics",
145     "values": {
146         "-1": "Not in universe",
147         "1": "Yes",
148         "2": "No"
149     }
150 },
151 "PEDISDRS": {
152     "label": "Does...have difficulty dressing or bathing?",
153     "universe": "PRPERTYP = 2",
154     "type": "Categorical",
155     "role": "Independent",
156     "topic": "Demographics",
157     "subtopic": "Individual characteristics",
158     "values": {
159         "-1": "Not in universe",
160         "1": "Yes",
161         "2": "No"
162     }
163 },
164 "PEDISEAR": {
165     "label": "Is...deaf or does ...have serious difficulty hearing?",
166     "universe": "PRPERTYP = 2",
167     "type": "Categorical",
168     "role": "Independent",
169     "topic": "Demographics",
170     "subtopic": "Individual characteristics",
171     "values": {
172         "-1": "Not in universe",
173         "1": "Yes",

```

```

174         "2": "No"
175     }
176 },
177 "PEDISEYE": {
178     "label": "Is...blind or does...have serious difficulty seeing even
179         when wearing glasses?",
180     "universe": "PRPERTYP = 2",
181     "type": "Categorical",
182     "role": "Independent",
183     "topic": "Demographics",
184     "subtopic": "Individual characteristics",
185     "values": {
186         "-1": "Not in universe",
187         "1": "Yes",
188         "2": "No"
189     }
190 },
191 "PEDISOUT": {
192     "label": "Because of a physical, mental, or emotional condition,
193         does...have difficulty doing errands along such as visiting a
194         doctor's office or shopping?",
195     "universe": "PRPERTYP = 2",
196     "type": "Categorical",
197     "role": "Independent",
198     "topic": "Demographics",
199     "subtopic": "Individual characteristics",
200     "values": {
201         "-1": "Not in universe",
202         "1": "Yes",
203         "2": "No"
204     }
205 },
206 "PEDISPHY": {
207     "label": "Does...have serious difficulty Walking or climbing stairs
208         ?",
209     "universe": "PRPERTYP = 2",

```

```

206     "type": "Categorical",
207     "role": "Independent",
208     "topic": "Demographics",
209     "subtopic": "Individual characteristics",
210     "values": {
211         "-1": "Not in universe",
212         "1": "Yes",
213         "2": "No"
214     }
215 },
216 "PREDISREM": {
217     "label": "Because of a physical, mental, or emotional condition,
                does...have serious difficulty concentrating, remembering, or
                making decisions?",
218     "universe": "PRPERTYP = 2",
219     "type": "Categorical",
220     "role": "Independent",
221     "topic": "Demographics",
222     "subtopic": "Individual characteristics",
223     "values": {
224         "-1": "Not in universe",
225         "1": "Yes",
226         "2": "No"
227     }
228 },
229 "PRDISFLG": {
230     "label": "Does this person have any of these disability conditions?
                ",
231     "universe": "PRPERTYP = 2",
232     "type": "Categorical",
233     "role": "Independent",
234     "topic": "Demographics",
235     "subtopic": "Individual characteristics",
236     "values": {
237         "-1": "Not in universe",
238         "1": "Yes",

```

```

239         "2": "No"
240     }
241 },
242 "PRCITSHP": {
243     "label": "Citizenship group",
244     "universe": "All persons",
245     "type": "Categorical",
246     "role": "Independent",
247     "topic": "Demographics",
248     "subtopic": "Individual characteristics",
249     "values": {
250         "1": "Native, born in US",
251         "2": "Native, born in PR or US outlying area",
252         "3": "Native, born abroad of US parent(s)",
253         "4": "Foreign born, US cit by naturalization",
254         "5": "Foreign born, not a US citizen"
255     }
256 },
257 "PRDTRACE": {
258     "label": "Race",
259     "universe": "All persons",
260     "type": "Categorical",
261     "role": "Independent",
262     "topic": "Demographics",
263     "subtopic": "Individual characteristics",
264     "values": {
265         "1": "White only",
266         "2": "Black only",
267         "3": "American Indian, Alaskan Native only (AI)",
268         "4": "Asian only",
269         "5": "Hawaiian/Pacific Islander only (HP)",
270         "6": "White-Black",
271         "7": "White-AI",
272         "8": "White-Asian",
273         "9": "White-HP",
274         "10": "Black-AI",

```

```

275         "11": "Black-Asian",
276         "12": "Black-HP",
277         "13": "AI-Asian",
278         "14": "AI-HP",
279         "15": "Asian-HP",
280         "16": "White-Black-AI",
281         "17": "White-Black-Asian",
282         "18": "White-Black-HP",
283         "19": "White-AI-Asian",
284         "20": "White-AI-HP",
285         "21": "White-Asian-HP",
286         "22": "Black-AI-Asian",
287         "23": "White-Black-AI-Asian",
288         "24": "White-AI-Asian-HP",
289         "25": "Other 3 race comb.",
290         "26": "Other 4 or 5 race comb."
291     }
292 },
293 "A_MJIND": {
294     "label": "Major industry code",
295     "universe": "A_CLSWKR = 1-7",
296     "type": "Categorical",
297     "role": "Independent",
298     "topic": "Basic CPS items",
299     "subtopic": "Edited labor force items",
300     "values": {
301         "0": "Not in universe, or children",
302         "1": "Agriculture, forestry,fishing, and hunting",
303         "2": "Mining",
304         "3": "Construction",
305         "4": "Manufacturing",
306         "5": "Wholesale and retail trade",
307         "6": "Transportation and utilities",
308         "7": "Information",
309         "8": "Financial activities",
310         "9": "Professional and business services",

```



```

311         "10": "Educational and health services",
312         "11": "Leisure and hospitality",
313         "12": "Other services",
314         "13": "Public administration",
315         "14": "Armed forces"
316     }
317 },
318 "A_MJOCC": {
319     "label": "Major occupation recode",
320     "universe": "A_CLSWKR = 1-7",
321     "type": "Categorical",
322     "role": "Independent",
323     "topic": "Basic CPS items",
324     "subtopic": "Edited labor force items",
325     "values": {
326         "0": "Not in universe or children",
327         "1": "Management, business, and financial occupations",
328         "2": "Professional and related occupations",
329         "3": "Service occupations",
330         "4": "Sales and related occupations",
331         "5": "Office and administrative support occupations",
332         "6": "Farming, fishing, and forestry occupations",
333         "7": "Construction and extraction occupations",
334         "8": "Installation, maintenance, and repair occupations",
335         "9": "Production occupations",
336         "10": "Transportation and material moving occupations",
337         "11": "Armed forces"
338     }
339 },
340 "PEIO1COW": {
341     "label": "Individual class of worker on first job",
342     "universe": "All persons",
343     "type": "Categorical",
344     "role": "Independent",
345     "topic": "Basic CPS items",
346     "subtopic": "Edited labor force items",

```

```

347     "values": {
348         "0": "NIU",
349         "1": "Government-federal",
350         "2": "Government-state",
351         "3": "Government - local",
352         "4": "Private, for profit",
353         "5": "Private, nonprofit",
354         "6": "Self-employed, incorporated",
355         "7": "Self-employed, unincorporated",
356         "8": "Without pay"
357     }
358 },
359 "PRDISC": {
360     "label": "Discouraged worker recode",
361     "universe": "All persons",
362     "type": "Categorical",
363     "role": "Independent",
364     "topic": "Basic CPS items",
365     "subtopic": "Edited labor force items",
366     "values": {
367         "0": "NIU",
368         "1": "Discouraged worker",
369         "2": "Conditionally interested",
370         "3": "Not available"
371     }
372 },
373 "PRUNTYPE": {
374     "label": "Individual class of worker on first job",
375     "universe": "All persons",
376     "type": "Categorical",
377     "role": "Independent",
378     "topic": "Basic CPS items",
379     "subtopic": "Edited labor force items",
380     "values": {
381         "0": "NIU",
382         "1": "Job loser/on layoff",

```

```

383         "2": "Other job loser",
384         "3": "Temporary job ended",
385         "4": "Job leaver",
386         "5": "Re-entrant",
387         "6": "New-entrant"
388     }
389 },
390     "A_GRSWK": {
391         "label": "How much does ... usually earn per week at this job
                 before deductions , subject to topcoding, the higher of either
                 the amount of item 25a times Item 25c or the actual item 25d
                 entry will be present",
392         "universe": "PRERELG=1",
393         "type": "Continuous",
394         "role": "Independent",
395         "topic": "Basic CPS items",
396         "subtopic": "Edited earnings items",
397         "values": {
398             "0": "Not in universe or children or armed forces",
399             "0001-2885": "Dollar amount"
400         }
401     },
402     "A_HRLYWK": {
403         "label": "Is ... paid by the hour on this job?",
404         "universe": "PRERELG=1",
405         "type": "Categorical",
406         "role": "Independent",
407         "topic": "Basic CPS items",
408         "subtopic": "Edited earnings items",
409         "values": {
410             "0": "Not in universe or children and armed forces",
411             "1": "Yes",
412             "2": "No"
413         }
414     },
415     "A_HRSPAY": {

```

```

416     "label": "How much does ... earn per hour?",
417     "universe": "A_HRLYWK=1",
418     "type": "Continuous",
419     "role": "Independent",
420     "topic": "Basic CPS items",
421     "subtopic": "Edited earnings items",
422     "values": {
423         "0": "Not in universe or children or armed forces",
424         "0001-9999": "Entry (2 implied decimal places)"
425     }
426 },
427 "PRERELG": {
428     "label": "Earnings eligibility flag",
429     "universe": "All persons",
430     "type": "Categorical",
431     "role": "Independent",
432     "topic": "Basic CPS items",
433     "subtopic": "Edited earnings items",
434     "values": {
435         "0": "Not earnings eligible",
436         "1": "Earnings eligible"
437     }
438 },
439 "A_CIVLF": {
440     "label": "Civilian labor force",
441     "universe": "All persons",
442     "type": "Categorical",
443     "role": "Independent",
444     "topic": "Basic CPS items",
445     "subtopic": "Labor force person recodes",
446     "values": {
447         "0": "Not in universe or children and Armed Forces",
448         "1": "In universe"
449     }
450 },
451 "A_CLSWKR": {

```

```

452     "label": "Class of worker",
453     "universe": "PEMLR=1-3 or (PEMLR=4-7 and person worked in the last
        12 months)",
454     "type": "Categorical",
455     "role": "Independent",
456     "topic": "Basic CPS items",
457     "subtopic": "Labor force person recodes",
458     "values": {
459         "0": "Not in universe or children and armed forces",
460         "1": "Private",
461         "2": "Federal government",
462         "3": "State government",
463         "4": "Local government",
464         "5": "Self-employed-incorporated",
465         "6": "Self-employed-not incorporated",
466         "7": "Without pay",
467         "8": "Never worked"
468     }
469 },
470 "A_EXPLF": {
471     "label": "Experienced labor force employment status",
472     "universe": "PEMLR=1-4",
473     "type": "Categorical",
474     "role": "Independent",
475     "topic": "Basic CPS items",
476     "subtopic": "Labor force person recodes",
477     "values": {
478         "0": "Not in experienced labor force",
479         "1": "Employed",
480         "2": "Unemployed"
481     }
482 },
483 "A_LFSR": {
484     "label": "Labor force status recode",
485     "universe": "All persons",
486     "type": "Categorical",

```

```

487     "role": "Independent",
488     "topic": "Basic CPS items",
489     "subtopic": "Labor force person recodes",
490     "values": {
491         "0": "Children or Armed Forces",
492         "1": "Working",
493         "2": "With job, not at work",
494         "3": "Unemployed, looking for work",
495         "4": "Unemployed, on layoff",
496         "7": "Nilf"
497     }
498 },
499 "A_UNCOV": {
500     "label": "On this job, is ... covered by a union or employee
501         association contract?",
502     "universe": "A_UNMEM=2",
503     "type": "Categorical",
504     "role": "Independent",
505     "topic": "Basic CPS items",
506     "subtopic": "Labor force person recodes",
507     "values": {
508         "0": "Not in universe or children and armed forces",
509         "1": "Yes",
510         "2": "No"
511     }
512 },
513 "A_UNMEM": {
514     "label": "On this job, is ... a member of a labor union or of an
515         employee association similar to a union?",
516     "universe": "PRERELG=1",
517     "type": "Categorical",
518     "role": "Independent",
519     "topic": "Basic CPS items",
520     "subtopic": "Labor force person recodes",
521     "values": {
522         "0": "Not in universe or children and armed forces",

```

```

521         "1": "Yes",
522         "2": "No"
523     }
524 },
525 "A_UNTYPE": {
526     "label": "Reason for unemployment",
527     "universe": "A_LFSR=3 or 4",
528     "type": "Categorical",
529     "role": "Independent",
530     "topic": "Basic CPS items",
531     "subtopic": "Labor force person recodes",
532     "values": {
533         "0": "Not in universe or children and Armed Forces",
534         "1": "Job loser - on layoff",
535         "2": "Other job loser",
536         "3": "Job leaver",
537         "4": "Re-entrant",
538         "5": "New entrant"
539     }
540 },
541 "A_USLHRS": {
542     "label": "How many hrs per week does ... usually work at this job?"
543     ,
544     "universe": "All persons",
545     "type": "Continuous",
546     "role": "Independent",
547     "topic": "Basic CPS items",
548     "subtopic": "Labor force person recodes",
549     "values": {
550         "-4": "Hours vary",
551         "-1": "Not in universe",
552         "00": "None, no hours",
553         "01-99": "Entry"
554     }
555 },
556 "A_WKSCH": {

```

```

556     "label": "Labor force by time worked or lost",
557     "universe": "All persons",
558     "type": "Categorical",
559     "role": "Independent",
560     "topic": "Basic CPS items",
561     "subtopic": "Labor force person recodes",
562     "values": {
563         "0": "Not in universe",
564         "1": "At work",
565         "2": "With job, not at work",
566         "3": "Unemployed, seeks FT",
567         "4": "Unemployed, seeks PT"
568     }
569 },
570 "A_WKSLK": {
571     "label": "Duration of unemployment",
572     "universe": "PEMLR=3 or 4",
573     "type": "Continuous",
574     "role": "Independent",
575     "topic": "Basic CPS items",
576     "subtopic": "Labor force person recodes",
577     "values": {
578         "000": "NIU, Children or Armed Forces",
579         "001-999": "Entry"
580     }
581 },
582 "A_WKSTAT": {
583     "label": "Full/part-time status",
584     "universe": "All persons",
585     "type": "Categorical",
586     "role": "Independent",
587     "topic": "Basic CPS items",
588     "subtopic": "Labor force person recodes",
589     "values": {
590         "0": "Children or Armed Forces",
591         "1": "Not in labor force",

```



```

592         "2": "Full-time schedules",
593         "3": "Part-time for economic reasons, usually FT",
594         "4": "Part-time for non-economic reasons, usually PT",
595         "5": "Part-time for economic reasons, usually PT",
596         "6": "Unemployed FT",
597         "7": "Unemployed PT"
598     }
599 },
600 "PEHRUSLT": {
601     "label": "Hours usually worked last week",
602     "universe": "All persons",
603     "type": "Continuous",
604     "role": "Independent",
605     "topic": "Basic CPS items",
606     "subtopic": "Labor force person recodes",
607     "values": {
608         "-4": "Hours vary",
609         "-1": "NIU - adult civilian",
610         "000": "NIU - children or Armed Forces or no hours",
611         "1-198": "# of hours"
612     }
613 },
614 "PEMLR": {
615     "label": "Major labor force recode",
616     "universe": "All persons",
617     "type": "Categorical",
618     "role": "Independent",
619     "topic": "Basic CPS items",
620     "subtopic": "Labor force person recodes",
621     "values": {
622         "0": "NIU",
623         "1": "Employed - at work",
624         "2": "Employed - absent",
625         "3": "Unemployed - on layoff",
626         "4": "Unemployed - looking",
627         "5": "Not in labor force - retired",

```

```

628         "6": "Not in labor force - disabled",
629         "7": "Not in labor force - other"
630     }
631 },
632     "PRCOW1": {
633         "label": "Class of worker recode-job 1",
634         "universe": "All persons",
635         "type": "Categorical",
636         "role": "Independent",
637         "topic": "Basic CPS items",
638         "subtopic": "Labor force person recodes",
639         "values": {
640             "0": "NIU",
641             "1": "Federal govt",
642             "2": "State govt",
643             "3": "Local govt",
644             "4": "Private (incl. self-employed incorp.)",
645             "5": "Self-employed, unincorp.",
646             "6": "Without pay"
647         }
648     },
649     "PRPTREA": {
650         "label": "Detailed reason for part-time",
651         "universe": "Part time workers",
652         "type": "Categorical",
653         "role": "Independent",
654         "topic": "Basic CPS items",
655         "subtopic": "Labor force person recodes",
656         "values": {
657             "0": "NIU",
658             "1": "Usually FT - slack work/business conditions",
659             "2": "Usually FT - seasonal work",
660             "3": "Usually FT - job started/ended during week",
661             "4": "Usually FT - vacation/personal day",
662             "5": "Usually FT - own illness/injury/medical appt",
663             "6": "Usually FT - holiday (religious or legal)",

```

```

664         "7": "Usually FT - child care problems",
665         "8": "Usually FT - other fam/pers obligations",
666         "9": "Usually FT - labor dispute",
667         "10": "Usually FT - weather affected job",
668         "11": "Usually FT - school/training",
669         "12": "Usually FT - civic/military duty",
670         "13": "Usually FT - other reason",
671         "14": "Usually PT - slack work/business conditions",
672         "15": "Usually PT - PT could only find PT work",
673         "16": "Usually PT - seasonal work",
674         "17": "Usually PT - child care problems",
675         "18": "Usually PT - other fam/pers obligations",
676         "19": "Usually PT - health/medical limitations",
677         "20": "Usually PT - school/training",
678         "21": "Usually PT - retired/social security limit on earnings",
679         "22": "Usually PT - workweek<35 hours",
680         "23": "Usually PT - other"
681     }
682 },
683 "PRWKSTAT": {
684     "label": "Full/part-time work status",
685     "universe": "All persons",
686     "type": "Categorical",
687     "role": "Independent",
688     "topic": "Basic CPS items",
689     "subtopic": "Labor force person recodes",
690     "values": {
691         "0": "NIU",
692         "1": "Not in labor force",
693         "2": "FT hours (35+), usually FT",
694         "3": "PT for economic reasons, usually FT",
695         "4": "PT for non-economic reasons, usually FT",
696         "5": "Not at work, usually FT",
697         "6": "PT hrs, usually PT for economic reasons",
698         "7": "PT hrs, usually PT for non-economic",
699         "8": "FT hours, usually PT for economic reasons",

```

```

700         "9": "FT hours, usually PT for non-economic reasons",
701         "10": "Not at work, usually part-time",
702         "11": "Unemployed FT",
703         "12": "Unemployed PT"
704     }
705 },
706 "CLWK": {
707     "label": "Longest job class of worker (recode)",
708     "universe": "All persons aged 15+",
709     "type": "Categorical",
710     "role": "Independent",
711     "topic": "Work experience",
712     "subtopic": "General",
713     "values": {
714         "0": "Niu",
715         "1": "Private",
716         "2": "Government",
717         "3": "Self-employed",
718         "4": "Without pay",
719         "5": "Never worked"
720     }
721 },
722 "EARNER": {
723     "label": "Earner status recode",
724     "universe": "All persons aged 15+",
725     "type": "Categorical",
726     "role": "Independent",
727     "topic": "Work experience",
728     "subtopic": "General",
729     "values": {
730         "0": "Niu",
731         "1": "Earner",
732         "2": "Nonearner"
733     }
734 },
735 "HRSWK": {

```

```

736     "label": "In the weeks that ... worked how may hours did ...
           usually work per week?",
737     "universe": "WKSWORK > 0",
738     "type": "Continuous",
739     "role": "Independent",
740     "topic": "Work experience",
741     "subtopic": "General",
742     "values": {
743         "0": "Niu",
744         "1": "1 hour",
745         "2-98": "2-98 hours",
746         "99": "99 hours plus"
747     }
748 },
749 "LJCW": {
750     "label": "Longest job class of worker",
751     "universe": "WKSWORK > 0",
752     "type": "Categorical",
753     "role": "Independent",
754     "topic": "Work experience",
755     "subtopic": "General",
756     "values": {
757         "0": "Niu",
758         "1": "Private",
759         "2": "Federal",
760         "3": "State",
761         "4": "Local",
762         "5": "Self employed incorporated, yes",
763         "6": "Self employed incorporated, no or farm",
764         "7": "Without pay"
765     }
766 },
767 "NWLKWK": {
768     "label": "How may different weeks was ... looking for work or on
           layoff?",
769     "universe": "NWLOOK = 1",

```

```

770     "type": "Continuous",
771     "role": "Independent",
772     "topic": "Work experience",
773     "subtopic": "General",
774     "values": {
775         "0": "Niu",
776         "1": "1 week",
777         "2-51": "2-51 weeks",
778         "52": "52 weeks"
779     }
780 },
781 "NWLOOK": {
782     "label": "Even though ... did not work in 20.. did spend and time
              trying to find a job or on layoff?",
783     "universe": "WORKYN = 2",
784     "type": "Categorical",
785     "role": "Independent",
786     "topic": "Work experience",
787     "subtopic": "General",
788     "values": {
789         "0": "Niu",
790         "1": "Yes",
791         "2": "No"
792     }
793 },
794 "PHMEMPRS": {
795     "label": "For how many employers did ... work in 20..? if more than
              one at same time, only count it as one employer",
796     "universe": "WKSWORK > 0",
797     "type": "Categorical",
798     "role": "Independent",
799     "topic": "Work experience",
800     "subtopic": "General",
801     "values": {
802         "0": "Niu",
803         "1": "One employer",

```

```

804         "2": "Two employers",
805         "3": "3 or more employers"
806     }
807 },
808 "RSNNOTW": {
809     "label": "What was the main reason ... did not work in 20..?",
810     "universe": "WORKYN = 2",
811     "type": "Categorical",
812     "role": "Independent",
813     "topic": "Work experience",
814     "subtopic": "General",
815     "values": {
816         "0": "Niu",
817         "1": "Ill or disabled",
818         "2": "Retired",
819         "3": "Taking care of home",
820         "4": "Going to school",
821         "5": "Could not find work",
822         "6": "Other"
823     }
824 },
825 "WECLW": {
826     "label": "Longest job class of worker (persons 15+)",
827     "universe": "All persons aged 15+",
828     "type": "Categorical",
829     "role": "Independent",
830     "topic": "Work experience",
831     "subtopic": "General",
832     "values": {
833         "0": "Not in universe",
834         "1": "Agriculture (Wage and salary)",
835         "2": "Agriculture (Self-employed)",
836         "3": "Agriculture (Unpaid)",
837         "4": "Nonagriculture (Private household)",
838         "5": "Nonagriculture (Other private)",
839         "6": "Nonagriculture (Government)",

```

```

840         "7": "Nonagriculture (Self-employed)",
841         "8": "Nonagriculture (Unpaid)",
842         "9": "Nonagriculture (Never worked)"
843     }
844 },
845     "WEWKRS": {
846         "label": "Weeks worked recode",
847         "universe": "All persons aged 15+",
848         "type": "Categorical",
849         "role": "Independent",
850         "topic": "Work experience",
851         "subtopic": "General",
852         "values": {
853             "0": "Niu",
854             "1": "Full-year worker (Full time)",
855             "2": "Full-year worker (Part time)",
856             "3": "Part-year worker (Full time)",
857             "4": "Part-year worker (Part time)",
858             "5": "Part-year worker (Nonworker)"
859         }
860     },
861     "WKSWORK": {
862         "label": "During 20.. in how many weeks did ... work even for a few
            hours? (include paid vacation and sick leave as work)",
863         "universe": "Persons 15+ with WORKYN = 1",
864         "type": "Continuous",
865         "role": "Independent",
866         "topic": "Work experience",
867         "subtopic": "General",
868         "values": {
869             "0": "Niu",
870             "1": "1 week",
871             "2-51": "2-51 weeks",
872             "52": "52 weeks"
873         }
874     },

```



```

875     "WORKYN": {
876         "label": "Did ... work at a job or business at any time during
            20..?",
877         "universe": "All persons aged 15+",
878         "type": "Categorical",
879         "role": "Independent",
880         "topic": "Work experience",
881         "subtopic": "General",
882         "values": {
883             "0": "Niu",
884             "1": "Yes",
885             "2": "No"
886         }
887     },
888     "WRK_CK": {
889         "label": "Worked last year recode, including temporary and part-
            time",
890         "universe": "All persons aged 15+",
891         "type": "Categorical",
892         "role": "Independent",
893         "topic": "Work experience",
894         "subtopic": "General",
895         "values": {
896             "0": "Niu",
897             "1": "Yes",
898             "2": "No"
899         }
900     },
901     "WTEMP": {
902         "label": "Did ... do any temporary, part-time, or seasonal work
            even for a few days during 20..?",
903         "universe": "WORKYN = 2",
904         "type": "Categorical",
905         "role": "Independent",
906         "topic": "Work experience",
907         "subtopic": "General",

```

```

908     "values": {
909         "0": "Niu",
910         "1": "Yes",
911         "2": "No"
912     }
913 },
914 "ERN_OTR": {
915     "label": "Wage and salary money earned from other work, Y/N",
916     "universe": "All persons aged 15+",
917     "type": "Categorical",
918     "role": "Independent",
919     "topic": "Income",
920     "subtopic": "Earnings",
921     "values": {
922         "0": "Niu",
923         "1": "Yes",
924         "2": "No"
925     }
926 },
927 "ERN_SRCE": {
928     "label": "Source of earnings from longest job",
929     "universe": "ERN_YN = 1",
930     "type": "Categorical",
931     "role": "Independent",
932     "topic": "Income",
933     "subtopic": "Earnings",
934     "values": {
935         "0": "Niu",
936         "1": "Wage and salary",
937         "2": "Self employment",
938         "3": "Farm self employment",
939         "4": "Without pay"
940     }
941 },
942 "ERN_VAL": {

```

```

943     "label": "How much did ... earn from this employer before
          deductions in 20..? what was ... net earnings from this
          business/ farm after expenses during 20..?",
944     "universe": "ERN_YN = 1",
945     "type": "Continuous",
946     "role": "Independent",
947     "topic": "Income",
948     "subtopic": "Earnings",
949     "values": {
950         "0": "None or Niu",
951         "-9,999 - 9,999,999": "Wages & self-employment"
952     }
953 },
954 "ERN_YN": {
955     "label": "Earnings from employer or net earnings from business/
          farm after expenses from longest job during 20.. ?",
956     "universe": "WORKYN=1 or WTEMP=1",
957     "type": "Categorical",
958     "role": "Independent",
959     "topic": "Income",
960     "subtopic": "Earnings",
961     "values": {
962         "0": "Niu",
963         "1": "Yes",
964         "2": "No"
965     }
966 },
967 "FRM_VAL": {
968     "label": "Amount of farm self-employment earnings from secondary
          source",
969     "universe": "FRMOTR = 1",
970     "type": "Continuous",
971     "role": "Independent",
972     "topic": "Income",
973     "subtopic": "Earnings",
974     "values": {

```

```

975         "0": "None or Niu",
976         "-999999-999999": "Farm self employment"
977     }
978 },
979     "FRMOTR": {
980         "label": "Receiving farm self-employment from secondary source",
981         "universe": "ERN_OTR = 1",
982         "type": "Categorical",
983         "role": "Independent",
984         "topic": "Income",
985         "subtopic": "Earnings",
986         "values": {
987             "0": "Niu",
988             "1": "Yes",
989             "2": "No"
990         }
991     },
992     "FRSE_VAL": {
993         "label": "Total amount of farm self-employment earnings",
994         "universe": "ERN_YN=1 or FRMOTR=1",
995         "type": "Continuous",
996         "role": "Independent",
997         "topic": "Income",
998         "subtopic": "Earnings",
999         "values": {
1000             "0": "None or Niu;",
1001             "-999999-999999": "Farm self employment"
1002         }
1003     },
1004     "FRSE_YN": {
1005         "label": "Receiving any farm self-employment",
1006         "universe": "ERN_YN=1 or FRMOTR=1",
1007         "type": "Categorical",
1008         "role": "Independent",
1009         "topic": "Income",
1010         "subtopic": "Earnings",

```

```

1011     "values": {
1012         "0": "Niu",
1013         "1": "Yes",
1014         "2": "No"
1015     }
1016 },
1017 "PEARNVAL": {
1018     "label": "Total persons earnings",
1019     "universe": "All persons aged 15+",
1020     "type": "Continuous",
1021     "role": "Independent",
1022     "topic": "Income",
1023     "subtopic": "Earnings",
1024     "values": {
1025         "0": "None;",
1026         "negative amt": "Income (loss);",
1027         "positive amt": "Income"
1028     }
1029 },
1030 "SE_VAL": {
1031     "label": "Amount of own business self-employment earnings from
           secondary source",
1032     "universe": "SEOTR = 1",
1033     "type": "Continuous",
1034     "role": "Independent",
1035     "topic": "Income",
1036     "subtopic": "Earnings",
1037     "values": {
1038         "0": "None or niu;",
1039         "-99999-999999": "Own business self employment"
1040     }
1041 },
1042 "SEMP_VAL": {
1043     "label": "Total own business self-employment earnings (combined
           amounts in ern-val, if ern-srce=2, and se-val)",
1044     "universe": "ERN_YN=1 or SEOTR=1",

```

```

1045     "type": "Continuous",
1046     "role": "Independent",
1047     "topic": "Income",
1048     "subtopic": "Earnings",
1049     "values": {
1050         "0": "None or niu;",
1051         "-99999-999999": "Own business self employment"
1052     }
1053 },
1054 "SEMP_YN": {
1055     "label": "Receiving own business self-employment, y/n",
1056     "universe": "ERN_YN=1 or SEOTR=1",
1057     "type": "Categorical",
1058     "role": "Independent",
1059     "topic": "Income",
1060     "subtopic": "Earnings",
1061     "values": {
1062         "0": "Niu",
1063         "1": "Yes",
1064         "2": "No"
1065     }
1066 },
1067 "SEOTR": {
1068     "label": "Receiving own business self-employment, y/n",
1069     "universe": "ERN_YN=1 or SEOTR=1",
1070     "type": "Categorical",
1071     "role": "Independent",
1072     "topic": "Income",
1073     "subtopic": "Earnings",
1074     "values": {
1075         "0": "Niu",
1076         "1": "Yes",
1077         "2": "No"
1078     }
1079 },
1080 "WAGEOTR": {

```

```

1081     "label": "Receiving wage and salary earnings from other employers,
        y/n",
1082     "universe": "ERN_OTR = 1",
1083     "type": "Categorical",
1084     "role": "Independent",
1085     "topic": "Income",
1086     "subtopic": "Earnings",
1087     "values": {
1088         "0": "Niu",
1089         "1": "Yes",
1090         "2": "No"
1091     }
1092 },
1093 "WS_VAL": {
1094     "label": "Amount of wage and salary earnings from other employers",
1095     "universe": "ERN_OTR = 1",
1096     "type": "Continuous",
1097     "role": "Independent",
1098     "topic": "Income",
1099     "subtopic": "Earnings",
1100     "values": {
1101         "0": "None or niu;",
1102         "1-9999999": "Wage and salary"
1103     }
1104 },
1105 "WSAL_VAL": {
1106     "label": "Total wage and salary earnings (combined amounts in ern-
        val, if ern-srce=1, and ws-val)",
1107     "universe": "ERN_YN=1 or WAGEOTR=1",
1108     "type": "Continuous",
1109     "role": "Independent",
1110     "topic": "Income",
1111     "subtopic": "Earnings",
1112     "values": {
1113         "0": "None or niu;",
1114         "1-9999999": "Wage and salary"

```

```

1115     }
1116 },
1117 "WSAL_YN": {
1118     "label": "Receiving wage and salary earnings",
1119     "universe": "ERN_YN=1 or WAGEOTR=1",
1120     "type": "Categorical",
1121     "role": "Independent",
1122     "topic": "Income",
1123     "subtopic": "Earnings",
1124     "values": {
1125         "0": "Niu",
1126         "1": "Yes",
1127         "2": "No"
1128     }
1129 },
1130 "ANN_VAL": {
1131     "label": "Retirement income, annuities amount",
1132     "universe": "ANN_YN = 1",
1133     "type": "Continuous",
1134     "role": "Independent",
1135     "topic": "Income",
1136     "subtopic": "Other income",
1137     "values": {
1138         "-1": "Niu",
1139         "0-999999": "Dollar amount"
1140     }
1141 },
1142 "ANN_YN": {
1143     "label": "Retirement income, annuities, y/n",
1144     "universe": "All Persons aged 15+",
1145     "type": "Categorical",
1146     "role": "Independent",
1147     "topic": "Income",
1148     "subtopic": "Other income",
1149     "values": {
1150         "0": "Niu",

```



```

1151         "1": "Yes",
1152         "2": "No"
1153     }
1154 },
1155 "CAP_VAL": {
1156     "label": "Capital gains value",
1157     "universe": "CAP_YN = 1",
1158     "type": "Continuous",
1159     "role": "Independent",
1160     "topic": "Income",
1161     "subtopic": "Other income",
1162     "values": {
1163         "0": "None or niu",
1164         "1-999999": "Captial gains amount"
1165     }
1166 },
1167 "CAP_YN": {
1168     "label": "Yes/no answer to 'Did you receive capital gain from your
        shares of stock or mutual fund?'",
1169     "universe": "DIV_YN = 1",
1170     "type": "Categorical",
1171     "role": "Independent",
1172     "topic": "Income",
1173     "subtopic": "Other income",
1174     "values": {
1175         "0": "Niu",
1176         "1": "Yes",
1177         "2": "No"
1178     }
1179 },
1180 "DBTN_VAL": {
1181     "label": "Total amount of retirement distributions received (
        dst_val1 + dst_val2)",
1182     "universe": "DST_VAL1>0 OR DST_VAL2>0",
1183     "type": "Continuous",
1184     "role": "Independent",

```

```

1185     "topic": "Income",
1186     "subtopic": "Other income",
1187     "values": {
1188         "0": "None or niu",
1189         "1-9999999": "Dollar amount"
1190     }
1191 },
1192 "DIS_SC1": {
1193     "label": "What was the source of disability income?",
1194     "universe": "DIS_YN=1",
1195     "type": "Categorical",
1196     "role": "Independent",
1197     "topic": "Income",
1198     "subtopic": "Other income",
1199     "values": {
1200         "0": "Niu",
1201         "1": "Worker's compensation",
1202         "2": "Company or union disability",
1203         "3": "Federal government disability",
1204         "4": "Us military retirement disability",
1205         "5": "State or local gov't employee disability",
1206         "6": "Us railroad retirement disability",
1207         "7": "Accident or disability insurance",
1208         "8": "Blacklung miners disability",
1209         "9": "State temporary sickness",
1210         "10": "Other or don't know"
1211     }
1212 },
1213 "DIS_SC2": {
1214     "label": "What was the source of disability income?",
1215     "universe": "DIS_YN=1",
1216     "type": "Categorical",
1217     "role": "Independent",
1218     "topic": "Income",
1219     "subtopic": "Other income",
1220     "values": {

```

```

1221         "0": "Niu",
1222         "1": "Worker's compensation",
1223         "2": "Company or union disability",
1224         "3": "Federal government disability",
1225         "4": "Us military retirement disability",
1226         "5": "State or local gov't employee disability",
1227         "6": "Us railroad retirement disability",
1228         "7": "Accident or disability insurance",
1229         "8": "Blacklung miners disability",
1230         "9": "State temporary sickness",
1231         "10": "Other or don't know"
1232     }
1233 },
1234 "DIS_VAL1": {
1235     "label": "How much did ... receive (source type) during 20.. ?",
1236     "universe": "DIS_SC1>0",
1237     "type": "Continuous",
1238     "role": "Independent",
1239     "topic": "Income",
1240     "subtopic": "Other income",
1241     "values": {
1242         "0": "None or niu",
1243         "1-999999": "Disability income"
1244     }
1245 },
1246 "DIS_VAL2": {
1247     "label": "How much did ... receive (source type) during 20.. ?",
1248     "universe": "DIS_SC2>0",
1249     "type": "Continuous",
1250     "role": "Independent",
1251     "topic": "Income",
1252     "subtopic": "Other income",
1253     "values": {
1254         "0": "None or niu",
1255         "1-999999": "Disability income"
1256     }

```

```

1257     },
1258     "DIS_YN": {
1259         "label": "Other than social security did ... receive any income in
                20.. as a result of health problems?",
1260         "universe": "All Persons aged 15+",
1261         "type": "Categorical",
1262         "role": "Independent",
1263         "topic": "Income",
1264         "subtopic": "Other income",
1265         "values": {
1266             "0": "Niu",
1267             "1": "Yes",
1268             "2": "No"
1269         }
1270     },
1271     "DIV_VAL": {
1272         "label": "How much did ... receive in dividends from stocks or
                mutual funds during 20.. ?",
1273         "universe": "DIV_YN = 1",
1274         "type": "Continuous",
1275         "role": "Independent",
1276         "topic": "Income",
1277         "subtopic": "Other income",
1278         "values": {
1279             "0": "None or niu",
1280             "1-999999": "Dividends"
1281         }
1282     },
1283     "DIV_YN": {
1284         "label": "Did ... receive dividends?",
1285         "universe": "All Persons aged 15+",
1286         "type": "Categorical",
1287         "role": "Independent",
1288         "topic": "Income",
1289         "subtopic": "Other income",
1290         "values": {

```

```

1291         "0": "Niu",
1292         "1": "Yes",
1293         "2": "No"
1294     }
1295 },
1296 "DSAB_VAL": {
1297     "label": "Total amount of disability income received, combined
1298         amounts in edited sources one and two",
1299     "universe": "DIS_VAL1>0 OR DIS_VAL2>0",
1300     "type": "Continuous",
1301     "role": "Independent",
1302     "topic": "Income",
1303     "subtopic": "Other income",
1304     "values": {
1305         "0": "None or niu",
1306         "1-999999": "Disability income"
1307     }
1308 },
1309 "DST_SC1": {
1310     "label": "Retirement income, distribution source 1",
1311     "universe": "DST_VAL1 > 0 and a_age >= 58",
1312     "type": "Categorical",
1313     "role": "Independent",
1314     "topic": "Income",
1315     "subtopic": "Other income",
1316     "values": {
1317         "0": "Niu",
1318         "1": "401k account",
1319         "2": "403b account",
1320         "3": "Roth ira",
1321         "4": "Regular ira",
1322         "5": "Keogh plan",
1323         "6": "Sep plan (simplified employee pension)",
1324         "7": "Other type of retirement account"
1325     }
1326 },

```

```

1326     "DST_SC1_YNG": {
1327         "label": "Retirement Distribution source 1, person under age 58",
1328         "universe": "DST_YN_YNG = 1 and a_age < 58",
1329         "type": "Categorical",
1330         "role": "Independent",
1331         "topic": "Income",
1332         "subtopic": "Other income",
1333         "values": {
1334             "0": "Niu",
1335             "1": "401k account",
1336             "2": "403b account",
1337             "3": "Roth ira",
1338             "4": "Regular ira",
1339             "5": "Keogh plan",
1340             "6": "Sep plan (simplified employee pension)",
1341             "7": "Other type of retirement account"
1342         }
1343     },
1344     "DST_SC2": {
1345         "label": "Retirement income, distribution source 2",
1346         "universe": "DST_VAL2 > 0 and a_age >= 58",
1347         "type": "Categorical",
1348         "role": "Independent",
1349         "topic": "Income",
1350         "subtopic": "Other income",
1351         "values": {
1352             "0": "Niu",
1353             "1": "401k account",
1354             "2": "403b account",
1355             "3": "Roth ira",
1356             "4": "Regular ira",
1357             "5": "Keogh plan",
1358             "6": "Sep plan (simplified employee pension)",
1359             "7": "Other type of retirement account"
1360         }
1361     },

```

```

1362     "DST_SC2_YNG": {
1363         "label": "Retirement Distribution source 2, person under age 58",
1364         "universe": "DST_VAL_YNG > 0 and a_age < 58",
1365         "type": "Categorical",
1366         "role": "Independent",
1367         "topic": "Income",
1368         "subtopic": "Other income",
1369         "values": {
1370             "0": "Niu",
1371             "1": "401k account",
1372             "2": "403b account",
1373             "3": "Roth ira",
1374             "4": "Regular ira",
1375             "5": "Keogh plan",
1376             "6": "Sep plan (simplified employee pension)",
1377             "7": "Other type of retirement account"
1378         }
1379     },
1380     "DST_VAL1": {
1381         "label": "Retirement income amount, distribution source 1",
1382         "universe": "DST_SC1 = 1",
1383         "type": "Continuous",
1384         "role": "Independent",
1385         "topic": "Income",
1386         "subtopic": "Other income",
1387         "values": {
1388             "0": "None or niu",
1389             "1- 999,999": "Amount withdrawn or distributed"
1390         }
1391     },
1392     "DST_VAL1_YNG": {
1393         "label": "Retirement Distribution amount 1, under age 58",
1394         "universe": "DST_SC1_YNG = 1",
1395         "type": "Continuous",
1396         "role": "Independent",
1397         "topic": "Income",

```

```

1398     "subtopic": "Other income",
1399     "values": {
1400         "0": "None or niu",
1401         "1- 999,999": "Amount withdrawn or distributed"
1402     }
1403 },
1404 "DST_VAL2": {
1405     "label": "Retirement income amount, distribution source 2",
1406     "universe": "DST_SC2 = 1",
1407     "type": "Continuous",
1408     "role": "Independent",
1409     "topic": "Income",
1410     "subtopic": "Other income",
1411     "values": {
1412         "0": "None or niu",
1413         "1- 999,999": "Amount withdrawn or distributed"
1414     }
1415 },
1416 "DST_VAL2_YNG": {
1417     "label": "Retirement Distribution amount 2, under age 58",
1418     "universe": "DST_SC2_YNG = 1",
1419     "type": "Continuous",
1420     "role": "Independent",
1421     "topic": "Income",
1422     "subtopic": "Other income",
1423     "values": {
1424         "0": "None or niu",
1425         "1- 999,999": "Amount withdrawn or distributed"
1426     }
1427 },
1428 "DST_YN": {
1429     "label": "Retirement income distribution y/n",
1430     "universe": "Persons aged 58 and over (a_age >= 58)",
1431     "type": "Categorical",
1432     "role": "Independent",
1433     "topic": "Income",

```



```

1434     "subtopic": "Other income",
1435     "values": {
1436         "0": "Niu",
1437         "1": "Yes",
1438         "2": "No"
1439     }
1440 },
1441 "DST_YN_YNG": {
1442     "label": "Retirement Distribution Reciprocity, person under age 58",
1443     "universe": "Persons under age 58 (a_age < 58)",
1444     "type": "Categorical",
1445     "role": "Independent",
1446     "topic": "Income",
1447     "subtopic": "Other income",
1448     "values": {
1449         "0": "Niu",
1450         "1": "Yes",
1451         "2": "No"
1452     }
1453 },
1454 "ED_VAL": {
1455     "label": "Total amount of educational assistance received (combined
           amounts in pell grant and other educational) assistance during
           20.. ?",
1456     "universe": "ED_YN = 1",
1457     "type": "Continuous",
1458     "role": "Independent",
1459     "topic": "Income",
1460     "subtopic": "Other income",
1461     "values": {
1462         "0": "None or niu",
1463         "1- 99,999": "Dollar amount"
1464     }
1465 },
1466 "ED_YN": {
1467     "label": "Did ... receive educational assistance?",

```

```

1468     "universe": "All Persons aged 15+",
1469     "type": "Categorical",
1470     "role": "Independent",
1471     "topic": "Income",
1472     "subtopic": "Other income",
1473     "values": {
1474         "0": "Niu",
1475         "1": "Yes",
1476         "2": "No"
1477     }
1478 },
1479 "FIN_VAL": {
1480     "label": "How much did ... receive in financial assistance income
           during 20.. ?",
1481     "universe": "FIN_YN = 1",
1482     "type": "Continuous",
1483     "role": "Independent",
1484     "topic": "Income",
1485     "subtopic": "Other income",
1486     "values": {
1487         "0": "None or niu",
1488         "1-999999": "Financial assistance"
1489     }
1490 },
1491 "FIN_YN": {
1492     "label": "Did ... receive financial assistance?",
1493     "universe": "All Persons aged 15+",
1494     "type": "Categorical",
1495     "role": "Independent",
1496     "topic": "Income",
1497     "subtopic": "Other income",
1498     "values": {
1499         "0": "Niu",
1500         "1": "Yes",
1501         "2": "No"
1502     }

```

```

1503     },
1504     "INT_VAL": {
1505         "label": "Edited total combined interest income",
1506         "universe": "INT_YN = 1",
1507         "type": "Continuous",
1508         "role": "Independent",
1509         "topic": "Income",
1510         "subtopic": "Other income",
1511         "values": {
1512             "0": "None or niu;",
1513             "1- 999,999": "Dollar amount"
1514         }
1515     },
1516     "INT_YN": {
1517         "label": "Edited total combined interest income, y/n",
1518         "universe": "All Persons aged 15+",
1519         "type": "Categorical",
1520         "role": "Independent",
1521         "topic": "Income",
1522         "subtopic": "Other income",
1523         "values": {
1524             "0": "Niu",
1525             "1": "Yes",
1526             "2": "No"
1527         }
1528     },
1529     "OED_TYP1": {
1530         "label": "Source 1 other than gi bill received (OED_TYP1- source of
1531             other government assistance)",
1532         "universe": "ED_YN = 1",
1533         "type": "Categorical",
1534         "role": "Independent",
1535         "topic": "Income",
1536         "subtopic": "Other income",
1537         "values": {
1538             "0": "Niu",

```

```

1538         "1": "Yes",
1539         "2": "No"
1540     }
1541 },
1542 "OED_TYP2": {
1543     "label": "Source 2 other than gi bill received (OED_TYP2-
           scholarships, grants etc. from the school)",
1544     "universe": "ED_YN = 1",
1545     "type": "Categorical",
1546     "role": "Independent",
1547     "topic": "Income",
1548     "subtopic": "Other income",
1549     "values": {
1550         "0": "Niu",
1551         "1": "Yes",
1552         "2": "No"
1553     }
1554 },
1555 "OED_TYP3": {
1556     "label": "Source other than gi bill received (OED_TYP3- other
           assistance (employers friends, etc.)",
1557     "universe": "ED_YN = 1",
1558     "type": "Categorical",
1559     "role": "Independent",
1560     "topic": "Income",
1561     "subtopic": "Other income",
1562     "values": {
1563         "0": "Niu",
1564         "1": "Yes",
1565         "2": "No"
1566     }
1567 },
1568 "OI_OFF": {
1569     "label": "Other income sources",
1570     "universe": "OI_YN = 1",
1571     "type": "Categorical",

```

```

1572     "role": "Independent",
1573     "topic": "Income",
1574     "subtopic": "Other income",
1575     "values": {
1576         "0": "Niu",
1577         "1": "Social security",
1578         "2": "Private pensions",
1579         "3": "Afdc",
1580         "4": "Other public assistance",
1581         "5": "Interest",
1582         "6": "Dividends",
1583         "7": "Rents or royalties",
1584         "8": "Estates or trusts",
1585         "9": "State disability payments (worker's comp)",
1586         "10": "Disability payments (own insurance)",
1587         "11": "Unemployment compensation",
1588         "12": "Strike benefits",
1589         "13": "Annuities or paid up insurance policies",
1590         "14": "Not income",
1591         "15": "Longest job",
1592         "16": "Wages or salary",
1593         "17": "Nonfarm self-employment",
1594         "18": "Farm self-employment",
1595         "19": "Anything else",
1596         "20": "Alimony"
1597     }
1598 },
1599 "OI_VAL": {
1600     "label": "How much did ... receive in other incomes",
1601     "universe": "OI_YN = 1",
1602     "type": "Continuous",
1603     "role": "Independent",
1604     "topic": "Income",
1605     "subtopic": "Other income",
1606     "values": {
1607         "0": "None or niu",

```

```

1608         "1-999999": "Other income"
1609     }
1610 },
1611 "OI_YN": {
1612     "label": "Did ... receive cash income not already covered from any
1613         other source?",
1614     "universe": "All Persons aged 15+",
1615     "type": "Categorical",
1616     "role": "Independent",
1617     "topic": "Income",
1618     "subtopic": "Other income",
1619     "values": {
1620         "0": "None or niu",
1621         "1": "Yes",
1622         "2": "No"
1623     }
1624 },
1625 "PEN_SC1": {
1626     "label": "Retirement income, pension source 1",
1627     "universe": "PEN_YN = 1",
1628     "type": "Categorical",
1629     "role": "Independent",
1630     "topic": "Income",
1631     "subtopic": "Other income",
1632     "values": {
1633         "0": "Niu",
1634         "1": "Company pension",
1635         "2": "Union pension",
1636         "3": "Federal government pension",
1637         "4": "State government pension",
1638         "5": "Local government pension",
1639         "6": "Us military pension",
1640         "7": "Us railroad retirement",
1641         "8": "Other"
1642     }
1643 },

```

```

1643     "PEN_SC2": {
1644         "label": "Retirement income, pension source 2",
1645         "universe": "PEN_VAL2 > 0",
1646         "type": "Categorical",
1647         "role": "Independent",
1648         "topic": "Income",
1649         "subtopic": "Other income",
1650         "values": {
1651             "0": "Niu",
1652             "1": "Company pension",
1653             "2": "Union pension",
1654             "3": "Federal government pension",
1655             "4": "State government pension",
1656             "5": "Local government pension",
1657             "6": "Us military pension",
1658             "7": "Us railroad retirement",
1659             "8": "Other"
1660         }
1661     },
1662     "PEN_VAL1": {
1663         "label": "Retirement income amount, pension source 1",
1664         "universe": "PEN_SC1 > 0",
1665         "type": "Continuous",
1666         "role": "Independent",
1667         "topic": "Income",
1668         "subtopic": "Other income",
1669         "values": {
1670             "0": "None or niu",
1671             "1-999,999": "Pension income"
1672         }
1673     },
1674     "PEN_VAL2": {
1675         "label": "Retirement income amount, pension source 2",
1676         "universe": "PEN_SC2 > 0",
1677         "type": "Continuous",
1678         "role": "Independent",

```

```

1679     "topic": "Income",
1680     "subtopic": "Other income",
1681     "values": {
1682         "0": "None or niu",
1683         "1-999,999": "Pension income"
1684     }
1685 },
1686 "PEN_YN": {
1687     "label": "Retirement income, pension y/n",
1688     "universe": "All Persons aged 15+",
1689     "type": "Categorical",
1690     "role": "Independent",
1691     "topic": "Income",
1692     "subtopic": "Other income",
1693     "values": {
1694         "0": "Niu",
1695         "1": "Yes",
1696         "2": "No"
1697     }
1698 },
1699 "PNSN_VAL": {
1700     "label": "Total combined amount of pension income received from all
           pension sources",
1701     "universe": "PEN_YN = 1",
1702     "type": "Continuous",
1703     "role": "Independent",
1704     "topic": "Income",
1705     "subtopic": "Other income",
1706     "values": {
1707         "0": "None or niu",
1708         "1-9,999,999": "Retirement income"
1709     }
1710 },
1711 "PTOTVAL": {
1712     "label": "Total persons income",
1713     "universe": "All Persons aged 15+",

```



```

1714     "type": "Continuous",
1715     "role": "Independent",
1716     "topic": "Income",
1717     "subtopic": "Other income",
1718     "values": {
1719         "0": "None",
1720         "negative amt": "Income (loss)",
1721         "positive amt": "Income"
1722     }
1723 },
1724 "RESNSS1": {
1725     "label": "What were the reasons (you/name) (was/were) getting
           Social Security Income last year?",
1726     "universe": "SS_YN = 1",
1727     "type": "Categorical",
1728     "role": "Independent",
1729     "topic": "Income",
1730     "subtopic": "Other income",
1731     "values": {
1732         "0": "Niu",
1733         "1": "Retired",
1734         "2": "Disabled (adult or child)",
1735         "3": "Widowed",
1736         "4": "Spouse",
1737         "5": "Surviving child",
1738         "6": "Dependent child",
1739         "7": "On behalf of surviving, dependent, or disabled child(ren)
           ",
1740         "8": "Other (adult or child)"
1741     }
1742 },
1743 "RESNSS2": {
1744     "label": "What were the reasons (you/name) (was/were) getting
           Social Security Income last year?",
1745     "universe": "SS_YN = 1",
1746     "type": "Categorical",

```

```

1747     "role": "Independent",
1748     "topic": "Income",
1749     "subtopic": "Other income",
1750     "values": {
1751         "0": "Niu",
1752         "1": "Retired",
1753         "2": "Disabled (adult or child)",
1754         "3": "Widowed",
1755         "4": "Spouse",
1756         "5": "Surviving child",
1757         "6": "Dependent child",
1758         "7": "On behalf of surviving, dependent, or disabled child(ren)
           ",
1759         "8": "Other (adult or child)"
1760     }
1761 },
1762 "RESNSSI1": {
1763     "label": "What were the reasons (you/name) (was/were) getting
           Supplemental Security Income last year?",
1764     "universe": "SSI_YN = 1",
1765     "type": "Categorical",
1766     "role": "Independent",
1767     "topic": "Income",
1768     "subtopic": "Other income",
1769     "values": {
1770         "0": "Niu",
1771         "1": "Disabled (adult or child)",
1772         "2": "Blind (adult or child)",
1773         "3": "On behalf of a disabled child",
1774         "4": "On behalf of a blind child",
1775         "5": "Other (adult or child)"
1776     }
1777 },
1778 "RESNSSI2": {
1779     "label": "What were the reasons (you/name) (was/were) getting
           Supplemental Security Income last year?",

```

```

1780     "universe": "SSI_YN = 1",
1781     "type": "Categorical",
1782     "role": "Independent",
1783     "topic": "Income",
1784     "subtopic": "Other income",
1785     "values": {
1786         "0": "Niu",
1787         "1": "Disabled (adult or child)",
1788         "2": "Blind (adult or child)",
1789         "3": "On behalf of a disabled child",
1790         "4": "On behalf of a blind child",
1791         "5": "Other (adult or child)"
1792     }
1793 },
1794 "RETCB_VAL": {
1795     "label": "Retirement contributiion, amount",
1796     "universe": "RETCB_YN = 1",
1797     "type": "Continuous",
1798     "role": "Independent",
1799     "topic": "Income",
1800     "subtopic": "Other income",
1801     "values": {
1802         "0": "None or niu",
1803         "1-99999": "Amount contributed"
1804     }
1805 },
1806 "RETCB_YN": {
1807     "label": "Retirement contribution, y/n",
1808     "universe": "All people 15 years and over",
1809     "type": "Categorical",
1810     "role": "Independent",
1811     "topic": "Income",
1812     "subtopic": "Other income",
1813     "values": {
1814         "0": "Niu",
1815         "1": "Yes",

```

```

1816         "2": "No"
1817     }
1818 },
1819 "RINT_SC1": {
1820     "label": "Interest income, retirement source 1",
1821     "universe": "RINT_YN = 1",
1822     "type": "Categorical",
1823     "role": "Independent",
1824     "topic": "Income",
1825     "subtopic": "Other income",
1826     "values": {
1827         "0": "Niu",
1828         "1": "401k account",
1829         "2": "403b account",
1830         "3": "Roth ira",
1831         "4": "Regular ira",
1832         "5": "Keogh plan",
1833         "6": "Sep plan (simplified employee pension)",
1834         "7": "Other type of retirement account"
1835     }
1836 },
1837 "RINT_SC2": {
1838     "label": "Interest income, retirement source 2",
1839     "universe": "RINT_YN = 1",
1840     "type": "Categorical",
1841     "role": "Independent",
1842     "topic": "Income",
1843     "subtopic": "Other income",
1844     "values": {
1845         "0": "Niu",
1846         "1": "401k account",
1847         "2": "403b account",
1848         "3": "Roth ira",
1849         "4": "Regular ira",
1850         "5": "Keogh plan",
1851         "6": "Sep plan (simplified employee pension)",

```

```

1852         "7": "Other type of retirement account"
1853     }
1854 },
1855 "RINT_VAL1": {
1856     "label": "Interest income amt, retirement source 1",
1857     "universe": "RINT_SC1 > 0",
1858     "type": "Continuous",
1859     "role": "Independent",
1860     "topic": "Income",
1861     "subtopic": "Other income",
1862     "values": {
1863         "0": "None or niu",
1864         "1-999999": "Ret interest income"
1865     }
1866 },
1867 "RINT_VAL2": {
1868     "label": "Interest income amt, retirement source 2",
1869     "universe": "RINT_SC2 > 0",
1870     "type": "Continuous",
1871     "role": "Independent",
1872     "topic": "Income",
1873     "subtopic": "Other income",
1874     "values": {
1875         "0": "None or niu",
1876         "1-999999": "Ret interest income"
1877     }
1878 },
1879 "RINT_YN": {
1880     "label": "Interest income - retirement, y/n",
1881     "universe": "All Persons aged 15+",
1882     "type": "Categorical",
1883     "role": "Independent",
1884     "topic": "Income",
1885     "subtopic": "Other income",
1886     "values": {
1887         "0": "Niu",

```

```

1888         "1": "Yes",
1889         "2": "No"
1890     }
1891 },
1892 "RNT_VAL": {
1893     "label": "How much did ... receive in income from rent after
1894         expenses during 20..?",
1895     "universe": "RNT_YN = 1",
1896     "type": "Continuous",
1897     "role": "Independent",
1898     "topic": "Income",
1899     "subtopic": "Other income",
1900     "values": {
1901         "0": "None or niu",
1902         "-9999-999999": "Rental income"
1903     }
1904 },
1905 "RNT_YN": {
1906     "label": "Did ... own any land, property, rented to others, or
1907         receive income from royalties, roomers or boarders, or from
1908         estates or trusts?",
1909     "universe": "All Persons aged 15+",
1910     "type": "Categorical",
1911     "role": "Independent",
1912     "topic": "Income",
1913     "subtopic": "Other income",
1914     "values": {
1915         "0": "Niu",
1916         "1": "Yes",
1917         "2": "No"
1918     }
1919 },
1920 "SRVS_VAL": {
1921     "label": "Total amount of survivor's income received (combined
1922         amounts in edited sources sur_val1 and sur_val2 plus the
1923         unedited sources 3 & 4 starting in 1995)",

```

```

1919     "universe": "SUR_YN = 1",
1920     "type": "Continuous",
1921     "role": "Independent",
1922     "topic": "Income",
1923     "subtopic": "Other income",
1924     "values": {
1925         "0": "None or niu",
1926         "1-999999": "Income amount"
1927     }
1928 },
1929 "SS_VAL": {
1930     "label": "How much did ... receive in social security payments
           during 20.. ?",
1931     "universe": "SS_YN = 1",
1932     "type": "Continuous",
1933     "role": "Independent",
1934     "topic": "Income",
1935     "subtopic": "Other income",
1936     "values": {
1937         "0": "None or niu",
1938         "1-999999": "Social security"
1939     }
1940 },
1941 "SS_YN": {
1942     "label": "Who received social security payments either for
           themselves or as combined payments with other family members?",
1943     "universe": "All Persons aged 15+",
1944     "type": "Categorical",
1945     "role": "Independent",
1946     "topic": "Income",
1947     "subtopic": "Other income",
1948     "values": {
1949         "0": "Niu",
1950         "1": "Yes",
1951         "2": "No"
1952     }

```

```

1953     },
1954     "SSI_VAL": {
1955         "label": "How much did ... receive in supplemental security income
1956             during 20..?",
1957         "universe": "SSI_YN = 1",
1958         "type": "Continuous",
1959         "role": "Independent",
1960         "topic": "Income",
1961         "subtopic": "Other income",
1962         "values": {
1963             "0": "None or niu",
1964             "1-99999": "Supplemental security income"
1965         }
1966     },
1967     "SSI_YN": {
1968         "label": "Did ... received ssi?",
1969         "universe": "All Persons aged 15+",
1970         "type": "Categorical",
1971         "role": "Independent",
1972         "topic": "Income",
1973         "subtopic": "Other income",
1974         "values": {
1975             "0": "Niu",
1976             "1": "Yes",
1977             "2": "No"
1978         }
1979     },
1980     "STRKUC": {
1981         "label": "At any time during 20.. did ... receive any union
1982             unemployment or strike benefits?",
1983         "universe": "UC_YN = 1",
1984         "type": "Categorical",
1985         "role": "Independent",
1986         "topic": "Income",
1987         "subtopic": "Other income",
1988         "values": {

```



```

1987         "0": "Niu",
1988         "1": "Yes",
1989         "2": "No"
1990     }
1991 },
1992 "SUBUC": {
1993     "label": "At any time during 20.. did ... receive any supplemental
1994         unemployment benefits?",
1995     "universe": "UC_YN = 1",
1996     "type": "Categorical",
1997     "role": "Independent",
1998     "topic": "Income",
1999     "subtopic": "Other income",
2000     "values": {
2001         "0": "Niu",
2002         "1": "Yes",
2003         "2": "No"
2004     }
2005 },
2006 "SUR_SC1": {
2007     "label": "What was the source of this other widow or survivor
2008         income?",
2009     "universe": "SUR_YN = 1",
2010     "type": "Categorical",
2011     "role": "Independent",
2012     "topic": "Income",
2013     "subtopic": "Other income",
2014     "values": {
2015         "0": "None or niu",
2016         "1": "Company or union survivor pension",
2017         "2": "Federal government",
2018         "3": "Us military retirement survivor pension",
2019         "4": "State or local gov't survivor pension",
2020         "5": "Us railroad retirement survivor pension",
2021         "6": "Worker compensation survivor",
2022         "7": "Black lung",

```

```

2021         "8": "Regular payments from estates or trusts",
2022         "9": "Regular payments from annuities or paid-up life insurance
2023             ",
2024         "10": "Other or don't know"
2025     },
2026     "SUR_SC2": {
2027         "label": "What was the source of this other widow or survivor
2028             income?",
2029         "universe": "SUR_YN = 1",
2030         "type": "Categorical",
2031         "role": "Independent",
2032         "topic": "Income",
2033         "subtopic": "Other income",
2034         "values": {
2035             "0": "None or niu",
2036             "1": "Company or union survivor pension",
2037             "2": "Federal government",
2038             "3": "Us military retirement survivor pension",
2039             "4": "State or local gov't survivor pension",
2040             "5": "Us railroad retirement survivor pension",
2041             "6": "Worker compensation survivor",
2042             "7": "Black lung",
2043             "8": "Regular payments from estates or trusts",
2044             "9": "Regular payments from annuities or paid-up life insurance
2045                 ",
2046             "10": "Other or don't know"
2047         }
2048     },
2049     "SUR_VAL1": {
2050         "label": "How much did ... receive (survivor source type) during
2051             20.. ?",
2052         "universe": "SUR_YN = 1",
2053         "type": "Continuous",
2054         "role": "Independent",
2055         "topic": "Income",

```

```

2053     "subtopic": "Other income",
2054     "values": {
2055         "0": "None or niu",
2056         "1-999,999": "Survivor's income"
2057     }
2058 },
2059 "SUR_VAL2": {
2060     "label": "How much did ... receive (source type) during 20.. ?",
2061     "universe": "SUR_YN = 1",
2062     "type": "Continuous",
2063     "role": "Independent",
2064     "topic": "Income",
2065     "subtopic": "Other income",
2066     "values": {
2067         "0": "None or niu",
2068         "1-999,999": "Survivor's income"
2069     }
2070 },
2071 "SUR_YN": {
2072     "label": "During 20.. did ... receive any survivor benefits such as
           widow's pensions, estates, trusts, insurance annuities, or
           other survivor's income?",
2073     "universe": "All Persons aged 15+",
2074     "type": "Categorical",
2075     "role": "Independent",
2076     "topic": "Income",
2077     "subtopic": "Other income",
2078     "values": {
2079         "0": "Niu",
2080         "1": "Yes",
2081         "2": "No"
2082     }
2083 },
2084 "TRDINT_VAL": {
2085     "label": "Interest amount, exclcuding retirment account interest",
2086     "universe": "INT_YN = 1",

```

```

2087     "type": "Continuous",
2088     "role": "Independent",
2089     "topic": "Income",
2090     "subtopic": "Other income",
2091     "values": {
2092         "all": "Dollar value"
2093     }
2094 },
2095 "UC_VAL": {
2096     "label": "How much did ... receive in unemployment benefits during
2097         20..?",
2098     "universe": "UC_YN = 1",
2099     "type": "Continuous",
2100     "role": "Independent",
2101     "topic": "Income",
2102     "subtopic": "Other income",
2103     "values": {
2104         "0": "None or niu",
2105         "1-99999": "Unemployment compensation"
2106     }
2107 },
2108 "UC_YN": {
2109     "label": "Any type of unemployment compensation? (Combination of
2110         subuc, strkuc, and uctot_yn)",
2111     "universe": "UC_YN = 1",
2112     "type": "Categorical",
2113     "role": "Independent",
2114     "topic": "Income",
2115     "subtopic": "Other income",
2116     "values": {
2117         "0": "Niu",
2118         "1": "Yes",
2119         "2": "No"
2120     }
2121 },
2122 "VET_TYP1": {

```

```

2121     "label": "What type of veterans payments did .... receive? (
          VET_TYP1- disability compensation?)",
2122     "universe": "VET_YN = 1",
2123     "type": "Categorical",
2124     "role": "Independent",
2125     "topic": "Income",
2126     "subtopic": "Other income",
2127     "values": {
2128         "0": "Niu",
2129         "1": "Yes",
2130         "2": "No"
2131     }
2132 },
2133 "VET_TYP2": {
2134     "label": "What type of veterans payments did .... receive? (
          VET_TYP2- survivor benefits?)",
2135     "universe": "VET_YN = 1",
2136     "type": "Categorical",
2137     "role": "Independent",
2138     "topic": "Income",
2139     "subtopic": "Other income",
2140     "values": {
2141         "0": "Niu",
2142         "1": "Yes",
2143         "2": "No"
2144     }
2145 },
2146 "VET_TYP3": {
2147     "label": "What type of veterans payments did .... receive? (
          VET_TYP3- veteran's pension?)",
2148     "universe": "VET_YN = 1",
2149     "type": "Categorical",
2150     "role": "Independent",
2151     "topic": "Income",
2152     "subtopic": "Other income",
2153     "values": {

```

```

2154         "0": "Niu",
2155         "1": "Yes",
2156         "2": "No"
2157     }
2158 },
2159 "VET_TYP4": {
2160     "label": "What type of veterans payments did .... receive? (
                VET_TYP4- education assistance?)",
2161     "universe": "VET_YN = 1",
2162     "type": "Categorical",
2163     "role": "Independent",
2164     "topic": "Income",
2165     "subtopic": "Other income",
2166     "values": {
2167         "0": "Niu",
2168         "1": "Yes",
2169         "2": "No"
2170     }
2171 },
2172 "VET_TYP5": {
2173     "label": "What type of veterans payments did .... receive? (
                VET_TYP5- other veteran's payments?)",
2174     "universe": "VET_YN = 1",
2175     "type": "Categorical",
2176     "role": "Independent",
2177     "topic": "Income",
2178     "subtopic": "Other income",
2179     "values": {
2180         "0": "Niu",
2181         "1": "Yes",
2182         "2": "No"
2183     }
2184 },
2185 "VET_VAL": {
2186     "label": "How much did ... receive from veterans' administration
                during 20..?",

```

```

2187     "universe": "VET_YN = 1",
2188     "type": "Continuous",
2189     "role": "Independent",
2190     "topic": "Income",
2191     "subtopic": "Other income",
2192     "values": {
2193         "0": "None or niu",
2194         "1-999999": "Veterans' payments"
2195     }
2196 },
2197 "VET_YN": {
2198     "label": "Did ... receive veterans' payments?",
2199     "universe": "All Persons aged 15+",
2200     "type": "Categorical",
2201     "role": "Independent",
2202     "topic": "Income",
2203     "subtopic": "Other income",
2204     "values": {
2205         "0": "Niu",
2206         "1": "Yes",
2207         "2": "No"
2208     }
2209 },
2210 "WC_TYPE": {
2211     "label": "What was source of these payments?",
2212     "universe": "WC_YN = 1",
2213     "type": "Categorical",
2214     "role": "Independent",
2215     "topic": "Income",
2216     "subtopic": "Other income",
2217     "values": {
2218         "0": "Not in universe",
2219         "1": "State worker's compensation",
2220         "2": "Employer or employers insurance",
2221         "3": "Own insurance",
2222         "4": "Other"

```

```

2223     }
2224 },
2225 "WC_VAL": {
2226     "label": "How much compensation did ... receive during 20..?",
2227     "universe": "WC_YN = 1",
2228     "type": "Continuous",
2229     "role": "Independent",
2230     "topic": "Income",
2231     "subtopic": "Other income",
2232     "values": {
2233         "0": "None or niu",
2234         "1-99999": "Worker's compensation"
2235     }
2236 },
2237 "WC_YN": {
2238     "label": "During 20.. did ... receive any worker's compensation
                payments or other payments as a result of a job related injury
                or illness?",
2239     "universe": "All Persons aged 15+",
2240     "type": "Categorical",
2241     "role": "Independent",
2242     "topic": "Income",
2243     "subtopic": "Other income",
2244     "values": {
2245         "0": "Niu",
2246         "1": "Yes",
2247         "2": "No"
2248     }
2249 },
2250 "PAW_TYP": {
2251     "label": "What type of program did... receive CASH assistance?",
2252     "universe": "PAW_YN = 1",
2253     "type": "Categorical",
2254     "role": "Independent",
2255     "topic": "Income",
2256     "subtopic": "Non-cash benefits",

```



```

2257     "values": {
2258         "0": "Niu",
2259         "1": "TANF/AFDC",
2260         "2": "Other",
2261         "3": "Both"
2262     }
2263 },
2264 "PAW_VAL": {
2265     "label": "How much did ... receive in public assistance or welfare
                during 20..?",
2266     "universe": "PAW_YN = 1",
2267     "type": "Continuous",
2268     "role": "Independent",
2269     "topic": "Income",
2270     "subtopic": "Non-cash benefits",
2271     "values": {
2272         "0": "None or niu",
2273         "1-99999": "Public assistance"
2274     }
2275 },
2276 "PAW_YN": {
2277     "label": "At any time during 20.., even for one month, did...
                receive an CASH assistance from a state or county welfare
                program such as (State program name fill)?",
2278     "universe": "All Persons aged 15+",
2279     "type": "Categorical",
2280     "role": "Independent",
2281     "topic": "Income",
2282     "subtopic": "Non-cash benefits",
2283     "values": {
2284         "0": "Niu",
2285         "1": "Yes",
2286         "2": "No"
2287     }
2288 },
2289 "PENINCL": {

```

```

2290     "label": "Was ... included in that plan?",
2291     "universe": "PENPLAN = 1",
2292     "type": "Categorical",
2293     "role": "Independent",
2294     "topic": "Income",
2295     "subtopic": "Non-cash benefits",
2296     "values": {
2297         "0": "Niu",
2298         "1": "Yes",
2299         "2": "No"
2300     }
2301 },
2302 "PENPLAN": {
2303     "label": "Other than social security did the employer or union that
                ... worked for in 20.. have a pension or other type of
                retirement plan?",
2304     "universe": "WRK_CK = 1",
2305     "type": "Categorical",
2306     "role": "Independent",
2307     "topic": "Income",
2308     "subtopic": "Non-cash benefits",
2309     "values": {
2310         "0": "Niu",
2311         "1": "Yes",
2312         "2": "No"
2313     }
2314 },
2315 "WICYN": {
2316     "label": "Who received WIC?",
2317     "universe": "Adult female",
2318     "type": "Categorical",
2319     "role": "Independent",
2320     "topic": "Income",
2321     "subtopic": "Non-cash benefits",
2322     "values": {
2323         "0": "Niu",

```

```

2324         "1": "Received WIC",
2325         "2": "Did not receive WIC"
2326     }
2327 },
2328 "CHCARE_YN": {
2329     "label": "Paid child care was needed for this child?",
2330     "universe": "Persons age 15+ with children",
2331     "type": "Categorical",
2332     "role": "Independent",
2333     "topic": "Income",
2334     "subtopic": "Supplemental poverty measure",
2335     "values": {
2336         "0": "Niu",
2337         "1": "Yes",
2338         "2": "No"
2339     }
2340 },
2341 "CHELSEW_YN": {
2342     "label": "Does this person have a child living outside the
                household?",
2343     "universe": "All persons aged 15+",
2344     "type": "Categorical",
2345     "role": "Independent",
2346     "topic": "Income",
2347     "subtopic": "Supplemental poverty measure",
2348     "values": {
2349         "0": "Niu",
2350         "1": "Yes",
2351         "2": "No"
2352     }
2353 },
2354 "CHELSEW_YN": {
2355     "label": "Does this person have a child living outside the
                household?",
2356     "universe": "All persons aged 15+",
2357     "type": "Categorical",

```

```

2358     "role": "Independent",
2359     "topic": "Income",
2360     "subtopic": "Supplemental poverty measure",
2361     "values": {
2362         "0": "Niu",
2363         "1": "Yes",
2364         "2": "No"
2365     }
2366 },
2367 "CHSP_VAL": {
2368     "label": "What is the annual amount of child support paid?",
2369     "universe": "CHSP_YN = 1",
2370     "type": "Continuous",
2371     "role": "Independent",
2372     "topic": "Income",
2373     "subtopic": "Supplemental poverty measure",
2374     "values": {
2375         "0": "Niu",
2376         "1-99999": "Amount paid in child support"
2377     }
2378 },
2379 "CHSP_YN": {
2380     "label": "Is this person required to pay child support?",
2381     "universe": "CHELSEW_YN",
2382     "type": "Categorical",
2383     "role": "Independent",
2384     "topic": "Income",
2385     "subtopic": "Supplemental poverty measure",
2386     "values": {
2387         "0": "Niu",
2388         "1": "Yes",
2389         "2": "No"
2390     }
2391 },
2392 "CSP_VAL": {
2393     "label": "How much did ... receive in child support payments?",

```

```

2394     "universe": "CHSP_YN = 1",
2395     "type": "Continuous",
2396     "role": "Independent",
2397     "topic": "Income",
2398     "subtopic": "Supplemental poverty measure",
2399     "values": {
2400         "0": "None or niu",
2401         "1-99999": "Child support"
2402     }
2403 },
2404 "CSP_YN": {
2405     "label": "Did ... receive child support payments?",
2406     "universe": "All Persons aged 15+",
2407     "type": "Categorical",
2408     "role": "Independent",
2409     "topic": "Income",
2410     "subtopic": "Supplemental poverty measure",
2411     "values": {
2412         "0": "Niu",
2413         "1": "Yes",
2414         "2": "No"
2415     }
2416 },
2417 "ACTC_CRD": {
2418     "label": "Additional child tax credit",
2419     "universe": "Tax unit head or dependent filer",
2420     "type": "Continuous",
2421     "role": "Independent",
2422     "topic": "Income",
2423     "subtopic": "Tax model items",
2424     "values": {
2425         "0": "None",
2426         "1-99999": "Dollar amount"
2427     }
2428 },
2429 "AGI": {

```

```

2430     "label": "Adjusted gross income",
2431     "universe": "Tax unit head or dependent filer",
2432     "type": "Continuous",
2433     "role": "Independent",
2434     "topic": "Income",
2435     "subtopic": "Tax model items",
2436     "values": {
2437         "0": "None",
2438         "-9999-9999999": "Dollar amount"
2439     }
2440 },
2441 "CTC_CRD": {
2442     "label": "Child tax credit",
2443     "universe": "Tax unit head or dependent filer",
2444     "type": "Continuous",
2445     "role": "Independent",
2446     "topic": "Income",
2447     "subtopic": "Tax model items",
2448     "values": {
2449         "0": "None",
2450         "1-99999": "Dollar amount"
2451     }
2452 },
2453 "EIT_CRED": {
2454     "label": "Earn income tax credit",
2455     "universe": "Tax unit head or dependent filer",
2456     "type": "Continuous",
2457     "role": "Independent",
2458     "topic": "Income",
2459     "subtopic": "Tax model items",
2460     "values": {
2461         "0": "None",
2462         "1-9999": "Dollar amount"
2463     }
2464 },
2465 "FED_RET": {

```

```

2466     "label": "Federal retirement payroll deduction",
2467     "universe": "Tax unit head or dependent filer",
2468     "type": "Continuous",
2469     "role": "Independent",
2470     "topic": "Income",
2471     "subtopic": "Tax model items",
2472     "values": {
2473         "0": "None",
2474         "1-999999": "Dollar amount"
2475     }
2476 },
2477 "FEDTAX_AC": {
2478     "label": "Federal income tax liability, after all credits",
2479     "universe": "Tax unit head or dependent filer",
2480     "type": "Continuous",
2481     "role": "Independent",
2482     "topic": "Income",
2483     "subtopic": "Tax model items",
2484     "values": {
2485         "0": "None",
2486         "-9999-99999999": "Dollar amount"
2487     }
2488 },
2489 "FEDTAX_BC": {
2490     "label": "Federal income tax liability, before credits",
2491     "universe": "Tax unit head or dependent filer",
2492     "type": "Continuous",
2493     "role": "Independent",
2494     "topic": "Income",
2495     "subtopic": "Tax model items",
2496     "values": {
2497         "0": "None",
2498         "-9999-99999999": "Dollar amount"
2499     }
2500 },
2501 "FICA": {

```

```

2502     "label": "Social security retirement payroll deduction",
2503     "universe": "All persons",
2504     "type": "Continuous",
2505     "role": "Independent",
2506     "topic": "Income",
2507     "subtopic": "Tax model items",
2508     "values": {
2509         "0": "None",
2510         "1-99999": "Dollar amount"
2511     }
2512 },
2513 "FILESTAT": {
2514     "label": "Tax filer status",
2515     "universe": "All persons",
2516     "type": "Categorical",
2517     "role": "Independent",
2518     "topic": "Income",
2519     "subtopic": "Tax model items",
2520     "values": {
2521         "1": "Joint, both<65",
2522         "2": "Joint, one >65 & one 65+",
2523         "3": "Joint, both 65+",
2524         "4": "Head of household",
2525         "5": "Single",
2526         "6": "Non-filer"
2527     }
2528 },
2529 "MARG_TAX": {
2530     "label": "Marginal tax rate",
2531     "universe": "Tax unit head or dependent filer",
2532     "type": "Continuous",
2533     "role": "Independent",
2534     "topic": "Income",
2535     "subtopic": "Tax model items",
2536     "values": {
2537         "0": "None",

```



```

2538         "1-99": "Marginal rate"
2539     }
2540 },
2541 "PRSWKXPNS": {
2542     "label": "Work expenses",
2543     "universe": "A_AGE > 17 or HHDFMX = 1,2,46, or 47",
2544     "type": "Continuous",
2545     "role": "Independent",
2546     "topic": "Income",
2547     "subtopic": "Tax model items",
2548     "values": {
2549         "0": "None",
2550         "1-1999": "Dollar amount"
2551     }
2552 },
2553 "STATETAX_A": {
2554     "label": "State income tax liability, after all credits",
2555     "universe": "Tax unit head or dependent filer",
2556     "type": "Continuous",
2557     "role": "Independent",
2558     "topic": "Income",
2559     "subtopic": "Tax model items",
2560     "values": {
2561         "0": "None",
2562         "-9999-9999999": "Dollar amount"
2563     }
2564 },
2565 "STATETAX_B": {
2566     "label": "State income tax liability, before credits",
2567     "universe": "Tax unit head or dependent filer",
2568     "type": "Continuous",
2569     "role": "Independent",
2570     "topic": "Income",
2571     "subtopic": "Tax model items",
2572     "values": {
2573         "0": "None",

```

```

2574         "-9999-99999999": "Dollar amount"
2575     }
2576 },
2577 "TAX_INC": {
2578     "label": "Taxable income amount",
2579     "universe": "Tax unit head or dependent filer",
2580     "type": "Continuous",
2581     "role": "Independent",
2582     "topic": "Income",
2583     "subtopic": "Tax model items",
2584     "values": {
2585         "0": "None",
2586         "-9999-99999999": "Dollar amount"
2587     }
2588 },
2589 "PERLIS": {
2590     "label": "Poverty level of persons (Subfamily members have primary
2591         family recode)",
2592     "universe": "All persons",
2593     "type": "Categorical",
2594     "role": "Independent",
2595     "topic": "Poverty",
2596     "subtopic": "Poverty",
2597     "values": {
2598         "-1": "Not in poverty universe",
2599         "1": "Below poverty level",
2600         "2": "100 - 124 percent of the poverty level",
2601         "3": "125 - 149 percent of the poverty level",
2602         "4": "150 and above the poverty level"
2603     }
2604 },
2605 "POV_UNIV": {
2606     "label": "Poverty universe flag",
2607     "universe": "All persons",
2608     "type": "Categorical",
2609     "role": "Independent",

```

```

2609     "topic": "Poverty",
2610     "subtopic": "Poverty",
2611     "values": {
2612         "0": "Not in poverty universe",
2613         "1": "In poverty universe"
2614     }
2615 },
2616 "HEA": {
2617     "label": "Health status",
2618     "universe": "All persons",
2619     "type": "Categorical",
2620     "role": "Independent",
2621     "topic": "Health insurance",
2622     "subtopic": "Health status",
2623     "values": {
2624         "1": "Excellent",
2625         "2": "Very good",
2626         "3": "Good",
2627         "4": "Fair",
2628         "5": "Poor"
2629     }
2630 },
2631 "SPM_ACTC": {
2632     "label": "SPM units Additional Child Tax Credit",
2633     "universe": "All persons",
2634     "type": "Continuous",
2635     "role": "Independent",
2636     "topic": "Supplemental poverty measure",
2637     "subtopic": "SPM unit characteristics",
2638     "values": {
2639         "0-99999": "Dollar amount"
2640     }
2641 }
2642 }

```

3.5.4 Python Modules

The utility module in Code 3.1 is for basic tasks such as creating a directory, backing up existing files before being overwritten, and importing and exporting a dictionary in JSON format. The encoding module in Code 3.2 is used solely during data encoding as its helper, not its main role. The dataset module in Code 3.3 helps importing and exporting dataset in both feather and CSV formats. The first employs LZ4 compression by default to bring a smaller file than the latter. The EDA module in Code 3.4 is primary for cross tabulation analysis. Its result is exported in CSV format, and its chart is saved in SVG, PGF and PDF formats.

Code 3.1: Utility module (module/utility.py)

```

1  import os
2  import time
3  import json
4
5  # Directory
6  def create_dir(dir):
7      try:
8          os.makedirs(dir)
9      except FileExistsError:
10         pass
11
12 # Backup
13 def backup_duplicate(file_dir, filename, format, backup_dir, info):
14     filepath = f"{file_dir}/{filename}.{format}"
15     date = time.strftime("%Y%m%d", time.localtime(time.time()))
16     if os.path.isfile(filepath):
17         backup_subdir = f"{backup_dir}/{date}/{file_dir.replace('../', '')}"
18         "
19         create_dir(backup_subdir)
20         filepath_backup = f"{backup_subdir}/{filename}-backup.{format}"
21         os.replace(filepath, filepath_backup)
22     if info:
23         print(f"{filepath} previously exists")
24         print(f"Back up to {filepath_backup}")

```

```

24     elif info:
25         print(f"{filepath} does not previously exists")
26
27 # Import/export dict/JSON
28 def import_dict(metadataapath):
29     with open(metadataapath) as myfile:
30         indep_contents = myfile.read()
31     return json.loads(indep_contents)
32
33 def export_json(dictfile, jsonfile):
34     with open(jsonfile, 'w', encoding='utf-8') as f:
35         json.dump(dictfile, f, ensure_ascii=False, indent=4)
36
37 def export_txt(string, txtfile):
38     f = open(txtfile, 'w')
39     f.write(string)
40     f.close()

```

Code 3.2: Encoding module (module/metaencode.py)

```

1  import pandas as pd
2
3  def extract_dict_cat(indep_dict):
4      return {attr: info for (attr, info) in indep_dict.items() if indep_dict
5                  [attr]['type'] == 'Categorical'}
6
7  def extract_dict_cont(indep_dict):
8      return {attr: info for (attr, info) in indep_dict.items() if indep_dict
9                  [attr]['type'] == 'Continuous'}
10
11 def sort_cols(df_indep, indep_dict):
12     sorted_cols = sorted(
13         df_indep.head(),
14         key=lambda attr: indep_dict[attr]['type'],
15         reverse=True
16     )

```

```

15     return df_indep[sorted_cols]
16
17 def indep_info(df_indep, indep_dict):
18     df_info = pd.DataFrame({'variable': df_indep.head().columns})
19     df_info['type'] = df_info['variable'].apply(lambda attr: indep_dict[
20         attr]['type'])
21     minmax = df_indep.agg(['min', 'max']).values.tolist()
22     df_info['min'] = minmax[0]
23     df_info['max'] = minmax[1]
24     del minmax
25     return df_info
26
27 def count_info(df_info):
28     df_count = df_info.groupby('type').count().reset_index()[['type', '
29         variable']]
30     df_count.rename(columns = {'variable': 'count'}, inplace=True)
31     df_count.sort_values('type', ascending=False, inplace=True,
32         ignore_index=True)
33     return df_count

```

Code 3.3: Dataset module (module/dataset.py)

```

1  import os
2  import urllib.request
3  import pandas as pd
4  import pyarrow
5
6  from module.utility import create_dir, backup_duplicate
7
8  # Import
9  def import_dataset(dataset_name, feather_dir, sas_dir='', sas_url=''):
10     filepath_feather = f"{feather_dir}/{dataset_name}.feather"
11
12     if os.path.isfile(filepath_feather):
13         print(f"{filepath_feather} is found")
14         print(f"{filepath_feather} was previously preprocessed")

```

```

15     df0 = pd.read_feather(filepath_feather)
16 else:
17     print(f"{filepath_feather} is not found")
18     if sas_dir == '':
19         raise Exception("SAS data directory is empty")
20     filepath_sas = f"sas_dir/{dataset_name}.sas7bdat"
21     if os.path.isfile(filepath_sas):
22         print(f"{filepath_sas} is found")
23     else:
24         print(f"{filepath_sas} is not found")
25         create_dir('original/data-orig')
26         print(f"{filepath_sas} will be downloaded")
27         print("Download starts")
28         try:
29             urllib.request.urlretrieve(sas_url, filepath_sas)
30             print("Download finishes")
31         except:
32             raise Exception("Download fails")
33         print(f"{filepath_sas} is successfully downloaded")
34     df0 = pd.read_sas(filepath_sas)
35
36     print(f"\nNumber of original data: {len(df0)}")
37     df0 = df0[df0['COV']!=0]
38     print(f"An infant born after calendar year (COV = 0) is excluded")
39     print(f"Number of training data: {len(df0)}")
40     return df0
41
42 # Export
43 def export_dataset(df, file_dir, dataset_name, format, info=True,
44                   backup_dir=''):
45     create_dir(file_dir)
46     if format == 'feather' or format == 'csv':
47         filepath = f"{file_dir}/{dataset_name}.{format}"
48         if backup_dir != '':
49             backup_duplicate(

```

```

50         format=format,
51         backup_dir=backup_dir, info=info
52     )
53     if format == 'feather':
54         df.to_feather(filepath)
55     else:
56         df.to_csv(filepath, index=False)
57     if info:
58         print(f"The dataframe is successfully exported to {filepath}")
59     else:
60         print(f"Input format {format} is unrecognized")

```

Code 3.4: EDA module (module/eda.py)

```

1  import sys
2  import time
3  import pandas as pd
4  import matplotlib.pyplot as plt
5
6  from module.utility import create_dir, backup_duplicate
7  from module.dataset import export_dataset
8
9  # Variables
10 def describe_var(var_dict, role='independent'):
11     num_cat = 0
12     num_cont = 0
13     for key in var_dict:
14         if var_dict[key]['type'] == 'Categorical':
15             num_cat += 1
16         else:
17             num_cont += 1
18     print(f"There are {num_cat + num_cont} {role} variables of interest: {
19         num_cat} categorical and {num_cont} continuous")
19
20 # Cross Tabulation Analysis

```



```

21 def crosstab(df, indep_dict, cont_bins, plot, output_dir, log_filepath,
    backup_dir=''):
22     dir_main = f"{output_dir}/tab-cbins-{cont_bins}"
23
24     for key, val in indep_dict.items():
25         fname_main = f"{key}-cbins-{cont_bins}"
26
27         if val['type'] == "Categorical":
28             crosstb = pd.crosstab(index=df[key].map(lambda x: val['values'
                ][str(x)]), columns=df['code'])
29         else:
30             dat = df[[key, 'code']].copy()
31             dat['bins'] = pd.cut(dat[key], bins=cont_bins)
32             crosstb = pd.crosstab(index=dat['bins'], columns=dat['code'])
33             del dat
34
35         print(key)
36         print(f"Label: {val['label']}")
37         print(f"Universe: {val['universe']}")
38         print(f"Type: {val['type']}")
39         print(f"Topic: {val['topic']}")
40         print(f"Subtopic: {val['subtopic']}")
41         print("\n")
42
43         print(f"Code: Employment-based plan (GRP) | Direct-purchase plan (
            DIR) | Public health insurance (PUB)")
44         print(crosstb)
45         '''
46         dir_crosstb = f"{dir_main}/cross-{cont_bins}"
47         create_dir(dir_crosstb)
48         export_dataset(
49             crosstb,
50             file_dir=f"{dir_crosstb}/feather", dataset_name=f"{fname_main}-
                cross",
51             format='feather', info=False,
52             backup_dir=backup_dir

```

