University of Ottawa

Fall 2019: CSI 5155 Final Project Report

Predicting a Binary Income Category from U.S. Census Data (1994-1995)

Author
James Dickens

Professor
Dr. Herna Viktor

December 3, 2019

Contents

1 Abstract

This paper examines the performance of a sample of machine learning algorithms used to classify a person's income based on a set of 41 attributes. The models are built, trained, and tested using a standard methodology for evaluating machine learning algorithms. The test and data set instances belong to either one of two categories: individuals who make more than $50K$ a year, and those who make less than or equal to $50K$ a year. The data set is heavily inbalanced in favour of those who make less. Extensive feature filtering and transformations are used to improve overall performance and keep training time reasonable. A common theme in the performance analysis of each model is a trade-off between *minority class recall*, and *precision*, which will be explored numerically. Indeed, the performance of each model reflects this trade-off. Moreover, from a sample of models spanning the categories of linear algorithms, tree algorithms, ensemble algorithms, distance-based algorithms and rule based algorithms, the most impressive results stem from an ensemble approach, delivering an accuracy of over 92% alongside a minority class recall of roughly 70%.

From the results of such an experiment, by examining the corresponding model predictions, one can determine probabilistic indicators for policy makers and economists alike to study which attributes are correlated with a higher income. This is of course contingent on the so-called *explainability* of the models; that is, the ability of the machine learning engineers to explain how the predictions were achieved from the finished models.

2 Introduction

In 1994, the real median household income in the United States was $54,000$ USD [1]. A household's income is defined to be the income of every resident within a home that is over the age of 15, including pre-tax wages and salaries, along with other sources of income such as investments, unemployment insurance payments, social security, disability payments, and child support [2].

On an individual level, many factors influence a person's income. A whole host of characterstics including education level, age, race, inherited wealth, occupation type, and overall health paint a probabilistic picture of whether or not an individual's yearly income falls within a given bracket. For example, according to the US. Bureau of Labor Statistics, in 2017, the median weekly earning of an individual with a bachelor's degree was roughly 1300$, as compared to slightly above 500$ for an individual with less than a high school diploma [3], see Figure 2. It is true that the models discussed in this paper reflect this reality in their predictions.

In this report, we will examine the performance of machine learning algorithms trained on the data set provided by Terran Lane and Ronny Kohavi employed at Silicon Graphics in the Data Mining and Visualization department. It is available at the UC Irving Machine Learning Repository website [4]. Each instance contains a set of attributes as well as its income category. This dataset presents a unique challenge as the set of attributes is relatively large, and the class distribution is heavily skewed. As per this unbalanced distribution, special care is taken to ensure that the models do not miss positive instances (i.e. people who make more than $50K$ a year), so as to not rely solely on accuracy as the most important measure of performance. Moreover, it is worth noting that the data set has flaws such as missing values, and does not account for salaries. In particular a salaried individual is given a *wage* value of 0, adding to the uncertainty of the predictive ability of the models. A similar dataset which is very well studied in the literature, and was the focus of a *Kaggle* competition, is available from this source, and so this problem is relatively saturated in machine learning pedagogy. The difference in this case being a much larger dataset, with an extended feature set for each instance.

For my project, I have chosen the application based approach, as described in the assignment description, to tackle this problem by evaluating five machine learning models to compare their performance in correctly predicting the income level of the test data. By comparing the performance of each model on the data set, we can hope to draw insights on both the nature of the models, and the learnability of the classification task. Additionally, one can learn more about their resepctive implementations and hyper-parameters. The implementation of this report's code is available at `https://github.com/JamesMcCullochDickens/Machine-Learning-Project`.