```

53     )
54     export_dataset(
55         crosstb,
56         file_dir=f"{dir_crosstb}/csv", dataset_name=f"{fname_main}-
           cross",
57         format='csv', info=False,
58         backup_dir=backup_dir
59     )
60     '''
61     print("\n")
62
63     if plot:
64         barplot = crosstb.plot.bar()
65         barplot.legend(title='(GRP,DIR,PUB)',
66                        bbox_to_anchor=(1,1.02),
67                        loc='upper left')
68         plt.title(val['label'])
69         plt.xlabel(key)
70         plt.ylabel('Frequency')
71         ls_format = ['svg', 'pgf', 'pdf']
72         for format in ls_format:
73             dir_fig = f"{dir_main}/figures/{format}"
74             figname = f"{key}-cbins-{cont_bins}"
75             figpath = f"{dir_fig}/{figname}.{format}"
76             create_dir(dir_fig)
77             backup_duplicate(
78                 file_dir=dir_fig, filename=figname,
79                 format=format,
80                 backup_dir=backup_dir, info=False
81             )
82             f = open(log_filepath, 'a')
83             temp = sys.stdout
84             sys.stdout = f
85             count, tries = 0, 4
86             success = False
87             while count < tries:

```

```

88         try:
89             plt.savefig(figpath, bbox_inches='tight')
90             success = True
91             break
92         except:
93             pass
94         count += 1
95     if not success:
96         curtime = time.strftime("%Y-%m-%d %H:%M:%S", time.
97                                 localtime(time.time()))
98         print(f"{curtime} | {key}: {figpath} cannot be saved")
99         sys.stdout = temp
100        f.close()
101    #plt.show()
102
103    dftb = crosstb.reset_index().rename_axis(None, axis=1)
104    dftb[dftb.columns[1:]] = dftb[dftb.columns[1:]].astype('uint32')
105    export_dataset(
106        dftb,
107        file_dir=f"{dir_main}/feather", dataset_name=fname_main,
108        format='feather', info=False,
109        backup_dir=backup_dir
110    )
111    export_dataset(
112        dftb,
113        file_dir=f"{dir_main}/csv", dataset_name=fname_main,
114        format='csv', info=False,
115        backup_dir=backup_dir
116    )
117    print("\n-----")

```

3.5.5 Python Classes

Pandas DataFrame is a two-dimensional columnwise data structure. Each column must have the same data type. Although it provides by default rich functionality for data manipulation, additional namespaces can be added to pandas objects by registering custom accessors to serve specific purposes. Health insurance dataset in SAS7BDAT file format is imported as a Pandas DataFrame. All columns are numerical, either `int64` or `float64`.

With the `thesis` namespace (Code 3.5), the data type of a column can be of smaller size through the `retype` method, three dependent variables of interest (GRP, DIR and PUB) can be coded to a string of three character literals, either Y (Yes) or N (No), by the `code` method, and these eight different codes are regrouped to five with numerical values assigned by the `recode` method. Since some categorical values do not start from 0 up to a positive integer as required by the box classifier proposed in Chapter 4, they are encoded to be in this format via the `data` namespace (Code 3.6). Any numerical flags representing a continuous NIU (not in universe) value are converted to zero to become more meaningful. A categorical NIU value is already changed by the previous reordering. The `info` namespace (Code 3.7) sets the number of splitting values or cuts as given on a feature appropriately, not exceeding the number of all possible values for a categorical feature.

Code 3.5: ThesisExtension class (cls/ThesisExtension.py)

```

1  import re
2  import pandas as pd
3
4  @pd.api.extensions.register_dataframe_accessor("thesis")
5  class ThesisExtension:
6      def __init__(self, pandas_obj):
7          #self._validate(pandas_obj, list(indep_dict.keys()) + ['COV'] +
8              dep_attrs)
9          self.dataset = pandas_obj
10
11      '''
12      @staticmethod
13      def _validate(obj, cols):
14          if any(x not in obj.columns for x in cols):
15              raise AttributeError("Some attributes are missing")

```

```

15     '''
16
17     def select(self, cols):
18         self.dataset.drop(self.dataset.columns.difference(cols), axis=1,
19                             inplace=True)
19
20     def show_type(self, option='short'):
21         if option.lower() == 'full':
22             with pd.option_context('display.max_rows', None, 'display.
23                                     max_columns', None):
24                 print(self.dataset.dtypes)
25         else:
26             print(self.dataset.dtypes)
26
27     @staticmethod
28     def retype(ser):
29         if all(ser.apply(lambda x: isinstance(x, int))):
30             flag_int = True
31         elif all(ser.apply(lambda x: x.is_integer())):
32             flag_int = True
33         else:
34             flag_int = False
35
36         if flag_int:
37             if all(ser.apply(lambda x: x>=0)):
38                 if max(ser) <= 255:
39                     return ser.astype('uint8')
40                 elif max(ser) <= 65535:
41                     return ser.astype('uint16')
42                 else:
43                     return ser.astype('uint32')
44             else:
45                 if min(ser) >= -128 and max(ser) <= 127:
46                     return ser.astype('int8')
47                 elif min(ser) >= -32768 and max(ser) <= 32767:
48                     return ser.astype('int16')

```

```

49         else:
50             return ser.astype('int32')
51     else:
52         return ser.astype('float32')
53
54     def code(self, indep_dict, dep_attrs):
55         self.select(list(indep_dict.keys()) + ['COV'] + dep_attrs)
56         for v in indep_dict.keys():
57             if indep_dict[v]['type'] == 'Categorical':
58                 self.dataset[v] = self.dataset[v].astype('int8').astype('
                    category')
59             else:
60                 self.dataset[v] = self.retype(self.dataset[v])
61         self.dataset['COV'] = self.dataset['COV'].astype('int8').astype('
            category')
62         self.dataset[dep_attrs] = self.dataset[dep_attrs].astype('int8')
63         self.dataset['class_orig'] = 0
64         self.dataset['code_orig'] = ""
65         for v in dep_attrs:
66             self.dataset[v] = self.dataset[v].replace([2.0, 1.0], [False,
                True])
67             self.dataset['class_orig'] = 2*self.dataset['class_orig'] +
                self.dataset[v]
68             self.dataset['code_orig'] = self.dataset['code_orig'] + self.
                dataset[v].replace([True, False], ['Y', 'N'])
69         self.dataset[dep_attrs] = self.dataset[dep_attrs].astype('category'
            )
70         self.dataset['class_orig'] = self.dataset['class_orig'].astype('
            int8').astype('category')
71         self.dataset['code_orig'] = self.dataset['code_orig'].astype('
            category')
72
73     def recode(self):
74         self.dataset['code'] = self.dataset['code_orig'].apply(
75             lambda v: 'NY_' if re.match('(NY)', v)

```

```

76         else 'Y1Y' if re.match(r'^Y(?:\w*Y)', v) # Raw string to
           prevent invalid escape sequence '\w'
77         else v
78     ).astype('category')
79     self.dataset['class'] = self.dataset[['class_orig', 'code']].apply(
80         lambda v: 2 if v['code'] == 'NY_'
81         else 3 if v['code'] == 'YNN'
82         else 4 if v['code'] == 'Y1Y'
83         else v['class_orig'],
84         axis=1
85     ).astype('int8').astype('category')

```

Code 3.6: Data class (cls/Data.py)

```

1  import re
2  import pandas as pd
3  from sklearn.preprocessing import LabelEncoder
4
5  @pd.api.extensions.register_dataframe_accessor("data")
6  class Data:
7      def __init__(self, pandas_obj, indep_dict):
8          self.dataset = pandas_obj
9          self.metadata = indep_dict
10
11      def encodecat(self):
12          cat_change = ""
13          for attr in self.metadata.keys():
14              if self.metadata[attr]['type'] == 'Categorical':
15                  le = LabelEncoder()
16                  le.fit(self.dataset[attr])
17                  self.dataset[attr] = list(le.transform(self.dataset[attr]).
                                           astype('int8'))
18                  newkeys = list()
19                  unseen = 0
20                  for strval in self.metadata[attr]['values'].keys():
21                      try:

```

```

22         newkeys.append(int(le.transform([int(strval)])))
23     except ValueError: # for previously unseen labels
24         unseen -= 1
25         newkeys.append(unseen)
26     if list(self.metadata[attr]['values'].keys()) != newkeys:
27         cat_change += attr+"\n"
28         newdict = {key: val for key, val in zip(newkeys, self.
29             metadata[attr]['values'].values())}
30         self.metadata[attr]['values'] = newdict
31     return cat_change[0:-1]
32
33 def encodecont(self):
34     pattern = r'(^|^[^\\w])(niu|universe)([^\\w]|$)' # Raw string to
35         prevent invalid escape sequence '\\w'
36     pattern = re.compile(pattern, re.IGNORECASE)
37     cont_nonpos = ""
38     for attr in self.metadata.keys():
39         if self.metadata[attr]['type'] == 'Continuous':
40             flag = False
41             for strval in self.metadata[attr]['values'].keys():
42                 if not flag:
43                     try:
44                         if int(strval) <= 0:
45                             text = self.metadata[attr]['values'][strval]
46                             matches = re.search(pattern, text.replace(',',
47                                 ' ').lower())
48                             if bool(matches):
49                                 flag = True
50                                 cont_nonpos += attr+"\n"
51                                 self.dataset[attr] = self.dataset[attr].
52                                     apply(lambda v: 0 if v < 0 else v)
53                                 break
54                     except:
55                         pass
56             if flag:
57                 try:

```



```

54             if int(strval) <= 0:
55                 self.metadata[attr]['values'].pop(strval,
56                                                     None)
57
58             except:
59                 pass
60
61             if flag:
62                 self.metadata[attr]['values']['0'] = 'NIU'
63
64     return cont_nonpos[0:-1]

```

Code 3.7: Info class (cls/Info.py)

```

1  import pandas as pd
2
3  # Delete the accessor to avoid warning
4  try:
5      del pd.DataFrame.info
6  except AttributeError:
7      pass
8
9  @pd.api.extensions.register_dataframe_accessor("info")
10 class Info:
11     def __init__(self, pandas_obj):
12         self._validate(pandas_obj, ['id', 'variable', 'type', 'min', 'max'
13                                     ])
14
15         self.dataset = pandas_obj
16
17     @staticmethod
18     def _validate(obj, cols):
19         if any(x not in obj.columns for x in cols):
20             raise AttributeError("Some attributes are missing")
21
22     def setcut(self, pcont, pcatmax):
23         self.dataset['cut'] = 0
24         self.dataset.loc[self.dataset['type'] == 'Continuous', 'cut'] =
25             pcont

```

```

23         self.dataset.loc[self.dataset['type'] == 'Categorical', 'cut'] =
            self.dataset['max'].map(lambda v: min(v, pcatmax))

```

3.5.6 Exploratory Data Analysis (EDA)

This dissertation considers health insurance factors from a range of topics and subtopics as shown in Table 3.5. All infants born after calendar year are excluded in this study because they are not in the scope of health insurance coverage. This results in 157,681 relevant survey participants. Code 3.8 performs exploratory data analysis by using the pandas accessor `thesis` in Code 3.5 to compute the cross tabulation between a health factor (independent variable) and a combination of categorical insurance coverage types (dependent variable) as illustrated in Table 3.6. All continuous values of an independent variables are segmented into 10 bins. In addition, it can significantly compress the original dataset of size 237.4 MB in SAS7BDAT format into the feather and CSV formats of size 14.2 MB and 68.1 MB respectively.

Table 3.5: Categories of health insurance factors

Topic	Subtopic	List of Variables
Demographics	Individual characteristics	A_AGE, A_EXPRRP, A_FAMTYP, A_HGA, A_MARITL,
		A_PFREL, A_SEX, P_STAT, PEAFEVER, PEDISDRS, PEDISEAR, PEDISEYE, PEDISOUT, PEDISPHY, PEDISREM, PRDISFLG, PRCITSH, PRDTRACE
Basic CPS items	Edited labor force items	A_MJIND, A_MJOCC, PEIO1COW, PRDISC, PRUNTYPE
	Edited earnings items	A_GRSWK, A_HRLYWK, A_HRSPAY, PRERELG
	Labor force person recodes	A_CIVLF, A_CLSWKR, A_EXPLF, A_LFSR, A_UNCOV, A_UNMEM, A_UNTYPE, A_USLHRS, A_WKSCH, A_WKSLK, A_WKSTAT, PEHRUSLT, PEMLR, PRCOW1, PRPTREA, PRWKSTAT
Work experience	General	CLWK, EARNER, HRSWK, LJCW, NWLKWK, NWLOOK, PHMEMPRS, RSNNOTW, WECLW, WEWKRS, WKSWORK, WORKYN, WRK_CK, WTEMP

Table 3.5: Categories of health insurance factors (continued)

Topic	Subtopic	List of Variables
Income	Earnings	ERN_OTR, ERN_SRCE, ERN_VAL, ERN_YN, FRM_VAL,
		FRMOTR, FRSE_VAL, FRSE_YN, PEARNVAL, SE_VAL,
		SEMP_VAL, SEMP_YN, SEOTR, WAGEOTR, WS_VAL,
		WSAL_VAL, WSAL_YN

Table 3.5: Categories of health insurance factors (continued)

Topic	Subtopic	List of Variables
	Other income	ANN_VAL, ANN_YN, CAP_VAL, CAP_YN, DBTN_VAL, DIS_SC1, DIS_SC2, DIS_VAL1, DIS_VAL2, DIS_YN, DIV_VAL, DIV_YN, DSAB_VAL, DST_SC1, DST_SC1_YNG, DST_SC2, DST_SC2_YNG, DST_VAL1, DST_VAL1_YNG, DST_VAL2, DST_VAL2_YNG, DST_YN, DST_YN_YNG, ED_VAL, ED_YN, FIN_VAL, FIN_YN, INT_VAL, INT_YN, OED_TYP1, OED_TYP2, OED_TYP3, OI_OFF, OI_VAL, OI_YN, PEN_SC1, PEN_SC2, PEN_VAL1, PEN_VAL2, PEN_YN, PNSN_VAL, PTOTVAL, RESNSS1, RESNSS2, RESNSSI1, RESNSSI2, RETCB_VAL, RETCB_YN, RINT_SC1, RINT_SC2, RINT_VAL1, RINT_VAL2, RINT_YN, RNT_VAL, RNT_YN, SRVS_VAL, SS_VAL, SS_YN, SSI_VAL, SSI_YN, STRKUC, SUBUC, SUR_SC1, SUR_SC2, SUR_VAL1, SUR_VAL2, SUR_YN, TRDINT_VAL, UC_VAL, UC_YN, VET_TYP1, VET_TYP2, VET_TYP3, VET_TYP4, VET_TYP5, VET_VAL, VET_YN, WC_TYPE, WC_VAL, WC_YN

Table 3.5: Categories of health insurance factors (continued)

Topic	Subtopic	List of Variables
	Non-cash benefits	PAW_TYP, PAW_VAL, PAW_YN, PENINCL, PENPLAN, WICYN
	Supplemental poverty measure	CHCARE_YN, CHELSEW_YN, CHSP_VAL, CHSP_YN, CSP_VAL, CSP_YN
	Tax model items	ACTC_CRD, AGI, CTC_CRD, EIT_CRED, FED_RET, FEDTAX_AC, FEDTAX_BC, FICA, FILESTAT, MARG_TAX, PRSWKXPNS, STATETAX_A, STATETAX_B, TAX_INC
Poverty	Poverty	PERLIS, POV_UNIV
Health insurance	Health status	HEA
Supplemental poverty measure	SPM unit characteristics	SPM_ACTC

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
A_AGE: Age					
Universe: All Persons					
(-0.085, 8.5]	1,407	5,834	789	628	9,795
(8.5, 17.0]	1,557	6,237	1,079	770	11,822
(17.0, 25.5]	2,238	2,475	1,043	414	8,017
(25.5, 34.0]	2,635	2,749	1,082	594	10,611
(34.0, 42.5]	2,271	2,146	976	613	11,509
(42.5, 51.0]	2,109	2,171	1,157	518	12,081
(51.0, 59.5]	1,606	2,403	1,223	471	9,864
(59.5, 68.0]	1,028	4,854	2,313	2,090	6,097
(68.0, 76.5]	105	5,404	2,602	2,044	254
(76.5, 85.0]	79	4,472	1,977	1,353	115

A_EXPRRP: Expanded relationship code

Universe: All Persons

Reference person with relatives

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Reference person without relatives	1,603	6,102	2,739	1,413	7,066
Husband	1,049	2,196	1,325	1,016	7,069
Wife	1,482	2,898	1,984	1,426	10,471
Own child	4,337	12,355	2,540	1,553	27,291
Grandchild	377	1,621	137	106	940
Parent	335	1,183	305	174	780
Brother/sister	352	636	127	50	680
Other relative	464	1,219	215	106	908
Foster child	2	107	2	44	2
Nonrelative with relatives	305	514	101	73	816
Partner/roommate	803	780	421	149	2,381
Nonrelative without relatives	233	312	91	20	358

A_FAMTYP: Family type

Universe: All Persons

Primary family

11,310 28,667 10,560 7,564 67,373

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Nonfamily householder	1,603	6,102	2,739	1,413	7,066
Related subfamily	779	2,263	327	232	2,169
Unrelated subfamily	59	175	32	29	223
Secondary individual	1,284	1,538	583	257	3,334
A_HGA: Educational attainment					
Universe: All Persons					
Children	2,431	10,167	1,488	1,160	17,629
Less than 1st grade	76	177	31	19	64
1st,2nd,3rd,or 4th grade	170	390	61	21	115
5th or 6th grade	412	666	105	52	283
7th and 8th grade	418	1,035	222	116	794
9th grade	480	1,208	231	126	1,381
10th grade	459	1,363	252	169	1,694
11th grade	495	1,443	307	172	1,814
12th grade no diploma	339	716	159	94	794

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
High school graduate - high school diploma or equivalent	4,267	9,614	3,563	2,174	13,304
Some college but no degree	2,177	4,642	2,282	1,357	10,203
Associate degree in college - occupation/vocation program	465	1,044	589	370	2,681
Associate degree in college - academic program	610	1,260	719	513	3,919
Bachelor's degree (for example: BA,AB,BS)	1,580	3,364	2,738	1,731	15,745
Master's degree (for example: MA,MS,MENG,MED,MSW, MBA)	530	1,221	1,041	1,017	7,264
Professional school degree (for example: MD,DDS,DVM,LLB,JD)	52	189	202	162	1,026
Doctorate degree (for example: PHD,EDD)	74	246	251	242	1,455
A_MARITL: Marital status					
Universe: All Persons					
Married - civilian spouse present	4,911	11,026	6,899	5,333	35,669
Married - AF spouse present	346	11	9	0	86
Married - spouse absent (exc.separated)	261	418	175	97	721
Widowed	282	3,671	1,344	784	741
Divorced	1,186	3,834	1,402	754	4,817

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Separated	356	723	144	101	758
Never married	7,693	19,062	4,268	2,426	37,373
A_PFREL: Primary family relationship					
Universe: All Persons					
Not in primary family	2,946	7,815	3,354	1,699	10,623
Husband	2,408	5,385	3,324	2,794	16,972
Wife	2,501	4,998	3,382	2,404	17,664
Own child	4,337	12,355	2,540	1,553	27,291
Other relative	1,528	4,659	784	436	3,308
Unmarried reference person	1,315	3,533	857	609	4,307
A_SEX: Sex					
Universe: All Persons					
Male	7,804	17,947	6,658	4,710	39,664
Female	7,231	20,798	7,583	4,785	40,501
P_STAT: Status of person identifier					

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Universe: All Persons					
Civilian 15+	12,186	28,562	12,747	8,334	62,431
Armed forces	418	16	6	1	105
Children 0-14	2,431	10,167	1,488	1,160	17,629
PEAFEVER: Did you ever serve on active duty in the U.S. Armed Forces?					
Universe: A_AGE greater than or equal to 17					
Not in universe	3,207	11,462	1,745	1,320	20,376
Yes	674	3,025	1,158	1,233	2,498
No	11,154	24,258	11,338	6,942	57,291
PEDISDRS: Does...have difficulty dressing or bathing?					
Universe: PRPERTYP = 2					
Not in universe	2,849	10,183	1,494	1,161	17,734
Yes	98	1,545	299	233	224
No	12,088	27,017	12,448	8,101	62,207
PEDISEAR: Is...deaf or does ...have serious difficulty hearing?					

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Universe: PRPERTYP = 2					
Not in universe	2,849	10,183	1,494	1,161	17,734
Yes	153	2,024	809	573	683
No	12,033	26,538	11,938	7,761	61,748
PEDISEYE: Is...blind or does...have serious difficulty seeing even when wearing glasses?					
Universe: PRPERTYP = 2					
Not in universe	2,849	10,183	1,494	1,161	17,734
Yes	110	1,116	280	202	358
No	12,076	27,446	12,467	8,132	62,073
PEDISOUT: Because of a physical, mental, or emotional condition, does...have difficulty doing errands along such as visiting a doctor's office or shopping?					
Universe: PRPERTYP = 2					
Not in universe	2,849	10,183	1,494	1,161	17,734
Yes	223	3,156	638	513	506

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
No	11,963	25,406	12,109	7,821	61,925
PEDISPHY: Does...have serious difficulty Walking or climbing stairs?					
Universe: PRPERTYP = 2					
Not in universe	2,849	10,183	1,494	1,161	17,734
Yes	339	4,767	1,210	900	933
No	11,847	23,795	11,537	7,434	61,498
PEDISREM: Because of a physical, mental, or emotional condition, does...have serious difficulty concentrating, remembering, or making decisions?					
Universe: PRPERTYP = 2					
Not in universe	2,849	10,183	1,494	1,161	17,734
Yes	292	2,489	519	367	762
No	11,894	26,073	12,228	7,967	61,669
PRDISFLG: Does this person have any of these disability conditions?					
Universe: PRPERTYP = 2					
Not in universe	2,849	10,183	1,494	1,161	17,734

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Yes	732	7,560	2,124	1,569	2,395
No	11,454	21,002	10,623	6,765	60,036
PRCITSH: Citizenship group					
Universe: All persons					
Native, born in US	11,006	32,887	12,065	8,403	70,326
Native, born in PR or US outlying area	82	345	60	49	326
Native, born abroad of US parent(s)	153	249	92	76	694
Foreign born, US cit by naturalization	1,004	2,975	1,067	650	4,851
Foreign born, not a US citizen	2,790	2,289	957	317	3,968
PRDTRACE: Race					
Universe: All persons					
White only	11,466	27,682	11,885	7,517	63,366
Black only	1,765	6,815	1,011	1,051	7,484
American Indian, Alaskan Native only (AI)	516	902	97	85	837
Asian only	745	2,010	962	561	5,947

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Hawaiian/Pacific Islander only (HP)	89	192	34	41	368
White-Black	150	428	70	58	600
White-AI	131	351	81	96	490
White-Asian	86	111	52	41	613
White-HP	17	50	15	13	112
Black-AI	26	67	5	12	58
Black-Asian	2	8	9	3	45
Black-HP	1	8	1	4	1
AI-Asian	2	6	1	0	6
AI-HP	0	4	0	0	4
Asian-HP	5	17	12	7	72
White-Black-AI	13	44	2	3	32
White-Black-Asian	12	8	0	1	34
White-Black-HP	0	1	0	0	5
White-AI-Asian	2	3	0	0	7
White-AI-HP	0	3	0	0	4

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
White-Asian-HP	4	35	1	2	65
Black-AI-Asian	1	0	0	0	1
White-Black-AI-Asian	0	0	2	0	5
Other 3 race comb.	1	0	0	0	3
Other 4 or 5 race comb.	1	0	1	0	6
A_MJIND: Major industry code					
Universe: A_CLSWKR = 1-7					
Not in universe, or children	6,704	30,326	8,393	5,873	29,260
Agriculture, forestry,fishing, and hunting	268	241	309	79	536
Mining	44	21	24	18	445
Construction	1,114	670	511	214	2,961
Manufacturing	551	501	331	346	5,528
Wholesale and retail trade	1,124	1,336	770	433	5,857
Transportation and utilities	480	474	276	185	2,865
Information	80	117	93	48	978

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Financial activities	310	336	437	233	3,752
Professional and business services	957	926	813	414	6,036
Educational and health services	1,209	1,607	1,088	957	13,296
Leisure and hospitality	1,346	1,367	629	278	3,561
Other services	589	615	457	185	1,854
Public administration	250	208	110	232	3,236
Armed forces	9	0	0	0	0
A_MJOCC: Major occupation recode					
Universe: A_CLSWKR = 1-7					
Not in universe or children	6,704	30,326	8,393	5,873	29,260
Management, business, and financial occupations	866	821	1,144	595	9,953
Professional and related occupations	964	1,023	1,142	951	14,527
Service occupations	2,265	2,597	1,125	547	6,665
Sales and related occupations	791	1,025	687	311	4,343
Office and administrative support occupations	661	797	589	423	5,469

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Farming, fishing, and forestry occupations	177	123	95	31	246
Construction and extraction occupations	948	536	326	160	2,154
Installation, maintenance, and repair occupations	327	215	129	127	1,622
Production occupations	484	417	228	194	2,728
Transportation and material moving occupations	839	865	383	283	3,198
Armed forces	9	0	0	0	0
PEIO1COW: Individual class of worker on first job					
Universe: All persons					
NIU	6,704	30,326	8,393	5,873	29,260
Government-federal	222	120	57	138	1,708
Government-state	189	237	151	213	3,210
Government - local	219	337	196	296	4,045
Private, for profit	6,214	5,951	3,369	2,233	34,815
Private, nonprofit	274	466	323	343	3,933
Self-employed, incorporated	325	323	756	152	1,484

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Self-employed, unincorporated	880	974	986	246	1,703
Without pay	8	11	10	1	7
PRDISC: Discouraged worker recode					
Universe: All persons					
NIU	14,880	38,437	14,165	9,452	79,861
Discouraged worker	40	83	18	4	57
Conditionally interested	73	159	34	28	145
Not available	42	66	24	11	102
PRUNTYPE: Individual class of worker on first job					
Universe: All persons					
NIU	14,304	37,763	13,967	9,302	78,459
Job loser /on layoff	252	341	136	72	797
Other job loser	127	130	38	52	329
Temporary job ended	82	97	17	14	93
Job leaver	69	64	14	11	138

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Re-entrant	162	266	62	38	275
New-entrant	39	84	7	6	74
<p>A_GRSWK: How much does ... usually earn per week at this job before deductions , subject to topcoding, the higher of either the amount of item 25a times Item 25c or the actual item 25d entry will be present</p> <p>Universe: PRERELG=1</p>					
(-2.885, 288.5]	14,066	37,929	13,596	9,036	72,547
(288.5, 577.0]	412	407	218	112	1,185
(577.0, 865.5]	285	213	159	122	1,652
(865.5, 1154.0]	111	88	102	92	1,522
(1154.0, 1442.5]	64	47	42	36	979
(1442.5, 1731.0]	34	18	33	27	714
(1731.0, 2019.5]	21	15	20	16	413
(2019.5, 2308.0]	10	9	15	9	314
(2308.0, 2596.5]	13	6	20	9	201

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(2596.5, 2885.0]	19	13	36	36	638
A_HRLYWK: Is ... paid by the hour on this job?					
Universe: PRRELG=1					
Not in universe or children and armed forces	13,245	37,057	13,165	8,715	67,548
Yes	1,320	1,289	662	468	6,463
No	470	399	414	312	6,154
A_HRSPAY: How much does ... earn per hour?					
Universe: A_HRLYWK=1					
(-10.901, 989.1]	14,314	38,046	13,813	9,201	76,286
(989.1, 1979.2]	563	582	312	203	2,116
(1979.2, 2969.3]	112	80	69	58	1,059
(2969.3, 3959.4]	28	24	20	19	391
(3959.4, 4949.5]	10	6	12	5	165
(4949.5, 5939.6]	5	4	10	6	76
(5939.6, 6929.7]	3	1	2	2	40

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(6929.7, 7919.8]	0	1	1	1	21
(7919.8, 8909.9]	0	0	0	0	7
(8909.9, 9900.0]	0	1	2	0	4
PRERELG: Earnings eligibility flag					
Universe: All persons					
Not earnings eligible	13,245	37,057	13,165	8,715	67,548
Earnings eligible	1,790	1,688	1,076	780	12,617
A_CIVLF: Civilian labor force					
Universe: All persons					
Not in universe or children and Armed Forces	6,798	30,466	8,496	5,960	29,588
In universe	8,237	8,279	5,745	3,535	50,577
A_CLSWKR: Class of worker					
Universe: PEMLR=1-3 or (PEMLR=4-7 and person worked in the last 12 months)					
Not in universe or children and armed forces	6,665	30,242	8,386	5,867	29,186

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Private	6,488	6,417	3,692	2,576	38,748
Federal government	222	120	57	138	1,708
State government	189	237	151	213	3,210
Local government	219	337	196	296	4,045
Self-employed-incorporated	325	323	756	152	1,484
Self-employed-not incorporated	880	974	986	246	1,703
Without pay	8	11	10	1	7
Never worked	39	84	7	6	74
A_EXPLF: Experienced labor force employment status					
Universe: PEMLR=1-4					
Not in experienced labor force	6,837	30,550	8,503	5,966	29,662
Employed	7,506	7,297	5,471	3,342	48,871
Unemployed	692	898	267	187	1,632
A_LFSR: Labor force status recode					
Universe: All persons					

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Children or Armed Forces	2,849	10,183	1,494	1,161	17,734
Working	7,178	6,826	5,136	3,181	46,957
With job, not at work	328	471	335	161	1,914
Unemployed, looking for work	479	641	138	121	909
Unemployed, on layoff	252	341	136	72	797
Nilf	3,949	20,283	7,002	4,799	11,854
A_UNCOV: On this job, is ... covered by a union or employee association contract?					
Universe: A_UNMEM=2					
Not in universe or children and armed forces	13,962	37,715	13,483	9,016	72,936
Yes	8	11	8	10	108
No	1,065	1,019	750	469	7,121
A_UNMEM: On this job, is ... a member of a labor union or of an employee association similar to a union?					
Universe: PRERELG=1					

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Not in universe or children and armed forces	13,909	37,669	13,451	8,957	71,925
Yes	53	46	32	59	1,011
No	1,073	1,030	758	479	7,229
A_UNTYPE: Reason for unemployment					
Universe: A_LFSR=3 or 4					
Not in universe or children and Armed Forces	14,304	37,763	13,967	9,302	78,459
Job loser - on layoff	252	341	136	72	797
Other job loser	209	227	55	66	422
Job leaver	69	64	14	11	138
Re-entrant	162	266	62	38	275
New entrant	39	84	7	6	74
A_USLHRS: How many hrs per week does ... usually work at this job?					
Universe: All persons					
(-4.103, 6.3]	8,214	32,313	9,452	6,448	33,848
(6.3, 16.6]	279	647	359	198	1,392

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(16.6, 26.9]	641	1,071	691	288	2,360
(26.9, 37.2]	935	1,099	622	362	3,750
(37.2, 47.5]	4,268	3,105	2,411	1,848	32,501
(47.5, 57.8]	436	291	412	234	4,378
(57.8, 68.1]	186	149	189	74	1,437
(68.1, 78.4]	45	46	57	22	289
(78.4, 88.7]	24	13	28	16	166
(88.7, 99.0]	7	11	20	5	44
A_WKSCH: Labor force by time worked or lost					
Universe: All persons					
Not in universe	6,798	30,466	8,496	5,960	29,588
At work	7,178	6,826	5,136	3,181	46,957
With job, not at work	328	471	335	161	1,914
Unemployed, seeks FT	618	722	197	136	1,316
Unemployed, seeks PT	113	260	77	57	390

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
A_WKSLK: Duration of unemployment					
Universe: PEMLR=3 or 4					
(-0.099, 9.9]	14,748	38,340	14,142	9,435	79,643
(9.9, 19.8]	118	150	44	27	237
(19.8, 29.7]	49	76	17	12	121
(29.7, 39.6]	26	50	9	7	66
(39.6, 49.5]	10	11	4	4	16
(49.5, 59.4]	45	50	11	5	42
(59.4, 69.3]	9	10	3	0	7
(69.3, 79.2]	4	2	0	0	1
(79.2, 89.1]	0	0	0	0	1
(89.1, 99.0]	26	56	11	5	31
A_WKSTAT: Full/part-time status					
Universe: All persons					
Children or Armed Forces	2,849	10,183	1,494	1,161	17,734

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Not in labor force	3,949	20,283	7,002	4,799	11,854
Full-time schedules	5,715	4,390	3,714	2,508	42,413
Part-time for economic reasons, usually FT	267	217	153	48	670
Part-time for non-economic reasons, usually PT	1,200	2,313	1,464	718	5,257
Part-time for economic reasons, usually PT	324	377	140	68	531
Unemployed FT	618	722	197	136	1,316
Unemployed PT	113	260	77	57	390
PEHRUSLT: Hours usually worked last week					
Universe: All persons					
(-4.144, 10.4]	8,336	32,561	9,610	6,541	34,614
(10.4, 24.8]	595	1,159	671	330	2,447
(24.8, 39.2]	1,147	1,420	805	444	4,613
(39.2, 53.6]	4,519	3,253	2,721	1,976	35,068
(53.6, 68.0]	333	257	306	147	2,691
(68.0, 82.4]	87	76	102	42	583

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(82.4, 96.8]	14	7	12	8	106
(96.8, 111.2]	4	11	13	7	36
(111.2, 125.6]	0	0	1	0	7
(125.6, 140.0]	0	1	0	0	0
PEMLR: Major labor force recode					
Universe: All persons					
NIU	2,849	10,183	1,494	1,161	17,734
Employed - at work	7,178	6,826	5,136	3,181	46,957
Employed - absent	328	471	335	161	1,914
Unemployed - on layoff	252	341	136	72	797
Unemployed - looking	479	641	138	121	909
Not in labor force - retired	543	11,004	5,087	3,754	1,768
Not in labor force - disabled	437	4,110	405	359	732
Not in labor force - other	2,969	5,169	1,510	686	9,354
PRCOW1: Class of worker recode-job 1					

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Universe: All persons					
NIU	6,704	30,326	8,393	5,873	29,260
Federal govt	222	120	57	138	1,708
State govt	189	237	151	213	3,210
Local govt	219	337	196	296	4,045
Private (incl. self-employed incorp.)	6,813	6,740	4,448	2,728	40,232
Self-employed, unincorp.	880	974	986	246	1,703
Without pay	8	11	10	1	7
PRPTREA: Detailed reason for part-time					
Universe: Part time workers					
NIU	12,873	35,620	12,343	8,513	71,585
Usually FT - slack work/business conditions	248	202	136	45	634
Usually FT - seasonal work	13	6	14	1	17
Usually FT - job started/ended during week	6	9	3	2	19
Usually FT - vacation/personal day	90	87	60	57	970

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Usually FT - own illness/injury/medical appt	109	124	79	57	669
Usually FT - holiday (religious or legal)	5	7	3	4	40
Usually FT - child care problems	4	5	7	7	52
Usually FT - other fam/pers obligations	32	25	20	17	206
Usually FT - labor dispute	2	1	0	0	4
Usually FT - weather affected job	70	30	10	5	70
Usually FT - school/training	5	5	1	0	18
Usually FT - civic/military duty	0	1	0	0	4
Usually FT - other reason	119	116	74	44	446
Usually PT - slack work/business conditions	206	223	95	40	345
Usually PT - PT could only find PT work	133	177	61	30	233
Usually PT - seasonal work	12	7	5	2	12
Usually PT - child care problems	64	116	40	16	236
Usually PT - other fam/pers obligations	271	343	248	111	1,221
Usually PT - health/medical limitations	51	199	54	44	123
Usually PT - school/training	303	450	245	98	1,713

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Usually PT - retired/social security limit on earnings	52	440	350	238	228
Usually PT - workweek<35 hours	260	407	251	106	952
Usually PT - other	107	145	142	58	368
PRWKSTAT: Full/part-time work status					
Universe: All persons					
NIU	2,849	10,183	1,494	1,161	17,734
Not in labor force	3,949	20,283	7,002	4,799	11,854
FT hours (35+), usually FT	4,995	3,679	3,226	2,189	38,324
PT for economic reasons, usually FT	267	217	153	48	670
PT for non-economic reasons, usually FT	436	401	254	191	2,479
Not at work, usually FT	227	238	179	105	1,389
PT hrs, usually PT for economic reasons	324	377	140	68	531
PT hrs, usually PT for non-economic	1,099	2,080	1,308	662	4,732
FT hours, usually PT for economic reasons	17	16	12	1	29
FT hours, usually PT for non-economic reasons	40	56	43	22	192

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Not at work, usually part-time	101	233	156	56	525
Unemployed FT	618	722	197	136	1,316
Unemployed PT	113	260	77	57	390
CLWK: Longest job class of worker (recode)					
Universe: All persons aged 15+					
Niu	2,431	10,167	1,488	1,160	17,629
Private	6,959	7,099	4,733	3,023	41,294
Government	1,009	747	446	710	9,436
Self-employed	849	992	1,008	253	1,614
Without pay	17	12	15	1	15
Never worked	3,770	19,728	6,551	4,348	10,177
EARNER: Earner status recode					
Universe: All persons aged 15+					
Niu	2,431	10,167	1,488	1,160	17,629
Earner	8,821	8,842	6,188	3,986	52,346

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Nonearner	3,783	19,736	6,565	4,349	10,190
HRSWK: In the weeks that ... worked how may hours did ... usually work per week?					
Universe: WKSWORK > 0					
(-0.099, 9.9]	6,347	30,317	8,296	5,648	28,472
(9.9, 19.8]	354	837	443	259	1,576
(19.8, 29.7]	875	1,550	858	390	2,922
(29.7, 39.6]	1,277	1,534	847	486	4,780
(39.6, 49.5]	5,110	3,719	2,826	2,191	34,221
(49.5, 59.4]	673	461	578	336	5,584
(59.4, 69.3]	276	228	263	122	1,929
(69.3, 79.2]	77	48	74	33	383
(79.2, 89.1]	41	33	33	20	222
(89.1, 99.0]	5	18	23	10	76
LJCW: Longest job class of worker					

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Universe: WKSWORK > 0					
Niu	6,201	29,895	8,039	5,508	27,806
Private	6,640	6,757	3,950	2,866	40,016
Federal	569	142	63	152	1,842
State	208	249	160	236	3,440
Local	232	356	223	322	4,154
Self employed incorporated, yes	319	342	783	157	1,278
Self employed incorporated, no or farm	849	992	1,008	253	1,614
Without pay	17	12	15	1	15

NWLKWK: How may different weeks was ... looking for work or on layoff?

Universe: NWLOOK = 1

(-0.052, 5.2]	14,892	38,462	14,188	9,469	79,995
(5.2, 10.4]	15	32	7	6	38
(10.4, 15.6]	13	29	4	0	17
(15.6, 20.8]	7	17	4	2	9

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(20.8, 26.0]	14	22	5	4	23
(26.0, 31.2]	3	7	1	0	2
(31.2, 36.4]	3	7	0	0	1
(36.4, 41.6]	6	17	1	1	5
(41.6, 46.8]	4	3	1	0	1
(46.8, 52.0]	78	149	30	13	74
<p>NWLOOK: Even though ... did not work in 20.. did spend and time trying to find a job or on layoff?</p> <p>Universe: WORKYN = 2</p>					
Niu	11,265	19,017	7,690	5,147	69,988
Yes	176	340	70	41	236
No	3,594	19,388	6,481	4,307	9,941
<p>PHMEMPRS: For how many employers did ... work in 20..? if more than one at same time, only count it as one employer</p> <p>Universe: WKSWORK > 0</p>					

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Niu	6,201	29,895	8,039	5,508	27,806
One employer	7,684	7,738	5,537	3,439	47,029
Two employers	857	848	535	439	4,433
3 or more employers	293	264	130	109	897
RSNNOTW: What was the main reason ... did not work in 20..?					
Universe: WORKYN = 2					
Niu	11,265	19,017	7,690	5,147	69,988
Ill or disabled	508	4,721	503	449	681
Retired	477	10,319	4,709	3,378	1,425
Taking care of home	1,331	1,690	562	231	2,816
Going to school	1,043	2,510	658	254	4,901
Could not find work	209	286	39	21	147
Other	202	202	80	15	207
WECLW: Longest job class of worker (persons 15+)					
Universe: All persons aged 15+					

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Not in universe	2,431	10,167	1,488	1,160	17,629
Agriculture (Wage and salary)	220	198	181	60	482
Agriculture (Self-employed)	51	58	120	32	106
Agriculture (Unpaid)	7	3	2	0	4
Nonagriculture (Private household)	100	138	60	18	133
Nonagriculture (Other private)	6,338	6,452	3,776	2,801	39,483
Nonagriculture (Government)	1,006	742	444	708	9,407
Nonagriculture (Self-employed)	1,102	1,250	1,606	367	2,733
Nonagriculture (Unpaid)	10	9	13	1	11
Nonagriculture (Never worked)	3,770	19,728	6,551	4,348	10,177

WEWKRS: Weeks worked recode

Universe: All persons aged 15+

Niu

Full-year worker (Full time)

Full-year worker (Part time)

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Part-year worker (Full time)	1,259	1,434	695	716	4,156
Part-year worker (Part time)	907	1,757	893	491	3,308
Part-year worker (Nonworker)	3,770	19,728	6,551	4,348	10,177

WKSWORK: During 20.. in how many weeks did ... work even for a few hours?
(include paid vacation and sick leave as work)

Universe: Persons 15+ with WORKYN = 1

(-0.052, 5.2]	6,329	30,179	8,164	5,588	28,130
(5.2, 10.4]	147	315	110	98	626
(10.4, 15.6]	180	343	147	104	716
(15.6, 20.8]	229	363	147	131	748
(20.8, 26.0]	318	518	218	197	926
(26.0, 31.2]	184	242	117	79	493
(31.2, 36.4]	235	266	155	111	733
(36.4, 41.6]	300	342	242	163	1,138
(41.6, 46.8]	267	292	165	126	986

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(46.8, 52.0]	6,846	5,885	4,776	2,898	45,669
WORKYN: Did ... work at a job or business at any time during 20..?					
Universe: All persons aged 15+					
Niu	2,431	10,167	1,488	1,160	17,629
Yes	8,727	8,684	6,108	3,938	52,062
No	3,877	19,894	6,645	4,397	10,474
WRK_CK: Worked last year recode, including temporary and part-time					
Universe: All persons aged 15+					
Niu	2,431	10,167	1,488	1,160	17,629
Yes	8,834	8,850	6,202	3,987	52,359
No	3,770	19,728	6,551	4,348	10,177
WTEMP: Did ... do any temporary, part-time, or seasonal work even for a few days during 20..?					
Universe: WORKYN = 2					
Niu	11,158	18,851	7,596	5,098	69,691

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Yes	107	166	94	49	297
No	3,770	19,728	6,551	4,348	10,177
ERN_OTR: Wage and salary money earned from other work, Y/N					
Universe: All persons aged 15+					
Niu	6,201	29,895	8,039	5,508	27,806
Yes	819	847	635	496	5,174
No	8,015	8,003	5,567	3,491	47,185
ERN_SRCE: Source of earnings from longest job					
Universe: ERN_YN = 1					
Niu	6,201	29,895	8,039	5,508	27,806
Wage and salary	7,968	7,846	5,179	3,733	50,730
Self employment	809	940	904	224	1,529
Farm self employment	40	52	104	29	85
Without pay	17	12	15	1	15

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
ERN_VAL: How much did ... earn from this employer before deductions in 20..? what was ... net earnings from this business/ farm after expenses during 20..?					
Universe: ERN_YN = 1					
(-11108.998, 101000.8]	14,748	38,542	13,748	9,127	72,515
(101000.8, 212000.6]	239	156	378	286	6,274
(212000.6, 323000.4]	22	24	56	54	780
(323000.4, 434000.2]	9	11	18	16	236
(434000.2, 545000.0]	6	6	13	6	114
(545000.0, 655999.8]	3	3	7	0	55
(655999.8, 766999.6]	1	0	4	1	23
(766999.6, 877999.4]	2	0	4	1	28
(877999.4, 988999.2]	1	0	1	1	21
(988999.2, 1099999.0]	4	3	12	3	119

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
ERN_YN: Earnings from employer or net earnings from business/ farm after expenses from longest job during 20.. ?					
Universe: WORKYN=1 or WTEMP=1					
Niu	2,431	10,167	1,488	1,160	17,629
Yes	8,817	8,838	6,187	3,986	52,344
No	3,787	19,740	6,566	4,349	10,192
FRM_VAL: Amount of farm self-employment earnings from secondary source					
Universe: FRMOTR = 1					
(-10288.999, 19000.9]	15,028	38,744	14,230	9,484	80,131
(19000.9, 48000.8]	3	1	7	3	25
(48000.8, 77000.7]	3	0	0	5	7
(77000.7, 106000.6]	1	0	4	3	1
(251000.1, 280000.0]	0	0	0	0	1
FRMOTR: Receiving farm self-employment from secondary source					
Universe: ERN_OTR = 1					

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Niu	14,212	37,902	13,606	9,002	75,001
Yes	86	56	73	43	478
No	737	787	562	450	4,686
FRSE_VAL: Total amount of farm self-employment earnings					
Universe: ERN_YN=1 or FRMOTR=1					
(-20767.998, 57001.8]	15,029	38,739	14,206	9,483	80,136
(57001.8, 134001.6]	6	5	29	10	25
(134001.6, 211001.4]	0	1	2	0	3
(211001.4, 288001.2]	0	0	3	1	1
(442000.8, 519000.6]	0	0	0	1	0
(673000.2, 750000.0]	0	0	1	0	0
FRSE_YN: Receiving any farm self-employment					
Universe: ERN_YN=1 or FRMOTR=1					
Niu	2,431	10,167	1,488	1,160	17,629
Yes	122	105	170	70	560

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
No	12,482	28,473	12,583	8,265	61,976
PEARINVAL: Total persons earnings					
Universe: All persons aged 15+					
(-12083.998, 198500.8]	14,962	38,669	14,069	9,370	78,229
(198500.8, 407000.6]	53	62	126	111	1,506
(407000.6, 615500.4]	11	11	22	8	220
(615500.4, 824000.2]	3	0	10	2	53
(824000.2, 1032500.0]	3	2	5	3	62
(1032500.0, 1240999.8]	3	1	8	1	93
(1240999.8, 1449499.6]	0	0	1	0	0
(1449499.6, 1657999.4]	0	0	0	0	1
(1657999.4, 1866499.2]	0	0	0	0	1

SE_VAL: Amount of own business self-employment earnings from secondary source

Universe: SEOTR = 1

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(-10558.999, 46000.9]	15,027	38,736	14,220	9,484	80,099
(46000.9, 102000.8]	8	7	14	6	45
(102000.8, 158000.7]	0	2	5	2	6
(158000.7, 214000.6]	0	0	0	2	4
(214000.6, 270000.5]	0	0	0	1	1
(270000.5, 326000.4]	0	0	2	0	5
(326000.4, 382000.3]	0	0	0	0	3
(382000.3, 438000.2]	0	0	0	0	1
(494000.1, 550000.0]	0	0	0	0	1

SEMP_VAL: Total own business self-employment earnings (combined amounts in ern-val, if ern-srce=2, and se-val)

Universe: ERN_YN=1 or SEOTR=1

(-21117.997, 92001.7]	14,989	38,698	14,106	9,464	79,943
(92001.7, 204001.4]	39	41	111	24	179
(204001.4, 316001.1]	2	3	15	4	20

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(316001.1, 428000.8]	0	2	2	2	11
(428000.8, 540000.5]	3	1	2	1	4
(540000.5, 652000.2]	0	0	1	0	2
(652000.2, 763999.9]	0	0	1	0	2
(763999.9, 875999.6]	0	0	1	0	1
(987999.3, 1099999.0]	2	0	2	0	3
SEMP_YN: Receiving own business self-employment, y/n					
Universe: ERN_YN=1 or SEOTR=1					
Niu	2,431	10,167	1,488	1,160	17,629
Yes	942	1,075	1,061	320	2,577
No	11,662	27,503	11,692	8,015	59,959
SEOTR: Receiving own business self-employment, y/n					
Universe: ERN_YN=1 or SEOTR=1					
Niu	14,214	37,904	13,607	9,000	74,996
Yes	148	149	171	101	1,077

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
No	673	692	463	394	4,092
WAGEOTR: Receiving wage and salary earnings from other employers, y/n					
Universe: ERN_OTR = 1					
Niu	14,218	37,901	13,607	9,002	74,994
Yes	786	807	590	471	4,927
No	31	37	44	22	244
WS_VAL: Amount of wage and salary earnings from other employers					
Universe: ERN_OTR = 1					
(-1099.999, 109999.9]	15,033	38,738	14,235	9,491	80,092
(109999.9, 219999.8]	1	7	5	3	59
(219999.8, 329999.7]	1	0	1	1	3
(329999.7, 439999.6]	0	0	0	0	5
(439999.6, 549999.5]	0	0	0	0	1
(879999.2, 989999.1]	0	0	0	0	3

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(989999.1, 1099999.0]	0	0	0	0	2
WSAL_VAL: Total wage and salary earnings (combined amounts in ern-val, if ern-srce=1, and ws-val)					
Universe: ERN_YN=1 or WAGEOTR=1					
(-1999.999, 199999.9]	14,976	38,684	14,113	9,393	78,320
(199999.9, 399999.8]	38	44	85	87	1,377
(399999.8, 599999.7]	13	13	25	9	247
(599999.7, 799999.6]	3	1	4	1	56
(799999.6, 999999.5]	3	0	4	2	49
(999999.5, 1199999.4]	2	3	10	3	114
(1199999.4, 1399999.3]	0	0	0	0	1
(1799999.1, 1999999.0]	0	0	0	0	1
WSAL_YN: Receiving wage and salary earnings					
Universe: ERN_YN=1 or WAGEOTR=1					
Niu	2,431	10,167	1,488	1,160	17,629

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Yes	8,025	7,920	5,259	3,764	50,886
No	4,579	20,658	7,494	4,571	11,650
ANN_VAL: Retirement income, annuities amount					
Universe: ANN_YN = 1					
(-396.0, 39600.0]	15,030	38,705	14,208	9,456	80,136
(39600.0, 79200.0]	4	28	23	34	18
(79200.0, 118800.0]	1	7	6	3	8
(118800.0, 158400.0]	0	3	2	0	2
(158400.0, 198000.0]	0	2	0	1	0
(356400.0, 396000.0]	0	0	2	1	1
ANN_YN: Retirement income, annuities, y/n					
Universe: All Persons aged 15+					
Niu	2,431	10,167	1,488	1,160	17,629
Yes	26	634	573	422	219

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
No	12,578	27,944	12,180	7,913	62,317
CAP_VAL: Capital gains value					
Universe: CAP_YN = 1					
(-999.999, 99999.9]	15,031	38,725	14,211	9,473	80,085
(99999.9, 199999.8]	2	13	16	16	35
(199999.8, 299999.7]	2	6	6	5	24
(299999.7, 399999.6]	0	1	3	0	9
(399999.6, 499999.5]	0	0	1	0	3
(499999.5, 599999.4]	0	0	1	1	0
(699999.3, 799999.2]	0	0	1	0	7
(899999.1, 999999.0]	0	0	2	0	2
CAP_YN: Yes/no answer to âDid you receive capital gain from your shares of stock or mutual fund?'					
Universe: DIV_YN = 1					
Niu	14,044	36,074	11,363	7,534	66,843

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Yes	176	734	958	693	3,176
No	815	1,937	1,920	1,268	10,146
DBTN_VAL: Total amount of retirement distributions received (dst_val1 + dst_val2)					
Universe: DST_VAL1>0 OR DST_VAL2>0					
(-999.999, 99999.9]	15,033	38,711	14,203	9,460	80,139
(99999.9, 199999.8]	2	32	35	32	23
(199999.8, 299999.7]	0	2	2	1	2
(299999.7, 399999.6]	0	0	0	1	0
(399999.6, 499999.5]	0	0	1	0	1
(899999.1, 999999.0]	0	0	0	1	0
DIS_SC1: What was the source of disability income?					
Universe: DIS_YN=1					
Niu	14,947	38,270	14,130	9,359	79,707
Worker's compensation	16	32	11	15	96

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Company or union disability	10	48	19	34	123
Federal government disability	6	58	9	10	15
Us military retirement disability	18	45	10	8	12
State or local gov't employee disability	14	92	21	25	56
Us railroad retirement disability	0	6	2	0	1
Accident or disability insurance	8	32	16	17	60
Blacklung miners disability	0	0	0	1	0
State temporary sickness	3	1	2	1	9
Other or don't know	13	161	21	25	86

DIS_SC2: What was the source of disability income?

Universe: DIS_YN=1

Niu	15,035	38,740	14,240	9,493	80,158
Federal government disability	0	0	1	0	0
Us military retirement disability	0	1	0	0	0
State or local gov't employee disability	0	2	0	1	3

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Other or don't know	0	2	0	1	4
DIS_VAL1: How much did ... receive (source type) during 20.. ?					
Universe: DIS_SC1>0					
(-100.0, 10000.0]	14,993	38,533	14,185	9,428	80,005
(10000.0, 20000.0]	26	144	25	31	78
(20000.0, 30000.0]	7	33	16	23	40
(30000.0, 40000.0]	4	13	4	4	15
(40000.0, 50000.0]	3	10	1	2	11
(50000.0, 60000.0]	1	0	0	1	1
(60000.0, 70000.0]	1	1	1	0	1
(70000.0, 80000.0]	0	1	1	1	4
(80000.0, 90000.0]	0	1	0	1	0
(90000.0, 100000.0]	0	9	8	4	10
DIS_VAL2: How much did ... receive (source type) during 20.. ?					
Universe: DIS_SC2>0					

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(-23.672, 2367.2]	15,035	38,740	14,240	9,493	80,158
(4734.4, 7101.6]	0	1	0	0	4
(7101.6, 9468.8]	0	0	0	0	1
(11836.0, 14203.2]	0	0	0	0	1
(14203.2, 16570.4]	0	3	1	2	0
(21304.8, 23672.0]	0	1	0	0	1
DIS_YN: Other than social security did ... receive any income in 20.. as a result of health problems?					
Universe: All Persons aged 15+					
Niu	2,431	10,167	1,488	1,160	17,629
Yes	88	475	111	136	458
No	12,516	28,103	12,642	8,199	62,078
DIV_VAL: How much did ... receive in dividends from stocks or mutual funds during 20.. ?					
Universe: DIV_YN = 1					

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(-999.999, 99999.9]	15,031	38,730	14,217	9,476	80,108
(99999.9, 199999.8]	4	10	14	14	36
(199999.8, 299999.7]	0	3	6	3	16
(299999.7, 399999.6]	0	2	2	0	2
(699999.3, 799999.2]	0	0	0	2	0
(899999.1, 999999.0]	0	0	2	0	3
DIV_YN: Did ... receive dividends?					
Universe: All Persons aged 15+					
Niu	2,431	10,167	1,488	1,160	17,629
Yes	583	1,873	2,246	1,575	8,875
No	12,021	26,705	10,507	6,760	53,661
DSAB_VAL: Total amount of disability income received, combined amounts in edited sources one and two					
Universe: DIS_VAL1>0 OR DIS_VAL2>0					
(-100.0, 10000.0]	14,993	38,529	14,184	9,427	80,002

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(10000.0, 20000.0]	26	147	25	32	77
(20000.0, 30000.0]	7	33	17	23	44
(30000.0, 40000.0]	4	14	4	3	15
(40000.0, 50000.0]	3	10	1	2	11
(50000.0, 60000.0]	1	0	0	2	1
(60000.0, 70000.0]	1	1	1	0	1
(70000.0, 80000.0]	0	1	1	1	4
(80000.0, 90000.0]	0	1	0	1	0
(90000.0, 100000.0]	0	9	8	4	10

DST_SC1: Retirement income, distribution source 1

Universe: DST_VAL1 > 0 and a_age >= 58

Niu	14,982	37,052	12,699	8,267	79,685
401k account	28	684	568	499	249
403b account	0	49	39	48	20
Roth ira	2	114	99	60	24

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Regular ira	17	739	715	499	115
Keogh plan	0	1	3	3	1
Sep plan (simplified employee pension)	1	12	27	18	5
Other type of retirement account	5	94	91	101	66
DST_SC1_YNG: Retirement Distribution source 1, person under age 58					
Universe: DST_YN_YNG = 1 and a_age < 58					
Niu	14,950	38,651	14,163	9,424	79,246
401k account	52	60	45	47	653
403b account	4	3	3	4	41
Roth ira	13	11	5	7	66
Regular ira	11	15	20	4	107
Sep plan (simplified employee pension)	0	1	1	0	3
Other type of retirement account	5	4	4	9	49
DST_SC2: Retirement income, distribution source 2					
Universe: DST_VAL2 > 0 and a_age >= 58					

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Niu	15,034	38,662	14,160	9,433	80,151
403b account	0	4	5	5	1
Roth ira	1	12	12	6	3
Regular ira	0	51	45	38	9
Keogh plan	0	0	1	0	0
Sep plan (simplified employee pension)	0	3	2	3	0
Other type of retirement account	0	13	16	10	1
DST_SC2_YNG: Retirement Distribution source 2, person under age 58					
Universe: DST_VAL_YNG > 0 and a_age < 58					
Niu	15,031	38,739	14,241	9,494	80,146
403b account	0	0	0	0	1
Roth ira	2	2	0	1	9
Regular ira	2	2	0	0	5
Sep plan (simplified employee pension)	0	2	0	0	3
Other type of retirement account	0	0	0	0	1

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
DST_VAL1: Retirement income amount, distribution source 1					
Universe: DST_SC1 = 1					
(-999.999, 99999.9]	15,033	38,711	14,207	9,463	80,139
(99999.9, 199999.8]	2	32	31	29	23
(199999.8, 299999.7]	0	2	2	1	2
(299999.7, 399999.6]	0	0	0	1	0
(399999.6, 499999.5]	0	0	1	0	1
(899999.1, 999999.0]	0	0	0	1	0
DST_VAL1_YNG: Retirement Distribution amount 1, under age 58					
Universe: DST_SC1_YNG = 1					
(-999.999, 99999.9]	15,033	38,743	14,240	9,494	80,137
(99999.9, 199999.8]	1	1	0	1	17
(199999.8, 299999.7]	0	1	1	0	6
(299999.7, 399999.6]	1	0	0	0	1
(399999.6, 499999.5]	0	0	0	0	3

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(899999.1, 999999.0]	0	0	0	0	1
DST_VAL2: Retirement income amount, distribution source 2					
Universe: DST_SC2 = 1					
(-75.0, 7500.0]	15,034	38,719	14,208	9,469	80,158
(7500.0, 15000.0]	1	20	21	15	4
(15000.0, 22500.0]	0	0	3	2	0
(22500.0, 30000.0]	0	0	1	5	3
(30000.0, 37500.0]	0	1	1	0	0
(37500.0, 45000.0]	0	0	1	1	0
(45000.0, 52500.0]	0	1	0	0	0
(52500.0, 60000.0]	0	1	4	1	0
(60000.0, 67500.0]	0	2	0	0	0
(67500.0, 75000.0]	0	1	2	2	0
DST_VAL2_YNG: Retirement Distribution amount 2, under age 58					
Universe: DST_SC2_YNG = 1					

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(-43.0, 4300.0]	15,032	38,742	14,241	9,494	80,157
(4300.0, 8600.0]	2	1	0	1	4
(8600.0, 12900.0]	0	1	0	0	1
(21500.0, 25800.0]	0	0	0	0	1
(30100.0, 34400.0]	0	1	0	0	2
(38700.0, 43000.0]	1	0	0	0	0
DST_YN: Retirement income distribution y/n					
Universe: Persons aged 58 and over (a_age >= 58)					
Niu	13,643	23,641	7,180	3,933	72,508
Yes	53	1,693	1,543	1,228	480
No	1,339	13,411	5,518	4,334	7,177
DST_YN_YNG: Retirement Distribution Reciprocity, person under age 58					
Universe: Persons under age 58 (a_age < 58)					
Niu	3,823	25,271	8,549	6,722	25,286
Yes	85	94	78	71	919

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
No	11,127	13,380	5,614	2,702	53,960
ED_VAL: Total amount of educational assistance received (combined amounts in pell grant and other educational) assistance during 20.. ?					
Universe: ED_YN = 1					
(-99,999, 9999.9]	14,940	38,640	14,141	9,451	79,622
(9999.9, 19999.8]	62	73	50	21	289
(19999.8, 29999.7]	20	17	26	10	141
(29999.7, 39999.6]	7	8	9	9	59
(39999.6, 49999.5]	2	2	2	1	28
(49999.5, 59999.4]	4	2	5	2	16
(59999.4, 69999.3]	0	1	4	0	3
(69999.3, 79999.2]	0	0	0	0	2
(79999.2, 89999.1]	0	2	2	0	3
(89999.1, 99999.0]	0	0	2	1	2
ED_YN: Did ... receive educational assistance?					

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Universe: All Persons aged 15+					
Niu	2,431	10,167	1,488	1,160	17,629
Yes	430	611	303	159	1,946
No	12,174	27,967	12,450	8,176	60,590
FIN_VAL: How much did ... receive in financial assistance income during 20..					
?					
Universe: FIN_YN = 1					
(-500.0, 50000.0]	15,033	38,742	14,238	9,491	80,147
(50000.0, 100000.0]	2	3	3	4	15
(100000.0, 150000.0]	0	0	0	0	2
(450000.0, 500000.0]	0	0	0	0	1
FIN_YN: Did ... receive financial assistance?					
Universe: All Persons aged 15+					
Niu	2,431	10,167	1,488	1,160	17,629
Yes	166	321	141	75	406

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
No	12,438	28,257	12,612	8,260	62,130
INT_VAL: Edited total combined interest income					
Universe: INT_YN = 1					
(-280.0, 28000.0]	14,979	38,527	13,944	9,220	78,544
(28000.0, 56000.0]	31	126	164	145	937
(56000.0, 84000.0]	16	41	60	46	281
(84000.0, 112000.0]	7	45	66	73	354
(112000.0, 140000.0]	1	4	7	10	35
(140000.0, 168000.0]	1	1	0	0	11
(168000.0, 196000.0]	0	0	0	1	1
(196000.0, 224000.0]	0	1	0	0	1
(252000.0, 280000.0]	0	0	0	0	1
INT_YN: Edited total combined interest income, y/n					
Universe: All Persons aged 15+					
Niu	2,431	10,167	1,488	1,160	17,629

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Yes	3,950	9,847	7,759	5,700	40,283
No	8,654	18,731	4,994	2,635	22,253
OED_TYP1: Source 1 other than gi bill received (OED_TYP1- source of other government assistance)					
Universe: ED_YN = 1					
Niu	14,584	38,089	13,928	9,331	78,173
Yes	102	144	62	44	321
No	349	512	251	120	1,671
OED_TYP2: Source 2 other than gi bill received (OED_TYP2- scholarships, grants etc. from the school)					
Universe: ED_YN = 1					
Niu	14,584	38,089	13,928	9,331	78,173
Yes	146	211	153	61	986
No	305	445	160	103	1,006

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
OED_TYP3: Source other than gi bill received (OED_TYP3- other assistance (employers friends, etc.) Universe: ED_YN = 1					
Niu	14,584	38,089	13,928	9,331	78,173
Yes	51	51	41	26	375
No	400	605	272	138	1,617
OI_OFF: Other income sources Universe: OI_YN = 1					
Niu	14,824	38,368	14,077	9,332	79,115
Social security	1	2	1	0	3
Private pensions	0	5	3	3	5
Afdc	6	6	3	0	13
Other public assistance	0	2	0	1	5
Dividends	0	1	0	0	0
Rents or royalties	2	1	3	0	7

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
State disability payments (worker's comp)	1	1	0	1	1
Disability payments (own insurance)	0	1	0	0	4
Annuities or paid up insurance policies	1	1	1	0	2
Anything else	192	330	137	150	969
Alimony	8	27	16	8	41
OI_VAL: How much did ... receive in other incomes					
Universe: OI_YN = 1					
(-950.0, 95000.0]	15,033	38,744	14,240	9,488	80,149
(95000.0, 190000.0]	2	0	1	5	12
(190000.0, 285000.0]	0	0	0	1	0
(285000.0, 380000.0]	0	1	0	0	1
(380000.0, 475000.0]	0	0	0	1	1
(475000.0, 570000.0]	0	0	0	0	1
(855000.0, 950000.0]	0	0	0	0	1
OI_YN: Did ... receive cash income not already covered from any other source?					

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Universe: All Persons aged 15+					
None or niu	2,431	10,167	1,488	1,160	17,629
Yes	211	377	164	163	1,050
No	12,393	28,201	12,589	8,172	61,486
PEN_SC1: Retirement income, pension source 1					
Universe: PEN_YN = 1					
Niu	14,862	36,035	12,394	7,307	79,002
Company pension	48	1,416	1,039	872	419
Union pension	15	264	176	183	94
Federal government pension	22	173	76	262	130
State government pension	21	524	397	643	336
Local government pension	10	162	84	168	129
Us military pension	56	118	15	15	35
Us railroad retirement	0	10	6	8	2
Other	1	43	54	37	18

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
PEN_SC2: Retirement income, pension source 2					
Universe: PEN_VAL2 > 0					
Niu	15,028	38,634	14,198	9,420	80,137
Union pension	1	21	16	20	4
Federal government pension	0	8	3	6	1
State government pension	1	17	9	29	8
Local government pension	0	9	4	6	6
Us military pension	5	49	5	11	7
Us railroad retirement	0	1	0	0	0
Other	0	6	6	3	2
PEN_VAL1: Retirement income amount, pension source 1					
Universe: PEN_SC1 > 0					
(-999.999, 99999.9]	15,031	38,709	14,220	9,454	80,129
(99999.9, 199999.8]	4	21	16	33	27
(199999.8, 299999.7]	0	3	1	3	3

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(299999.7, 399999.6]	0	3	1	1	1
(399999.6, 499999.5]	0	3	1	0	2
(599999.4, 699999.3]	0	2	0	0	0
(699999.3, 799999.2]	0	1	0	0	0
(899999.1, 999999.0]	0	3	2	4	3
PEN_VAL2: Retirement income amount, pension source 2					
Universe: PEN_SC2 > 0					
(-360.0, 36000.0]	15,033	38,737	14,239	9,485	80,158
(36000.0, 72000.0]	1	6	1	7	6
(72000.0, 108000.0]	1	1	1	2	1
(108000.0, 144000.0]	0	0	0	1	0
(324000.0, 360000.0]	0	1	0	0	0
PEN_YN: Retirement income, pension y/n					
Universe: All Persons aged 15+					
Niu	2,431	10,167	1,488	1,160	17,629

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Yes	173	2,710	1,847	2,188	1,163
No	12,431	25,868	10,906	6,147	61,373
PNSN_VAL: Total combined amount of pension income received from all pension sources					
Universe: PEN_YN = 1					
(-999.999, 99999.9]	15,030	38,707	14,219	9,451	80,125
(99999.9, 199999.8]	5	22	17	36	31
(199999.8, 299999.7]	0	3	1	3	3
(299999.7, 399999.6]	0	4	1	1	1
(399999.6, 499999.5]	0	3	1	0	2
(599999.4, 699999.3]	0	2	0	0	0
(699999.3, 799999.2]	0	1	0	0	0
(899999.1, 999999.0]	0	3	2	4	3
PTOTVAL: Total persons income					
Universe: All Persons aged 15+					

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(-12094.703, 199571.3]	14,933	38,563	13,963	9,239	77,720
(199571.3, 409141.6]	78	150	209	217	1,918
(409141.6, 618711.9]	13	21	35	24	282
(618711.9, 828282.2]	5	5	14	3	74
(828282.2, 1037852.5]	3	4	4	7	60
(1037852.5, 1247422.8]	3	2	13	5	100
(1247422.8, 1456993.1]	0	0	2	0	8
(1456993.1, 1666563.4]	0	0	0	0	1
(1876133.7, 2085704.0]	0	0	1	0	2

RESNSSI: What were the reasons (you/name) (was/were) getting Social

Security Income last year?

Universe: SS_YN = 1

Niu

Retired

Disabled (adult or child)

14,638	25,268	8,599	5,024	78,937
195	10,639	5,128	3,924	693
138	2,272	280	266	293

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Widowed	25	208	93	57	51
Spouse	4	89	39	45	9
Surviving child	16	54	11	18	77
Dependent child	9	59	12	7	36
On behalf of surviving, dependent, or disabled child(ren)	8	61	6	10	51
Other (adult or child)	2	95	73	144	18
RESNSS2: What were the reasons (you/name) (was/were) getting Social Security Income last year?					
Universe: SS_YN = 1					
Niu	15,018	38,345	14,129	9,409	80,099
Disabled (adult or child)	2	164	28	20	7
Widowed	0	103	50	31	3
Spouse	3	20	4	4	3
Surviving child	0	5	2	0	3
Dependent child	0	4	0	0	2

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
On behalf of surviving, dependent, or disabled child(ren)	11	89	22	21	47
Other (adult or child)	1	15	6	10	1
RESNSSI1: What were the reasons (you/name) (was/were) getting Supplemental Security Income last year? Universe: SSI_YN = 1 Niu	14,976	36,504	14,140	9,303	80,055
Disabled (adult or child)	39	1,992	77	159	66
Blind (adult or child)	0	25	2	1	2
On behalf of a disabled child	16	58	6	10	25
On behalf of a blind child	0	2	0	0	1
Other (adult or child)	4	164	16	22	16
RESNSSI2: What were the reasons (you/name) (was/were) getting Supplemental Security Income last year? Universe: SSI_YN = 1 Niu	15,031	38,715	14,240	9,493	80,162

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Blind (adult or child)	0	5	0	0	0
On behalf of a disabled child	2	14	0	1	1
Other (adult or child)	2	11	1	1	2
RETCB_VAL: Retirement contribution, amount					
Universe: RETCB_YN = 1					
(-32.0, 3200.0]	14,564	38,456	13,704	8,916	67,888
(3200.0, 6400.0]	256	114	243	252	5,011
(6400.0, 9600.0]	63	60	116	117	2,102
(9600.0, 12800.0]	62	47	52	56	1,625
(12800.0, 16000.0]	31	18	22	30	945
(16000.0, 19200.0]	37	10	50	46	1,617
(19200.0, 22400.0]	10	17	18	23	279
(22400.0, 25600.0]	12	20	32	48	632
(25600.0, 28800.0]	0	0	0	2	22
(28800.0, 32000.0]	0	3	4	5	44

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
RETCB_YN: Retirement contribution, y/n					
Universe: All people 15 years and over					
Niu	13,470	34,901	10,249	6,228	53,437
Yes	1,034	793	1,070	1,247	21,810
No	531	3,051	2,922	2,020	4,918
RINT_SC1: Interest income, retirement source 1					
Universe: RINT_YN = 1					
Niu	13,470	34,901	10,249	6,228	53,437
401k account	973	1,925	1,791	1,791	19,885
403b account	60	121	118	188	2,112
Roth ira	216	421	583	292	1,465
Regular ira	163	1,063	1,207	711	1,239
Keogh plan	0	5	11	4	23
Sep plan (simplified employee pension)	19	49	98	43	305
Other type of retirement account	134	260	184	238	1,699

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
RINT_SC2: Interest income, retirement source 2					
Universe: RINT_YN = 1					
Niu	14,818	38,284	13,614	8,981	75,781
403b account	10	27	23	34	351
Roth ira	92	113	154	163	2,018
Regular ira	65	255	342	228	1,284
Keogh plan	0	1	6	0	10
Sep plan (simplified employee pension)	7	16	48	18	162
Other type of retirement account	43	49	54	71	559
RINT_VAL1: Interest income amt, retirement source 1					
Universe: RINT_SC1 > 0					
(-100.0, 10000.0]	14,936	38,372	13,795	9,102	77,436
(10000.0, 20000.0]	51	173	178	147	1,160
(20000.0, 30000.0]	17	60	86	68	496
(30000.0, 40000.0]	9	45	56	40	274

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(40000.0, 50000.0]	6	28	43	46	287
(50000.0, 60000.0]	3	5	16	7	85
(60000.0, 70000.0]	3	9	12	10	75
(70000.0, 80000.0]	5	13	9	17	71
(80000.0, 90000.0]	0	2	6	4	26
(90000.0, 100000.0]	5	38	40	54	255
RINT_VAL2: Interest income amt, retirement source 2					
Universe: RINT_SC2 > 0					
(-100.0, 10000.0]	15,015	38,701	14,182	9,431	79,816
(10000.0, 20000.0]	9	14	22	25	140
(20000.0, 30000.0]	0	13	14	16	44
(30000.0, 40000.0]	2	2	2	4	39
(40000.0, 50000.0]	3	6	7	3	15
(50000.0, 60000.0]	2	2	1	3	11
(60000.0, 70000.0]	1	0	1	1	14

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(70000.0, 80000.0]	0	2	2	3	12
(80000.0, 90000.0]	0	0	1	0	9
(90000.0, 100000.0]	3	5	9	9	65
RNT_YN: Interest income - retirement, y/n					
Universe: All Persons aged 15+					
Niu	2,431	10,167	1,488	1,160	17,629
Yes	1,565	3,844	3,992	3,267	26,728
No	11,039	24,734	8,761	5,068	35,808
RNT_VAL: How much did ... receive in income from rent after expenses during 20..?					
Universe: RNT_YN = 1					
(-11008.998, 91000.8]	15,031	38,718	14,217	9,473	80,117
(91000.8, 192000.6]	2	25	18	20	26
(192000.6, 293000.4]	0	1	1	0	10
(293000.4, 394000.2]	1	1	1	0	6

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(394000.2, 495000.0]	0	0	1	1	2
(495000.0, 595999.8]	0	0	0	0	1
(595999.8, 696999.6]	1	0	0	0	1
(898999.2, 999999.0]	0	0	3	1	2
RNT_YN: Did ... own any land, property, rented to others, or receive income from royalties, roomers or boarders, or from estates or trusts? Universe: All Persons aged 15+					
Niu	2,431	10,167	1,488	1,160	17,629
Yes	290	918	1,088	677	2,802
No	12,314	27,660	11,665	7,658	59,734
SRVS_VAL: Total amount of survivor's income received (combined amounts in edited sources sur_val1 and sur_val2 plus the unedited sources 3 & 4 starting in 1995) Universe: SUR_YN = 1 (-200.0, 20000.0]					
	15,022	38,674	14,181	9,420	80,073

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(20000.0, 40000.0]	7	39	39	48	47
(40000.0, 60000.0]	4	18	8	14	13
(60000.0, 80000.0]	0	1	3	0	8
(80000.0, 100000.0]	2	11	8	13	20
(100000.0, 120000.0]	0	1	1	0	1
(120000.0, 140000.0]	0	1	1	0	1
(140000.0, 160000.0]	0	0	0	0	1
(180000.0, 200000.0]	0	0	0	0	1
SS_VAL: How much did ... receive in social security payments during 20.. ?					
Universe: SS_YN = 1					
(-80.0, 8000.0]	14,729	27,315	9,197	5,611	79,192
(8000.0, 16000.0]	185	5,828	1,913	1,388	471
(16000.0, 24000.0]	91	3,923	2,002	1,553	335
(24000.0, 32000.0]	20	1,192	846	695	113
(32000.0, 40000.0]	2	203	146	140	21

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(40000.0, 48000.0]	8	279	136	107	30
(48000.0, 56000.0]	0	3	1	0	0
(56000.0, 64000.0]	0	0	0	1	1
(72000.0, 80000.0]	0	2	0	0	2
SS_YN: Who received social security payments either for themselves or as combined payments with other family members?					
Universe: All Persons aged 15+					
Niu	2,431	10,167	1,488	1,160	17,629
Yes	397	13,477	5,642	4,471	1,228
No	12,207	15,101	7,111	3,864	61,308
SSI_VAL: How much did ... receive in supplemental security income during 20..?					
Universe: SSI_YN = 1					
(-50.0, 5000.0]	14,990	37,145	14,170	9,351	80,087
(5000.0, 10000.0]	35	1,032	35	77	47

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(10000.0, 15000.0]	3	388	21	44	21
(15000.0, 20000.0]	1	107	7	10	4
(20000.0, 25000.0]	2	41	3	9	3
(25000.0, 30000.0]	3	31	5	4	3
(45000.0, 50000.0]	1	1	0	0	0
SSI_YN: Did ... received ssi?					
Universe: All Persons aged 15+					
Niu	2,431	10,167	1,488	1,160	17,629
Yes	59	2,241	101	192	110
No	12,545	26,337	12,652	8,143	62,426
STRKUC: At any time during 20.. did ... receive any union unemployment or strike benefits?					
Universe: UC_YN = 1					
Niu	2,431	10,167	1,488	1,160	17,629
Yes	4	10	3	4	27

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
No	12,600	28,568	12,750	8,331	62,509
SUBUC: At any time during 20.. did ... receive any supplemental unemployment benefits?					
Universe: UC_YN = 1					
Niu	2,431	10,167	1,488	1,160	17,629
Yes	11	28	9	8	47
No	12,593	28,550	12,744	8,327	62,489
SUR_SC1: What was the source of this other widow or survivor income?					
Universe: SUR_YN = 1					
None or niu	14,986	38,246	13,934	9,233	79,856
Company or union survivor pension	10	206	134	106	44
Federal government	7	49	25	41	26
Us military retirement survivor pension	2	48	10	10	9
State or local gov't survivor pension	3	44	34	39	19
Us railroad retirement survivor pension	2	14	6	3	5

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Worker compensation survivor	0	2	0	3	3
Black lung	0	1	0	0	1
Regular payments from estates or trusts	8	40	34	17	79
Regular payments from annuities or paid-up life insurance	6	29	30	15	42
Other or don't know	11	66	34	28	81
SUR_SC2: What was the source of this other widow or survivor income?					
Universe: SUR_YN = 1					
None or niu	15,034	38,731	14,233	9,490	80,152
Federal government	0	2	0	0	0
Us military retirement survivor pension	1	2	0	1	0
State or local gov't survivor pension	0	2	3	1	7
Worker compensation survivor	0	1	0	0	0
Black lung	0	0	0	1	0
Regular payments from estates or trusts	0	0	1	0	1
Regular payments from annuities or paid-up life insurance	0	5	1	2	0

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Other or don't know	0	2	3	0	5
SUR_VAL1: How much did ... receive (survivor source type) during 20.. ?					
Universe: SUR_YN = 1					
(-100.0, 10000.0]	15,009	38,539	14,106	9,366	80,014
(10000.0, 20000.0]	13	137	78	56	61
(20000.0, 30000.0]	6	35	25	36	32
(30000.0, 40000.0]	1	5	14	11	15
(40000.0, 50000.0]	3	14	5	10	6
(50000.0, 60000.0]	1	3	3	4	8
(60000.0, 70000.0]	0	0	1	1	7
(70000.0, 80000.0]	0	1	2	0	1
(90000.0, 100000.0]	2	11	7	11	21
SUR_VAL2: How much did ... receive (source type) during 20.. ?					
Universe: SUR_YN = 1					
(-100.0, 10000.0]	15,035	38,741	14,237	9,493	80,160

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(10000.0, 20000.0]	0	1	1	1	0
(20000.0, 30000.0]	0	0	0	0	1
(30000.0, 40000.0]	0	1	0	0	0
(60000.0, 70000.0]	0	1	1	1	0
(90000.0, 100000.0]	0	1	2	0	4
SUR_YN: During 20.. did ... receive any survivor benefits such as widow's pensions, estates, trusts, insurance annuities, or other survivor's income?					
Universe: All Persons aged 15+					
Niu	2,431	10,167	1,488	1,160	17,629
Yes	49	499	307	262	309
No	12,555	28,079	12,446	8,073	62,227
TRDINT_VAL: Interest amount, excluding retirement account interest					
Universe: INT_YN = 1					
(-99.999, 9999.9]	15,018	38,629	14,089	9,398	79,874
(9999.9, 19999.8]	8	69	87	53	147

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(19999.8, 29999.7]	3	21	23	14	64
(29999.7, 39999.6]	2	8	16	8	24
(39999.6, 49999.5]	0	5	4	2	9
(49999.5, 59999.4]	1	6	6	4	14
(59999.4, 69999.3]	1	1	1	2	11
(69999.3, 79999.2]	1	1	3	4	7
(79999.2, 89999.1]	1	0	1	2	3
(89999.1, 99999.0]	0	5	11	8	12

UC_VAL: How much did ... receive in unemployment benefits during 20..?

Universe: UC_YN = 1

(-99.999, 9999.9]	15,013	38,710	14,224	9,465	80,074
(9999.9, 19999.8]	21	26	13	26	79
(19999.8, 29999.7]	1	6	0	1	5
(29999.7, 39999.6]	0	1	0	1	0
(39999.6, 49999.5]	0	1	1	0	4

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(49999.5, 59999.4]	0	1	3	2	1
(69999.3, 79999.2]	0	0	0	0	1
(89999.1, 99999.0]	0	0	0	0	1
UC_YN: Any type of unemployment compensation? (Combination of subuc, strkuc, and uctot_yn)					
Universe: UC_YN = 1					
Niu	2,431	10,167	1,488	1,160	17,629
Yes	180	305	119	154	805
No	12,424	28,273	12,634	8,181	61,731
VET_TYP1: What type of veterans payments did receive? (VET_TYP1-disability compensation?)					
Universe: VET_YN = 1					
Niu	14,764	37,749	14,043	9,176	79,766
Yes	203	675	131	264	322

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
No	68	321	67	55	77
VET_TYP2: What type of veterans payments did receive? (VET_TYP2-survivor benefits?)					
Universe: VET_YN = 1					
Niu	14,764	37,749	14,043	9,176	79,766
Yes	4	80	16	14	5
No	267	916	182	305	394
VET_TYP3: What type of veterans payments did receive? (VET_TYP3-veteran's pension?)					
Universe: VET_YN = 1					
Niu	14,764	37,749	14,043	9,176	79,766
Yes	76	245	41	42	48
No	195	751	157	277	351
VET_TYP4: What type of veterans payments did receive? (VET_TYP4-education assistance?)					

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Universe: VET_YN = 1					
Niu	14,764	37,749	14,043	9,176	79,766
Yes	14	18	3	7	24
No	257	978	195	312	375
VET_TYP5: What type of veterans payments did receive? (VET_TYP5- other veteran's payments?)					
Universe: VET_YN = 1					
Niu	14,764	37,749	14,043	9,176	79,766
Yes	8	33	11	7	12
No	263	963	187	312	387
VET_VAL: How much did ... receive from veterans' administration during 20..?					
Universe: VET_YN = 1					
(-100.0, 10000.0]	14,845	38,124	14,132	9,317	79,960
(10000.0, 20000.0]	61	292	49	77	98

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(20000.0, 30000.0]	67	121	20	42	59
(30000.0, 40000.0]	23	134	24	34	25
(40000.0, 50000.0]	18	55	9	19	16
(50000.0, 60000.0]	3	8	2	2	1
(60000.0, 70000.0]	7	3	1	2	0
(70000.0, 80000.0]	4	0	1	0	0
(80000.0, 90000.0]	4	2	2	0	3
(90000.0, 100000.0]	3	6	1	2	3
VET_YN: Did ... receive veterans' payments?					
Universe: All Persons aged 15+					
Niu	2,431	10,167	1,488	1,160	17,629
Yes	271	996	198	319	399
No	12,333	27,582	12,555	8,016	62,137
WC_TYPE: What was source of these payments?					
Universe: WC_YN = 1					

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Not in universe	14,980	38,653	14,204	9,447	79,891
State worker's compensation	15	40	14	15	74
Employer or employers insurance	39	42	23	30	187
Own insurance	0	1	0	0	5
Other	1	9	0	3	8
WC_VAL: How much compensation did ... receive during 20..?					
Universe: WC_YN = 1					
(-99.999, 9999.9]	15,009	38,712	14,227	9,467	80,086
(9999.9, 19999.8]	17	18	6	19	44
(19999.8, 29999.7]	5	8	2	2	15
(29999.7, 39999.6]	1	6	5	6	12
(39999.6, 49999.5]	0	0	0	0	3
(49999.5, 59999.4]	1	0	0	1	0
(59999.4, 69999.3]	0	1	0	0	3
(89999.1, 99999.0]	2	0	1	0	2

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
WC_YN: During 20.. did ... receive any worker's compensation payments or other payments as a result of a job related injury or illness?					
Universe: All Persons aged 15+					
Niu	2,431	10,167	1,488	1,160	17,629
Yes	55	92	37	48	274
No	12,549	28,486	12,716	8,287	62,262
PAW_TYP: What type of program did... receive CASH assistance?					
Universe: PAW_YN = 1					
Niu	15,011	38,275	14,214	9,382	80,127
TANF/AFDC	14	327	13	51	16
Other	8	130	14	60	21
Both	2	13	0	2	1
PAW_VAL: How much did ... receive in public assistance or welfare during 20..?					
Universe: PAW_YN = 1					

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(-25.0, 2500.0]	15,018	38,508	14,228	9,445	80,143
(2500.0, 5000.0]	6	115	7	28	8
(5000.0, 7500.0]	5	53	4	5	6
(7500.0, 10000.0]	2	42	1	8	3
(10000.0, 12500.0]	3	17	0	5	4
(12500.0, 15000.0]	1	6	0	0	0
(15000.0, 17500.0]	0	1	0	1	0
(17500.0, 20000.0]	0	0	0	2	1
(20000.0, 22500.0]	0	2	0	0	0
(22500.0, 25000.0]	0	1	1	1	0

PAW_YN: At any time during 20..., even for one month, did... receive an CASH assistance from a state or county welfare program such as (State program name fill)?

Universe: All Persons aged 15+

Niu 2,431 10,167 1,488 1,160 17,629

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Yes	24	470	27	113	38
No	12,580	28,108	12,726	8,222	62,498
PENINCL: Was ... included in that plan?					
Universe: PENPLAN = 1					
Niu	12,999	36,775	12,935	7,709	54,529
Yes	1,334	996	775	1,381	21,824
No	702	974	531	405	3,812
PENPLAN: Other than social security did the employer or union that ... worked for in 20.. have a pension or other type of retirement plan?					
Universe: WRK_CK = 1					
Niu	6,201	29,895	8,039	5,508	27,806
Yes	2,036	1,970	1,306	1,786	25,636
No	6,798	6,880	4,896	2,201	26,723
WICYN: Who received WIC?					
Universe: Adult female					

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Niu	10,363	30,214	11,865	8,177	56,383
Received WIC	207	717	59	110	390
Did not receive WIC	4,465	7,814	2,317	1,208	23,392
CHCARE_YN: Paid child care was needed for this child?					
Universe: Persons age 15+ with children					
Niu	12,604	28,578	12,753	8,335	62,536
Yes	361	1,381	252	233	4,405
No	2,070	8,786	1,236	927	13,224
CHELSEW_YN: Does this person have a child living outside the household?					
Universe: All persons aged 15+					
Niu	2,431	10,167	1,488	1,160	17,629
Yes	386	443	163	129	1,438
No	12,218	28,135	12,590	8,206	61,098
CHSP_VAL: What is the annual amount of child support paid?					
Universe: CHSP_YN = 1					

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(-99.999, 9999.9]	15,003	38,723	14,222	9,484	79,970
(9999.9, 19999.8]	26	19	14	7	141
(19999.8, 29999.7]	4	1	1	2	41
(29999.7, 39999.6]	1	1	4	0	5
(39999.6, 49999.5]	1	0	0	1	2
(49999.5, 59999.4]	0	0	0	1	1
(59999.4, 69999.3]	0	1	0	0	1
(69999.3, 79999.2]	0	0	0	0	1
(89999.1, 99999.0]	0	0	0	0	3
CHSP_YN: Is this person required to pay child support?					
Universe: CHELSEW_YN					
Niu	14,649	38,302	14,078	9,366	78,727
Yes	194	136	70	41	681
No	192	307	93	88	757
CSP_VAL: How much did ... receive in child support payments?					

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Universe: CHSP_YN = 1					
(-99.999, 9999.9]	15,010	38,682	14,215	9,484	79,977
(9999.9, 19999.8]	19	48	18	8	148
(19999.8, 29999.7]	5	10	5	1	23
(29999.7, 39999.6]	0	4	1	1	11
(39999.6, 49999.5]	1	0	1	1	2
(49999.5, 59999.4]	0	0	0	0	1
(69999.3, 79999.2]	0	0	1	0	0
(89999.1, 99999.0]	0	1	0	0	3
CSP_YN: Did ... receive child support payments?					
Universe: All Persons aged 15+					
Niu	2,431	10,167	1,488	1,160	17,629
Yes	201	560	112	136	1,080
No	12,403	28,018	12,641	8,199	61,456
ACTC_CRD: Additional child tax credit					

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Universe: Tax unit head or dependent filer					
(-11.1, 1110.0]	13,939	37,125	13,926	9,144	78,392
(1110.0, 2220.0]	534	804	153	168	833
(2220.0, 3330.0]	359	525	102	119	560
(3330.0, 4440.0]	153	215	45	42	256
(4440.0, 5550.0]	27	33	5	12	59
(5550.0, 6660.0]	17	29	8	8	41
(6660.0, 7770.0]	3	8	2	1	15
(7770.0, 8880.0]	2	4	0	0	4
(8880.0, 9990.0]	1	2	0	1	4
(9990.0, 11100.0]	0	0	0	0	1
AGI: Adjusted gross income					
Universe: Tax unit head or dependent filer					
(-12341.073, 224208.3]	14,924	38,542	13,917	9,179	77,141
(224208.3, 458415.6]	89	171	256	278	2,438

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(458415.6, 692622.9]	14	21	33	21	325
(692622.9, 926830.2]	4	5	16	4	98
(926830.2, 1161037.5]	4	5	11	9	87
(1161037.5, 1395244.8]	0	0	4	2	56
(1395244.8, 1629452.1]	0	1	1	2	7
(1629452.1, 1863659.4]	0	0	1	0	1
(1863659.4, 2097866.7]	0	0	1	0	6
(2097866.7, 2332074.0]	0	0	1	0	6
CTC_CRD: Child tax credit					
Universe: Tax unit head or dependent filer					
(-18.0, 1800.0]	13,956	38,047	13,477	8,913	69,728
(1800.0, 3600.0]	646	462	418	331	5,280
(3600.0, 5400.0]	327	186	250	182	3,845
(5400.0, 7200.0]	73	41	78	52	1,015
(7200.0, 9000.0]	26	8	15	15	236

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(9000.0, 10800.0]	5	1	2	2	40
(10800.0, 12600.0]	2	0	0	0	17
(12600.0, 14400.0]	0	0	0	0	2
(14400.0, 16200.0]	0	0	1	0	0
(16200.0, 18000.0]	0	0	0	0	2
EIT_CRED: Earn income tax credit					
Universe: Tax unit head or dependent filer					
(-6.557, 655.7]	13,787	36,710	13,872	9,134	78,356
(655.7, 1311.4]	106	159	45	40	348
(1311.4, 1967.1]	127	149	72	55	330
(1967.1, 2622.8]	153	229	44	46	281
(2622.8, 3278.5]	135	248	45	54	207
(3278.5, 3934.2]	263	420	62	60	266
(3934.2, 4589.9]	92	184	36	24	120
(4589.9, 5245.6]	88	152	20	26	86

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(5245.6, 5901.3]	168	306	28	39	117
(5901.3, 6557.0]	116	188	17	17	54
FED_RET: Federal retirement payroll deduction					
Universe: Tax unit head or dependent filer					
(-16.9, 1690.0]	15,032	38,744	14,241	9,491	80,153
(1690.0, 3380.0]	0	0	0	0	2
(3380.0, 5070.0]	1	1	0	0	2
(5070.0, 6760.0]	2	0	0	1	4
(6760.0, 8450.0]	0	0	0	0	1
(8450.0, 10140.0]	0	0	0	2	2
(10140.0, 11830.0]	0	0	0	0	1
(15210.0, 16900.0]	0	0	0	1	0
FEDTAX_AC: Federal income tax liability, after all credits					
Universe: Tax unit head or dependent filer					
(-10797.046, 69805.6]	15,001	38,684	14,139	9,415	79,276

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(69805.6, 149610.2]	22	49	66	62	605
(149610.2, 229414.8]	8	5	18	6	94
(229414.8, 309219.4]	2	3	7	4	62
(309219.4, 389024.0]	2	3	7	6	91
(389024.0, 468828.6]	0	1	1	2	23
(468828.6, 548633.2]	0	0	2	0	4
(628437.8, 708242.4]	0	0	1	0	6
(708242.4, 788047.0]	0	0	0	0	4
FEDTAX_BC: Federal income tax liability, before credits					
Universe: Tax unit head or dependent filer					
(-788.047, 78804.7]	15,006	38,696	14,150	9,434	79,411
(78804.7, 157609.4]	18	37	59	43	473
(157609.4, 236414.1]	7	5	14	6	96
(236414.1, 315218.8]	2	3	7	4	62
(315218.8, 394023.5]	2	3	7	6	90

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(394023.5, 472828.2]	0	1	1	2	19
(472828.2, 551632.9]	0	0	2	0	4
(630437.6, 709242.3]	0	0	1	0	6
(709242.3, 788047.0]	0	0	0	0	4
FICA: Social security retirement payroll deduction					
Universe: All persons					
(-55.449, 5544.9]	14,080	38,087	12,928	8,678	63,814
(5544.9, 11089.8]	821	521	979	661	14,090
(11089.8, 16634.7]	98	99	209	123	1,751
(16634.7, 22179.6]	23	29	85	19	287
(22179.6, 27724.5]	6	5	21	9	78
(27724.5, 33269.4]	5	4	13	5	134
(33269.4, 38814.3]	0	0	4	0	6
(38814.3, 44359.2]	1	0	2	0	1
(44359.2, 49904.1]	1	0	0	0	3

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(49904.1, 55449.0]	0	0	0	0	1
FILESTAT: Tax filer status					
Universe: All persons					
Joint, both <65	4,721	3,600	2,931	1,621	33,473
Joint, one ><65 & one 65+	235	1,045	692	782	1,812
Joint, both 65+	67	3,661	2,693	2,660	271
Head of household	764	1,485	350	299	3,024
Single	4,246	5,595	3,652	1,956	17,561
Non-filer	5,002	23,359	3,923	2,177	24,024
MARG_TAX: Marginal tax rate					
Universe: Tax unit head or dependent filer					
(-0.037, 3.7]	9,196	31,832	8,644	5,356	45,074
(7.4, 11.1]	1,801	2,645	1,229	717	3,139
(11.1, 14.8]	3,127	2,994	2,557	1,813	14,677
(18.5, 22.2]	687	920	1,267	1,088	11,655

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(22.2, 25.9]	174	259	404	403	4,335
(29.6, 33.3]	15	39	53	62	523
(33.3, 37.0]	35	56	87	56	762
PRSWKXPNS: Work expenses					
Universe: A_AGE > 17 or HHDFMX = 1,2,46, or 47					
(-2.065, 206.5]	6,481	30,475	8,279	5,658	29,096
(206.5, 413.0]	131	275	104	94	470
(413.0, 619.5]	175	312	141	101	591
(619.5, 826.0]	210	347	136	124	670
(826.0, 1032.5]	131	225	119	86	416
(1032.5, 1239.0]	352	504	210	178	879
(1239.0, 1445.5]	228	252	155	108	696
(1445.5, 1652.0]	292	336	238	161	1,100
(1652.0, 1858.5]	265	284	167	124	969
(1858.5, 2065.0]	6,770	5,735	4,692	2,861	45,278

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
STATETAX_A: State income tax liability, after all credits					
Universe: Tax unit head or dependent filer					
(-6490.585, 19727.5]	15,009	38,704	14,157	9,429	79,338
(19727.5, 45686.0]	20	37	63	54	637
(45686.0, 71644.5]	6	3	15	6	113
(71644.5, 97603.0]	0	0	2	6	35
(97603.0, 123561.5]	0	1	4	0	25
(123561.5, 149520.0]	0	0	0	0	10
(149520.0, 175478.5]	0	0	0	0	1
(175478.5, 201437.0]	0	0	0	0	3
(201437.0, 227395.5]	0	0	0	0	2
(227395.5, 253354.0]	0	0	0	0	1
STATETAX_B: State income tax liability, before credits					
Universe: Tax unit head or dependent filer					
(-253.354, 25335.4]	15,017	38,718	14,185	9,458	79,632

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(25335.4, 50670.8]	12	23	38	28	377
(50670.8, 76006.2]	6	3	12	4	83
(76006.2, 101341.6]	0	0	2	5	39
(101341.6, 126677.0]	0	1	4	0	18
(126677.0, 152012.4]	0	0	0	0	9
(152012.4, 177347.8]	0	0	0	0	1
(177347.8, 202683.2]	0	0	0	0	3
(202683.2, 228018.6]	0	0	0	0	2
(228018.6, 253354.0]	0	0	0	0	1

TAX_INC: Taxable income amount

Universe: Tax unit head or dependent filer

(-2298.214, 229821.4]	14,968	38,607	14,027	9,280	78,079
(229821.4, 459642.8]	49	112	153	185	1,604
(459642.8, 689464.2]	11	17	34	14	250
(689464.2, 919285.6]	5	4	10	4	78

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(919285.6, 1149107.0]	2	4	11	9	93
(1149107.0, 1378928.4]	0	1	3	3	45
(1378928.4, 1608749.8]	0	0	2	0	4
(1608749.8, 1838571.2]	0	0	0	0	1
(1838571.2, 2068392.6]	0	0	0	0	6
(2068392.6, 2298214.0]	0	0	1	0	5
PERLIS: Poverty level of persons (Subfamily members have primary family recode)					
Universe: All persons					
Not in poverty universe	29	173	9	37	46
Below poverty level	2,650	10,405	1,038	549	1,873
100 - 124 percent of the poverty level	872	3,558	448	302	898
125 - 149 percent of the poverty level	968	3,113	506	303	1,240
150 and above the poverty level	10,516	21,496	12,240	8,304	76,108
POV_UNIV: Poverty universe flag					

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
Universe: All persons					
Not in poverty universe	29	173	9	37	46
In poverty universe	15,006	38,572	14,232	9,458	80,119
HEA: Health status					
Universe: All persons					
Excellent	4,703	8,539	4,173	2,207	32,776
Very good	4,895	9,678	4,540	3,038	29,492
Good	4,164	11,856	3,859	2,899	15,028
Fair	1,039	6,158	1,247	1,007	2,439
Poor	234	2,514	422	344	430
SPM_ACTC: SPM units Additional Child Tax Credit					
Universe: All persons					
(-11.1, 1110.0]	11,509	28,742	13,080	8,266	72,935
(1110.0, 2220.0]	1,538	3,848	513	507	3,105
(2220.0, 3330.0]	1,172	3,423	362	420	2,227

Table 3.6: Number of survey participants by health factors and five insurance coverage combinations of enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) (continued)

Variable	Insurance Coverage Type (GRP, DIR, PUB)				
	NNN	NNY	NY_	Y1Y	YNN
(3330.0, 4440.0]	583	1,834	215	176	1,141
(4440.0, 5550.0]	111	393	26	55	337
(5550.0, 6660.0]	74	314	36	56	233
(6660.0, 7770.0]	25	111	9	2	116
(7770.0, 8880.0]	11	41	0	1	43
(8880.0, 9990.0]	9	32	0	12	15
(9990.0, 11100.0]	3	7	0	0	13

Code 3.8: Exploratory data analysis (describe.py)

```

1  import os
2  import pandas as pd
3  import warnings
4
5  from module.utility import create_dir, import_dict
6  from module.eda import *
7  from module.dataset import *
8  from cls.ThesisExtension import *
9
10 texlive_binpath = '/usr/local/texlive/2024/bin/x86_64-linux'
11 os.environ['PATH'] += os.pathsep + texlive_binpath
12
13 pd.set_option('display.max_columns', None)
14 pd.set_option('display.width', 1000)
15 warnings.filterwarnings('ignore')
16
17 # Given Information
18 dataset_name = "pppub20"
19
20 # Predefined Directories
21 meta_dir = "../../Data/Original/metadata"
22 feather_dir = "../../Data/Original/feather"
23 csv_dir = "../../Data/Original/csv"
24
25 output_dir = f"../../Outputs/Main/EDA/{dataset_name}"
26 log_dir = f"../../Logs/preprocessing"
27 log_filepath = f"{log_dir}/describe.log"
28
29 backup_dir = "../../Backups"
30
31 create_dir(log_dir)
32
33 # Data Preparation
34 indep_dict = import_dict(metadata_path=f"{meta_dir}/meta-indep.json")
35 dep_attrs = ['GRP', 'DIR', 'PUB']

```

```

36 print()
37 describe_var(indep_dict)
38 print()
39 df = import_dataset(dataset_name=dataset_name, feather_dir=feather_dir)
40 print()
41 dep_features = ['class_orig', 'code_orig', 'code', 'class']
42 acpt_types = {'category', 'int16', 'int32', 'int8', 'uint16', 'uint32', 'uint8'}
43 preprocess = True
44
45 if all(feats in df.columns for feat in dep_features):
46     col_types = set()
47     for col in df.columns:
48         col_types.add(str(df[col].dtype))
49     if col_types == acpt_types:
50         preprocess = False
51
52 if preprocess:
53     df.thesis.code(indep_dict, dep_attrs)
54     df.thesis.recode()
55
56 filepath_feather = f"{feather_dir}/{dataset_name}.feather"
57 filepath_csv = f"{csv_dir}/{dataset_name}.csv"
58
59 if not os.path.isfile(filepath_feather):
60     export_dataset(df, file_dir='data/feather', dataset_name=dataset_name,
61                    format='feather')
62
63 if not os.path.isfile(filepath_csv):
64     dfther = pd.read_feather(filepath_feather)
65     export_dataset(dfther, file_dir='data/csv', dataset_name=dataset_name,
66                    format='csv')
67
68 # Univariate Data Analysis
69 df.thesis.show_type(option='full')
70 print()

```

```

69 df[['GRP', 'DIR', 'PUB', 'class_orig', 'code_orig', 'code', 'class']].
    drop_duplicates().sort_values('class').reset_index(drop=True)
70 print(f"Code: Employment-based plan (GRP) | Direct-purchase plan (DIR) |
    Public health insurance (PUB)")
71 print(df.groupby('code').size())
72 print('\n'*2)
73
74 # Cross Tabulation Analysis
75 print("-----")
76 crosstab(df=df, indep_dict=indep_dict, cont_bins=10, plot=True, output_dir
    =output_dir, log_filepath=log_filepath, backup_dir=backup_dir)

```

3.5.7 Data Encoding

Code 3.9 encodes the input dataset in the correct format, zero for a continuous NIU (not in universe) value and 0 up to a positive integer for a categorical value, by instantiating the Data class defined in Code 3.6. The state of this instance is maintained by two attached attributes **dataset**, a pandas DataFrame extended by the **data** accessor, and **metadata**, a Python list. The nonstatic methods **encodecat** and **encodecont** for encoding categorical and continuous features change the object into multiple states. This dissertation excessively uses the shallow copies of attributes by calling the method **copy** to protect the originals. Unlike a deep copy, a shallow copy inserts reference to an original object to the extent possible.

Code 3.9: Data encoding (convert.py)

```

1 import os
2 import pandas as pd
3 import pyarrow
4
5 from module.utility import create_dir, import_dict, export_json,
    export_txt
6 from module.metaencode import *
7 from cls.Data import *
8
9 # Given Information
10 dataset_inname = "pppub20"

```

```

11 dataset_encname = f"{dataset_inname}enc"
12 dataset_procname = "proc20"
13
14 # Predefined Directories
15 meta_indir = "../.../Data/Original/metadata"
16 meta_extra_indir = f"{meta_indir}/extra"
17 feather_indir = "../.../Data/Original/feather"
18 csv_indir = "../.../Data/Original/csv"
19
20 meta_encdir = "../.../Data/Encoded/metadata"
21 meta_extra_encdir = f"{meta_encdir}/extra"
22 feather_encdir = "../.../Data/Encoded/feather"
23 csv_encdir = "../.../Data/Encoded/csv"
24 info_encdir = "../.../Data/Encoded/info"
25
26 csv_procdir = "../.../Data/Processed/csv"
27
28 create_dir(meta_extra_indir)
29 create_dir(feather_indir)
30 create_dir(csv_indir)
31 create_dir(meta_extra_encdir)
32 create_dir(feather_encdir)
33 create_dir(csv_encdir)
34 create_dir(info_encdir)
35 create_dir(csv_procdir)
36
37 # Metadata
38 indep_dict = import_dict(metadata_path=f"{meta_indir}/meta-indep.json")
39 export_json(extract_dict_cat(indep_dict), f"{meta_extra_indir}/meta-indep-
    cat.json")
40 export_json(extract_dict_cont(indep_dict), f"{meta_extra_indir}/meta-indep-
    cont.json")
41
42 # Imported Dataset
43 if os.path.isfile(f"{feather_indir}/{dataset_inname}.feather"):
44     df = pd.read_feather(f"{feather_indir}/{dataset_inname}.feather")

```

```

45     if not os.path.isfile(f"{csv_indir}/{dataset_inname}.csv"):
46         df.to_csv(f"{csv_indir}/{dataset_inname}.csv", index=False)
47     else:
48         df = pd.read_csv(f"{csv_indir}/{dataset_inname}.csv")
49
50     # Encoded Dataset and Dictionary
51     data_obj = Data(df.copy(), indep_dict.copy())
52     cat_var_change = data_obj.encodecat()
53     cont_var_nonpos = data_obj.encodecont()
54     df_enc = data_obj.dataset
55     indep_dict_enc = data_obj.metadata
56
57     # Processed Dataset
58     dep_attrs = ['GRP', 'DIR', 'PUB']
59     class_attrs = ['class_orig', 'code_orig', 'code', 'class']
60     df_proc_enc = df_enc.drop(columns=['COV']+dep_attrs+class_attrs)
61     df_proc_enc = sort_cols(df_proc_enc, indep_dict_enc).join(df_enc['class'])
62     df_proc_info = indep_info(df_proc_enc.loc[:, df_proc_enc.columns != 'class'],
63                               indep_dict_enc)
64
65     df_count_info = count_info(df_proc_info)
66
67     # Exported Results
68     df_enc.to_feather(f"{feather_encdir}/{dataset_encname}.feather")
69     df_enc.to_csv(f"{csv_encdir}/{dataset_encname}.csv", index=False)
70     export_json(
71         indep_dict_enc,
72         f"{meta_encdir}/meta-indep-{dataset_encname}.json"
73     )
74     export_json(
75         extract_dict_cat(indep_dict_enc),
76         f"{meta_extra_encdir}/meta-indep-cat-{dataset_encname}.json"
77     )
78     df_proc_enc.to_csv(f"{csv_procdir}/{dataset_procname}.csv", header=True,
79                        index=False)

```

```

79 df_proc_info.index = df_proc_info.index + 1
80 df_proc_info.to_csv(f"{info_encdir}/{dataset_encname}-info.csv",
    index_label="id")
81 df_count_info.to_csv(f"{info_encdir}/{dataset_encname}-countinfo.csv",
    header=True, index=False)
82
83 export_txt(cat_var_change, f"{meta_extra_encdir}/catchchange-{
    dataset_encname}.txt")
84 export_txt(cont_var_nonpos, f"{meta_extra_encdir}/contnonpos-{
    dataset_encname}.txt")

```

3.5.8 Sampling using SelectKBest

Because the classifier proposed in Chapter 4 is exponentially expensive, certain features are preselected by evaluating their scores against a target variable. Code 3.10 considers 3, 4 and 8 highest scores based on the mutual information for a discrete target. In addition, 100 out of 157,681 survey participants are sampled of equal class size by calling two methods `groupby` and `sample`. Due to its random nature, the sampling result changes in each call. The use of the model is illustrated in Chapter 5 with only three preselected features.

Code 3.10: SelectKBest (selectkbest.py)

```

1 import pandas as pd
2 from functools import partial
3 from sklearn.feature_selection import mutual_info_classif, SelectKBest
4
5 from module.utility import create_dir
6
7 sel_num_ls = [3, 4, 8]
8 train_eachclass_num = 20
9
10 data_filepath = "../../../Data/Processed/csv/proc20.csv"
11 info_filepath = "../../../Data/Encoded/info/pppub20enc-info.csv"
12
13 data_selname = "selproc20"
14 train_name = "seltrain20"

```



```

15 test_name = "seltest20"
16
17 # Predefined Directories
18 sample_dir = "../.../Samples/random"
19 sel_dir = f"{sample_dir}/{data_selname}"
20
21 data_dir = f"{sel_dir}/data"
22 info_dir = f"{sel_dir}/info"
23 feat_dir = f"{sel_dir}/features"
24 score_dir = f"{sel_dir}/scores"
25 train_dir = f"{sel_dir}/train"
26 test_dir = f"{sel_dir}/test"
27
28 create_dir(data_dir)
29 create_dir(info_dir)
30 create_dir(feat_dir)
31 create_dir(score_dir)
32 create_dir(train_dir)
33 create_dir(test_dir)
34
35 # Univariate Feature Selection
36 def feat_select(df_indata, df_info, sel_num):
37     discrete_feat_idx = df_info.index[df_info['type']=='Categorical']
38     score_func = partial(mutual_info_classif, discrete_features=
39         discrete_feat_idx)
40     feat_selector = SelectKBest(score_func, k=sel_num)
41     feat_selector.fit(df_indata.drop('class', axis=1), df_indata['class'])
42
43     df_scores = pd.DataFrame()
44     df_scores["Attribute"] = df_indata.drop('class', axis=1).columns
45     df_scores['Type'] = df_info['type']
46     df_scores["Support"] = feat_selector.get_support()
47     df_scores["F Score"] = feat_selector.scores_
48     df_scores["P Value"] = feat_selector.pvalues_

```

```

49     df_selffeat = df_scores[df_scores['Support']].drop('Support', axis=1).
        reset_index(drop=True)
50     df_seldata = df_indata[df_selffeat['Attribute']].join(df_indata['class'
        ])
51
52     minmax = df_seldata.loc[:, df_seldata.columns != 'class'].agg(['min', '
        max']).values.tolist()
53     df_selffeat['Min'] = minmax[0]
54     df_selffeat['Max'] = minmax[1]
55     del minmax
56
57     return df_seldata, df_selffeat, df_scores
58
59 # Implementation
60 df_indata = pd.read_csv(data_filepath)
61 df_info = pd.read_csv(info_filepath)
62
63 print(f"\n{df_indata.head()}\n")
64 print(f"{df_info.head()}\n")
65
66 for sel_num in sel_num_ls:
67
68     # Univariate feature selection
69     df_seldata, df_selffeat, df_scores = feat_select(df_indata=df_indata,
        df_info=df_info, sel_num=sel_num)
70
71     # Display results (selected features)
72     print(f"Select {sel_num} features:\n")
73     print(f"{df_selffeat}\n")
74
75     # Train-test split
76     df_seltrain = df_seldata.groupby('class', group_keys=False).apply(
77         lambda x: x.sample(train_eachclass_num)
78     )
79     df_seltest = df_seldata.drop(df_seltrain.index)
80

```

```

81     # Exported results
82     df_seldata.to_csv(f"{data_dir}/{data_selname}num{sel_num}.csv", header=
        True, index=False)
83
84     df_selfeat.to_csv(f"{feat_dir}/fnum{sel_num}.csv", header=True, index=
        False)
85     df_scores.to_csv(f"{score_dir}/snum{sel_num}.csv", header=True, index=
        False)
86
87     df_selfeat.index = df_selfeat.index + 1
88     df_selinfo = df_selfeat.drop(['F Score', 'P Value'], axis=1)
89     df_selinfo.columns = ['variable', 'type', 'min', 'max']
90     df_selinfo.to_csv(f"{info_dir}/{data_selname}num{sel_num}info.csv",
        index_label='id')
91
92     df_seltrain.to_csv(f"{train_dir}/{train_name}num{sel_num}each{
        train_eachclass_num}.csv", header=True, index=False)
93     df_seltest.to_csv(f"{test_dir}/{test_name}num{sel_num}exc{
        train_eachclass_num}.csv", header=True, index=False)

```

3.5.9 Setting Number of Variable Splits

Provided that two and three splits or cuts are of interest, Code 3.11 determines an appropriate number of splits on an individual feature in the health insurance dataset of all noninfant survey participants with full features and previously preselected 3, 4 and 8 features. For example, in the case of three splits, up to two splits are allowed on the feature SS_YN representing the answer, including NIU (not in universe), to the yes/no question regarding social security payments. The column of these numbers is inserted into the DataFrame as an additional information directly through the pandas accessor `info` in Code 3.7 without explicit class instantiation.

Code 3.11: Setting number of variable splits (setcut.py)

```

1  import pandas as pd
2
3  from module.utility import create_dir
4  from cls.Info import *

```

```

5
6 # Given Information
7 pcut_ls = [2, 3]
8 info_ls = []
9 info_ls.append({
10     'indir': "../../../Data/Encoded/info",
11     'infile': "pppub20enc-info.csv",
12     'outdir': "../../../Samples/proc20/cuts"
13 })
14 extra_infile_ls = [
15     "selproc20num3info.csv",
16     "selproc20num4info.csv",
17     "selproc20num8info.csv"
18 ]
19 for file in extra_infile_ls:
20     info_ls.append({
21         'indir': "../../../Samples/selproc20/info",
22         'infile': file,
23         'outdir': "../../../Samples/selproc20/cuts"
24     })
25 print(f"\n{info_ls}\n")
26
27 # Implementation
28 for dc in info_ls:
29     for pcut in pcut_ls:
30
31         # Import
32         inpath = f"{dc['indir']}/{dc['infile']}"
33         df = pd.read_csv(inpath)
34
35         # Set cuts
36         pcont, pcatmax = pcut, pcut
37         df.info.setcut(pcont, pcatmax)
38
39         # Set output path

```

```
40     infilename = dc['infile'].replace('.csv', '').replace('info', '').  
        replace('-', '')  
41     cutfilename = f"{infilename}co{pcont}ca{pcatmax}cutinfo"  
42     outpath = f"{dc['outdir']}/{cutfilename}.csv"  
43  
44     # Display results  
45     print(f"Input: {inpath}")  
46     print(f"NUmber of features: {len(df)}")  
47     print(f"Number of continuous cuts: {pcont}")  
48     print(f"Number of maximum categorical cuts: {pcatmax}")  
49     print(f"Output: {outpath}\n")  
50     print(f"{df.head()}\n")  
51  
52     # Export  
53     create_dir(dc['outdir'])  
54     df.to_csv(outpath, header=True, index=False)
```

CHAPTER IV

PROPOSED CLASSIFIER

4.1 Proposed Model for Selecting Continuous Factors

Suppose a training dataset of dimension \tilde{d} excluding its target variable has N instances, and every feature $1 \leq \tilde{j} \leq \tilde{d}$ is continuous. Each training instance $\tilde{x}^i = (\tilde{x}_{\tilde{j}}^i)_{1 \leq \tilde{j} \leq \tilde{d}} \in \mathbb{R}^{\tilde{d}}$ where $1 \leq i \leq N$ has an integer class label between 0 and n . Let y_k^i specify whether a training instance \tilde{x}^i is in class k for $0 \leq k \leq n$. Assume that at most $1 \leq d \leq \tilde{d}$ contributing factors are considered. It follows that a reduced instance $x^i = (x_j^i)_{1 \leq j \leq d} \in \mathbb{R}^d$ is a partial selection of the components of the original instance \tilde{x}^i :

$$\begin{aligned} x_j^i &= \sum_{\tilde{j}=1}^{\tilde{d}} c_{j,\tilde{j}} \tilde{x}_{\tilde{j}}^i \\ \sum_{\tilde{j}=1}^{\tilde{d}} c_{j,\tilde{j}} &\leq 1 \\ \sum_{\tilde{j}=1}^{\tilde{d}} c_{j,\tilde{j}} &\leq 1 \\ c_{j,\tilde{j}} &\in \{0, 1\}. \end{aligned}$$

An original feature \tilde{j} is selected and considered significant when

$$\sum_{\tilde{j}=1}^{\tilde{d}} c_{j,\tilde{j}} = 1$$

and it becomes a new feature j , uniquely, for $c_{j,\tilde{j}} = 1$.

Every selected, rearranged feature $1 \leq j \leq d$ is assumed to have $p_j \geq 0$ splitting values: $b_{j,1} \leq \dots \leq b_{j,p_j}$. Two endpoints are assumed: $b_{j,0} = -M$ and $b_{j,p_j+1} = M$ for sufficiently large positive M such as $\max\{|x_j^i|\}$. All splitting points along each new axis forms $B = (p_1 + 1) \cdots (p_d + 1)$ decision boxes. A box S_β is defined in the following manner:

$$S_\beta = \prod_{j=1}^d \sum_{q=0}^{p_j} \beta_{j,q} [b_{j,q}, b_{j,q+1}]$$

where $b_{j,0}$ and b_{j,p_j+1} are sufficiently small negative and large positive,

$$\begin{aligned}\beta &= \sum_{j=1}^d \left[\prod_{j_0=0}^{j-1} (p_{j_0} + 1) \right] \left[\sum_{q=0}^{p_j} q \beta_{j,q} \right] \\ \sum_{q=0}^{p_j} \beta_{j,q} &= 1 \\ \beta_{j,q} &\in \{0, 1\}\end{aligned}$$

and $p_0 = 1$.

Each $x_j^i \in \mathbb{R}$ is in an open interval $(b_{j,q}, b_{j,q+1})$ for some $0 \leq q \leq p_j$, and its existence is indicated by a boolean variable $\alpha_{j,q}^i$:

$$\begin{aligned}\sum_{j=1}^d c_{j,j} \tilde{x}_j^i &= x_j^i \in \sum_{q=0}^{p_j} \alpha_{j,q}^i [b_{j,q} + m_j, b_{j,q+1} - m_j] = \sum_{q=0}^{p_j} [l_{j,q}^i, r_{j,q}^i] \\ \sum_{q=0}^{p_j} \alpha_{j,q}^i &= 1 \\ \alpha_{j,q}^i &\in \{0, 1\}\end{aligned}$$

for sufficiently small positive m_j such as

$$m_j = \frac{1}{2} \min\{|x_j^{i_1} - x_j^{i_2}| : x_j^{i_1} \neq x_j^{i_2}\}$$

and for some $l_{j,q}^i$ and $r_{j,q}^i$. Both terms are introduced to linearize the nonlinear products $\alpha_{j,q}^i(b_{j,q} + m_j)$ and $\alpha_{j,q}^i(b_{j,q+1} - m_j)$ respectively. Proven constructively, Theorem 4.1 ensures the linearizability.

Theorem 4.1. Two intervals $\alpha_{j,q}^i [b_{j,q} + m_j, b_{j,q+1} - m_j]$ and $[l_{j,q}^i, r_{j,q}^i]$ are identical only when

$$\begin{aligned}l_{j,q}^i &\in [-M, b_{j,q} + m_j] + M(1 - \alpha_{j,q}^i) \\ l_{j,q}^i &\in [b_{j,q} + m_j, M] - M(1 - \alpha_{j,q}^i) \\ r_{j,q}^i &\in [-M, b_{j,q+1} - m_j] + M(1 - \alpha_{j,q}^i) \\ r_{j,q}^i &\in [b_{j,q+1} - m_j, M] - M(1 - \alpha_{j,q}^i).\end{aligned}$$

Proof. It suffices to show that $l_{j,q}^i = \alpha_{j,q}^i(b_{j,q} + m_j)$ under the given constraints because substitution $b_{j,q}$ and m_j with $b_{j,q+1}$ and $-m_j$ results in the expression for $r_{j,q}^i$. The equivalent condition for the nonlinear product is given by for sufficiently large positive M_1 , M_2 , M_3 and M_4

$$l_{j,q}^i = \begin{cases} 0, & \text{for } \alpha_{j,q}^i = 0 \\ b_{j,q} + m_j, & \text{for } \alpha_{j,q}^i = 1 \end{cases}$$

$$\in \begin{cases} [-M_1, 0] \cap [0, M_2], & \text{for } \alpha_{j,q}^i = 0 \\ [b_{j,q} + m_j, M_3] \cap [-M_4, b_{j,q} + m_j], & \text{for } \alpha_{j,q}^i = 1. \end{cases}$$

Consider how each interval changes when $\alpha_{j,q}^i$ moves from 0 to 1:

$$[b_{j,q} + m_j, M_3] = [-M_1, 0] + [b_{j,q} + m_j + M_1, M_3]$$

$$[-M_4, b_{j,q} + m_j] = [0, M_2] + [-M_4, b_{j,q} + m_j - M_2].$$

Hence the translations are given by $(1 - \alpha_{j,q}^i)[b_{j,q} + m_j + M_1, M_3]$ and $(1 - \alpha_{j,q}^i)[-M_4, b_{j,q} + m_j - M_2]$. To remove all nonlinear terms, choose M_1 and M_2 such that $b_{j,q} + m_j + M_1$ and $b_{j,q} + m_j - M_2$ are constant. One example of such the ordered tuple (M_1, M_2, M_3, M_4) is $(M - b_{j,q} - m_j, M + b_{j,q} + m_j, M, M)$. \square

Governed by a boolean variable γ_β^i , an instance $x^i \in \mathbb{R}^d$ is also located in one of these boxes labeled by $0 \leq \beta \leq B - 1$:

$$\sum_{j=1}^d \left[\prod_{j_0=0}^{j-1} (p_{j_0} + 1) \right] \left[\sum_{q=0}^{p_j} q \alpha_{j,q}^i \right] = \sum_{\beta=0}^{B-1} \beta \gamma_\beta^i$$

$$\sum_{\beta=0}^{B-1} \gamma_\beta^i = 1$$

$$\gamma_\beta^i \in \{0, 1\}.$$

By majority voting, a decision box β therefore predicts exactly one class label from the following set

$$\Theta_\beta = \operatorname{argmax}_{0 \leq k \leq n} \left\{ \sum_{i=1}^N y_k^i \gamma_\beta^i \right\}.$$

In total, there are

$$N - \sum_{\beta=0}^{B-1} \max_{0 \leq k \leq n} \left\{ \sum_{i=1}^N y_k^i \gamma_\beta^i \right\} = N + \sum_{\beta=0}^{B-1} \min_{0 \leq k \leq n} \left\{ - \sum_{i=1}^N y_k^i \gamma_\beta^i \right\}$$

misclassified instances.

Theorem 4.2. The optimal value of the program

$$\begin{aligned}
 & \text{minimize} && h_\beta \\
 & \text{subject to} && h_\beta + \sum_{i=1}^N y_k^i \gamma_\beta^i + N z_{\beta,k} \geq 0, \\
 & && \sum_{k=0}^n z_{\beta,k} = n, \\
 & && z_{\beta,k} \in \{0, 1\}
 \end{aligned}$$

is given by

$$\min_{0 \leq k \leq n} \left\{ - \sum_{i=1}^N y_k^i \gamma_\beta^i \right\}.$$

Proof. Let \mathcal{P} be the original problem. It can be partitioned into $n + 1$ subproblems, each of which \mathcal{P}_{k_0} for $0 \leq k_0 \leq n$ has the following restriction:

$$z_{\beta,k} = \begin{cases} 0, & \text{for } k = k_0 \\ 1, & \text{for } k \neq k_0. \end{cases}$$

For each subproblem \mathcal{P}_{k_0} ,

$$h_\beta \geq - \sum_{i=1}^N y_{k_0}^i \gamma_\beta^i = 0 - \sum_{i=1}^N y_{k_0}^i \gamma_\beta^i \geq - \sum_{i=1}^N y_k^i \gamma_\beta^i - N z_{\beta,k}$$

and this implies

$$\min(\mathcal{P}_{k_0}) = - \sum_{i=1}^N y_{k_0}^i \gamma_\beta^i.$$

Hence

$$\min(\mathcal{P}) = \min_{0 \leq k_0 \leq n} (\min(\mathcal{P}_{k_0})) = \min_{0 \leq k_0 \leq n} \left\{ - \sum_{i=1}^N y_{k_0}^i \gamma_\beta^i \right\}.$$

□

By Theorems 4.1 and 4.2, the selection model for continuous dataset is given by

$$\begin{aligned}
 & \text{minimize} && \sum_{\beta=0}^{B-1} h_\beta \\
 & \text{subject to} && \sum_{\tilde{j}=1}^{\tilde{d}} c_{j,\tilde{j}} \leq 1, \\
 & && \sum_{j=1}^d c_{j,\tilde{j}} \leq 1, \\
 & && b_{j,q+1} - b_{j,q} \geq 0,
 \end{aligned}$$

$$\begin{aligned}
& \sum_{j=1}^d \tilde{x}_{j,\tilde{j}}^i c_{j,\tilde{j}} - \sum_{q=0}^{p_j} l_{j,q}^i \geq 0, \\
& \sum_{j=1}^d \tilde{x}_{j,\tilde{j}}^i c_{j,\tilde{j}} - \sum_{q=0}^{p_j} r_{j,q}^i \leq 0, \\
& l_{j,q}^i + M\alpha_{j,q}^i \geq 0, \\
& l_{j,q}^i - M\alpha_{j,q}^i \leq 0, \\
& l_{j,q}^i - b_{j,q} + M\alpha_{j,q}^i \leq M + m_j, \\
& l_{j,q}^i - b_{j,q} - M\alpha_{j,q}^i \geq -M + m_j, \\
& r_{j,q}^i + M\alpha_{j,q}^i \geq 0, \\
& r_{j,q}^i - M\alpha_{j,q}^i \leq 0, \\
& r_{j,q}^i - b_{j,q+1} + M\alpha_{j,q}^i \leq M - m_j, \\
& r_{j,q}^i - b_{j,q+1} - M\alpha_{j,q}^i \geq -M - m_j, \\
& \sum_{j=1}^d \left[\prod_{j_0=0}^{j-1} (p_{j_0} + 1) \right] \left[\sum_{q=0}^{p_j} q\alpha_{j,q}^i \right] - \sum_{\beta=0}^{B-1} \beta\gamma_{\beta}^i = 0, \\
& \sum_{q=0}^{p_j} \alpha_{j,q}^i = 1, \\
& \sum_{\beta=0}^{B-1} \gamma_{\beta}^i = 1, \\
& h_{\beta} + \sum_{i=1}^N y_k^i \gamma_{\beta}^i + Nz_{\beta,k} \geq 0, \\
& \sum_{k=0}^n z_{\beta,k} = n, \\
& l_{j,q}^i, r_{j,q}^i, b_{j,q}, h_{\beta} \in \mathbb{R}, \\
& c_{j,\tilde{j}}, \alpha_{j,q}^i, \gamma_{\beta}^i, z_{\beta,k} \in \{0, 1\}
\end{aligned}$$

where the artificial splitting values $b_{j,0}$ and b_{j,p_j+1} are also treated as decision variables, and it produces a training accuracy of

$$1 + \frac{\sum_{\beta=0}^{B-1} h_{\beta}^*}{N} \leq 1.$$

4.2 Selection of Mixed-Type Features

More generally, a training instance $\tilde{x}^i \in \mathbb{R}^{\tilde{d}}$ has a mixed-type component $\tilde{x}_j^i \in \mathbb{R}$ in feature \tilde{j} . The index sets of continuous and categorical features are denoted by $\tilde{\mathcal{C}}_{\text{cont}}$ and $\tilde{\mathcal{C}}_{\text{cat}}$ where

$$\tilde{\mathcal{C}}_{\text{cont}} \cup \tilde{\mathcal{C}}_{\text{cat}} = \{1, 2, \dots, \tilde{d}\}.$$

The continuous features are initially selected, whereas all categorical features are kept. The latter will be subsequently selected. The sets $\mathcal{C}_{\text{cont}}$ and \mathcal{C}_{cat} represent new continuous and intermediate categorical components respectively where

$$\begin{aligned} |\mathcal{C}_{\text{cont}}| &\leq |\tilde{\mathcal{C}}_{\text{cont}}| \\ |\mathcal{C}_{\text{cat}}| &= |\tilde{\mathcal{C}}_{\text{cat}}| \\ \mathcal{C}_{\text{cont}} \cup \mathcal{C}_{\text{cat}} &= \{1, 2, \dots, d\}. \end{aligned}$$

These conditions above can be satisfied specifically, as illustrated on the health insurance dataset in Chapter 5, when $\mathcal{C}_{\text{cont}} \subseteq \tilde{\mathcal{C}}_{\text{cont}}$ and $\mathcal{C}_{\text{cat}} = \tilde{\mathcal{C}}_{\text{cat}}$, for instance. In the case of continuous data type, the constraints of feature selection become

$$\begin{aligned} x_j^i &= \sum_{\tilde{j} \in \tilde{\mathcal{C}}_{\text{cont}}} c_{j,\tilde{j}} \tilde{x}_{\tilde{j}}^i, & j \in \mathcal{C}_{\text{cont}} \\ \sum_{\tilde{j} \in \tilde{\mathcal{C}}_{\text{cont}}} c_{j,\tilde{j}} &\leq 1, & j \in \mathcal{C}_{\text{cont}} \\ \sum_{j \in \mathcal{C}_{\text{cont}}} c_{j,\tilde{j}} &\leq 1, & \tilde{j} \in \tilde{\mathcal{C}}_{\text{cont}} \\ c_{j,\tilde{j}} &\in \{0, 1\}, & (j, \tilde{j}) \in \mathcal{C}_{\text{cont}} \times \mathcal{C}_{\text{cont}}. \end{aligned}$$

Since at most $|\mathcal{C}_{\text{cont}}|$ out of $|\tilde{\mathcal{C}}_{\text{cont}}|$ continuous features are selected, the following condition holds:

$$\sum_{(j,\tilde{j}) \in \mathcal{C}_{\text{cont}} \times \tilde{\mathcal{C}}_{\text{cont}}} c_{j,\tilde{j}} \leq |\mathcal{C}_{\text{cont}}|.$$

A selected, rearranged component $x_j^i \in \mathbb{R}$ for a feature $1 \leq j \leq d$ is now either continuous or categorical. A continuous feature $j \in \mathcal{C}_{\text{cont}}$ is similarly assumed to have p_j splitting points, namely $b_{j,q} \in \mathbb{R}$ where $1 \leq q \leq p_j$. Usually, p_j is assumed to be constant across all new continuous features because the new explicit order of this selection is unknown before optimization. A categorical feature $j \in \mathcal{C}_{\text{cat}}$ comprises finite discrete values which are also assumed to form $p_j + 1$ new small groups labeled with $0 \leq u_j \leq p_j$.

A box $0 \leq \beta \leq B - 1$ along a categorical feature, as opposed to a continuous feature, lacks continuity because its entry is simply a singleton. Algebraically, it is represented by a set

$$S_\beta = \prod_{j \in \mathcal{C}_{\text{cont}}} \sum_{q=0}^{p_j} \beta_{j,q} [b_{j,q}, b_{j,q+1}] \times \prod_{j \in \mathcal{C}_{\text{cat}}} \{u_j\}$$

where

$$\begin{aligned} \beta &= \sum_{j \in \mathcal{C}_{\text{cont}}} \left[\prod_{0 \leq j_0 < j} (p_{j_0} + 1) \right] \left[\sum_{q=0}^{p_j} q \beta_{j,q} \right] \\ &\quad + \sum_{j \in \mathcal{C}_{\text{cat}}} \left[\prod_{0 \leq j_0 < j} (p_{j_0} + 1) \right] u_j \\ \sum_{q=0}^{p_j} \beta_{j,q} &= 1, & j \in \mathcal{C}_{\text{cont}} \\ \beta_{j,q} &\in \{0, 1\}, & j \in \mathcal{C}_{\text{cont}} \\ u_j &\in \{0, 1, \dots, p_j\}, & j \in \mathcal{C}_{\text{cat}} \end{aligned}$$

and $p_0 = 0$. The existence of $b_{j,0}$ and b_{j,p_j+1} where $j \in \mathcal{C}_{\text{cat}}$ is shown in the previous section. Numerically, each box can also be identified by the unique combination of binary $(\beta_{j,q})_{j \in \mathcal{C}_{\text{cont}}}$ and integer $(u_j)_{j \in \mathcal{C}_{\text{cat}}}$.

For a categorical feature $j \in \mathcal{C}_{\text{cat}}$, an original categorical label $x_j^i \in \mathbb{R}$ is reassigned to a new integer group label $0 \leq v_{j,x_j^i} \leq p_j$. As a result, the following conditions must hold:

$$\begin{aligned} \sum_{\beta=0}^{B-1} \beta \gamma_\beta^i &= \sum_{j \in \mathcal{C}_{\text{cont}}} \left[\prod_{0 \leq j_0 < j} (p_{j_0} + 1) \right] \left[\sum_{q=0}^{p_j} q \alpha_{j,q}^i \right] \\ &\quad + \sum_{j \in \mathcal{C}_{\text{cat}}} \left[\prod_{0 \leq j_0 < j} (p_{j_0} + 1) \right] v_{j,x_j^i} \\ \sum_{q=0}^{p_j} \alpha_{j,q}^i &= 1, & j \in \mathcal{C}_{\text{cont}} \\ \sum_{\beta=0}^{B-1} \gamma_\beta^i &= 1, \\ \beta_{j,q} &\in \{0, 1\}, & j \in \mathcal{C}_{\text{cont}} \\ v_{j,x_j^i} &\in \{0, 1, \dots, p_j\}, & j \in \mathcal{C}_{\text{cat}}. \end{aligned}$$

A boolean variable $f_j \in \{0, 1\}$ is defined to determine whether a categorical feature j is significant. All categorical labels of an insignificant feature are grouped together. Its necessary, though insufficient, condition can be obtained:

$$-M f_j \leq v_{j,x_j^i} \leq M f_j.$$

If at most d_{cat} out of $|\mathcal{C}_{\text{cat}}|$ categorical features are of interest, the following condition holds:

$$\sum_{j \in \mathcal{C}_{\text{cat}}} f_j \leq d_{\text{cat}}.$$

There are at most $|\mathcal{C}_{\text{cont}}| + d_{\text{cat}} \leq d \leq \tilde{d}$ contributing factors, $|\mathcal{C}_{\text{cont}}| \leq |\tilde{\mathcal{C}}_{\text{cont}}|$ of which are continuous and $d_{\text{cat}} \leq |\mathcal{C}_{\text{cat}}| = |\tilde{\mathcal{C}}_{\text{cat}}|$ categorical:

$$\sum_{(j, \tilde{j}) \in \mathcal{C}_{\text{cont}} \times \tilde{\mathcal{C}}_{\text{cont}}} c_{j, \tilde{j}} + \sum_{j \in \mathcal{C}_{\text{cat}}} f_j \leq d.$$

A continuous feature $\tilde{j} \in \tilde{\mathcal{C}}_{\text{cont}}$ is deemed significant when

$$\sum_{j \in \mathcal{C}_{\text{cont}}} c_{j, \tilde{j}} = 1,$$

and for an original categorical feature $\tilde{j} \in \tilde{\mathcal{C}}_{\text{cat}}$ corresponding to $j \in \mathcal{C}_{\text{cat}}$ a new group label v_{j, x_j^i} is nonconstant across all training instances x^i .

The final selection model is proposed:

$$\text{minimize} \quad \sum_{\beta=0}^{B-1} h_{\beta}$$

subject to

$$\begin{aligned} \sum_{\tilde{j} \in \tilde{\mathcal{C}}_{\text{cont}}} c_{j, \tilde{j}} &\leq 1, & j \in \mathcal{C}_{\text{cont}}, \\ \sum_{j \in \mathcal{C}_{\text{cont}}} c_{j, \tilde{j}} &\leq 1, & j \in \tilde{\mathcal{C}}_{\text{cont}}, \\ b_{j, q+1} - b_{j, q} &\geq 0, & j \in \mathcal{C}_{\text{cont}}, \\ \sum_{\tilde{j} \in \tilde{\mathcal{C}}_{\text{cont}}} \tilde{x}_j^i c_{j, \tilde{j}} - \sum_{q=0}^{p_j} l_{j, q}^i &\geq 0, & j \in \mathcal{C}_{\text{cont}}, \\ \sum_{\tilde{j} \in \tilde{\mathcal{C}}_{\text{cont}}} \tilde{x}_j^i c_{j, \tilde{j}} - \sum_{q=0}^{p_j} r_{j, q}^i &\leq 0, & j \in \mathcal{C}_{\text{cont}}, \\ l_{j, q}^i + M \alpha_{j, q}^i &\geq 0, & j \in \mathcal{C}_{\text{cont}}, \\ l_{j, q}^i - M \alpha_{j, q}^i &\leq 0, & j \in \mathcal{C}_{\text{cont}}, \\ l_{j, q}^i - b_{j, q} + M \alpha_{j, q}^i &\leq M + m_j, & j \in \mathcal{C}_{\text{cont}}, \\ l_{j, q}^i - b_{j, q} - M \alpha_{j, q}^i &\geq -M + m_j, & j \in \mathcal{C}_{\text{cont}}, \\ r_{j, q}^i + M \alpha_{j, q}^i &\geq 0, & j \in \mathcal{C}_{\text{cont}}, \\ r_{j, q}^i - M \alpha_{j, q}^i &\leq 0, & j \in \mathcal{C}_{\text{cont}}, \\ r_{j, q}^i - b_{j, q+1} + M \alpha_{j, q}^i &\leq M - m_j, & j \in \mathcal{C}_{\text{cont}}, \\ r_{j, q}^i - b_{j, q+1} - M \alpha_{j, q}^i &\geq -M - m_j, & j \in \mathcal{C}_{\text{cont}}, \end{aligned}$$

$$\begin{aligned}
& \sum_{j \in \mathcal{C}_{\text{cont}}} \left[\prod_{0 \leq j_0 < j} (p_{j_0} + 1) \right] \left[\sum_{q=0}^{p_j} q \alpha_{j,q}^i \right] \\
& + \sum_{j \in \mathcal{C}_{\text{cat}}} \left[\prod_{0 \leq j_0 < j} (p_{j_0} + 1) \right] v_{j,x_j^i} \\
& - \sum_{\beta=0}^{B-1} \beta \gamma_{\beta}^i = 0, \\
& \sum_{q=0}^{p_j} \alpha_{j,q}^i = 1, \quad j \in \mathcal{C}_{\text{cont}}, \\
& v_{j,x_j^i} + M f_j \geq 0, \quad j \in \mathcal{C}_{\text{cat}}, \\
& v_{j,x_j^i} - M f_j \leq 0, \quad j \in \mathcal{C}_{\text{cat}}, \\
& \sum_{(j,\tilde{j}) \in \mathcal{C}_{\text{cont}} \times \tilde{\mathcal{C}}_{\text{cont}}} c_{j,\tilde{j}} + \sum_{j \in \mathcal{C}_{\text{cat}}} f_j \leq d, \\
& \sum_{\beta=0}^{B-1} \gamma_{\beta}^i = 1, \\
& h_{\beta} + \sum_{i=1}^N y_k^i \gamma_{\beta}^i + N z_{\beta,k} \geq 0, \\
& \sum_{k=0}^n z_{\beta,k} = n, \\
& l_{j,q}^i, r_{j,q}^i, b_{j,q} \in \mathbb{R}, \quad j \in \mathcal{C}_{\text{cont}}, \\
& h_{\beta} \in \mathbb{R}, \\
& c_{j,\tilde{j}} \in \{0, 1\}, \quad (j, \tilde{j}) \in \mathcal{C}_{\text{cont}} \times \tilde{\mathcal{C}}_{\text{cont}}, \\
& \alpha_{j,q}^i \in \{0, 1\}, \quad j \in \mathcal{C}_{\text{cont}}, \\
& f_j \in \{0, 1\}, \quad j \in \mathcal{C}_{\text{cat}}, \\
& v_{j,x_j^i} \in \{0, 1, \dots, p_j\}, \quad j \in \mathcal{C}_{\text{cat}}, \\
& \alpha_{j,q}^i, \gamma_{\beta}^i, z_{\beta,k} \in \{0, 1\}.
\end{aligned}$$

4.3 CPLEX OPL Modeling

The proposed classifier heavily relies on 0-1 mixed integer programming (MIP). The CPLEX optimizer (version 22.1.1) is used to solve for the classifier including its splitting values and the set of predicted class labels in each decision box. Although achieving higher performance, manual adjustment of internal optimization procedures such as a node selection during branching and a combination of multiple techniques in cut generation is beyond the scope of this dissertation. The MIP problem is very large, and its information is stored in a huge tree data structure. Multiple lock-free nodes can be executed simultaneously in parallel by utilizing all available CPU cores. CPLEX uses in-memory computation.

When a central memory is consumed more than its upper limit which is 2048 MB by default, some nodes are transferred from the in-memory set to node files which are also in memory and compressed by default. Optionally, they can be flushed to disk, in either uncompressed or compressed form, where speed is sacrificed for more storage space. As more solutions are explored, the branch-and-cut tree grows larger. When its size exceeds its upper limit, which is set at 10^{75} MB by default, the optimization process terminates. The solver also stops when a memory is exhausted or a disk is fully occupied depending on whether node files are stored in memory or on disk. CPLEX parameters related to this dissertation is included in Table 4.1.

Table 4.1: Relevant CPLEX parameters

Parameter	Description
<code>cplex.intsollim</code>	MIP solution number limit
<code>cplex.tilim</code>	Time limit per optimizer call (in seconds)
<code>cplex.threads</code>	Parallel threads (default: 0 implying up to 32 threads)
<code>cplex.workmem</code>	Working memory before compression and swap (in MB) (default: 2048)
<code>cplex.treelim</code>	Uncompressed tree limit (in MB) (default: 10^{75})
<code>cplex.nodefileind</code>	Node storage file switch 0: No node file 1: Node file in memory and compressed (default) 2: Node file on disk 3: Node file on disk and compressed
<code>cplex.status</code>	Solution status code 1: Optimal for simplex and barrier methods

Table 4.1: Relevant CPLEX parameters (continued)

Parameter	Description
	11: Time limit exceeded
	101: Optimal for MIP model
	102: Optimal within predefined MIP gap tolerance
	104: Limit on mixed integer solutions
	111: Tree memory limit exceeded and integer solution found
	112: Tree memory limit exceeded and no integer solution

Two following classification files are written in Optimization Programming Language (OPL), supported by default. Code 4.1 is the main execution of the classification model in Code 4.2. Two data structures are employed: an array and a tuple. Once the first is declared, its size is unchanged. The latter is used as a secondary option only when a combination of indexes cannot perfectly fit in an array format. As illustrated in Chapter 5, only three features are considered: A_AGE, PEMLR and SS_YN. Three splits are assumed except two for SS_YN representing both whether social security payments are paid and whether a survey participant is in the universe of this question. Two most significant factors are of interest. The cardinality of a new continuous component $|\mathcal{C}_{\text{cont}}|$ is assumed to be the minimum of its given counterpart $|\tilde{\mathcal{C}}_{\text{cont}}| = 1$ and an upper bound on the number of significant features $d = 2$. The continuous feature selection can be partially concluded by the condition $c_{j,\tilde{j}}^* = 1$. The sufficiently small positive number m_0 is set to be 0.01. The execution time is limited up to 24 hours or one day. Code 4.1 records every MIP solution, feasible but not necessarily optimal, thereby calling a CPLEX solver multiple times. After the working memory exceeds 2 GB, some nodes are transferred to disk in compressed form. The uncompressed tree size is limited to 200 GB.

Code 4.1: Main OPL model

```

1  /*****
2  * OPL 22.1.1.0 Model
3  * Author: songkomkrit
4  * Creation Date: Nov 4, 2024 at 12:24:05 AM
5  *****/
6
7  /*****

```



```

8  * NOTES
9  * pl.bc.solutionValue[thisOplModel.mPairs.find(1,0)]
10 *****/
11
12 /*****
13  * Class Labels
14  * Input file: 0, 1, 2, ..., n
15  * Algorithm: 0, 1, 2, ..., n
16  * Output file: 0, 1, 2, ..., n
17  *****/
18
19 /*****
20  * INPUTS
21  *****/
22 int mdimold = 3; // dimension    // 4 or 184 or 8 or 4
23 int mdimcontold = 1; // continuous dimension // 2 or 66 or 3 or 2
24 //int mdimcat = 2; // categorical dimension // 2 or 118 or 5 or 2
25 int mN = 100; // number of instances // 8 or 157681 or 100 or 100
26 int mn = 4; // the value of n = (number of classes) - 1 // 1 or 4 or 4
27
28 int mseltol = 2; // given number of total selected cont/cat dimensions (at
    most)
29
30 // Initialized UB on number of selected continuous dimensions
31 int mselcont = mdimcontold;
32 execute {
33     if (mselcont > mseltol)
34         mselcont = mseltol;
35 }
36
37 int mexccont = mdimcontold - mselcont; // computed LB on number of
    excluded continuous dimensions
38 int mdim = mdimold - mexccont;
39 int mdimcont = mselcont;
40
41 range mDS = 1..mdim;

```

```

42 range mDSCONTOLD = 1..mdimcontold; // old continuous
43 range mDSCONT = 1..mselcont; // new continuous
44 range mDSCAT = mdimcont+1..mdim; // shifted categorical
45 range mIS = 1..mN;
46 float mxcontold[mIS][mDSCONTOLD]; // x along continuous dimensions
47 int mxcat[mIS][mDSCAT]; // x along categorical dimensions
48 int my[mIS];
49 int mmaxlab[mDSCAT]; // maximum labels for categorical dimensions
50 float mM[mDS]; // big-M for all new/shifted dimensions (continuous and
    categorical)
51 float mm[mDSCONT]; // small-m for continuous dimensions
52 int mp[mDS]; // number of cuts along axes
53 int mcoef[mDS];
54
55 /*****
56  * TUPLES
57  *****/
58 tuple ContPairType { // index for continuous cut
59     int j;
60     int q;
61 };
62
63 {ContPairType} mContPairs = {<j, q> | j in mDSCONT, q in 0..mp[j]+1};
64
65 tuple ContTripleType { // index for continuous cut of each individual
    instance
66     int i;
67     int j;
68     int q;
69 };
70
71 {ContTripleType} mContTriples = {<i, j, q> | i in mIS, j in mDSCONT, q in
    0..mp[j]};
72
73 tuple CatPairType { // index for categorical group
74     int j;

```

```

75     int l;
76 };
77
78 {CatPairType} mCatPairs = {<j, l> | j in mDSCAT, l in 0..mmaxlab[j]};
79
80 tuple tuplePred {
81     key int b;
82     sorted {int} label;
83 }
84 sorted {tuplePred} mpred;
85 {int} memptyset = {};
86
87 /*****
88  * OUTSIDE EXECUTION
89  *****/
90 execute {
91     thisOplModel.settings.run_engineLog = "tmp/current-engine.log"; //
        temporary engine log
92 }
93
94 /*****
95  * MAIN EXECUTION
96  *****/
97 main {
98     var ftime = Opl.round((new Date()).getTime()/1000) % 100000; // first
        timestamp (in seconds)
99
100    // Input/variable filenames
101    var infilename = "input/seltrain20num3each20.csv"; // input filename
102    var varfilename = "input/selproc20num3co3ca3cutinfo.csv"; // variable
        filename (6 columns)
103
104    // Prefix of all output files
105    var prefixout = "output/" + ftime + "-";
106    prefixout += infilename.split("/")[1].split(".")[0] + "-";
107

```

```

108     // Inputs
109     //var M0 = 500;    // big-M (float)
110     var m0 = 0.01;    // small-m (float)
111     var pcont0 = 3;    // max number of cuts along continuous axis (integer)
112
113     // Customization
114     var timelimit = 1; // whether set total time limits (1 = limit / 0 =
        none)
115     var limit = 1;    // whether customize performance settings (1 =
        customize / 0 = none)
116     var perf = 1;    // whether set limits (1 = limit / 0 = none)
117
118     // Custom time limit parameter
119     if (timelimit == 1)
120         var acctimelimin = 24*60; // accumulated time limit (in minutes)
121
122     // Cplex limit parameters (excluding time limit)
123     if (limit == 1) {
124         var intsollim = 1; // MIP solution number limit (in each iteration)
125     }
126
127     // Cplex performance parameters
128     if (perf == 1) {
129         var threads = 0; // parallel threads (default: 0 = at most 32
            threads)
130         var workmemgb = 2; // working memory before compression and swap (
            in GB) (default: 2 GB) (only marginally improved efficiency)
131         var trelimgb = 200; // uncompressed tree memory limit (in GB) (
            default: around 1e+72 GB)
132
133         /* Node storage file switch
134         * 0 = No node file
135         * 1 = Node file in memory and compressed (default)
136         * 2 = Node file on disk
137         * 3 = Node file on disk and compressed
138         */

```

```

139     var nodefileind = 3;
140
141     /* Note on directory for temporary working files
142      * cplex.workdir = ...;
143      * CPLEX Error 1422: Could not open file for writing
144      */
145
146     // Calculation
147     var workmem = 1024*workmemgb; // working memory before compression
        and swap (in MB) (default: 2048 MB)
148     var trelim = 1024*trelimgb; // uncompressed tree memory limit (in
        MB) (default: 1e+75 MB)
149 }
150
151 // Postfixes
152 var cpostfixname = "mfullaltseltol-" + thisOplModel.mseltol; // common
    postfix name
153 if (timelimit == 1)
154     cpostfixname += "-t-" + acctimelimmin + ".csv";
155 else
156     cpostfixname += ".csv";
157 var postfixerror = "-" + cpostfixname; // postfix of error file
158 var postfixout = "-pcont-" + pcont0 + "-" + cpostfixname; // postfix of
    all other output files
159
160 // Output filenames
161 var outerrorname = prefixout + "export-error" + postfixerror;
162 var outinstancename = prefixout + "export-predict-instance" +
    postfixout;
163 var outcutcontname = prefixout + "export-cutcont-full" + postfixout;
164 var outcutcatname = prefixout + "export-cutcat-full" + postfixout;
165 // The existence of region is not checked here
166 // In fact, it can be check through enumeration of certain binary
    representations
167 var outregionname = prefixout + "export-predict-region" + postfixout;

```

```

168     var outselvarintname = prefixout + "export-select-var-int" + postfixout
        ; // selected variables (integer)
169     var outselvarstrname = prefixout + "export-select-var-str" + postfixout
        ; // selected variables (string)
170
171     // Engine log (initialized)
172     var logfilename = "log/" + ftime + "-engine-" + cpostfixname.split(".")
        [0] + ".log";
173     var outlog = new IloOplOutputFile(logfilename);
174
175     // OPL
176     var source = new IloOplModelSource("p-mixed-cuts-alt-seltol.mod");
177     var cplex = new IloCplex();
178     var def = new IloOplModelDefinition(source);
179     var opl = new IloOplModel(def,cplex);
180     var data = new IloOplDataElements();
181
182     data.dimold = thisOplModel.mdimold;
183     data.dimcontold = thisOplModel.mdimcontold;
184     data.dim = thisOplModel.mdim;
185     data.dimcont = thisOplModel.mdimcont;
186     //data.dimcat = thisOplModel.mdimcat;
187     data.N = thisOplModel.mN;
188     data.n = thisOplModel.mn;
189     data.xcontold = thisOplModel.mxcontold;
190     data.xcat = thisOplModel.mxcat;
191     data.y = thisOplModel.my;
192
193     var pred = thisOplModel.mpred; // set of predicted labels
194
195     data.seltol = thisOplModel.mseltol;
196     data.selcont = thisOplModel.mselcont;
197     data.exccont = thisOplModel.mexccont;
198
199     data.m = thisOplModel.mm;
200     for (var j=1; j<=data.dimcont; j++)

```

```

201     data.m[j] = m0;
202
203     var f = new IloOplInputFile(infile); // training dataset
204     f.readline(); // skip a header
205     for (var i=1; i<=data.N; i++) {
206         var myitem = f.readline().split(",");
207         data.y[i] = Opl.intValue(myitem[data.dimold]);
208         for (var j=1; j<=data.dimcontold; j++)
209             data.xcontold[i][j] = Opl.floatValue(myitem[j-1]);
210         for (var j=data.dimcontold+1; j<=data.dimold; j++)
211             data.xcat[i][j-data.exccont] = Opl.intValue(myitem[j-1]);
212     }
213     f.close();
214
215     data.p = thisOplModel.mp;
216     for (var j=1; j<=data.dimcont; j++)
217         data.p[j] = pcont0;
218
219     data.M = thisOplModel.mM;
220     data.maxlab = thisOplModel.mmaxlab;
221     var M0cont = 1;
222     var f = new IloOplInputFile(varfile); // variable info
223     f.readline(); // skip a header
224     for (var j=1; j<=data.dimold; j++) {
225         var myitem = f.readline().split(",");
226         if (j <= data.dimcontold) {
227             var curMcont = 1 + Opl.max1(Opl.abs(Opl.intValue(myitem[3])),
228                                         Opl.abs(Opl.intValue(myitem[4])));
229             M0cont = Opl.max1(M0cont, curMcont);
230         }
231         else {
232             data.p[j-data.exccont] = Opl.intValue(myitem[5]);
233             data.maxlab[j-data.exccont] = Opl.intValue(myitem[4]);
234             data.M[j-data.exccont] = 1 + Opl.intValue(myitem[5]);
235         }
236     }

```

```

236     f.close();
237
238     for (var j=1; j<=data.dimcont; j++)
239         data.M[j] = M0cont;
240
241     data.coef = thisOplModel.mcoef;
242     data.coef[1] = 1;
243     for (var j=2; j<=data.dim; j++)
244         data.coef[j] = data.coef[j-1]*(data.p[j]+1);
245
246     var nump = 0; // total number of cuts
247     for (var j=1; j<=data.dim; j++)
248         nump += data.p[j];
249
250     opl.addDataSource(data);
251     opl.generate();
252     opl.settings.mainEndEnabled = true;
253
254     // Cplex limits (excluding time limit)
255     if (limit == 1) {
256         cplex.intsollim = intsollim; // MIP solution number limit (> 0)
257     }
258
259     // Cplex performance
260     if (perf == 1) {
261         cplex.threads = threads; // parallel threads
262         cplex.workmem = workmem; // working memory before compression and
            swap (in MB)
263         cplex.treelim = trelim; // uncompressed tree memory limit (in MB)
264         cplex.nodefileind = nodefileind; // node storage file switch
265     }
266
267     // Initialization
268     var status = -9; // solution status code (initialized)
269     var iter = 0; // iteration
270     var acctime = 0; // accumulated running time (in seconds)

```



```

271     var texceed = 0; // whether acctime > tilimmin (1 = total time limit
        exceeded / 0 = not)

272
273     // Calculation
274     if (timelimit == 1)
275         var acctimelim = 60*acctimelimmin; // accumulated time limit (in
            seconds)
276     else
277         var acctimelim = -1;
278
279     // Optimization
280     while (texceed == 0) { // accumulated time limit not exceeded
281
282         // Exit status codes
283         if (status == 1) // 1: CPX_STAT_OPTIMAL
284             break;
285         else if (status == 101) // 101: CPXMIP_OPTIMAL
286             break;
287         else if (status == 102) // 102: CPXMIP_OPTIMAL_TOL
288             break;
289         else if (status == 111) // 111: CPXMIP_MEM_LIM_FEAS
290             break;
291         else if (status == 112) // 112: CPXMIP_MEM_LIM_INFEAS
292             break;
293
294         /* Non-exit status codes
295         * 11: CPX_STAT_ABORT_TIME_LIM
296         * 104: CPXMIP_SOL_LIM
297         */
298
299         // In the case when the previous status is not one of the above
300         if (timelimit == 1) // time limit for each call to optimizer (in
            seconds)
301             cplex.tilim = acctimelim - acctime;
302         var start = new Date(); // begin a timer
303

```

```

304     pred.clear(); // clear previous set of predicted labels
305
306     // Solve
307     if (cplex.solve()) {
308
309         var end = new Date(); // end a timer
310         var solvetime = end.getTime() - start.getTime(); // compute
            solving time
311         acctime += solvetime/1000; // accumulated running time (in s)
312
313         if ((timelimit == 1) && (acctime >= acctimelim)) // total time
            limit exceeded (in seconds)
314             texceed = 1;
315
316         iter += 1; // update iteration
317
318         var error = data.N + cplex.getObjValue(); // the number of
            misclassified instances
319         var accuracy = (1-error/data.N)*100; // training accuracy
320
321         status = cplex.status; // solution status code (1 = opt / 11 =
            time limit / ...)
322         var lberr = data.N + cplex.getBestObjValue(); // LB on minimum
            (optimal) error
323         var relgap = cplex.getMIPRelativeGap(); // relative objective
            gap for MIP
324
325         // Open output text files (append = true)
326         var outerror = new IloOplOutputFile(outerrorname, true);
327         var outinstance = new IloOplOutputFile(outinstancename, true);
328         var outcutcont = new IloOplOutputFile(outcutcontname, true);
329         var outcutcat = new IloOplOutputFile(outcutcatname, true);
330         var outregion = new IloOplOutputFile(outregionname, true);
331         var outselvarint = new IloOplOutputFile(outselvarintname, true);

```

```

332     var outselvarstr = new IloOplOutputFile(outselvarstrname, true);

333
334     // outerror
335     if (!outerror.exists) {
336         outerror.write("iter,");
337         for (var j=1; j<=data.dim; j++)
338             outerror.write("p", j, ",");
339         outerror.write("error,accuracy,ms,acctmin,status,lberr,
            relgap");
340     }
341     outerror.write("\n", iter, ",");
342     for (var j=1; j<=data.dim; j++)
343         outerror.write(data.p[j], ",");
344     outerror.write(error, ",", accuracy, ",");
345     outerror.write(solvetime, ",", acctime/60, ",");
346     outerror.write(status, ",", lberr, ",", relgap);
347
348     // Scripting logs 1
349     writeln("\n-----");
350     writeln("Iteration ", iter);
351     writeln("Bounds on # of cuts = ", nump, " with", data.p);
352     writeln("Error = ", error, " (out of ", data.N, " instances)");
353     writeln("Accuracy = ", accuracy);
354     writeln("Solving time = ", solvetime/60000, " min (minutes)");
355     writeln("Accumulated time = ", acctime/60, " min (minutes)");
356     writeln("\nSolution status code = ", status);
357     writeln("LB on error = ", lberr);
358     writeln("Relative objective gap = ", relgap);
359     writeln("\nSelected variables:");
360
361     // Create a set of predicted labels (majority voting)
362     for (var b=0; b<opl.B; b++) {
363         var lset = Opl.operatorUNION(thisOplModel.emptyset,
            thisOplModel.emptyset);
364         var maxnum = 0;

```

```

365         for (var k=0; k<=data.n; k++) {
366             var num = 0;
367             for (var i=1; i<=data.N; i++)
368                 num += (data.y[i] == k)*opl.g.solutionValue[i][b];

369             if (num == maxnum)
370                 lset.add(k);
371             else if (num > maxnum) {
372                 maxnum = num;
373                 lset.clear();
374                 lset.add(k);
375             }
376         }
377         pred.add(b, lset);
378     }
379
380     // outinstance
381     if (!outinstance.exists)
382         outinstance.write("iter,id,class,region,predict");
383     for (var i=1; i<=data.N; i++) {
384         outinstance.write("\n", iter, ",", i, ",", data.y[i], ",");
385         for (var b=0; b<opl.B; b++)
386             if (opl.g.solutionValue[i][b] == 1) { // occur only once
387                 outinstance.write(b, ",");
388                 outinstance.write(pred.get(b).label);
389                 break; // terminate the loop
390             }
391     }
392
393     // outcutcont
394     if (!outcutcont.exists)
395         outcutcont.write("iter,j,q,bc");
396     for (var j=1; j<=data.dimcont; j++) {
397         for (var q=1; q<=data.p[j]; q++) {
398             outcutcont.write("\n", iter, ",", j, ",", q, ",");

```

```

399         outcutcont.write(opl.bc.solutionValue[thisOplModel.
              mContPairs.find(j,q)]));
400     }
401 }
402
403 // outcutcat
404 if (!outcutcat.exists)
405     outcutcat.write("iter,j,l,v");
406 for (var j=data.dimcont+1; j<=data.dim; j++) {
407     for (var l=0; l<=data.maxlab[j]; l++) {
408         outcutcat.write("\n", iter, ",", j, ",", l, ",");
409         outcutcat.write(opl.v.solutionValue[thisOplModel.
              mCatPairs.find(j,l)]);
410     }
411 }
412
413 // outregion
414 if (!outregion.exists)
415     outregion.write("iter,region,occupy,predict");
416 for (var b=0; b<opl.B; b++) {
417     outregion.write("\n", iter, ",", b, ",");
418     var s = 0; // initialize s (presumably unoccupied)
419     for (var i=1; i<=data.N; i++)
420         if (opl.g.solutionValue[i][b] == 1) { // occupied
421             s = 1;
422             break; // interminate the loop
423         }
424     outregion.write(s, ",");
425     outregion.write(pred.get(b).label);
426 }
427
428 // outselvarint
429 if (!outselvarint.exists)
430     outselvarint.write("iter,j,jold,mselect,type"); // mselect =
              model select (not actual)

```

```

431     for (var j=1; j<=data.dimcont; j++) { // selected continuous
        features
432         outselvarint.write("\n", iter, ",", j, ",");
433         var seljold = -1;
434         for (var jold=1; jold<=data.dimcontold; jold++)
435             // Determine which old continuous feature is selected
436             if (opl.ccont.solutionValue[j][jold] == 1) {
437                 seljold = jold;
438                 break; // terminate the loop
439             }
440         outselvarint.write(seljold, ",");
441         outselvarint.write("1,"); // Based on model, all new cont
            features are selected
442         outselvarint.write("cont");
443     }
444     for (var j=data.dimcont+1; j<=data.dim; j++) { // categorical
        feature
445         outselvarint.write("\n", iter, ",", j, ",", j+data.excont,
            ",");
446         if (opl.f.solutionValue[j] == 1) // selected categorical
            feature (model)
447             outselvarint.write("1,");
448         else // unselected categorical feature (model)
449             outselvarint.write("0,");
450         outselvarint.write("cat");
451     }
452
453     // outselvarstr
454     if (!outselvarstr.exists)
455         outselvarstr.write("iter,jold,jnew,aselect,type,variable");
            // aselect = actual select
456     var varinfile = new IloOplInputFile(varfilename); // variable
        info
457     varinfile.readline(); // skip a header
458     var numselcont = 0; // initialized number of actually selected
        continuous features

```

```

459     var numselcat = 0; // initialized number of actually selected
        categorical features
460     for (var jold=1; jold<=data.dimcontold; jold++) { // CONTINUOUS
461         outselvarstr.write("\n", iter, ",", jold, ",");
462         var jnew = -1;
463         var aselect = 0; // initialized to be unselected (continuous
            )
464         for (var j=1; j<=data.dimcont; j++)
465             // Determine whether a current old continuous feature is
                selected
466             if (opl.ccont.solutionValue[j][jold] == 1) { // selected
                    (actual 1/2)
467                 jnew = j;
468                 break; // terminate the loop
469             }
470         outselvarstr.write(jnew, ",");
471         var myitem = varinfile.readline().split(",");
472         if (jnew > 0) { // selected continuous feature (actual 1/2)
473             aselect = 1; // seem to be selected (initialization for
                actual 2/2)
474             for (var q=0; q<=data.p[jnew]; q++) {
475                 var bcleft = opl.bc.solutionValue[thisOplModel.
                    mContPairs.find(jnew,q)];
476                 var bcrigh = opl.bc.solutionValue[thisOplModel.
                    mContPairs.find(jnew,q+1)];
477                 var minxjnew = Opl.intValue(myitem[3]);
478                 var maxxjnew = Opl.intValue(myitem[4]);
479                 if ((bcleft <= minxjnew) && (bcrigh >= maxxjnew)) {
                    // cover [min,max]
480                     aselect = 0; // unselected (actual 2/2)
481                     break;
482                 }
483             }
484         }
485         outselvarstr.write(aselect, ",");
486         if (aselect == 1) { // actually selected continuous feature

```

```

487         // Scripting logs 2 (continuous)
488         write("\t", myitem[1], " (Continuous)\n");
489         numselcont += 1;
490     }
491     outselvarstr.write("cont,");
492     outselvarstr.write(myitem[1]); // variable name
493 }
494 for (var jold=data.dimcontold+1; jold<=data.dimold; jold++) { //
    CATEGORICAL
495     var jnew = jold-data.exccont;
496     outselvarstr.write("\n", iter, ",", jold, ",", jnew, ",");
497     var aselect = 0; // initialized to be unselected (
        categorical)
498     var myitem = varinfile.readline().split(",");
499     if (opl.f.solutionValue[jnew] == 1) { // selected
        categorical feature (actual 1/2)
500         var vat0 = opl.v.solutionValue[thisOplModel.mCatPairs.
            find(jnew,0)];
501         for (var l=1; l<=data.maxlab[jnew]; l++) {
502             var vcur = opl.v.solutionValue[thisOplModel.mCatPairs
                .find(jnew,l)];
503             if (vcur != vat0) { // distinct new groups are
                detected
504                 aselect = 1; // selected categorical feature (
                    actual 2/2)
505                 break;
506             }
507         }
508     }
509     outselvarstr.write(aselect, ",");
510     if (aselect == 1) { // actually selected categorical feature
        // Scripting logs 2 (categorical)
511         write("\t", myitem[1], " (Categorical)\n");
512         numselcat += 1;
513     }
514 }
515 outselvarstr.write("cat,");

```



```

516         outselvarstr.write(myitem[1]);
517     }
518     varinfile.close();
519
520     // Scripting logs 3
521     var numselall = numselcont + numselcat;
522     writeln("\nNumber of selected variables = ", numselall, " (",
523         numselcont, " continuous + ", numselcat, " categorical)");
524     writeln("-----");
525
526     // Closing output text files
527     outerror.close();
528     outinstance.close();
529     outcutcont.close();
530     outcutcat.close();
531     outregion.close();
532     outselvarint.close();
533     outselvarstr.close();
534 }
535 else
536     writeln("No solution");
537
538 opl.end();
539 data.end();
540 def.end();
541 cplex.end();
542 source.end();
543
544 // Engine log (exported)
545 var inlog = new IloOplInputFile("tmp/current-engine.log");
546 while (!inlog.eof) {
547     outlog.writeln(inlog.readline());
548 }
549 inlog.close();
550 outlog.close();

```

551 }

Code 4.2: Box classifier OPL model

```

1  /*****
2   * OPL 22.1.1.0 Model
3   * Author: songkomkrit
4   * Creation Date: Nov 4, 2024 at 1:15:57 AM
5   *****/
6
7  /*****
8   * DATA INFORMATION (INPUTS)
9   *****/
10 int dimold = ...; // old dimension
11 int dimcontold = ...; // old continuous dimension
12 int dim = ...; // new dimension
13 int dimcont = ...; // new continuous dimension
14 //int dimcat = ...; // categorical dimension
15 int N = ...; // number of instances
16 int n = ...; // number of classes
17
18 /*****
19  * FEATURE SELECTION (INPUTS)
20  *****/
21 int seltol = ...; // given number of total selected cont/cat dimensions (
    at most)
22 int selcont = ...; // UB on number of selected continuous dimensions
23 int excont = ...; // computed LB on number of excluded continuous
    dimensions
24
25 /*****
26  * INDEX RANGES 1
27  *****/
28 range DS = 1..dim; // for dimensions
29 range DSCONTOLD = 1..dimcontold; // for old continuous dimensions
30 range DSCONT = 1..dimcont; // for new continuous dimensions

```

```

31 range DSCAT = dimcont+1..dim; // for shifted categorical dimensions
32 range IS = 1..N; // for instances
33 range KS = 0..n; // for classes
34
35 /*****
36  * INITIAL PARAMETERS (INPUTS)
37  *****/
38 float M[DS] = ...; // big-M for all new/shifted dimensions (continuous
    and categorical)
39 float m[DSCONT] = ...; // small-m for new continuous dimensions
40
41 /*****
42  * DATA EXTRACTION (INPUTS)
43  *****/
44 float xcontold[IS][DSCONTOLD] = ...; // instances along old continuous
    dimensions
45 int xcat[IS][DSCAT] = ...; // instances along shifted categorical
    dimensions
46 int y[IS] = ...; // targets
47 int maxlab[DSCAT] = ...; // maximum labels for new categorical dimensions
48 int p[DS] = ...; // number of cuts along axes
49 int coef[DS] = ...; // product coefficients
50
51 /*****
52  * NUMBER OF BOXES
53  *****/
54 int B = 1; // initialize the number of boxes
55 execute {
56     for (var j in DS)
57         B = B*(p[j]+1); // compute the number of boxes
58 }
59
60 /*****
61  * INDEX RANGES 2
62  *****/
63 range BS = 0..B-1; // for regions

```

```

64
65 /*****
66  * TUPLES
67  *****/
68 tuple ContPairType { // index for continuous cut
69     int j;
70     int q;
71 };
72
73 {ContPairType} ContPairs = {<j, q> | j in DSCONT, q in 0..p[j]+1};
74
75 tuple ContTripleType { // index for continuous cut of each individual
76     instance
77     int i;
78     int j;
79     int q;
80 };
81 {ContTripleType} ContTriples = {<i, j, q> | i in IS, j in DSCONT, q in 0..
82     p[j]};
83
84 tuple CatPairType { // index for categorical group
85     int j;
86     int l;
87 };
88 {CatPairType} CatPairs = {<j, l> | j in DSCAT, l in 0..maxlab[j]};
89
90 /*****
91  * DECISION VARIABLES
92  *****/
93 dvar float l[ContTriples];
94 dvar float r[ContTriples];
95 dvar float bc[ContPairs]; // bc is in R (c = cut)
96 // Note that b is used for beta indexing
97 dvar float h[BS]; // h

```

```

98 dvar boolean a[ContTriples]; // alpha
99 dvar int+ v[CatPairs]; // v (categorical features)
100 dvar boolean g[IS][BS]; // gamma
101 dvar boolean z[BS][KS]; //
102 // Feature selection
103 dvar boolean ccont[DSCONT][DSCONTOLD]; // select continuous dimensions
104 dvar boolean f[DSCAT]; // select categorical dimensions
105
106 /*****
107  * OBJECTIVE FUNCTION
108  *****/
109 minimize sum(b in BS) h[b]; // min total number of misclassified
    instances
110
111 /*****
112  * CONSTRAINTS
113  *****/
114 subject to {
115
116     forall(j in DSCONT)
117         getnewcont:
118             sum(jold in DSCONTOLD) ccont[j][jold] <= 1;
119
120     forall(jold in DSCONTOLD)
121         seloldcont:
122             sum(j in DSCONT) ccont[j][jold] <= 1;
123
124     forall(j in DSCONT, q in 0..p[j])
125         bc[<j,q+1>] - bc[<j,q>] >= 0;
126
127     forall(i in IS, j in DSCONT) {
128         lbound:
129             (sum(jold in DSCONTOLD) xcontold[i][jold]*ccont[j][jold]) - (
                sum(q in 0..p[j]) l[<i,j,q>]) >= 0;
130         rbound:

```

```

131         (sum(jold in DSCONTOLD) xcontold[i][jold]*ccont[j][jold]) - (
            sum(q in 0..p[j]) r[<i,j,q>]) <= 0;
132     }
133
134     forall(i in IS, j in DSCONT, q in 0..p[j]) {
135         l[<i,j,q>] + M[j]*a[<i,j,q>] >= 0;
136         l[<i,j,q>] - M[j]*a[<i,j,q>] <= 0;
137         l[<i,j,q>] - bc[<j,q>] + M[j]*a[<i,j,q>] <= M[j] + m[j];
138         l[<i,j,q>] - bc[<j,q>] - M[j]*a[<i,j,q>] >= -M[j] + m[j];
139         r[<i,j,q>] + M[j]*a[<i,j,q>] >= 0;
140         r[<i,j,q>] - M[j]*a[<i,j,q>] <= 0;
141         r[<i,j,q>] - bc[<j,q+1>] + M[j]*a[<i,j,q>] <= M[j] - m[j];
142         r[<i,j,q>] - bc[<j,q+1>] - M[j]*a[<i,j,q>] >= -M[j] - m[j];
143     }
144
145     forall(i in IS)
146         (sum(j in DSCONT) coef[j]*(sum(q in 0..p[j]) q*a[<i,j,q>])) + (sum(
            j in DSCAT) coef[j]*v[<j,xcat[i][j]>]) - (sum(b in BS) b*g[i][b
            ]) == 0;
147
148     forall(i in IS, j in DSCONT)
149         pregon:
150             sum(q in 0..p[j]) a[<i,j,q>] == 1;
151
152     forall(i in IS) {
153         bregion:
154             sum(b in BS) g[i][b] == 1;
155     }
156
157     forall(b in BS, k in KS)
158         error1:
159             h[b] + (sum(i in IS) (y[i] == k)*g[i][b]) + N*z[b][k] >= 0;
160
161     forall(b in BS)
162         error2:
163             sum(k in KS) z[b][k] == n;

```

```

164
165     forall(j in DSCAT, l in 0..maxlab[j])
166         v[<j,l>] <= p[j];
167
168     forall(i in IS, j in DSCAT) {
169         selcat1:
170             v[<j,xcat[i][j]>] + M[j]*f[j] >= 0;
171         selcat2:
172             v[<j,xcat[i][j]>] - M[j]*f[j] <= 0;
173     }
174
175     seltolnum:
176         (sum(j in DSCONT, jold in DSCONTOLD) ccont[j][jold]) + (sum(j in
            DSCAT) f[j]) <= seltol;
177 }

```

4.4 Recalculation of Decision Boxes

Some of selected d features may be trivial; therefore, they cannot be contributing factors. This occurs when two consecutive splitting values along a continuous feature covers an entire dataset or all categorical values are reallocated to the same group. Moreover, the proposed classification model usually assumes that there are up to d new continuous features ($|\mathcal{C}_{\text{cont}}| \leq d$), but a new continuous feature $j \in \mathcal{C}_{\text{cont}}$ may turn unselected: $c_{j,\tilde{j}}^* = 0$ for all $\tilde{j} \in \tilde{\mathcal{C}}_{\text{cont}}$. All of these circumstances lead to excessive number of decision boxes. A close examination of optimal splitting values $b_{j,q}^*$ and $v_{j,x_j^i}^*$ can further provide which feature is actually important and should be finally selected, thereby reducing number of boxes. To determine which two distinct boxes can be merged, all numerical decision box labels are recalculated through a transformation g to new labels in a final feature space.

Suppose only d' out of d features are finally selected. The feature map $\sigma : \{0, 1, \dots, d\} \rightarrow \{-1\} \cup \{0, 1, \dots, d'\}$ is defined by

$$\sigma(j) = \begin{cases} \text{feature in new space,} & \text{for finally selected feature } j \\ -1, & \text{for finally unselected feature } j \\ 0, & \text{if } j = 0. \end{cases}$$

There is a one-to-one corresponding between j and $\sigma(j) \geq 0$, and the image of σ includes $0, 1, \dots, d'$. Consider a decision box $1 \leq \beta \leq B$. Define its position along a feature j by

$$q_j = \begin{cases} \sum_{q=0}^{p_j} q\beta_{j,q}, & \text{for continuous feature } j \\ u_j, & \text{for categorical feature } j. \end{cases}$$

Let $w = \min\{j : q_j \neq 0\}$. If $w = 1$, then both positions of the current box β and the previous counterpart $\beta - 1$ along the first feature differ by 1. For $w > 1$, the previous box $\beta - 1$ locates at position p_j along every feature $j < w$, and the position of both boxes at feature w differs by 1. Based on this observation, the following recurrence relation of new box labels can be obtained:

$$g(\beta) - g(\beta - 1) = - \sum_{j=1}^{w-1} p_j \prod_{j' \in \Sigma_j} (p_{j'} + 1) + 1 \cdot \prod_{j' \in \Sigma_w} (p_{j'} + 1)$$

where $g(0) = 0$ and $\Sigma_j = \{j' : 0 \leq \sigma(j') < \sigma(j)\}$.

The utility module in Code 4.3 includes file copying, floating point number rounding, retrieving all keys of maximum dictionary value, finding an interval containing a given number, and exporting DataFrame with nonduplicate entries. The typecasting module in Code 4.4 can convert a set in string format to a Python set and vice versa, and also express an immutable interval object in string format. The recalculation module in Code 4.5 computes a full list of final numerical decision regions $g(\beta)$. Modules 4.6 and 4.7 returns the dictionaries of selected features and their splitting values respectively. True decision regions including their predicted class labels are computed by Module 4.8. Similar results generated by Module 4.9 is based solely on numerical decision regions, possibly redundant before merging, and their predicted class labels directly reported by CPLEX optimizer. As shown in Chapter 5, CPLEX solutions are inconsistent and therefore infeasible during first few iterations. Module 4.10 calculates the number of correctly classified instances based on the true decision region from Module 4.8 and the CPLEX counterpart from Module 4.9. Clearly, the first is more accurate than the latter. Code 4.11 is the main execution file. A DataFrame iterator initially constructed by the method `itertuples` is utilized only when a DataFrame, an iterable, can be iterated row by row using the method `next` during an informational query; nonetheless, its usage is not recommended when a query answer is scattered over rows.

Code 4.3: Basic utility for recalculation of region (module/operation/xutil.py)

```

1  import os
2  import shutil
3  import json
4  import math
5  import numpy as np
6  import pandas as pd
7
8  # Create directory (if not exist)
9  def create_dir(dir):
10     '''
11         Usage: create directory (if not exist)
12         Required arguments:
13             dir: directory name
14     '''
15
16     try: os.makedirs(dir)
17     except FileExistsError: pass
18
19
20 # Copy single file
21 def copy(srcpath, destpath):
22     '''
23         Usage: copy single file
24         Required arguments:
25             srcpath: source pathname
26             destpath: destination pathname
27     '''
28
29     # Split path into directory and file
30     srcdir, srcfile = os.path.split(srcpath) # source
31     destdir, destfile = os.path.split(destpath) # destination
32
33     # Create destination directory (if not exist)
34     create_dir(destdir)

```

```

35
36     # Copy source file into destination folder (filename unchanged)
37     shutil.copy2(srcpath, destdir) # preserve file metadata
38
39     # Rename copied file to correct destination filename
40     os.rename(f"{destdir}/{srcfile}", destpath)
41
42
43 # Round up or down number to decimal places
44 def round_num(number, decimals, direction):
45     '''
46     Usage: round up or down number to decimal places
47     Required arguments:
48         number: number to be rounded
49         decimals: number of decimal places to round to
50         direction: either up or down ('up', 'down')
51     Outputs:
52         rounded number to specified decimal places
53     '''
54
55     if isinstance(decimals, int) or isinstance(decimals, np.integer):
56         if decimals >= 0:
57             if direction == 'up':
58                 return math.ceil(number*10**decimals)/10**decimals
59             elif direction == 'down':
60                 return math.floor(number*10**decimals)/10**decimals
61             else:
62                 raise TypeError("Direction can be either up or down")
63         else:
64             raise TypeError("Number of decimal places to round to must be
65                               nonnegative")
66     else:
67         raise TypeError("Number of decimal places must be an integer")
68
69 # Find maximum value of dictionary and key set

```

```

70 def max_dictval(dc):
71     '''
72     Usage: find maximum value of dictionary and all of its
           corresponding keys
73     Required arguments:
74         dc: dictionary
75     Outputs:
76         kmax: set of all keys of maximum value
77         vmax: maximum value
78     '''
79
80     kmax = set()
81     vmax = dc[next(iter(dc))] # value of first key
82     for k, v in dc.items():
83         if v > vmax:
84             vmax = v
85             kmax = {k}
86         elif v == vmax:
87             kmax.add(k)
88
89     return kmax, vmax
90
91
92 # Find interval index of specific value from list of real-line splits
93 def itvpos(x, splits, closed='neither'):
94     '''
95     Usage: find interval index of specific value from array of real-
           line splits
96     Required arguments:
97         x: specific value of interest
98         splits: list of real line splits
99         closed: whether intervals are closed on left-side, right-side
           or neither ('left', 'right', 'neither')
100     Outputs:
101         interval index of specific input value
102     '''

```

```

103
104     if closed == 'left': # (_, s), [s, _)
105         for i, s in enumerate(splits):
106             if x < s: return i
107     elif closed == 'neither': # (_, s), (s, _)
108         for s in splits:
109             if x == s:
110                 raise Exception(f"Open intervals are chosen but input value
                                {x} is at split value {s}")
111         closed = 'right' # now safe to be extended to (_, s], (s, _]
112
113     if closed == 'right': # (_, s], (s, _]
114         for i, s in enumerate(splits):
115             if x <= s:
116                 return i
117
118     # Last interval
119     return i + 1
120
121
122 # Return left and right endpoints of rounded interval
123 def itvtopts(itv, decimals=2, extend=True):
124     '''
125     Usage: return left and right endpoints of rounded interval
126     Required arguments:
127         itv: Pandas interval to be rounded
128     Optional arguments:
129         decimals: number of decimal places to round to (default: 2)
130         extend: whether extend (true) or shrink (default) interval (
                default: True)
131     Outputs:
132         lpt: left endpoint of rounded interval
133         rpt: right endpoint of rounded interval
134     '''
135
136     if isinstance(itv, pd._libs.interval.Interval):

```

```

137         if extend:
138             ldirect, rdirect = 'down', 'up'
139         else:
140             ldirect, rdirect = 'up', 'down'
141
142         if np.isinf(itv.left):
143             lpt = itv.left
144         else:
145             lpt = round_num(itv.left, decimals, ldirect)
146
147         if np.isinf(itv.right):
148             rpt = itv.right
149         else:
150             rpt = round_num(itv.right, decimals, rdirect)
151
152         return lpt, rpt
153
154     else:
155         raise TypeError("Only Pandas intervals are allowed")
156
157
158 # Import dictionary from JSON file
159 def import_dict(jsonpath):
160     '''
161         Usage: parse JSON data into dictionary
162         Required arguments:
163             jsonpath: JSON filepath (usually metadata filepath)
164         Outputs:
165             dictionary
166     '''
167
168     with open(jsonpath) as file:
169         contents = file.read()
170
171     # JSON data is parsed into dictionary
172     return json.loads(contents)

```

```

173
174
175 # Export dataframe with nonduplicate entries
176 def nondup(df, ndcols, intcols=list(), intdtype='Int16'):
177     '''
178         Usage: export dataframe with nonduplicate entries
179         Required arguments:
180             df: dataframe
181             ndcols: two-dimensional multilevel column lists with
182                     nonduplicate entries
183         Optional arguments:
184             intcols: integer columns (default: empty list)
185             intdtype: Pandas integer data type (default: 'Int16' or pd.
186                     Int16Dtype())
187         Outputs: same dataframe but without duplicate entries
188     '''
189
190     dfn = df.copy(deep=True)
191     for i in range(len(ndcols),0,-1): # iterate over multilevel column
192         lists with nonduplicate entries
193         ccols = [f for cols in ndcols[0:i] for f in cols]
194         dfn.loc[dfn[ccols].duplicated(), ccols] = pd.NA
195     for col in intcols:
196         dfn[col] = pd.array(dfn[col], dtype=intdtype)
197
198     return dfn

```

Code 4.4: Typecasting (module/operation/typecast.py)

```

1  import re
2  import numpy as np
3  import pandas as pd
4
5  from module.operation.xutil import itvtopts
6
7
8  # Convert set/number in string format to Python set
9  def strtoset(setstr):
10     '''
11         Usage: convert set/number in string format to Python set
12         Required arguments:
13             setstr: set/number in string format
14         Outputs: corresponding set
15     '''
16
17     elems = re.findall(r'[^{};,\\s]+', setstr)
18     numset = set(map(int, elems))
19
20     return numset
21
22
23 # Convert set to string
24 def settostr(st, sep=',', left='{', right='}'):
25     '''
26         Usage: convert set to string
27         Required arguments:
28             st: set
29         Optional arguments:
30             sep: separator (default: ',')
31             left: left symbol (default: '{')
32             right: right symbol (default: '}')
33         Outputs: string representing given set
34     '''

```

```

35
36     stre = sep.join([str(e) for e in st])
37
38     return f"{left}{stre}{right}"
39
40
41 # Convert Pandas interval to string
42 def itvtostr(itv, decimals=2, extend=True):
43     '''
44         Usage: convert Pandas interval to string
45         Required arguments:
46             itv: Pandas interval
47         Optional arguments:
48             decimals: number of decimal places to round to (default: 2)
49             extend: whether extend (true) or shrink (default) interval (
                     default: True)
50         Outputs: string interval
51     '''
52
53     lpt, rpt = itvtopts(itv, decimals, extend)
54     l = f"{lpt:.{decimals}f}"
55     r = f"{rpt:.{decimals}f}"
56
57     if itv.closed == 'neither': return f"({l}, {r})"
58     elif itv.closed == 'left': return f"[{l}, {r})"
59     elif itv.closed == 'right': return f"({l}, {r}]"
60     else: return f"[{l}, {r}]"
61
62
63 # Describe Pandas interval in text format
64 def itvtodesc(itv, decimals=2, extend=True):
65     '''
66         Usage: describe Pandas interval in text format
67         Required arguments:
68             itv: Pandas interval
69         Optional arguments:

```



```

70         decimals: number of decimal places to round to (default: 2)
71         extend: whether extend (true) or shrink (default) interval (
                    default: True)
72     Outputs: description of interval in text format
73     '''
74
75     lpt, rpt = itvtopts(itv, decimals, extend)
76     l = f"{lpt:.{decimals}f}"
77     r = f"{rpt:.{decimals}f}"
78
79     esum = itv.left + itv.right
80     if np.isnan(esum): # -np.inf, np.inf
81         return "any number"
82     elif not np.isinf(esum): # num, num
83         return f"between {l} and {r}"
84     elif esum < 0: # -np.inf, num
85         return f"below {r}"
86     else: # num, np.inf
87         return f"above {l}"

```

Code 4.5: Recalculation of regions (module/operation/calregs.py)

```

1  import numpy as np
2
3
4  # Calculate new corresponding region label (helper)
5  def hcalbn(bo, bnprev, idxn, pcuto, pocum, pncumx):
6      '''
7          Usage: calculate new corresponding region label (helper)
8          Required arguments:
9              bo: region label for old features (nonzero)
10             bnprev: previous region label for new features
11             idxn: new feature indexes
12             pcuto: old cut numbers
13             pocum: cumulative number of box regions across old features

```

```

14         pncumx: cumulative number of extended box regions across new
                features
15     Outputs: corresponding region label
16     '''
17
18     # bo must be between 1 and np.prod(pcuto+1)-1
19     bn = bnprev
20     for jmax in range(len(pcuto)-1,-1,-1):
21         # bo (incremented by 1) in base representation has the last nonzero
                at digit jmax
22         if bo%pocum[jmax] == 0:
23             for j in range(jmax):
24                 bn -= pcuto[j]*pncumx[idxn[j]]
25                 bn += pncumx[idxn[jmax]]
26             break
27
28     return bn
29
30
31 # Calculate corresponding decision regions (helper)
32 def hcalregs(B0, idxn, pcuto, pocum, pncumx):
33     '''
34         Usage: calculate corresponding decision regions (helper)
35         Required arguments:
36             B0: total number of old box regions
37             idxn: new feature indexes
38             pcuto: old cut numbers
39             pocum: cumulative number of box regions across old features
40             pncumx: cumulative number of extended box regions across new
                    features
41     Outputs: corresponding region label
42     '''
43
44     bns = [0] # list of corresponding box regions (region 0)
45     for bo in range(1, B0):
46         bnprev = bns[-1]

```

```

47         bn = hcalbn(bo, bnprev, idxn, pcuto, pocum, pncumx)
48         bns.append(bn)
49
50     return bns
51
52
53 # Calculate new corresponding decision regions (main)
54 def calregs(pcuto, sidx, pdtype=np.int16, idtype=np.int16, rdtype=np.int16
55            ):
56     '''
57     Usage: calculate new corresponding decision regions (main)
58     Required arguments:
59         pcuto: old cut numbers
60         sidx: selected feature indexes (in order)
61     Optional arguments:
62         pdtype: NumPy data type of cut number (default: np.int16)
63         idtype: NumPy data type of index (default: np.int16)
64         rdtype: NumPy data type of region number (default: np.int16)
65     Outputs: new corresponding regions
66     '''
67     # Typecasting
68     pcuto = np.array(pcuto, dtype=pdtype)
69     sidx = np.array(sidx, dtype=idtype)
70
71     # Basic calculation
72     dimo = pcuto.size # old dimension
73     dimn = sidx.size # new dimension
74     pcutn = pcuto[sidx] # new cut numbers
75     B0 = np.prod(pcuto+1).astype(rdtype) # number of old regions
76     BN = np.prod(pcutn+1).astype(rdtype) # number of new regions
77
78     # New feature indexes
79     idxn = np.full(dimo, -1, dtype=idtype)
80     idxn[sidx] = np.arange(dimn, dtype=idtype)
81     idxn[idxn < 0] = np.arange(dimn, dimo, dtype=idtype)

```

```

82
83     # Cumulative number of box regions
84     pocum = np.cumprod(np.append([1], pcuto[0:-1]+1), dtype=rdtype) # old
85     pncum = np.cumprod(np.append([1], pcutn[0:-1]+1), dtype=rdtype) # new
86     pncumx = np.concatenate((pncum, np.zeros(dimo-dimn, dtype=rdtype))) #
        new and extended
87
88     # New corresponding regions (helper function called)
89     bns = np.array(hcalregs(B0, idxn, pcuto, pocum, pncumx), dtype=rdtype)
90
91     # Output
92     return bns
93
94
95 # Illustration
96 '''
97 print('pcuto: {0}\nsidx: {1}\nbns: {2}\n'.format(pcuto:= [3, 4], sidx:= [0],
        calregs(pcuto, sidx)))
98 print('pcuto: {0}\nsidx: {1}\nbns: {2}\n'.format(pcuto:= [3, 4], sidx:= [1],
        calregs(pcuto, sidx)))
99 print('pcuto: {0}\nsidx: {1}\nbns: {2}\n'.format(pcuto:= [3, 4], sidx:= [0,
        1], calregs(pcuto, sidx)))
100 print('pcuto: {0}\nsidx: {1}\nbns: {2}\n'.format(pcuto:= [3, 4], sidx:= [1,
        0], calregs(pcuto, sidx)))
101 '''

```

Code 4.6: Feature selection (module/model/findsels.py)

```

1  # Find feature selection
2  def findsels(itsel, pcuto):
3      '''
4          Usage: find feature selection (per file)
5          Required arguments:
6              itsel: selected string variables (DataFrame iterator)
7              pcuto: old cut numbers
8          Outputs:
9              tsels: dictionary of selected variables and given number of
                     cuts
10         '''
11
12         csrow = next(itsel) # iterator of selected string variables across all
                             # iterations
13         tsels = dict() # selected variables and given number of cuts
14
15         citer = -1 # current iteration
16         while True:
17             try:
18                 if csrow.aselect == 1: # for selected variable
19                     if csrow.iter != citer:
20                         citer = csrow.iter
21                         tsels[citer] = {
22                             'variables': list(), # selected feature
23                             'types': list(), # type of selected feature
24                             'js': list(), # selected index
25                             'ps': list() # given cut number
26                         }
27                         tsels[citer]['variables'].append(csrow.variable)
28                         tsels[citer]['types'].append(csrow.type)
29                         tsels[citer]['js'].append(csrow.jnew)
30                         tsels[citer]['ps'].append(pcuto[csrow.jnew-1])
31                 csrow = next(itsel) # update DataFrame iterator
32             except StopIteration:

```

```

33         break
34
35     return tsels

```

Code 4.7: Cuts or split values (module/model/findcuts.py)

```

1  import numpy as np
2  import pandas as pd
3
4  # Find cuts and groups
5  def findcuts(tsels, itcont, itcat, intvclosed='neither', intvsubtype='
    float32'):
6      '''
7          Usage: find cuts and groups (per file)
8          Required arguments:
9              tsels: dictionary of selected variables and given number of
                cuts
10             itcont: full continuous cuts (DataFrame iterator)
11             itcat: full categorical cuts (DataFrame iterator)
12          Optional arguments:
13             intvclosed: types of Pandas interval sides (values: 'left', '
                right', 'both', 'neither')
14             intvsubtype: types of Pandas interval bounds (subtype of pandas.
                IntervalDtype)
15          Outputs:
16             tcuts: dictionary of cuts and groups along all selected
                features
17      '''
18
19      ccontrow = next(itcont) # iterator of full continuous cuts across all
        iterations
20      ccatrow = next(itcat) # iterator of full categorical cuts across all
        iterations
21      tcuts = dict() # cuts and groups along all selected features
22
23      for citer, sel in tsels.items(): # cuts across all selected features

```

```

24     tcuts[citer] = dict()
25     for ind, j in enumerate(sel['js']):
26         tcuts[citer][j] = {
27             'variable': tsels[citer]['variables'][ind],
28             'type': tsels[citer]['types'][ind],
29             'cuts': list(),
30             'groups': dict()
31         }
32
33     # Cuts
34     while ccontrow.iter < citer: # previous iteration may select no
        continuous feature
35         ccontrow = next(itcont)
36     while ccatrow.iter < citer: # previous iteration may select no
        categorical feature
37         ccatrow = next(itcat)
38     for jcur in sorted(sel['js']): # numerically sorted features
        selected
39         cuts = tcuts[citer][jcur]['cuts'] # list of cuts along specific
            selected feature
40         try: # iterate over full continuous cuts
41             while ccontrow.iter == citer:
42                 if ccontrow.j > jcur: # seek no more than current
                    feature
43                     break
44                 else:
45                     if ccontrow.j == jcur: # at current selected feature
46                         cuts.append(ccontrow.bc) # continuous feature
                            seen
47                         ccontrow = next(itcont) # update DataFrame iterator
48         except StopIteration:
49             pass
50         try: # iterate over full categorical cuts
51             while ccatrow.iter == citer:
52                 if ccatrow.j > jcur: # seek no more than current feature
53                     break

```

```

54         else:
55             if ccatrow.j == jcur: # at current selected feature
56                 cuts.append(ccatrow.v) # categorical feature seen
57                 ccatrow = next(itcat) # update DataFrame iterator
58     except StopIteration:
59         pass
60
61     # Groups
62     pcutdc = dict(zip(tsels[citer]['js'], tsels[citer]['ps'])) # cut
        numbers along selected features
63     for j, info in tcuts[citer].items():
64         pnum = pcutdc[j] # number of cuts on current selected feature
65         cuts = info['cuts']
66         if info['type'] == 'cont': # continuous feature
67             excuts = [-np.inf] + cuts + [np.inf]
68             intvs = pd.arrays.IntervalArray.from_breaks(
69                 breaks=excuts,
70                 copy=False, # default: False
71                 closed=intvclosed, # types of Pandas interval sides
72                 dtype=pd.IntervalDtype(subtype=intvsubtype) # types of
                    Pandas interval bounds
73             )
74             info['groups'] = {gr: intvs[gr] for gr in range(pnum+1)}
75         else: # categorical feature
76             info['groups'] = {gr: set() for gr in range(pnum+1)}
77             for val, gr in enumerate(cuts):
78                 info['groups'][gr].add(val) # categorical value in cut/
                    group
79
80     return tcuts

```


Code 4.8: True decision regions (module/model/findtrejs.py)

```

1  import numpy as np
2  import pandas as pd
3
4  from module.operation.xutil import max_dictval, itvpos
5
6
7  # Calculate new true decision regions and predictions (truly correct)
8  def findtrejs(tsels, tcuts, df, pdtype=np.int16):
9      '''
10         Usage: calculate new true decision regions and predictions (per
11                file)
12
13         Required arguments:
14
15             tsels: dictionary of selected variables and given number of
16                   cuts
17
18             tcuts: dictionary of cuts and groups along all selected
19                   features
20
21             df: training dataset including target variable (DataFrame, not
22                 iterator)
23
24         Optional arguments:
25
26             pdtype: NumPy data type of cut number (default: np.int16)
27
28         Outputs:
29
30             ttregs: dictionary of new true decision regions and their
31                   predicted classes
32
33         '''
34
35     ttregs = dict() # new true regions with predicted classes (truly
36                    correct)
37
38     classes = df['class'].unique() # all possible classes
39
40
41     for citer in tsels.keys():
42         regs = pd.Series([0]*len(df))
43         js = tsels[citer]['js']
44         pcutn = np.array(tsels[citer]['ps'], dtype=pdtype) # new cut
45                    numbers

```

```

28     pncum = np.cumprod(np.append([1], pcutn[0:-1]+1), dtype=pdtype) #
        cumulative number of new box regions
29     BN = np.prod(pcutn+1) # number of new regions
30
31     # Convert base representation of decision region to base 10
32     for ind, j in enumerate(js):
33         info = tcuts[citer][j]
34         attr = info['variable']
35         cuts = info['cuts']
36         if info['type'] == 'cont': # continuous feature
37             regs = regs + pncum[ind]*df[attr].apply(lambda x: itvpos(x,
                cuts))
38         else: # categorical feature
39             regs = regs + pncum[ind]*pd.Series([cuts[x] for x in df[attr
                ]])
40
41     # Find predicted classes in decision regions
42     ttregs[citer] = {
43         b: {
44             'classes': set(), # true predicted class set
45             'correct': 0, # number of instances correctly predicted
46             'ninst': 0, # number of training instances (total)
47             'ncinst': {n: 0 for n in range(len(classes))} # number of
                training instances in targets
48         } for b in range(BN)
49     }
50     for i in range(len(df)):
51         ttregs[citer][regs[i]]['ninst'] += 1 # instance in region
52         ttregs[citer][regs[i]]['ncinst'][df['class'][i]] += 1 #
            instance of specific target in region
53     for b in range(BN):
54         kmax, vmax = max_dictval(ttregs[citer][b]['ncinst']) # true
            majority voting
55         ttregs[citer][b]['classes'] = kmax # all classes that have
            maximum number of instances

```

```

56         ttregs[citer][b]['correct'] = vmax # maximum number of
           instances
57
58     return ttregs

```

Code 4.9: CPLEX decision regions (module/model/findcregs.py)

```

1  import numpy as np
2
3  from module.operation.typecast import strtoset
4  from module.operation.calregs import calregs
5
6
7  # Calculate new cplex decision regions and predictions (partially correct)
8  def findcregs(tsels, itpred, pcuto, idtype=np.int16, pdtype=np.int16):
9      '''
10         Usage: calculate new cplex decision regions and predictions (per
              file)
11         Required arguments:
12             tsels: dictionary of selected variables and given number of
                  cuts
13             itpred: individual result of cplex prediction (DataFrame
                  iterator)
14             pcuto: old cut numbers
15         Optional arguments:
16             pdtype: NumPy data type of cut number (default: np.int16)
17             idtype: NumPy data type of index (default: np.int16)
18         Outputs:
19             tcregs: dictionary of new cplex decision regions and their
                  predicted classes
20         '''
21
22     cproW = next(itpred) # iterator of instance predictions across all
           iterations
23     tcregs = dict() # new cplex regions with predicted classes (partially
           correct)

```

```

24     classes = set() # set all possible classes (collected from training
        dataset)
25
26     citer = -1 # current iteration
27
28     while True: # reported by cplex as occupied region
29         try:
30             if cprow.iter != citer: # new iteration
31                 citer = cprow.iter
32             if citer in tsels.keys(): # current iteration actually
                selects at least one feature
33                 keep = True # keep doing in this while loop
34                 pcutn = np.array(tsels[citer]['ps'], dtype=pdtype)
35                 sidx = np.array(tsels[citer]['js'], dtype=idtype) - 1 #
                    index starts at 0
36                 BN = np.prod(pcutn+1) # number of new regions
37                 bns = calregs(pcutn, sidx) # new corresponding regions
38                 tcregs[citer] = {
39                     b: {
40                         'lclasses': list(), # list of cplex predicted
                            class set
41                         'nlcinst': list() # list of instance number in
                            corresponding cplex class set
42                     } for b in range(BN)
43                 }
44             else: # current iteration selects no feature
45                 keep = False # update iterator and go to the next while
                    loop
46             if keep and cprow.iter == citer: # every record in iteration
                that selects feature
47                 creg = tcregs[citer][bns[cprow.region]] # new cplex region
48                 pset = strtoset(cpro.predict) # current set of classes
                    predicted by cplex
49                 classes = classes.union(pset) # add to set of all possible
                    classes
50             try: # current set of predicted classes already exists

```

```

51         creg['nlcinst'][creg['lclasses'].index(pset)] += 1
52     except ValueError: # new set of predicted classes
53         creg['lclasses'].append(pset)
54         creg['nlcinst'].append(1)
55         cprow = next(itpred) # update DataFrame iterator
56     except StopIteration:
57         break
58
59     for cregs in tcregs.values(): # reported by cplex as unoccupied region
60         for creg in cregs.values():
61             if not creg['lclasses']:
62                 creg['lclasses'] = [classes] # predict only one of the
63                 # entire set
64                 nlcinst = [0] # no instance reported by cplex in the rest of
65                 # new regions
66
67     return tcregs

```

Code 4.10: Classification correctness (module/model/findcorr.py)

```

1  # Find both true and recalculated cplex correctness
2  def findcorr(ttregs, tcregs):
3      '''
4          Usage: find both true and recalculated cplex correctness (per file)
5          Required arguments:
6              ttregs: dictionary of new true decision regions and their
6                  predicted classes
7              tcregs: dictionary of new cplex decision regions and their
7                  predicted classes
8          Outputs:
9              tcorr: true number of correctly classified instances per region
10             ccorr: recalculated cplex number of correctly classified
11                 instances per region
12             '''
13     tcorr = dict() # true correctness

```

```

14     ccorr = dict() # cplex correctness
15     for citer, tregs in tregs.items(): # true classification
16         tcorr[citer] = {
17             'correct': 0,
18             'detail': {b: tregs[b]['correct'] for b in tregs.keys()}
19         }
20         tcorr[citer]['correct'] = sum(tcorr[citer]['detail'].values())
21     for citer, cregs in tcregs.items(): # cplex classification
22         ccorr[citer] = {
23             'correct': 0,
24             'detail': {b: 0 for b in cregs.keys()}
25         }
26         for b in cregs.keys():
27             for soc in tcregs[citer][b]['lclasses']:
28                 ccorr[citer]['detail'][b] = max([tregs[citer][b]['ncinst'][
29                     c] for c in soc])
30         ccorr[citer]['correct'] = sum(ccorr[citer]['detail'].values())
31     return tcorr, ccorr

```

Code 4.11: Final mixed box classifier (finalbox.py)

```

1  import csv
2  import re
3  import pandas as pd
4
5  from module.operation.xutil import *
6  from module.operation.typecast import *
7  from module.operation.calregs import calregs
8  from module.model.findsels import findsels
9  from module.model.findcuts import findcuts
10 from module.model.findtregs import findtregs
11 from module.model.findcregs import findcregs
12 from module.model.findcorr import findcorr
13
14

```

```

15 # Parameters
16 pcuto = [3,3,2] # original cut numbers across all given features
17 isexample = True # whether example is shown
18 issreport = True # whether reports of feature selection are written
19 isrreport = True # whether reports of detailed decision regions are
    written
20
21 # Informational prefixes/postfixes
22 ts = "75305" # last digits of timestamp
23 data = "seltrain20num3each20" # data name (no file extension)
24 inprefix = f"{ts}-{data}-export-" # input filename prefix
25 inpostfix = "-mfullaltseltol-2-t-1440" # input filename postfix
26
27 # Required inputs
28 datdir = "../../../Projects/Box Classifiers/alternative/input" # directory
    of training instances (cplex inputs)
29 indir = "../../../Projects/Box Classifiers/alternative/output" # main
    input directory (cplex results)
30 datfile = f"{data}.csv" # training dataset with target variable
31 datpredfile = f"{inprefix}predict-instance-pcont-3{inpostfix}.csv" #
    individual result of cplex prediction
32 inerrfile = f"{inprefix}error{inpostfix}.csv" # classification errors and
    performance metrics
33 inselffile = f"{inprefix}select-var-str-pcont-3{inpostfix}.csv" # selected
    string variables
34 incutcontfile = f"{inprefix}cutcont-full-pcont-3{inpostfix}.csv" #
    continuous cuts
35 incutcatfile = f"{inprefix}cutcat-full-pcont-3{inpostfix}.csv" #
    categorical cuts
36
37 # Optional inputs
38 if issreport: # reports of feature selection must be written
39     metadir = "../../../Data/Encoded/metadata" # metadata directory
40     metafile = "meta-indep-pppub20enc.json" # metadata (after encoding)
    file

```

```

41     # Relabel case-insensitive NIU values for all selected categorical
        features
42     niudc = {'SS_YN': "NIU (aged below 15)", 'PEMLR': "NIU"}
43     if isrrreport: # reports of detailed decision regions must be written
44         clabels = {0: 'NNN', 1: 'NNY', 2: 'NY_', 3: 'YNN', 4: 'Y1Y'}
45
46     # Required outputs
47     outdir = f"../../Outputs/Main/Box/{data}" # main output directory
48     outeperffile = f"{ts}-eperf.csv" # classification performances (accuracy/
        error/time)
49     outselfile = f"{ts}-selvarfin.csv" # selected string variables, cuts and
        groups
50     outregfile = f"{ts}-predregfin.csv" # full decision regions
51
52     # Optional outputs
53     outcutcontfile = f"{ts}-cutcont.csv" # continuous cuts
54     outcutcatfile = f"{ts}-cutcat.csv" # categorical cuts
55     if issreport: # reports of feature selection must be written
56         outsrepwdfile = f"{ts}-report-sel-dup.csv" # with duplicate entries
57         outsrepndfile = f"{ts}-report-sel-nondup.csv" # with nonduplicate
            entries
58     if isrrreport: # reports of detailed decision regions
59         outrrepwdfile = f"{ts}-report-reg-dup.csv" # with duplicate entries
60         outrrepndfile = f"{ts}-report-reg-nondup.csv" # with nonduplicate
            entries
61
62     # Create main output directory (if not exist)
63     create_dir(outdir)
64
65     # Import datasets
66     dfe = pd.read_csv(f"{indir}/{inerrfile}") # cplex classification errors
        and performance metrics
67     dfs = pd.read_csv(f"{indir}/{inselfile}") # selected string variables
68     dfcont = pd.read_csv(f"{indir}/{incutcontfile}") # full continuous cuts
69     dfcat = pd.read_csv(f"{indir}/{incutcatfile}") # full categorical cuts

```



```

70 df = pd.read_csv(f"{datdir}/{datfile}") # training dataset including
    target variable
71 dfp = pd.read_csv(f"{indir}/{datpredfile}") # individual result of cplex
    prediction
72
73 # Initialize DataFrame iterators
74 itsel = dfs.itertuples() # selected string variables
75 itcont = dfcont.itertuples() # full continuous cuts
76 itcat = dfcat.itertuples() # full categorical cuts
77 itpred = dfp.itertuples() # individual result of cplex prediction
78
79 # Main execution
80 tsels = findsels(itself, pcuto) # selected variables
81 tcuts = findcuts(tsels, itcont, itcat) # cuts along all selected features
82 ttregs = findtregs(tsels, tcuts, df) # new true regions and predicted
    classes
83 tcregs = findcregs(tsels, itpred, pcuto) # new cplex regions and predicted
    classes
84 tcorr, ccorr = findcorr(ttregs, tcregs) # true/cplex correctness
85
86 # Calculate performance results
87 dfen = pd.DataFrame({
88     'iter': tcorr.keys(), # iteration that selects feature
89     'taccuracy': [info['correct']*100/len(df) for info in tcorr.values()],
        # true accuracies
90     'caccuracy': [info['correct']*100/len(df) for info in ccorr.values()],
        # recalculated cplex accuracies
91     'terror': [len(df) - info['correct'] for info in tcorr.values()], #
        true errors
92     'cerror': [len(df) - info['correct'] for info in ccorr.values()] #
        recalculated cplex errors
93 })
94 dfen = pd.merge(dfen, dfe, how='outer')
95 dfen.rename(columns = {
96     'error': 'rerror', # reported cplex errors
97     'accuracy': 'raccuracy' # reported cplex accuracies

```

```

98 }, inplace=True)
99 cols = dfen.columns.tolist()
100 new_cols = cols[0:1] + cols[5:5+len(pcuto)] + cols[1:3] + cols[-6:-5] +
        cols[3:5] + cols[-7:-6] + cols[-5:]
101 dfen = dfen[new_cols] # rearranged columns
102 dfen['ms'] = dfen['ms']/60000 # convert milliseconds to minutes
103 dfen = dfen.rename(columns={'ms':'minute'})
104
105 # Display performance results
106 print(f"\n{dfen}\n")
107
108 # Examples
109 if isexample:
110     iters = [1, 2, 15]
111     for citer in iters:
112         try:
113             print(f"Selected features (iteration {citer})\n{tsels[citer]}\n")
114             print(f"Cuts (iteration {citer})\n{tcuts[citer]}\n")
115             print(f"True decision regions (iteration {citer})\n{ttregs[
                citer]}\n")
116             print(f"Cplex decision regions (iteration {citer})\n{tcregs[
                citer]}\n")
117             print(f"True correctness (iteration {citer})\n{tcorr[citer]}\n")
118             print(f"Cplex correctness (iteration {citer})\n{ccorr[citer]}\n")
119         except KeyError:
120             print(f"Iteration {citer} selects no features\n")
121
122 # Export non-edited information
123 copy(f"{indir}/{incutcontfile}", f"{outdir}/{outcutcontfile}") #
        continuous cuts
124 copy(f"{indir}/{incutcatfile}", f"{outdir}/{outcutcatfile}") # categorical
        cuts
125

```

```

126 # Export performance results (accuracy/error/time)
127 dfen.to_csv(f"{outdir}/{outeperffile}", float_format="%.2f", header=True,
            index=False)
128
129 # Export selected variables, cuts and groups
130 with open(f"{outdir}/{outselfile}", 'w', newline='') as file:
131     writer = csv.DictWriter(
132         file,
133         fieldnames = [
134             'iter', 'jfin', 'j', 'var', 'type',
135             'p', 'cuts', 'groups'
136         ]
137     )
138     writer.writeheader()
139 for citer, info in tsels.items():
140     cuts = [[round(cut, 2) for cut in tcuts[citer][j]['cuts']] for j in
            info['js']]
141     groups = list()
142     for ind, j in enumerate(info['js']):
143         if info['types'][ind] == 'cont': # continuous feature
144             jgrs = dict()
145             for gr, member in tcuts[citer][j]['groups'].items():
146                 jgrs[gr] = itvtostr(member)
147             groups.append(jgrs)
148         else: # categorical feature
149             groups.append(tcuts[citer][j]['groups'])
150     dfstmp = pd.DataFrame({
151         'iter': citer,
152         'jfin': range(1, len(info['js'])+1), # 1, 2, ...
153         'j': info['js'], # j in cplex model
154         'variable': info['variables'],
155         'type': info['types'],
156         'p': info['ps'],
157         'cuts': cuts,
158         'groups': groups
159     })

```

```

160     dfstamp.to_csv(f"{outdir}/{outselfile}", mode='a', header=False, index=
        False)
161 del dfstamp
162
163 # Export predicted classes and number of instances in all decision regions
164 with open(f"{outdir}/{outregfile}", 'w', newline='') as file:
165     writer = csv.DictWriter(
166         file,
167         fieldnames = [
168             'iter', 'reg', 'ninst', 'tpred', 'lrpreds',
169             'tcorr', 'ccorr', 'ncinst'
170         ]
171     )
172     writer.writeheader()
173     for citer, tregs in ttregs.items():
174         for b, treg in tregs.items():
175             # List of predicted classes reported by cplex in string format
176             lrpreds = settostr(map(settostr, tregs[citer][b]['lclasses']),
                left='[', right=']')
177             writer.writerow({
178                 'iter': citer,
179                 'reg': b,
180                 'ninst': treg['ninst'], # number of instances
181                 'tpred': settostr(treg['classes']), # true predicted class
182                 'lrpreds': lrpreds,
183                 'tcorr': tcorr[citer]['detail'][b], # true correctness
184                 'ccorr': ccorr[citer]['detail'][b], # cplex correctness
185                 'ncinst': treg['ncinst'] # targets and number of member
                    instances
186             })
187
188
189 # Export final reports of feature selection (with duplicate/nonduplicate
        entries) (if specified)
190
191 if issreport: # reports of feature selection must be written

```

```

192
193     # New labels of selected categorical features (catvdc)
194     metadc = import_dict(jsonpath=f"{metadir}/{metafile}") # metadata after
        encoding
195     catvars = set() # all selected categorical features (initialized)
196     pattern = r'(^|[\^w])(niu)([\^w]|$)' # regex to search for niu
197     pattern = re.compile(pattern, re.IGNORECASE)
198     for info in tsels.values():
199         for ind, attr in enumerate(info['variables']):
200             if info['types'][ind] == 'cat':
201                 catvars.add(attr)
202     catvdc = {attr: metadc[attr]['values'] for attr in catvars} # labels of
        selected categorical features
203     for attr, valdc in catvdc.items():
204         for val, desc in valdc.items():
205             matches = re.search(pattern, desc.replace(',', ' '))
206             if bool(matches): # case-insensitive value label containing niu
207                 try:
208                     catvdc[attr][val] = niudc[attr] # relabel
209                 except KeyError: # new NIU label of current feature is
                    missing
210                 pass
211
212     # True classification accuracies and performance metrics
213     efields = ['iter', 'taccuracy', 'minute', 'acctmin', 'status']
214
215     # Groups
216     grls = list() # list of all member groups across all features and
        iterations
217     for citer, scuts in tcuts.items():
218         for j, info in scuts.items(): # cuts along all selected feature
219             vartype = 'Continuous' if info['type']=='cat' else 'Categorical'
                ,
220
                if info['type'] == 'cont': # continuous feature (groups not
                    displayed for convenience)
221                 for gr, member in info['groups'].items():

```

```

222         dc = {
223             'iter': citer,
224             'j': j, 'variable': info['variable'],
225             'type': 'Continuous',
226             'label': metadc[info['variable']]['label'],
227             'group': gr,
228             'member': itvtostr(member),
229             'desc': itvtodesc(member, decimals=0, extend=False).
                capitalize()
230         }
231         grls.append(dc)
232     else: # categorical feature (groups displayed)
233         for gr, member in info['groups'].items():
234             for elem in member: # all elements in group member
235                 desc = catvdc[info['variable']][str(elem)]
236                 dc = {
237                     'iter': citer,
238                     'j': j, 'variable': info['variable'],
239                     'type': 'Categorical',
240                     'label': metadc[info['variable']]['label'],
241                     'group': gr,
242                     'member': elem,
243                     'desc': desc
244                 }
245                 grls.append(dc)
246 dfg = pd.DataFrame(grls) # group dataframe
247
248 # Report dataframe of feature selection with duplicate entries (dfrp)
249 dfsrp = pd.merge(dfen[efields], dfg) # merge two dataframes: error/
    metric and group
250
251 # Report dataframe of feature selection with nonduplicate entries (dfn)
252 dfsrpn = nondup(
253     dfsrp,
254     ndcols=[
255         ['iter', 'taccuracy', 'minute', 'acctmin', 'status'],

```

```

256         ['j', 'variable', 'type', 'label'],
257         ['group']
258     ],
259     intcols=['iter', 'status', 'j', 'group'] # integer columns
260 )
261
262 # Export final reports of feature selection
263 dfsrp.to_csv( # with duplicate entries
264     f"{outdir}/{outsrepwdfile}",
265     float_format="%.2f",
266     header=True, index=False
267 )
268 dfsrpn.to_csv( # with nonduplicate entries
269     f"{outdir}/{outsrepndfile}",
270     sep=',', na_rep='',
271     float_format="%.2f",
272     header=True, index=False
273 )
274
275 print(f"{dfsrp.head()}\n") # feature selection (with duplicate entries)
276 print(f"{dfsrpn.head()}\n") # feature selection (with nonduplicate entries)
277
278
279 # Export final reports of detailed decision regions (with duplicate/
    nonduplicate entries) (if specified)
280
281 if isrrreport: # reports of detailed decision regions must be written
282
283     # Export final reports of detailed regions (with duplicate entries)
284     with open(f"{outdir}/{outrrepwdfile}", 'w', newline='') as file:
285         writer = csv.DictWriter(
286             file,
287             fieldnames = [
288                 'iter',
289                 'ordvars', 'strvars',

```



```

318         'strtpreds': ', '.join([clabels[v] for v in treg['
            classes']]), # true predicted class labels
319         'ninst': treg['ninst'] # number of training instances in
            region
320     })
321     for ind in range(len(ps)): # increment base representation
        of region for next for loop
322         qs[ind] += 1 # increment by 1
323         if qs[ind] > ps[ind]: qs[ind] = 0 # new leading one
324         else: break # same leading one
325
326     # Export final reports of detailed regions (with nonduplicate entries)
327     dfrrp = pd.read_csv(f"{outdir}/{outrrrepwdfile}")
328     dfrrpn = nondup(dfrrp, ndcols=[['iter', 'ordvars', 'strvars']], intcols
        =['iter'])
329     dfrrpn.to_csv( # with nonduplicate entries
330         f"{outdir}/{outrrrepndfile}",
331         sep=',', na_rep='',
332         header=True, index=False
333     )
334
335     print(f"{dfrrp.head()}\n") # detailed decision regions (with duplicate
        entries)
336     print(f"{dfrrpn.head()}\n") # detailed decision regions (with nonduplicate
        entries)
337
338
339 # Reexamination of CPLEX Results
340
341 # Additional output files
342 outexffile = f"{ts}-exam-full.csv" # full cplex reexamination
343 outexdfile = f"{ts}-exam-diff.csv" # difference in new decision regions
344 outexnfile = f"{ts}-exam-diffnum.csv" # number of difference
345
346 # Convert full coordinate to position in new feature space
347 def tonpos(citer, coord):

```

```

348     ls = list()
349     for j in tsels[citer]['js']:
350         if tcuts[citer][j]['type'] == 'cont':
351             ls.append(itvpos(coord[j-1], tcuts[citer][j]['cuts']))
352         else:
353             ls.append(tcuts[citer][j]['cuts'][coord[j-1]])
354     return tuple(ls)
355
356 # Compute new numerical region from given position to new feature space
357 def tonreg(citer, pos):
358     pcutn = np.array(tsels[citer]['ps'], dtype=np.int16)
359     pncum = np.cumprod(np.append([1], pcutn[0:-1]+1), dtype=np.int16)
360     return np.dot(pncum, pos)
361
362 dfpn = dfp.copy() # copy of individual result of cplex prediction
363 dfpn = dfpn[dfpn['iter'].isin(tsels.keys())] # exclude iterations of no
    feature selection
364
365 nregdc = dict() # new numerical regions in all iterations
366 for citer, info in tsels.items():
367     nregdc[citer] = calregs(pcuto=pcuto, sidx=np.array(info['js'])-1)
368 dfpn['creg'] = dfpn.apply(lambda x: nregdc[x.iter][x.region], axis=1) #
    new region based on cplex result
369 dfpn['cpred'] = dfpn.apply(lambda x: ttregs[x.iter][x.creg]['classes'],
    axis=1) # cplex predicted class
370
371 dfc = pd.merge(df, dfpn, how='right', left_on=df.index+1, right_on='id',
    suffixes=('', '_pn')) # include instance
372 del dfc['class_pn']
373 cols = dfc.columns.tolist()
374 new_cols = cols[len(pcuto)+1:len(pcuto)+3] + cols[0:len(pcuto)+1] + cols
    [-4:]
375 dfc = dfc[new_cols]
376 dfc = dfc.rename(columns={'region': 'rreg', 'predict': 'rpred'})
377 dfc['rpred'] = dfc['rpred'].apply(strtoset)
378

```

```

379 dfc['coord'] = dfc.iloc[:,2:len(pcuto)+2].apply(tuple, axis=1) # full
    original coordinate
380 dfc['tpos'] = dfc.apply(lambda x: tonpos(x.iter, x.coord), axis=1) # true
    position in new feature space
381 dfc['treg'] = dfc.apply(lambda x: tonreg(x.iter, x.tpos), axis=1) # true
    decision region
382 dfc['tpred'] = dfc.apply(lambda x: ttregs[x.iter][x.treg]['classes'], axis
    =1) # true predicted class
383
384 dfcd = dfc[dfc['creg'] != dfc['treg']] # new cplex region differs from new
    true region
385 dfcn = dfcd.groupby('iter').size().reset_index(name='dnum') # number of
    difference
386
387 print(f"{dfcn}\n") # display number of difference in region recalculation
388 print(f"{dfcd}\n") # display difference in new regions
389
390 # Export cplex reexamination results
391 dfc.to_csv(f"{outdir}/{outexffile}", header=True, index=False) # full
    cplex reexamination
392 dfcd.to_csv(f"{outdir}/{outexdfile}", header=True, index=False) #
    difference in new decision regions
393 dfcn.to_csv(f"{outdir}/{outexnfile}", header=True, index=False) #
    difference number

```

CHAPTER V

RESULTS ON HEALTH INSURANCE

5.1 Training Data

The box classifier proposed in Chapter 4 is illustrated on the sample of size 100 (25 per class) and three preselected features: A_AGE, PEMLR and SS_YN. The variable description and cross tabulation analysis with five bins on a continuous feature is displayed in Table 5.1. Each bin covers at least two different insurance coverage types. Although survey participants are unique, some sample records can be the same in feature and even in target due to initial preselection of features and resultant partial loss of personal information. The sampling result can be seen during Iteration 7 in Table 5.7. This chapter investigates two contributing factors out of three based solely on highest training accuracy.

Table 5.1: Cross tabulation of sample data by preselected variables and health insurance coverage types

Preselected Variable	Insurance Coverage Type (GRP, DIR, PUB)					
	NNN	NNY	NY_	Y1Y	YNN	
A_AGE: Age						
Universe: All persons						
(1.917, 18.6]	4	8	2	0	5	
(18.6, 35.2]	10	2	1	4	8	
(35.2, 51.8]	5	1	5	2	5	
(51.8, 68.4]	1	4	8	6	2	
(68.4, 85.0]	0	5	4	8	0	
PEMLR: Major labor force recode						
Universe: All persons						
0: NIU	4	5	2	0	4	
1: Employed - at work	8	3	7	9	12	
2: Employed - absent	0	0	3	1	0	
3: Unemployed - on layoff	1	1	0	0	0	
4: Unemployed - looking	1	1	1	0	2	
5: Not in labor force - retired	0	5	5	9	0	
6: Not in labor force - disabled	0	2	1	0	0	

Table 5.1: Cross tabulation of sample data by preselected variables and health insurance coverage types (continued)

Preselected Variable	Insurance Coverage Type (GRP, DIR, PUB)					
	NNN	NNY	NY_	Y1Y	YNN	
7: Not in labor force - other	6	3	1	1	2	
SS_YN: Who received social security payments either for themselves or as combined payments with other family members ?						
Universe: All persons aged 15+						
0: NIU	3	5	2	0	4	
1: Yes	0	9	7	10	1	
2: No	17	6	11	10	15	

5.2 Decision Tree

The goal is to find up to two significant determinants of health insurance coverage out of three features namely A_AGE, PEMLR and SS_YN. The first is continuous whereas the last two are categorical. Three splits are assumed in Code 4.1 on an individual feature. Since SS_YN has only three possible values, this feature can have up to two splits. In total, there should be at most $(3 + 1)(3 + 1) = 16$ decision boxes. As a result, decision trees of at least depth 3 and at most 16 leaf nodes are considered. Code 5.1 computes the trees of depths 3, 4 and 5 built by the Gini impurity within 5 seconds each as displayed in Figures 5.1, 5.2 and 5.3 respectively. They give training accuracies of 45%, 50% and 54% with 7, 11 and 15 splitting values in total and 8, 12 and 16 decision boxes. The two splits A_AGE = 70.5 and A_AGE = 75 in Figures 5.2 and 5.3 are redundant because both cannot distinguish the classes of training instances in left and right nodes by predicting the same class label 4.

Figure 5.1: Gini-based decision tree with depth 3, 7 non-leaf nodes and 8 leaf nodes giving a training accuracy of 45%

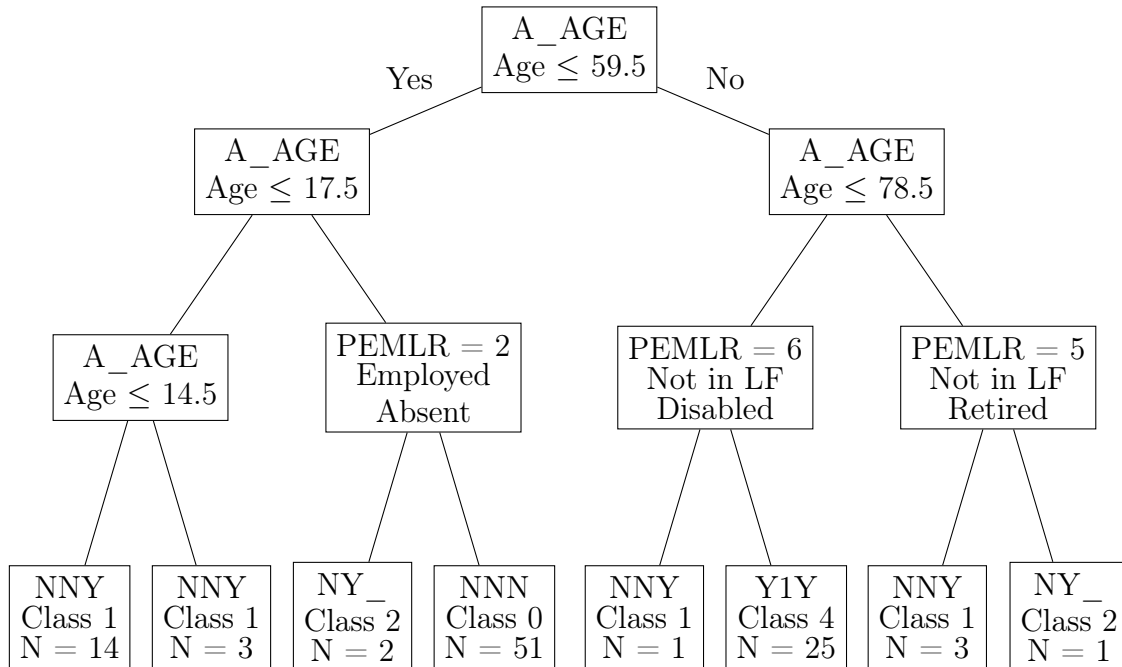


Figure 5.2: Gini-based decision tree with depth 4, 11 non-leaf nodes and 12 leaf nodes giving training accuracy of 50%

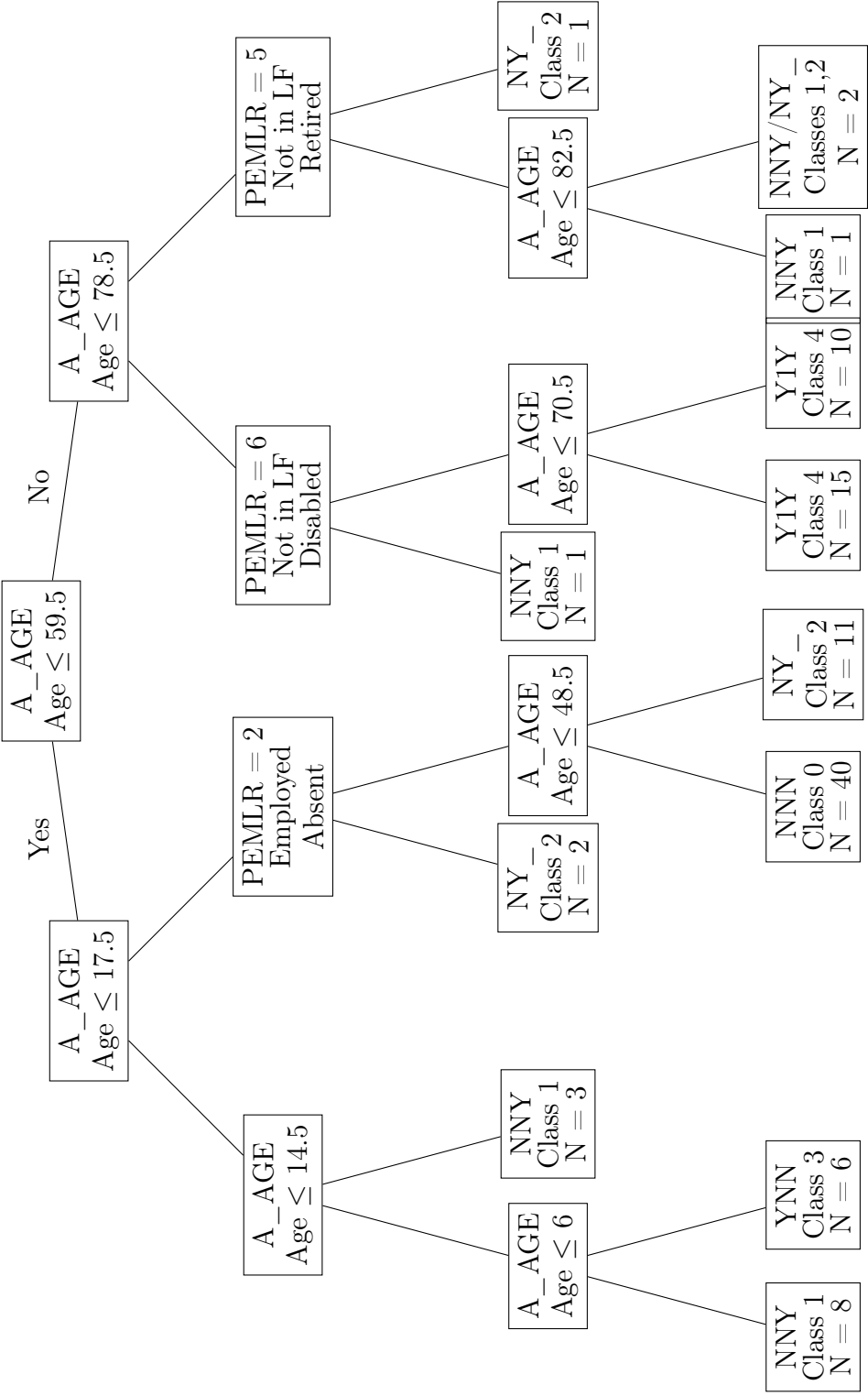
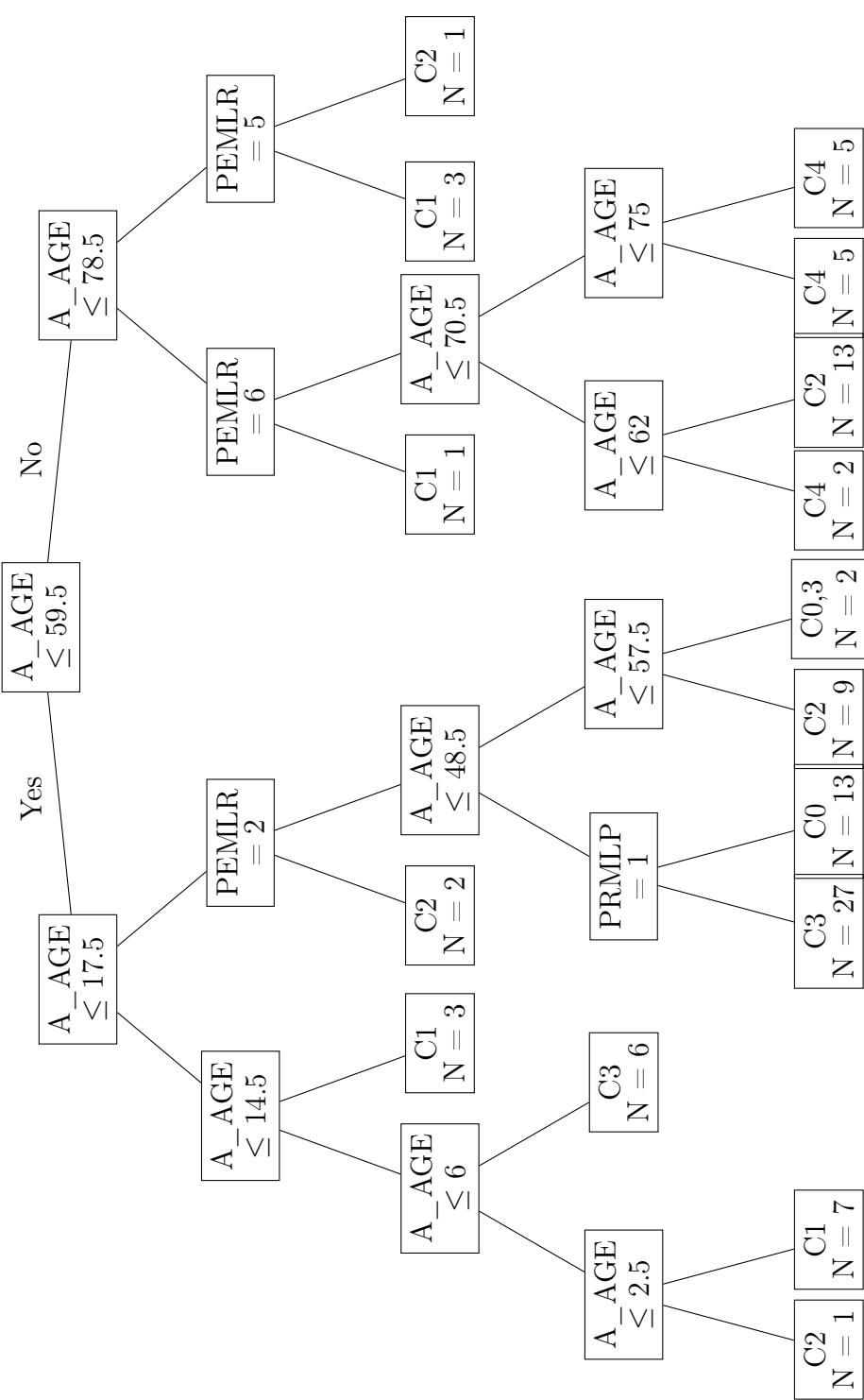


Figure 5.3: Gini-based decision tree with depth 5, 15 non-leaf nodes and 16 leaf nodes giving training accuracy of 54%



Code 5.1: Gini-based decision tree classifier

```

1  import matplotlib.pyplot as plt
2  import pandas as pd
3  import numpy as np
4  import csv
5  import os
6  from sklearn.tree import DecisionTreeClassifier, export_text, plot_tree
7
8  def create_dir(dir):
9      try:
10         os.makedirs(dir)
11     except FileExistsError:
12         pass
13
14  # Given Information
15  data_ls = []
16  data_ls.append({
17      'data': "../../../Samples/cplex/seltrain20num3each20.csv",
18      'info': "../../../Samples/cplex/selproc20num3co3ca3cutinfo.csv",
19      'configs': [
20          {'max_depth': 3, 'max_leaves': 16},
21          {'max_depth': 4, 'max_leaves': 16},
22          {'max_depth': 5, 'max_leaves': 16}
23      ],
24      'outdir': "../../../Outputs/Main/Tree"
25  })
26  print(f"{data_ls}\n")
27
28  # Decision Tree
29  def dtree(df_data, df_info, max_depth, max_leaves, data_path='', info_path
30           == ''):
31
32      # One-hot encoding
33      feat_cat = list(df_info[df_info['type'] == 'Categorical']['variable'])

```

```

33     for v in feat_cat:
34         df_data[v] = df_data[v].astype('category')
35     one_hot_data = pd.get_dummies(df_data[feat_cat], drop_first=True)
36     X = df_data.iloc[:,0:-(len(feat_cat)+1)].join(one_hot_data)
37     y = df_data['class']
38
39     # Build decision tree
40     clf = DecisionTreeClassifier(
41         max_depth=max_depth,
42         max_leaf_nodes=max_leaves,
43         random_state=0
44     )
45     clf.fit(X, y)
46
47     # Performance
48     score = clf.score(X, y)
49     y_pred = clf.predict(X)
50     err_ind = (y_pred != y.to_numpy().flatten()).astype(int)
51     error = np.count_nonzero(err_ind)
52     accuracy = (1-error/len(y_pred))*100
53
54     # Tree structure
55     depth = clf.tree_.max_depth
56     nodes = clf.tree_.node_count
57     leaves = clf.tree_.n_leaves
58     splits = nodes - leaves
59
60     # Decision tree summary
61     summary = {
62         'error': error, 'accuracy': accuracy, 'score': score,
63         'depth': depth,
64         'nodes': nodes, 'leaves': leaves, 'splits': splits
65     }
66
67     # Decision rules
68     rules = export_text(clf, feature_names=list(X.columns))

```

```

69
70     # Predicted values
71     df_pred = pd.DataFrame({
72         'y_true': df_data['class'],
73         'y_pred': y_pred,
74         'e': err_ind
75     })
76
77     # Display results
78     if data_path != '':
79         print(f"Data: {data_path}")
80     if info_path != '':
81         print(f"Info: {info_path}")
82     print(f"Maximum depth: {max_depth}")
83     print(f"Maximum number of leaves: {max_leaves}\n")
84     print(f"Categorical features: {feat_cat}")
85     print(f"X: {X.columns.values}\n")
86     print(f"Summary:")
87     print(f"\tDepth = {depth} | Leaves = {leaves}")
88     print(f"\tError = {error} | Accuracy = {accuracy} | Score = {score}")
89     print(f"\tNodes = {nodes} | Splits = {splits}\n")
90     print(f"Decision rules:\n{rules}\n")
91
92     # Return statement
93     return clf, summary, rules, df_pred
94
95 # Implementation
96 for dc in data_ls:
97
98     # Export information
99     datname = os.path.splitext(os.path.basename(dc['data']))[0] # without
        file extension
100     outdatdir = f"{dc['outdir']}/{datname}"
101     outprefix = datname
102     outsumfile = f"{outdatdir}/{outprefix}-summary.csv"
103     outruledir = f"{outdatdir}/rules"

```

```

104     outpreddir = f"{outdatdir}/prediction"
105     outfigdir = f"{outdatdir}/figures"
106
107     # Import
108     df_data = pd.read_csv(dc['data'])
109     df_info = pd.read_csv(dc['info'])
110
111     # Exported figure formats
112     fig_formats = ['svg', 'pgf', 'pdf']
113
114     # Create directories
115     create_dir(f"{outdatdir}/rules")
116     create_dir(f"{outdatdir}/prediction")
117     for format in fig_formats:
118         create_dir(f"{outdatdir}/figures/{format}")
119
120     # Export summary file in CSV format
121     with open(outsumfile, 'w') as sumfile:
122
123         sumheader = [
124             'mdepth', 'mleaves', 'depth', 'leaves',
125             'error', 'accuracy', 'score',
126             'nodes', 'splits'
127         ]
128         writer = csv.DictWriter(sumfile, fieldnames=sumheader)
129         writer.writeheader()
130
131         for config in dc['configs']:
132
133             # Tree configuration
134             mdepth = config['max_depth'] # depth
135             mleaves = config['max_leaves'] # number of leaves
136
137             # Postfix of exported files with specific depth and number of
138                 leaves
139             outpostfix = f"mdepth-{mdepth}-mleaves-{mleaves}"

```

```

139
140     # Decision tree
141     clf, summary, rules, df_pred = dtree(
142         df_data, df_info, mdepth, mleaves,
143         data_path=dc['data'], info_path=dc['info']
144     )
145
146     # Export summary result to CSV file
147     summary['mdepth'] = mdepth
148     summary['mleaves'] = mleaves
149     writer.writerow(summary)
150
151     # Decision rules
152     with open(f"{outruledir}/{outprefix}-rule-{outpostfix}.txt", 'w
153         ') as rulefile:
154         rulefile.write(rules)
155
156     # Prediction
157     outpredfile = f"{outpredidir}/{outprefix}-pred-{outpostfix}.csv"
158     df_pred.index = df_pred.index + 1
159     df_pred.to_csv(outpredfile, index_label='id')
160
161     # Tree plots
162     plot_tree(clf)
163     #plot_tree(clf, label='none', impurity=False)
164     for format in fig_formats:
165         outfigfile = f"{outfigdir}/{format}/{outprefix}-fig-{
166             outpostfix}.{format}"
167         plt.savefig(outfigfile, bbox_inches='tight')
168     #plt.show()
169
170     # Newline
171     print()

```

5.3 Proposed Model

A record of an MIP solution returned by a CPLEX solver is counted as an iteration. The proposed box classifier is given within 15 iterations as reported by the solver, or 13 iterations by careful reexamination, before all CPLEX node files fully occupy the reserved disk space of 200 GB where the optimal solution status is inconclusive. As shown in Tables 5.2 and 5.3, the box classifier gives six splitting values in total, three per each contributing factor, whereas all three decision trees at least seven. It achieves a high training accuracy of 51%, compared to the trees of 12 and 16 boxes at 50% and 54%. Although the first requires a significantly longer building time of at least 78.88 minutes (iteration 13) or up to 209.93 minutes (last iteration 15), the latter two output superfluous 11 and 15 total splits. Interestingly, the box classifier and all three decision trees consider A_AGE and PEMLR significant features, and they have consistent, though nonidentical, categorical splitting values on PEMLR. Based on the box classifier, $PEMLR = 3, 4, 5$ and 7 share similar characteristics, and they are grouped together as a new single unit or splitting value. Another group of $PEMLR = 0$ and 6 is also generated. Nonetheless, all decision trees lack the capability to bundle similar categorical values.

The training accuracy, the execution time and the minimum storage size of a box classifier per iteration are reported in Table 5.4. Feature selection occurs as of iteration 2. The training accuracy directly reported by a CPLEX solver as the negative of the objective value differs from the true accuracy produced and recomputed by the proposed box classifier based solely on the splitting values during the first 13 iterations. Certain training instances do not exactly lie until iteration 10 in their CPLEX decision regions, whether original or merged, as indicated by inconsistency between both CPLEX and true training accuracies when both region types are assumed to generate identical predictions. The acceptable box classifier of training accuracy 51% is given since iteration 13 within 78.88 minutes, taking up at least 5.92 GB of disk space but no more than 7 GB, and with a relative MIP gap of 6.35 defined by the relative difference between the best integer objective and the objective of the best CPLEX tree node remaining. The CPLEX engine log can be examined in an appendix.

Groups of values on selected features and their resultant box regions including predicted class labels are shown in Tables 5.5 and 5.6 respectively. Some bins as a result of feature splits may be empty, and their corresponding decision boxes are therefore nonexistent. The dimension of new continuous features in Code 4.1 is one, but iterations 2 to 9 select only categorical features. As a result, splits on the continuous feature `A_AGE` is redundant, and the number of decision boxes is overly reported by a CPLEX solver. After recalculating numerical decision regions and merging boxes, the difference between CPLEX and true decision regions occurs as illustrated on a per-instance basis in Table 5.7. This is possibly due to the insufficiently small CPLEX feasibility tolerance of 10^{-6} by default. At least 41 training instances suffer from this inconsistency, and all especially in iteration 7. No difference can be detected as of iteration 10.

Table 5.2: Comparison between multiple decision tree of depths 3 to 5 and proposed classifier in iterations 13 to 15 based on number of splitting values, number of decision boxes, training accuracy and execution time

Classification Model		Num of Splitting Values				Num of Boxes	Training Accuracy (%)	Execution Time (min)
Model	Specification	A_ AGE	PEMLR	SS_ YN	Total			
Decision tree	Depth of 3	4	3	0	7	8	45	0.08
	Depth of 4	8	3	0	11	12	50	
	Depth of 5	12	3	0	15	16	54	
Proposed classifier	Iteration 13	3	3	0	6	16	51	78.88
	Iteration 14	3	3	0	6	16	51	82.02
	Iteration 15	3	3	0	6	16	51	209.93

Table 5.3: Splitting values on features of multiple decision tree of depths 3 to 5 and proposed classifier in iterations 13 to 15

Classification Model		Splitting Values			Training Accuracy (%)
Model	Specification	A_AGE	PEMLR	SS_YN	
Decision tree	Depth of 3	14.5, 17.5, 59.5, 78.5	{2}, {5}, {6}, {0, 1, 3, 4, 7}	-	45
	Depth of 4	6, 14.5, 17.5, 48.5, 59.5, 70.5, 78.5, 82.5	{2}, {5}, {6}, {0, 1, 3, 4, 7}	-	50
	Depth of 5	2.5, 6, 14.5, 17.5, 48.5, 57.5, 59.5, 62, 70.5, 75, 78.5	{2}, {5}, {6}, {0, 1, 3, 4, 7}	-	54
Proposed classifier	Iteration 13	24.99, 55.99, 64.99	{2}, {1}, {3, 4, 5, 7}, {0, 6}	-	51
	Iterations 14 to 15	24.01, 55.99, 64.99	{2}, {1}, {3, 4, 5, 7}, {0, 6}	-	51

Table 5.4: Training accuracy, execution time, minimum storage usage, relative MIP gap and number of inconsistent data across all iterations

Iteration	Accuracy (%)		Execution Time (min)		Min Storage (GB)			Rel Gap	Inconsistent
	True	CPLEX	Reported	Each	Accum	Tree	Nodes	Comp	
1			20	0	0			279	
2	38	35	28	0.03	0.03			27.57	41
3	38	35	31	0.01	0.04			22.14	41
4	38	35	36	0.01	0.06			17.25	41
5	38	35	38	0.03	0.09			15.5	41
6	40	36	39	13.3	13.39	0.99	0	8.67	41
7	40	30	40	5.27	18.66	1.24	0	8.42	100
8	43	40	43	4.64	23.3	2.74	0.49	7.75	41
9	44	42	44	7.67	30.97	3.68	1.3	7.54	41
10	47	47	46	37.23	68.2	3.35	1.34	7.01	
11	48	48	48	1.18	69.38	3.46	1.5	6.67	
12	50	50	49	7.17	76.55	4.11	1.64	6.51	
13	51	51	50	2.33	78.88	8.13	5.92	6.35	
14	51	51	51	3.14	82.02	9.06	7	6.2	
15	51	51	51	127.91	209.93	192.68	190.58	167.06	6.08

Table 5.5: Selected variables and groups of values across all iterations

Iteration	Selected Variable			Group	Member	
	Index	Symbol	Type		Index	Label
2	2	PEMLR	Categorical	0	1	Employed - at work
					3	Unemployed - on layoff
					7	Not in labor force - other
					5	Not in labor force - retired
				2		
				3	0	NIU
	3	SS_YN	Categorical	0	2	Employed - absent
					4	Unemployed - looking
					6	Not in labor force - disabled
					2	No
				1	1	Yes
				2	0	NIU (aged below 15)
3	2	PEMLR	Categorical	0	1	Employed - at work
					3	Unemployed - on layoff
					7	Not in labor force - other
				2	5	Not in labor force - retired
				3	0	NIU

Table 5.5: Selected variables and groups of values across all iterations (continued)

Iteration	Selected Variable			Group	Member	
	Index	Symbol	Type		Index	Label
	3	SS_YN	Categorical	0	2	Employed - absent
					4	Unemployed - looking
					6	Not in labor force - disabled
					2	No
					1	Yes
					0	NIU (aged below 15)
	4	PEMLR	Categorical	0	1	Employed - at work
					3	Unemployed - on layoff
					7	Not in labor force - other
					5	Not in labor force - retired
					0	NIU
					2	Employed - absent
					4	Unemployed - looking
					6	Not in labor force - disabled
	3	SS_YN	Categorical	0	2	No
					1	Yes

Table 5.5: Selected variables and groups of values across all iterations (continued)

Iteration	Selected Variable		Group	Member	
	Index	Symbol	Type	Index	Label
5	2	PEMLR	Categorical	2	NIU (aged below 15)
				0	
				1	Employed - at work
				3	Unemployed - on layoff
				7	Not in labor force - other
				5	Not in labor force - retired
				0	NIU
	3	SS_YN	Categorical	2	Employed - absent
				4	Unemployed - looking
				6	Not in labor force - disabled
				2	No
				1	Yes
				0	NIU (aged below 15)
6	2	PEMLR	Categorical	0	Employed - at work
				3	Unemployed - on layoff
				7	Not in labor force - other
				2	Employed - absent

Table 5.5: Selected variables and groups of values across all iterations (continued)

Iteration	Selected Variable			Group	Member	
	Index	Symbol	Type		Index	Label
	3	SS_YN	Categorical	2	5	Not in labor force - retired
				3	0	NIU
					4	Unemployed - looking
					6	Not in labor force - disabled
				0	2	No
				1	1	Yes
				2	0	NIU (aged below 15)
					1	Employed - at work
					2	Employed - absent
					4	Unemployed - looking
	2	PEMLR	Categorical	0	0	NIU
					3	Unemployed - on layoff
					6	Not in labor force - disabled
					7	Not in labor force - other
					5	Not in labor force - retired
					0	NIU (aged below 15)
					1	Employed - at work
					2	Employed - absent
					4	Unemployed - looking
					0	NIU
	3	SS_YN	Categorical	2	3	Unemployed - on layoff
					6	Not in labor force - disabled
					7	Not in labor force - other
					5	Not in labor force - retired
				3	0	NIU (aged below 15)
					1	Employed - at work
					2	Employed - absent
					4	Unemployed - looking
					0	NIU
					3	Unemployed - on layoff

Table 5.5: Selected variables and groups of values across all iterations (continued)

Iteration	Selected Variable		Group	Member	
	Index	Symbol	Type	Index	Label
8	2	PEMLR	Categorical	2	No
				1	Yes
				2	Employed - absent
				1	Employed - at work
				6	Not in labor force - disabled
				0	NIU
				3	Unemployed - on layoff
				4	Unemployed - looking
				7	Not in labor force - other
				5	Not in labor force - retired
3	SS_YN	Categorical	3	2	No
			2	0	NIU (aged below 15)
				1	Yes
			0	2	Employed - absent
			1	1	Employed - at work
			2	0	NIU
9	2	PEMLR	Categorical	0	Employed - absent
				1	Employed - at work
				2	NIU

Table 5.5: Selected variables and groups of values across all iterations (continued)

Iteration	Selected Variable			Group	Member	
	Index	Symbol	Type		Index	Label
	3	SS_YN	Categorical		3	Unemployed - on layoff
					4	Unemployed - looking
					6	Not in labor force - disabled
					7	Not in labor force - other
					5	Not in labor force - retired
				3		
				0	2	No
				2	0	NIU (aged below 15)
					1	Yes
10	1	A_AGE	Continuous	0	$(-\infty, 24.01)$	Below 24
					$(24.01, 40.99)$	Between 25 and 40
					$(40.99, 65.99)$	Between 41 and 65
					$(65.99, \infty)$	Above 66
				3		
				0	2	Employed - absent
				1	1	Employed - at work
					7	Not in labor force - other
				2	4	Unemployed - looking

Table 5.5: Selected variables and groups of values across all iterations (continued)

Iteration	Selected Variable		Group	Member	
	Index	Symbol	Type	Index	Label
11	1	A_AGE	Continuous	5	Not in labor force - retired
				0	NIU
				3	Unemployed - on layoff
				6	Not in labor force - disabled
	2	PEMLR	Categorical	0	Below 24
				1	Between 25 and 40
				2	Between 41 and 64
				3	Above 65
	3			2	Employed - absent
				1	Employed - at work
				7	Not in labor force - other
				4	Unemployed - looking
12	1			5	Not in labor force - retired
				0	NIU
				3	Unemployed - on layoff
				6	Not in labor force - disabled

Table 5.5: Selected variables and groups of values across all iterations (continued)

Iteration	Selected Variable			Group	Member	
	Index	Symbol	Type		Index	Label
12	1	A_AGE	Continuous	0	$(-\infty, 24.99)$	Below 24
				1	$(24.99, 40.01)$	Between 25 and 40
				2	$(40.00, 64.01)$	Between 41 and 64
				3	$(64.01, \infty)$	Above 65
	2	PEMLR	Categorical	0	2	Employed - absent
				1	1	Employed - at work
				2	4	Unemployed - looking
					5	Not in labor force - retired
					7	Not in labor force - other
				3	0	NIU
					3	Unemployed - on layoff
					6	Not in labor force - disabled
13	1	A_AGE	Continuous	0	$(-\infty, 24.99)$	Below 24
				1	$(24.99, 55.99)$	Between 25 and 55
				2	$(55.99, 64.99)$	Between 56 and 64
				3	$(64.99, \infty)$	Above 65

Table 5.5: Selected variables and groups of values across all iterations (continued)

Iteration	Selected Variable			Group	Member	
	Index	Symbol	Type		Index	Label
	2	PEMLR	Categorical	0	2	Employed - absent
				1	1	Employed - at work
				2	3	Unemployed - on layoff
					4	Unemployed - looking
					5	Not in labor force - retired
					7	Not in labor force - other
				3	0	NIU
14	1	A_AGE	Continuous	0	$(-\infty, 24.01)$	Below 24
				1	$(24.01, 55.99)$	Between 25 and 55
				2	$(55.99, 64.99)$	Between 56 and 64
				3	$(64.99, \infty)$	Above 65
	2	PEMLR	Categorical	0	2	Employed - absent
				1	1	Employed - at work
				2	3	Unemployed - on layoff
					4	Unemployed - looking
					6	Not in labor force - disabled

Table 5.5: Selected variables and groups of values across all iterations (continued)

Iteration	Selected Variable		Group	Member	
	Index	Symbol	Type	Index	Label
15	1	A_AGE	Continuous	5	Not in labor force - retired
				7	Not in labor force - other
				0	NIU
				6	Not in labor force - disabled
	2	PEMLR	Categorical	0	Below 24
				1	Between 25 and 55
				2	Between 56 and 64
				3	Above 65
	3			2	Employed - absent
				1	Employed - at work
				3	Unemployed - on layoff
				4	Unemployed - looking
	4			5	Not in labor force - retired
				7	Not in labor force - other
				0	NIU
				6	Not in labor force - disabled

Table 5.6: Decision regions and predicted class labels across all iterations

Iter	Selected Variables		Decision Region			Predicted Classes		Num
	Tuple	Symbol	Ind	Tuple	Cross Product	Ind	Label	
2	(2,3)	PEMLR, SS_YN	0	(0,0)	$\{1, 3, 7\} \times \{2\}$	0	NNN	48
			1	(1,0)	$\emptyset \times \{2\}$	0,1,2,3,4	NNN, NNY, NY_-, YNN, Y1Y	0
			2	(2,0)	$\{5\} \times \{2\}$	2	NY_-	3
			3	(3,0)	$\{0, 2, 4, 6\} \times \{2\}$	2	NY_-	8
			4	(0,1)	$\{1, 3, 7\} \times \{1\}$	2,4	NY_-, Y1Y	6
			5	(1,1)	$\emptyset \times \{1\}$	0,1,2,3,4	NNN, NNY, NY_-, YNN, Y1Y	0
			6	(2,1)	$\{5\} \times \{1\}$	4	Y1Y	16
			7	(3,1)	$\{0, 2, 4, 6\} \times \{1\}$	1	NNY	5
			8	(0,2)	$\{1, 3, 7\} \times \{0\}$	0,1,2,3,4	NNN, NNY, NY_-, YNN, Y1Y	0
			9	(1,2)	$\emptyset \times \{0\}$	0,1,2,3,4	NNN, NNY, NY_-, YNN, Y1Y	0
			10	(2,2)	$\{5\} \times \{0\}$	0,1,2,3,4	NNN, NNY, NY_-, YNN, Y1Y	0
			11	(3,2)	$\{0, 2, 4, 6\} \times \{0\}$	1	NNY	14
3	(2,3)	PEMLR, SS_YN	0	(0,0)	$\{1, 3, 7\} \times \{2\}$	0	NNN	48
			1	(1,0)	$\emptyset \times \{2\}$	0,1,2,3,4	NNN, NNY, NY_-, YNN, Y1Y	0
			2	(2,0)	$\{5\} \times \{2\}$	2	NY_-	3
			3	(3,0)	$\{0, 2, 4, 6\} \times \{2\}$	2	NY_-	8

Table 5.6: Decision regions and predicted class labels across all iterations (continued)

Iter	Selected Variables		Decision Region			Predicted Classes		Num
	Tuple	Symbol	Ind	Tuple	Cross Product	Ind	Label	
4	(2,3)	PEMLR, SS_YN	4	(0,1)	$\{1, 3, 7\} \times \{1\}$	2,4	NY_-, Y1Y	6
			5	(1,1)	$\emptyset \times \{1\}$	0,1,2,3,4	NNN, NNY, NY_-, YNN, Y1Y	0
			6	(2,1)	$\{5\} \times \{1\}$	4	Y1Y	16
			7	(3,1)	$\{0, 2, 4, 6\} \times \{1\}$	1	NNY	5
			8	(0,2)	$\{1, 3, 7\} \times \{0\}$	0,1,2,3,4	NNN, NNY, NY_-, YNN, Y1Y	0
			9	(1,2)	$\emptyset \times \{0\}$	0,1,2,3,4	NNN, NNY, NY_-, YNN, Y1Y	0
			10	(2,2)	$\{5\} \times \{0\}$	0,1,2,3,4	NNN, NNY, NY_-, YNN, Y1Y	0
			11	(3,2)	$\{0, 2, 4, 6\} \times \{0\}$	1	NNY	14
			0	(0,0)	$\{1, 3, 7\} \times \{2\}$	0	NNN	48
			1	(1,0)	$\emptyset \times \{2\}$	0,1,2,3,4	NNN, NNY, NY_-, YNN, Y1Y	0
			2	(2,0)	$\{5\} \times \{2\}$	2	NY_-	3
5	(2,3)	PEMLR, SS_YN	3	(3,0)	$\{0, 2, 4, 6\} \times \{2\}$	2	NY_-	8
			4	(0,1)	$\{1, 3, 7\} \times \{1\}$	2,4	NY_-, Y1Y	6
			5	(1,1)	$\emptyset \times \{1\}$	0,1,2,3,4	NNN, NNY, NY_-, YNN, Y1Y	0
			6	(2,1)	$\{5\} \times \{1\}$	4	Y1Y	16
			7	(3,1)	$\{0, 2, 4, 6\} \times \{1\}$	1	NNY	5

Table 5.6: Decision regions and predicted class labels across all iterations (continued)

Iter	Selected Variables		Decision Region			Predicted Classes		Num
	Tuple	Symbol	Ind	Tuple	Cross Product	Ind	Label	
5	(2,3)	PEMLR, SS_YN	8	(0,2)	$\{1, 3, 7\} \times \{0\}$	0,1,2,3,4	NNN, NNY, NY_-, YNN, Y1Y	0
			9	(1,2)	$\emptyset \times \{0\}$	0,1,2,3,4	NNN, NNY, NY_-, YNN, Y1Y	0
			10	(2,2)	$\{5\} \times \{0\}$	0,1,2,3,4	NNN, NNY, NY_-, YNN, Y1Y	0
			11	(3,2)	$\{0, 2, 4, 6\} \times \{0\}$	1	NNY	14
			0	(0,0)	$\{1, 3, 7\} \times \{2\}$	0	NNN	48
			1	(1,0)	$\emptyset \times \{2\}$	0,1,2,3,4	NNN, NNY, NY_-, YNN, Y1Y	0
			2	(2,0)	$\{5\} \times \{2\}$	2	NY_-	3
			3	(3,0)	$\{0, 2, 4, 6\} \times \{2\}$	2	NY_-	8
			4	(0,1)	$\{1, 3, 7\} \times \{1\}$	2,4	NY_-, Y1Y	6
			5	(1,1)	$\emptyset \times \{1\}$	0,1,2,3,4	NNN, NNY, NY_-, YNN, Y1Y	0
			6	(2,1)	$\{5\} \times \{1\}$	4	Y1Y	16
6	(2,3)	PEMLR, SS_YN	7	(3,1)	$\{0, 2, 4, 6\} \times \{1\}$	1	NNY	5
			8	(0,2)	$\{1, 3, 7\} \times \{0\}$	0,1,2,3,4	NNN, NNY, NY_-, YNN, Y1Y	0
			9	(1,2)	$\emptyset \times \{0\}$	0,1,2,3,4	NNN, NNY, NY_-, YNN, Y1Y	0
			10	(2,2)	$\{5\} \times \{0\}$	0,1,2,3,4	NNN, NNY, NY_-, YNN, Y1Y	0
			11	(3,2)	$\{0, 2, 4, 6\} \times \{0\}$	1	NNY	14
			0	(0,0)	$\{1, 3, 7\} \times \{2\}$	0	NNN	48

Table 5.6: Decision regions and predicted class labels across all iterations (continued)

Iter	Selected Variables		Decision Region			Predicted Classes		Num
	Tuple	Symbol	Ind	Tuple	Cross Product	Ind	Label	
6	(2,3)	PEMLR, SS_YN	0	(0,0)	$\{1, 3, 7\} \times \{2\}$	0	NNN	48
			1	(1,0)	$\{2\} \times \{2\}$	2	NY_	3
			2	(2,0)	$\{5\} \times \{2\}$	2	NY_	3
			3	(3,0)	$\{0, 4, 6\} \times \{2\}$	0,3	NNN, YNN	5
			4	(0,1)	$\{1, 3, 7\} \times \{1\}$	2,4	NY_ , Y1Y	6
			5	(1,1)	$\{2\} \times \{1\}$	2	NY_	1
			6	(2,1)	$\{5\} \times \{1\}$	4	Y1Y	16
			7	(3,1)	$\{0, 4, 6\} \times \{1\}$	1	NNY	4
			8	(0,2)	$\{1, 3, 7\} \times \{0\}$	0,1,2,3,4	NNN, NNY, NY_ , YNN, Y1Y	0
			9	(1,2)	$\{2\} \times \{0\}$	0,1,2,3,4	NNN, NNY, NY_ , YNN, Y1Y	0
			10	(2,2)	$\{5\} \times \{0\}$	0,1,2,3,4	NNN, NNY, NY_ , YNN, Y1Y	0
			11	(3,2)	$\{0, 4, 6\} \times \{0\}$	1	NNY	14

Table 5.6: Decision regions and predicted class labels across all iterations (continued)

Iter	Selected Variables		Decision Region			Predicted Classes		Num	
	Tuple	Symbol	Ind	Tuple	Cross Product	Ind	Label		
			4	(0,1)	$\{1, 2, 4\} \times \{0, 2\}$	3	YNN	42	
			5	(1,1)	$\emptyset \times \{0, 2\}$	0,1,2,3,4	NNN, NNY, NY $_{-}$, YNN, Y1Y	0	
			6	(2,1)	$\{0, 3, 6, 7\} \times \{0, 2\}$	0	NNN	28	
			7	(3,1)	$\{5\} \times \{0, 2\}$	2	NY $_{-}$	3	
			8	(0,2)	$\{1, 2, 4\} \times \{1\}$	2	NY $_{-}$	6	
			9	(1,2)	$\emptyset \times \{1\}$	0,1,2,3,4	NNN, NNY, NY $_{-}$, YNN, Y1Y	0	
			10	(2,2)	$\{0, 3, 6, 7\} \times \{1\}$	1	NNY	5	
			11	(3,2)	$\{5\} \times \{1\}$	4	Y1Y	16	
	8	(2,3)	PEMLR, SS_YN	0	(0,0)	$\{2\} \times \{2\}$	2	NY $_{-}$	3
				1	(1,0)	$\{1, 6\} \times \{2\}$	3	YNN	35
				2	(2,0)	$\{0, 3, 4, 7\} \times \{2\}$	0	NNN	18
3				(3,0)	$\{5\} \times \{2\}$	2	NY $_{-}$	3	
4				(0,1)	$\{2\} \times \emptyset$	0,1,2,3,4	NNN, NNY, NY $_{-}$, YNN, Y1Y	0	
5				(1,1)	$\{1, 6\} \times \emptyset$	0,1,2,3,4	NNN, NNY, NY $_{-}$, YNN, Y1Y	0	
6				(2,1)	$\{0, 3, 4, 7\} \times \emptyset$	0,1,2,3,4	NNN, NNY, NY $_{-}$, YNN, Y1Y	0	
7	(3,1)	$\{5\} \times \emptyset$	0,1,2,3,4	NNN, NNY, NY $_{-}$, YNN, Y1Y	0				

Table 5.6: Decision regions and predicted class labels across all iterations (continued)

Iter	Selected Variables		Decision Region		Predicted Classes		Num
	Tuple	Symbol	Ind	Tuple	Cross Product	Ind	Label
9	(2,3)	PEMLR, SS_YN	8	(0,2)	$\{2\} \times \{0,1\}$	2	NY_
			9	(1,2)	$\{1,6\} \times \{0,1\}$	2	NY_
			10	(2,2)	$\{0,3,4,7\} \times \{0,1\}$	1	NNY
			11	(3,2)	$\{5\} \times \{0,1\}$	4	Y1Y
			0	(0,0)	$\{2\} \times \{2\}$	2	NY_
			1	(1,0)	$\{1\} \times \{2\}$	3	YNN
			2	(2,0)	$\{0,3,4,6,7\} \times \{2\}$	0	NNN
			3	(3,0)	$\{5\} \times \{2\}$	2	NY_
			4	(0,1)	$\{2\} \times \emptyset$	0,1,2,3,4	NNN, NNY, NY_-, YNN, Y1Y
			5	(1,1)	$\{1\} \times \emptyset$	0,1,2,3,4	NNN, NNY, NY_-, YNN, Y1Y
			6	(2,1)	$\{0,3,4,6,7\} \times \emptyset$	0,1,2,3,4	NNN, NNY, NY_-, YNN, Y1Y
10	(2,2)	PEMLR, SS_YN	7	(3,1)	$\{5\} \times \emptyset$	0,1,2,3,4	NNN, NNY, NY_-, YNN, Y1Y
			8	(0,2)	$\{2\} \times \{0,1\}$	2	NY_
			9	(1,2)	$\{1\} \times \{0,1\}$	2	NY_
			10	(2,2)	$\{0,3,4,6,7\} \times \{0,1\}$	1	NNY
			11	(3,2)	$\{5\} \times \{0,1\}$	4	Y1Y
			0	(0,0)	$\{2\} \times \{2\}$	2	NY_
			1	(1,0)	$\{1\} \times \{2\}$	3	YNN
			2	(2,0)	$\{0,3,4,6,7\} \times \{2\}$	0	NNN
			3	(3,0)	$\{5\} \times \{2\}$	2	NY_
			4	(0,1)	$\{2\} \times \emptyset$	0,1,2,3,4	NNN, NNY, NY_-, YNN, Y1Y
			5	(1,1)	$\{1\} \times \emptyset$	0,1,2,3,4	NNN, NNY, NY_-, YNN, Y1Y
11	(2,2)	PEMLR, SS_YN	6	(2,1)	$\{0,3,4,6,7\} \times \emptyset$	0,1,2,3,4	NNN, NNY, NY_-, YNN, Y1Y
			7	(3,1)	$\{5\} \times \emptyset$	0,1,2,3,4	NNN, NNY, NY_-, YNN, Y1Y
			8	(0,2)	$\{2\} \times \{0,1\}$	2	NY_
			9	(1,2)	$\{1\} \times \{0,1\}$	2	NY_
			10	(2,2)	$\{0,3,4,6,7\} \times \{0,1\}$	1	NNY
			11	(3,2)	$\{5\} \times \{0,1\}$	4	Y1Y
			0	(0,0)	$\{2\} \times \{2\}$	2	NY_
			1	(1,0)	$\{1\} \times \{2\}$	3	YNN
			2	(2,0)	$\{0,3,4,6,7\} \times \{2\}$	0	NNN
			3	(3,0)	$\{5\} \times \{2\}$	2	NY_
			4	(0,1)	$\{2\} \times \emptyset$	0,1,2,3,4	NNN, NNY, NY_-, YNN, Y1Y
			5	(1,1)	$\{1\} \times \emptyset$	0,1,2,3,4	NNN, NNY, NY_-, YNN, Y1Y

Table 5.6: Decision regions and predicted class labels across all iterations (continued)

Iter	Selected Variables		Decision Region			Predicted Classes		Num
	Tuple	Symbol	Ind	Tuple	Cross Product	Ind	Label	
10	(1,2)	A_ AGE, PEMPLR	0	(0,0)	$(-\infty, 24.01) \times \{2\}$	0,1,2,3,4	NNN, NNY, NY_-, YNN, Y1Y	0
			1	(1,0)	$(24.01, 40.99) \times \{2\}$	2	NY_-	2
			2	(2,0)	$(40.99, 65.99) \times \{2\}$	4	Y1Y	1
			3	(3,0)	$(65.99, \infty) \times \{2\}$	2	NY_-	1
			4	(0,1)	$(-\infty, 24.01) \times \{1, 7\}$	0	NNN	11
			5	(1,1)	$(24.01, 40.99) \times \{1, 7\}$	3	YNN	17
			6	(2,1)	$(40.99, 65.99) \times \{1, 7\}$	3	YNN	20
			7	(3,1)	$(65.99, \infty) \times \{1, 7\}$	2,4	NY_-, Y1Y	4
			8	(0,2)	$(-\infty, 24.01) \times \{4, 5\}$	1,3	NNY, YNN	2
			9	(1,2)	$(24.01, 40.99) \times \{4, 5\}$	0,3	NNN, YNN	2
			10	(2,2)	$(40.99, 65.99) \times \{4, 5\}$	2	NY_-	4
			11	(3,2)	$(65.99, \infty) \times \{4, 5\}$	4	Y1Y	16
			12	(0,3)	$(-\infty, 24.01) \times \{0, 3, 6\}$	1	NNY	15
			13	(1,3)	$(24.01, 40.99) \times \{0, 3, 6\}$	0	NNN	1
			14	(2,3)	$(40.99, 65.99) \times \{0, 3, 6\}$	1	NNY	3
			15	(3,3)	$(65.99, \infty) \times \{0, 3, 6\}$	1	NNY	1

Table 5.6: Decision regions and predicted class labels across all iterations (continued)

Iter	Selected Variables		Decision Region			Predicted Classes		Num
	Tuple	Symbol	Ind	Tuple	Cross Product	Ind	Label	
11	(1,2)	A_ AGE, PEMPLR	0	(0,0)	$(-\infty, 24.01) \times \{2\}$	0,1,2,3,4	NNN, NNY, NY_-, YNN, Y1Y	0
			1	(1,0)	$(24.01, 40.99) \times \{2\}$	2	NY_-	2
			2	(2,0)	$(40.99, 64.99) \times \{2\}$	4	Y1Y	1
			3	(3,0)	$(64.99, \infty) \times \{2\}$	2	NY_-	1
			4	(0,1)	$(-\infty, 24.01) \times \{1, 7\}$	0	NNN	11
			5	(1,1)	$(24.01, 40.99) \times \{1, 7\}$	3	YNN	17
			6	(2,1)	$(40.99, 64.99) \times \{1, 7\}$	3	YNN	18
			7	(3,1)	$(64.99, \infty) \times \{1, 7\}$	2,4	NY_-, Y1Y	6
			8	(0,2)	$(-\infty, 24.01) \times \{4, 5\}$	1,3	NNY, YNN	2
			9	(1,2)	$(24.01, 40.99) \times \{4, 5\}$	0,3	NNN, YNN	2
			10	(2,2)	$(40.99, 64.99) \times \{4, 5\}$	2	NY_-	4
			11	(3,2)	$(64.99, \infty) \times \{4, 5\}$	4	Y1Y	16
			12	(0,3)	$(-\infty, 24.01) \times \{0, 3, 6\}$	1	NNY	15
			13	(1,3)	$(24.01, 40.99) \times \{0, 3, 6\}$	0	NNN	1
			14	(2,3)	$(40.99, 64.99) \times \{0, 3, 6\}$	1	NNY	3
15	(3,3)	$(64.99, \infty) \times \{0, 3, 6\}$	1	NNY	1			

Table 5.6: Decision regions and predicted class labels across all iterations (continued)

Iter	Selected Variables		Decision Region		Predicted Classes		Num
	Tuple	Symbol	Ind	Tuple	Cross Product	Ind	Label
12	(1,2)	A AGE, PEMLR	0	(0,0)	$(-\infty, 24.99) \times \{2\}$	0,1,2,3,4	NNN, NNY, NY_-, YNN, Y1Y
			1	(1,0)	$(24.99, 40.01) \times \{2\}$	2	NY_-
			2	(2,0)	$(40.00, 64.01) \times \{2\}$	4	Y1Y
			3	(3,0)	$(64.01, \infty) \times \{2\}$	2	NY_-
			4	(0,1)	$(-\infty, 24.99) \times \{1\}$	0	NNN
			5	(1,1)	$(24.99, 40.01) \times \{1\}$	3	YNN
			6	(2,1)	$(40.00, 64.01) \times \{1\}$	3	YNN
			7	(3,1)	$(64.01, \infty) \times \{1\}$	2	NY_-
			8	(0,2)	$(-\infty, 24.99) \times \{4, 5, 7\}$	1	NNY
			9	(1,2)	$(24.99, 40.01) \times \{4, 5, 7\}$	0	NNN
			10	(2,2)	$(40.00, 64.01) \times \{4, 5, 7\}$	2	NY_-
			11	(3,2)	$(64.01, \infty) \times \{4, 5, 7\}$	4	Y1Y
			12	(0,3)	$(-\infty, 24.99) \times \{0, 3, 6\}$	1	NNY
			13	(1,3)	$(24.99, 40.01) \times \{0, 3, 6\}$	0	NNN
			14	(2,3)	$(40.00, 64.01) \times \{0, 3, 6\}$	1	NNY
			15	(3,3)	$(64.01, \infty) \times \{0, 3, 6\}$	1	NNY

Table 5.6: Decision regions and predicted class labels across all iterations (continued)

Iter	Selected Variables		Decision Region			Predicted Classes		Num
	Tuple	Symbol	Ind	Tuple	Cross Product	Ind	Label	
13	(1,2)	A_ AGE, PEMLR	0	(0,0)	$(-\infty, 24.99) \times \{2\}$	0,1,2,3,4	NNN, NNY, NY_-, YNN, Y1Y	0
			1	(1,0)	$(24.99, 55.99) \times \{2\}$	2	NY_	2
			2	(2,0)	$(55.99, 64.99) \times \{2\}$	4	Y1Y	1
			3	(3,0)	$(64.99, \infty) \times \{2\}$	2	NY_	1
			4	(0,1)	$(-\infty, 24.99) \times \{1\}$	0	NNN	7
			5	(1,1)	$(24.99, 55.99) \times \{1\}$	3	YNN	23
			6	(2,1)	$(55.99, 64.99) \times \{1\}$	3	YNN	4
			7	(3,1)	$(64.99, \infty) \times \{1\}$	2	NY_	5
			8	(0,2)	$(-\infty, 24.99) \times \{3, 4, 5, 7\}$	1	NNY	6
			9	(1,2)	$(24.99, 55.99) \times \{3, 4, 5, 7\}$	0	NNN	9
			10	(2,2)	$(55.99, 64.99) \times \{3, 4, 5, 7\}$	2	NY_	7
			11	(3,2)	$(64.99, \infty) \times \{3, 4, 5, 7\}$	4	Y1Y	17
			12	(0,3)	$(-\infty, 24.99) \times \{0, 6\}$	1	NNY	15
			13	(1,3)	$(24.99, 55.99) \times \{0, 6\}$	1	NNY	1
			14	(2,3)	$(55.99, 64.99) \times \{0, 6\}$	2	NY_	1
		15	(3,3)	$(64.99, \infty) \times \{0, 6\}$	1	NNY	1	

Table 5.6: Decision regions and predicted class labels across all iterations (continued)

Iter	Selected Variables		Decision Region		Predicted Classes		Num
	Tuple	Symbol	Ind	Tuple	Cross Product	Ind	Label
14	(1,2)	A AGE, PEMLR	0	(0,0)	$(-\infty, 24.01) \times \{2\}$	0,1,2,3,4	NNN, NNY, NY ₋ , YNN, Y1Y
			1	(1,0)	$(24.01, 55.99) \times \{2\}$	2	NY ₋
			2	(2,0)	$(55.99, 64.99) \times \{2\}$	4	Y1Y
			3	(3,0)	$(64.99, \infty) \times \{2\}$	2	NY ₋
			4	(0,1)	$(-\infty, 24.01) \times \{1\}$	0	NNN
			5	(1,1)	$(24.01, 55.99) \times \{1\}$	3	YNN
			6	(2,1)	$(55.99, 64.99) \times \{1\}$	3	YNN
			7	(3,1)	$(64.99, \infty) \times \{1\}$	2	NY ₋
			8	(0,2)	$(-\infty, 24.01) \times \{3, 4, 5, 7\}$	1	NNY
			9	(1,2)	$(24.01, 55.99) \times \{3, 4, 5, 7\}$	0	NNN
			10	(2,2)	$(55.99, 64.99) \times \{3, 4, 5, 7\}$	2	NY ₋
			11	(3,2)	$(64.99, \infty) \times \{3, 4, 5, 7\}$	4	Y1Y
			12	(0,3)	$(-\infty, 24.01) \times \{0, 6\}$	1	NNY
			13	(1,3)	$(24.01, 55.99) \times \{0, 6\}$	1	NNY
			14	(2,3)	$(55.99, 64.99) \times \{0, 6\}$	2	NY ₋
			15	(3,3)	$(64.99, \infty) \times \{0, 6\}$	1	NNY

Table 5.6: Decision regions and predicted class labels across all iterations (continued)

Iter	Selected Variables		Decision Region		Predicted Classes		Num
	Tuple	Symbol	Ind	Tuple	Cross Product	Ind	Label
15	(1,2)	A AGE, PEMLR	0	(0,0)	$(-\infty, 24.01) \times \{2\}$	0,1,2,3,4	NNN, NNY, NY ₋ , YNN, Y1Y
			1	(1,0)	$(24.01, 55.99) \times \{2\}$	2	NY ₋
			2	(2,0)	$(55.99, 64.99) \times \{2\}$	4	Y1Y
			3	(3,0)	$(64.99, \infty) \times \{2\}$	2	NY ₋
			4	(0,1)	$(-\infty, 24.01) \times \{1\}$	0	NNN
			5	(1,1)	$(24.01, 55.99) \times \{1\}$	3	YNN
			6	(2,1)	$(55.99, 64.99) \times \{1\}$	3	YNN
			7	(3,1)	$(64.99, \infty) \times \{1\}$	2	NY ₋
			8	(0,2)	$(-\infty, 24.01) \times \{3, 4, 5, 7\}$	1	NNY
			9	(1,2)	$(24.01, 55.99) \times \{3, 4, 5, 7\}$	0	NNN
			10	(2,2)	$(55.99, 64.99) \times \{3, 4, 5, 7\}$	2	NY ₋
			11	(3,2)	$(64.99, \infty) \times \{3, 4, 5, 7\}$	4	Y1Y
			12	(0,3)	$(-\infty, 24.01) \times \{0, 6\}$	1	NNY
			13	(1,3)	$(24.01, 55.99) \times \{0, 6\}$	1	NNY
			14	(2,3)	$(55.99, 64.99) \times \{0, 6\}$	2	NY ₋
			15	(3,3)	$(64.99, \infty) \times \{0, 6\}$	1	NNY

Table 5.7: Inconsistency between numerical CPLEX and true decision regions

Iter	Training Instance					Reported		CPLEX		True	
	ID	A	AGE	PEMLR	SSYN	Target	Region	Predict	Region	Predict	Region
2	8	4	0	0	0	0	38	1	9	0, 1, 2, 3, 4	11
	10	12	0	0	0	0	38	1	9	0, 1, 2, 3, 4	11
	20	10	0	0	0	0	38	1	9	0, 1, 2, 3, 4	11
	21	85	5	5	1	1	22	4	5	0, 1, 2, 3, 4	6
	22	74	5	5	1	1	22	4	5	0, 1, 2, 3, 4	6
	23	64	5	5	1	1	22	4	5	0, 1, 2, 3, 4	6
	24	73	5	5	1	1	22	4	5	0, 1, 2, 3, 4	6
	26	5	0	0	0	1	38	1	9	0, 1, 2, 3, 4	11
	27	4	0	0	0	1	38	1	9	0, 1, 2, 3, 4	11
	28	10	0	0	0	1	38	1	9	0, 1, 2, 3, 4	11
	29	54	6	6	1	1	26	1	6	4	7
	30	3	0	0	0	1	38	1	9	0, 1, 2, 3, 4	11
	33	17	4	4	1	1	26	1	6	4	7
	35	77	6	6	1	1	26	1	6	4	7
	36	5	0	0	0	1	38	1	9	0, 1, 2, 3, 4	11
	37	80	5	5	1	1	22	4	5	0, 1, 2, 3, 4	6

Table 5.7: Inconsistency between numerical CPLEX and true decision regions (continued)

Iter	Training Instance				Reported		CPLEX		True	
	ID	A	AGE	PEMLR	SSYN	Target	Region	Predict	Region	Predict
40	21	7	1	1	1	14	2	2	3	4
44	79	1	1	1	2	14	2	2	3	4
47	5	0	0	0	2	38	1	0, 1, 2, 3, 4	9	11
48	76	5	1	1	2	22	4	0, 1, 2, 3, 4	5	6
51	2	0	0	0	2	38	1	0, 1, 2, 3, 4	9	11
53	67	1	1	1	2	14	2	2	3	4
54	67	5	1	1	2	22	4	0, 1, 2, 3, 4	5	6
56	85	5	1	1	2	22	4	0, 1, 2, 3, 4	5	6
58	70	2	1	1	2	26	1	4	6	7
60	56	6	1	1	2	26	1	4	6	7
64	63	1	1	1	3	14	2	2	3	4
65	14	0	0	0	3	38	1	0, 1, 2, 3, 4	9	11
74	4	0	0	0	3	38	1	0, 1, 2, 3, 4	9	11
75	12	0	0	0	3	38	1	0, 1, 2, 3, 4	9	11
78	7	0	0	0	3	38	1	0, 1, 2, 3, 4	9	11
87	73	5	1	1	4	22	4	0, 1, 2, 3, 4	5	6
90	76	5	1	1	4	22	4	0, 1, 2, 3, 4	5	6

Table 5.7: Inconsistency between numerical CPLEX and true decision regions (continued)

Iter	Training Instance					Reported		CPLEX		True	
	ID	A	AGE	PEMLR	SSYN	Target	Region	Predict	Region	Predict	Region
	91	77	5	1	1	4	22	4	5	0, 1, 2, 3, 4	6
	93	71	1	1	1	4	14	2	3	2	4
	94	70	5	1	1	4	22	4	5	0, 1, 2, 3, 4	6
	95	78	5	1	1	4	22	4	5	0, 1, 2, 3, 4	6
	96	67	7	1	1	4	14	2	3	2	4
	97	71	5	1	1	4	22	4	5	0, 1, 2, 3, 4	6
	98	66	5	1	1	4	22	4	5	0, 1, 2, 3, 4	6
	99	67	5	1	1	4	22	4	5	0, 1, 2, 3, 4	6
	3	8	4	0	0	0	38	1	9	0, 1, 2, 3, 4	11
	10	12	0	0	0	0	38	1	9	0, 1, 2, 3, 4	11
	20	10	0	0	0	0	38	1	9	0, 1, 2, 3, 4	11
	21	85	5	1	1	1	22	4	5	0, 1, 2, 3, 4	6
	22	74	5	1	1	1	22	4	5	0, 1, 2, 3, 4	6
	23	64	5	1	1	1	22	4	5	0, 1, 2, 3, 4	6
	24	73	5	1	1	1	22	4	5	0, 1, 2, 3, 4	6
	26	5	0	0	0	1	38	1	9	0, 1, 2, 3, 4	11

Table 5.7: Inconsistency between numerical CPLEX and true decision regions (continued)

Iter	Training Instance				Reported		CPLEX		True	
	ID	A	AGE	PEMLR	SSYN	Target	Region	Predict	Region	Predict
27	4	0	0	0	0	1	38	1	9	0, 1, 2, 3, 4
28	10	0	0	0	0	1	38	1	9	0, 1, 2, 3, 4
29	54	6	1	1	1	1	26	1	6	4
30	3	0	0	0	0	1	38	1	9	0, 1, 2, 3, 4
33	17	4	1	1	1	1	26	1	6	4
35	77	6	1	1	1	1	26	1	6	4
36	5	0	0	0	0	1	38	1	9	0, 1, 2, 3, 4
37	80	5	1	1	1	1	22	4	5	0, 1, 2, 3, 4
40	21	7	1	1	1	1	14	2	3	2
44	79	1	1	1	1	2	14	2	3	2
47	5	0	0	0	0	2	38	1	9	0, 1, 2, 3, 4
48	76	5	1	1	1	2	22	4	5	0, 1, 2, 3, 4
51	2	0	0	0	0	2	38	1	9	0, 1, 2, 3, 4
53	67	1	1	1	1	2	14	2	3	2
54	67	5	1	1	1	2	22	4	5	0, 1, 2, 3, 4
56	85	5	1	1	1	2	22	4	5	0, 1, 2, 3, 4
58	70	2	1	1	1	2	26	1	6	4

Table 5.7: Inconsistency between numerical CPLEX and true decision regions (continued)

Iter	Training Instance				Reported		CPLEX		True	
	ID	A	AGE	PEMLR	SSYN	Target	Region	Predict	Region	Predict
	60	56	6	6	1	2	26	1	6	4
	64	63	1	1	1	3	14	2	3	2
	65	14	0	0	0	3	38	1	9	0, 1, 2, 3, 4
	74	4	0	0	0	3	38	1	9	0, 1, 2, 3, 4
	75	12	0	0	0	3	38	1	9	0, 1, 2, 3, 4
	78	7	0	0	0	3	38	1	9	0, 1, 2, 3, 4
	87	73	5	5	1	4	22	4	5	0, 1, 2, 3, 4
	90	76	5	5	1	4	22	4	5	0, 1, 2, 3, 4
	91	77	5	5	1	4	22	4	5	0, 1, 2, 3, 4
	93	71	1	1	1	4	14	2	3	2
	94	70	5	5	1	4	22	4	5	0, 1, 2, 3, 4
	95	78	5	5	1	4	22	4	5	0, 1, 2, 3, 4
	96	67	7	7	1	4	14	2	3	2
	97	71	5	5	1	4	22	4	5	0, 1, 2, 3, 4
	98	66	5	5	1	4	22	4	5	0, 1, 2, 3, 4
	99	67	5	5	1	4	22	4	5	0, 1, 2, 3, 4

Table 5.7: Inconsistency between numerical CPLEX and true decision regions (continued)

Iter	Training Instance				Reported		CPLEX		True	
	ID	A	AGE	PEMLR	SSYN	Target	Region	Predict	Region	Predict
4	8	4		0	0	0	38	1	9	0, 1, 2, 3, 4
	10	12		0	0	0	38	1	9	0, 1, 2, 3, 4
	20	10		0	0	0	38	1	9	0, 1, 2, 3, 4
	21	85		5	1	1	22	4	5	0, 1, 2, 3, 4
	22	74		5	1	1	22	4	5	0, 1, 2, 3, 4
	23	64		5	1	1	22	4	5	0, 1, 2, 3, 4
	24	73		5	1	1	22	4	5	0, 1, 2, 3, 4
	26	5		0	0	1	38	1	9	0, 1, 2, 3, 4
	27	4		0	0	1	38	1	9	0, 1, 2, 3, 4
	28	10		0	0	1	38	1	9	0, 1, 2, 3, 4
	29	54		6	1	1	26	1	6	4
	30	3		0	0	1	38	1	9	0, 1, 2, 3, 4
	33	17		4	1	1	26	1	6	4
	35	77		6	1	1	26	1	6	4
	36	5		0	0	1	38	1	9	0, 1, 2, 3, 4
	37	80		5	1	1	22	4	5	0, 1, 2, 3, 4

Table 5.7: Inconsistency between numerical CPLEX and true decision regions (continued)

Iter	Training Instance				Reported		CPLEX		True	
	ID	A	AGE	PEMLR	SSYN	Target	Region	Predict	Region	Predict
40	21	7	1	1	1	14	2	2	3	4
44	79	1	1	1	2	14	2	2	3	4
47	5	0	0	0	2	38	1	0, 1, 2, 3, 4	9	11
48	76	5	1	1	2	22	4	0, 1, 2, 3, 4	5	6
51	2	0	0	0	2	38	1	0, 1, 2, 3, 4	9	11
53	67	1	1	1	2	14	2	2	3	4
54	67	5	1	1	2	22	4	0, 1, 2, 3, 4	5	6
56	85	5	1	1	2	22	4	0, 1, 2, 3, 4	5	6
58	70	2	1	1	2	26	1	4	6	7
60	56	6	1	1	2	26	1	4	6	7
64	63	1	1	1	3	14	2	2	3	4
65	14	0	0	0	3	38	1	0, 1, 2, 3, 4	9	11
74	4	0	0	0	3	38	1	0, 1, 2, 3, 4	9	11
75	12	0	0	0	3	38	1	0, 1, 2, 3, 4	9	11
78	7	0	0	0	3	38	1	0, 1, 2, 3, 4	9	11
87	73	5	1	1	4	22	4	0, 1, 2, 3, 4	5	6
90	76	5	1	1	4	22	4	0, 1, 2, 3, 4	5	6

Table 5.7: Inconsistency between numerical CPLEX and true decision regions (continued)

Iter	Training Instance				Reported		CPLEX		True	
	ID	A	AGE	PEMLR	SSYN	Target	Region	Predict	Region	Predict
	27	4		0	0	1	38	1	9	0, 1, 2, 3, 4
	28	10		0	0	1	38	1	9	0, 1, 2, 3, 4
	29	54		6	1	1	26	1	6	4
	30	3		0	0	1	38	1	9	0, 1, 2, 3, 4
	33	17		4	1	1	26	1	6	4
	35	77		6	1	1	26	1	6	4
	36	5		0	0	1	38	1	9	0, 1, 2, 3, 4
	37	80		5	1	1	22	4	5	0, 1, 2, 3, 4
	40	21		7	1	1	14	2	3	2
	44	79		1	1	2	14	2	3	2
	47	5		0	0	2	38	1	9	0, 1, 2, 3, 4
	48	76		5	1	2	22	4	5	0, 1, 2, 3, 4
	51	2		0	0	2	38	1	9	0, 1, 2, 3, 4
	53	67		1	1	2	14	2	3	2
	54	67		5	1	2	22	4	5	0, 1, 2, 3, 4
	56	85		5	1	2	22	4	5	0, 1, 2, 3, 4
	58	70		2	1	2	26	1	6	4

Table 5.7: Inconsistency between numerical CPLEX and true decision regions (continued)

Iter	Training Instance				Reported		CPLEX		True	
	ID	A	AGE	PEMLR	SSYN	Target	Region	Predict	Region	Predict
	60	56	6	6	1	2	26	1	6	4
	64	63	1	1	1	3	14	2	3	2
	65	14	0	0	0	3	38	1	9	0, 1, 2, 3, 4
	74	4	0	0	0	3	38	1	9	0, 1, 2, 3, 4
	75	12	0	0	0	3	38	1	9	0, 1, 2, 3, 4
	78	7	0	0	0	3	38	1	9	0, 1, 2, 3, 4
	87	73	5	5	1	4	22	4	5	0, 1, 2, 3, 4
	90	76	5	5	1	4	22	4	5	0, 1, 2, 3, 4
	91	77	5	5	1	4	22	4	5	0, 1, 2, 3, 4
	93	71	1	1	1	4	14	2	3	2
	94	70	5	5	1	4	22	4	5	0, 1, 2, 3, 4
	95	78	5	5	1	4	22	4	5	0, 1, 2, 3, 4
	96	67	7	7	1	4	14	2	3	2
	97	71	5	5	1	4	22	4	5	0, 1, 2, 3, 4
	98	66	5	5	1	4	22	4	5	0, 1, 2, 3, 4
	99	67	5	5	1	4	22	4	5	0, 1, 2, 3, 4

Table 5.7: Inconsistency between numerical CPLEX and true decision regions (continued)

Iter	Training Instance				Reported		CPLEX		True	
	ID	A	AGE	PEMLR	SSYN	Target	Region	Predict	Region	Predict
40	21	7	1	1	1	14	2, 3	0, 3	4	2, 4
44	79	1	1	1	2	14	2, 3	0, 3	4	2, 4
47	5	0	0	0	2	38	1	0, 1, 2, 3, 4	11	1
48	76	5	1	1	2	22	4	2	6	4
51	2	0	0	0	2	38	1	0, 1, 2, 3, 4	11	1
53	67	1	1	1	2	14	2, 3	0, 3	4	2, 4
54	67	5	1	1	2	22	4	2	6	4
56	85	5	1	1	2	22	4	2	6	4
58	70	2	1	1	2	18	2	2, 4	5	2
60	56	6	1	1	2	26	1	4	7	1
64	63	1	1	1	3	14	2, 3	0, 3	4	2, 4
65	14	0	0	0	3	38	1	0, 1, 2, 3, 4	11	1
74	4	0	0	0	3	38	1	0, 1, 2, 3, 4	11	1
75	12	0	0	0	3	38	1	0, 1, 2, 3, 4	11	1
78	7	0	0	0	3	38	1	0, 1, 2, 3, 4	11	1
87	73	5	1	1	4	22	4	2	6	4
90	76	5	1	1	4	22	4	2	6	4

Table 5.7: Inconsistency between numerical CPLEX and true decision regions (continued)

Iter	Training Instance					Reported		CPLEX		True	
	ID	A	AGE	PEMLR	SSYN	Target	Region	Predict	Region	Predict	Region
	91	77	5	1	1	4	22	4	5	2	6
	93	71	1	1	1	4	14	2, 3	3	0, 3	4
	94	70	5	1	1	4	22	4	5	2	6
	95	78	5	1	1	4	22	4	5	2	6
	96	67	7	1	1	4	14	2, 3	3	0, 3	4
	97	71	5	1	1	4	22	4	5	2	6
	98	66	5	1	1	4	22	4	5	2	6
	99	67	5	1	1	4	22	4	5	2	6
	1	24	1	2	2	0	14	3	3	0, 1, 2, 3, 4	4
	2	58	7	2	2	0	22	0	5	0, 1, 2, 3, 4	6
7	3	24	1	2	2	0	14	3	3	0, 1, 2, 3, 4	4
	4	40	7	2	2	0	22	0	5	0, 1, 2, 3, 4	6
	5	24	1	2	2	0	14	3	3	0, 1, 2, 3, 4	4
	6	26	1	2	2	0	14	3	3	0, 1, 2, 3, 4	4
	7	18	7	2	2	0	22	0	5	0, 1, 2, 3, 4	6
	8	4	0	0	0	0	22	0	5	0, 1, 2, 3, 4	6

Table 5.7: Inconsistency between numerical CPLEX and true decision regions (continued)

Iter	Training Instance				Reported		CPLEX		True	
	ID	A	AGE	PEMLR	SSYN	Target	Region	Predict	Region	Predict
	9	38		3	2	0	22	0	5	0, 1, 2, 3, 4
	10	12		0	0	0	22	0	5	0, 1, 2, 3, 4
	11	46		7	2	0	22	0	5	0, 1, 2, 3, 4
	12	26		1	2	0	14	3	3	0, 1, 2, 3, 4
	13	35		7	2	0	22	0	5	0, 1, 2, 3, 4
	14	19		7	2	0	22	0	5	0, 1, 2, 3, 4
	15	29		4	2	0	14	3	3	0, 1, 2, 3, 4
	16	24		0	2	0	22	0	5	0, 1, 2, 3, 4
	17	35		1	2	0	14	3	3	0, 1, 2, 3, 4
	18	48		1	2	0	14	3	3	0, 1, 2, 3, 4
	19	41		1	2	0	14	3	3	0, 1, 2, 3, 4
	20	10		0	0	0	22	0	5	0, 1, 2, 3, 4
	21	85		5	1	1	38	4	9	0, 1, 2, 3, 4
	22	74		5	1	1	38	4	9	0, 1, 2, 3, 4
	23	64		5	1	1	38	4	9	0, 1, 2, 3, 4
	24	73		5	1	1	38	4	9	0, 1, 2, 3, 4
	25	15		7	2	1	22	0	5	0, 1, 2, 3, 4

Table 5.7: Inconsistency between numerical CPLEX and true decision regions (continued)

Iter	Training Instance				Reported		CPLEX		True	
	ID	A	AGE	PEMLR	SSYN	Target	Region	Predict	Region	Predict
	26	5		0	0	1	22	0	5	0, 1, 2, 3, 4
	27	4		0	0	1	22	0	5	0, 1, 2, 3, 4
	28	10		0	0	1	22	0	5	0, 1, 2, 3, 4
	29	54		6	1	1	34	1	8	2
	30	3		0	0	1	22	0	5	0, 1, 2, 3, 4
	31	45		3	2	1	22	0	5	0, 1, 2, 3, 4
	32	28		1	2	1	14	3	3	0, 1, 2, 3, 4
	33	17		4	1	1	26	2	6	0
	34	57		1	2	1	14	3	3	0, 1, 2, 3, 4
	35	77		6	1	1	34	1	8	2
	36	5		0	0	1	22	0	5	0, 1, 2, 3, 4
	37	80		5	1	1	38	4	9	0, 1, 2, 3, 4
	38	16		1	2	1	14	3	3	0, 1, 2, 3, 4
	39	57		7	2	1	22	0	5	0, 1, 2, 3, 4
	40	21		7	1	1	34	1	8	2
	41	56		4	2	2	14	3	3	0, 1, 2, 3, 4
	42	64		5	2	2	26	2	6	0

Table 5.7: Inconsistency between numerical CPLEX and true decision regions (continued)

Iter	Training Instance				Reported		CPLEX		True	
	ID	A	AGE	PEMLR	SSYN	Target	Region	Predict	Region	Predict
43	38	2	2	2	2	2	14	3	3	0, 1, 2, 3, 4
44	79	1	1	1	2	2	26	2	6	0
45	57	7	2	2	2	2	22	0	5	0, 1, 2, 3, 4
46	65	1	2	2	2	2	14	3	3	0, 1, 2, 3, 4
47	5	0	0	0	2	2	22	0	5	0, 1, 2, 3, 4
48	76	5	1	1	2	2	38	4	9	0, 1, 2, 3, 4
49	49	1	2	2	2	2	14	3	3	0, 1, 2, 3, 4
50	37	2	2	2	2	2	14	3	3	0, 1, 2, 3, 4
51	2	0	0	0	2	2	22	0	5	0, 1, 2, 3, 4
52	41	1	2	2	2	2	14	3	3	0, 1, 2, 3, 4
53	67	1	1	1	2	2	26	2	6	0
54	67	5	1	1	2	2	38	4	9	0, 1, 2, 3, 4
55	63	5	2	2	2	2	26	2	6	0
56	85	5	1	1	2	2	38	4	9	0, 1, 2, 3, 4
57	19	1	2	2	2	2	14	3	3	0, 1, 2, 3, 4
58	70	2	1	1	2	2	26	2	6	0
59	38	1	2	2	2	2	14	3	3	0, 1, 2, 3, 4

Table 5.7: Inconsistency between numerical CPLEX and true decision regions (continued)

Iter	Training Instance				Reported		CPLEX		True	
	ID	A	AGE	PEMLR	SSYN	Target	Region	Predict	Region	Predict
60	56	6	1	2	34	1	8	2	10	1
61	29	1	2	3	14	3	3	0, 1, 2, 3, 4	4	3
62	26	1	2	3	14	3	3	0, 1, 2, 3, 4	4	3
63	59	1	2	3	14	3	3	0, 1, 2, 3, 4	4	3
64	63	1	1	3	26	2	6	0	8	2
65	14	0	0	3	22	0	5	0, 1, 2, 3, 4	6	0
66	22	4	2	3	14	3	3	0, 1, 2, 3, 4	4	3
67	25	7	2	3	22	0	5	0, 1, 2, 3, 4	6	0
68	18	1	2	3	14	3	3	0, 1, 2, 3, 4	4	3
69	25	1	2	3	14	3	3	0, 1, 2, 3, 4	4	3
70	46	1	2	3	14	3	3	0, 1, 2, 3, 4	4	3
71	40	1	2	3	14	3	3	0, 1, 2, 3, 4	4	3
72	29	4	2	3	14	3	3	0, 1, 2, 3, 4	4	3
73	33	1	2	3	14	3	3	0, 1, 2, 3, 4	4	3
74	4	0	0	3	22	0	5	0, 1, 2, 3, 4	6	0
75	12	0	0	3	22	0	5	0, 1, 2, 3, 4	6	0
76	51	7	2	3	22	0	5	0, 1, 2, 3, 4	6	0

Table 5.7: Inconsistency between numerical CPLEX and true decision regions (continued)

Iter	Training Instance					Reported		CPLEX		True	
	ID	A	AGE	PEMLR	SSYN	Target	Region	Predict	Region	Predict	Region
	77	29		1	2	3	14	3	3	0, 1, 2, 3, 4	4
	78	7		0	0	3	22	0	5	0, 1, 2, 3, 4	6
	79	51		1	2	3	14	3	3	0, 1, 2, 3, 4	4
	80	41		1	2	3	14	3	3	0, 1, 2, 3, 4	4
	81	78		5	2	4	26	2	6	0	7
	82	60		2	2	4	14	3	3	0, 1, 2, 3, 4	4
	83	27		1	2	4	14	3	3	0, 1, 2, 3, 4	4
	84	65		1	2	4	14	3	3	0, 1, 2, 3, 4	4
	85	22		1	2	4	14	3	3	0, 1, 2, 3, 4	4
	86	42		1	2	4	14	3	3	0, 1, 2, 3, 4	4
	87	73		5	1	4	38	4	9	0, 1, 2, 3, 4	11
	88	45		1	2	4	14	3	3	0, 1, 2, 3, 4	4
	89	26		1	2	4	14	3	3	0, 1, 2, 3, 4	4
	90	76		5	1	4	38	4	9	0, 1, 2, 3, 4	11
	91	77		5	1	4	38	4	9	0, 1, 2, 3, 4	11
	92	27		1	2	4	14	3	3	0, 1, 2, 3, 4	4
	93	71		1	1	4	26	2	6	0	8

Table 5.7: Inconsistency between numerical CPLEX and true decision regions (continued)

Iter	Training Instance					Reported			CPLEX			True	
	ID	A	AGE	PEMLR	SSYN	Target	Region	Predict	Region	Predict	Region	Predict	Region
	28	10		0	0	1	34	1	8	2	10	1	1
	29	54		6	1	1	30	2	7	0, 1, 2, 3, 4	9	2	2
	30	3		0	0	1	34	1	8	2	10	1	1
	33	17		4	1	1	34	1	8	2	10	1	1
	35	77		6	1	1	30	2	7	0, 1, 2, 3, 4	9	2	2
	36	5		0	0	1	34	1	8	2	10	1	1
	37	80		5	1	1	38	4	9	2	11	4	4
	40	21		7	1	1	34	1	8	2	10	1	1
	44	79		1	1	2	30	2	7	0, 1, 2, 3, 4	9	2	2
	47	5		0	0	2	34	1	8	2	10	1	1
	48	76		5	1	2	38	4	9	2	11	4	4
	51	2		0	0	2	34	1	8	2	10	1	1
	53	67		1	1	2	30	2	7	0, 1, 2, 3, 4	9	2	2
	54	67		5	1	2	38	4	9	2	11	4	4
	56	85		5	1	2	38	4	9	2	11	4	4
	58	70		2	1	2	26	2	6	0, 1, 2, 3, 4	8	2	2
	60	56		6	1	2	30	2	7	0, 1, 2, 3, 4	9	2	2

Table 5.7: Inconsistency between numerical CPLEX and true decision regions (continued)

Iter	Training Instance					Reported		CPLEX		True	
	ID	A	AGE	PEMLR	SSYN	Target	Region	Predict	Region	Predict	Region
	64	63	1	1	1	3	30	2	7	0, 1, 2, 3, 4	9
	65	14	0	0	0	3	34	1	8	2	10
	74	4	0	0	0	3	34	1	8	2	10
	75	12	0	0	0	3	34	1	8	2	10
	78	7	0	0	0	3	34	1	8	2	10
	87	73	5	5	1	4	38	4	9	2	11
	90	76	5	5	1	4	38	4	9	2	11
	91	77	5	5	1	4	38	4	9	2	11
	93	71	1	1	1	4	30	2	7	0, 1, 2, 3, 4	9
	94	70	5	5	1	4	38	4	9	2	11
	95	78	5	5	1	4	38	4	9	2	11
	96	67	7	7	1	4	34	1	8	2	10
	97	71	5	5	1	4	38	4	9	2	11
	98	66	5	5	1	4	38	4	9	2	11
	99	67	5	5	1	4	38	4	9	2	11
9	8	4	0	0	0	0	34	1	8	2	10
											1

Table 5.7: Inconsistency between numerical CPLEX and true decision regions (continued)

Iter	Training Instance					Reported		CPLEX		True	
	ID	A	AGE	PEMLR	SSYN	Target	Region	Predict	Region	Predict	Region
	10	12	0	0	0	0	34	1	8	2	10
	20	10	0	0	0	0	34	1	8	2	10
	21	85	5	5	1	1	38	4	9	2	11
	22	74	5	5	1	1	38	4	9	2	11
	23	64	5	5	1	1	38	4	9	2	11
	24	73	5	5	1	1	38	4	9	2	11
	26	5	0	0	0	1	34	1	8	2	10
	27	4	0	0	0	1	34	1	8	2	10
	28	10	0	0	0	1	34	1	8	2	10
	29	54	6	6	1	1	34	1	8	2	10
	30	3	0	0	0	1	34	1	8	2	10
	33	17	4	4	1	1	34	1	8	2	10
	35	77	6	6	1	1	34	1	8	2	10
	36	5	0	0	0	1	34	1	8	2	10
	37	80	5	5	1	1	38	4	9	2	11
	40	21	7	7	1	1	34	1	8	2	10
	44	79	1	1	1	2	30	2	7	0, 1, 2, 3, 4	9

Table 5.7: Inconsistency between numerical CPLEX and true decision regions (continued)

Iter	Training Instance				Reported		CPLEX		True	
	ID	A_	AGE	PEMLR	SSYN	Target	Region	Predict	Region	Predict
	94	70		5	1	4	38	4	9	2
	95	78		5	1	4	38	4	9	2
	96	67		7	1	4	34	1	8	2
	97	71		5	1	4	38	4	9	2
	98	66		5	1	4	38	4	9	2
	99	67		5	1	4	38	4	9	2

CHAPTER VI

CONCLUDING REMARKS

Throughout this dissertation, the 2020 person-level CPS ASEC health insurance dataset in SAS7BDAT format is converted to feather and CSV formats. The file sizes markedly reduce by 94.02% and 71.31% respectively. Five combinations of health insurance enrollment in employment-based plan (GRP), direct-purchase plan (DIR) and public health insurance (PUB) are considered, leading to five possible classes. All codes are written in Python, well-known for data analysis, except the proposed box classifier in OPL embedded in CPLEX Optimization Studio. A Python class and a pandas DataFrame accessor are introduced so that a method can be called on a DataFrame at any time. All classification models, a Gini-based decision tree and the proposed classifier, are tested on a remote virtual machine to prevent the intervention in local computing resources and also to flexibly configure hardware and operating system. Python 3.13 with the global interpreter lock (GIL) still enabled is built from source. The GitHub repository is also available at <https://github.com/songkomkrit/phd>.

The proposed box classifier is heavily based on the rigorous formulation of 0-1 MILP problem, and it is very large-scale. Only 100 out of 157,681 noninfant survey participants are randomly selected as a sample of equal class size. Prior to the investigation of 2 contributing factors, 3 out of 184 independent variables are preselected by the SelectKBest using mutual information from a mixture of continuous and categorical features. Compared to the decision tree of multiple depths, the proposed model achieves a high training accuracy and low number of total splits within an hour and a half, though optimality not guaranteed, it constructs the branch-and-cut tree of large size between 6 GB and 7 GB, and it can group together similar categorical values to provide better insight into a selected categorical feature. A limitation of this study includes the lack of high-performance computing (HPC) technology of aggregating multiple computer clusters to efficiently serve massive computation required by the proposed model in the nature of 0-1 MILP. Therefore, further investigation into its approximation algorithm with theoretically derived bound on training accuracy compared to the exact 0-1 MILP model is suggested.

References

- Bernstein, D. J., Duif, N., Lange, T., Schwabe, P., and Yang, B.-Y. (2012). High-speed high-security signatures. Journal of cryptographic engineering, 2(2):77–89.
- Cebula, R. J. (2006). A further analysis of determinants of health insurance coverage. International Advances in Economic Research, 12(3):382–389.
- Cover, T. M. and Thomas, J. A. T. (2005). Elements of Information Theory. John Wiley Sons, Ltd.
- Dolinsky, A. and Caputo, R. K. (1997). Psychological and demographic characteristics as determinants of women’s health insurance coverage. Journal of Consumer Affairs, 31(2):218–237.
- Jin, Y., Hou, Z., and Zhang, D. (2016). Determinants of health insurance coverage among people aged 45 and over in china: Who buys public, private and multiple insurance. PLOS ONE, 11(8):1–15.
- Markowitz, M. A., Gold, M., and Rice, T. (1991). Determinants of health insurance status among young adults. Medical care, pages 6–19.
- Mulenga, J., Mulenga, M. C., Musonda, K., and Phiri, C. (2021). Examining gender differentials and determinants of private health insurance coverage in zambia. BMC Health Services Research, 21(1):1–11.
- Rivest, R. L., Shamir, A., and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2):120–126.
- Ross, B. C. (2014). Mutual information between discrete and continuous data sets. PLOS ONE, 9(2):1–5.
- Scikit-learn (2024a). Decision trees. <https://scikit-learn.org/1.5/modules/tree.html>. Accessed: 2024-11-18.
- Scikit-learn (2024b). Selectkbest. https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html. Accessed: 2024-11-18.

APPENDICES

APPENDIX A

CPLEX ENGINE LOG

<<< setup

Version identifier: 22.1.1.0 | 2022-11-28 | 9160aff4d

CPXPARAM_MIP_Strategy_File	3
CPXPARAM_MIP_Limits_Solutions	1
CPXPARAM_TimeLimit	86400
CPXPARAM_MIP_Limits_TreeMemory	204800

Tried aggregator 1 time.

MIP Presolve eliminated 402 rows and 800 columns.

MIP Presolve modified 200 coefficients.

Reduced MIP has 4004 rows, 5507 columns, and 22553 nonzeros.

Reduced MIP has 4643 binaries, 11 generals, 0 SOSs, and 0 indicators.

Presolve time = 0.01 sec. (17.75 ticks)

Found incumbent of value -20.000000 after 0.02 sec. (24.01 ticks)

Root node processing (before b&c):

Real time = 0.02 sec. (24.25 ticks)

Parallel b&c, 8 threads:

Real time = 0.00 sec. (0.00 ticks)

Sync time (average) = 0.00 sec.

Wait time (average) = 0.00 sec.

Total (root+branch&cut) = 0.02 sec. (24.25 ticks)

Iteration 1

Bounds on # of cuts = 8 with [3 3 2]

Error = 80 (out of 100 instances)

Accuracy = 20

Solving time = 0.0003894 min (minutes)

Accumulated time = 0.0003894 min (minutes)

Solution status code = 104

LB on error = -5500

Relative objective gap = 278.999999999

Selected variables:

Number of selected variables = 0 (0 continuous + 0 categorical)

Version identifier: 22.1.1.0 | 2022-11-28 | 9160aff4d

CPXPARAM_MIP_Strategy_File	3
CPXPARAM_MIP_Limits_Solutions	1
CPXPARAM_TimeLimit	86399.976635986328
CPXPARAM_MIP_Limits_TreeMemory	204800

Probing time = 0.01 sec. (4.62 ticks)

Cover probing fixed 8 vars, tightened 40 bounds.

Clique table members: 11812.

MIP emphasis: balance optimality and feasibility.

MIP search method: dynamic search.

Parallel mode: deterministic, using up to 8 threads.

Root relaxation solution time = 0.03 sec. (35.79 ticks)

Nodes				Cuts/				
Node	Left	Objective	IInf	Best Integer	Best Bound	ItCnt	Gap	
*	0+	0		-20.0000	-5600.0000		---	
0	0	-800.0000	472	-20.0000	-800.0000	1209	---	
0	0	-800.0000	346	-20.0000	Cuts: 512	1987	---	
0	0	-800.0000	651	-20.0000	Cuts: 874	3508	---	
*	0+	0		-28.0000	-800.0000		---	

GUB cover cuts applied: 29

Clique cuts applied: 10

Cover cuts applied: 51

Implied bound cuts applied: 242

Flow cuts applied: 6

Mixed integer rounding cuts applied: 186

Zero-half cuts applied: 77

Lift and project cuts applied: 7

Gomory fractional cuts applied: 16

Root node processing (before b&c):

Real time = 1.78 sec. (1803.05 ticks)

Parallel b&c, 8 threads:

Real time = 0.00 sec. (0.00 ticks)

Sync time (average) = 0.00 sec.

Wait time (average) = 0.00 sec.

Total (root+branch&cut) = 1.78 sec. (1803.05 ticks)

Iteration 2

Bounds on # of cuts = 8 with [3 3 2]

Error = 72 (out of 100 instances)

Accuracy = 28

Solving time = 0.029740967 min (minutes)

Accumulated time = 0.030130367 min (minutes)

Solution status code = 104

LB on error = -700

Relative objective gap = 27.571428571

Selected variables:

PEMLR (Categorical)

SS_YN (Categorical)

Number of selected variables = 2 (0 continuous + 2 categorical)

Version identifier: 22.1.1.0 | 2022-11-28 | 9160aff4d

CPXPARAM_MIP_Strategy_File 3

CPXPARAM_MIP_Limits_Solutions 1

CPXPARAM_TimeLimit 86398.192177978519

CPXPARAM_MIP_Limits_TreeMemory 204800

MIP emphasis: balance optimality and feasibility.

MIP search method: dynamic search.

Parallel mode: deterministic, using up to 8 threads.

Nodes				Cuts/			
Node	Left	Objective	IInf	Best Integer	Best Bound	ItCnt	Gap
*	0+	0		-31.0000	-717.7485		---

```

GUB cover cuts applied: 41
Clique cuts applied: 73
Cover cuts applied: 433
Implied bound cuts applied: 315
Flow cuts applied: 8
Mixed integer rounding cuts applied: 447
Zero-half cuts applied: 145
Lift and project cuts applied: 13
Gomory fractional cuts applied: 57

Root node processing (before b&c):
Real time          =    0.74 sec. (861.25 ticks)
Parallel b&c, 8 threads:
Real time          =    0.00 sec. (0.00 ticks)
Sync time (average) =    0.00 sec.
Wait time (average) =    0.00 sec.
-----
Total (root+branch&cut) =    0.74 sec. (861.25 ticks)

-----

Iteration 3
Bounds on # of cuts = 8 with [3 3 2]
Error = 69 (out of 100 instances)
Accuracy = 31
Solving time = 0.01229578 min (minutes)
Accumulated time = 0.042426147 min (minutes)

Solution status code = 104
LB on error = -617.482727096
Relative objective gap = 22.1446041

Selected variables:
PEMLR (Categorical)
SS_YN (Categorical)

Number of selected variables = 2 (0 continuous + 2 categorical)
-----
Version identifier: 22.1.1.0 | 2022-11-28 | 9160aff4d

```



```

CPXPARAM_MIP_Strategy_File          3
CPXPARAM_MIP_Limits_Solutions      1
CPXPARAM_TimeLimit                 86397.45443115235
CPXPARAM_MIP_Limits_TreeMemory     204800

```

MIP emphasis: balance optimality and feasibility.

MIP search method: dynamic search.

Parallel mode: deterministic, using up to 8 threads.

Nodes				Cuts/			
Node	Left	Objective	IInf	Best Integer	Best Bound	ItCnt	Gap
*	0+	0		-36.0000	-657.1275		---

GUB cover cuts applied: 41

Clique cuts applied: 73

Cover cuts applied: 623

Implied bound cuts applied: 329

Flow cuts applied: 12

Mixed integer rounding cuts applied: 562

Zero-half cuts applied: 191

Lift and project cuts applied: 22

Gomory fractional cuts applied: 108

Root node processing (before b&c):

Real time = 0.82 sec. (913.50 ticks)

Parallel b&c, 8 threads:

Real time = 0.00 sec. (0.00 ticks)

Sync time (average) = 0.00 sec.

Wait time (average) = 0.00 sec.

Total (root+branch&cut) = 0.82 sec. (913.50 ticks)

Iteration 4

Bounds on # of cuts = 8 with [3 3 2]

Error = 64 (out of 100 instances)

Accuracy = 36

Solving time = 0.013641048 min (minutes)

Accumulated time = 0.056067196 min (minutes)

Solution status code = 104
 LB on error = -557.127521455
 Relative objective gap = 17.253542263

Selected variables:
 PEMLR (Categorical)
 SS_YN (Categorical)

Number of selected variables = 2 (0 continuous + 2 categorical)

Version identifier: 22.1.1.0 | 2022-11-28 | 9160aff4d

CPXPARAM_MIP_Strategy_File	3
CPXPARAM_MIP_Limits_Solutions	1
CPXPARAM_TimeLimit	86396.635968261719
CPXPARAM_MIP_Limits_TreeMemory	204800

MIP emphasis: balance optimality and feasibility.

MIP search method: dynamic search.

Parallel mode: deterministic, using up to 8 threads.

Nodes				Cuts/			
Node	Left	Objective	IInf	Best Integer	Best Bound	ItCnt	Gap
*	0+	0		-38.0000	-626.9345		---

GUB cover cuts applied: 82

Clique cuts applied: 73

Cover cuts applied: 1063

Implied bound cuts applied: 407

Flow cuts applied: 35

Mixed integer rounding cuts applied: 819

Zero-half cuts applied: 258

Lift and project cuts applied: 22

Gomory fractional cuts applied: 160

Root node processing (before b&c):

Real time = 1.96 sec. (1928.89 ticks)

Parallel b&c, 8 threads:

Real time = 0.00 sec. (0.00 ticks)

Sync time (average) = 0.00 sec.

Wait time (average) = 0.00 sec.

Total (root+branch&cut) = 1.96 sec. (1928.89 ticks)

Iteration 5

Bounds on # of cuts = 8 with [3 3 2]

Error = 62 (out of 100 instances)

Accuracy = 38

Solving time = 0.032725952 min (minutes)

Accumulated time = 0.088793148 min (minutes)

Solution status code = 104

LB on error = -526.934511415

Relative objective gap = 15.498276616

Selected variables:

PEMLR (Categorical)

SS_YN (Categorical)

Number of selected variables = 2 (0 continuous + 2 categorical)

Version identifier: 22.1.1.0 | 2022-11-28 | 9160aff4d

CPXPARAM_MIP_Strategy_File 3

CPXPARAM_MIP_Limits_Solutions 1

CPXPARAM_TimeLimit 86394.672411132808

CPXPARAM_MIP_Limits_TreeMemory 204800

MIP emphasis: balance optimality and feasibility.

MIP search method: dynamic search.

Parallel mode: deterministic, using up to 8 threads.

Nodes					Cuts/		
Node	Left	Objective	IInf	Best Integer	Best Bound	ItCnt	Gap
0	0	-577.3658	659	-38.0000	Cuts: 836	28237	---
0	0	-558.5105	640	-38.0000	Cuts: 955	31741	---
0	0	-540.9147	613	-38.0000	Cuts: 870	34307	---
0	0	-539.0391	710	-38.0000	Cuts: 924	36234	---

```

0      0      -538.9354   762      -38.0000      Cuts: 989      37794      ---
Detecting symmetries...
0      0      -538.8822   778      -38.0000      Cuts: 830      39029      ---
0      0      -538.8578   826      -38.0000      Cuts: 708      40186      ---
0      0      -538.8409   806      -38.0000      Cuts: 266      40928      ---
0      0      -538.8265   840      -38.0000      Cuts: 601      41623      ---
0      2      -538.8265   827      -38.0000      -538.8265      41623      ---
Elapsed time = 5.26 sec. (5435.47 ticks, tree = 0.02 MB, solutions = 5)
2      4      -532.4711   622      -38.0000      -538.8264      44441      ---
9      9      -530.6872   643      -38.0000      -538.8264      47088      ---
27     20     -521.8493   667      -38.0000      -538.6068      60887      ---
46     20     -531.9657   614      -38.0000      -538.6066      60999      ---
80     68     -509.9472   575      -38.0000      -538.6066      103610     ---
118    57     -528.6696   612      -38.0000      -538.6066      98680      ---
156    138     -490.7266   504      -38.0000      -538.6066      147852     ---
194    169     -486.6126   511      -38.0000      -538.6066      164110     ---
248    209     -484.0715   570      -38.0000      -538.6066      181896     ---
625    468     -387.6828   467      -38.0000      -538.6066      243471     ---
Elapsed time = 8.32 sec. (8694.74 ticks, tree = 6.06 MB, solutions = 5)
1551   1044     infeasible      -38.0000      -538.6066      323452     ---

Performing restart 1

Repeating presolve.
Tried aggregator 1 time.
MIP Presolve eliminated 447 rows and 48 columns.
MIP Presolve modified 2098 coefficients.
Reduced MIP has 3557 rows, 5459 columns, and 21635 nonzeros.
Reduced MIP has 4603 binaries, 51 generals, 0 SOSs, and 0 indicators.
Presolve time = 0.01 sec. (20.08 ticks)
Tried aggregator 1 time.
MIP Presolve eliminated 1 rows and 0 columns.
MIP Presolve modified 300 coefficients.
Reduced MIP has 3556 rows, 5459 columns, and 21533 nonzeros.
Reduced MIP has 4603 binaries, 51 generals, 0 SOSs, and 0 indicators.
Presolve time = 0.02 sec. (21.21 ticks)
Represolve time = 0.18 sec. (172.19 ticks)
1603    0      -531.3154   530      -38.0000      Cuts: 989      388606      ---
1603    0      -507.2228   677      -38.0000      Cuts: 989      394828      ---

```

1603	0	-483.0125	703	-38.0000	Cuts: 989	399749	---
1603	0	-460.7636	713	-38.0000	Cuts: 989	407166	---
1603	0	-451.8578	687	-38.0000	Cuts: 989	412425	---
1603	0	-450.6323	805	-38.0000	Cuts: 989	415841	---
1603	0	-432.3823	759	-38.0000	Cuts: 989	423001	---
1603	0	-431.4684	871	-38.0000	Cuts: 989	426280	---
1603	0	-418.8128	830	-38.0000	Cuts: 989	433824	---
1603	0	-417.3207	854	-38.0000	Cuts: 989	437138	998.21%
1603	0	-412.4347	847	-38.0000	Cuts: 989	442602	985.35%
1603	0	-412.0400	919	-38.0000	Cuts: 989	445973	984.32%
1603	0	-411.2439	902	-38.0000	Cuts: 989	449769	980.32%
1603	0	-405.6804	852	-38.0000	Cuts: 989	458674	967.58%
1603	0	-405.2740	821	-38.0000	Cuts: 989	461351	962.76%
1603	0	-400.9631	855	-38.0000	Cuts: 989	468469	952.28%
1603	0	-400.5521	861	-38.0000	Cuts: 989	472372	952.28%
1603	0	-399.9329	893	-38.0000	Cuts: 989	475615	952.28%
1603	0	-397.2191	915	-38.0000	Cuts: 989	483998	944.52%
1603	0	-397.1061	974	-38.0000	Cuts: 989	487153	944.52%
1603	0	-396.3444	963	-38.0000	Cuts: 989	492117	943.01%
1603	0	-395.8637	958	-38.0000	Cuts: 989	496720	939.08%
1603	0	-395.7821	987	-38.0000	Cuts: 989	498869	938.39%
1603	0	-393.1402	932	-38.0000	Cuts: 989	506111	934.58%
1603	0	-393.0317	970	-38.0000	Cuts: 989	508897	934.29%
1603	0	-392.7950	1024	-38.0000	Cuts: 989	513782	933.67%
1603	0	-391.5060	909	-38.0000	Cuts: 989	518934	930.28%
1603	0	-391.4094	932	-38.0000	Cuts: 989	523923	930.02%
1603	0	-390.7816	965	-38.0000	Cuts: 989	530008	928.37%
1603	0	-390.4502	996	-38.0000	Cuts: 989	535960	927.50%
1603	0	-389.7746	975	-38.0000	Cuts: 964	544136	925.72%
1603	0	-389.7179	1028	-38.0000	Cuts: 989	548551	925.57%
1603	0	-389.2127	1004	-38.0000	Cuts: 779	559361	924.24%
1603	0	-389.1541	1044	-38.0000	Cuts: 989	563246	924.09%
1603	0	-388.9571	1041	-38.0000	Cuts: 550	570153	923.57%
1603	0	-388.9327	1102	-38.0000	Cuts: 989	573533	923.51%
1603	0	-388.7011	1102	-38.0000	Cuts: 689	580181	922.90%
1603	0	-388.6569	1153	-38.0000	Cuts: 989	583864	922.78%
1603	2	-388.6569	1138	-38.0000	-388.6569	583864	922.78%
1604	3	-388.2777	1073	-38.0000	-388.2776	587877	921.78%
1605	4	-387.6984	1112	-38.0000	-387.6983	589040	920.26%

1606	5	-387.2199	1098	-38.0000	-387.2194	590656	919.00%
1607	6	-386.8095	1049	-38.0000	-387.0084	594070	918.44%
1609	4	-386.1028	771	-38.0000	-387.0084	595848	918.44%
1610	5	-384.6422	738	-38.0000	-387.0084	598389	918.44%
1612	8	-382.0306	768	-38.0000	-387.0084	613444	918.44%
1615	9	-383.3599	777	-38.0000	-386.9557	622553	918.30%

Elapsed time = 129.55 sec. (136324.17 ticks, tree = 0.02 MB, solutions = 5)

1616	9	-375.8867	788	-38.0000	-386.9557	626524	918.30%
1618	12	-381.5367	781	-38.0000	-386.9557	649547	918.30%
1620	11	-384.0428	927	-38.0000	-386.9557	645526	918.30%
1621	7	-385.0541	787	-38.0000	-386.9557	604066	918.30%
1624	17	-380.8858	736	-38.0000	-386.8091	710376	917.92%
1626	18	-380.7050	773	-38.0000	-386.8091	720185	917.92%
1628	20	-383.5446	949	-38.0000	-386.8091	752988	917.92%
1629	23	-382.1894	814	-38.0000	-386.1685	802390	916.23%
1633	19	-379.8805	765	-38.0000	-386.1685	724806	916.23%
1636	21	-382.9042	965	-38.0000	-386.1685	754400	916.23%

Elapsed time = 144.26 sec. (150551.65 ticks, tree = 0.16 MB, solutions = 5)

1638	23	-380.8078	875	-38.0000	-386.1685	784761	916.23%
1640	30	-378.6604	789	-38.0000	-386.1685	871097	916.23%
1642	33	-382.5092	979	-38.0000	-386.1685	905127	916.23%
1644	28	-369.0237	733	-38.0000	-386.1685	859325	916.23%
1645	37	-371.9556	867	-38.0000	-386.1685	939036	916.23%
1648	39	-371.2651	710	-38.0000	-386.1685	956044	916.23%
1650	41	-372.1191	850	-38.0000	-386.1685	974080	916.23%
1653	42	-379.9721	743	-38.0000	-386.1685	985124	916.23%
1658	49	-377.9725	784	-38.0000	-386.1685	1012953	916.23%
1660	42	-368.8209	739	-38.0000	-386.1685	980397	916.23%

Elapsed time = 158.38 sec. (165820.30 ticks, tree = 0.22 MB, solutions = 5)

1662	46	-371.9569	788	-38.0000	-386.1685	996170	916.23%
1664	45	-378.6304	890	-38.0000	-386.1685	993788	916.23%
1666	48	-362.4336	921	-38.0000	-386.1685	1004351	916.23%
1669	57	-375.2631	783	-38.0000	-386.1685	1054343	916.23%
1672	65	-377.0938	785	-38.0000	-386.1685	1077462	916.23%
1676	56	-370.4028	811	-38.0000	-386.1685	1048798	916.23%
1677	58	-377.8983	718	-38.0000	-386.1685	1057061	916.23%
1680	69	-377.3027	879	-38.0000	-386.1685	1098444	916.23%
1682	73	-377.2401	751	-38.0000	-386.1685	1119275	916.23%
1687	64	-366.9964	711	-38.0000	-386.1685	1081207	916.23%

Elapsed time = 170.66 sec. (179644.29 ticks, tree = 0.33 MB, solutions = 5)

1689	80	-376.0566	805	-38.0000	-386.1685	1152637	916.23%
1692	81	-364.2601	795	-38.0000	-386.1685	1158452	916.23%
1698	86	-375.6997	713	-38.0000	-386.1685	1176524	916.23%
1702	78	-367.0278	782	-38.0000	-386.1685	1148330	916.23%
1705	87	-362.6076	808	-38.0000	-386.1685	1186831	916.23%
1709	87	-372.5778	688	-38.0000	-386.1685	1182617	916.23%
1715	91	-361.2418	775	-38.0000	-386.1685	1198439	916.23%
1718	96	-364.3288	787	-38.0000	-386.1685	1229751	916.23%
1722	97	-361.7048	671	-38.0000	-386.1685	1223041	916.23%
1731	101	-371.0484	819	-38.0000	-386.1685	1241877	916.23%

Elapsed time = 181.55 sec. (190828.34 ticks, tree = 0.48 MB, solutions = 5)

1738	101	-352.9145	701	-38.0000	-386.1685	1224916	916.23%
1747	105	-348.2397	651	-38.0000	-386.1685	1226350	916.23%
1751	92	-355.5354	732	-38.0000	-386.1685	1201408	916.23%
1753	98	-363.3957	800	-38.0000	-386.1685	1236017	916.23%
1760	109	-360.8998	699	-38.0000	-386.1685	1258257	916.23%
1766	106	-362.0373	768	-38.0000	-386.1685	1251129	916.23%
1770	138	-369.8963	847	-38.0000	-386.1685	1315878	916.23%
1776	157	-359.2809	751	-38.0000	-386.1685	1371681	916.23%
1780	143	-372.8468	866	-38.0000	-386.1685	1336188	916.23%
1788	159	-357.3907	752	-38.0000	-386.1685	1376458	916.23%

Elapsed time = 192.07 sec. (201530.64 ticks, tree = 1.48 MB, solutions = 5)

1793	165	-351.1548	720	-38.0000	-386.1685	1382812	916.23%
1800	146	-330.0804	647	-38.0000	-386.1685	1313355	916.23%
1809	168	-354.1876	662	-38.0000	-386.1685	1388199	916.23%
1819	169	-347.8706	660	-38.0000	-386.1685	1390338	916.23%
1827	171	-347.0562	700	-38.0000	-386.1685	1392341	916.23%
1838	198	-359.3410	735	-38.0000	-386.1685	1468649	916.23%
1844	189	-316.1421	609	-38.0000	-386.1685	1413172	916.23%
1856	184	-366.0754	822	-38.0000	-386.1685	1431628	916.23%
1862	177	-342.0989	643	-38.0000	-386.1685	1401987	916.23%
1872	185	-368.7856	775	-38.0000	-386.1685	1433055	916.23%

Elapsed time = 202.84 sec. (212543.16 ticks, tree = 2.11 MB, solutions = 5)

1886	204	-348.5624	768	-38.0000	-386.1685	1470065	916.23%
1896	187	-367.8768	775	-38.0000	-386.1685	1439100	916.23%
1910	263	-366.6514	725	-38.0000	-386.1685	1563807	916.23%
1917	226	-366.2143	745	-38.0000	-386.1685	1526100	916.23%
1936	223	-329.7481	750	-38.0000	-386.1685	1508197	916.23%

1943	280	-352.0908	798	-38.0000	-386.1685	1611855	916.23%
1954	306	-346.5994	704	-38.0000	-386.1685	1668764	916.23%
1963	266	-359.3957	727	-38.0000	-386.1685	1578568	916.23%
1976	227	-330.0316	709	-38.0000	-386.1685	1517288	916.23%
1996	304	-332.9077	756	-38.0000	-386.1685	1652826	916.23%

Elapsed time = 212.95 sec. (223101.71 ticks, tree = 5.77 MB, solutions = 5)

2005	237	-359.0799	637	-38.0000	-386.1685	1547380	916.23%
2023	289	-351.0669	792	-38.0000	-386.1685	1631819	916.23%
2045	312	-332.4457	739	-38.0000	-386.1685	1662091	916.23%
2068	366	-350.4486	785	-38.0000	-386.1685	1774184	916.23%
2081	393	-327.5920	631	-38.0000	-386.1685	1810141	916.23%
2099	326	-322.0228	695	-38.0000	-386.1685	1696440	916.23%
2119	349	-325.3107	627	-38.0000	-386.1685	1722349	916.23%
2140	448	-321.3074	722	-38.0000	-386.1685	1913614	916.23%
2160	460	-315.9675	684	-38.0000	-386.1685	1927645	916.23%
2227	375	-329.5555	813	-38.0000	-386.1685	1801495	916.23%

Elapsed time = 225.67 sec. (235995.28 ticks, tree = 6.47 MB, solutions = 5)

2329	554	-274.9106	575	-38.0000	-386.1685	2020145	916.23%
2462	603	-208.4551	608	-38.0000	-386.1685	2106858	916.23%
2643	662	-287.5155	621	-38.0000	-386.1685	2198449	916.23%
2816	632	-274.9940	683	-38.0000	-386.1685	2159172	916.23%
2986	735	-213.5904	523	-38.0000	-386.1685	2277454	916.23%
3306	787	-211.7584	632	-38.0000	-385.3111	2315535	913.98%
3607	1286	-201.8962	558	-38.0000	-385.3111	2674488	913.98%
3977	1303	-183.7525	692	-38.0000	-385.3111	2693379	913.98%
4008	1540	-376.5161	957	-38.0000	-385.3111	2835562	913.98%
4055	1700	-376.8232	922	-38.0000	-385.3111	2930975	913.98%

Elapsed time = 265.35 sec. (274668.79 ticks, tree = 65.53 MB, solutions = 5)

4113	1703	-375.8357	891	-38.0000	-385.3111	2941519	913.98%
4283	2263	-129.2319	583	-38.0000	-384.4635	3322625	911.75%
4472	2267	-374.2307	1055	-38.0000	-384.4635	3388151	911.75%
4510	2280	-365.4293	795	-38.0000	-384.4635	3426661	911.75%
4538	2416	-346.9335	718	-38.0000	-381.9426	3507655	905.11%
4576	2480	-361.8407	815	-38.0000	-381.9426	3618609	905.11%
4615	2528	-373.4181	888	-38.0000	-381.9426	3742100	905.11%
4658	2532	-342.0634	836	-38.0000	-381.9426	3734502	905.11%
4699	2533	-365.4533	944	-38.0000	-381.9426	3763000	905.11%
4747	2657	-310.5418	677	-38.0000	-381.9426	4014791	905.11%

Elapsed time = 303.11 sec. (313289.88 ticks, tree = 111.76 MB, solutions = 5)

4802	2620	-349.3655	890	-38.0000	-381.9426	3957330	905.11%
4871	2755	-323.3668	697	-38.0000	-381.9426	4199276	905.11%
4946	2741	-290.9565	601	-38.0000	-381.9426	4189091	905.11%
5043	2816	-273.6839	761	-38.0000	-381.9426	4291508	905.11%
5155	2962	-201.2710	658	-38.0000	-381.9426	4460142	905.11%
5291	2981	-169.8593	604	-38.0000	-381.9426	4478921	905.11%
5466	3076	-203.9541	682	-38.0000	-381.9426	4584024	905.11%
5694	3180	-135.7850	678	-38.0000	-381.9426	4698677	905.11%
6097	3555	-75.2412	434	-38.0000	-381.9426	4847836	905.11%
6335	3538	-100.6562	464	-38.0000	-381.9426	4949312	905.11%
Elapsed time = 342.63 sec. (351762.11 ticks, tree = 158.31 MB, solutions = 5)							
6614	4051	-82.9797	391	-38.0000	-381.9426	5198382	905.11%
7157	4043	-93.9551	441	-38.0000	-381.9426	5261948	905.11%
7752	4029	-193.8106	526	-38.0000	-381.9426	5254080	905.11%
7876	4590	-83.9348	406	-38.0000	-381.8931	5514496	904.98%
7902	4881	-379.3565	919	-38.0000	-381.8926	5595047	904.98%
7940	5145	-286.1287	658	-38.0000	-380.8071	5682204	902.12%
8002	4691	-379.3689	774	-38.0000	-380.6354	5544630	901.67%
8035	5148	-364.5840	753	-38.0000	-380.6354	5716992	901.67%
8098	5346	-324.6925	717	-38.0000	-379.9667	5809066	899.91%
8209	5380	-263.0652	689	-38.0000	-379.9667	5827011	899.91%
Elapsed time = 383.55 sec. (391445.00 ticks, tree = 250.41 MB, solutions = 5)							
8407	5393	-359.8021	721	-38.0000	-379.9667	5914698	899.91%
8481	5521	-262.1683	689	-38.0000	-379.9667	6008749	899.91%
8682	5483	-357.5335	722	-38.0000	-379.9667	6039212	899.91%
8840	5744	-352.5118	627	-38.0000	-379.9667	6188503	899.91%
9256	5975	-93.5178	383	-38.0000	-379.9667	6283362	899.91%
9630	6102	-222.7763	518	-38.0000	-379.9667	6388913	899.91%
9957	6395	-332.9427	599	-38.0000	-379.9667	6566131	899.91%
10206	6704	-102.7602	493	-38.0000	-379.9667	6620570	899.91%
10687	6744	-356.8449	804	-38.0000	-379.9667	6676558	899.91%
10892	7279	-141.4255	485	-38.0000	-379.9667	6824257	899.91%
Elapsed time = 424.74 sec. (430070.66 ticks, tree = 348.74 MB, solutions = 5)							
11285	7549	-266.8955	713	-38.0000	-379.9667	6935942	899.91%
11952	8078	-81.0221	475	-38.0000	-379.9667	7048892	899.91%
12136	8219	-376.5899	831	-38.0000	-379.7943	7146826	899.46%
12316	8696	-376.1854	831	-38.0000	-379.5824	7253016	898.90%
12762	9331	-109.6829	395	-38.0000	-379.5824	7366582	898.90%
13127	9413	-307.3537	678	-38.0000	-379.4554	7421367	898.57%

13190	9725	-370.0417	752	-38.0000	-379.4554	7491216	898.57%
13369	10087	-365.0055	759	-38.0000	-379.4554	7647384	898.57%
13522	9992	-149.8716	574	-38.0000	-379.3906	7584555	898.40%
13675	10455	-169.6634	556	-38.0000	-379.3906	7707912	898.40%
Elapsed time = 472.46 sec. (468453.20 ticks, tree = 464.06 MB, solutions = 5)							
13959	10554	-275.5156	638	-38.0000	-379.3906	7826355	898.40%
14081	10676	-330.6031	587	-38.0000	-379.3841	7853249	898.38%
14380	10903	-299.8063	554	-38.0000	-379.2996	7908540	898.16%
14811	10991	-84.2419	244	-38.0000	-379.2886	7914970	898.13%
15473	11856	-43.7849	209	-38.0000	-379.2886	8097559	898.13%
15621	11659	-375.0829	765	-38.0000	-379.2886	8079509	898.13%
15745	12045	-279.4488	234	-38.0000	-379.2886	8159239	898.13%
16259	12480	-122.9856	334	-38.0000	-379.2886	8247673	898.13%
16560	12619	-150.5545	539	-38.0000	-379.2386	8302917	898.00%
16678	12987	-260.3273	396	-38.0000	-378.8563	8406230	896.99%
Elapsed time = 525.90 sec. (506688.39 ticks, tree = 537.86 MB, solutions = 5)							
16832	13408	-360.3564	681	-38.0000	-378.8563	8512516	896.99%
17110	13421	-347.1104	577	-38.0000	-378.7315	8526769	896.66%
17190	13641	-337.1913	715	-38.0000	-378.5983	8577198	896.31%
17403	13718	-266.2754	489	-38.0000	-378.5983	8642161	896.31%
17723	13869	-246.2897	615	-38.0000	-378.5983	8701973	896.31%
17846	14453	-147.7591	476	-38.0000	-378.5983	8901628	896.31%
18013	14743	-257.4287	619	-38.0000	-378.5983	9008331	896.31%
18451	14774	-193.0102	557	-38.0000	-378.5983	9013834	896.31%
18659	14808	-112.1777	501	-38.0000	-378.5983	9017455	896.31%
18954	15194	-365.5685	865	-38.0000	-378.5983	9123572	896.31%
Elapsed time = 577.52 sec. (546429.72 ticks, tree = 545.80 MB, solutions = 5)							
18993	14989	-304.6462	216	-38.0000	-378.5190	9079117	896.10%
19220	15840	-359.2220	537	-38.0000	-378.5190	9298493	896.10%
19362	15500	-367.9160	862	-38.0000	-378.3778	9199784	895.73%
19647	16099	-337.1348	625	-38.0000	-378.2779	9416366	895.47%
19967	16207	-348.9415	288	-38.0000	-378.2779	9475112	895.47%
20375	16345	-375.7467	838	-38.0000	-378.2215	9569876	895.32%
20568	16421	-210.0809	171	-38.0000	-378.2029	9586501	895.27%
20898	16905	-48.9183	177	-38.0000	-378.1858	9664318	895.23%
21209	17362	-43.6742	267	-38.0000	-378.1858	9772573	895.23%
21460	17380	-195.1753	181	-38.0000	-378.1382	9776799	895.10%
Elapsed time = 628.76 sec. (585005.60 ticks, tree = 564.44 MB, solutions = 5)							
21731	17569	-176.2266	368	-38.0000	-378.1382	9846289	895.10%

22006	18252	-234.4369	589	-38.0000	-378.1353	10008342	895.09%
22183	18306	-306.6087	349	-38.0000	-378.1353	9991426	895.09%
22423	18469	-121.7009	505	-38.0000	-378.1353	10072247	895.09%
22692	18987	-121.6388	336	-38.0000	-378.1353	10213880	895.09%
22850	19137	-56.8695	394	-38.0000	-378.1353	10254885	895.09%
22918	19013	-364.3899	709	-38.0000	-378.0981	10236729	894.99%
23147	19464	-325.3539	713	-38.0000	-377.9287	10374695	894.55%
23527	19550	-169.3183	533	-38.0000	-377.9287	10393813	894.55%
24049	19625	-364.2002	903	-38.0000	-377.8836	10371003	894.43%
Elapsed time = 682.61 sec. (623723.92 ticks, tree = 682.22 MB, solutions = 5)							
24686	20421	-368.3340	750	-38.0000	-377.8294	10657403	894.29%
25245	19621	-341.1563	713	-38.0000	-377.8294	10466998	894.29%
25810	20807	-353.1728	676	-38.0000	-377.8294	10767293	894.29%
26049	21383	-358.5244	487	-38.0000	-377.8294	10845444	894.29%
26370	21135	-277.1734	655	-38.0000	-377.7041	10818422	893.96%
26824	21172	-182.8045	538	-38.0000	-377.6195	10821038	893.74%
27218	22670	-296.3888	360	-38.0000	-377.6195	11004288	893.74%
27628	22783	-189.3246	127	-38.0000	-377.6147	11054059	893.72%
28136	22825	-270.7104	612	-38.0000	-377.6147	11112939	893.72%
28294	24138	-209.7610	529	-38.0000	-377.6147	11307267	893.72%
Elapsed time = 734.91 sec. (662090.80 ticks, tree = 797.77 MB, solutions = 5)							
28605	23711	-234.1514	552	-38.0000	-377.6147	11253825	893.72%
28840	24553	-268.2504	475	-38.0000	-377.5816	11391896	893.64%
29426	24982	-166.5687	513	-38.0000	-377.5816	11485504	893.64%
29687	25483	-371.1550	894	-38.0000	-377.5816	11577943	893.64%
30202	25692	-274.5559	499	-38.0000	-377.4552	11622202	893.30%
30909	25657	-63.7559	371	-38.0000	-377.4257	11604346	893.23%
31597	25853	-118.5099	565	-38.0000	-377.4257	11717188	893.23%
32092	26336	-181.8973	511	-38.0000	-377.4257	11767598	893.23%
33050	26745	-46.3389	148	-38.0000	-377.4257	11832881	893.23%
33558	27309	-53.9421	87	-38.0000	-377.3971	11887058	893.15%
Elapsed time = 781.18 sec. (700363.36 ticks, tree = 1010.72 MB, solutions = 5)							
33666	27434	-282.0341	190	-38.0000	-377.2214	11958972	892.69%
* 33853+29275				-39.0000	-377.1435		867.03%
33922	29276	-367.3141	816	-39.0000	-377.1435	12240781	867.03%
33978	29609	-373.9386	762	-39.0000	-377.1435	12286072	867.03%
34107	29321	-272.3192	625	-39.0000	-377.1435	12257306	867.03%

GUB cover cuts applied: 745

```

Clique cuts applied: 45
Cover cuts applied: 3303
Implied bound cuts applied: 47
Flow cuts applied: 81
Mixed integer rounding cuts applied: 882
Zero-half cuts applied: 110
Lift and project cuts applied: 6
Gomory fractional cuts applied: 196

Root node processing (before b&c):
Real time          = 5.07 sec. (5253.09 ticks)
Parallel b&c, 8 threads:
Real time          = 792.79 sec. (713089.45 ticks)
Sync time (average) = 91.30 sec.
Wait time (average) = 0.07 sec.
-----
Total (root+branch&cut) = 797.86 sec. (718342.54 ticks)

-----

Iteration 6
Bounds on # of cuts = 8 with [3 3 2]
Error = 61 (out of 100 instances)
Accuracy = 39
Solving time = 13.297700484 min (minutes)
Accumulated time = 13.386493632 min (minutes)

Solution status code = 104
LB on error = -277.143152611
Relative objective gap = 8.670337246

Selected variables:
PEMLR (Categorical)
SS_YN (Categorical)

Number of selected variables = 2 (0 continuous + 2 categorical)
-----
Version identifier: 22.1.1.0 | 2022-11-28 | 9160aff4d
CPXPARAM_MIP_Strategy_File          3
CPXPARAM_MIP_Limits_Solutions      1

```

CPXPARAM_TimeLimit	85596.810382080075
CPXPARAM_MIP_Limits_TreeMemory	204800

Nodes		Cuts/						
Node	Left	Objective	IInf	Best Integer	Best Bound	ItCnt	Gap	
34184	30011	infeasible		-39.0000	-377.1432	12462046	867.03%	
Elapsed time = 0.56 sec. (7.69 ticks, tree = 1131.55 MB, solutions = 6)								
34185	30011	infeasible		-39.0000	-377.1432	12463576	867.03%	
34186	30012	-353.1274	307	-39.0000	-377.1432	12467117	867.03%	
34230	30050	-240.2890	140	-39.0000	-377.1432	12469248	867.03%	
34284	30101	-96.2412	68	-39.0000	-377.1432	12469654	867.03%	
34322	30026	-310.4203	254	-39.0000	-377.1432	12474494	867.03%	
34358	30062	-208.3211	168	-39.0000	-377.1432	12474783	867.03%	
34418	30117	-61.5243	49	-39.0000	-377.1432	12474986	867.03%	
34429	30013	-375.1626	767	-39.0000	-377.1432	12465551	867.03%	
34430	30013	-368.5689	712	-39.0000	-377.1432	12480749	867.03%	
34467	30044	-274.8883	182	-39.0000	-377.1369	12485663	867.02%	
Elapsed time = 4.67 sec. (3790.87 ticks, tree = 1118.48 MB, solutions = 6)								
34559	30016	-375.0799	850	-39.0000	-377.1369	12488423	867.02%	
34566	30025	-347.9018	626	-39.0000	-376.9781	12489041	866.61%	
34577	30032	-332.5967	607	-39.0000	-376.9781	12490820	866.61%	
34601	30027	-337.4018	605	-39.0000	-376.9781	12498141	866.61%	
34627	30041	-305.9294	499	-39.0000	-376.9781	12501938	866.61%	
34686	30147	-296.6126	233	-39.0000	-376.9781	12501312	866.61%	
34810	30028	-351.9984	446	-39.0000	-376.9781	12506723	866.61%	
34871	30070	-210.9115	477	-39.0000	-376.9781	12496496	866.61%	
34894	30030	-347.0129	721	-39.0000	-376.9781	12516063	866.61%	
34921	30128	-340.5311	403	-39.0000	-376.9781	12501487	866.61%	
Elapsed time = 18.01 sec. (13637.71 ticks, tree = 1129.36 MB, solutions = 6)								
35000	30164	-248.6996	190	-39.0000	-376.9781	12503763	866.61%	
35127	30084	-205.3721	505	-39.0000	-376.9781	12522308	866.61%	
35293	30114	-133.2772	471	-39.0000	-376.9781	12523058	866.61%	
35359	30236	-342.9975	706	-39.0000	-376.9781	12518300	866.61%	
35553	30295	-154.2894	114	-39.0000	-376.9781	12510201	866.61%	
35761	30483	-66.1665	53	-39.0000	-376.9781	12512438	866.61%	
35798	30258	-297.3622	651	-39.0000	-376.9781	12521841	866.61%	
35816	30266	-281.3269	625	-39.0000	-376.9781	12523129	866.61%	
35843	30276	-256.0856	595	-39.0000	-376.9744	12523344	866.60%	

35885	30140	-302.2962	240	-39.0000	-376.9744	12517050	866.60%
Elapsed time = 30.67 sec. (23289.65 ticks, tree = 1133.67 MB, solutions = 6)							
36002	30180	-252.7491	441	-39.0000	-376.9744	12529649	866.60%
36062	30299	-206.5953	589	-39.0000	-376.9744	12527129	866.60%
36101	30230	-57.4126	281	-39.0000	-376.9744	12532623	866.60%
36126	30319	-159.6736	546	-39.0000	-376.9744	12528123	866.60%
36145	30340	-344.3312	457	-39.0000	-376.9744	12533604	866.60%
36233	30409	-163.6412	356	-39.0000	-376.9744	12535025	866.60%
36303	30347	-91.1935	479	-39.0000	-376.9744	12529582	866.60%
36329	30235	-375.3955	857	-39.0000	-376.9744	12539899	866.60%
36545	30368	cutoff		-39.0000	-376.9744	12531654	866.60%
36575	30265	-348.5818	255	-39.0000	-376.9744	12541452	866.60%
Elapsed time = 42.20 sec. (33038.41 ticks, tree = 1127.58 MB, solutions = 6)							
36709	30470	-351.3221	624	-39.0000	-376.9744	12544142	866.60%
36729	30559	-215.9049	649	-39.0000	-376.9744	12544298	866.60%
36812	30436	-161.8911	120	-39.0000	-376.9744	12548856	866.60%
36944	30486	-328.9821	597	-39.0000	-376.9744	12550670	866.60%
37174	30492	-322.6494	608	-39.0000	-376.9744	12552363	866.60%
37271	30718	-149.7022	112	-39.0000	-376.9744	12556065	866.60%
37335	30604	-117.6572	523	-39.0000	-376.9744	12548367	866.60%
37361	30612	-97.8061	497	-39.0000	-376.9744	12549204	866.60%
37508	30622	-74.8865	483	-39.0000	-376.9744	12549545	866.60%
37547	30284	-269.1983	676	-39.0000	-376.9744	12570795	866.60%
Elapsed time = 56.00 sec. (42712.64 ticks, tree = 1132.84 MB, solutions = 6)							
37587	30639	-46.6761	404	-39.0000	-376.9744	12551100	866.60%
37639	30414	-311.7083	637	-39.0000	-376.9744	12561840	866.60%
37916	30226	-75.1134	38	-39.0000	-376.9744	12586701	866.60%
37975	30522	-298.5992	190	-39.0000	-376.9744	12567952	866.60%
38358	30734	-49.6809	34	-39.0000	-376.9744	12573259	866.60%
38425	30896	-312.8846	395	-39.0000	-376.9744	12576993	866.60%
38560	30651	-351.1738	707	-39.0000	-376.9744	12567726	866.60%
38703	30659	-338.3736	682	-39.0000	-376.9744	12569044	866.60%
38722	30923	-251.4943	422	-39.0000	-376.9744	12578618	866.60%
38807	30678	-300.9916	641	-39.0000	-376.9744	12570330	866.60%
Elapsed time = 69.11 sec. (52474.66 ticks, tree = 1168.75 MB, solutions = 6)							
38865	30114	-364.9152	785	-39.0000	-376.9744	12605868	866.60%
39094	30118	-364.5336	776	-39.0000	-376.9744	12608499	866.60%
39163	30390	-290.9313	188	-39.0000	-376.9744	12593666	866.60%
39318	30330	-128.7170	102	-39.0000	-376.9744	12608320	866.60%

39378	30824	-371.6508	666	-39.0000	-376.9744	12583385	866.60%
39448	30859	-235.6664	169	-39.0000	-376.9744	12587080	866.60%
39572	30207	-115.2196	106	-39.0000	-376.9744	12620071	866.60%
39664	30963	-184.5348	344	-39.0000	-376.9744	12600785	866.60%
39767	30781	-243.6394	154	-39.0000	-376.9744	12609310	866.60%
39849	30937	-367.9700	804	-39.0000	-376.9744	12597338	866.60%

Elapsed time = 80.41 sec. (62223.15 ticks, tree = 1187.62 MB, solutions = 6)

39854	30851	-372.7405	686	-39.0000	-376.9744	12613720	866.60%
39993	30935	-102.8985	77	-39.0000	-376.9744	12616877	866.60%
40140	30982	-256.2504	319	-39.0000	-376.9744	12605444	866.60%
40214	31050	-74.6662	160	-39.0000	-376.9744	12606445	866.60%
40237	30486	-373.3192	818	-39.0000	-376.9744	12620511	866.60%
40365	30487	-371.8640	807	-39.0000	-376.9744	12623950	866.60%
40369	31131	-374.6936	763	-39.0000	-376.9744	12621218	866.60%
40456	31135	-198.9005	131	-39.0000	-376.9744	12617239	866.60%
40555	30500	-355.6002	607	-39.0000	-376.9744	12631140	866.60%
40570	30508	-331.8984	543	-39.0000	-376.9744	12632773	866.60%

Elapsed time = 92.34 sec. (72321.10 ticks, tree = 1149.15 MB, solutions = 6)

40596	30518	-328.9640	539	-39.0000	-376.9744	12633058	866.60%
40632	30271	-259.0082	190	-39.0000	-376.9744	12650779	866.60%
40800	31223	-90.5794	81	-39.0000	-376.9744	12635395	866.60%
41073	31344	-64.5276	40	-39.0000	-376.9744	12637685	866.60%
41160	30618	-133.3406	487	-39.0000	-376.9744	12643767	866.60%
41210	31110	-356.1415	734	-39.0000	-376.9744	12623652	866.60%
41230	30356	-355.0434	236	-39.0000	-376.9744	12664838	866.60%
41364	31124	-323.7738	674	-39.0000	-376.9744	12626631	866.60%
41379	31356	-369.4440	735	-39.0000	-376.9744	12647780	866.60%
41481	30734	-138.1924	86	-39.0000	-376.9744	12656975	866.60%

Elapsed time = 104.01 sec. (81980.95 ticks, tree = 1160.34 MB, solutions = 6)

41544	30398	-298.4723	615	-39.0000	-376.9744	12669908	866.60%
41678	31417	-241.3441	167	-39.0000	-376.9744	12654919	866.60%
41866	31505	-61.6078	64	-39.0000	-376.9744	12655794	866.60%
41914	31163	-244.9421	559	-39.0000	-376.9744	12639460	866.60%
42050	31172	-220.4289	522	-39.0000	-376.9744	12640470	866.60%
42082	30440	-203.7039	545	-39.0000	-376.9744	12674888	866.60%
42117	30505	-307.1978	643	-39.0000	-376.9744	12694878	866.60%
42157	30458	-162.6077	487	-39.0000	-376.9744	12676762	866.60%
42257	31346	-321.7987	248	-39.0000	-376.9744	12666744	866.60%
42771	31228	-124.3631	93	-39.0000	-376.9744	12686599	866.60%

Elapsed time = 119.11 sec. (94489.24 ticks, tree = 1192.42 MB, solutions = 6)

43224	31270	-374.6367	958	-39.0000	-376.9744	12694136	866.60%
43751	33045	-161.2916	114	-39.0000	-376.9744	12987648	866.60%
44530	30774	-150.7804	111	-39.0000	-376.9744	12718084	866.60%
44812	30814	-374.2969	1033	-39.0000	-376.9744	12725929	866.60%
45132	30671	cutoff		-39.0000	-376.9744	12724888	866.60%
45505	31494	-360.9840	699	-39.0000	-376.9744	12709907	866.60%
45992	31897	-85.7831	53	-39.0000	-376.9744	12902866	866.60%
46284	35065	-111.8791	76	-39.0000	-376.9744	13253046	866.60%
46578	31053	-356.1008	650	-39.0000	-376.9744	12774822	866.60%
46906	31958	-274.4820	382	-39.0000	-376.9744	12935080	866.60%

Elapsed time = 164.80 sec. (133139.60 ticks, tree = 1235.78 MB, solutions = 6)

47493	31337	-291.1340	219	-39.0000	-376.9744	12792486	866.60%
48138	31430	-374.7112	1029	-39.0000	-376.9744	12799082	866.60%
48546	32033	-116.6233	179	-39.0000	-376.9194	12782627	866.46%
49011	32202	-357.4841	370	-39.0000	-376.9194	12958830	866.46%
50019	32432	-103.9418	68	-39.0000	-376.9194	12967930	866.46%
50531	31080	-370.9851	885	-39.0000	-376.9194	12828929	866.46%
* 50701+31549				-40.0000	-376.9194		842.30%

51048	32626	-160.4344	119	-40.0000	-376.9194	12988233	842.30%
51323	33736	-344.6369	244	-40.0000	-376.9194	13106949	842.30%
51999	37235	-345.6706	331	-40.0000	-376.9194	13626666	842.30%
52188	31438	-349.4261	701	-40.0000	-376.9194	12876494	842.30%

Elapsed time = 209.24 sec. (171662.93 ticks, tree = 1199.03 MB, solutions = 7)

52481	37432	-201.6757	314	-40.0000	-376.9194	13648219	842.30%
53422	35758	-365.5130	684	-40.0000	-376.9194	13371629	842.30%
53912	37682	-130.6639	94	-40.0000	-376.9194	13667768	842.30%
54122	34391	-222.5263	261	-40.0000	-376.9194	13156341	842.30%
54537	34575	-118.8445	76	-40.0000	-376.9194	13161634	842.30%
54944	32867	-274.5121	285	-40.0000	-376.9194	13066188	842.30%
55210	36351	-126.9557	87	-40.0000	-376.9194	13427861	842.30%
55473	36429	-261.6947	186	-40.0000	-376.9194	13441222	842.30%
55684	34630	-361.6861	651	-40.0000	-376.9194	13188259	842.30%
56056	31799	-294.4002	652	-40.0000	-376.9194	12963851	842.30%

Elapsed time = 253.04 sec. (209922.64 ticks, tree = 1190.36 MB, solutions = 7)

56741	31813	-374.9630	885	-40.0000	-376.9194	12984867	842.30%
57071	36555	-349.2594	626	-40.0000	-376.9194	13480838	842.30%
57695	31966	-342.6780	723	-40.0000	-376.9194	13006011	842.30%
58149	32052	-133.2492	96	-40.0000	-376.9194	13020291	842.30%

58577	36670	-319.4597	209	-40.0000	-376.9194	13512903	842.30%
59334	38194	infeasible		-40.0000	-376.9194	13776794	842.30%
59411	36778	-351.9628	635	-40.0000	-376.9194	13531178	842.30%
59775	33722	-371.4617	918	-40.0000	-376.9194	13181426	842.30%
59948	36900	-366.2275	661	-40.0000	-376.9194	13549234	842.30%
60447	32460	-335.7909	735	-40.0000	-376.9194	13080118	842.30%
Elapsed time = 296.23 sec. (248581.39 ticks, tree = 1274.28 MB, solutions = 7)							
60791	37101	-185.5764	181	-40.0000	-376.9194	13570876	842.30%
61392	34217	-374.0933	734	-40.0000	-376.9194	13219253	842.30%
62039	36439	-363.2180	698	-40.0000	-376.9194	13336882	842.30%
62196	36566	-46.5830	229	-40.0000	-376.9194	13345567	842.30%
62482	34472	-340.0144	758	-40.0000	-376.9194	13248238	842.30%

Began writing nodes to disk (directory ./cpxhGkJOU created)

GUB cover cuts applied: 872

Clique cuts applied: 53

Cover cuts applied: 3794

Implied bound cuts applied: 59

Flow cuts applied: 95

Mixed integer rounding cuts applied: 1264

Zero-half cuts applied: 118

Lift and project cuts applied: 8

Gomory fractional cuts applied: 197

Root node processing (before b&c):

Real time = 0.00 sec. (0.68 ticks)

Parallel b&c, 8 threads:

Real time = 316.13 sec. (270209.62 ticks)

Sync time (average) = 21.13 sec.

Wait time (average) = 0.00 sec.

Total (root+branch&cut) = 316.14 sec. (270210.30 ticks)

Iteration 7

Bounds on # of cuts = 8 with [3 3 2]

Error = 60 (out of 100 instances)

Accuracy = 40

Solving time = 5.268966785 min (minutes)

Accumulated time = 18.655460417 min (minutes)

Solution status code = 104

LB on error = -276.833555011

Relative objective gap = 8.420838875

Selected variables:

PEMLR (Categorical)

SS_YN (Categorical)

Number of selected variables = 2 (0 continuous + 2 categorical)

Version identifier: 22.1.1.0 | 2022-11-28 | 9160aff4d

CPXPARAM_MIP_Strategy_File	3
CPXPARAM_MIP_Limits_Solutions	1
CPXPARAM_TimeLimit	85280.672374999995
CPXPARAM_MIP_Limits_TreeMemory	204800

Nodes				Cuts/				
Node	Left	Objective	IInf	Best Integer	Best Bound	ItCnt	Gap	
62493	56328	-371.5512	875	-40.0000	-376.8336	16793405	842.08%	
Elapsed time = 1.28 sec. (381.47 ticks, tree = 2553.13 MB, solutions = 7)								
Nodefile size = 505.19 MB (457.73 MB after compression)								
62494	56329	-371.1453	802	-40.0000	-376.8336	16795022	842.08%	
62497	56331	-371.0317	798	-40.0000	-376.8336	16796271	842.08%	
62498	56328	-368.1504	812	-40.0000	-376.8336	16797169	842.08%	
62512	56334	-374.4435	690	-40.0000	-376.8336	16798167	842.08%	
62525	56343	-367.3256	732	-40.0000	-376.8336	16801668	842.08%	
62532	56335	-369.7631	729	-40.0000	-376.8336	16799811	842.08%	
62542	56346	infeasible		-40.0000	-376.8336	16803611	842.08%	
62547	56350	-365.8593	652	-40.0000	-376.8336	16804496	842.08%	
62560	56356	-364.8589	634	-40.0000	-376.8336	16805402	842.08%	
62618	56340	-373.4930	705	-40.0000	-376.8336	16804652	842.08%	
Elapsed time = 5.50 sec. (4460.10 ticks, tree = 2546.50 MB, solutions = 7)								
Nodefile size = 505.19 MB (457.73 MB after compression)								
62639	56342	-376.2051	828	-40.0000	-376.8205	16808904	842.05%	
62665	56385	-373.6476	439	-40.0000	-376.8205	16811958	842.05%	
62722	56376	-285.1413	212	-40.0000	-376.8205	16814049	842.05%	

62904	56365	-320.2108	319	-40.0000	-376.8205	16832674	842.05%
62969	56414	-207.5316	159	-40.0000	-376.8205	16836614	842.05%
63094	56358	-361.8225	455	-40.0000	-376.8205	16829520	842.05%
63139	56390	-350.5564	411	-40.0000	-376.8205	16833177	842.05%
63164	56347	-366.7584	820	-40.0000	-376.8205	16827937	842.05%
63232	56547	-261.9177	265	-40.0000	-376.8205	16828650	842.05%
63383	56506	-372.3550	659	-40.0000	-376.8205	16845155	842.05%

Elapsed time = 17.85 sec. (15002.70 ticks, tree = 2538.21 MB, solutions = 7)

Nodefile size = 505.19 MB (457.73 MB after compression)

63431	56370	-361.1593	662	-40.0000	-376.8205	16836599	842.05%
63518	56528	-366.3616	580	-40.0000	-376.8205	16851365	842.05%
63551	56668	-372.7898	614	-40.0000	-376.8205	16838522	842.05%
63617	56356	infeasible		-40.0000	-376.8205	16847550	842.05%
63657	56433	-373.2525	857	-40.0000	-376.5778	16842129	841.44%
63720	56584	infeasible		-40.0000	-376.5778	16866006	841.44%
63742	56708	-347.8130	343	-40.0000	-376.5778	16851562	841.44%
63817	56407	-372.9694	713	-40.0000	-376.5778	16853277	841.44%
63875	56438	-336.4001	379	-40.0000	-376.5778	16856671	841.44%
63937	56457	-370.4856	719	-40.0000	-376.5778	16855716	841.44%

Elapsed time = 29.24 sec. (24884.76 ticks, tree = 2542.06 MB, solutions = 7)

Nodefile size = 505.19 MB (457.73 MB after compression)

63986	56760	-361.2265	352	-40.0000	-376.5778	16863808	841.44%
* 64088+56788				-42.0000	-376.5778		796.61%
64161	56427	-308.4527	287	-42.0000	-376.5778	16861750	796.61%
64305	56426	-364.4031	448	-42.0000	-376.5778	16867050	796.61%
64344	56429	cutoff		-42.0000	-376.5778	16870294	796.61%
64408	56433	-361.1858	402	-42.0000	-376.5778	16864366	796.61%
64514	56592	-369.9594	867	-42.0000	-376.5778	16880737	796.61%
64661	56555	-117.7107	86	-42.0000	-376.5778	16881851	796.61%
64713	56603	-366.8417	779	-42.0000	-376.5778	16886930	796.61%
64780	56605	-366.4698	401	-42.0000	-376.5778	16885487	796.61%
64920	56444	-257.3129	304	-42.0000	-376.5778	16880818	796.61%

Elapsed time = 40.45 sec. (34952.40 ticks, tree = 2527.61 MB, solutions = 8)

Nodefile size = 505.19 MB (457.73 MB after compression)

65008	56483	-373.2735	748	-42.0000	-376.5778	16879719	796.61%
65038	56496	-371.2082	756	-42.0000	-376.5778	16884054	796.61%
65110	56516	-350.0730	368	-42.0000	-376.5778	16888980	796.61%
65158	56558	-235.2681	191	-42.0000	-376.5778	16892346	796.61%
65285	56665	-371.7312	780	-42.0000	-376.5778	16902786	796.61%

65319	56612	-366.8891	1013	-42.0000	-376.5778	16899193	796.61%
65328	56515	-369.2649	813	-42.0000	-376.5778	16898243	796.61%
65349	56619	-365.9130	961	-42.0000	-376.5778	16903443	796.61%
65377	56656	-374.9636	869	-42.0000	-376.5778	16903553	796.61%
65461	56566	-354.8815	435	-42.0000	-376.5778	16906832	796.61%

Elapsed time = 51.75 sec. (45094.74 ticks, tree = 2537.59 MB, solutions = 8)

Nodefile size = 505.19 MB (457.73 MB after compression)

65538	56652	-352.6881	414	-42.0000	-376.5778	16912258	796.61%
65579	56663	-373.6537	807	-42.0000	-376.5778	16911756	796.61%
65603	56591	-375.1709	881	-42.0000	-376.5778	16912590	796.61%
65614	56685	-368.6663	587	-42.0000	-376.5778	16926075	796.61%
65654	56606	-326.8965	270	-42.0000	-376.5778	16919118	796.61%
65840	56689	-344.0531	352	-42.0000	-376.5778	16932035	796.61%
65863	56702	-353.1667	420	-42.0000	-376.5778	16935329	796.61%
65889	56719	-319.7314	372	-42.0000	-376.5778	16938567	796.61%
65911	56725	-322.5031	341	-42.0000	-376.5778	16941732	796.61%
65958	56694	-373.2894	1005	-42.0000	-376.5778	16931254	796.61%

Elapsed time = 62.90 sec. (55166.59 ticks, tree = 2544.63 MB, solutions = 8)

Nodefile size = 505.19 MB (457.73 MB after compression)

65997	56768	-253.7349	280	-42.0000	-376.5778	16947850	796.61%
66033	56799	-200.2774	224	-42.0000	-376.5778	16951162	796.61%
66088	56579	-363.1327	912	-42.0000	-376.5778	16939742	796.61%
66104	56529	-366.7705	832	-42.0000	-376.5778	16919965	796.61%
66121	56540	-363.5003	629	-42.0000	-376.5778	16923506	796.61%
66256	56603	-333.1568	482	-42.0000	-376.5778	16927307	796.61%
66353	56781	-358.3175	407	-42.0000	-376.5778	16950761	796.61%
66420	56807	-333.2784	378	-42.0000	-376.5778	16954452	796.61%
66540	56939	-273.2919	280	-42.0000	-376.5778	16973671	796.61%
66663	56741	-346.7942	445	-42.0000	-376.5778	16960552	796.61%

Elapsed time = 74.45 sec. (65498.15 ticks, tree = 2558.25 MB, solutions = 8)

Nodefile size = 505.19 MB (457.73 MB after compression)

66746	56738	-367.2268	477	-42.0000	-376.5778	16960134	796.61%
67127	56776	-342.2970	517	-42.0000	-376.5778	16968251	796.61%
67238	56999	-371.4690	752	-42.0000	-376.5778	16983874	796.61%
67266	57021	-333.9642	341	-42.0000	-376.5778	16987177	796.61%
67424	56898	-76.4479	160	-42.0000	-376.5778	16978530	796.61%
67530	57053	-373.1439	621	-42.0000	-376.5778	16993918	796.61%
67784	57039	-91.9323	75	-42.0000	-376.5778	16976965	796.61%
67923	57209	-374.6908	824	-42.0000	-376.5778	17000708	796.61%

67941	56855	-372.2216	428	-42.0000	-376.5778	16977830	796.61%
68103	56928	-331.4462	469	-42.0000	-376.5778	16990292	796.61%
Elapsed time = 85.79 sec. (75199.50 ticks, tree = 2566.21 MB, solutions = 8)							
Nodefile size = 505.19 MB (457.73 MB after compression)							
68294	56955	-247.9800	326	-42.0000	-376.5778	16984262	796.61%
68361	56926	infeasible		-42.0000	-376.5778	16995251	796.61%
68454	56941	-342.9196	259	-42.0000	-376.5778	16998376	796.61%
68580	56992	-249.5999	176	-42.0000	-376.5778	17002343	796.61%
68785	57111	-334.3761	321	-42.0000	-376.5778	16998461	796.61%
68946	56636	-370.2219	1051	-42.0000	-376.5778	16959597	796.61%
68960	57248	infeasible		-42.0000	-376.5778	17004796	796.61%
69028	57254	infeasible		-42.0000	-376.5778	17007632	796.61%
69137	57046	cutoff		-42.0000	-376.5778	17011825	796.61%
69146	57258	-374.6623	732	-42.0000	-376.5778	17013586	796.61%
Elapsed time = 96.76 sec. (84893.06 ticks, tree = 2553.89 MB, solutions = 8)							
Nodefile size = 505.19 MB (457.73 MB after compression)							
69174	57079	-374.6247	714	-42.0000	-376.5778	17010021	796.61%
69379	57535	-154.2234	96	-42.0000	-376.5778	17040415	796.61%
69542	57048	-351.2630	495	-42.0000	-376.5778	17024459	796.61%
69611	57092	-249.8471	236	-42.0000	-376.5778	17027482	796.61%
69791	57123	-266.5293	180	-42.0000	-376.5778	17021100	796.61%
69896	57588	infeasible		-42.0000	-376.5778	17051599	796.61%
69900	57201	-372.7224	744	-42.0000	-376.5778	17026463	796.61%
69921	57253	-355.6185	324	-42.0000	-376.5778	17037474	796.61%
70000	57290	-294.2659	218	-42.0000	-376.5778	17038577	796.61%
70742	57498	-268.3576	172	-42.0000	-376.5778	17052046	796.61%
Elapsed time = 111.68 sec. (97530.70 ticks, tree = 2556.97 MB, solutions = 9)							
Nodefile size = 505.19 MB (457.73 MB after compression)							
71350	57728	-348.3798	248	-42.0000	-376.5778	17063553	796.61%
71743	57460	-373.4932	966	-42.0000	-376.5778	17049682	796.61%
71755	57653	-370.4464	636	-42.0000	-376.5778	17078692	796.61%
72187	57551	-215.6947	184	-42.0000	-376.5778	17066389	796.61%
72276	57850	-367.7073	661	-42.0000	-376.5778	17096223	796.61%
72618	57937	-157.8367	118	-42.0000	-376.5778	17107344	796.61%
73589	58103	-366.4535	645	-42.0000	-376.5778	17117318	796.61%
74114	58372	-220.1499	157	-42.0000	-376.5778	17127622	796.61%
74271	57688	-366.0440	1022	-42.0000	-376.5778	17107476	796.61%
74330	58468	-278.5357	406	-42.0000	-376.5778	17158803	796.61%
Elapsed time = 151.49 sec. (136817.80 ticks, tree = 2572.25 MB, solutions = 9)							

Nodefile size = 505.19 MB (457.73 MB after compression)

74466	57137	-373.4744	786	-42.0000	-376.5778	17075738	796.61%
74579	57229	-125.4892	163	-42.0000	-376.5778	17083210	796.61%
74997	58823	-124.1569	77	-42.0000	-376.5778	17191261	796.61%
75138	57699	-351.9220	642	-42.0000	-376.5778	17150191	796.61%
75183	57727	-312.4915	310	-42.0000	-376.5778	17159655	796.61%
75598	57528	cutoff		-42.0000	-376.5778	17113472	796.61%
75757	57645	-372.7641	917	-42.0000	-376.5778	17123059	796.61%
75781	57652	-363.8514	654	-42.0000	-376.5778	17133198	796.61%
75914	57770	-368.0621	641	-42.0000	-376.5778	17141831	796.61%
76307	57888	-367.1160	710	-42.0000	-376.5778	17151066	796.61%

Elapsed time = 182.24 sec. (176632.97 ticks, tree = 2640.89 MB, solutions = 10)

Nodefile size = 505.19 MB (457.73 MB after compression)

76625	58372	-276.7328	275	-42.0000	-376.5778	17215778	796.61%
76896	58881	-363.3072	789	-42.0000	-376.5778	17266359	796.61%
77088	58341	-205.9310	178	-42.0000	-376.5778	17178206	796.61%
77452	58913	-289.3936	190	-42.0000	-376.5778	17282930	796.61%
77896	58805	-59.1187	71	-42.0000	-376.5778	17249555	796.61%
78198	58777	-104.5215	100	-42.0000	-376.5778	17202653	796.61%
78213	58926	-368.4008	652	-42.0000	-376.5778	17267554	796.61%
78401	58786	-372.5803	1028	-42.0000	-376.5778	17218675	796.61%
78547	59186	-304.1862	222	-42.0000	-376.5778	17287687	796.61%
78819	59304	-355.0582	246	-42.0000	-376.5778	17297787	796.61%

Elapsed time = 213.20 sec. (216193.23 ticks, tree = 2715.45 MB, solutions = 11)

Nodefile size = 505.19 MB (457.73 MB after compression)

* 78861+59332				-43.0000	-376.5778		775.76%
78863	58919	-372.1730	923	-43.0000	-376.5778	17245546	775.76%
78865	58921	-362.0419	695	-43.0000	-376.5778	17262028	775.76%
78908	58961	-287.4919	224	-43.0000	-376.5778	17274297	775.76%
79046	59079	-351.1997	373	-43.0000	-376.5778	17285981	775.76%
79251	59269	-136.6352	93	-43.0000	-376.5778	17296280	775.76%
79485	59473	-331.8259	277	-43.0000	-376.5778	17304844	775.76%
79610	59575	-368.4471	613	-43.0000	-376.5778	17311192	775.76%
79779	59736	-235.4136	155	-43.0000	-376.5778	17320410	775.76%
79874	59817	infeasible		-43.0000	-376.5778	17331372	775.76%
79976	59913	-352.6447	359	-43.0000	-376.5778	17343498	775.76%

Elapsed time = 240.73 sec. (260469.65 ticks, tree = 2807.01 MB, solutions = 12)

Nodefile size = 505.19 MB (457.73 MB after compression)

80225	60135	-351.0370	334	-43.0000	-376.5778	17359520	775.76%
-------	-------	-----------	-----	----------	-----------	----------	---------

80482	60367	-372.6844	1128	-43.0000	-376.5778	17379949	775.76%
80486	60371	-370.6348	854	-43.0000	-376.5778	17391940	775.76%
80489	60374	-366.6817	652	-43.0000	-376.5778	17402505	775.76%
80618	60496	-366.6781	671	-43.0000	-376.5778	17414997	775.76%
80896	60756	cutoff		-43.0000	-376.5778	17426348	775.76%
81024	60865	-368.8018	769	-43.0000	-376.5778	17439191	775.76%
81161	60978	infeasible		-43.0000	-376.5778	17452029	775.76%
81372	61172	-368.2417	708	-43.0000	-376.5778	17463161	775.76%

GUB cover cuts applied: 916

Clique cuts applied: 53

Cover cuts applied: 3875

Implied bound cuts applied: 59

Flow cuts applied: 100

Mixed integer rounding cuts applied: 1398

Zero-half cuts applied: 121

Lift and project cuts applied: 9

Gomory fractional cuts applied: 198

Root node processing (before b&c):

Real time = 0.00 sec. (0.97 ticks)

Parallel b&c, 8 threads:

Real time = 278.62 sec. (311816.15 ticks)

Sync time (average) = 11.93 sec.

Wait time (average) = 0.00 sec.

Total (root+branch&cut) = 278.62 sec. (311817.12 ticks)

Iteration 8

Bounds on # of cuts = 8 with [3 3 2]

Error = 57 (out of 100 instances)

Accuracy = 43

Solving time = 4.643691231 min (minutes)

Accumulated time = 23.299151648 min (minutes)

Solution status code = 104

LB on error = -276.380316895

Relative objective gap = 7.753030625

Selected variables:

PEMLR (Categorical)

SS_YN (Categorical)

Number of selected variables = 2 (0 continuous + 2 categorical)

Version identifier: 22.1.1.0 | 2022-11-28 | 9160aff4d

CPXPARAM_MIP_Strategy_File	3
CPXPARAM_MIP_Limits_Solutions	1
CPXPARAM_TimeLimit	85002.050901123046
CPXPARAM_MIP_Limits_TreeMemory	204800

Nodes		Cuts/					
Node	Left	Objective	IInf	Best Integer	Best Bound	ItCnt	Gap
81435	71926	-369.0867	475	-43.0000	-376.3803	19472128	775.30%
Elapsed time = 3.10 sec. (2384.36 ticks, tree = 3375.96 MB, solutions = 13)							
Nodefile size = 1328.84 MB (1208.58 MB after compression)							
81438	71929	-351.7855	242	-43.0000	-376.3803	19472768	775.30%
81471	71961	-270.8111	173	-43.0000	-376.3803	19473216	775.30%
81515	72001	-157.5837	98	-43.0000	-376.3803	19473641	775.30%
81563	71926	-374.3382	753	-43.0000	-376.3803	19473609	775.30%
81564	71928	-372.4402	715	-43.0000	-376.3803	19481748	775.30%
81567	71930	-371.4346	663	-43.0000	-376.3803	19482880	775.30%
81571	71933	-371.2108	644	-43.0000	-376.3803	19483675	775.30%
81572	71934	-370.7107	636	-43.0000	-376.3803	19484433	775.30%
81576	71931	-360.4175	763	-43.0000	-376.3803	19502020	775.30%
81661	72012	-144.5408	107	-43.0000	-376.3803	19488685	775.30%
Elapsed time = 9.37 sec. (7373.28 ticks, tree = 3370.21 MB, solutions = 13)							
Nodefile size = 1328.84 MB (1208.58 MB after compression)							
81706	72044	-358.3163	625	-43.0000	-376.3803	19488229	775.30%
81757	72071	-303.3940	260	-43.0000	-376.3803	19492258	775.30%
82046	72027	-154.9862	247	-43.0000	-376.3803	19515571	775.30%
82190	72089	-266.6773	179	-43.0000	-376.3803	19502642	775.30%
82339	72085	-249.3634	162	-43.0000	-376.3803	19503964	775.30%
82548	71934	-359.2701	768	-43.0000	-376.3803	19543860	775.30%
82551	71937	-357.5116	530	-43.0000	-376.3803	19547438	775.30%
82574	71952	-339.6193	324	-43.0000	-376.3803	19550709	775.30%

82657	72026	-155.4003	164	-43.0000	-376.3803	19552766	775.30%
82707	72064	-355.4909	247	-43.0000	-376.3803	19555144	775.30%
Elapsed time = 23.10 sec. (18322.56 ticks, tree = 3345.19 MB, solutions = 13)							
Nodefile size = 1328.84 MB (1208.58 MB after compression)							
82804	72154	-136.7153	96	-43.0000	-376.3527	19557777	775.24%
82850	72307	-360.9410	741	-43.0000	-376.3527	19521065	775.24%
82855	72311	-342.6034	249	-43.0000	-376.1368	19523927	774.74%
82983	71936	-362.8146	626	-43.0000	-376.1368	19553992	774.74%
82985	72161	-371.0562	970	-43.0000	-376.1368	19520957	774.74%
82987	72193	-374.9001	951	-43.0000	-376.1368	19562191	774.74%
82993	71941	-358.9686	430	-43.0000	-376.1368	19567870	774.74%
82999	71946	-348.8783	255	-43.0000	-376.1368	19572436	774.74%
83050	71987	-258.2577	225	-43.0000	-376.1368	19574685	774.74%
83115	72047	-106.9395	149	-43.0000	-376.1368	19575766	774.74%
Elapsed time = 35.85 sec. (28977.15 ticks, tree = 3351.66 MB, solutions = 13)							
Nodefile size = 1328.84 MB (1208.58 MB after compression)							
83146	72066	infeasible		-43.0000	-376.1368	19578155	774.74%
83150	72197	-355.5337	338	-43.0000	-376.1368	19576926	774.74%
83255	72221	-222.3896	129	-43.0000	-376.1368	19540293	774.74%
83429	72318	cutoff		-43.0000	-376.1368	19579829	774.74%
83434	72191	-359.3736	818	-43.0000	-376.1368	19542416	774.74%
83449	72287	-371.6337	619	-43.0000	-376.1368	19550427	774.74%
83487	72317	-265.8980	174	-43.0000	-376.1368	19553782	774.74%
83744	72447	-333.6008	253	-43.0000	-376.1368	19553268	774.74%
83839	72511	-194.0038	122	-43.0000	-376.1368	19554512	774.74%
83950	72252	-192.8584	143	-43.0000	-376.1368	19555244	774.74%
Elapsed time = 48.25 sec. (39354.87 ticks, tree = 3364.70 MB, solutions = 13)							
Nodefile size = 1328.84 MB (1208.58 MB after compression)							
84017	72081	-354.4707	699	-43.0000	-376.1368	19578452	774.74%
84023	72069	-366.0459	957	-43.0000	-376.1368	19597048	774.74%
84036	71956	-348.2522	252	-43.0000	-376.1368	19588460	774.74%
84184	72414	-333.2872	216	-43.0000	-376.1368	19572926	774.74%
84316	72094	-351.1716	513	-43.0000	-376.1368	19591693	774.74%
84492	72137	-253.3089	162	-43.0000	-376.1368	19595305	774.74%
84589	72520	-372.8630	619	-43.0000	-376.1368	19581721	774.74%
84631	72559	-273.3236	176	-43.0000	-376.1368	19584387	774.74%
84755	72337	-342.0960	228	-43.0000	-376.1368	19606229	774.74%
84999	72657	-350.7128	344	-43.0000	-376.1368	19589927	774.74%
Elapsed time = 59.71 sec. (49483.74 ticks, tree = 3372.85 MB, solutions = 13)							

Nodefile size = 1328.84 MB (1208.58 MB after compression)

85177	72284	-200.4956	128	-43.0000	-376.1368	19608089	774.74%
85252	72564	-366.6790	841	-43.0000	-376.1368	19577339	774.74%
85269	72536	infeasible		-43.0000	-376.1368	19591276	774.74%
85318	72802	-272.0554	179	-43.0000	-376.1368	19598160	774.74%
85458	72106	-271.7240	204	-43.0000	-376.1368	19630436	774.74%
85583	72343	-369.7473	635	-43.0000	-376.1368	19620884	774.74%
85753	72537	-368.2100	867	-43.0000	-376.1368	19598411	774.74%
85869	72101	-319.9908	214	-43.0000	-376.1368	19628379	774.74%
86144	72573	-271.7490	196	-43.0000	-376.1368	19604341	774.74%
86291	72116	-306.6315	350	-43.0000	-376.1368	19628397	774.74%

Elapsed time = 72.00 sec. (59610.39 ticks, tree = 3339.28 MB, solutions = 13)

Nodefile size = 1328.84 MB (1208.58 MB after compression)

86447	72541	-125.2543	65	-43.0000	-376.1368	19631988	774.74%
86788	72227	-46.0606	174	-43.0000	-376.1368	19635812	774.74%
86904	72855	-90.1545	54	-43.0000	-376.1368	19614400	774.74%
86987	72371	-183.6352	116	-43.0000	-376.1368	19641703	774.74%
87059	72230	-371.5048	694	-43.0000	-376.1368	19641837	774.74%
87140	72762	-210.2776	139	-43.0000	-376.1368	19616474	774.74%
87328	72220	-366.7426	657	-43.0000	-376.1368	19654622	774.74%
87338	72227	-342.5130	261	-43.0000	-376.1368	19658216	774.74%
87497	73022	-331.2281	315	-43.0000	-376.1368	19630982	774.74%
87569	73080	-184.5925	181	-43.0000	-376.1368	19631567	774.74%

Elapsed time = 83.56 sec. (69415.07 ticks, tree = 3374.71 MB, solutions = 13)

Nodefile size = 1328.84 MB (1208.58 MB after compression)

87722	72426	-196.0292	354	-43.0000	-376.1368	19658817	774.74%
87813	72470	-313.7015	224	-43.0000	-376.1368	19657117	774.74%
88010	73139	-332.3763	232	-43.0000	-376.1368	19638071	774.74%
88193	72344	-369.2216	688	-43.0000	-376.1368	19671787	774.74%
88226	72439	-344.1819	306	-43.0000	-376.1368	19663453	774.74%
88382	73331	-108.0697	65	-43.0000	-376.1368	19645596	774.74%
88585	72524	-154.1781	103	-43.0000	-376.1368	19666440	774.74%
88643	72580	-341.4974	385	-43.0000	-376.1368	19679295	774.74%
88686	72613	-276.5086	221	-43.0000	-376.1368	19683720	774.74%
88853	73404	-204.9500	146	-43.0000	-376.1368	19657466	774.74%

Elapsed time = 96.42 sec. (79164.76 ticks, tree = 3376.03 MB, solutions = 13)

Nodefile size = 1328.84 MB (1208.58 MB after compression)

88930	72572	-371.0431	728	-43.0000	-376.1368	19675302	774.74%
88947	72490	-365.6207	494	-43.0000	-376.1368	19682400	774.74%

89059	72630	-210.4036	137	-43.0000	-376.1368	19680104	774.74%
89353	73556	-174.7126	116	-43.0000	-376.1368	19669157	774.74%
89437	72881	-348.1156	280	-43.0000	-376.1368	19641906	774.74%
89803	72768	-138.0596	89	-43.0000	-376.1368	19686828	774.74%
90099	73621	-369.2874	668	-43.0000	-376.1368	19676578	774.74%
90187	72870	-145.8195	96	-43.0000	-376.1368	19692884	774.74%
90391	72907	-350.2219	270	-43.0000	-376.1368	19696002	774.74%
90568	72953	-260.3118	162	-43.0000	-376.1368	19699570	774.74%

Elapsed time = 107.23 sec. (88918.59 ticks, tree = 3355.42 MB, solutions = 13)

Nodefile size = 1328.84 MB (1208.58 MB after compression)

90747	73000	-232.2603	144	-43.0000	-376.1368	19670487	774.74%
90865	73685	-212.0829	144	-43.0000	-376.1368	19685798	774.74%
91021	73711	-147.6431	128	-43.0000	-376.1368	19686698	774.74%
91106	73041	-279.0274	184	-43.0000	-376.1368	19662197	774.74%
91217	72694	-367.0982	639	-43.0000	-376.1368	19714979	774.74%
91226	72696	-364.1187	634	-43.0000	-376.1368	19718381	774.74%
91279	73087	-334.1832	285	-43.0000	-376.1368	19684478	774.74%
91581	72886	-239.2958	177	-43.0000	-376.1368	19731529	774.74%
91880	72773	-172.1287	147	-43.0000	-376.1368	19724728	774.74%
92103	72979	-318.1041	204	-43.0000	-376.1368	19741671	774.74%

Elapsed time = 122.50 sec. (101620.07 ticks, tree = 3349.38 MB, solutions = 13)

Nodefile size = 1328.84 MB (1208.58 MB after compression)

92488	73193	-370.3096	645	-43.0000	-376.1368	19705612	774.74%
92693	73253	-263.1795	257	-43.0000	-376.1368	19716357	774.74%
93074	72883	-356.9105	851	-43.0000	-376.1368	19744324	774.74%
93449	72909	-298.8729	213	-43.0000	-376.1368	19753841	774.74%
94224	73670	-245.3925	164	-43.0000	-376.1368	19727039	774.74%
95151	73027	-142.9352	102	-43.0000	-376.1368	19773585	774.74%
96004	73918	-234.3377	168	-43.0000	-376.1368	19744955	774.74%
96537	75077	-76.1794	41	-43.0000	-376.1368	19787753	774.74%
97188	73361	-339.1905	244	-43.0000	-376.1368	19803446	774.74%
97957	73408	-98.3815	50	-43.0000	-376.1368	19811535	774.74%

Elapsed time = 169.17 sec. (140231.77 ticks, tree = 3363.58 MB, solutions = 13)

Nodefile size = 1328.84 MB (1208.58 MB after compression)

98359	73123	-368.1863	739	-43.0000	-376.1368	19803156	774.74%
99166	73425	-362.4546	512	-43.0000	-376.1368	19828990	774.74%
99573	74050	-348.9752	262	-43.0000	-376.1368	19832253	774.74%

*100206+75506				-44.0000	-376.1368		754.86%
---------------	--	--	--	----------	-----------	--	---------

100260	74255	-187.1122	111	-44.0000	-376.1368	19841225	754.86%
--------	-------	-----------	-----	----------	-----------	----------	---------

100755	73923	-141.6525	76	-44.0000	-376.1368	19855053	754.86%
101360	74183	-254.1317	165	-44.0000	-376.1368	19869790	754.86%
101983	74396	-365.9683	679	-44.0000	-376.1368	19880718	754.86%
102256	73251	-358.7644	844	-44.0000	-376.1368	19850813	754.86%
102534	74551	-369.3515	969	-44.0000	-376.1368	19886540	754.86%
102636	74685	-201.0157	189	-44.0000	-376.1368	19906540	754.86%

Elapsed time = 213.97 sec. (178886.70 ticks, tree = 3369.28 MB, solutions = 14)

Nodefile size = 1328.84 MB (1208.58 MB after compression)

103075	74690	-137.0039	86	-44.0000	-376.1368	19858793	754.86%
103324	74781	-259.5830	176	-44.0000	-376.1368	19870758	754.86%
103868	73707	-357.3355	710	-44.0000	-376.1368	19916863	754.86%
104488	74966	-156.8062	105	-44.0000	-376.1368	19935045	754.86%
104840	73993	-298.3118	195	-44.0000	-376.1368	19937028	754.86%
105230	74463	-370.2003	680	-44.0000	-376.1368	19945863	754.86%
106185	75111	-328.7164	230	-44.0000	-376.1368	19921098	754.86%
106948	74994	-64.9639	51	-44.0000	-376.1368	19976173	754.86%
107545	75108	-275.3044	179	-44.0000	-376.1368	19985828	754.86%
108190	74573	-122.1881	75	-44.0000	-376.1368	19949611	754.86%

Elapsed time = 255.53 sec. (217150.33 ticks, tree = 3470.33 MB, solutions = 15)

Nodefile size = 1328.84 MB (1208.58 MB after compression)

108890	74604	-302.1580	203	-44.0000	-376.1368	19988782	754.86%
109683	74939	-72.1704	42	-44.0000	-376.1368	19996434	754.86%
110213	75844	-152.6764	113	-44.0000	-376.1368	20025497	754.86%
110687	74738	-372.4462	654	-44.0000	-376.1368	19979491	754.86%
111104	75218	-255.9594	181	-44.0000	-376.1368	20021953	754.86%
111226	75914	-280.6563	210	-44.0000	-376.1368	20052210	754.86%
111818	75067	-108.0074	75	-44.0000	-376.1368	20004989	754.86%
112086	74183	-175.7369	112	-44.0000	-376.1368	20000307	754.86%
112565	76266	-241.4723	169	-44.0000	-376.1368	20079748	754.86%
113030	76439	-164.3090	212	-44.0000	-376.1368	20090558	754.86%

Elapsed time = 300.62 sec. (255370.86 ticks, tree = 3493.07 MB, solutions = 16)

Nodefile size = 1328.84 MB (1208.58 MB after compression)

113440	75423	-356.3437	390	-44.0000	-376.1368	20066747	754.86%
113846	75655	-329.7125	232	-44.0000	-376.1368	20074973	754.86%
114376	76956	-372.7014	765	-44.0000	-376.1368	20119752	754.86%
114992	75269	-357.7460	861	-44.0000	-376.1368	20084688	754.86%
115064	75899	-244.1868	170	-44.0000	-376.1368	20104973	754.86%
115422	75281	-356.3431	761	-44.0000	-376.1368	20099647	754.86%
115659	77257	-354.9574	229	-44.0000	-376.1368	20156509	754.86%

116481	77486	-350.4063	259	-44.0000	-376.1368	20166270	754.86%
117012	74851	-338.6518	338	-44.0000	-376.1368	20099673	754.86%
117672	76593	-73.3108	55	-44.0000	-376.1368	20139708	754.86%
Elapsed time = 340.22 sec. (294342.29 ticks, tree = 3590.75 MB, solutions = 16)							
Nodefile size = 1328.84 MB (1208.58 MB after compression)							
118392	76837	-340.1003	223	-44.0000	-376.1368	20149430	754.86%
118661	76951	-338.5554	252	-44.0000	-376.1368	20159026	754.86%
119232	78202	-370.3891	879	-44.0000	-376.1368	20206881	754.86%
119504	78208	-365.8875	675	-44.0000	-376.1368	20215279	754.86%
120370	78339	-294.4108	228	-44.0000	-376.1368	20224887	754.86%
120736	77663	-371.1597	751	-44.0000	-376.1368	20197661	754.86%
121137	78468	-263.1958	171	-44.0000	-376.1368	20240044	754.86%
121380	76760	-361.0483	578	-44.0000	-376.1368	20206996	754.86%
121504	78546	-372.0299	951	-44.0000	-376.1368	20254287	754.86%
121709	77052	-189.2273	149	-44.0000	-376.1368	20224660	754.86%
Elapsed time = 369.68 sec. (333296.33 ticks, tree = 3591.38 MB, solutions = 17)							
Nodefile size = 1328.84 MB (1208.58 MB after compression)							
121787	77894	-347.8439	314	-44.0000	-376.1368	20232409	754.86%
121944	78563	-362.8561	674	-44.0000	-376.1368	20281358	754.86%
122228	78030	-369.6159	952	-44.0000	-376.1368	20251103	754.86%
122239	78035	-365.2604	668	-44.0000	-376.1368	20260623	754.86%
122604	79033	-116.1095	76	-44.0000	-376.1368	20304884	754.86%
122641	77112	-360.7584	740	-44.0000	-376.1368	20264111	754.86%
122909	79061	-359.0364	761	-44.0000	-376.1368	20328240	754.86%
122912	79064	-345.4132	728	-44.0000	-376.1368	20339694	754.86%
123088	79214	-340.4910	331	-44.0000	-376.1368	20349332	754.86%
123335	79434	-366.9963	736	-44.0000	-376.1368	20359227	754.86%
Elapsed time = 386.93 sec. (378266.74 ticks, tree = 3689.23 MB, solutions = 18)							
Nodefile size = 1328.84 MB (1208.58 MB after compression)							
123606	77360	-367.9078	813	-44.0000	-376.1368	20304015	754.86%
123789	77402	-270.6668	213	-44.0000	-376.1368	20313171	754.86%
124252	77565	-147.9091	111	-44.0000	-376.1368	20322273	754.86%
124710	80275	-339.1575	259	-44.0000	-376.1368	20399552	754.86%
125071	80455	-130.4602	96	-44.0000	-376.1368	20407189	754.86%
125120	80488	-364.5023	979	-44.0000	-376.1368	20414180	754.86%
125393	78074	-67.5268	51	-44.0000	-376.1368	20356949	754.86%
125548	78206	-61.4737	25	-44.0000	-376.1368	20365875	754.86%
125824	80617	-365.8242	646	-44.0000	-376.1368	20434361	754.86%
126311	80820	-149.0581	106	-44.0000	-376.1368	20445898	754.86%

Elapsed time = 402.92 sec. (417930.32 ticks, tree = 3750.55 MB, solutions = 18)

Nodefile size = 1328.84 MB (1208.58 MB after compression)

126818	78688	-359.1233	621	-44.0000	-376.1368	20397046	754.86%
127130	78818	-361.6743	326	-44.0000	-376.1368	20405785	754.86%
127645	81464	infeasible		-44.0000	-376.1368	20480980	754.86%
127773	79050	-370.5096	967	-44.0000	-376.1368	20420424	754.86%
128047	79052	-369.8842	919	-44.0000	-376.1368	20426470	754.86%
128292	82017	-100.5857	62	-44.0000	-376.1368	20513017	754.86%
128338	82042	-364.7610	328	-44.0000	-376.1368	20523898	754.86%
128527	79071	-360.6063	650	-44.0000	-376.1368	20446426	754.86%
128837	82360	-201.0485	146	-44.0000	-376.1368	20543286	754.86%
129157	82638	-369.1021	836	-44.0000	-376.1368	20553236	754.86%

Elapsed time = 433.80 sec. (460667.96 ticks, tree = 3767.71 MB, solutions = 19)

Nodefile size = 1328.84 MB (1208.58 MB after compression)

129298	82771	-49.6236	49	-44.0000	-376.1368	20560041	754.86%
129563	83000	-360.7373	441	-44.0000	-376.1368	20569164	754.86%
129908	83292	-150.2737	107	-44.0000	-376.1368	20580126	754.86%
130141	83494	-240.2222	141	-44.0000	-376.1368	20590944	754.86%
130362	83680	-370.2243	736	-44.0000	-376.1368	20601487	754.86%
130505	83814	-359.7878	388	-44.0000	-376.1368	20614667	754.86%
130871	84128	-128.4628	80	-44.0000	-376.1368	20623685	754.86%

GUB cover cuts applied: 1043

Clique cuts applied: 57

Cover cuts applied: 4277

Implied bound cuts applied: 68

Flow cuts applied: 118

Mixed integer rounding cuts applied: 1735

Zero-half cuts applied: 125

Lift and project cuts applied: 9

Gomory fractional cuts applied: 199

Root node processing (before b&c):

Real time = 0.00 sec. (1.39 ticks)

Parallel b&c, 8 threads:

Real time = 460.44 sec. (493387.72 ticks)

Sync time (average) = 9.29 sec.

Wait time (average) = 0.00 sec.

Total (root+branch&cut) = 460.44 sec. (493389.11 ticks)

Iteration 9

Bounds on # of cuts = 8 with [3 3 2]

Error = 56 (out of 100 instances)

Accuracy = 44

Solving time = 7.674096716 min (minutes)

Accumulated time = 30.973248364 min (minutes)

Solution status code = 104

LB on error = -275.942710447

Relative objective gap = 7.54415251

Selected variables:

PEMLR (Categorical)

SS_YN (Categorical)

Number of selected variables = 2 (0 continuous + 2 categorical)

Version identifier: 22.1.1.0 | 2022-11-28 | 9160aff4d

CPXPARAM_MIP_Strategy_File	3
CPXPARAM_MIP_Limits_Solutions	1
CPXPARAM_TimeLimit	84541.60509814453
CPXPARAM_MIP_Limits_TreeMemory	204800

Nodes				Cuts/			
Node	Left	Objective	IInf	Best Integer	Best Bound	ItCnt	Gap
130908	115417	infeasible		-44.0000	-375.9427	24852547	754.42%
Elapsed time = 0.21 sec. (11.76 ticks, tree = 4875.97 MB, solutions = 20)							
Nodefile size = 2828.82 MB (2551.19 MB after compression)							
130946	115453	-75.7118	39	-44.0000	-375.9427	24852830	754.42%
130960	115461	infeasible		-44.0000	-375.9427	24854245	754.42%
130961	115417	infeasible		-44.0000	-375.9427	24855273	754.42%
130962	115419	-375.5477	854	-44.0000	-375.9427	24854727	754.42%
130963	115462	-374.5850	952	-44.0000	-375.9427	24856324	754.42%
130964	115419	-374.8863	762	-44.0000	-375.9239	24859471	754.37%
130966	115420	-370.7308	599	-44.0000	-375.9239	24861461	754.37%

130970	115420	-365.1263	677	-44.0000	-375.9239	24865366	754.37%
130974	115425	-353.5129	272	-44.0000	-375.9239	24863328	754.37%
131072	115431	-318.7079	260	-44.0000	-375.9239	24868843	754.37%

Elapsed time = 6.88 sec. (4499.63 ticks, tree = 4853.98 MB, solutions = 20)

Nodefile size = 2828.82 MB (2551.19 MB after compression)

131222	115489	-186.2469	144	-44.0000	-375.9239	24869233	754.37%
131419	115588	-227.5296	151	-44.0000	-375.9064	24867646	754.33%
131578	115473	-365.3839	660	-44.0000	-375.9064	24885085	754.33%
131721	115419	-364.6390	889	-44.0000	-375.9064	24870560	754.33%
131724	115420	-363.4877	908	-44.0000	-375.9064	24872570	754.33%
131728	115601	-368.4619	691	-44.0000	-375.9064	24886633	754.33%
131749	115614	-344.4963	237	-44.0000	-375.9064	24889538	754.33%
131877	115538	-365.3678	477	-44.0000	-375.9064	24884596	754.33%
131914	115567	-282.2076	191	-44.0000	-375.9064	24887643	754.33%
132035	115773	-357.5782	691	-44.0000	-375.9064	24887689	754.33%

Elapsed time = 20.65 sec. (15475.37 ticks, tree = 4891.62 MB, solutions = 20)

Nodefile size = 2828.82 MB (2551.19 MB after compression)

132298	115478	-363.3619	938	-44.0000	-375.9064	24905374	754.33%
132305	115885	-365.9278	666	-44.0000	-375.9064	24893668	754.33%
132438	115476	-371.2272	830	-44.0000	-375.9064	24902926	754.33%
132450	115660	-367.9299	777	-44.0000	-375.6031	24899669	753.64%
132461	115721	-365.2211	688	-44.0000	-375.6031	24906865	753.64%
132632	115757	-133.2122	81	-44.0000	-375.6031	24904356	753.64%
132765	115485	-337.9961	301	-44.0000	-375.6031	24920324	753.64%
132903	115483	-354.2930	485	-44.0000	-375.6031	24918321	753.64%
132974	115524	-276.5625	264	-44.0000	-375.6031	24920543	753.64%
133138	116022	-319.0843	203	-44.0000	-375.6031	24912566	753.64%

Elapsed time = 33.13 sec. (25766.01 ticks, tree = 4914.84 MB, solutions = 20)

Nodefile size = 2828.82 MB (2551.19 MB after compression)

133451	115867	-144.2583	99	-44.0000	-375.6031	24916493	753.64%
133606	116211	-45.0030	24	-44.0000	-375.6031	24916465	753.64%
133745	115906	-235.1235	148	-44.0000	-375.6031	24924857	753.64%
134004	115642	-254.9202	165	-44.0000	-375.6031	24933906	753.64%
134329	115597	-79.8389	61	-44.0000	-375.6031	24927955	753.64%
134423	116045	-141.1138	100	-44.0000	-375.6031	24933151	753.64%
134467	115718	-373.6129	866	-44.0000	-375.6031	24938524	753.64%
134536	115666	-199.6527	129	-44.0000	-375.6031	24932904	753.64%
134612	115721	-354.9792	377	-44.0000	-375.6031	24943712	753.64%
134717	116230	-335.6979	239	-44.0000	-375.6031	24937404	753.64%

Elapsed time = 44.97 sec. (35418.05 ticks, tree = 4934.31 MB, solutions = 20)

Nodefile size = 2828.82 MB (2551.19 MB after compression)

135026	115538	-238.0279	154	-44.0000	-375.6031	24941038	753.64%
135204	115612	-369.4843	771	-44.0000	-375.6031	24941848	753.64%
135514	116263	-248.8541	173	-44.0000	-375.6031	24945872	753.64%
135735	115696	-158.7882	110	-44.0000	-375.6031	24946026	753.64%
135885	116414	-164.6318	107	-44.0000	-375.6031	24949736	753.64%
136157	115950	-98.4789	83	-44.0000	-375.6031	24954128	753.64%
136177	116049	-371.4216	844	-44.0000	-375.6031	24942096	753.64%
136184	115729	-359.2447	647	-44.0000	-375.6031	24956224	753.64%
136316	116080	-98.6373	67	-44.0000	-375.6031	24961109	753.64%
136393	116100	-349.1181	249	-44.0000	-375.6031	24963534	753.64%

Elapsed time = 57.63 sec. (45651.53 ticks, tree = 4905.68 MB, solutions = 20)

Nodefile size = 2828.82 MB (2551.19 MB after compression)

136849	115888	-253.3435	164	-44.0000	-375.6031	24959565	753.64%
137050	116213	-367.0710	403	-44.0000	-375.6031	24967308	753.64%
137303	115892	-370.9447	739	-44.0000	-375.6031	24969401	753.64%
137389	116655	-171.9714	119	-44.0000	-375.6031	24969048	753.64%
137495	115924	-298.0111	212	-44.0000	-375.6031	24973699	753.64%
137631	115998	-263.0363	183	-44.0000	-375.6031	24969968	753.64%
137795	116758	-248.3035	154	-44.0000	-375.6031	24976581	753.64%
137891	116269	-369.5074	798	-44.0000	-375.6031	24966502	753.64%
137901	115848	-360.4711	938	-44.0000	-375.5768	24990340	753.58%
137906	116274	-365.3659	725	-44.0000	-375.5768	24971064	753.58%

Elapsed time = 70.87 sec. (56061.54 ticks, tree = 4932.76 MB, solutions = 20)

Nodefile size = 2828.82 MB (2551.19 MB after compression)

137960	116487	-316.4243	288	-44.0000	-375.5768	24985812	753.58%
138142	116067	-228.0601	154	-44.0000	-375.5768	24985266	753.58%
138227	115731	-362.3989	865	-44.0000	-375.5768	24979413	753.58%
138244	116283	-344.5858	239	-44.0000	-375.5768	24983061	753.58%
138307	115736	-359.8629	779	-44.0000	-375.5768	24983469	753.58%
138489	115938	-175.9894	129	-44.0000	-375.5768	25004586	753.58%
138590	115775	-255.4406	180	-44.0000	-375.5768	24988921	753.58%
138814	116345	-359.1724	760	-44.0000	-375.5768	24995243	753.58%
139074	115974	-105.0916	65	-44.0000	-375.5768	24992675	753.58%
139113	116128	-374.1367	970	-44.0000	-375.5768	24991767	753.58%

Elapsed time = 82.98 sec. (65890.40 ticks, tree = 4914.26 MB, solutions = 20)

Nodefile size = 2828.82 MB (2551.19 MB after compression)

139288	116450	cutoff		-44.0000	-375.5768	25002072	753.58%
--------	--------	--------	--	----------	-----------	----------	---------

139503	116586	-365.1134	478	-44.0000	-375.5768	25000336	753.58%
139743	116674	-294.4688	225	-44.0000	-375.5768	24997586	753.58%
139909	116748	-108.7938	89	-44.0000	-375.5768	24997959	753.58%
139971	116095	-335.3009	237	-44.0000	-375.5768	25004249	753.58%
140220	116203	-69.1186	46	-44.0000	-375.5768	25004973	753.58%
140240	116121	-362.5128	582	-44.0000	-375.5768	25007014	753.58%
140401	116148	-303.0626	212	-44.0000	-375.5768	25009101	753.58%
140517	116574	-348.2827	284	-44.0000	-375.5768	25019146	753.58%
140699	116276	-258.9445	196	-44.0000	-375.5768	25012673	753.58%

Elapsed time = 94.85 sec. (75581.51 ticks, tree = 4917.26 MB, solutions = 20)

Nodefile size = 2828.82 MB (2551.19 MB after compression)

140877	116078	-121.0623	82	-44.0000	-375.5768	25036048	753.58%
140925	116350	-343.0377	230	-44.0000	-375.5768	25017733	753.58%
141100	116151	-232.2341	177	-44.0000	-375.5768	25039954	753.58%
141253	117269	-234.1615	167	-44.0000	-375.5768	25034933	753.58%
141416	116174	-281.4930	181	-44.0000	-375.5768	25019843	753.58%
141745	116720	-364.4253	455	-44.0000	-375.5768	25034074	753.58%
141861	116802	-175.3061	289	-44.0000	-375.5768	25036560	753.58%
142147	116886	-280.5945	190	-44.0000	-375.5768	25034867	753.58%
142406	117437	-150.9468	103	-44.0000	-375.5768	25044378	753.58%
142489	116212	-373.2448	888	-44.0000	-375.5768	25048786	753.58%

Elapsed time = 106.60 sec. (85221.60 ticks, tree = 4909.58 MB, solutions = 20)

Nodefile size = 2828.82 MB (2551.19 MB after compression)

142508	117488	-336.2286	214	-44.0000	-375.5768	25048940	753.58%
142640	116225	-363.3862	631	-44.0000	-375.5768	25033233	753.58%
142775	116458	-157.0598	131	-44.0000	-375.5768	25036898	753.58%
142949	117590	-353.3226	361	-44.0000	-375.5768	25056638	753.58%
143076	116219	-361.4628	313	-44.0000	-375.5768	25060174	753.58%
143254	118022	-345.2241	249	-44.0000	-375.5297	25176661	753.48%
143691	116631	-45.6814	55	-44.0000	-375.5297	25047642	753.48%
143831	116896	-258.8147	156	-44.0000	-375.5297	25056621	753.48%
143932	117151	-370.4339	702	-44.0000	-375.5297	25050563	753.48%
144503	116351	-361.2385	431	-44.0000	-375.5297	25058610	753.48%

Elapsed time = 124.41 sec. (98029.26 ticks, tree = 4935.34 MB, solutions = 20)

Nodefile size = 2828.82 MB (2551.19 MB after compression)

145598	118282	-84.7192	45	-44.0000	-375.5297	25200709	753.48%
146286	116663	-146.1327	110	-44.0000	-375.5297	25086583	753.48%
146983	118364	-196.2860	147	-44.0000	-375.5297	25217258	753.48%
147695	116842	-288.8602	173	-44.0000	-375.5297	25106005	753.48%

148179	118682	-364.6394	681	-44.0000	-375.5297	25235746	753.48%
148895	118808	-45.9513	35	-44.0000	-375.5297	25239563	753.48%
149564	117128	-102.2161	125	-44.0000	-375.5297	25113787	753.48%
150155	117500	-250.4974	176	-44.0000	-375.5297	25113826	753.48%
151053	117401	-133.1193	72	-44.0000	-375.5297	25149498	753.48%
152268	117266	-74.6968	45	-44.0000	-375.5297	25133951	753.48%

Elapsed time = 171.10 sec. (136215.45 ticks, tree = 5020.68 MB, solutions = 20)

Nodefile size = 2828.82 MB (2551.19 MB after compression)

153265	117421	-131.6317	86	-44.0000	-375.5297	25148312	753.48%
153651	123366	-204.8835	135	-44.0000	-375.5297	25710075	753.48%
154291	127032	-273.1402	193	-44.0000	-375.4562	26185133	753.31%
154685	132699	-230.5562	138	-44.0000	-375.4562	26809062	753.31%
155410	135455	-86.7902	48	-44.0000	-375.0224	27113648	752.32%
156010	135713	-353.6035	369	-44.0000	-374.9424	27159569	752.14%
156428	136562	-373.1474	752	-44.0000	-374.9424	27240959	752.14%
156856	136991	-207.2019	119	-44.0000	-374.9424	27278840	752.14%
157325	137622	-282.8151	202	-44.0000	-374.6467	27376398	751.47%
157554	137746	-361.1068	365	-44.0000	-374.6467	27400626	751.47%

Elapsed time = 229.01 sec. (174835.59 ticks, tree = 6750.00 MB, solutions = 20)

Nodefile size = 4694.67 MB (4271.79 MB after compression)

158142	137958	-121.8675	82	-44.0000	-374.6467	27423219	751.47%
158452	138468	-76.5513	58	-44.0000	-374.3973	27479789	750.90%
158893	138456	-347.6235	270	-44.0000	-374.3973	27513832	750.90%
159308	139453	-83.8846	50	-44.0000	-374.3973	27639581	750.90%
159480	139089	-351.9897	348	-44.0000	-374.2840	27601366	750.65%
159932	140136	-288.8156	190	-44.0000	-374.2840	27773711	750.65%
160342	139970	-351.9906	293	-44.0000	-374.2840	27751169	750.65%
161057	140601	-366.0149	640	-44.0000	-374.2840	27867684	750.65%
162160	140791	-223.6580	127	-44.0000	-374.2101	27879510	750.48%
163097	142224	-222.4034	148	-44.0000	-374.1965	28026597	750.45%

Elapsed time = 278.81 sec. (213139.08 ticks, tree = 6770.20 MB, solutions = 20)

Nodefile size = 4714.67 MB (4281.11 MB after compression)

163912	142512	infeasible		-44.0000	-374.1965	28049351	750.45%
164581	143728	-93.8205	59	-44.0000	-374.0908	28152876	750.21%
165350	144119	-79.5072	47	-44.0000	-374.0698	28225664	750.16%
165945	144892	-229.1467	130	-44.0000	-374.0698	28296508	750.16%
166536	145100	-279.1562	190	-44.0000	-374.0698	28313847	750.16%
167263	145224	-248.0966	153	-44.0000	-374.0500	28324091	750.11%
167930	146403	-350.6970	275	-44.0000	-373.9883	28442310	749.97%

168968	146615	-100.5712	70	-44.0000	-373.9312	28450581	749.84%
169395	147938	-355.1343	366	-44.0000	-373.9312	28581082	749.84%
170326	148098	-257.4100	169	-44.0000	-373.9312	28588582	749.84%

Elapsed time = 328.74 sec. (251329.20 ticks, tree = 6859.09 MB, solutions = 20)

Nodefile size = 4790.43 MB (4339.89 MB after compression)

170842	148288	-349.1396	362	-44.0000	-373.9124	28607750	749.80%
171150	149503	-320.2788	218	-44.0000	-373.9124	28727565	749.80%
171832	149966	-350.6827	359	-44.0000	-373.8929	28794113	749.76%
172192	150174	-170.2027	122	-44.0000	-373.8929	28843720	749.76%
172580	149983	-359.4415	786	-44.0000	-373.8929	28820435	749.76%
172880	151136	-335.9286	284	-44.0000	-373.8929	28944854	749.76%
173494	151420	-241.2605	162	-44.0000	-373.8929	28978189	749.76%
173961	151599	-79.5849	38	-44.0000	-373.8929	29002456	749.76%
174586	152338	-361.9703	801	-44.0000	-373.8929	29088367	749.76%
175313	152491	-326.2307	218	-44.0000	-373.8929	29085315	749.76%

Elapsed time = 380.88 sec. (289676.74 ticks, tree = 7073.81 MB, solutions = 20)

Nodefile size = 4990.52 MB (4516.97 MB after compression)

176015	152689	-289.1985	182	-44.0000	-373.8929	29144336	749.76%
176855	154160	-77.4398	37	-44.0000	-373.8929	29266758	749.76%
177448	154194	-290.7869	193	-44.0000	-373.8929	29274335	749.76%
177800	155086	-352.6978	552	-44.0000	-373.8929	29417626	749.76%
178144	155126	-277.9166	175	-44.0000	-373.8929	29427343	749.76%
178488	155853	-325.1799	217	-44.0000	-373.6771	29539764	749.27%
178978	155690	-227.7448	269	-44.0000	-373.6771	29534376	749.27%
179678	156294	-172.5863	99	-44.0000	-373.6130	29606670	749.12%
180144	157102	-355.1944	657	-44.0000	-373.6099	29747844	749.11%
180869	157097	-338.7220	212	-44.0000	-373.5628	29743077	749.01%

Elapsed time = 433.30 sec. (328264.55 ticks, tree = 7285.22 MB, solutions = 20)

Nodefile size = 5230.30 MB (4731.06 MB after compression)

181719	157876	-303.7777	268	-44.0000	-373.5628	29865351	749.01%
182794	159272	-237.1395	147	-44.0000	-373.5628	29984307	749.01%
183348	159346	-351.1395	663	-44.0000	-373.5628	30006348	749.01%
184056	160314	-367.6314	687	-44.0000	-373.4833	30086392	748.83%
184762	160989	-99.2792	65	-44.0000	-373.4409	30149317	748.73%
185495	161553	-357.1583	641	-44.0000	-373.4409	30227053	748.73%
186459	162127	-156.6030	90	-44.0000	-373.4409	30284992	748.73%
187261	161936	-61.7549	32	-44.0000	-373.3481	30259262	748.52%
188114	162740	-73.0164	36	-44.0000	-373.3443	30325642	748.51%
189070	163956	-152.9141	96	-44.0000	-373.3443	30438865	748.51%

Elapsed time = 486.25 sec. (367163.04 ticks, tree = 7512.82 MB, solutions = 20)

Nodefile size = 5447.40 MB (4922.11 MB after compression)

189644	164386	-367.8376	887	-44.0000	-373.3443	30477271	748.51%
190685	164981	-87.2124	61	-44.0000	-373.3443	30514247	748.51%
191184	166418	-333.6920	221	-44.0000	-373.3443	30646990	748.51%
192458	166950	-208.3170	120	-44.0000	-373.3443	30699405	748.51%
193547	167529	-198.3154	118	-44.0000	-373.3443	30739547	748.51%
194774	168011	-168.6088	199	-44.0000	-373.2787	30794964	748.36%
195925	168908	-142.5090	148	-44.0000	-373.2427	30835364	748.28%
196620	171308	-224.7571	158	-44.0000	-373.2408	30997697	748.27%
197264	171376	-371.6623	716	-44.0000	-373.2408	31005841	748.27%
198034	171635	-208.1538	134	-44.0000	-373.2055	31055553	748.19%

Elapsed time = 539.69 sec. (405452.81 ticks, tree = 7961.62 MB, solutions = 20)

Nodefile size = 5907.38 MB (5338.30 MB after compression)

198412	172029	-289.5103	235	-44.0000	-373.2055	31095374	748.19%
198962	172723	-143.3916	95	-44.0000	-373.2055	31160733	748.19%
199414	172871	-308.1767	199	-44.0000	-373.2055	31188479	748.19%
200445	174277	-67.7922	28	-44.0000	-373.1191	31301581	748.00%
201225	174685	-365.1012	671	-44.0000	-373.1191	31351164	748.00%
201850	175086	-310.6747	205	-44.0000	-373.1191	31384774	748.00%
202420	176028	-299.8808	228	-44.0000	-373.1191	31500120	748.00%
202989	176150	-356.5293	345	-44.0000	-373.1191	31508469	748.00%
203966	177277	-349.3249	252	-44.0000	-373.1191	31616486	748.00%
204804	177320	-351.0572	257	-44.0000	-373.0455	31638301	747.83%

Elapsed time = 590.71 sec. (444013.14 ticks, tree = 8114.88 MB, solutions = 20)

Nodefile size = 6058.27 MB (5468.92 MB after compression)

205628	178179	-111.3682	81	-44.0000	-373.0143	31717251	747.76%
206144	178259	-197.6408	123	-44.0000	-373.0143	31727454	747.76%
206619	178812	-178.9469	189	-44.0000	-372.9949	31789311	747.72%
207249	179762	-363.1710	818	-44.0000	-372.9899	31871857	747.70%
207745	180890	-212.8827	131	-44.0000	-372.9899	32040405	747.70%
208953	181296	-86.9530	56	-44.0000	-372.9899	32078113	747.70%
209931	181837	-93.0117	192	-44.0000	-372.9480	32163706	747.61%
210844	182022	-195.7135	130	-44.0000	-372.9480	32157809	747.61%
211479	182394	-336.0640	243	-44.0000	-372.8824	32214463	747.46%
211879	183270	-333.1627	209	-44.0000	-372.8748	32262683	747.44%

Elapsed time = 646.29 sec. (482797.42 ticks, tree = 8350.07 MB, solutions = 20)

Nodefile size = 6294.05 MB (5672.22 MB after compression)

212147	184598	-363.4645	1062	-44.0000	-372.8748	32370505	747.44%
--------	--------	-----------	------	----------	-----------	----------	---------

212919	185068	-288.2498	184	-44.0000	-372.8748	32431657	747.44%
214028	185646	-210.1929	142	-44.0000	-372.8748	32487431	747.44%
214899	185756	-200.4004	137	-44.0000	-372.8600	32475142	747.41%
215323	186568	-146.7879	101	-44.0000	-372.8563	32572067	747.40%
215585	186646	-265.0594	174	-44.0000	-372.8430	32582774	747.37%
215930	186866	-296.3604	233	-44.0000	-372.8430	32646715	747.37%
216342	188005	-368.3260	1030	-44.0000	-372.8430	32726916	747.37%
216864	188736	-183.1399	110	-44.0000	-372.8430	32848795	747.37%
217090	189090	-331.5462	245	-44.0000	-372.8430	32903193	747.37%

Elapsed time = 697.47 sec. (521071.74 ticks, tree = 8599.97 MB, solutions = 20)

Nodefile size = 6544.44 MB (5890.61 MB after compression)

217790	188686	-323.9812	247	-44.0000	-372.8014	32870258	747.28%
218356	189541	-133.4137	76	-44.0000	-372.7773	32936831	747.22%
219174	190551	-351.0221	300	-44.0000	-372.7773	33038249	747.22%
219578	190300	-368.7349	934	-44.0000	-372.7684	33027649	747.20%
219598	190825	-355.7343	627	-44.0000	-372.7631	33100093	747.19%
219718	191769	-139.0102	86	-44.0000	-372.7377	33251637	747.13%
220184	191764	-131.8059	121	-44.0000	-372.7377	33230928	747.13%
221011	192191	-217.5934	136	-44.0000	-372.7272	33365455	747.11%
221677	192658	-312.0576	207	-44.0000	-372.7272	33417939	747.11%
222408	192605	-131.1298	82	-44.0000	-372.6994	33406887	747.04%

Elapsed time = 751.00 sec. (560245.75 ticks, tree = 8875.72 MB, solutions = 20)

Nodefile size = 6819.28 MB (6143.16 MB after compression)

222807	193252	-351.8520	359	-44.0000	-372.6994	33474914	747.04%
223697	193819	-368.2402	1014	-44.0000	-372.6705	33555864	746.98%
224291	194508	-168.4572	113	-44.0000	-372.6705	33597439	746.98%
224688	194980	-328.9159	330	-44.0000	-372.6705	33680048	746.98%
225604	195221	-350.0778	362	-44.0000	-372.6431	33710002	746.92%
226659	196294	-277.8216	183	-44.0000	-372.6029	33779087	746.82%
227209	195824	-306.6847	195	-44.0000	-372.6029	33766707	746.82%
228433	197223	-367.8317	689	-44.0000	-372.5677	33854373	746.74%
229269	199080	-361.1116	655	-44.0000	-372.5534	34022057	746.71%
230336	199294	-175.2973	112	-44.0000	-372.5531	34046870	746.71%

Elapsed time = 804.93 sec. (599059.75 ticks, tree = 9113.38 MB, solutions = 20)

Nodefile size = 7058.01 MB (6351.40 MB after compression)

231216	200588	-358.3446	489	-44.0000	-372.5531	34129659	746.71%
231583	200216	-268.3678	255	-44.0000	-372.5531	34115070	746.71%
232305	201798	-46.5832	26	-44.0000	-372.5531	34219507	746.71%
233137	202370	-185.0193	120	-44.0000	-372.5031	34307455	746.60%

234418	201984	-225.0413	192	-44.0000	-372.4962	34280003	746.58%
235315	203798	-343.5671	300	-44.0000	-372.4823	34427805	746.55%
235540	203201	-194.1673	132	-44.0000	-372.4823	34396450	746.55%
236303	204300	-90.3678	54	-44.0000	-372.4732	34472118	746.53%
236745	204491	-353.6942	922	-44.0000	-372.4732	34519696	746.53%
237725	205433	-362.9104	718	-44.0000	-372.4587	34591948	746.50%

Elapsed time = 858.28 sec. (637821.53 ticks, tree = 9338.06 MB, solutions = 20)

Nodefile size = 7269.93 MB (6536.27 MB after compression)

238274	205925	-85.8155	50	-44.0000	-372.4587	34647302	746.50%
238981	207487	-231.9881	149	-44.0000	-372.3994	34779028	746.36%
239878	207765	-93.4593	61	-44.0000	-372.3994	34786500	746.36%
240120	207756	-365.9616	411	-44.0000	-372.3994	34822268	746.36%
240478	209365	-199.8041	130	-44.0000	-372.3584	34940305	746.27%
240821	209539	-360.1367	638	-44.0000	-372.3584	34972410	746.27%
241271	209541	-363.1588	1122	-44.0000	-372.3584	34983161	746.27%
241424	210053	-361.2374	950	-44.0000	-372.3584	35084084	746.27%
241576	210292	-347.5688	297	-44.0000	-372.3584	35112261	746.27%
241997	210422	-368.2257	1017	-44.0000	-372.3584	35145629	746.27%

Elapsed time = 912.88 sec. (676882.31 ticks, tree = 9709.04 MB, solutions = 20)

Nodefile size = 7655.42 MB (6887.57 MB after compression)

242245	210584	-240.9517	157	-44.0000	-372.3584	35171767	746.27%
242570	210687	-338.0566	350	-44.0000	-372.3584	35275535	746.27%
243231	210809	-332.0380	238	-44.0000	-372.2883	35284325	746.11%
243525	211377	-366.4540	682	-44.0000	-372.2883	35363030	746.11%
243864	211549	-332.6650	319	-44.0000	-372.2883	35413331	746.11%
244343	212034	-284.4207	206	-44.0000	-372.2542	35461002	746.03%
244916	211665	-371.1470	1019	-44.0000	-372.2542	35425781	746.03%
245241	213063	-339.6566	564	-44.0000	-372.2542	35635322	746.03%
245969	213431	-346.9792	377	-44.0000	-372.2542	35676293	746.03%
246514	214170	-356.2338	419	-44.0000	-372.2542	35731859	746.03%

Elapsed time = 966.08 sec. (716168.10 ticks, tree = 9727.48 MB, solutions = 20)

Nodefile size = 7670.55 MB (6894.27 MB after compression)

247304	214362	-248.7154	181	-44.0000	-372.1984	35793183	745.91%
247544	214195	-368.2592	1074	-44.0000	-372.1984	35773514	745.91%
248320	215215	-339.6080	224	-44.0000	-372.1984	35912774	745.91%
248943	215127	-219.3915	150	-44.0000	-372.1939	35887771	745.90%
249564	215696	-329.7974	226	-44.0000	-372.1939	35961847	745.90%
250132	216976	-264.5542	167	-44.0000	-372.1916	36040514	745.89%
250395	216722	-370.6489	959	-44.0000	-372.1916	36031786	745.89%

Performing restart 2

Repeating presolve.

Tried aggregator 1 time.

Reduced MIP has 3556 rows, 5459 columns, and 23781 nonzeros.

Reduced MIP has 4603 binaries, 51 generals, 0 SOSs, and 0 indicators.

Presolve time = 0.01 sec. (12.05 ticks)

Tried aggregator 1 time.

Reduced MIP has 3556 rows, 5459 columns, and 23781 nonzeros.

Reduced MIP has 4603 binaries, 51 generals, 0 SOSs, and 0 indicators.

Presolve time = 0.02 sec. (16.72 ticks)

Represolve time = 1.99 sec. (423.12 ticks)

250594	0	-385.6923	1361	-44.0000	Cuts: 281 36368218	745.89%
250594	0	-385.5894	1250	-44.0000	Cuts: 88 36373095	745.89%
250594	0	-385.5320	1275	-44.0000	Cuts: 631 36377740	745.89%
250594	0	-385.4713	1249	-44.0000	Cuts: 545 36381282	745.89%
250594	0	-385.4204	1259	-44.0000	Cuts: 957 36387163	745.89%
250594	0	-385.3847	1222	-44.0000	Cuts: 654 36390902	745.89%
250594	0	-385.3577	1237	-44.0000	Cuts: 790 36394642	745.89%
250594	0	-385.3485	1245	-44.0000	Cuts: 703 36396546	745.89%
250594	0	-385.3400	1274	-44.0000	Cuts: 658 36398676	745.89%
250594	2	-385.3400	1250	-44.0000	-372.1916 36398676	745.89%
250597	5	-382.4444	787	-44.0000	-372.1916 36411834	745.89%
250602	9	-380.3559	797	-44.0000	-372.1916 36423958	745.89%

Elapsed time = 1108.28 sec. (855683.12 ticks, tree = 0.02 MB, solutions = 20)

250611	6	-378.8050	657	-44.0000	-372.1916 36417553	745.89%
250635	35	-376.0446	672	-44.0000	-372.1916 36531095	745.89%
250672	68	-370.5626	763	-44.0000	-372.1916 36616128	745.89%
250694	92	-371.9434	943	-44.0000	-372.1916 36701820	745.89%
250721	118	-362.8932	629	-44.0000	-372.1916 36801968	745.89%
250761	125	-361.9080	779	-44.0000	-372.1916 36885878	745.89%
250789	184	-368.8459	821	-44.0000	-372.1916 37004980	745.89%
250841	217	-339.4235	586	-44.0000	-372.1916 37066500	745.89%
251153	486	-210.0063	143	-44.0000	-372.1916 37133536	745.89%
251322	495	-364.4991	902	-44.0000	-372.1916 37206131	745.89%

Elapsed time = 1152.80 sec. (895288.35 ticks, tree = 14.88 MB, solutions = 20)

251351	672	-367.3628	827	-44.0000	-372.1916 37251757	745.89%
251394	715	-364.1056	761	-44.0000	-372.1916 37328615	745.89%

251440	744	-268.2901	507	-44.0000	-372.1916	37368491	745.89%
251521	826	-318.6446	749	-44.0000	-372.1916	37511071	745.89%
251833	831	-324.7711	304	-44.0000	-372.1916	37519578	745.89%
252407	1017	-289.7608	602	-44.0000	-372.1916	37636368	745.89%
252474	1569	-230.7063	364	-44.0000	-372.1916	37760627	745.89%
252551	1641	-175.7780	280	-44.0000	-372.1916	37815845	745.89%
252648	1761	-350.1193	690	-44.0000	-372.1916	37915832	745.89%
252770	1798	-297.9704	428	-44.0000	-372.1916	37931390	745.89%

Elapsed time = 1202.91 sec. (933976.92 ticks, tree = 72.71 MB, solutions = 20)

252957	1879	-365.0530	907	-44.0000	-372.1916	38019591	745.89%
253190	1961	-374.1327	746	-44.0000	-372.1916	38105061	745.89%
253209	2081	-366.6955	727	-44.0000	-372.1916	38154827	745.89%
253245	2107	-349.4859	750	-44.0000	-372.1916	38232352	745.89%
253296	2321	-323.2255	719	-44.0000	-372.1916	38306497	745.89%
253335	2350	-338.7238	810	-44.0000	-372.1916	38399274	745.89%
253431	2430	-223.8642	266	-44.0000	-372.1916	38441630	745.89%
253627	2511	-375.7077	894	-44.0000	-372.1916	38536324	745.89%
253710	2600	-267.8143	648	-44.0000	-372.1916	38634838	745.89%
253859	2807	-130.4952	121	-44.0000	-372.1916	38706683	745.89%

Elapsed time = 1250.04 sec. (972969.48 ticks, tree = 103.61 MB, solutions = 20)

253990	2793	-354.1329	866	-44.0000	-372.1916	38772365	745.89%
254125	2847	-378.1667	932	-44.0000	-372.1916	38821293	745.89%
254160	3057	-345.8537	710	-44.0000	-372.1916	38991676	745.89%
254272	3174	-338.3096	397	-44.0000	-372.1916	39057856	745.89%
254829	3402	-355.6737	381	-44.0000	-372.1916	39162730	745.89%
255039	3352	-378.1796	878	-44.0000	-372.1916	39140890	745.89%
255759	3966	-377.3962	848	-44.0000	-372.1916	39289058	745.89%
256132	4683	-376.0871	913	-44.0000	-372.1916	39411724	745.89%
256571	4897	-110.1847	135	-44.0000	-372.1916	39448784	745.89%
256899	5006	-264.7572	206	-44.0000	-372.1916	39493263	745.89%

Elapsed time = 1299.80 sec. (1011923.48 ticks, tree = 185.99 MB, solutions = 20)

257633	5503	-108.6815	304	-44.0000	-372.1916	39551337	745.89%
257704	5875	-357.2719	386	-44.0000	-372.1916	39627575	745.89%
258222	6596	-269.8295	197	-44.0000	-372.1916	39752633	745.89%
258860	6676	-377.5047	795	-44.0000	-372.1916	39771980	745.89%
259177	6867	-342.8244	368	-44.0000	-372.1916	39847164	745.89%
259604	7816	-126.7714	273	-44.0000	-372.1916	39962898	745.89%
259992	8066	-346.3133	285	-44.0000	-372.1916	40043650	745.89%
260634	8324	-250.4738	183	-44.0000	-372.1916	40101264	745.89%

261423	8587	-117.3785	72	-44.0000	-372.1916	40156633	745.89%
261811	8623	cutoff		-44.0000	-372.1916	40168545	745.89%
Elapsed time = 1345.55 sec. (1050174.00 ticks, tree = 360.71 MB, solutions = 20)							
262463	9727	-346.0636	360	-44.0000	-372.1916	40305990	745.89%
262827	10015	-53.0047	53	-44.0000	-372.1916	40336812	745.89%
263497	10204	-291.6292	329	-44.0000	-372.1916	40409727	745.89%
263810	10834	-368.7024	900	-44.0000	-372.1916	40508382	745.89%
264034	11380	-299.6250	270	-44.0000	-372.1916	40589629	745.89%
264353	11408	-333.6570	364	-44.0000	-372.1916	40631990	745.89%
265209	11627	-127.5949	81	-44.0000	-372.1916	40690974	745.89%
265375	12058	-373.9214	775	-44.0000	-372.1916	40825316	745.89%
265683	12067	-374.6869	871	-44.0000	-372.1916	40813199	745.89%
266145	12871	-108.7038	66	-44.0000	-372.1916	40958008	745.89%
Elapsed time = 1392.76 sec. (1089307.50 ticks, tree = 376.25 MB, solutions = 20)							
267009	13160	-339.9349	292	-44.0000	-372.1916	41042304	745.89%
267684	13451	-235.2024	155	-44.0000	-372.1916	41068056	745.89%
268135	14202	-301.8026	239	-44.0000	-372.1916	41179259	745.89%
269063	14802	-332.9360	201	-44.0000	-372.1916	41233945	745.89%
269908	15031	-151.4007	185	-44.0000	-372.1916	41298728	745.89%
270417	15902	-355.9540	424	-44.0000	-372.1916	41379785	745.89%
271179	15998	-182.1055	124	-44.0000	-372.1916	41374578	745.89%
271618	16779	-343.4644	297	-44.0000	-372.1916	41483634	745.89%
272154	17272	-266.0548	154	-44.0000	-372.1916	41561427	745.89%
272757	17853	-107.8635	54	-44.0000	-372.1916	41620379	745.89%
Elapsed time = 1441.86 sec. (1127546.03 ticks, tree = 531.02 MB, solutions = 20)							
273371	18302	-282.6906	198	-44.0000	-372.1916	41702255	745.89%
274130	18595	-244.6982	252	-44.0000	-372.1916	41749235	745.89%
274538	18916	-352.0270	368	-44.0000	-372.1916	41810597	745.89%
275533	19350	-242.7152	146	-44.0000	-372.1916	41846033	745.89%
276032	19990	-54.5599	49	-44.0000	-372.1916	41908637	745.89%
276526	20374	-115.2920	72	-44.0000	-372.1916	41972219	745.89%
277076	21354	-362.0541	532	-44.0000	-372.1916	42106109	745.89%
277609	20404	-368.4605	1077	-44.0000	-372.1916	41979761	745.89%
278059	22108	-337.7553	273	-44.0000	-372.1916	42232415	745.89%
278647	22119	-320.3121	220	-44.0000	-372.1916	42214941	745.89%
Elapsed time = 1491.47 sec. (1166019.37 ticks, tree = 615.31 MB, solutions = 20)							
279632	23591	-334.9430	280	-44.0000	-372.1916	42392818	745.89%
280126	23804	-128.5027	69	-44.0000	-372.1916	42401404	745.89%
280920	23955	-138.4359	120	-44.0000	-372.1916	42477580	745.89%

281437	24950	-129.1561	99	-44.0000	-372.1916	42575431	745.89%
281989	25082	-134.7455	71	-44.0000	-372.1916	42585611	745.89%
282711	25369	-371.6460	532	-44.0000	-372.1916	42618193	745.89%
283322	26147	-343.3787	304	-44.0000	-372.1916	42729297	745.89%
284664	26732	-334.3227	329	-44.0000	-372.1916	42818857	745.89%
285866	26665	-363.2964	860	-44.0000	-372.1916	42775391	745.89%
286482	28580	-247.8636	167	-44.0000	-372.1916	42941939	745.89%
Elapsed time = 1539.98 sec. (1204311.03 ticks, tree = 860.28 MB, solutions = 20)							
287505	28813	-299.1811	243	-44.0000	-372.0951	42963649	745.67%
288523	29633	-250.3703	162	-44.0000	-372.0110	43033224	745.48%
289106	30046	-295.2430	201	-44.0000	-372.0110	43069346	745.48%
289455	30190	-192.9309	112	-44.0000	-372.0040	43074509	745.46%
289993	31045	-141.5293	173	-44.0000	-372.0040	43163423	745.46%
290206	31157	-348.8130	419	-44.0000	-371.9483	43202763	745.34%
290699	31651	-341.1091	335	-44.0000	-371.9469	43228780	745.33%
291317	32466	-341.1582	315	-44.0000	-371.9469	43373635	745.33%
291837	32743	-352.7835	540	-44.0000	-371.6923	43438115	744.76%
292068	32468	-351.9463	516	-44.0000	-371.6923	43398040	744.76%
Elapsed time = 1589.93 sec. (1242794.09 ticks, tree = 1076.94 MB, solutions = 20)							
292671	33486	-239.0993	208	-44.0000	-371.6923	43533295	744.76%
292992	33266	-363.6098	582	-44.0000	-371.6923	43504347	744.76%
293493	33938	-118.9946	101	-44.0000	-371.5623	43606836	744.46%
293974	34166	-353.3337	727	-44.0000	-371.5062	43668357	744.33%
294519	34909	-280.2416	186	-44.0000	-371.5062	43776237	744.33%
295353	35538	-189.7857	111	-44.0000	-371.2609	43849416	743.77%
295790	35885	-52.6185	40	-44.0000	-371.2083	43882368	743.66%
296289	35883	-314.7165	258	-44.0000	-371.2083	43920204	743.66%
296578	35201	-354.8640	645	-44.0000	-371.2083	43857528	743.66%
296765	36460	-354.9554	528	-44.0000	-371.2083	44019420	743.66%
Elapsed time = 1642.27 sec. (1284198.61 ticks, tree = 1170.56 MB, solutions = 20)							
296907	36344	-353.1478	725	-44.0000	-371.2083	44017102	743.66%
297436	37190	-175.1358	102	-44.0000	-371.2083	44086601	743.66%
297836	37240	-360.3308	492	-44.0000	-371.0292	44096222	743.25%
298225	37776	-368.8773	710	-44.0000	-371.0292	44237544	743.25%
298876	37904	-76.2539	71	-44.0000	-371.0292	44259745	743.25%
299088	38497	-366.8987	596	-44.0000	-371.0292	44375456	743.25%
299492	38089	-358.5517	265	-44.0000	-370.9771	44293705	743.13%
300346	38897	-311.8971	282	-44.0000	-370.9771	44428265	743.13%
300919	39134	-357.5479	819	-44.0000	-370.9771	44454488	743.13%

301172	39614	-319.6664	202	-44.0000	-370.8041	44571989	742.74%
Elapsed time = 1691.83 sec. (1323114.08 ticks, tree = 1204.52 MB, solutions = 20)							
301646	39448	-344.0955	491	-44.0000	-370.8041	44557156	742.74%
301836	40335	-103.6051	83	-44.0000	-370.8041	44707703	742.74%
302225	40202	-358.3735	487	-44.0000	-370.8041	44704677	742.74%
302610	40600	-355.9007	952	-44.0000	-370.8041	44777613	742.74%
302988	41309	-298.4826	193	-44.0000	-370.6520	44872708	742.39%
303374	40766	-283.5381	167	-44.0000	-370.6520	44825751	742.39%
303891	41485	-304.3835	215	-44.0000	-370.6520	44908146	742.39%
304356	41571	-74.9176	60	-44.0000	-370.6109	44934370	742.30%
304928	42264	-364.8487	443	-44.0000	-370.4962	45035283	742.04%
305462	42397	-359.8320	501	-44.0000	-370.4951	45041146	742.03%
Elapsed time = 1741.50 sec. (1361578.50 ticks, tree = 1299.40 MB, solutions = 20)							
305990	43081	-87.9339	46	-44.0000	-370.4951	45139238	742.03%
306368	43162	-215.9043	149	-44.0000	-370.4950	45145034	742.03%
307093	43334	-343.6828	419	-44.0000	-370.3971	45170187	741.81%
307363	44228	-335.0437	293	-44.0000	-370.3971	45262113	741.81%
307994	44774	-209.7137	119	-44.0000	-370.3477	45337005	741.70%
308082	44372	-360.5717	1018	-44.0000	-370.3477	45292973	741.70%
308313	44758	-359.0325	970	-44.0000	-370.3477	45372677	741.70%
308796	45146	-329.1411	285	-44.0000	-370.2988	45421832	741.59%
309247	45738	-85.4863	51	-44.0000	-370.2988	45556907	741.59%
309734	46020	-367.5729	426	-44.0000	-370.2988	45588630	741.59%
Elapsed time = 1792.12 sec. (1401098.88 ticks, tree = 1498.55 MB, solutions = 20)							
310298	46140	-357.3188	1014	-44.0000	-370.1730	45686049	741.30%
310880	46723	-313.9365	262	-44.0000	-370.0978	45733688	741.13%
311618	47314	-351.6886	376	-44.0000	-370.0978	45815752	741.13%
312071	47781	-268.8162	238	-44.0000	-370.0978	45873219	741.13%
312444	47505	-244.1141	152	-44.0000	-370.0978	45846117	741.13%
313037	48077	-223.3538	252	-44.0000	-370.0978	45934014	741.13%
313489	48766	-231.3842	174	-44.0000	-370.0978	45982195	741.13%
314106	49110	-286.7892	161	-44.0000	-369.8680	46027720	740.61%
314277	49227	-352.6875	410	-44.0000	-369.7946	46049462	740.44%
315052	49499	-356.5130	502	-44.0000	-369.7890	46093606	740.43%
Elapsed time = 1844.27 sec. (1439867.53 ticks, tree = 1727.94 MB, solutions = 20)							
315430	50490	infeasible		-44.0000	-369.7665	46227867	740.38%
315633	50873	-116.4103	66	-44.0000	-369.7325	46279007	740.30%
316395	50950	-319.2967	221	-44.0000	-369.7325	46305582	740.30%
317079	51065	-360.8404	415	-44.0000	-369.6889	46312413	740.20%

317677	51183	-365.7037	1061	-44.0000	-369.6668	46348889	740.15%
318344	51962	-125.6950	92	-44.0000	-369.6668	46450552	740.15%
318572	51990	-368.0280	694	-44.0000	-369.6179	46457249	740.04%
318765	52695	-357.1016	1077	-44.0000	-369.5690	46541748	739.93%
319126	52963	-352.8065	686	-44.0000	-369.5690	46586748	739.93%
319440	53430	-260.1408	161	-44.0000	-369.5690	46663375	739.93%
Elapsed time = 1896.37 sec. (1479194.25 ticks, tree = 1951.39 MB, solutions = 20)							
320144	53982	-234.4145	165	-44.0000	-369.5411	46755129	739.87%
320530	53857	-207.5893	175	-44.0000	-369.5411	46726581	739.87%
320617	53926	-344.9031	756	-44.0000	-369.5411	46789304	739.87%
321019	54333	-331.2943	278	-44.0000	-369.5411	46876025	739.87%
321384	54728	-330.9338	393	-44.0000	-369.4818	46956127	739.73%
Began writing nodes to disk (directory ./cpx6hXQcQ created)							
321929	55231	-205.4632	119	-44.0000	-369.4596	47068125	739.68%
322419	55580	-324.8614	213	-44.0000	-369.4596	47114860	739.68%
322789	55897	-364.3820	405	-44.0000	-369.3454	47171636	739.42%
323522	56158	-364.5998	889	-44.0000	-369.3454	47201561	739.42%
323533	56166	-341.7744	430	-44.0000	-369.3454	47208791	739.42%
Elapsed time = 1948.33 sec. (1518462.54 ticks, tree = 2111.35 MB, solutions = 20)							
Nodefile size = 58.62 MB (51.63 MB after compression)							
323942	57003	-96.3349	92	-44.0000	-369.3155	47332465	739.35%
324164	57093	-340.3201	295	-44.0000	-369.3155	47380106	739.35%
324358	57091	-342.8492	664	-44.0000	-369.3155	47399665	739.35%
324721	57452	-332.6673	256	-44.0000	-369.3155	47485341	739.35%
325593	57417	-80.5629	105	-44.0000	-369.2700	47458323	739.25%
325870	57756	-344.9406	339	-44.0000	-369.2662	47566012	739.24%
326635	58368	-235.1431	177	-44.0000	-369.2522	47647462	739.21%
326735	58498	-310.1259	223	-44.0000	-369.2522	47686593	739.21%
327241	58889	-332.4587	307	-44.0000	-369.2522	47744689	739.21%
328157	58732	-348.2701	337	-44.0000	-369.2483	47714755	739.20%
Elapsed time = 2000.08 sec. (1556813.49 ticks, tree = 2191.10 MB, solutions = 20)							
Nodefile size = 138.10 MB (121.57 MB after compression)							
328696	59366	-349.9339	591	-44.0000	-369.2483	47866301	739.20%
329258	59928	-200.9429	112	-44.0000	-369.2483	47903916	739.20%
330085	59816	-360.8325	274	-44.0000	-369.2169	47888319	739.13%
330696	60569	-251.8809	167	-44.0000	-369.1513	47944891	738.98%
331215	61510	-350.2743	253	-44.0000	-369.1513	48067414	738.98%
331643	61684	-252.4282	148	-44.0000	-369.1169	48074972	738.90%
332328	62542	-300.0368	269	-44.0000	-369.1169	48153942	738.90%

333052	62613	-172.4415	137	-44.0000	-369.1169	48170227	738.90%
333503	63456	-273.7816	225	-44.0000	-369.0920	48254726	738.85%
334319	64313	-129.4382	75	-44.0000	-369.0066	48319508	738.65%

Elapsed time = 2053.05 sec. (1595009.26 ticks, tree = 2609.90 MB, solutions = 20)

Nodefile size = 546.80 MB (482.47 MB after compression)

334526	64366	-353.2900	430	-44.0000	-369.0066	48353456	738.65%
335096	64236	-348.8633	383	-44.0000	-369.0066	48342110	738.65%
335482	64995	-357.2174	248	-44.0000	-369.0066	48410627	738.65%
335751	65112	-108.3985	85	-44.0000	-369.0066	48422213	738.65%
336342	65566	-342.3519	298	-44.0000	-369.0066	48508876	738.65%
336872	65367	-278.6137	218	-44.0000	-369.0030	48496547	738.64%
337666	66284	-241.4814	163	-44.0000	-369.0030	48624088	738.64%
338557	67101	-71.7748	33	-44.0000	-368.7571	48737948	738.08%
339421	67550	-280.2961	200	-44.0000	-368.7571	48788027	738.08%
340094	67778	-72.4334	62	-44.0000	-368.7571	48792340	738.08%

Elapsed time = 2105.71 sec. (1633306.25 ticks, tree = 2773.62 MB, solutions = 20)

Nodefile size = 694.67 MB (612.65 MB after compression)

340708	68252	-89.3731	41	-44.0000	-368.7571	48849031	738.08%
341145	68828	-235.1402	140	-44.0000	-368.7364	48907267	738.04%
342133	69534	-227.6095	155	-44.0000	-368.6620	49008465	737.87%
343080	69560	cutoff		-44.0000	-368.6620	48981554	737.87%
343436	70905	-359.4257	669	-44.0000	-368.6480	49119605	737.84%
344064	69862	-316.9504	228	-44.0000	-368.6321	49055558	737.80%
344917	71337	-358.6754	317	-44.0000	-368.6321	49158258	737.80%
345459	71450	-241.4689	183	-44.0000	-368.5854	49178803	737.69%
345806	72770	-350.6402	414	-44.0000	-368.5854	49313002	737.69%
346180	72776	-344.2606	682	-44.0000	-368.5854	49334729	737.69%

Elapsed time = 2158.28 sec. (1671605.97 ticks, tree = 3247.77 MB, solutions = 20)

Nodefile size = 1193.67 MB (1058.69 MB after compression)

346629	73045	-306.0790	188	-44.0000	-368.5714	49381248	737.66%
346989	73311	-326.8352	274	-44.0000	-368.5714	49410860	737.66%
347251	73656	-159.0263	107	-44.0000	-368.5025	49428784	737.51%
347944	74202	-186.4601	103	-44.0000	-368.4952	49535435	737.49%
348859	74293	-334.4883	288	-44.0000	-368.4952	49557222	737.49%
349336	74834	-85.2232	153	-44.0000	-368.4952	49628825	737.49%
349792	75367	-246.7459	197	-44.0000	-368.4952	49673306	737.49%
350409	75634	-214.3057	148	-44.0000	-368.4689	49693562	737.43%
351125	76103	-65.2717	23	-44.0000	-368.4689	49729789	737.43%
351528	76502	-121.4386	73	-44.0000	-368.4689	49803180	737.43%

Elapsed time = 2212.26 sec. (1709831.49 ticks, tree = 3429.06 MB, solutions = 20)

Nodefile size = 1374.50 MB (1216.81 MB after compression)

351877 76695	-210.5933	124	-44.0000	-368.3886	49818966	737.25%
*352083+76771			-46.0000	-368.3886		700.84%
352358 77214	-203.4235	147	-46.0000	-368.3886	49867602	700.84%
352954 77661	-354.4451	635	-46.0000	-368.3848	49922905	700.84%

GUB cover cuts applied: 1479

Clique cuts applied: 53

Cover cuts applied: 4469

Implied bound cuts applied: 115

Flow cuts applied: 171

Mixed integer rounding cuts applied: 2859

Zero-half cuts applied: 135

Lift and project cuts applied: 20

Gomory fractional cuts applied: 182

Root node processing (before b&c):

Real time = 0.00 sec. (2.63 ticks)

Parallel b&c, 8 threads:

Real time = 2233.60 sec. (1733502.38 ticks)

Sync time (average) = 300.51 sec.

Wait time (average) = 0.08 sec.

Total (root+branch&cut) = 2233.60 sec. (1733505.01 ticks)

Iteration 10

Bounds on # of cuts = 8 with [3 3 2]

Error = 54 (out of 100 instances)

Accuracy = 46

Solving time = 37.2267415 min (minutes)

Accumulated time = 68.199989864 min (minutes)

Solution status code = 104

LB on error = -268.366653275

Relative objective gap = 7.007970723

Selected variables:

A_AGE (Continuous)

PEMLR (Categorical)

Number of selected variables = 2 (1 continuous + 1 categorical)

Version identifier: 22.1.1.0 | 2022-11-28 | 9160aff4d

CPXPARAM_MIP_Strategy_File	3
CPXPARAM_MIP_Limits_Solutions	1
CPXPARAM_TimeLimit	82308.000608154296
CPXPARAM_MIP_Limits_TreeMemory	204800

Nodes		Cuts/					
Node	Left	Objective	IInf	Best Integer	Best Bound	ItCnt	Gap

352957 78439 infeasible -46.0000 -368.3667 50058943 700.80%

Elapsed time = 0.58 sec. (13.44 ticks, tree = 3578.62 MB, solutions = 21)

Nodefile size = 1531.15 MB (1354.94 MB after compression)

352959 78441	-355.0696	622	-46.0000	-368.3667	50059517	700.80%
352960 78442	-354.9621	614	-46.0000	-368.3667	50059978	700.80%
352961 78441	-368.1422	467	-46.0000	-368.3667	50059688	700.80%
352963 78442	-366.8929	459	-46.0000	-368.3667	50061062	700.80%
352970 78443	-364.4998	478	-46.0000	-368.3667	50063349	700.80%
352991 78459	-359.1808	287	-46.0000	-368.3667	50061782	700.80%
353012 78471	-341.2399	250	-46.0000	-368.3667	50062182	700.80%
353032 78481	-314.3891	241	-46.0000	-368.3667	50062713	700.80%
353075 78464	-352.9182	243	-46.0000	-368.3667	50065349	700.80%
353290 78522	-216.3510	139	-46.0000	-368.3667	50066815	700.80%

Elapsed time = 5.58 sec. (3453.73 ticks, tree = 3574.72 MB, solutions = 21)

Nodefile size = 1531.15 MB (1354.94 MB after compression)

353412 78478	-313.4469	240	-46.0000	-368.3065	50067622	700.67%
353582 78573	-120.4601	54	-46.0000	-368.3065	50068969	700.67%
353756 78583	-366.5572	569	-46.0000	-368.3065	50069592	700.67%
353915 78593	-350.4133	356	-46.0000	-368.3065	50071762	700.67%
354013 78638	-255.5095	182	-46.0000	-368.3065	50073098	700.67%
354172 78670	-225.7101	139	-46.0000	-368.3065	50075904	700.67%
354385 78504	-226.4890	127	-46.0000	-368.2929	50093543	700.64%
354717 78838	-68.3427	35	-46.0000	-368.2929	50076924	700.64%
354895 78735	-366.5307	385	-46.0000	-368.2929	50081410	700.64%
355013 78725	-358.3571	260	-46.0000	-368.2929	50081358	700.64%

Elapsed time = 18.04 sec. (13079.83 ticks, tree = 3596.11 MB, solutions = 21)

Nodefile size = 1531.15 MB (1354.94 MB after compression)

355350	78840	-106.1301	61	-46.0000	-368.2929	50082313	700.64%
355421	78637	-265.3412	275	-46.0000	-368.2929	50101647	700.64%
355525	78674	-185.2515	141	-46.0000	-368.2802	50103370	700.61%
*355558+78900				-47.0000	-368.2802		683.57%
355620	78594	-339.7121	378	-47.0000	-368.2802	50098468	683.57%
355766	78770	-294.4367	166	-47.0000	-368.2802	50106522	683.57%
356139	78635	-175.6360	123	-47.0000	-368.2802	50098267	683.57%
356248	78595	-327.9391	290	-47.0000	-368.2802	50116071	683.57%
356341	78649	-216.7224	142	-47.0000	-368.2802	50118193	683.57%
356578	78816	-170.0356	106	-47.0000	-368.2802	50108680	683.57%
356638	78453	-344.8623	449	-47.0000	-368.2802	50116537	683.57%

Elapsed time = 29.93 sec. (23092.66 ticks, tree = 3533.95 MB, solutions = 24)

Nodefile size = 1531.15 MB (1354.94 MB after compression)

356671	78466	-341.4183	357	-47.0000	-368.2802	50117958	683.57%
356751	78750	-256.5897	150	-47.0000	-368.2802	50111786	683.57%
356870	78469	-366.3548	433	-47.0000	-368.2802	50121648	683.57%
356949	78525	-255.7729	163	-47.0000	-368.2802	50124119	683.57%
357102	78851	-349.7279	232	-47.0000	-368.2802	50117120	683.57%
*357191+78873				-48.0000	-368.2802		667.25%
357191	78711	-367.6334	1100	-48.0000	-368.2802	50124118	667.25%
357194	78714	-367.6270	1104	-48.0000	-368.2802	50124781	667.25%
357195	78609	-342.4514	413	-48.0000	-368.2802	50139710	667.25%
357262	78665	-248.5754	138	-48.0000	-368.2802	50141684	667.25%
357380	78718	-354.8980	803	-48.0000	-368.2802	50142887	667.25%

Elapsed time = 44.43 sec. (43023.32 ticks, tree = 3542.02 MB, solutions = 27)

Nodefile size = 1531.15 MB (1354.94 MB after compression)

357385	78720	-354.1313	745	-48.0000	-368.2802	50144502	667.25%
357389	78723	-352.5103	493	-48.0000	-368.2802	50146723	667.25%
357394	78727	-339.5890	283	-48.0000	-368.2802	50148761	667.25%
357420	78744	-332.3197	331	-48.0000	-368.2802	50149654	667.25%
357451	78769	-280.8813	168	-48.0000	-368.2802	50150722	667.25%
357488	78795	-215.9856	158	-48.0000	-368.2802	50151746	667.25%
357527	78746	-354.7789	1031	-48.0000	-368.2802	50162814	667.25%
357530	78749	-354.6699	1031	-48.0000	-368.2802	50163649	667.25%
357531	78750	-346.2256	653	-48.0000	-368.2802	50170449	667.25%
357533	78752	-344.5490	744	-48.0000	-368.2802	50171923	667.25%

Elapsed time = 55.57 sec. (57967.17 ticks, tree = 3535.25 MB, solutions = 27)

Nodefile size = 1531.15 MB (1354.94 MB after compression)

357536	78821	-366.0749	1161	-48.0000	-368.2802	50157991	667.25%
357540	78823	-366.0599	1158	-48.0000	-368.2802	50158619	667.25%
357556	78769	-325.5245	302	-48.0000	-368.2802	50177346	667.25%
357590	78792	-291.4648	232	-48.0000	-368.2802	50178499	667.25%
357621	78817	-233.7738	198	-48.0000	-368.2802	50179962	667.25%
357636	78824	-365.1710	595	-48.0000	-368.2802	50183101	667.25%
357669	78849	-340.1584	284	-48.0000	-368.2802	50185185	667.25%
357722	78826	-361.9055	1039	-48.0000	-368.2802	50168111	667.25%
357723	78827	-354.3601	752	-48.0000	-368.2802	50176673	667.25%
357725	78829	-352.8350	745	-48.0000	-368.2802	50178651	667.25%

Elapsed time = 63.73 sec. (73838.94 ticks, tree = 3542.48 MB, solutions = 28)

Nodefile size = 1531.15 MB (1354.94 MB after compression)

357727	78831	-351.3868	663	-48.0000	-368.2802	50180601	667.25%
357729	78833	-349.5185	509	-48.0000	-368.2802	50182598	667.25%
357733	78835	-339.0805	430	-48.0000	-368.2802	50184554	667.25%
357757	78857	-321.4356	295	-48.0000	-368.2802	50185975	667.25%
357781	78864	-367.4109	559	-48.0000	-368.2802	50189238	667.25%
357788	78868	-362.7362	469	-48.0000	-368.2802	50191101	667.25%
357803	78879	-356.6581	374	-48.0000	-368.2802	50192864	667.25%

GUB cover cuts applied: 1515

Clique cuts applied: 53

Cover cuts applied: 4487

Implied bound cuts applied: 116

Flow cuts applied: 171

Mixed integer rounding cuts applied: 3009

Zero-half cuts applied: 135

Lift and project cuts applied: 20

Gomory fractional cuts applied: 183

Root node processing (before b&c):

Real time = 0.00 sec. (1.95 ticks)

Parallel b&c, 8 threads:

Real time = 70.85 sec. (82977.94 ticks)

Sync time (average) = 1.52 sec.

Wait time (average) = 0.00 sec.

Total (root+branch&cut) = 70.85 sec. (82979.89 ticks)

Iteration 11

Bounds on # of cuts = 8 with [3 3 2]

Error = 52 (out of 100 instances)

Accuracy = 48

Solving time = 1.180936951 min (minutes)

Accumulated time = 69.380926815 min (minutes)

Solution status code = 104

LB on error = -268.191364056

Relative objective gap = 6.670653418

Selected variables:

A_AGE (Continuous)

PEMLR (Categorical)

Number of selected variables = 2 (1 continuous + 1 categorical)

Version identifier: 22.1.1.0 | 2022-11-28 | 9160aff4d

CPXPARAM_MIP_Strategy_File	3
CPXPARAM_MIP_Limits_Solutions	1
CPXPARAM_TimeLimit	82237.144391113281
CPXPARAM_MIP_Limits_TreeMemory	204800

Nodes				Cuts/			
Node	Left	Objective	IInf	Best Integer	Best Bound	ItCnt	Gap
357813	81848	infeasible		-48.0000	-368.1914	50513898	667.07%
Elapsed time = 0.66 sec. (292.33 ticks, tree = 3726.48 MB, solutions = 29)							
Nodefile size = 1679.11 MB (1484.78 MB after compression)							
357826	81861	-292.1446	183	-48.0000	-368.1914	50513559	667.07%
357851	81885	-234.4599	151	-48.0000	-368.1914	50514204	667.07%
357887	81913	-152.0224	89	-48.0000	-368.1909	50514540	667.06%
357920	81854	-357.2824	437	-48.0000	-368.1909	50516152	667.06%
357957	81858	-352.0901	310	-48.0000	-368.1909	50519329	667.06%
358005	81876	-328.8730	208	-48.0000	-368.1909	50519839	667.06%
358061	81952	-365.6237	549	-48.0000	-368.1909	50516426	667.06%
358123	81944	-173.5007	110	-48.0000	-368.1909	50520624	667.06%

358187	81986	-63.5680	31	-48.0000	-368.1909	50520867	667.06%
358404	81869	-356.0177	335	-48.0000	-368.1843	50525639	667.05%

Elapsed time = 4.90 sec. (3429.12 ticks, tree = 3706.40 MB, solutions = 29)

Nodefile size = 1679.11 MB (1484.78 MB after compression)

358736	81977	-104.3151	66	-48.0000	-368.1843	50527782	667.05%
359077	81930	-213.6064	155	-48.0000	-368.1843	50539357	667.05%
359402	81958	-126.7440	131	-48.0000	-368.1843	50534882	667.05%
359558	82114	-323.8779	203	-48.0000	-368.1843	50527920	667.05%
360011	82123	-110.9767	62	-48.0000	-368.1843	50534255	667.05%
360279	82251	-303.4537	198	-48.0000	-368.1843	50531448	667.05%
360674	82270	-333.9872	256	-48.0000	-368.1843	50533165	667.05%
360773	82324	-216.5179	149	-48.0000	-368.1433	50535459	666.97%
360935	81876	-334.8739	394	-48.0000	-368.1433	50554007	666.97%
361126	82233	-126.4382	85	-48.0000	-368.1433	50538997	666.97%

Elapsed time = 16.84 sec. (13063.54 ticks, tree = 3721.75 MB, solutions = 29)

Nodefile size = 1679.11 MB (1484.78 MB after compression)

361224	82365	-356.3953	443	-48.0000	-368.1433	50542559	666.97%
361281	82131	-357.4859	303	-48.0000	-368.1433	50550840	666.97%
361426	82156	-326.2501	224	-48.0000	-368.1433	50553536	666.97%
361646	82220	-176.9277	120	-48.0000	-368.1433	50556001	666.97%
361770	82452	-180.0865	199	-48.0000	-368.1433	50554805	666.97%
361955	82503	-57.9050	56	-48.0000	-368.1433	50556571	666.97%
362344	82097	-142.5888	80	-48.0000	-368.1433	50569144	666.97%
362616	82168	-298.4494	201	-48.0000	-368.1433	50555327	666.97%
362791	82238	-120.0840	76	-48.0000	-368.1433	50557846	666.97%
363008	82410	-348.3597	333	-48.0000	-368.1433	50568317	666.97%

Elapsed time = 29.19 sec. (22675.86 ticks, tree = 3702.14 MB, solutions = 29)

Nodefile size = 1679.11 MB (1484.78 MB after compression)

363419	82370	-120.3047	72	-48.0000	-368.1433	50561564	666.97%
363578	82189	-249.4025	251	-48.0000	-368.1433	50581892	666.97%
363716	82216	-192.8838	220	-48.0000	-368.1433	50583993	666.97%
364045	82643	-113.9610	77	-48.0000	-368.1433	50563974	666.97%
364090	82681	-358.5332	390	-48.0000	-368.1433	50576810	666.97%
364122	82699	-341.6880	311	-48.0000	-368.1433	50578930	666.97%
364284	82291	-317.6356	231	-48.0000	-368.1433	50591814	666.97%
364631	82674	-357.5447	338	-48.0000	-368.1433	50572614	666.97%
364749	82708	-296.0979	196	-48.0000	-368.1433	50574904	666.97%
364984	82786	-87.9932	99	-48.0000	-368.1433	50575721	666.97%

Elapsed time = 41.83 sec. (32254.33 ticks, tree = 3717.44 MB, solutions = 29)

Nodefile size = 1679.11 MB (1484.78 MB after compression)

365223	82504	-191.5209	217	-48.0000	-368.1433	50591049	666.97%
365504	82549	-331.3050	208	-48.0000	-368.1433	50601365	666.97%
365992	82893	-127.0245	73	-48.0000	-368.1433	50588165	666.97%
366170	82544	-366.6969	623	-48.0000	-368.1433	50597281	666.97%
366176	82548	-355.5594	469	-48.0000	-368.1433	50601349	666.97%
366189	82559	-351.7782	427	-48.0000	-368.1433	50603729	666.97%
366204	82569	-339.4731	368	-48.0000	-368.1433	50607134	666.97%
366249	82599	-272.1125	282	-48.0000	-368.1433	50608232	666.97%
366396	82671	-114.3516	71	-48.0000	-368.1433	50609897	666.97%
366634	82736	-309.9270	209	-48.0000	-368.1433	50611201	666.97%

Elapsed time = 54.67 sec. (42460.70 ticks, tree = 3710.17 MB, solutions = 29)

Nodefile size = 1679.11 MB (1484.78 MB after compression)

366756	82821	-111.5596	54	-48.0000	-368.1433	50612395	666.97%
366807	82858	-343.9349	258	-48.0000	-368.1433	50613675	666.97%
366896	82914	-211.1564	174	-48.0000	-368.1433	50615760	666.97%
366985	82971	-366.4920	430	-48.0000	-368.1433	50617063	666.97%
367034	82400	-351.3165	559	-48.0000	-368.1433	50587667	666.97%
367120	83037	-244.8728	179	-48.0000	-368.1433	50620481	666.97%
367337	82703	-334.1119	233	-48.0000	-368.1433	50628475	666.97%
367706	82986	-316.4094	188	-48.0000	-368.1433	50617616	666.97%
367821	83126	-356.0752	340	-48.0000	-368.1433	50627458	666.97%
368044	82503	-135.1360	111	-48.0000	-368.1433	50593666	666.97%

Elapsed time = 67.00 sec. (52139.75 ticks, tree = 3695.91 MB, solutions = 29)

Nodefile size = 1679.11 MB (1484.78 MB after compression)

368238	82922	-164.7335	92	-48.0000	-368.1433	50634114	666.97%
368430	83224	-123.8483	141	-48.0000	-368.1433	50637485	666.97%
368773	83201	-128.9948	87	-48.0000	-368.1433	50629760	666.97%
368913	83264	-357.5110	311	-48.0000	-368.1433	50641918	666.97%
369098	83112	-324.3748	198	-48.0000	-368.1433	50639612	666.97%
369400	83329	-191.7452	204	-48.0000	-368.1433	50645762	666.97%
369572	83433	-223.7018	132	-48.0000	-368.1433	50636921	666.97%
369806	82380	-287.5304	169	-48.0000	-368.1433	50648326	666.97%
370042	83155	-230.0538	152	-48.0000	-368.1433	50627192	666.97%
370296	83206	-105.6802	84	-48.0000	-368.1433	50627997	666.97%

Elapsed time = 79.67 sec. (61734.30 ticks, tree = 3719.22 MB, solutions = 29)

Nodefile size = 1679.11 MB (1484.78 MB after compression)

370473	83648	-363.5100	330	-48.0000	-368.1433	50644687	666.97%
370572	83690	-301.8013	207	-48.0000	-368.1433	50648101	666.97%

370729	83483	-197.3347	130	-48.0000	-368.1387	50658662	666.96%
370929	83541	-365.7080	488	-48.0000	-368.1387	50660523	666.96%
371088	83221	-351.9229	387	-48.0000	-368.1387	50653084	666.96%
371487	83670	-73.6233	44	-48.0000	-368.1387	50663896	666.96%
371836	83297	-205.4547	120	-48.0000	-368.1387	50656731	666.96%
371955	84037	-363.3737	398	-48.0000	-368.1387	50658794	666.96%
372245	84150	-104.1935	62	-48.0000	-368.1387	50660639	666.96%
372599	84174	-356.4815	416	-48.0000	-368.1387	50662713	666.96%

Elapsed time = 91.72 sec. (71418.27 ticks, tree = 3710.01 MB, solutions = 29)

Nodefile size = 1679.11 MB (1484.78 MB after compression)

372758	83830	-350.2123	218	-48.0000	-368.1387	50673011	666.96%
372917	83910	-185.9764	100	-48.0000	-368.1337	50675434	666.95%
373114	82774	-318.4387	220	-48.0000	-368.1337	50678770	666.95%
373417	84320	-338.0692	224	-48.0000	-368.1337	50669801	666.95%
373666	83414	-320.7274	235	-48.0000	-368.1337	50655149	666.95%
374022	83475	-168.1748	126	-48.0000	-368.1337	50656810	666.95%
374273	84491	-280.5796	200	-48.0000	-368.1337	50676353	666.95%
374608	84074	-110.7467	74	-48.0000	-368.1337	50687542	666.95%
374870	83955	-200.9511	127	-48.0000	-368.1337	50773859	666.95%
375060	84635	-264.2193	176	-48.0000	-368.1337	50681732	666.95%

Elapsed time = 104.63 sec. (80997.62 ticks, tree = 3712.07 MB, solutions = 29)

Nodefile size = 1679.11 MB (1484.78 MB after compression)

375289	83674	-313.4418	201	-48.0000	-368.1337	50666202	666.95%
375652	83029	-311.7428	205	-48.0000	-368.1337	50695634	666.95%
375861	83121	-243.7553	181	-48.0000	-368.1337	50707677	666.95%
376042	83812	-322.3605	225	-48.0000	-368.1337	50672426	666.95%
376328	83907	-76.8829	55	-48.0000	-368.1337	50673938	666.95%
376562	84043	-315.9558	248	-48.0000	-368.1337	50786727	666.95%
376701	84097	-203.0215	121	-48.0000	-368.1337	50788420	666.95%
376892	83281	-94.0053	88	-48.0000	-368.1337	50707850	666.95%
376963	84416	-300.0476	169	-48.0000	-368.1337	50708399	666.95%
377680	83473	-295.0804	190	-48.0000	-368.1337	50717425	666.95%

Elapsed time = 120.42 sec. (93442.31 ticks, tree = 3693.86 MB, solutions = 29)

Nodefile size = 1679.11 MB (1484.78 MB after compression)

378589	84784	-62.4796	50	-48.0000	-368.1337	50723926	666.95%
379167	84588	-357.1111	256	-48.0000	-368.1295	50809279	666.94%
379898	84778	-210.9325	156	-48.0000	-368.1295	50814162	666.94%
381019	83619	-115.1233	68	-48.0000	-368.1295	50707029	666.94%
382720	83365	-140.2585	105	-48.0000	-368.1295	50688807	666.94%

384080	83309	-101.5742	51	-48.0000	-368.1295	50766747	666.94%
384963	83446	-105.2105	49	-48.0000	-368.1295	50774019	666.94%
386015	84519	-346.9758	297	-48.0000	-368.1265	50783476	666.93%
386910	85693	-155.1845	151	-48.0000	-368.1265	50775095	666.93%
387595	84934	-333.2850	217	-48.0000	-368.1265	50794129	666.93%

Elapsed time = 167.93 sec. (131619.11 ticks, tree = 3741.29 MB, solutions = 29)

Nodefile size = 1679.11 MB (1484.78 MB after compression)

388775	86007	-55.9050	38	-48.0000	-368.1265	50790936	666.93%
389667	89849	-58.8835	63	-48.0000	-368.1265	51217466	666.93%
390428	84259	-235.6584	138	-48.0000	-368.1265	50820389	666.93%
391907	85686	-200.5898	122	-48.0000	-368.1265	50817259	666.93%
393428	86618	-190.2522	146	-48.0000	-368.1265	51023481	666.93%
394894	85686	-338.0291	262	-48.0000	-368.1265	50892802	666.93%
396235	86317	-283.0987	189	-48.0000	-368.1265	50835383	666.93%
397505	86197	-70.7395	64	-48.0000	-368.1265	50901768	666.93%
398333	86550	-347.5027	457	-48.0000	-368.1265	50846264	666.93%
399245	86427	-177.6206	95	-48.0000	-368.1265	50911461	666.93%

Elapsed time = 217.63 sec. (169797.21 ticks, tree = 3979.15 MB, solutions = 29)

Nodefile size = 1679.11 MB (1484.78 MB after compression)

400219	86967	-365.8791	488	-48.0000	-368.1265	50855776	666.93%
401660	91267	-243.3707	155	-48.0000	-368.1265	51274192	666.93%
403296	87512	-341.1402	231	-48.0000	-368.1265	50866250	666.93%
404989	84983	-140.2182	86	-48.0000	-368.0929	50859564	666.86%
406525	88101	-184.9819	105	-48.0000	-368.0929	50876062	666.86%
407805	92086	-259.9844	159	-48.0000	-368.0929	51296163	666.86%
409441	86511	-258.7671	146	-48.0000	-368.0929	50913194	666.86%
411560	92832	-360.5152	318	-48.0000	-368.0929	51306048	666.86%
413551	93075	-71.0918	41	-48.0000	-368.0929	51311387	666.86%
415320	93351	-366.3324	416	-48.0000	-368.0856	51316812	666.84%

Elapsed time = 280.95 sec. (207992.61 ticks, tree = 4136.27 MB, solutions = 29)

Nodefile size = 1679.11 MB (1484.78 MB after compression)

416761	93635	-356.6959	307	-48.0000	-368.0856	51321961	666.84%
417641	103611	-333.2727	267	-48.0000	-368.0856	52115151	666.84%
418995	86717	-91.4659	93	-48.0000	-368.0856	50923115	666.84%
420375	95010	-345.7725	227	-48.0000	-368.0856	51592947	666.84%
422388	97745	-208.8593	113	-48.0000	-368.0856	51806281	666.84%
*424820+95327				-49.0000	-368.0856		651.20%
424875	86974	-117.5452	83	-49.0000	-368.0856	50941301	651.20%
426829	88296	-141.1596	108	-49.0000	-368.0856	51174273	651.20%

428842	87107	-81.6216	48	-49.0000	-368.0856	50956823	651.20%
430040	88694	-143.4552	113	-49.0000	-368.0856	51183846	651.20%
431654	88993	-56.4649	57	-49.0000	-368.0856	51187570	651.20%

Elapsed time = 330.56 sec. (246162.32 ticks, tree = 3912.30 MB, solutions = 30)

Nodefile size = 1679.11 MB (1484.78 MB after compression)

432487	89155	-309.7494	252	-49.0000	-368.0856	51192129	651.20%
433820	98871	-248.0751	169	-49.0000	-368.0856	51844744	651.20%
434923	89508	-95.4429	54	-49.0000	-368.0856	50996457	651.20%
436455	90671	-329.4717	245	-49.0000	-368.0856	51072363	651.20%
438281	90148	-118.6082	102	-49.0000	-368.0856	51006030	651.20%
439742	90497	-207.3705	167	-49.0000	-368.0856	51216254	651.20%
441567	90674	-173.8533	102	-49.0000	-368.0856	51016082	651.20%
443828	91216	-303.3454	206	-49.0000	-368.0856	51223635	651.20%
445236	87736	-175.7545	94	-49.0000	-368.0856	51043195	651.20%
447049	91633	-313.8024	195	-49.0000	-368.0856	51113941	651.20%

Elapsed time = 373.44 sec. (284339.53 ticks, tree = 4204.39 MB, solutions = 30)

Nodefile size = 1679.11 MB (1484.78 MB after compression)

447980	100629	-348.9983	289	-49.0000	-368.0856	51894673	651.20%
449086	92335	-127.1360	75	-49.0000	-368.0856	51244067	651.20%
449528	92362	-357.3279	1052	-49.0000	-368.0856	51255961	651.20%

GUB cover cuts applied: 1587

Clique cuts applied: 53

Cover cuts applied: 4561

Implied bound cuts applied: 116

Flow cuts applied: 178

Mixed integer rounding cuts applied: 3530

Zero-half cuts applied: 136

Lift and project cuts applied: 20

Gomory fractional cuts applied: 183

Root node processing (before b&c):

Real time = 0.00 sec. (2.15 ticks)

Parallel b&c, 8 threads:

Real time = 430.29 sec. (305295.57 ticks)

Sync time (average) = 41.19 sec.

Wait time (average) = 0.00 sec.

Total (root+branch&cut) = 430.29 sec. (305297.72 ticks)

Iteration 12

Bounds on # of cuts = 8 with [3 3 2]

Error = 51 (out of 100 instances)

Accuracy = 49

Solving time = 7.171601351 min (minutes)

Accumulated time = 76.552528166 min (minutes)

Solution status code = 104

LB on error = -267.975324274

Relative objective gap = 6.509700495

Selected variables:

A_AGE (Continuous)

PEMLR (Categorical)

Number of selected variables = 2 (1 continuous + 1 categorical)

Version identifier: 22.1.1.0 | 2022-11-28 | 9160aff4d

CPXPARAM_MIP_Strategy_File	3
CPXPARAM_MIP_Limits_Solutions	1
CPXPARAM_TimeLimit	81806.848310058587
CPXPARAM_MIP_Limits_TreeMemory	204800

Nodes				Cuts/			
Node	Left	Objective	IInf	Best Integer	Best Bound	ItCnt	Gap
449529	148551	infeasible		-49.0000	-367.9753	54370598	650.97%
Elapsed time = 0.47 sec. (14.98 ticks, tree = 8107.53 MB, solutions = 30)							
Nodefile size = 6060.88 MB (5290.89 MB after compression)							
449531	148553	-359.5659	442	-49.0000	-367.9753	54371140	650.97%
449538	148551	infeasible		-49.0000	-367.9753	54371316	650.97%
449555	148565	-356.7244	318	-49.0000	-367.9753	54371964	650.97%
449593	148577	-332.7352	218	-49.0000	-367.9753	54373247	650.97%
449639	148602	-280.7380	198	-49.0000	-367.9753	54373435	650.97%
449702	148606	-260.2627	157	-49.0000	-367.9753	54373161	650.97%
449786	148649	-166.4203	107	-49.0000	-367.9753	54373940	650.97%
449880	148675	-94.9784	61	-49.0000	-367.9753	54374029	650.97%

449971	148638	-203.4032	129	-49.0000	-367.9753	54375734	650.97%
450059	148698	-366.2099	443	-49.0000	-367.9753	54375106	650.97%

Elapsed time = 5.04 sec. (3168.00 ticks, tree = 8124.01 MB, solutions = 30)

Nodefile size = 6060.88 MB (5290.89 MB after compression)

450236	148594	-295.4153	184	-49.0000	-367.9753	54382327	650.97%
450577	148682	-63.8978	27	-49.0000	-367.9753	54385978	650.97%
450591	148559	-360.5034	415	-49.0000	-367.8207	54398193	650.65%
450738	148908	-173.4438	111	-49.0000	-367.8207	54383205	650.65%
450900	148662	-146.4658	101	-49.0000	-367.8207	54401032	650.65%
451012	148976	-333.3717	213	-49.0000	-367.8207	54385969	650.65%
451296	148560	-363.7896	978	-49.0000	-367.6854	54392980	650.38%
451342	149129	-304.8566	181	-49.0000	-367.6854	54388385	650.38%
451481	148830	-361.6500	361	-49.0000	-367.6854	54396382	650.38%
451603	148878	-283.0614	180	-49.0000	-367.6854	54397899	650.38%

Elapsed time = 17.82 sec. (12954.41 ticks, tree = 8108.39 MB, solutions = 30)

Nodefile size = 6060.88 MB (5290.89 MB after compression)

451826	148953	-82.9050	50	-49.0000	-367.6854	54398543	650.38%
451879	148704	-340.0657	313	-49.0000	-367.6854	54393842	650.38%
451961	149414	-296.1621	239	-49.0000	-367.6854	54395518	650.38%
452167	148607	-276.8603	214	-49.0000	-367.6854	54403955	650.38%
452428	149031	-219.7767	147	-49.0000	-367.6854	54406146	650.38%
452701	148652	-365.3107	433	-49.0000	-367.6854	54406178	650.38%
453008	148875	-265.3827	181	-49.0000	-367.6854	54403380	650.38%
453289	149184	-154.6563	92	-49.0000	-367.6854	54411487	650.38%
453348	148960	-360.0414	372	-49.0000	-367.6854	54406637	650.38%
453502	149046	-148.5781	74	-49.0000	-367.6854	54407919	650.38%

Elapsed time = 30.74 sec. (22572.71 ticks, tree = 8146.19 MB, solutions = 30)

Nodefile size = 6060.88 MB (5290.89 MB after compression)

453759	148853	-256.7948	204	-49.0000	-367.6854	54415433	650.38%
453922	148937	-60.9847	87	-49.0000	-367.6854	54416165	650.38%
454060	149366	-363.9560	517	-49.0000	-367.6854	54420697	650.38%
*454067+148947				-50.0000	-367.6854		635.37%
454103	149399	-314.6991	203	-50.0000	-367.6854	54422606	635.37%
454237	149500	-56.4004	29	-50.0000	-367.6854	54423429	635.37%
454247	148825	-356.6694	887	-50.0000	-367.6854	54420679	635.37%
454277	148835	-290.6358	185	-50.0000	-367.6854	54422588	635.37%
454414	148923	-67.9049	38	-50.0000	-367.6854	54423303	635.37%
454472	148930	-363.1861	506	-50.0000	-367.6854	54424890	635.37%
454545	149615	-91.8766	82	-50.0000	-367.6854	54431081	635.37%

Elapsed time = 43.16 sec. (32219.69 ticks, tree = 8192.55 MB, solutions = 31)

Nodefile size = 6060.88 MB (5290.89 MB after compression)

454566	149628	-364.7786	412	-50.0000	-367.6854	54432926	635.37%
454641	149685	-257.6134	147	-50.0000	-367.6854	54434532	635.37%
454765	149232	-338.9140	319	-50.0000	-367.6854	54425621	635.37%
454818	149263	-264.9478	165	-50.0000	-367.6854	54427210	635.37%
455006	148623	-232.9872	171	-50.0000	-367.6854	54453546	635.37%
455113	148698	-365.3190	406	-50.0000	-367.6854	54454977	635.37%
455205	148741	-298.9809	232	-50.0000	-367.6854	54456259	635.37%
455387	149424	-232.1075	128	-50.0000	-367.6854	54435736	635.37%
455507	148735	-264.2029	170	-50.0000	-367.6854	54427522	635.37%
455726	149833	-203.7847	140	-50.0000	-367.6854	54447876	635.37%

Elapsed time = 54.10 sec. (41946.00 ticks, tree = 8217.64 MB, solutions = 32)

Nodefile size = 6060.88 MB (5290.89 MB after compression)

455921	149884	infeasible		-50.0000	-367.6854	54452377	635.37%
455956	149909	-340.9434	221	-50.0000	-367.6854	54453965	635.37%
456118	148703	-322.7887	332	-50.0000	-367.6854	54448276	635.37%
456211	150028	-361.5807	311	-50.0000	-367.6854	54457854	635.37%
456551	150147	-90.9050	48	-50.0000	-367.6854	54458849	635.37%
456710	150218	-244.2240	146	-50.0000	-367.6854	54460397	635.37%
456937	150290	-365.6604	620	-50.0000	-367.6854	54461963	635.37%
457104	150332	-301.2754	178	-50.0000	-367.6854	54463828	635.37%
457325	149253	-317.2975	220	-50.0000	-367.6854	54475276	635.37%
457456	149351	-66.9050	55	-50.0000	-367.6854	54475951	635.37%

Elapsed time = 68.03 sec. (53360.20 ticks, tree = 8134.99 MB, solutions = 33)

Nodefile size = 6060.88 MB (5290.89 MB after compression)

457550	149392	-299.9797	180	-50.0000	-367.6854	54477445	635.37%
457686	148828	-349.3764	432	-50.0000	-367.6854	54460618	635.37%
457810	148869	-286.1150	166	-50.0000	-367.6854	54462183	635.37%
458021	148956	-366.7730	1041	-50.0000	-367.6854	54465966	635.37%
458023	148958	-365.0671	968	-50.0000	-367.6854	54471615	635.37%
458024	148959	-363.9372	1032	-50.0000	-367.6854	54476003	635.37%
458026	148959	infeasible		-50.0000	-367.6854	54482253	635.37%
458028	149619	-355.0212	967	-50.0000	-367.6854	54505490	635.37%
458029	149620	-354.8428	952	-50.0000	-367.6854	54509299	635.37%
458031	148960	-361.1371	1077	-50.0000	-367.6854	54490873	635.37%

Elapsed time = 75.01 sec. (76144.16 ticks, tree = 8099.72 MB, solutions = 33)

Nodefile size = 6060.88 MB (5290.89 MB after compression)

458035	149623	-338.0811	587	-50.0000	-367.6854	54515789	635.37%
--------	--------	-----------	-----	----------	-----------	----------	---------

458072	149654	-292.3925	206	-50.0000	-367.6854	54517524	635.37%
458195	149745	-366.2947	371	-50.0000	-367.6854	54518192	635.37%
458268	149801	-265.0918	206	-50.0000	-367.6854	54519721	635.37%
458362	148969	-339.0310	358	-50.0000	-367.6854	54499208	635.37%
458394	148993	-294.8755	208	-50.0000	-367.6854	54501358	635.37%
458507	149871	-352.6250	646	-50.0000	-367.6854	54535166	635.37%
458509	149873	-348.5649	612	-50.0000	-367.6854	54536702	635.37%
458514	149878	-344.9235	428	-50.0000	-367.6854	54539075	635.37%
458536	149895	-297.6408	238	-50.0000	-367.6854	54541041	635.37%

Elapsed time = 88.09 sec. (94500.64 ticks, tree = 8197.13 MB, solutions = 33)

Nodefile size = 6060.88 MB (5290.89 MB after compression)

458680	149994	-364.9267	362	-50.0000	-367.6854	54542471	635.37%
458801	149115	-328.4475	206	-50.0000	-367.6854	54526877	635.37%
459008	150143	-344.6698	288	-50.0000	-367.6854	54544620	635.37%
459201	150256	-366.0108	502	-50.0000	-367.6854	54545677	635.37%
459262	150301	-288.3912	191	-50.0000	-367.6854	54547706	635.37%
459384	149217	cutoff		-50.0000	-367.6854	54536299	635.37%
459407	150400	-355.7017	347	-50.0000	-367.6854	54550553	635.37%
459462	150443	-285.5415	194	-50.0000	-367.6854	54552575	635.37%
459552	150512	-80.8691	63	-50.0000	-367.6854	54553583	635.37%
459612	150559	-306.0035	185	-50.0000	-367.6854	54555246	635.37%

Elapsed time = 95.25 sec. (104655.04 ticks, tree = 8273.17 MB, solutions = 33)

Nodefile size = 6060.88 MB (5290.89 MB after compression)

459757	150659	-364.0543	363	-50.0000	-367.6854	54556163	635.37%
460012	150786	-365.0957	367	-50.0000	-367.6854	54557236	635.37%
460166	150852	-257.4208	157	-50.0000	-367.6854	54558763	635.37%
460271	150928	-364.7451	440	-50.0000	-367.6854	54559507	635.37%
460315	150961	-322.3830	231	-50.0000	-367.6854	54560849	635.37%
460444	151056	-365.3064	533	-50.0000	-367.6854	54562520	635.37%
460460	151063	-349.8631	236	-50.0000	-367.6854	54564641	635.37%
460548	151134	-168.2526	125	-50.0000	-367.6854	54565692	635.37%
460596	149345	-353.9889	773	-50.0000	-367.6854	54557078	635.37%
460634	149374	-285.4250	215	-50.0000	-367.6854	54563745	635.37%

Elapsed time = 104.38 sec. (117804.09 ticks, tree = 8133.27 MB, solutions = 33)

Nodefile size = 6060.88 MB (5290.89 MB after compression)

460972	151200	-309.7938	276	-50.0000	-367.6854	54579043	635.37%
460986	149605	-354.4270	858	-50.0000	-367.6854	54576765	635.37%
461156	149721	-353.2440	644	-50.0000	-367.6854	54594267	635.37%
461200	149747	-319.7503	298	-50.0000	-367.6854	54601578	635.37%

461459	149970	-140.8536	83	-50.0000	-367.6854	54607404	635.37%
461869	150271	-362.2076	370	-50.0000	-367.6854	54611721	635.37%
462205	150520	-348.8006	380	-50.0000	-367.6854	54618159	635.37%
462684	150874	-141.8029	82	-50.0000	-367.6854	54623422	635.37%
462904	151045	-365.6643	568	-50.0000	-367.6854	54630449	635.37%
463109	151192	-349.4642	263	-50.0000	-367.6854	54638262	635.37%

Elapsed time = 136.21 sec. (163970.28 ticks, tree = 8327.89 MB, solutions = 34)

Nodefile size = 6060.88 MB (5290.89 MB after compression)

GUB cover cuts applied: 1636

Clique cuts applied: 53

Cover cuts applied: 4620

Implied bound cuts applied: 117

Flow cuts applied: 182

Mixed integer rounding cuts applied: 3798

Zero-half cuts applied: 137

Lift and project cuts applied: 21

Gomory fractional cuts applied: 183

Root node processing (before b&c):

Real time = 0.00 sec. (2.40 ticks)

Parallel b&c, 8 threads:

Real time = 139.89 sec. (165844.88 ticks)

Sync time (average) = 3.07 sec.

Wait time (average) = 0.00 sec.

Total (root+branch&cut) = 139.89 sec. (165847.28 ticks)

Iteration 13

Bounds on # of cuts = 8 with [3 3 2]

Error = 50 (out of 100 instances)

Accuracy = 50

Solving time = 2.331538167 min (minutes)

Accumulated time = 78.884066333 min (minutes)

Solution status code = 104

LB on error = -267.498135184

Relative objective gap = 6.349962704

Selected variables:

A_AGE (Continuous)

PEMLR (Categorical)

Number of selected variables = 2 (1 continuous + 1 categorical)

Version identifier: 22.1.1.0 | 2022-11-28 | 9160aff4d

CPXPARAM_MIP_Strategy_File	3
CPXPARAM_MIP_Limits_Solutions	1
CPXPARAM_TimeLimit	81666.956020019526
CPXPARAM_MIP_Limits_TreeMemory	204800

Nodes		Cuts/					
Node	Left	Objective	IInf	Best Integer	Best Bound	ItCnt	Gap
463135	158775	infeasible		-50.0000	-367.4981	55198048	635.00%
Elapsed time = 0.72 sec. (15.17 ticks, tree = 9214.61 MB, solutions = 35)							
Nodefile size = 7167.73 MB (6279.59 MB after compression)							
463136	158777	-366.5895	490	-50.0000	-367.4981	55198583	635.00%
463140	158777	-366.5176	642	-50.0000	-367.4981	55198925	635.00%
463162	158789	-348.1658	230	-50.0000	-367.4981	55199716	635.00%
463191	158808	-305.6908	217	-50.0000	-367.4981	55200151	635.00%
463231	158839	-228.0707	125	-50.0000	-367.4981	55200362	635.00%
463277	158868	-155.1797	87	-50.0000	-367.4981	55200885	635.00%
463346	158903	-73.9050	40	-50.0000	-367.4981	55201169	635.00%
463386	158807	-332.2260	227	-50.0000	-367.4981	55203386	635.00%
463443	158821	-309.8742	204	-50.0000	-367.4981	55202374	635.00%
463885	158912	-57.9050	30	-50.0000	-367.4981	55206450	635.00%
Elapsed time = 4.78 sec. (3333.40 ticks, tree = 9185.11 MB, solutions = 35)							
Nodefile size = 7167.73 MB (6279.59 MB after compression)							
463977	158951	-311.4922	247	-50.0000	-367.4981	55207360	635.00%
464319	159014	-129.0656	74	-50.0000	-367.4981	55211007	635.00%
464400	159039	-365.9692	556	-50.0000	-367.4981	55214011	635.00%
464435	158914	-363.0478	611	-50.0000	-367.4981	55217781	635.00%
464654	158929	-332.7367	352	-50.0000	-367.4981	55213624	635.00%
464900	158970	-238.0384	158	-50.0000	-367.3236	55215938	634.65%
465154	159315	-57.9050	43	-50.0000	-367.3236	55221298	634.65%
465181	159068	-320.7091	311	-50.0000	-367.3236	55222588	634.65%

465201	158920	-354.1757	744	-50.0000	-367.3236	55218060	634.65%
465256	159111	-246.3611	193	-50.0000	-367.3236	55228446	634.65%

Elapsed time = 18.42 sec. (13757.59 ticks, tree = 9195.97 MB, solutions = 35)

Nodefile size = 7167.73 MB (6279.59 MB after compression)

465358	159084	-314.4182	271	-50.0000	-367.3236	55226164	634.65%
465456	159131	-187.6688	114	-50.0000	-367.3236	55228695	634.65%
465661	158957	-277.9212	163	-50.0000	-367.3236	55224025	634.65%
466060	159341	-322.8122	208	-50.0000	-367.3236	55231554	634.65%
466373	159214	-320.7883	207	-50.0000	-367.3236	55238808	634.65%
466710	159450	-363.6421	406	-50.0000	-367.3236	55235082	634.65%
467092	159174	-82.9050	46	-50.0000	-367.2274	55234189	634.45%
467517	159450	-80.0285	38	-50.0000	-367.2274	55243721	634.45%
467643	159181	-361.5120	332	-50.0000	-367.2274	55237667	634.45%
467857	159282	-137.5505	86	-50.0000	-367.2274	55239206	634.45%

Elapsed time = 30.58 sec. (23322.97 ticks, tree = 9206.18 MB, solutions = 35)

Nodefile size = 7167.73 MB (6279.59 MB after compression)

468131	159579	-98.6959	54	-50.0000	-367.2274	55248828	634.45%
468495	159424	-138.5116	92	-50.0000	-367.2274	55251419	634.45%
468860	159757	-299.5947	187	-50.0000	-367.2274	55249968	634.45%
469263	159717	-102.4586	55	-50.0000	-367.2274	55254216	634.45%
469473	159677	-150.4201	83	-50.0000	-367.2274	55251829	634.45%
469748	159820	-174.4052	108	-50.0000	-367.2274	55257568	634.45%
470173	159481	-323.6713	253	-50.0000	-367.2274	55252810	634.45%
470415	158856	-199.7805	124	-50.0000	-367.2274	55276184	634.45%
470751	159959	-175.7107	99	-50.0000	-367.2274	55263101	634.45%
471015	159827	-151.8560	89	-50.0000	-367.2274	55267815	634.45%

Elapsed time = 42.21 sec. (32903.35 ticks, tree = 9179.86 MB, solutions = 35)

Nodefile size = 7167.73 MB (6279.59 MB after compression)

471249	160104	-129.7341	76	-50.0000	-367.2274	55266272	634.45%
471728	159717	cutoff		-50.0000	-367.2274	55262790	634.45%
472043	158873	-146.7093	92	-50.0000	-367.2274	55269576	634.45%
472112	159045	-359.1444	299	-50.0000	-367.2274	55288895	634.45%
472255	159107	-255.2068	161	-50.0000	-367.2274	55290947	634.45%
472607	160018	-340.2480	214	-50.0000	-367.2274	55279581	634.45%
473049	160115	-107.4381	59	-50.0000	-367.2274	55281245	634.45%
473256	160127	-327.8756	214	-50.0000	-367.2274	55275796	634.45%
473588	159992	-67.9050	32	-50.0000	-367.2274	55278335	634.45%
473847	159971	-349.1251	284	-50.0000	-367.2274	55276168	634.45%

Elapsed time = 53.92 sec. (42468.69 ticks, tree = 9219.60 MB, solutions = 35)

Nodefile size = 7167.73 MB (6279.59 MB after compression)

*473936+160266				-51.0000	-367.2274	620.05%
474206 160073	-106.9895	61		-51.0000	-367.2274 55277413	620.05%
474463 160635	-153.2612	94		-51.0000	-367.2274 55286212	620.05%
474720 160190	-231.7403	139		-51.0000	-367.2274 55285078	620.05%
474998 160214	-188.9340	119		-51.0000	-367.2274 55293399	620.05%
475115 158909	-351.2163	434		-51.0000	-367.2274 55286348	620.05%
475182 160297	-322.9094	208		-51.0000	-367.2274 55297449	620.05%
475430 159398	-178.7548	109		-51.0000	-367.2274 55314927	620.05%
475645 160241	-343.1800	338		-51.0000	-367.2274 55290066	620.05%
475689 159453	-352.9342	244		-51.0000	-367.2274 55318299	620.05%
475854 159546	-132.8329	74		-51.0000	-367.2274 55319648	620.05%

Elapsed time = 65.35 sec. (52182.49 ticks, tree = 9149.24 MB, solutions = 37)

Nodefile size = 7167.73 MB (6279.59 MB after compression)

476031 160718	-240.2005	152		-51.0000	-367.2274 55306433	620.05%
476142 159580	-362.2827	332		-51.0000	-367.2274 55324358	620.05%
476249 160829	-310.6718	209		-51.0000	-367.2274 55309938	620.05%
476542 159639	-255.0189	186		-51.0000	-367.2274 55330168	620.05%
476682 159724	-364.8578	329		-51.0000	-367.2274 55332953	620.05%
476863 159830	-123.2319	72		-51.0000	-367.2274 55334306	620.05%
477115 159915	-264.5046	167		-51.0000	-367.2274 55335820	620.05%
477332 159993	-363.7070	330		-51.0000	-367.2274 55337385	620.05%
477539 160005	-346.0468	269		-51.0000	-367.2274 55340473	620.05%
477743 160033	-289.9295	263		-51.0000	-367.2274 55343355	620.05%

Elapsed time = 76.10 sec. (61944.06 ticks, tree = 9151.21 MB, solutions = 37)

Nodefile size = 7167.73 MB (6279.59 MB after compression)

478067 160528	-58.4099	23		-51.0000	-367.2274 55320438	620.05%
478272 161231	-353.4015	234		-51.0000	-367.2274 55327993	620.05%
478528 161319	-127.3948	80		-51.0000	-367.2274 55330121	620.05%
478842 160780	-107.5241	73		-51.0000	-367.2274 55324854	620.05%
479065 160251	-364.3253	339		-51.0000	-367.2274 55353325	620.05%
479151 160321	-214.6277	139		-51.0000	-367.2274 55354988	620.05%
479262 160382	-363.8436	407		-51.0000	-367.2274 55356639	620.05%
479450 159282	-67.9050	30		-51.0000	-367.2274 55326580	620.05%
479573 160516	-80.8811	56		-51.0000	-367.2274 55359174	620.05%
479698 159343	-257.4662	179		-51.0000	-367.2274 55329998	620.05%

Elapsed time = 87.26 sec. (71569.74 ticks, tree = 9167.39 MB, solutions = 37)

Nodefile size = 7167.73 MB (6279.59 MB after compression)

479968 159410	-78.9050	42		-51.0000	-367.2274 55331365	620.05%
---------------	----------	----	--	----------	--------------------	---------

480111	160658	-364.4200	444	-51.0000	-367.2274	55364600	620.05%
480282	159204	-331.5965	250	-51.0000	-367.2274	55319373	620.05%
480731	161741	-101.9345	64	-51.0000	-367.2274	55352367	620.05%
480977	160912	-99.4795	61	-51.0000	-367.2274	55369544	620.05%
481190	161873	-98.8474	60	-51.0000	-367.2274	55355666	620.05%
481381	159573	-347.1083	234	-51.0000	-367.2274	55341855	620.05%
481707	160908	-133.3562	68	-51.0000	-367.2274	55346119	620.05%
481897	160938	-363.3551	308	-51.0000	-367.2274	55348000	620.05%
481964	162036	-343.0073	212	-51.0000	-367.2274	55364517	620.05%

Elapsed time = 98.33 sec. (81154.62 ticks, tree = 9209.33 MB, solutions = 37)

Nodefile size = 7167.73 MB (6279.59 MB after compression)

482104	162103	-209.0701	120	-51.0000	-367.2274	55366952	620.05%
482251	161057	-106.7135	99	-51.0000	-367.2274	55353243	620.05%
482354	162205	-281.0030	193	-51.0000	-367.2274	55370235	620.05%
482515	161109	-330.1069	237	-51.0000	-367.2274	55357746	620.05%
482708	162411	-81.9050	43	-51.0000	-367.2274	55372148	620.05%
482820	162458	-269.5278	203	-51.0000	-367.2274	55374343	620.05%
482988	161088	-309.6921	433	-51.0000	-367.2274	55399414	620.05%
483039	162556	-348.2095	215	-51.0000	-367.2274	55378317	620.05%
483226	162656	-89.9044	79	-51.0000	-367.2274	55379615	620.05%
483938	161562	-209.0311	125	-51.0000	-367.2274	55374982	620.05%

Elapsed time = 113.95 sec. (93675.38 ticks, tree = 9187.73 MB, solutions = 37)

Nodefile size = 7167.73 MB (6279.59 MB after compression)

484586	159413	-172.3045	113	-51.0000	-367.2274	55359518	620.05%
485529	161722	-101.4604	54	-51.0000	-367.2274	55433697	620.05%
486131	161941	-232.0853	139	-51.0000	-367.2274	55443663	620.05%
486946	162526	-81.6550	59	-51.0000	-367.2274	55404513	620.05%
487527	162760	-156.1149	92	-51.0000	-367.2274	55412036	620.05%
488098	162756	-160.2393	94	-51.0000	-367.2274	55463473	620.05%
488873	163196	-364.3675	591	-51.0000	-367.2274	55428823	620.05%
489167	163437	-131.5328	86	-51.0000	-367.2274	55437245	620.05%
489499	163605	-353.8026	297	-51.0000	-367.2274	55443068	620.05%
490304	163909	-255.8353	167	-51.0000	-367.2274	55449013	620.05%

Elapsed time = 143.38 sec. (132189.37 ticks, tree = 9198.49 MB, solutions = 37)

Nodefile size = 7167.73 MB (6279.59 MB after compression)

490905	160344	-153.9110	84	-51.0000	-367.2274	55422663	620.05%
491333	160512	-362.6557	903	-51.0000	-367.2274	55435036	620.05%
491496	164109	-56.2383	27	-51.0000	-367.2274	55478391	620.05%
492194	160785	-95.2200	49	-51.0000	-367.2274	55448138	620.05%

492741	160943	-352.7760	226	-51.0000	-367.2274	55453097	620.05%
493057	164745	-364.0406	1032	-51.0000	-367.2274	55509590	620.05%
493058	164746	-359.9377	895	-51.0000	-367.2274	55516078	620.05%
493120	164797	-239.5931	151	-51.0000	-367.2274	55522135	620.05%
493562	165115	-101.7102	52	-51.0000	-367.2274	55526531	620.05%
493915	165390	-364.7521	472	-51.0000	-367.2274	55531114	620.05%

Elapsed time = 181.06 sec. (181989.34 ticks, tree = 9277.64 MB, solutions = 37)

Nodefile size = 7167.73 MB (6279.59 MB after compression)

494186	165607	-206.6682	119	-51.0000	-367.2274	55535763	620.05%
--------	--------	-----------	-----	----------	-----------	----------	---------

GUB cover cuts applied: 1670

Clique cuts applied: 55

Cover cuts applied: 4658

Implied bound cuts applied: 117

Flow cuts applied: 184

Mixed integer rounding cuts applied: 4034

Zero-half cuts applied: 137

Lift and project cuts applied: 21

Gomory fractional cuts applied: 185

Root node processing (before b&c):

Real time = 0.00 sec. (2.58 ticks)

Parallel b&c, 8 threads:

Real time = 188.35 sec. (189207.47 ticks)

Sync time (average) = 8.78 sec.

Wait time (average) = 0.00 sec.

Total (root+branch&cut) = 188.35 sec. (189210.06 ticks)

Iteration 14

Bounds on # of cuts = 8 with [3 3 2]

Error = 49 (out of 100 instances)

Accuracy = 51

Solving time = 3.139274398 min (minutes)

Accumulated time = 82.023340731 min (minutes)

Solution status code = 104

LB on error = -267.006174534

Relative objective gap = 6.196199501

Selected variables:

A_AGE (Continuous)

PEMLR (Categorical)

Number of selected variables = 2 (1 continuous + 1 categorical)

Version identifier: 22.1.1.0 | 2022-11-28 | 9160aff4d

CPXPARAM_MIP_Strategy_File	3
CPXPARAM_MIP_Limits_Solutions	1
CPXPARAM_TimeLimit	81478.599556152345
CPXPARAM_MIP_Limits_TreeMemory	204800

Nodes				Cuts/			
Node	Left	Objective	IInf	Best Integer	Best Bound	ItCnt	Gap
494456	182201	infeasible		-51.0000	-367.0062	56675674	619.62%
Elapsed time = 0.24 sec. (15.33 ticks, tree = 9524.00 MB, solutions = 37)							
Nodefile size = 7477.22 MB (6520.03 MB after compression)							
494475	182210	-351.9932	224	-51.0000	-367.0062	56676084	619.62%
494530	182230	-312.2393	195	-51.0000	-367.0062	56676456	619.62%
494589	182255	-259.6874	171	-51.0000	-367.0062	56676687	619.62%
494665	182282	-180.5270	126	-51.0000	-367.0062	56676863	619.62%
494755	182309	-122.6764	100	-51.0000	-367.0062	56677062	619.62%
494863	182335	cutoff		-51.0000	-367.0062	56677152	619.62%
494937	182324	-78.8158	40	-51.0000	-367.0062	56678670	619.62%
494991	182280	-211.8674	166	-51.0000	-367.0062	56678002	619.62%
495064	182323	-88.9050	60	-51.0000	-367.0062	56678696	619.62%
495298	182342	-361.4826	400	-51.0000	-367.0062	56679876	619.62%
Elapsed time = 4.70 sec. (3143.89 ticks, tree = 9530.75 MB, solutions = 37)							
Nodefile size = 7477.22 MB (6520.03 MB after compression)							
495594	182388	-251.8648	214	-51.0000	-367.0062	56683074	619.62%
495860	182577	-108.7543	80	-51.0000	-366.9523	56694950	619.51%
496088	182540	-189.7373	134	-51.0000	-366.9144	56684616	619.44%
496320	182525	-286.3870	178	-51.0000	-366.8762	56685783	619.37%
496537	182580	-340.4613	383	-51.0000	-366.8762	56706383	619.37%
496821	182691	-147.6478	81	-51.0000	-366.8762	56689431	619.37%
497372	182733	-88.9050	52	-51.0000	-366.8762	56690879	619.37%

497664	182720	-358.8124	331	-51.0000	-366.8762	56710686	619.37%
498118	182764	-237.3959	195	-51.0000	-366.8762	56719970	619.37%
498480	182825	-73.9050	56	-51.0000	-366.8762	56720555	619.37%

Elapsed time = 17.54 sec. (12704.97 ticks, tree = 9531.09 MB, solutions = 37)

Nodefile size = 7477.22 MB (6520.03 MB after compression)

498995	182744	-91.4849	132	-51.0000	-366.8762	56695063	619.37%
499164	182989	-359.6095	298	-51.0000	-366.8762	56716539	619.37%
499601	182253	-244.0278	167	-51.0000	-366.8762	56701358	619.37%
500209	183036	-344.9522	226	-51.0000	-366.8762	56700971	619.37%
500569	183145	cutoff		-51.0000	-366.8762	56701706	619.37%
500875	183215	-223.8724	139	-51.0000	-366.8681	56703419	619.35%
501515	183296	-345.3847	265	-51.0000	-366.8681	56704825	619.35%
502158	182831	-92.9050	50	-51.0000	-366.8681	56713383	619.35%
502527	183166	-226.7766	170	-51.0000	-366.8681	56734624	619.35%
502909	182911	-262.1243	189	-51.0000	-366.8681	56716455	619.35%

Elapsed time = 31.50 sec. (22253.24 ticks, tree = 9560.30 MB, solutions = 37)

Nodefile size = 7477.22 MB (6520.03 MB after compression)

503332	183801	-327.4005	260	-51.0000	-366.8681	56775885	619.35%
503678	183393	-63.9050	28	-51.0000	-366.8681	56750272	619.35%
503992	183427	-231.5535	138	-51.0000	-366.8681	56739737	619.35%
504587	183223	-69.9050	35	-51.0000	-366.8681	56721762	619.35%
504948	183535	-301.0740	190	-51.0000	-366.8681	56742532	619.35%
505495	184109	-189.3199	149	-51.0000	-366.8681	56735249	619.35%
505823	184172	-347.1391	255	-51.0000	-366.8681	56736378	619.35%
506089	184248	-152.5782	92	-51.0000	-366.8681	56737407	619.35%
506643	185098	-102.4718	45	-51.0000	-366.8681	56816717	619.35%
507163	183514	-94.8596	38	-51.0000	-366.8681	56762693	619.35%

Elapsed time = 46.09 sec. (31831.63 ticks, tree = 9612.19 MB, solutions = 37)

Nodefile size = 7507.91 MB (6546.68 MB after compression)

507847	183617	-150.3053	85	-51.0000	-366.8681	56764277	619.35%
508280	183802	-328.5517	206	-51.0000	-366.8681	56736660	619.35%
508940	182769	-294.7658	190	-51.0000	-366.8681	56727257	619.35%
509517	183778	-95.4761	51	-51.0000	-366.8681	56769219	619.35%
509986	183838	-290.1692	199	-51.0000	-366.8681	56771014	619.35%
510665	185497	-128.6918	82	-51.0000	-366.8681	56826754	619.35%
511065	184321	-310.4287	177	-51.0000	-366.8681	56758291	619.35%
511453	184748	-323.7543	217	-51.0000	-366.8681	56799934	619.35%
511733	185661	-352.0178	342	-51.0000	-366.8681	56831520	619.35%
512296	184878	-331.4491	218	-51.0000	-366.8681	56802445	619.35%

Elapsed time = 57.78 sec. (41390.10 ticks, tree = 9769.69 MB, solutions = 37)

Nodefile size = 7582.35 MB (6611.74 MB after compression)

512740	183738	-354.3047	315	-51.0000	-366.8681	56746494	619.35%
513064	183797	-218.5924	179	-51.0000	-366.8681	56747822	619.35%
513481	184788	-81.0717	42	-51.0000	-366.8681	56759505	619.35%
513901	185908	-76.0885	72	-51.0000	-366.8681	56837494	619.35%
514340	185153	-329.4190	291	-51.0000	-366.8681	56808466	619.35%
514728	186009	-167.6558	111	-51.0000	-366.8681	56840056	619.35%
515145	186083	-313.4338	183	-51.0000	-366.8681	56841160	619.35%
515566	186176	-351.2108	233	-51.0000	-366.8681	56842573	619.35%
516099	183458	-234.0999	156	-51.0000	-366.8681	56749879	619.35%
516357	184213	-145.4731	71	-51.0000	-366.8681	56758558	619.35%

Elapsed time = 70.16 sec. (50950.48 ticks, tree = 9712.96 MB, solutions = 37)

Nodefile size = 7477.22 MB (6520.03 MB after compression)

516670	183638	-94.4050	69	-51.0000	-366.8681	56751485	619.35%
516864	185539	-353.3629	464	-51.0000	-366.8681	56816763	619.35%
517026	184475	-139.9466	79	-51.0000	-366.8681	56762655	619.35%
517314	183765	-107.6787	60	-51.0000	-366.8681	56756087	619.35%
517555	183845	-229.9116	137	-51.0000	-366.8681	56757069	619.35%
517758	185685	-338.3882	320	-51.0000	-366.8681	56822383	619.35%
517891	186354	-219.8397	132	-51.0000	-366.8681	56852885	619.35%
518073	184508	-358.8005	571	-51.0000	-366.8681	56769302	619.35%
518144	186464	-294.3353	217	-51.0000	-366.8681	56855343	619.35%
518452	184522	-338.0686	327	-51.0000	-366.8681	56773560	619.35%

Elapsed time = 82.11 sec. (60586.41 ticks, tree = 9748.86 MB, solutions = 37)

Nodefile size = 7477.22 MB (6520.03 MB after compression)

518550	186598	-301.8589	186	-51.0000	-366.8681	56857731	619.35%
518811	184586	-223.8810	142	-51.0000	-366.8681	56777456	619.35%
519155	186711	-339.9077	213	-51.0000	-366.8681	56860244	619.35%
519520	184342	-334.0798	215	-51.0000	-366.8681	56774326	619.35%
520022	184788	-364.9442	404	-51.0000	-366.8681	56781028	619.35%
520231	184896	-112.3531	60	-51.0000	-366.8681	56782057	619.35%
520511	186342	-61.9044	48	-51.0000	-366.8681	56836938	619.35%
520730	185006	-324.4910	234	-51.0000	-366.8681	56821850	619.35%
521264	186394	-278.5470	174	-51.0000	-366.8681	56840495	619.35%
521704	187004	infeasible		-51.0000	-366.8681	56869988	619.35%

Elapsed time = 94.06 sec. (70165.35 ticks, tree = 10016.79 MB, solutions = 37)

Nodefile size = 7722.86 MB (6734.35 MB after compression)

522368	185197	-202.6451	126	-51.0000	-366.8681	56827042	619.35%
--------	--------	-----------	-----	----------	-----------	----------	---------

523238	186645	-324.7403	228	-51.0000	-366.8681	56843991	619.35%
523980	184871	-232.7469	163	-51.0000	-366.8681	56807304	619.35%
524708	185358	-157.1693	125	-51.0000	-366.8681	56831143	619.35%
525280	185365	-269.3653	153	-51.0000	-366.8681	56795954	619.35%
526076	187442	-206.7137	130	-51.0000	-366.8681	56875020	619.35%
526790	187097	-172.2746	99	-51.0000	-366.8681	56849651	619.35%
527291	187608	-120.3658	68	-51.0000	-366.8681	56877135	619.35%
527767	187253	-100.7237	59	-51.0000	-366.8681	56851707	619.35%
528154	185854	-221.2637	143	-51.0000	-366.8681	56837647	619.35%

Elapsed time = 105.04 sec. (79721.78 ticks, tree = 9885.67 MB, solutions = 37)

Nodefile size = 7507.91 MB (6546.68 MB after compression)

528448	185109	-360.5449	304	-51.0000	-366.8681	56798386	619.35%
528973	187895	-53.9050	24	-51.0000	-366.8681	56882736	619.35%
529424	187460	-214.0870	144	-51.0000	-366.8681	56857690	619.35%
529852	185284	-258.7113	179	-51.0000	-366.8681	56801753	619.35%
530303	185125	-310.7185	195	-51.0000	-366.8681	56801060	619.35%
530717	186242	-282.9556	190	-51.0000	-366.8681	56845048	619.35%
531229	188181	-355.5929	249	-51.0000	-366.8681	56888841	619.35%
531749	185561	-244.8323	151	-51.0000	-366.8681	56806389	619.35%
532217	187856	-154.1270	103	-51.0000	-366.8681	56865472	619.35%
534491	186745	-336.7932	237	-51.0000	-366.8681	56853455	619.35%

Elapsed time = 119.93 sec. (92134.22 ticks, tree = 9989.67 MB, solutions = 37)

Nodefile size = 7507.91 MB (6546.68 MB after compression)

537173	186121	-92.9050	64	-51.0000	-366.8681	56817216	619.35%
539433	186104	-360.0615	320	-51.0000	-366.8681	56829949	619.35%
541783	189363	-350.7550	227	-51.0000	-366.8681	56911498	619.35%
544525	189732	-354.4750	399	-51.0000	-366.8681	56916526	619.35%
547325	187207	-180.8447	112	-51.0000	-366.8681	56852153	619.35%
549534	187269	-359.7960	299	-51.0000	-366.8681	56841886	619.35%
552483	188561	-76.0000	37	-51.0000	-366.8681	56890908	619.35%
555018	187941	-292.0210	169	-51.0000	-366.8681	56850235	619.35%
557686	188231	-248.4755	165	-51.0000	-366.8681	56855537	619.35%
560583	190787	-328.4668	210	-51.0000	-366.8681	56924690	619.35%

Elapsed time = 165.90 sec. (130289.67 ticks, tree = 10459.45 MB, solutions = 37)

Nodefile size = 7582.35 MB (6611.74 MB after compression)

563125	189628	-59.9050	34	-51.0000	-366.8681	56911187	619.35%
565396	189136	-263.8448	164	-51.0000	-366.8681	56885160	619.35%
568071	190242	-138.8590	81	-51.0000	-366.8209	56920270	619.26%
570320	189734	-104.5327	70	-51.0000	-366.8209	56893599	619.26%

573161	191852	-256.3504	179	-51.0000	-366.8209	57017459	619.26%
575843	191145	-210.8755	128	-51.0000	-366.8209	56934904	619.26%
578475	190142	-345.4650	248	-51.0000	-366.8209	56896990	619.26%
581192	192533	-166.7373	98	-51.0000	-366.8209	57036295	619.26%
583459	192174	-248.0752	140	-51.0000	-366.8209	56949279	619.26%
585497	192912	-207.3005	124	-51.0000	-366.8209	57048453	619.26%

Elapsed time = 227.81 sec. (168444.95 ticks, tree = 10705.98 MB, solutions = 37)

Nodefile size = 8470.98 MB (7389.34 MB after compression)

587923	194446	-219.1194	128	-51.0000	-366.8209	57002407	619.26%
590344	193814	-191.1955	108	-51.0000	-366.8209	56985418	619.26%
592510	191113	-132.6357	88	-51.0000	-366.8209	56935710	619.26%
594613	193928	-278.0600	157	-51.0000	-366.8209	57069018	619.26%
596034	194250	-121.5324	72	-51.0000	-366.8209	57074094	619.26%
597396	191764	-298.5444	188	-51.0000	-366.8209	56955753	619.26%
598784	192141	-156.6665	97	-51.0000	-366.8209	56955666	619.26%
599769	192343	-327.9550	264	-51.0000	-366.8209	56962010	619.26%
600940	192582	-61.4004	30	-51.0000	-366.8209	56968128	619.26%
602947	191282	-360.3925	339	-51.0000	-366.8209	56966375	619.26%

Elapsed time = 273.41 sec. (206606.47 ticks, tree = 10538.83 MB, solutions = 37)

Nodefile size = 7477.22 MB (6520.03 MB after compression)

604824	195159	-209.5223	165	-51.0000	-366.8209	57110082	619.26%
606991	194686	-228.1652	141	-51.0000	-366.8209	57033764	619.26%
609377	196058	-89.4794	40	-51.0000	-366.8209	57072395	619.26%
611483	196054	-210.2735	128	-51.0000	-366.8209	57126646	619.26%
612145	196417	-241.6389	147	-51.0000	-366.8209	57130810	619.26%
612256	195015	-359.0492	1034	-51.0000	-366.8125	57046859	619.24%
612368	217470	-320.2838	211	-51.0000	-366.7797	57738968	619.18%
612887	228501	-284.2145	276	-51.0000	-366.7797	58010282	619.18%
613500	239780	-358.5293	274	-51.0000	-366.7797	58330379	619.18%
613861	251538	-360.7471	397	-51.0000	-366.7514	58625105	619.12%

Elapsed time = 387.78 sec. (245504.90 ticks, tree = 17529.07 MB, solutions = 37)

Nodefile size = 15466.73 MB (13518.12 MB after compression)

615353	239610	-359.5604	1040	-51.0000	-366.7514	58326157	619.12%
616586	263789	-273.0794	174	-51.0000	-366.7514	58939204	619.12%
617658	274839	-355.0167	266	-51.0000	-366.3450	59284748	618.32%
618974	275150	-281.4216	230	-51.0000	-366.3450	59302104	618.32%
620336	276782	-209.2482	130	-51.0000	-366.3450	59362700	618.32%
621684	277961	-214.6874	119	-51.0000	-366.3450	59441273	618.32%
622900	278715	-298.6923	213	-51.0000	-366.3450	59465451	618.32%

624291	279922	-315.4854	231	-51.0000	-366.3450	59533597	618.32%
625218	281037	-120.8353	63	-51.0000	-366.3450	59570495	618.32%
625993	281792	-361.4253	347	-51.0000	-366.3450	59617364	618.32%
Elapsed time = 469.11 sec. (283897.44 ticks, tree = 21071.25 MB, solutions = 37)							
Nodefile size = 18993.43 MB (16609.64 MB after compression)							
626629	282086	-355.1436	393	-51.0000	-366.3450	59634771	618.32%
627551	282737	-161.8955	100	-51.0000	-366.3450	59683802	618.32%
628412	283151	-358.0560	333	-51.0000	-366.3450	59702143	618.32%
629606	283908	-223.1311	128	-51.0000	-366.3358	59768358	618.31%
630987	285351	-186.6854	131	-51.0000	-365.9983	59853631	617.64%
631970	286256	-60.4004	37	-51.0000	-365.9983	59881553	617.64%
632808	286815	-190.7839	136	-51.0000	-365.9983	59919228	617.64%
634294	287409	-119.7052	96	-51.0000	-365.9983	59983074	617.64%
635715	288834	-355.2322	320	-51.0000	-365.8797	60038286	617.41%
636500	289526	-336.6266	237	-51.0000	-365.8797	60093169	617.41%
Elapsed time = 527.49 sec. (322134.89 ticks, tree = 22006.45 MB, solutions = 37)							
Nodefile size = 19922.83 MB (17427.82 MB after compression)							
637899	289868	-261.2537	170	-51.0000	-365.8532	60117036	617.36%
639303	291156	-357.4336	362	-51.0000	-365.8468	60164756	617.35%
640364	292296	-357.0002	225	-51.0000	-365.8468	60205310	617.35%
641826	293296	-129.2980	67	-51.0000	-365.8468	60231289	617.35%
643710	294344	-324.7103	219	-51.0000	-365.8468	60275209	617.35%
645087	295929	-357.5296	251	-51.0000	-365.8468	60347341	617.35%
647574	297096	-361.3072	347	-51.0000	-365.8468	60403205	617.35%
650132	298525	-157.9824	91	-51.0000	-365.8468	60437070	617.35%
652370	300285	-215.2495	130	-51.0000	-365.8468	60481233	617.35%
653642	302565	-225.2487	127	-51.0000	-365.8468	60540176	617.35%
Elapsed time = 588.75 sec. (360318.24 ticks, tree = 23214.25 MB, solutions = 37)							
Nodefile size = 21147.62 MB (18482.80 MB after compression)							
655359	303578	-241.1639	143	-51.0000	-365.8468	60576128	617.35%
656778	304539	-233.3815	133	-51.0000	-365.7387	60611664	617.13%
658460	305910	-334.7923	278	-51.0000	-365.6873	60683628	617.03%
661036	307178	-306.5428	187	-51.0000	-365.6873	60721530	617.03%
662791	308542	-360.1018	349	-51.0000	-365.6873	60767056	617.03%
664931	310615	-73.4004	33	-51.0000	-365.6873	60843282	617.03%
666181	311479	-80.9050	38	-51.0000	-365.5489	60873920	616.76%
667712	313460	-341.7799	221	-51.0000	-365.4342	60942375	616.54%
668738	314033	-250.3615	164	-51.0000	-365.4342	60969439	616.54%
670378	314777	-340.3181	254	-51.0000	-365.4342	61007417	616.54%

Elapsed time = 648.51 sec. (398512.68 ticks, tree = 24220.43 MB, solutions = 37)

Nodefile size = 22149.07 MB (19347.63 MB after compression)

671660	316624	-190.6028	117	-51.0000	-365.4342	61082628	616.54%
673339	316740	-255.8649	201	-51.0000	-365.4342	61117104	616.54%
675296	318619	-119.2922	64	-51.0000	-365.4342	61162844	616.54%
677401	320444	-231.1208	167	-51.0000	-365.4342	61223447	616.54%
679864	321923	-307.6958	207	-51.0000	-365.4342	61254347	616.54%
681222	323429	-355.7521	293	-51.0000	-365.4342	61293481	616.54%
682604	324985	-164.2641	90	-51.0000	-365.3694	61405616	616.41%
684015	325743	-179.6358	117	-51.0000	-365.3555	61458609	616.38%
685518	327413	-250.6560	141	-51.0000	-365.3071	61539485	616.29%
687434	328626	-146.6897	95	-51.0000	-365.3062	61584384	616.29%

Elapsed time = 708.04 sec. (436685.01 ticks, tree = 25451.98 MB, solutions = 37)

Nodefile size = 23372.09 MB (20413.94 MB after compression)

689399	329467	-99.3480	36	-51.0000	-365.2479	61635355	616.17%
691000	331344	-346.8231	238	-51.0000	-365.2050	61692904	616.09%
692237	332230	-329.2119	241	-51.0000	-365.2050	61733192	616.09%
693523	332898	-86.5479	42	-51.0000	-365.2050	61759830	616.09%
694552	333849	-187.8206	113	-51.0000	-365.1849	61826192	616.05%
696545	334946	-64.0000	28	-51.0000	-365.1849	61889587	616.05%
698225	336435	-223.7746	142	-51.0000	-365.1395	61969519	615.96%
700385	337181	-141.2558	76	-51.0000	-365.1393	61999411	615.96%
701942	339213	-228.3630	129	-51.0000	-365.1151	62061295	615.91%
703039	340138	-336.6943	518	-51.0000	-365.1074	62106040	615.90%

Elapsed time = 769.48 sec. (474868.46 ticks, tree = 26523.33 MB, solutions = 37)

Nodefile size = 24455.97 MB (21345.49 MB after compression)

703806	340972	-154.5474	96	-51.0000	-365.1074	62151674	615.90%
704648	342280	-338.7462	252	-51.0000	-365.1031	62231946	615.89%
705803	342510	-349.4838	328	-51.0000	-365.0948	62264654	615.87%
706909	343736	-348.7916	269	-51.0000	-365.0948	62339333	615.87%
708639	344121	-208.9617	132	-51.0000	-365.0948	62361186	615.87%
710029	345547	-179.6985	112	-51.0000	-365.0948	62447113	615.87%
711771	346626	-185.8141	108	-51.0000	-365.0912	62486059	615.87%
712627	348066	-362.4201	343	-51.0000	-365.0912	62582067	615.87%
713715	348780	-80.0606	43	-51.0000	-365.0912	62624224	615.87%
715391	349314	-306.6991	235	-51.0000	-365.0912	62666144	615.87%

Elapsed time = 824.55 sec. (513173.77 ticks, tree = 26881.08 MB, solutions = 37)

Nodefile size = 24808.87 MB (21640.42 MB after compression)

717378	350702	-353.4336	324	-51.0000	-365.0912	62715620	615.87%
--------	--------	-----------	-----	----------	-----------	----------	---------

719540	351649	-316.9172	222	-51.0000	-365.0912	62746033	615.87%
721322	353579	-162.3109	86	-51.0000	-365.0912	62799262	615.87%
723113	355134	-63.9143	35	-51.0000	-365.0289	62843640	615.74%
724802	356270	-143.9895	81	-51.0000	-365.0169	62884338	615.72%
726441	357146	-175.7427	100	-51.0000	-365.0169	62911220	615.72%
728427	358874	-343.8456	331	-51.0000	-365.0169	63006867	615.72%
731003	361195	-342.7434	262	-51.0000	-365.0169	63074158	615.72%
733427	361941	-268.4267	171	-51.0000	-364.9677	63090909	615.62%
735290	363459	cutoff		-51.0000	-364.9677	63130298	615.62%

Elapsed time = 885.78 sec. (551330.14 ticks, tree = 28246.52 MB, solutions = 37)

Nodefile size = 26162.99 MB (22819.78 MB after compression)

737115	365594	-144.9050	89	-51.0000	-364.9635	63178021	615.61%
738541	366505	-117.0880	69	-51.0000	-364.9446	63192669	615.58%
739326	367895	-338.1909	310	-51.0000	-364.9446	63273313	615.58%
740151	368444	-354.4144	323	-51.0000	-364.9446	63300507	615.58%
742291	369430	-258.6669	164	-51.0000	-364.9336	63409029	615.56%
745357	370124	-133.1314	67	-51.0000	-364.9268	63420455	615.54%
747378	373306	-70.4004	24	-51.0000	-364.9221	63486694	615.53%
749580	374547	-60.0000	43	-51.0000	-364.9158	63533026	615.52%
752396	376503	-224.8379	132	-51.0000	-364.9158	63590702	615.52%
753817	377198	-110.4004	59	-51.0000	-364.9158	63601596	615.52%

Elapsed time = 947.27 sec. (589510.67 ticks, tree = 29838.63 MB, solutions = 37)

Nodefile size = 27723.94 MB (24182.95 MB after compression)

754664	378350	-313.3719	241	-51.0000	-364.9158	63637522	615.52%
755945	380009	-132.6635	77	-51.0000	-364.8896	63692159	615.47%
756996	381032	-114.4004	55	-51.0000	-364.8721	63750087	615.44%
758046	382017	-353.4691	330	-51.0000	-364.8721	63794359	615.44%
758657	382456	-354.2886	246	-51.0000	-364.8721	63843088	615.44%
759881	382519	-240.3827	153	-51.0000	-364.8710	63859500	615.43%
761113	384017	-109.7342	57	-51.0000	-364.8710	63946364	615.43%
762231	384489	-201.7776	137	-51.0000	-364.8699	63967543	615.43%
763687	385146	-167.7341	97	-51.0000	-364.8699	64005961	615.43%
764990	386464	-360.0574	326	-51.0000	-364.8699	64058084	615.43%

Elapsed time = 1005.19 sec. (627814.24 ticks, tree = 30794.27 MB, solutions = 37)

Nodefile size = 28692.51 MB (25023.74 MB after compression)

765714	386900	-139.3959	80	-51.0000	-364.8697	64109034	615.43%
766817	387531	-295.4339	190	-51.0000	-364.8697	64159777	615.43%
767913	388740	-70.4004	29	-51.0000	-364.8697	64211795	615.43%
768975	390150	-137.9050	71	-51.0000	-364.8697	64307062	615.43%

771045	390586	-353.6152	274	-51.0000	-364.8488	64348089	615.39%
773557	392244	-62.7203	32	-51.0000	-364.8292	64409874	615.35%
775988	394237	cutoff		-51.0000	-364.8095	64462634	615.31%
777702	396139	-166.8143	88	-51.0000	-364.8095	64503623	615.31%
779614	396520	-122.6436	68	-51.0000	-364.7910	64539613	615.28%
782146	398855	-312.5603	192	-51.0000	-364.7910	64610935	615.28%

Elapsed time = 1065.82 sec. (665982.42 ticks, tree = 32000.35 MB, solutions = 37)

Nodefile size = 29928.83 MB (26098.76 MB after compression)

784000	399838	-224.3325	136	-51.0000	-364.7910	64651290	615.28%
785883	401786	-194.0282	115	-51.0000	-364.7910	64740881	615.28%
787276	402494	-236.2952	187	-51.0000	-364.7910	64773173	615.28%
788203	403332	-208.3501	132	-51.0000	-364.7626	64821019	615.22%
789654	404080	-274.5377	173	-51.0000	-364.7626	64871019	615.22%
791185	405717	-136.7522	74	-51.0000	-364.7626	64928476	615.22%
792944	407063	-143.8552	81	-51.0000	-364.7489	65009012	615.19%
794449	407935	-60.4004	35	-51.0000	-364.7489	65055525	615.19%
796012	408937	-244.0269	148	-51.0000	-364.7489	65090206	615.19%
797658	410676	-108.4004	65	-51.0000	-364.7316	65157440	615.16%

Elapsed time = 1123.50 sec. (704203.36 ticks, tree = 32906.29 MB, solutions = 37)

Nodefile size = 30835.32 MB (26886.41 MB after compression)

799447	411601	-334.4220	270	-51.0000	-364.7302	65214712	615.16%
800825	412665	-327.5934	259	-51.0000	-364.7141	65259056	615.13%
801941	413971	-339.3653	241	-51.0000	-364.7058	65332119	615.11%
803933	414717	-68.2934	28	-51.0000	-364.7058	65352630	615.11%
806553	416163	-182.9371	105	-51.0000	-364.7058	65420516	615.11%
808329	417827	-92.0000	46	-51.0000	-364.6968	65446803	615.09%
809926	419512	-324.4454	225	-51.0000	-364.6968	65503730	615.09%
811015	420726	-346.9501	222	-51.0000	-364.6855	65546728	615.07%
812351	421285	-332.8297	220	-51.0000	-364.6809	65576229	615.06%
813635	422486	-188.0082	108	-51.0000	-364.6809	65605887	615.06%

Elapsed time = 1183.30 sec. (742384.03 ticks, tree = 34000.77 MB, solutions = 37)

Nodefile size = 31911.50 MB (27825.65 MB after compression)

815552	423494	-73.9775	32	-51.0000	-364.6777	65663571	615.05%
817134	424794	-341.0814	263	-51.0000	-364.6777	65690837	615.05%
818641	426281	-180.3990	110	-51.0000	-364.6777	65757727	615.05%
820682	426974	-165.8716	91	-51.0000	-364.6777	65800359	615.05%
822833	428289	-84.4004	53	-51.0000	-364.6777	65824189	615.05%
824603	430176	-322.1515	209	-51.0000	-364.6777	65871864	615.05%
826717	432281	-222.8604	118	-51.0000	-364.6777	65938161	615.05%

828986	432669	-221.0570	168	-51.0000	-364.6777	65943629	615.05%
830980	435000	-274.5929	162	-51.0000	-364.6777	66001113	615.05%
833801	436907	-324.0902	213	-51.0000	-364.6475	66128348	615.00%
Elapsed time = 1245.90 sec. (780552.64 ticks, tree = 35882.93 MB, solutions = 37)							
Nodefile size = 33776.48 MB (29467.99 MB after compression)							
835709	439181	-133.2156	72	-51.0000	-364.6475	66178885	615.00%
837674	439908	-175.4624	114	-51.0000	-364.6475	66189714	615.00%
839601	441162	-250.9862	176	-51.0000	-364.6475	66240455	615.00%
841355	442464	-300.2932	199	-51.0000	-364.5726	66291801	614.85%
842695	444227	-317.4591	246	-51.0000	-364.5726	66361667	614.85%
844185	445795	-174.4656	98	-51.0000	-364.5726	66440393	614.85%
845686	446846	-285.6416	179	-51.0000	-364.5718	66500594	614.85%
847592	448069	-114.7342	59	-51.0000	-364.5542	66550404	614.81%
849312	448835	-257.9658	167	-51.0000	-364.5542	66582371	614.81%
851394	450013	-314.9399	211	-51.0000	-364.5542	66633782	614.81%
Elapsed time = 1307.35 sec. (818726.26 ticks, tree = 37244.89 MB, solutions = 37)							
Nodefile size = 35133.82 MB (30656.34 MB after compression)							
853084	452324	-312.0660	196	-51.0000	-364.5542	66693652	614.81%
854812	453846	-356.9786	261	-51.0000	-364.5542	66761242	614.81%
855968	454651	-63.4004	24	-51.0000	-364.5542	66802143	614.81%
857129	455667	-171.9293	96	-51.0000	-364.5490	66843806	614.80%
858381	456135	-163.1172	96	-51.0000	-364.5490	66892873	614.80%
860461	456922	-61.1046	29	-51.0000	-364.5490	66953083	614.80%
862972	458277	-214.0353	124	-51.0000	-364.5200	66982830	614.75%
865304	460473	-180.0223	111	-51.0000	-364.5200	67059361	614.75%
867660	461169	-321.5716	208	-51.0000	-364.4985	67072536	614.70%
870293	463316	-340.5852	217	-51.0000	-364.4819	67128946	614.67%
Elapsed time = 1368.78 sec. (856888.40 ticks, tree = 38648.48 MB, solutions = 37)							
Nodefile size = 36539.17 MB (31877.98 MB after compression)							
873538	467608	-217.7467	139	-51.0000	-364.4664	67233746	614.64%
876593	467125	-244.4728	135	-51.0000	-364.4574	67228464	614.62%
879178	470882	-154.9824	104	-51.0000	-364.4499	67296718	614.61%
881958	472445	-284.8297	214	-51.0000	-364.4351	67334944	614.58%
884732	474861	-279.1491	156	-51.0000	-364.4351	67377191	614.58%
887617	477416	-315.0698	194	-51.0000	-364.4211	67435378	614.55%
890379	478124	-214.2439	123	-51.0000	-364.4195	67448458	614.55%
892245	480785	-325.1725	248	-51.0000	-364.4066	67510545	614.52%
894557	482750	-324.9394	205	-51.0000	-364.4066	67569546	614.52%
896840	483406	-343.9195	287	-51.0000	-364.3964	67595487	614.50%

Elapsed time = 1434.42 sec. (895055.27 ticks, tree = 41047.78 MB, solutions = 37)

Nodefile size = 38972.02 MB (34000.81 MB after compression)

898846	485804	-362.7281	405	-51.0000	-364.3964	67685395	614.50%
900772	486739	-93.3809	45	-51.0000	-364.3964	67709447	614.50%
902306	488326	-249.0641	223	-51.0000	-364.3780	67756185	614.47%
904443	489887	-182.8472	107	-51.0000	-364.3780	67817292	614.47%
906206	491028	-234.4828	133	-51.0000	-364.3780	67848657	614.47%
907466	491892	-312.0740	182	-51.0000	-364.3676	67899357	614.45%
908931	493915	-326.9604	233	-51.0000	-364.3676	67959190	614.45%
910191	494806	-275.8860	191	-51.0000	-364.3676	68017211	614.45%
911432	496455	-70.3965	32	-51.0000	-364.3676	68085484	614.45%
912334	496717	-64.4004	28	-51.0000	-364.3676	68095822	614.45%

Elapsed time = 1492.98 sec. (933225.40 ticks, tree = 41910.78 MB, solutions = 37)

Nodefile size = 39834.84 MB (34729.80 MB after compression)

913439	497838	-194.3815	105	-51.0000	-364.3676	68177102	614.45%
914980	498261	-152.9293	86	-51.0000	-364.3676	68196338	614.45%
915804	499766	-323.8244	224	-51.0000	-364.3575	68290289	614.43%
916628	500079	-302.3960	211	-51.0000	-364.3575	68305529	614.43%
918039	500368	-343.3726	228	-51.0000	-364.3403	68340234	614.39%
920007	501929	-275.8006	198	-51.0000	-364.3317	68419509	614.38%
922741	503623	-362.6283	406	-51.0000	-364.3275	68467641	614.37%
925606	505643	-142.1086	79	-51.0000	-364.3179	68521503	614.35%
928546	508163	-217.8474	137	-51.0000	-364.3103	68580599	614.33%
930886	509848	-158.5826	87	-51.0000	-364.2995	68622180	614.31%

Elapsed time = 1554.53 sec. (971382.80 ticks, tree = 43251.15 MB, solutions = 37)

Nodefile size = 41149.86 MB (35884.09 MB after compression)

932899	511986	-119.3072	74	-51.0000	-364.2902	68685278	614.29%
934637	512936	-269.4526	187	-51.0000	-364.2902	68703163	614.29%
936749	513505	-198.9682	162	-51.0000	-364.2805	68733415	614.28%
939019	516384	-100.4004	61	-51.0000	-364.2805	68830529	614.28%
940965	516910	-213.4897	124	-51.0000	-364.2729	68840204	614.26%
943366	518019	-329.1310	247	-51.0000	-364.2674	68885751	614.25%
946119	520442	-350.7558	252	-51.0000	-364.2625	68944343	614.24%
949262	523916	-170.8749	94	-51.0000	-364.2478	69045300	614.21%
951577	525662	-284.4097	170	-51.0000	-364.2457	69087488	614.21%
953822	526036	-343.7839	272	-51.0000	-364.2457	69097886	614.21%

Elapsed time = 1618.93 sec. (1009540.36 ticks, tree = 44976.81 MB, solutions = 37)

Nodefile size = 42900.58 MB (37403.42 MB after compression)

956067	528253	-114.9327	57	-51.0000	-364.2457	69141165	614.21%
--------	--------	-----------	----	----------	-----------	----------	---------

958128	529673	-203.9004	114	-51.0000	-364.2349	69171360	614.19%
960075	531911	-80.4004	39	-51.0000	-364.2178	69250399	614.15%
961383	532901	-192.4865	122	-51.0000	-364.2178	69269827	614.15%
963351	534290	-331.6833	237	-51.0000	-364.2171	69334703	614.15%
964998	535026	-292.4749	209	-51.0000	-364.2171	69368474	614.15%
966596	537588	-233.1557	129	-51.0000	-364.2171	69455104	614.15%
968783	537908	infeasible		-51.0000	-364.1968	69462106	614.11%
970664	538749	-158.7677	91	-51.0000	-364.1944	69525096	614.11%
971877	540717	-303.3489	187	-51.0000	-364.1830	69565557	614.08%

Elapsed time = 1681.97 sec. (1047705.63 ticks, tree = 46825.36 MB, solutions = 37)

Nodefile size = 44749.88 MB (39030.34 MB after compression)

972591	541449	-360.6409	389	-51.0000	-364.1830	69614316	614.08%
973284	542393	-346.1220	219	-51.0000	-364.1810	69668081	614.08%
974504	543255	-356.6271	308	-51.0000	-364.1748	69742147	614.07%
975615	544122	-82.3339	49	-51.0000	-364.1721	69809177	614.06%
976405	544162	-359.4143	328	-51.0000	-364.1721	69815975	614.06%
978363	545748	-101.4004	57	-51.0000	-364.1668	69906213	614.05%
979846	546197	-351.0579	406	-51.0000	-364.1654	69950306	614.05%
981386	547870	-81.4004	39	-51.0000	-364.1654	69980541	614.05%
982550	549015	-333.7721	202	-51.0000	-364.1654	70056096	614.05%
983567	549656	-147.8308	79	-51.0000	-364.1654	70073713	614.05%

Elapsed time = 1742.23 sec. (1085912.50 ticks, tree = 47716.70 MB, solutions = 37)

Nodefile size = 45612.43 MB (39779.51 MB after compression)

984554	551413	cutoff		-51.0000	-364.1654	70161318	614.05%
985491	551428	-150.5823	93	-51.0000	-364.1654	70191726	614.05%
986750	552700	-282.6257	188	-51.0000	-364.1467	70255403	614.01%
988694	553598	-362.4581	443	-51.0000	-364.1387	70317096	614.00%
990679	555384	-217.3788	127	-51.0000	-364.1311	70394011	613.98%
992846	556974	-362.6461	391	-51.0000	-364.1311	70465635	613.98%
994875	556704	-249.0947	144	-51.0000	-364.1311	70449904	613.98%
996495	558587	-56.8487	28	-51.0000	-364.1193	70519524	613.96%
998699	558897	-98.4589	53	-51.0000	-364.1120	70551356	613.95%
1000118	561480	-343.5201	231	-51.0000	-364.1120	70621249	613.95%

Elapsed time = 1798.66 sec. (1124089.14 ticks, tree = 48342.35 MB, solutions = 37)

Nodefile size = 46248.91 MB (40321.99 MB after compression)

1002022	563411	-277.8063	195	-51.0000	-364.1120	70676416	613.95%
1003930	564748	-65.4004	19	-51.0000	-364.1120	70710284	613.95%
1006162	565013	-362.5691	336	-51.0000	-364.1120	70739515	613.95%
1008681	566024	-125.8004	73	-51.0000	-364.1120	70757009	613.95%

1010748	569646	-67.4004	33	-51.0000	-364.1120	70829790	613.95%
1012267	569841	-291.9516	204	-51.0000	-364.1120	70837248	613.95%
1013891	572086	-161.2518	85	-51.0000	-364.1120	70909039	613.95%
1015221	573589	-211.6472	122	-51.0000	-364.1120	70967323	613.95%
1016223	573677	-312.2773	198	-51.0000	-364.1120	70974601	613.95%
1018064	575321	-64.4004	27	-51.0000	-364.0706	71125303	613.86%

Elapsed time = 1860.88 sec. (1162268.08 ticks, tree = 49744.02 MB, solutions = 37)

Nodefile size = 47631.36 MB (41523.58 MB after compression)

1019166	576465	-115.9403	63	-51.0000	-364.0706	71178541	613.86%
1020846	576679	-64.4004	35	-51.0000	-364.0568	71195308	613.84%
1022213	578055	-173.5622	90	-51.0000	-364.0568	71266779	613.84%
1023857	578647	-65.4004	29	-51.0000	-364.0531	71297045	613.83%
1026303	581678	-234.5717	164	-51.0000	-364.0414	71391239	613.81%
1029153	583567	-158.5802	93	-51.0000	-364.0414	71445861	613.81%
1032346	585534	-348.8949	248	-51.0000	-364.0236	71475979	613.77%
1035328	585926	-347.8431	215	-51.0000	-364.0196	71480905	613.76%
1037744	587920	-211.7419	131	-51.0000	-364.0029	71521216	613.73%
1040441	591033	-303.7881	176	-51.0000	-363.9979	71589175	613.72%

Elapsed time = 1926.76 sec. (1200428.91 ticks, tree = 51781.74 MB, solutions = 37)

Nodefile size = 49671.30 MB (43311.39 MB after compression)

1042352	593264	-71.9796	30	-51.0000	-363.9963	71626550	613.72%
1043678	593042	-324.0286	219	-51.0000	-363.9963	71627371	613.72%
1045587	595732	-200.1915	114	-51.0000	-363.9832	71708209	613.69%
1047688	596461	-254.6115	163	-51.0000	-363.9832	71730582	613.69%
1049872	597740	-287.0454	175	-51.0000	-363.9832	71785578	613.69%
1051704	600386	-91.4004	47	-51.0000	-363.9832	71849627	613.69%
1053617	602054	-261.5171	153	-51.0000	-363.9772	71927071	613.68%
1055976	603674	-132.4849	72	-51.0000	-363.9772	71965461	613.68%
1058607	604897	-255.3902	153	-51.0000	-363.9635	72011301	613.65%
1060413	605986	-354.7212	330	-51.0000	-363.9578	72031929	613.64%

Elapsed time = 1988.67 sec. (1238609.25 ticks, tree = 53778.28 MB, solutions = 37)

Nodefile size = 51644.80 MB (45060.90 MB after compression)

1061914	607317	cutoff		-51.0000	-363.9507	72074436	613.63%
1063680	609340	-360.3288	326	-51.0000	-363.9390	72148771	613.61%
1065328	610155	-124.4046	63	-51.0000	-363.9390	72161179	613.61%
1066797	611383	-59.0000	29	-51.0000	-363.9390	72237860	613.61%
1068142	612237	-224.3689	137	-51.0000	-363.9354	72271213	613.60%
1070142	614705	-76.4004	67	-51.0000	-363.9267	72406469	613.58%
1071992	614979	-246.9402	155	-51.0000	-363.9267	72398235	613.58%

1073903	617010	-228.2843	137	-51.0000	-363.9267	72494327	613.58%
1075706	617392	-204.5836	114	-51.0000	-363.9267	72507148	613.58%
1077501	619338	-86.0025	40	-51.0000	-363.9267	72547889	613.58%

Elapsed time = 2046.66 sec. (1276770.39 ticks, tree = 54677.74 MB, solutions = 37)

Nodefile size = 52565.31 MB (45853.57 MB after compression)

1079161	620410	-209.3248	118	-51.0000	-363.9135	72608861	613.56%
1081649	622851	-300.1106	181	-51.0000	-363.8983	72686256	613.53%
1083926	623234	-323.1341	202	-51.0000	-363.8983	72691650	613.53%
1086076	625246	-193.9602	117	-51.0000	-363.8939	72732098	613.52%
1088179	627756	-283.5262	204	-51.0000	-363.8939	72790211	613.52%
1091190	628092	-287.6054	202	-51.0000	-363.8939	72814198	613.52%
1093431	629524	-254.8223	146	-51.0000	-363.8939	72854966	613.52%
1095986	632197	-346.0530	225	-51.0000	-363.8939	72931006	613.52%
1098500	634277	-179.1473	104	-51.0000	-363.8939	72978948	613.52%
1100539	636794	-187.8855	109	-51.0000	-363.8939	73044618	613.52%

Elapsed time = 2112.64 sec. (1314938.67 ticks, tree = 56546.65 MB, solutions = 37)

Nodefile size = 54466.62 MB (47506.30 MB after compression)

1103600	637668	-312.4463	201	-51.0000	-363.8939	73072583	613.52%
1106082	638900	-267.2381	173	-51.0000	-363.8939	73105694	613.52%
1108007	641241	-209.4077	120	-51.0000	-363.8939	73190173	613.52%
1109950	644769	-99.5139	51	-51.0000	-363.8939	73272081	613.52%
1111945	645280	-92.0000	48	-51.0000	-363.8939	73287605	613.52%
1113936	646411	-228.3004	131	-51.0000	-363.8939	73319493	613.52%
1116470	648988	-229.0410	145	-51.0000	-363.8939	73413311	613.52%
1119248	649821	-191.2266	109	-51.0000	-363.8939	73426213	613.52%
1121744	651852	-229.0925	141	-51.0000	-363.8939	73474305	613.52%
1124550	652993	-117.5139	51	-51.0000	-363.8939	73494935	613.52%

Elapsed time = 2177.62 sec. (1353097.69 ticks, tree = 58255.81 MB, solutions = 37)

Nodefile size = 56118.36 MB (48944.37 MB after compression)

1127139	655456	-90.7754	40	-51.0000	-363.8939	73547001	613.52%
1130118	657939	-351.8666	276	-51.0000	-363.8281	73617943	613.39%
1132460	659122	-340.0214	322	-51.0000	-363.8281	73638023	613.39%
1135268	661999	-350.3458	253	-51.0000	-363.8281	73703788	613.39%
1138155	663270	-349.7733	221	-51.0000	-363.7914	73727031	613.32%
1141315	666997	-109.6681	61	-51.0000	-363.7801	73803719	613.29%
1143035	667300	-319.7683	197	-51.0000	-363.7685	73809554	613.27%
1144567	670776	-253.0387	158	-51.0000	-363.7537	73913216	613.24%
1146521	670545	-259.3532	152	-51.0000	-363.7537	73901076	613.24%
1149449	672203	-348.4441	219	-51.0000	-363.7500	73968135	613.24%

Elapsed time = 2242.11 sec. (1391262.97 ticks, tree = 60633.18 MB, solutions = 37)

Nodefile size = 58549.92 MB (51084.18 MB after compression)

1152049	673941	-272.6028	196	-51.0000	-363.7500	74029925	613.24%
1154335	675960	-232.3749	144	-51.0000	-363.7346	74096293	613.21%
1156256	677959	-342.2852	228	-51.0000	-363.7346	74162994	613.21%
1158262	679301	-283.8515	185	-51.0000	-363.7346	74213670	613.21%
1160659	681152	-329.8099	233	-51.0000	-363.7346	74291318	613.21%
1163080	682052	-356.6172	229	-51.0000	-363.7346	74302383	613.21%
1164962	684898	-239.2677	149	-51.0000	-363.7346	74363108	613.21%
1166859	686253	-215.5065	147	-51.0000	-363.7035	74429048	613.14%
1169268	687434	-137.5978	79	-51.0000	-363.7026	74462687	613.14%
1171545	689622	-209.2428	119	-51.0000	-363.7026	74521715	613.14%

Elapsed time = 2302.32 sec. (1429431.71 ticks, tree = 61810.58 MB, solutions = 37)

Nodefile size = 59703.04 MB (52075.58 MB after compression)

1173411	692218	-356.7395	294	-51.0000	-363.7002	74586749	613.14%
1175378	692552	-168.7210	87	-51.0000	-363.6861	74594723	613.11%
1177260	694405	-353.1794	310	-51.0000	-363.6861	74666144	613.11%
1178498	695421	-323.5001	196	-51.0000	-363.6825	74706969	613.10%
1180148	696748	-157.0337	90	-51.0000	-363.6825	74757697	613.10%
1181894	698492	-174.7770	102	-51.0000	-363.6749	74851125	613.09%
1183386	700062	-74.6611	40	-51.0000	-363.6749	74931587	613.09%
1185302	699922	-80.6800	35	-51.0000	-363.6678	74926413	613.07%
1186617	701400	-271.7142	167	-51.0000	-363.6671	74988295	613.07%
1189228	702436	-317.6486	188	-51.0000	-363.6556	75016408	613.05%

Elapsed time = 2360.98 sec. (1467594.59 ticks, tree = 62502.87 MB, solutions = 37)

Nodefile size = 60366.86 MB (52639.96 MB after compression)

1191792	705430	-255.5873	151	-51.0000	-363.6524	75108073	613.04%
1194437	706237	-124.2892	69	-51.0000	-363.6470	75131286	613.03%
1197089	707759	-307.1593	218	-51.0000	-363.6470	75162684	613.03%
1199984	710769	-130.2921	67	-51.0000	-363.6466	75226744	613.03%
1202782	713542	infeasible		-51.0000	-363.6466	75289008	613.03%
1205543	714868	-250.1788	148	-51.0000	-363.6466	75306792	613.03%
1207640	716798	-324.8750	197	-51.0000	-363.6272	75347915	612.99%
1209905	717923	-274.8872	173	-51.0000	-363.6272	75381128	612.99%
1211507	720196	-128.8731	68	-51.0000	-363.6147	75451092	612.97%
1212988	721254	-185.9255	105	-51.0000	-363.6147	75476208	612.97%

Elapsed time = 2430.66 sec. (1505753.18 ticks, tree = 65025.03 MB, solutions = 37)

Nodefile size = 62942.95 MB (54909.80 MB after compression)

1214464	721679	-114.8208	59	-51.0000	-363.6147	75481562	612.97%
---------	--------	-----------	----	----------	-----------	----------	---------

1216487	723625	-345.3600	243	-51.0000	-363.6147	75547226	612.97%
1218075	723995	-323.9574	195	-51.0000	-363.6134	75552962	612.97%
1219748	725238	-302.7574	216	-51.0000	-363.6134	75612915	612.97%
1221521	726612	-361.9645	476	-51.0000	-363.5945	75637366	612.93%
1222967	728865	-342.9362	292	-51.0000	-363.5945	75726583	612.93%
1224384	730826	-247.8571	152	-51.0000	-363.5945	75797131	612.93%
1225757	731044	-330.0338	261	-51.0000	-363.5945	75791560	612.93%
1226952	732586	-258.1447	163	-51.0000	-363.5845	75882721	612.91%
1228618	732872	-321.9352	194	-51.0000	-363.5839	75927746	612.91%

Elapsed time = 2489.07 sec. (1543964.51 ticks, tree = 66176.85 MB, solutions = 37)

Nodefile size = 64093.97 MB (55912.45 MB after compression)

1230703	733149	-351.7396	436	-51.0000	-363.5839	75968280	612.91%
1233278	735609	-131.9364	71	-51.0000	-363.5839	76043170	612.91%
1234652	737416	-235.5213	139	-51.0000	-363.5839	76127813	612.91%
1236454	738419	-181.1871	107	-51.0000	-363.5549	76156762	612.85%
1237699	740772	-129.0000	66	-51.0000	-363.5549	76254593	612.85%
1239535	740626	-147.0000	76	-51.0000	-363.5501	76250246	612.84%
1241284	742271	-246.1793	138	-51.0000	-363.5421	76335355	612.83%
1243215	744506	-336.5570	217	-51.0000	-363.5421	76435943	612.83%
1245194	745177	-297.8112	177	-51.0000	-363.5421	76453687	612.83%
1247711	747555	-348.0724	218	-51.0000	-363.5421	76533897	612.83%

Elapsed time = 2544.68 sec. (1582125.76 ticks, tree = 66528.97 MB, solutions = 37)

Nodefile size = 64413.10 MB (56160.38 MB after compression)

1250771	749319	-339.8761	240	-51.0000	-363.5421	76568677	612.83%
1253428	749875	-124.3338	66	-51.0000	-363.5421	76582998	612.83%
1255484	752046	-313.9867	199	-51.0000	-363.5421	76616090	612.83%
1257505	755090	-350.9171	242	-51.0000	-363.5421	76681612	612.83%
1260061	756810	-221.2307	125	-51.0000	-363.5421	76709254	612.83%
1262802	758964	-239.4641	151	-51.0000	-363.5421	76823900	612.83%
1266020	760556	-67.0000	27	-51.0000	-363.5421	76853753	612.83%
1268615	762001	-360.4381	313	-51.0000	-363.5421	76882255	612.83%
1270717	764173	-272.7824	204	-51.0000	-363.5421	76921149	612.83%
1273013	764966	-349.1395	221	-51.0000	-363.5421	76934663	612.83%

Elapsed time = 2612.51 sec. (1620287.49 ticks, tree = 68888.14 MB, solutions = 37)

Nodefile size = 66752.03 MB (58218.78 MB after compression)

1274910	766152	-207.9977	119	-51.0000	-363.5421	76966938	612.83%
1276895	767806	-303.6618	187	-51.0000	-363.5421	77002202	612.83%
1279341	770025	-65.0000	32	-51.0000	-363.5421	77074936	612.83%
1281908	771422	-347.6908	259	-51.0000	-363.5421	77108674	612.83%

1284101	773070	-177.3216	96	-51.0000	-363.5421	77190936	612.83%
1286070	776897	-256.3404	143	-51.0000	-363.5421	77312454	612.83%
1287980	777168	-310.7619	207	-51.0000	-363.5421	77318855	612.83%
1290484	778922	-188.5016	112	-51.0000	-363.5421	77387526	612.83%
1293069	781489	-349.1820	220	-51.0000	-363.5421	77460748	612.83%
1296398	783285	-98.9651	55	-51.0000	-363.4446	77499923	612.64%

Elapsed time = 2669.17 sec. (1658442.71 ticks, tree = 70071.06 MB, solutions = 37)

Nodefile size = 67958.53 MB (59253.49 MB after compression)

1299543	786097	-336.3001	223	-51.0000	-363.4300	77552721	612.61%
1303025	787564	-273.5397	156	-51.0000	-363.4226	77573312	612.59%
1306367	789080	-165.2899	99	-51.0000	-363.4165	77602295	612.58%
1309683	790823	-118.8209	58	-51.0000	-363.4165	77631755	612.58%
1312927	792605	-219.8976	134	-51.0000	-363.4015	77657849	612.55%
1315995	795803	-116.3636	55	-51.0000	-363.3930	77719274	612.54%
1318964	798022	-189.8624	104	-51.0000	-363.3865	77756398	612.52%
1322287	801721	-105.9631	46	-51.0000	-363.3788	77824176	612.51%
1325904	803857	-200.5226	112	-51.0000	-363.3710	77862574	612.49%
1328164	806865	-213.8837	137	-51.0000	-363.3676	77911316	612.49%

Elapsed time = 2740.62 sec. (1696601.21 ticks, tree = 73281.65 MB, solutions = 37)

Nodefile size = 71175.40 MB (62084.28 MB after compression)

1330725	808432	-253.4076	158	-51.0000	-363.3637	77946512	612.48%
1333424	811368	-83.0000	38	-51.0000	-363.3566	77996494	612.46%
1335998	812588	-207.2111	132	-51.0000	-363.3501	78032150	612.45%
1338136	815838	-262.3210	156	-51.0000	-363.3501	78120654	612.45%
1340355	816178	-79.0000	32	-51.0000	-363.3501	78126079	612.45%
1342750	818717	-179.9456	121	-51.0000	-363.3501	78197929	612.45%
1344977	819655	-64.0000	23	-51.0000	-363.3495	78246621	612.45%
1347424	820968	-220.0034	147	-51.0000	-363.3341	78263692	612.42%
1350340	824500	-337.0347	270	-51.0000	-363.3323	78361658	612.42%
1353068	826086	-218.3378	126	-51.0000	-363.3240	78392139	612.40%

Elapsed time = 2808.76 sec. (1734769.59 ticks, tree = 75570.94 MB, solutions = 37)

Nodefile size = 73464.03 MB (64075.14 MB after compression)

1354929	828737	-249.5033	167	-51.0000	-363.3240	78453276	612.40%
1357188	828878	-58.0000	31	-51.0000	-363.3240	78474265	612.40%
1360989	829429	-120.5256	54	-51.0000	-363.3071	78489831	612.37%
1363767	832535	-233.8816	139	-51.0000	-363.3009	78554923	612.35%
1366058	836220	-353.4513	283	-51.0000	-363.3009	78615429	612.35%
1368168	836811	-168.0117	96	-51.0000	-363.2916	78636188	612.34%
1370621	838448	-150.5858	71	-51.0000	-363.2852	78668575	612.32%

1373310	840832	-192.6639	118	-51.0000	-363.2852	78723896	612.32%
1376021	842744	-116.0000	61	-51.0000	-363.2846	78755757	612.32%
1378544	843891	-303.4139	177	-51.0000	-363.2846	78798222	612.32%

Elapsed time = 2876.38 sec. (1772947.12 ticks, tree = 78009.93 MB, solutions = 37)

Nodefile size = 75864.31 MB (66187.42 MB after compression)

1380852	844714	-104.8889	54	-51.0000	-363.2846	78805091	612.32%
1382951	849567	-266.2673	184	-51.0000	-363.2846	78916286	612.32%
1385337	849918	-354.9558	288	-51.0000	-363.2591	78921259	612.27%
1387121	850980	-220.9833	127	-51.0000	-363.2554	78943781	612.27%
1388896	853784	-311.5685	191	-51.0000	-363.2506	79021815	612.26%
1390673	854058	-319.6543	213	-51.0000	-363.2506	79027093	612.26%
1392561	856415	-215.9617	135	-51.0000	-363.2506	79109073	612.26%
1394362	856773	-336.7539	231	-51.0000	-363.2487	79114447	612.25%
1396269	858330	-314.7448	196	-51.0000	-363.2487	79170234	612.25%
1398329	859496	-278.4522	165	-51.0000	-363.2457	79188574	612.25%

Elapsed time = 2939.17 sec. (1811109.90 ticks, tree = 80161.14 MB, solutions = 37)

Nodefile size = 77996.90 MB (68060.51 MB after compression)

1400362	861629	-94.8928	60	-51.0000	-363.2457	79278236	612.25%
1402182	863414	-210.8249	123	-51.0000	-363.2457	79322997	612.25%
1403647	864988	-356.3679	277	-51.0000	-363.2134	79365317	612.18%
1405435	865338	-146.7865	78	-51.0000	-363.2088	79370410	612.17%
1406589	867010	-359.7785	358	-51.0000	-363.2088	79449597	612.17%
1407357	867801	-175.8509	103	-51.0000	-363.2088	79473735	612.17%
1408480	869141	-357.7632	286	-51.0000	-363.2060	79559155	612.17%
1409072	869005	-292.4575	185	-51.0000	-363.2030	79576725	612.16%
1409931	870613	-317.9388	207	-51.0000	-363.2030	79678668	612.16%
1410846	871221	-300.5855	191	-51.0000	-363.2030	79725732	612.16%

Elapsed time = 3001.29 sec. (1849268.80 ticks, tree = 81426.69 MB, solutions = 37)

Nodefile size = 79340.13 MB (69235.09 MB after compression)

1412565	871839	-198.5468	146	-51.0000	-363.1993	79794879	612.16%
1414222	872610	-355.1605	245	-51.0000	-363.1976	79824561	612.15%
1416451	873746	-136.8493	75	-51.0000	-363.1940	79887929	612.15%
1418802	875876	-262.7594	150	-51.0000	-363.1940	79985747	612.15%
1420667	877986	-120.8353	67	-51.0000	-363.1940	80044684	612.15%
1422897	879153	-68.0000	29	-51.0000	-363.1912	80105755	612.14%
1424675	880527	-102.5792	49	-51.0000	-363.1912	80124655	612.14%
1426442	881127	-214.1609	129	-51.0000	-363.1912	80157146	612.14%
1428467	883401	-182.3962	109	-51.0000	-363.1707	80225126	612.10%
1430584	885240	-157.4481	95	-51.0000	-363.1701	80286129	612.10%

Elapsed time = 3060.58 sec. (1887439.56 ticks, tree = 82485.78 MB, solutions = 37)

Nodefile size = 80396.88 MB (70146.52 MB after compression)

1431870	886774	-284.1700	176	-51.0000	-363.1701	80338190	612.10%
1433655	887329	-192.0080	106	-51.0000	-363.1701	80363893	612.10%
1435683	887996	-221.9601	130	-51.0000	-363.1701	80378007	612.10%
1437806	890614	-173.4306	90	-51.0000	-363.1701	80521614	612.10%
1439772	890705	-346.6358	222	-51.0000	-363.1634	80541956	612.09%
1441413	894088	-326.9882	225	-51.0000	-363.1634	80639579	612.09%
1443300	893400	-70.0000	28	-51.0000	-363.1542	80610357	612.07%
1445035	896243	-208.7591	119	-51.0000	-363.1453	80712144	612.05%
1446281	897127	-197.6167	112	-51.0000	-363.1419	80754887	612.04%
1447762	898657	-91.8217	40	-51.0000	-363.1419	80807489	612.04%

Elapsed time = 3123.21 sec. (1925598.32 ticks, tree = 83544.11 MB, solutions = 37)

Nodefile size = 81452.99 MB (71050.24 MB after compression)

1449076	899736	-322.1691	203	-51.0000	-363.1419	80869707	612.04%
1450689	899823	-232.5075	136	-51.0000	-363.1318	80862276	612.02%
1452809	901413	-225.7858	139	-51.0000	-363.1314	80941867	612.02%
1454893	903274	-358.8973	388	-51.0000	-363.1282	81042552	612.02%
1457193	905392	-173.3702	95	-51.0000	-363.1282	81100657	612.02%
1459630	905749	-248.7621	160	-51.0000	-363.1229	81106133	612.01%
1462412	908405	-327.4000	317	-51.0000	-363.1193	81191230	612.00%
1465284	910018	-269.0734	163	-51.0000	-363.1193	81234773	612.00%
1467995	911192	-127.3125	68	-51.0000	-363.1193	81272725	612.00%
1470171	915051	-214.2308	131	-51.0000	-363.1193	81338792	612.00%

Elapsed time = 3185.87 sec. (1963768.00 ticks, tree = 85097.25 MB, solutions = 37)

Nodefile size = 82996.73 MB (72399.07 MB after compression)

1472123	915840	-61.2873	25	-51.0000	-363.0963	81359705	611.95%
1474944	917352	-150.6935	103	-51.0000	-363.0963	81406102	611.95%
1477413	918699	-206.5598	113	-51.0000	-363.0963	81437361	611.95%
1480258	921424	-164.2992	111	-51.0000	-363.0819	81482776	611.93%
1482580	921768	-268.5418	165	-51.0000	-363.0819	81488923	611.93%
1485060	924969	-143.3574	86	-51.0000	-363.0736	81569087	611.91%
1487196	925334	-228.5638	136	-51.0000	-363.0736	81574482	611.91%
1488705	927554	-163.6258	104	-51.0000	-363.0736	81634624	611.91%
1490312	928638	-309.5214	190	-51.0000	-363.0642	81673852	611.89%
1491785	929380	-358.3981	270	-51.0000	-363.0642	81684293	611.89%

Elapsed time = 3251.39 sec. (2001939.32 ticks, tree = 87079.65 MB, solutions = 37)

Nodefile size = 84914.94 MB (74090.46 MB after compression)

1493123	932100	-361.1607	431	-51.0000	-363.0642	81782395	611.89%
---------	--------	-----------	-----	----------	-----------	----------	---------

1494409	933384	-226.9344	133	-51.0000	-363.0459	81815947	611.85%
1495695	934332	-340.4590	462	-51.0000	-363.0412	81876672	611.85%
1496875	935790	-83.4127	43	-51.0000	-363.0360	81930955	611.84%
1498611	936080	-340.2493	219	-51.0000	-363.0359	81938184	611.84%
1500352	938166	-344.8650	243	-51.0000	-363.0359	82058846	611.84%
1501886	939423	-214.2697	147	-51.0000	-363.0359	82133186	611.84%
1504021	940301	-358.9608	298	-51.0000	-363.0299	82162519	611.82%
1505915	940547	-111.5071	60	-51.0000	-363.0219	82169070	611.81%
1508308	942596	-290.0645	164	-51.0000	-363.0181	82259386	611.80%

Elapsed time = 3310.11 sec. (2040140.88 ticks, tree = 87780.04 MB, solutions = 37)

Nodefile size = 85628.88 MB (74705.56 MB after compression)

1509877	943462	-362.1250	345	-51.0000	-363.0181	82279281	611.80%
1511487	944392	-250.2819	153	-51.0000	-363.0181	82311234	611.80%
1513253	946218	-108.1399	52	-51.0000	-363.0181	82347204	611.80%
1515387	948834	-189.7285	105	-51.0000	-363.0181	82452582	611.80%
1517182	949556	-75.4540	32	-51.0000	-363.0181	82477198	611.80%
1519399	949444	infeasible		-51.0000	-363.0181	82514269	611.80%
1521136	952636	-323.2530	196	-51.0000	-363.0181	82576083	611.80%
1522540	953923	-233.5832	139	-51.0000	-363.0181	82642685	611.80%
1524673	955141	-324.2844	195	-51.0000	-363.0181	82669446	611.80%
1527134	957400	-270.0783	153	-51.0000	-363.0181	82761101	611.80%

Elapsed time = 3375.30 sec. (2078300.10 ticks, tree = 89832.75 MB, solutions = 37)

Nodefile size = 87735.09 MB (76556.10 MB after compression)

1529246	958614	-201.0558	130	-51.0000	-362.9817	82793638	611.73%
1531688	958996	-161.7118	88	-51.0000	-362.9817	82798680	611.73%
1533866	961145	-311.7875	206	-51.0000	-362.9680	82872188	611.70%
1536179	963907	-186.8495	106	-51.0000	-362.9665	82919347	611.70%
1538696	963676	-352.0906	262	-51.0000	-362.9639	82919358	611.69%
1541291	967962	-352.4511	278	-51.0000	-362.9596	83060386	611.69%
1543733	968242	-357.3868	235	-51.0000	-362.9596	83066483	611.69%
1546056	969753	-131.0000	70	-51.0000	-362.9596	83094379	611.69%
1548313	973663	-358.5955	292	-51.0000	-362.9596	83188282	611.69%
1550146	974076	-337.6329	235	-51.0000	-362.9596	83193612	611.69%

Elapsed time = 3438.78 sec. (2116458.32 ticks, tree = 91621.92 MB, solutions = 37)

Nodefile size = 89472.03 MB (78071.33 MB after compression)

1552066	975198	-257.4841	153	-51.0000	-362.9596	83217109	611.69%
1554054	978192	-362.7689	413	-51.0000	-362.9596	83329259	611.69%
1555755	978851	-143.0000	75	-51.0000	-362.9596	83355737	611.69%
1557601	980081	-111.0000	62	-51.0000	-362.9596	83402580	611.69%

1559461	980883	-65.5000	31	-51.0000	-362.9596	83416493	611.69%
1561060	981959	-95.2712	54	-51.0000	-362.9596	83457941	611.69%
1563252	984242	-362.0169	413	-51.0000	-362.9596	83525920	611.69%
1565359	985051	-357.4587	294	-51.0000	-362.9233	83550747	611.61%
1567576	986032	-149.0000	85	-51.0000	-362.9233	83574660	611.61%
1569513	987717	-279.5274	163	-51.0000	-362.9233	83634697	611.61%

Elapsed time = 3504.02 sec. (2154615.60 ticks, tree = 93479.99 MB, solutions = 37)

Nodefile size = 91371.26 MB (79754.57 MB after compression)

1571974	990722	-362.4365	380	-51.0000	-362.9233	83694670	611.61%
1573897	992646	-68.4032	28	-51.0000	-362.9114	83746398	611.59%
1575392	992729	-330.2044	206	-51.0000	-362.9114	83779146	611.59%
1577047	993978	-358.7650	359	-51.0000	-362.9057	83809952	611.58%
1578890	995730	-127.0000	61	-51.0000	-362.8989	83877449	611.57%
1580686	998467	-287.4760	169	-51.0000	-362.8989	83985755	611.57%
1582486	998774	-184.8990	96	-51.0000	-362.8989	83990762	611.57%
1584991	998842	-356.8110	390	-51.0000	-362.8859	84005528	611.54%
1587493	999579	-77.0000	35	-51.0000	-362.8859	84044828	611.54%
1589567	1002790	-131.3414	85	-51.0000	-362.8732	84134590	611.52%

Elapsed time = 3565.26 sec. (2192780.39 ticks, tree = 94791.64 MB, solutions = 37)

Nodefile size = 92672.34 MB (80888.26 MB after compression)

1592670	1004787	-297.9113	190	-51.0000	-362.8725	84180248	611.51%
1595731	1005466	-95.7042	50	-51.0000	-362.8672	84210720	611.50%
1599386	1006137	-255.3546	166	-51.0000	-362.8664	84220217	611.50%
1602374	1011809	-258.4069	167	-51.0000	-362.8652	84331862	611.50%
1604997	1010524	-360.4317	388	-51.0000	-362.8581	84307783	611.49%
1607859	1015041	-290.6650	167	-51.0000	-362.8530	84391761	611.48%
1610787	1017262	-348.8495	243	-51.0000	-362.8508	84436563	611.47%
1613632	1021191	-135.1203	72	-51.0000	-362.8508	84518537	611.47%
1615516	1020433	-216.2144	140	-51.0000	-362.8410	84504224	611.45%
1617251	1022386	-270.4581	162	-51.0000	-362.8333	84547261	611.44%

Elapsed time = 3635.70 sec. (2230941.04 ticks, tree = 97508.37 MB, solutions = 37)

Nodefile size = 95355.95 MB (83241.89 MB after compression)

1619480	1026326	-106.0000	53	-51.0000	-362.8282	84646400	611.43%
1621653	1026348	-355.3272	241	-51.0000	-362.8246	84638448	611.42%
1624106	1028420	-165.5907	118	-51.0000	-362.8246	84737509	611.42%
1626046	1031119	-197.2253	111	-51.0000	-362.8246	84803674	611.42%
1628354	1031483	-272.8441	187	-51.0000	-362.8182	84808486	611.41%
1630900	1033785	-136.0000	74	-51.0000	-362.8149	84879529	611.40%
1633280	1035225	-178.0256	106	-51.0000	-362.8095	84897759	611.39%

1635519	1036730	-352.3538	246	-51.0000	-362.8073	84951001	611.39%
1637650	1037407	-127.6583	75	-51.0000	-362.8062	84968280	611.38%
1640581	1037971	-353.0674	237	-51.0000	-362.8062	84995175	611.38%

Elapsed time = 3699.91 sec. (2269105.18 ticks, tree = 99299.35 MB, solutions = 37)

Nodefile size = 97165.67 MB (84826.35 MB after compression)

1643113	1042917	-123.0000	65	-51.0000	-362.7964	85104583	611.37%
1645682	1045072	-98.0000	43	-51.0000	-362.7915	85155504	611.36%
1647968	1044321	-247.7274	144	-51.0000	-362.7915	85146785	611.36%
1649894	1048126	-262.8989	152	-51.0000	-362.7845	85227817	611.34%
1651814	1049151	-186.8173	101	-51.0000	-362.7845	85253195	611.34%
1653327	1049473	-356.1169	318	-51.0000	-362.7795	85258164	611.33%
1655002	1053056	-355.6073	278	-51.0000	-362.7795	85359222	611.33%
1656520	1053373	-135.0000	67	-51.0000	-362.7795	85407938	611.33%
1658361	1054656	-101.0000	57	-51.0000	-362.7795	85449771	611.33%
1659888	1055695	-256.1699	146	-51.0000	-362.7795	85516761	611.33%

Elapsed time = 3765.79 sec. (2307266.26 ticks, tree = 101441.55 MB, solutions = 37)

Nodefile size = 99318.60 MB (86720.36 MB after compression)

1661149	1056024	-127.0000	72	-51.0000	-362.7795	85521668	611.33%
1662587	1057258	-353.0349	265	-51.0000	-362.7795	85568921	611.33%
1663677	1059201	-312.0521	192	-51.0000	-362.7795	85649252	611.33%
1665018	1060718	-346.8945	253	-51.0000	-362.7795	85699888	611.33%
1667093	1061145	-233.1731	124	-51.0000	-362.7795	85762078	611.33%
1668662	1063518	-185.8541	99	-51.0000	-362.7540	85814491	611.28%
1670566	1063921	-292.5807	179	-51.0000	-362.7471	85869834	611.27%
1673003	1064269	-93.0000	51	-51.0000	-362.7471	85876458	611.27%
1674618	1066561	-318.0390	230	-51.0000	-362.7460	85962658	611.27%
1676422	1066460	-355.5526	277	-51.0000	-362.7419	85956170	611.26%

Elapsed time = 3822.33 sec. (2345444.75 ticks, tree = 102413.17 MB, solutions = 37)

Nodefile size = 100313.89 MB (87590.71 MB after compression)

1678428	1069357	-120.0000	61	-51.0000	-362.7356	86082885	611.25%
1680900	1070444	-127.0000	79	-51.0000	-362.7356	86127269	611.25%
1684018	1072004	-195.2310	115	-51.0000	-362.7356	86162816	611.25%
1686950	1074672	-198.9341	121	-51.0000	-362.7213	86217540	611.22%
1688817	1075569	-344.0144	261	-51.0000	-362.7205	86234976	611.22%
1690232	1077618	-349.8671	358	-51.0000	-362.7205	86267376	611.22%
1692562	1078890	-230.6096	136	-51.0000	-362.7201	86305304	611.22%
1694242	1081904	-93.0000	43	-51.0000	-362.7066	86455402	611.19%
1695461	1080671	-211.5977	136	-51.0000	-362.7044	86405574	611.19%
1697499	1082638	-347.2280	227	-51.0000	-362.7018	86477065	611.18%

Elapsed time = 3887.72 sec. (2383611.26 ticks, tree = 103831.56 MB, solutions = 37)

Nodefile size = 101675.30 MB (88760.66 MB after compression)

1699409	1083919	-347.0445	252	-51.0000	-362.7018	86523820	611.18%
1701903	1087178	-164.0000	86	-51.0000	-362.6903	86637076	611.16%
1705244	1087473	-111.0000	54	-51.0000	-362.6903	86641463	611.16%
1708528	1091154	-351.8627	327	-51.0000	-362.6816	86710675	611.14%
1711345	1091455	-305.5254	226	-51.0000	-362.6810	86716115	611.14%
1714058	1093531	-125.0000	68	-51.0000	-362.6810	86742733	611.14%
1716044	1093882	-270.4370	177	-51.0000	-362.6810	86747997	611.14%
1717546	1096977	-165.0000	85	-51.0000	-362.6732	86820470	611.12%
1718830	1098665	-355.7772	430	-51.0000	-362.6672	86862102	611.11%
1720069	1100901	-135.0000	80	-51.0000	-362.6656	86925829	611.11%

Elapsed time = 3953.78 sec. (2421784.63 ticks, tree = 106417.14 MB, solutions = 37)

Nodefile size = 104319.82 MB (91096.33 MB after compression)

1721339	1101680	-117.0000	56	-51.0000	-362.6656	86974958	611.11%
1722319	1102272	-218.3016	136	-51.0000	-362.6596	87030638	611.10%
1723245	1103269	-275.9999	162	-51.0000	-362.6596	87056591	611.10%
1724223	1103312	-96.0000	46	-51.0000	-362.6596	87073826	611.10%
1725584	1104731	-177.3169	117	-51.0000	-362.6596	87136461	611.10%
1726911	1106217	-188.3971	143	-51.0000	-362.6574	87241554	611.09%
1728561	1106444	-176.0748	89	-51.0000	-362.6574	87287791	611.09%
1730521	1108181	-327.5582	201	-51.0000	-362.6574	87316342	611.09%
1732599	1108551	-74.0000	35	-51.0000	-362.6574	87322004	611.09%
1734632	1110364	-337.1043	250	-51.0000	-362.6574	87399542	611.09%

Elapsed time = 4014.37 sec. (2459957.65 ticks, tree = 107224.35 MB, solutions = 37)

Nodefile size = 105067.96 MB (91749.43 MB after compression)

1737166	1112205	-193.2310	114	-51.0000	-362.6459	87446556	611.07%
1739419	1112996	-173.7686	92	-51.0000	-362.6459	87486378	611.07%
1741249	1114769	-347.1913	214	-51.0000	-362.6401	87522106	611.06%
1743358	1115064	-192.2962	107	-51.0000	-362.6384	87515063	611.06%
1744968	1118074	-328.4236	210	-51.0000	-362.6359	87592968	611.05%
1746865	1119076	-191.1902	112	-51.0000	-362.6287	87643898	611.04%
1748365	1121241	-81.0000	37	-51.0000	-362.6225	87699408	611.02%
1750085	1121481	-324.0813	201	-51.0000	-362.6225	87705667	611.02%
1752508	1124145	-60.0000	19	-51.0000	-362.6205	87805130	611.02%
1755390	1123880	infeasible		-51.0000	-362.6205	87785448	611.02%

Elapsed time = 4081.48 sec. (2498117.32 ticks, tree = 109186.03 MB, solutions = 37)

Nodefile size = 107010.45 MB (93469.83 MB after compression)

1757833	1127056	-215.2144	120	-51.0000	-362.6205	87862964	611.02%
---------	---------	-----------	-----	----------	-----------	----------	---------

1759906	1130330	-126.6583	64	-51.0000	-362.6205	87942162	611.02%
1761605	1131195	-158.9016	98	-51.0000	-362.6205	87952404	611.02%
1763935	1132655	-353.8302	251	-51.0000	-362.6059	88000738	610.99%
1766383	1135120	-358.4267	298	-51.0000	-362.5995	88069117	610.98%
1769340	1134674	-351.0958	290	-51.0000	-362.5959	88065237	610.97%
1770949	1137435	-354.5922	298	-51.0000	-362.5959	88133999	610.97%
1772901	1138195	-317.6154	230	-51.0000	-362.5891	88152987	610.96%
1774852	1138989	-319.6824	186	-51.0000	-362.5858	88197180	610.95%
1777181	1141667	-228.0794	142	-51.0000	-362.5778	88238065	610.94%

Elapsed time = 4149.17 sec. (2536284.58 ticks, tree = 111606.07 MB, solutions = 37)

Nodefile size = 109464.13 MB (95628.95 MB after compression)

1779471	1142822	-356.9082	300	-51.0000	-362.5778	88301076	610.94%
1781560	1145486	-345.4786	300	-51.0000	-362.5709	88354961	610.92%
1783964	1146383	-142.0000	75	-51.0000	-362.5683	88384760	610.92%
1786599	1149409	-156.9016	93	-51.0000	-362.5632	88449409	610.91%
1788528	1150674	-286.3848	175	-51.0000	-362.5575	88476984	610.90%
1790262	1152003	-152.8889	82	-51.0000	-362.5554	88530369	610.89%
1792157	1154316	-312.5151	205	-51.0000	-362.5519	88608998	610.89%
1794239	1155329	-81.4111	38	-51.0000	-362.5519	88632851	610.89%
1796795	1155687	-226.8191	121	-51.0000	-362.5490	88638195	610.88%
1798889	1157971	-242.5105	141	-51.0000	-362.5490	88695455	610.88%

Elapsed time = 4216.83 sec. (2574439.80 ticks, tree = 113971.04 MB, solutions = 37)

Nodefile size = 111820.30 MB (97713.78 MB after compression)

1801257	1160786	-326.2609	217	-51.0000	-362.5476	88785387	610.88%
1803904	1160584	-197.5598	116	-51.0000	-362.5476	88778037	610.88%
1806916	1163684	-112.0000	59	-51.0000	-362.5476	88855097	610.88%
1809860	1164061	-164.0000	89	-51.0000	-362.5359	88860117	610.85%
1812427	1167790	-353.8110	266	-51.0000	-362.5338	88946081	610.85%
1813938	1169137	-355.4961	313	-51.0000	-362.5313	88984565	610.85%
1815156	1169515	-95.0000	50	-51.0000	-362.5280	88989276	610.84%
1816884	1172293	-109.0000	56	-51.0000	-362.5280	89054712	610.84%
1819029	1172222	-359.4854	274	-51.0000	-362.5165	89088292	610.82%
1820702	1175341	-101.0000	59	-51.0000	-362.5139	89172352	610.81%

Elapsed time = 4283.62 sec. (2612607.16 ticks, tree = 116452.58 MB, solutions = 37)

Nodefile size = 114352.58 MB (99959.10 MB after compression)

1821717	1175520	-316.6016	211	-51.0000	-362.5105	89178976	610.80%
1822802	1175743	-356.9446	302	-51.0000	-362.5105	89208707	610.80%
1825186	1177728	-359.8573	307	-51.0000	-362.5019	89292196	610.79%
1828466	1178524	-311.3564	177	-51.0000	-362.4955	89352642	610.78%

1831697	1179704	-254.7012	147	-51.0000	-362.4947	89383091	610.77%
1833684	1185363	-346.5593	242	-51.0000	-362.4913	89494458	610.77%
1835958	1185747	-65.0000	35	-51.0000	-362.4905	89499237	610.77%
1838222	1187800	-349.6080	333	-51.0000	-362.4905	89556219	610.77%
1840349	1189274	-190.9810	106	-51.0000	-362.4891	89593726	610.76%
1842574	1190077	-131.0000	66	-51.0000	-362.4891	89607149	610.76%

Elapsed time = 4348.73 sec. (2650771.96 ticks, tree = 118254.77 MB, solutions = 37)

Nodefile size = 116123.16 MB (101513.80 MB after compression)

1844137	1190950	-337.9764	239	-51.0000	-362.4773	89621043	610.74%
1845868	1194067	-163.0000	91	-51.0000	-362.4773	89702644	610.74%
1847428	1194442	infeasible		-51.0000	-362.4773	89707519	610.74%
1848545	1195399	-338.9939	269	-51.0000	-362.4715	89749797	610.73%
1849976	1197189	-312.5297	186	-51.0000	-362.4715	89842164	610.73%
1852539	1197765	-172.7495	102	-51.0000	-362.4715	89864964	610.73%
1855071	1199117	-174.5109	95	-51.0000	-362.4673	89907349	610.72%
1857581	1199427	-79.0000	33	-51.0000	-362.4673	89912585	610.72%
1860128	1204628	-139.0000	78	-51.0000	-362.4673	90040175	610.72%
1862424	1204806	-92.0000	38	-51.0000	-362.4673	90046923	610.72%

Elapsed time = 4412.31 sec. (2688942.58 ticks, tree = 119835.92 MB, solutions = 37)

Nodefile size = 117703.76 MB (102891.72 MB after compression)

1864012	1205110	-334.7566	231	-51.0000	-362.4673	90052032	610.72%
1865715	1206430	-248.0777	143	-51.0000	-362.4673	90089209	610.72%
1867880	1208849	-360.7170	315	-51.0000	-362.4673	90124930	610.72%
1869710	1210002	-253.9861	148	-51.0000	-362.4461	90184073	610.68%
1871605	1212587	-271.6025	172	-51.0000	-362.4461	90320048	610.68%
1873471	1213584	-156.0000	98	-51.0000	-362.4461	90347220	610.68%
1875249	1215234	-346.3866	246	-51.0000	-362.4461	90387853	610.68%
1876789	1216880	-136.0000	70	-51.0000	-362.4317	90467261	610.65%
1878428	1216583	-360.9537	396	-51.0000	-362.4300	90462464	610.65%
1879496	1218118	-271.7228	172	-51.0000	-362.4300	90528095	610.65%

Elapsed time = 4475.52 sec. (2727125.87 ticks, tree = 121765.60 MB, solutions = 37)

Nodefile size = 119662.56 MB (104632.68 MB after compression)

1879927	1219133	-331.2837	228	-51.0000	-362.4208	90557727	610.63%
1881200	1219700	-354.6783	283	-51.0000	-362.4206	90586562	610.63%
1882723	1221442	-339.9416	253	-51.0000	-362.4201	90642101	610.63%
1883876	1222681	-353.1613	241	-51.0000	-362.4167	90733610	610.62%
1884976	1222419	-356.8062	237	-51.0000	-362.4167	90727814	610.62%
1885808	1224246	-174.6478	96	-51.0000	-362.4148	90820647	610.62%
1887127	1224675	-96.0000	47	-51.0000	-362.4143	90824870	610.62%

1888119	1226301	-354.7976	297	-51.0000	-362.4143	90942207	610.62%
1889542	1227449	-292.9870	200	-51.0000	-362.4143	90980258	610.62%
1891491	1227878	-201.0640	120	-51.0000	-362.4065	90984614	610.60%

Elapsed time = 4533.80 sec. (2765317.84 ticks, tree = 122538.28 MB, solutions = 37)

Nodefile size = 120337.04 MB (105215.91 MB after compression)

1893792	1227996	-162.0000	91	-51.0000	-362.4065	91035314	610.60%
1896328	1230354	-246.3380	142	-51.0000	-362.4053	91099099	610.60%
1898667	1232386	-348.0567	224	-51.0000	-362.4034	91151758	610.59%
1900424	1234216	-202.3876	116	-51.0000	-362.3982	91202203	610.58%
1902409	1235516	-146.0000	82	-51.0000	-362.3963	91228751	610.58%
1904297	1236559	-294.4467	221	-51.0000	-362.3939	91256879	610.58%
1905946	1237273	-169.0000	91	-51.0000	-362.3926	91297562	610.57%
1908247	1238212	-359.1546	327	-51.0000	-362.3923	91327857	610.57%
1910238	1241889	-93.0000	43	-51.0000	-362.3840	91454153	610.56%
1912123	1242108	-253.7352	161	-51.0000	-362.3840	91443338	610.56%

Elapsed time = 4597.07 sec. (2803482.88 ticks, tree = 124122.99 MB, solutions = 37)

Nodefile size = 121915.48 MB (106601.21 MB after compression)

1913823	1244810	-293.5904	165	-51.0000	-362.3840	91560056	610.56%
1915261	1242831	-348.9132	224	-51.0000	-362.3840	91487721	610.56%
1917087	1245691	-357.7825	398	-51.0000	-362.3736	91596278	610.54%
1919377	1247859	-350.8520	301	-51.0000	-362.3727	91640207	610.53%
1920957	1249365	-349.1644	264	-51.0000	-362.3727	91704716	610.53%
1922448	1251919	-241.0255	135	-51.0000	-362.3727	91787046	610.53%
1924299	1251385	-139.0000	89	-51.0000	-362.3727	91745691	610.53%
1925928	1254633	-335.5796	212	-51.0000	-362.3727	91864879	610.53%
1927733	1253942	-248.7582	148	-51.0000	-362.3727	91863422	610.53%
1929800	1254954	-161.5000	98	-51.0000	-362.3727	91914276	610.53%

Elapsed time = 4665.96 sec. (2841657.11 ticks, tree = 125876.31 MB, solutions = 37)

Nodefile size = 123736.62 MB (108198.02 MB after compression)

1932915	1257567	-356.6930	308	-51.0000	-362.3727	91971230	610.53%
1935557	1257895	-235.1592	138	-51.0000	-362.3485	91975728	610.49%
1937930	1259507	-133.0000	76	-51.0000	-362.3470	92043433	610.48%
1939112	1262952	-268.7837	146	-51.0000	-362.3423	92119640	610.48%
1941120	1263559	-336.0770	228	-51.0000	-362.3380	92141033	610.47%
1943317	1266134	-78.0000	38	-51.0000	-362.3380	92201328	610.47%
1945343	1266470	-81.0000	35	-51.0000	-362.3380	92230682	610.47%
1947101	1268368	-140.3414	74	-51.0000	-362.3309	92291980	610.45%
1948917	1269242	-223.0782	125	-51.0000	-362.3309	92327936	610.45%
1950788	1269706	-57.0000	25	-51.0000	-362.3309	92331814	610.45%

Elapsed time = 4731.66 sec. (2879825.80 ticks, tree = 127608.51 MB, solutions = 37)

Nodefile size = 125386.48 MB (109656.60 MB after compression)

1952593	1272828	-163.0000	86	-51.0000	-362.3309	92422770	610.45%
1955188	1273992	-348.8436	266	-51.0000	-362.3309	92476904	610.45%
1957494	1273935	-179.5526	97	-51.0000	-362.3309	92458164	610.45%
1958998	1275371	-357.5784	362	-51.0000	-362.3309	92499131	610.45%
1960528	1278994	-183.2555	100	-51.0000	-362.3309	92595513	610.45%
1962501	1279390	-143.0000	79	-51.0000	-362.3309	92614185	610.45%
1963753	1279515	-181.5109	103	-51.0000	-362.3309	92622257	610.45%
1965266	1282516	-101.0000	45	-51.0000	-362.3309	92718093	610.45%
1966607	1283982	-337.5188	281	-51.0000	-362.3309	92799904	610.45%
1968190	1282938	-315.7303	204	-51.0000	-362.3309	92761351	610.45%

Elapsed time = 4795.38 sec. (2917991.21 ticks, tree = 129154.59 MB, solutions = 37)

Nodefile size = 127050.39 MB (111116.43 MB after compression)

1970548	1286382	-334.5310	206	-51.0000	-362.3309	92905255	610.45%
1972793	1287538	-249.0323	140	-51.0000	-362.3309	92945474	610.45%
1974081	1286958	-223.8645	140	-51.0000	-362.3309	92914229	610.45%
1975090	1289465	-301.1046	172	-51.0000	-362.3309	93018059	610.45%
1976045	1288874	-355.0767	231	-51.0000	-362.3309	92974520	610.45%
1977011	1292121	-361.9902	290	-51.0000	-362.3309	93088426	610.45%
1978501	1293014	-358.8972	302	-51.0000	-362.3309	93119372	610.45%
1980427	1292664	-196.8761	136	-51.0000	-362.3074	93113119	610.41%
1981659	1294107	-96.5679	44	-51.0000	-362.3074	93290304	610.41%
1982761	1295504	-257.0260	157	-51.0000	-362.3074	93318116	610.41%

Elapsed time = 4857.33 sec. (2956165.23 ticks, tree = 130261.01 MB, solutions = 37)

Nodefile size = 128107.74 MB (112032.73 MB after compression)

1984197	1296305	-228.6609	123	-51.0000	-362.3074	93336345	610.41%
1985492	1296197	-177.2575	101	-51.0000	-362.3074	93393577	610.41%
1986946	1298745	-119.0000	59	-51.0000	-362.3074	93460887	610.41%
1988715	1298777	-360.0183	343	-51.0000	-362.3074	93496963	610.41%
1990344	1301907	-355.9441	253	-51.0000	-362.3074	93607408	610.41%
1992706	1302258	-120.0000	62	-51.0000	-362.2874	93612558	610.37%
1995018	1304182	-225.4859	128	-51.0000	-362.2874	93683921	610.37%
1997601	1303431	-155.8836	85	-51.0000	-362.2874	93659824	610.37%
1999960	1306782	-179.7059	100	-51.0000	-362.2874	93741768	610.37%
2001806	1307133	-277.2162	157	-51.0000	-362.2534	93746417	610.30%

Elapsed time = 4924.31 sec. (2994343.65 ticks, tree = 131623.21 MB, solutions = 37)

Nodefile size = 129402.80 MB (113165.07 MB after compression)

2003608	1310011	cutoff		-51.0000	-362.2505	93852745	610.30%
---------	---------	--------	--	----------	-----------	----------	---------

2005876	1311929	-360.7152	340	-51.0000	-362.2420	93872511	610.28%
2007520	1314349	-361.2690	324	-51.0000	-362.2401	93949400	610.27%
2010050	1315474	-358.1109	347	-51.0000	-362.2348	93980261	610.26%
2012141	1314837	-121.0000	78	-51.0000	-362.2319	93967126	610.26%
2014117	1316900	-210.4484	137	-51.0000	-362.2319	94047490	610.26%
2016539	1320203	-301.7363	185	-51.0000	-362.2225	94177706	610.24%
2018363	1319559	-90.0000	35	-51.0000	-362.2210	94148369	610.24%
2019749	1322440	-109.2500	66	-51.0000	-362.2210	94261607	610.24%
2021774	1322860	-353.5627	239	-51.0000	-362.2210	94250994	610.24%

Elapsed time = 4990.11 sec. (3032511.21 ticks, tree = 133163.44 MB, solutions = 37)

Nodefile size = 130977.97 MB (114541.34 MB after compression)

2023532	1324766	-201.9114	114	-51.0000	-362.2210	94318165	610.24%
2025242	1327799	-353.9849	299	-51.0000	-362.2100	94439879	610.22%
2026170	1326401	-158.0000	94	-51.0000	-362.2098	94361700	610.22%
2026901	1329537	-287.7297	180	-51.0000	-362.2078	94472233	610.21%
2028133	1330094	-58.4032	26	-51.0000	-362.2078	94513762	610.21%
2029923	1331143	-65.4032	29	-51.0000	-362.2065	94618610	610.21%
2031440	1331107	-318.8100	196	-51.0000	-362.2065	94607266	610.21%
2032934	1332754	-265.4758	167	-51.0000	-362.2058	94669472	610.21%
2034589	1334098	-153.0000	81	-51.0000	-362.2021	94746572	610.20%
2036138	1334328	-243.3282	168	-51.0000	-362.2021	94751540	610.20%

Elapsed time = 5053.58 sec. (3070737.59 ticks, tree = 134474.79 MB, solutions = 37)

Nodefile size = 132323.41 MB (115713.57 MB after compression)

2038712	1333925	-78.5139	50	-51.0000	-362.2021	94744505	610.20%
2040376	1337795	infeasible		-51.0000	-362.1926	94873080	610.18%
2042249	1338711	-95.0000	53	-51.0000	-362.1918	94889167	610.18%
2043289	1339688	-196.9467	171	-51.0000	-362.1918	94930297	610.18%
2044354	1340831	-339.2202	210	-51.0000	-362.1918	94959160	610.18%
2045538	1341759	-316.6392	222	-51.0000	-362.1891	94978591	610.17%
2047212	1344458	-298.5127	183	-51.0000	-362.1891	95116161	610.17%
2048192	1343500	-315.8810	205	-51.0000	-362.1891	95075051	610.17%
2050162	1346617	-258.7810	146	-51.0000	-362.1891	95248518	610.17%
2052107	1347065	-209.5024	121	-51.0000	-362.1834	95285696	610.16%

Elapsed time = 5113.04 sec. (3108900.99 ticks, tree = 135743.20 MB, solutions = 37)

Nodefile size = 133587.80 MB (116827.78 MB after compression)

2054327	1347767	-179.1118	105	-51.0000	-362.1834	95328232	610.16%
2056734	1350306	-355.7437	324	-51.0000	-362.1701	95403631	610.14%
2057895	1349123	-285.9940	159	-51.0000	-362.1701	95387785	610.14%
2059298	1353824	-177.2253	104	-51.0000	-362.1701	95512857	610.14%

2061571	1354213	-235.3818	137	-51.0000	-362.1701	95517218	610.14%
2062721	1354490	-178.5830	101	-51.0000	-362.1701	95521973	610.14%
2063887	1354683	-359.1123	335	-51.0000	-362.1619	95552808	610.12%
2066171	1357304	-216.6867	128	-51.0000	-362.1588	95666338	610.12%
2068842	1358733	-338.9059	219	-51.0000	-362.1540	95690539	610.11%
2071197	1359079	-151.0000	86	-51.0000	-362.1505	95694410	610.10%

Elapsed time = 5183.43 sec. (3147063.12 ticks, tree = 137247.59 MB, solutions = 37)

Nodefile size = 135065.62 MB (118120.50 MB after compression)

2073960	1362024	infeasible		-51.0000	-362.1492	95770534	610.10%
2076419	1362562	-356.2101	269	-51.0000	-362.1458	95796757	610.09%
2078532	1365651	-229.5769	161	-51.0000	-362.1441	95842208	610.09%
2080519	1367251	-81.0000	42	-51.0000	-362.1406	95880610	610.08%
2082831	1367594	-215.7704	129	-51.0000	-362.1361	95886294	610.07%
2085431	1371304	-165.5000	93	-51.0000	-362.1349	95986118	610.07%
2087706	1371516	-311.5010	186	-51.0000	-362.1292	95993806	610.06%
2090838	1372474	-196.4747	136	-51.0000	-362.1266	96028276	610.05%
2093888	1376175	-268.9368	163	-51.0000	-362.1222	96150851	610.04%
2097037	1376513	-93.0000	48	-51.0000	-362.1195	96156174	610.04%

Elapsed time = 5252.06 sec. (3185223.84 ticks, tree = 139227.82 MB, solutions = 37)

Nodefile size = 137044.02 MB (119868.56 MB after compression)

2099977	1377822	-309.5230	191	-51.0000	-362.1194	96195538	610.04%
2102737	1383997	-216.9528	122	-51.0000	-362.1141	96313606	610.03%
2105386	1382784	-357.4801	259	-51.0000	-362.1109	96296349	610.02%
2107072	1384958	-243.3201	190	-51.0000	-362.1080	96353624	610.02%
2109715	1389908	-91.0000	41	-51.0000	-362.1018	96431278	610.00%
2112892	1389386	-237.2620	142	-51.0000	-362.1014	96428949	610.00%
2116344	1392877	-76.0000	37	-51.0000	-362.0952	96518047	609.99%
2119601	1396055	-239.8675	163	-51.0000	-362.0902	96571138	609.98%
2122701	1397588	-162.0000	92	-51.0000	-362.0878	96597500	609.98%
2125585	1399419	-107.0000	56	-51.0000	-362.0837	96623631	609.97%

Elapsed time = 5323.57 sec. (3223386.22 ticks, tree = 142291.72 MB, solutions = 37)

Nodefile size = 140136.28 MB (122587.56 MB after compression)

2127637	1401788	-98.9772	55	-51.0000	-362.0837	96673819	609.97%
2129462	1403019	-101.0000	53	-51.0000	-362.0777	96689106	609.96%
2130967	1403384	-180.0312	103	-51.0000	-362.0736	96694336	609.95%
2132398	1407650	-249.8456	150	-51.0000	-362.0716	96832474	609.94%
2134190	1407911	-259.4266	153	-51.0000	-362.0716	96840144	609.94%
2136287	1409592	-347.6216	223	-51.0000	-362.0716	96926465	609.94%
2138800	1410407	-107.4540	54	-51.0000	-362.0673	96963154	609.94%

2140533	1410269	-356.8189	314	-51.0000	-362.0623	96948737	609.93%
2141817	1413348	-314.1180	205	-51.0000	-362.0608	97027902	609.92%
2143712	1415198	-361.9409	378	-51.0000	-362.0608	97076664	609.92%

Elapsed time = 5387.73 sec. (3261572.73 ticks, tree = 144059.10 MB, solutions = 37)

Nodefile size = 141929.99 MB (124151.38 MB after compression)

2145303	1415470	-243.6396	140	-51.0000	-362.0548	97110721	609.91%
2147224	1415864	-346.9044	241	-51.0000	-362.0548	97087785	609.91%
2149106	1420398	-353.5467	272	-51.0000	-362.0502	97254799	609.90%
2151211	1420743	-195.9210	100	-51.0000	-362.0502	97260086	609.90%
2153737	1421083	-290.3449	215	-51.0000	-362.0502	97264392	609.90%
2155994	1424276	-173.5964	95	-51.0000	-362.0502	97359502	609.90%
2157424	1423832	-188.4168	116	-51.0000	-362.0502	97352625	609.90%
2158038	1425742	-231.6470	143	-51.0000	-362.0502	97404097	609.90%
2159500	1428123	-166.8025	112	-51.0000	-362.0502	97507292	609.90%
2161270	1427691	-233.3973	129	-51.0000	-362.0502	97484141	609.90%

Elapsed time = 5455.78 sec. (3299734.67 ticks, tree = 145843.98 MB, solutions = 37)

Nodefile size = 143666.89 MB (125689.91 MB after compression)

2163098	1428744	-310.5486	196	-51.0000	-362.0502	97518331	609.90%
2164625	1429691	-72.0000	29	-51.0000	-362.0502	97530450	609.90%
2166592	1430389	-262.9791	183	-51.0000	-362.0502	97612273	609.90%
2168915	1431725	-356.9467	343	-51.0000	-362.0502	97621871	609.90%
2170713	1434358	-258.5424	166	-51.0000	-362.0502	97704062	609.90%
2172463	1434714	-357.2590	274	-51.0000	-362.0502	97708705	609.90%
2174588	1437891	-344.3184	227	-51.0000	-362.0284	97794992	609.86%
2176317	1438251	-145.9176	80	-51.0000	-362.0284	97800479	609.86%
2177202	1438198	-333.6357	210	-51.0000	-362.0284	97815908	609.86%
2179165	1439916	-353.4805	424	-51.0000	-362.0284	97872104	609.86%

Elapsed time = 5521.06 sec. (3337907.15 ticks, tree = 147605.70 MB, solutions = 37)

Nodefile size = 145495.21 MB (127311.62 MB after compression)

2181324	1440866	-283.2540	176	-51.0000	-362.0284	97891679	609.86%
2182556	1442766	-158.4380	82	-51.0000	-362.0284	97941562	609.86%
2184500	1444493	-314.4764	222	-51.0000	-362.0284	97994623	609.86%
2186406	1444801	-224.1898	124	-51.0000	-362.0284	98013376	609.86%
2188871	1447973	-339.4077	214	-51.0000	-362.0284	98109830	609.86%
2190940	1449999	-290.2922	199	-51.0000	-361.9977	98233021	609.80%
2193520	1449073	-361.5637	435	-51.0000	-361.9977	98128061	609.80%
2196434	1451931	-314.6463	224	-51.0000	-361.9971	98276855	609.80%
2199381	1453652	-185.5070	97	-51.0000	-361.9971	98315726	609.80%
2201426	1454807	-117.0000	60	-51.0000	-361.9971	98362893	609.80%

Elapsed time = 5587.16 sec. (3376070.73 ticks, tree = 149537.40 MB, solutions = 37)

Nodefile size = 147385.66 MB (128968.65 MB after compression)

2204276	1456451	-225.5052	131	-51.0000	-361.9971	98391992	609.80%
2206451	1460380	-271.7723	166	-51.0000	-361.9904	98473773	609.79%
2209077	1462514	-361.1800	374	-51.0000	-361.9904	98527678	609.79%
2212458	1462879	-197.8400	113	-51.0000	-361.9809	98522461	609.77%
2215420	1467461	-259.2272	145	-51.0000	-361.9735	98624285	609.75%
2218387	1468647	-123.6583	67	-51.0000	-361.9723	98654373	609.75%
2221372	1471853	-350.9130	263	-51.0000	-361.9667	98702789	609.74%
2225070	1470218	-121.0000	64	-51.0000	-361.9636	98673690	609.73%
2228119	1471931	-201.6745	126	-51.0000	-361.9572	98699736	609.72%
2230869	1478734	-305.8009	190	-51.0000	-361.9524	98808636	609.71%

Elapsed time = 5661.26 sec. (3414225.98 ticks, tree = 153064.91 MB, solutions = 37)

Nodefile size = 150931.10 MB (132094.45 MB after compression)

2232763	1478343	-189.2024	111	-51.0000	-361.9492	98802610	609.70%
2234988	1480112	-312.4234	187	-51.0000	-361.9440	98828947	609.69%
2237925	1484751	-313.7701	180	-51.0000	-361.9388	98937741	609.68%
2240558	1486010	-87.0000	43	-51.0000	-361.9364	98964759	609.68%
2242539	1487781	-275.2400	187	-51.0000	-361.9309	99031246	609.67%
2245035	1487728	-219.6291	122	-51.0000	-361.9281	99003844	609.66%
2246529	1491934	-179.1228	94	-51.0000	-361.9255	99095848	609.66%
2247954	1491580	-351.5553	243	-51.0000	-361.9245	99089066	609.66%
2249384	1491991	-291.4054	198	-51.0000	-361.9245	99093972	609.66%
2250277	1494816	-344.8796	224	-51.0000	-361.9245	99189716	609.66%

Elapsed time = 5727.78 sec. (3452393.37 ticks, tree = 155438.60 MB, solutions = 37)

Nodefile size = 153292.60 MB (134181.29 MB after compression)

2251810	1495637	-112.0000	58	-51.0000	-361.9219	99235307	609.65%
2252985	1496849	-242.5733	165	-51.0000	-361.9219	99290827	609.65%
2253778	1497620	-345.1575	417	-51.0000	-361.9219	99333352	609.65%
2254166	1498135	-355.5885	305	-51.0000	-361.9157	99318850	609.64%
2254590	1497715	-156.6532	90	-51.0000	-361.9157	99344381	609.64%
2255374	1499614	-341.7492	301	-51.0000	-361.9143	99483056	609.64%
2256612	1499695	-256.9245	148	-51.0000	-361.9143	99461630	609.64%
2258131	1500600	-112.0000	59	-51.0000	-361.9143	99521815	609.64%
2259968	1502063	-74.0000	31	-51.0000	-361.9143	99596909	609.64%
2261589	1501021	-217.7357	145	-51.0000	-361.9143	99544139	609.64%

Elapsed time = 5789.89 sec. (3490642.38 ticks, tree = 156097.56 MB, solutions = 37)

Nodefile size = 153881.77 MB (134701.90 MB after compression)

2263540	1503689	-351.0484	271	-51.0000	-361.9143	99662117	609.64%
---------	---------	-----------	-----	----------	-----------	----------	---------

2265593	1505294	-257.5113	150	-51.0000	-361.9143	99690177	609.64%
2267732	1505621	-77.0000	31	-51.0000	-361.9143	99694602	609.64%
2269137	1508711	-65.5679	35	-51.0000	-361.9071	99827984	609.62%
2270876	1509033	-240.1320	134	-51.0000	-361.9071	99832992	609.62%
2272906	1509090	-314.8264	233	-51.0000	-361.9071	99822254	609.62%
2274629	1511601	-189.8173	101	-51.0000	-361.9002	99916173	609.61%
2275232	1511926	-147.0000	73	-51.0000	-361.9002	99933267	609.61%
2276649	1513432	infeasible		-51.0000	-361.9002	99961904	609.61%
2284484	1518423	-347.9256	237	-51.0000	-361.8881	1.00e+08	609.58%

Elapsed time = 5869.03 sec. (3540255.65 ticks, tree = 158053.08 MB, solutions = 37)

Nodefile size = 155868.06 MB (136448.34 MB after compression)

2293888	1523878	-359.4920	234	-51.0000	-361.8779	1.00e+08	609.56%
2300244	1531155	-143.7059	77	-51.0000	-361.8641	1.01e+08	609.54%
2305495	1533878	-312.1851	184	-51.0000	-361.8641	1.01e+08	609.54%
2310737	1538466	-359.9087	375	-51.0000	-361.8504	1.01e+08	609.51%
2319427	1544696	-187.5489	104	-51.0000	-361.8504	1.01e+08	609.51%
2329798	1548090	-351.7499	300	-51.0000	-361.8348	1.01e+08	609.48%
2340901	1560129	-303.0342	183	-51.0000	-361.8223	1.01e+08	609.46%
2352039	1567246	-183.5988	116	-51.0000	-361.8057	1.02e+08	609.42%
2362020	1573455	infeasible		-51.0000	-361.7982	1.02e+08	609.41%
2374073	1581038	-228.1897	133	-51.0000	-361.7858	1.02e+08	609.38%

Elapsed time = 6146.10 sec. (3692873.43 ticks, tree = 166163.23 MB, solutions = 37)

Nodefile size = 163955.13 MB (143560.66 MB after compression)

2383552	1588797	-311.2394	181	-51.0000	-361.7788	1.02e+08	609.37%
2391426	1596592	-100.0000	67	-51.0000	-361.7660	1.02e+08	609.35%
2399501	1604465	-353.4838	253	-51.0000	-361.7523	1.03e+08	609.32%
2406212	1609092	-187.2150	110	-51.0000	-361.7359	1.03e+08	609.29%
2414023	1613458	-99.6122	48	-51.0000	-361.7213	1.03e+08	609.26%
2423894	1621187	-346.7928	202	-51.0000	-361.7141	1.03e+08	609.24%
2432402	1623753	-181.7823	93	-51.0000	-361.7031	1.03e+08	609.22%
2441756	1636254	-339.1537	432	-51.0000	-361.6890	1.03e+08	609.19%
2448558	1638942	-357.1295	284	-51.0000	-361.6797	1.03e+08	609.18%
2454329	1646493	-207.1387	147	-51.0000	-361.6752	1.04e+08	609.17%

Elapsed time = 6408.01 sec. (3845475.54 ticks, tree = 173640.02 MB, solutions = 37)

Nodefile size = 171517.61 MB (150199.52 MB after compression)

2458581	1647298	-162.0000	89	-51.0000	-361.6752	1.04e+08	609.17%
2466444	1654276	-315.4581	198	-51.0000	-361.6752	1.04e+08	609.17%
2470355	1658826	-353.0127	356	-51.0000	-361.6752	1.04e+08	609.17%
2473616	1660657	-169.0000	93	-51.0000	-361.6752	1.04e+08	609.17%

2477627	1664097	-293.9190	163	-51.0000	-361.6752	1.05e+08	609.17%
2482952	1664957	-317.8915	228	-51.0000	-361.6626	1.05e+08	609.14%
2490159	1670626	-299.0124	185	-51.0000	-361.6626	1.05e+08	609.14%
2496441	1674807	-179.6537	100	-51.0000	-361.6489	1.05e+08	609.12%
2499857	1680158	-360.0926	333	-51.0000	-361.6489	1.05e+08	609.12%
2503644	1682909	-223.7228	137	-51.0000	-361.6489	1.05e+08	609.12%

Elapsed time = 6660.25 sec. (3998105.33 ticks, tree = 177713.50 MB, solutions = 37)

Nodefile size = 175582.00 MB (153769.34 MB after compression)

2508133	1685701	-351.1965	242	-51.0000	-361.6489	1.05e+08	609.12%
2513566	1689957	-315.2063	204	-51.0000	-361.6489	1.06e+08	609.12%
2522621	1692554	-159.2441	88	-51.0000	-361.6054	1.06e+08	609.03%
2531222	1701571	-351.4425	327	-51.0000	-361.6054	1.06e+08	609.03%
2537166	1707313	-344.7554	276	-51.0000	-361.6054	1.06e+08	609.03%
2543854	1710306	-288.1213	179	-51.0000	-361.6054	1.06e+08	609.03%
2548956	1715402	-352.9752	266	-51.0000	-361.6054	1.06e+08	609.03%
2556366	1719215	-90.8929	53	-51.0000	-361.6054	1.07e+08	609.03%
2563936	1724929	-348.9798	269	-51.0000	-361.5820	1.07e+08	608.98%
2572853	1730095	-143.0000	96	-51.0000	-361.5413	1.07e+08	608.90%

Elapsed time = 6920.25 sec. (4150726.32 ticks, tree = 184835.85 MB, solutions = 37)

Nodefile size = 182651.48 MB (160032.23 MB after compression)

2580724	1735982	-348.7664	273	-51.0000	-361.5161	1.07e+08	608.86%
2585129	1742238	-340.2640	247	-51.0000	-361.5073	1.07e+08	608.84%
2592516	1746730	-357.0770	344	-51.0000	-361.4975	1.07e+08	608.82%
2598719	1750962	-323.6565	219	-51.0000	-361.4916	1.08e+08	608.81%
2606882	1756673	-120.0000	64	-51.0000	-361.4748	1.08e+08	608.77%
2611677	1761982	-164.3744	85	-51.0000	-361.4690	1.08e+08	608.76%
2619177	1766883	-178.8411	97	-51.0000	-361.4666	1.08e+08	608.76%
2627954	1770167	-107.0000	47	-51.0000	-361.4526	1.08e+08	608.73%
2635043	1780029	-123.1203	62	-51.0000	-361.4435	1.09e+08	608.71%
2639585	1781709	-314.0933	199	-51.0000	-361.4365	1.09e+08	608.70%

Elapsed time = 7185.81 sec. (4303333.35 ticks, tree = 191557.34 MB, solutions = 37)

Nodefile size = 189428.39 MB (166019.78 MB after compression)

2647726	1786498	-104.0000	54	-51.0000	-361.4335	1.09e+08	608.69%
2653388	1791883	-89.0000	45	-51.0000	-361.4254	1.09e+08	608.68%
2659559	1796095	-248.7363	154	-51.0000	-361.4195	1.09e+08	608.67%
2664920	1800003	-208.3245	115	-51.0000	-361.4124	1.09e+08	608.65%
2671339	1803572	-242.4922	143	-51.0000	-361.4098	1.09e+08	608.65%
2679485	1809454	-163.4830	93	-51.0000	-361.4098	1.10e+08	608.65%
2687816	1818604	-161.0000	82	-51.0000	-361.4098	1.10e+08	608.65%

2692818	1820449	-106.8889	48	-51.0000	-361.3861	1.10e+08	608.60%
2698954	1825943	-120.0000	69	-51.0000	-361.3847	1.10e+08	608.60%
2705641	1829933	-340.4938	220	-51.0000	-361.3663	1.10e+08	608.56%

Elapsed time = 7443.22 sec. (4455943.64 ticks, tree = 197301.95 MB, solutions = 37)

Nodefile size = 195152.26 MB (171064.80 MB after compression)

2716311	1836247	-164.0000	97	-51.0000	-361.3485	1.11e+08	608.53%
2724612	1841495	-209.1877	122	-51.0000	-361.3366	1.11e+08	608.50%
2736277	1851272	-354.5079	276	-51.0000	-361.3239	1.11e+08	608.48%
2746256	1859680	-304.4639	194	-51.0000	-361.3177	1.11e+08	608.47%
2752715	1865169	-314.2210	188	-51.0000	-361.3065	1.11e+08	608.44%
2760747	1868729	-334.1027	236	-51.0000	-361.2981	1.11e+08	608.43%
2767824	1874686	-358.6653	314	-51.0000	-361.2860	1.12e+08	608.40%
2776708	1879875	-108.5139	58	-51.0000	-361.2798	1.12e+08	608.39%
2780036	1883524	-346.3044	728	-51.0000	-361.2798	1.12e+08	608.39%

GUB cover cuts applied: 2067

Clique cuts applied: 60

Cover cuts applied: 5679

Implied bound cuts applied: 140

Flow cuts applied: 220

Mixed integer rounding cuts applied: 7001

Zero-half cuts applied: 147

Lift and project cuts applied: 28

Gomory fractional cuts applied: 187

Root node processing (before b&c):

Real time = 0.01 sec. (4.06 ticks)

Parallel b&c, 8 threads:

Real time = 7674.31 sec. (4597796.80 ticks)

Sync time (average) = 2890.35 sec.

Wait time (average) = 0.00 sec.

Total (root+branch&cut) = 7674.31 sec. (4597800.86 ticks)

Iteration 15

Bounds on # of cuts = 8 with [3 3 2]

Error = 49 (out of 100 instances)

Accuracy = 51

Solving time = 127.905276652 min (minutes)
Accumulated time = 209.928617383 min (minutes)

Solution status code = 111
LB on error = -261.279772855
Relative objective gap = 6.083917115

Selected variables:
A_AGE (Continuous)
PEMLR (Categorical)

Number of selected variables = 2 (1 continuous + 1 categorical)

main returns 0

<<< main

<<< done

Biography

Songkomkrit Chaiyakan was born in Hatyai, Thailand, on August 12, 1991. He had been studying Mathematics and Applied Mathematics-Economics at Brown University, United States of America, from 2011 to 2013. In 2014, he transferred to a university in Thailand and received the Bachelor of Science (B.Sc.) degree in Mathematics from Prince of Songkla University, Thailand, in 2017. The Master of Science (M.Sc.) degree in Applied Mathematics and Computational Science was conferred by Chulalongkorn University, Thailand, in 2020. Currently, he is pursuing the Doctor of Philosophy (Ph.D.) program in Business Analytics and Data Science at National Institute of Development Administration (NIDA), Thailand.

Regarding work experience, he served as a homework grader for two undergraduate-level courses in calculus and microeconomics at Brown University from September 2012 to May 2013. He also worked as an academic officer at Learn Corporation from June 2019 to November 2019. At Chulalongkorn University, he served as a teaching assistant for two graduate-level courses in mathematical programming and real analysis in addition to three undergraduate-level courses in calculus and stochastic processes from January 2018 to April 2020. At National Institute of Development Administration, he assisted professors with their graduate classes in basic programming and database management, applied machine learning, and data streaming and real-time analytics from August 2022 to May 2024.

His research interest is to develop quantitative tools and achieve a breakthrough in finance, optimization, statistics and artificial intelligence (AI). In his spare time, he enjoys tackling unsolvable problems and also proving or providing interesting insights into commonly used, yet partially theoretically substantiated, statements.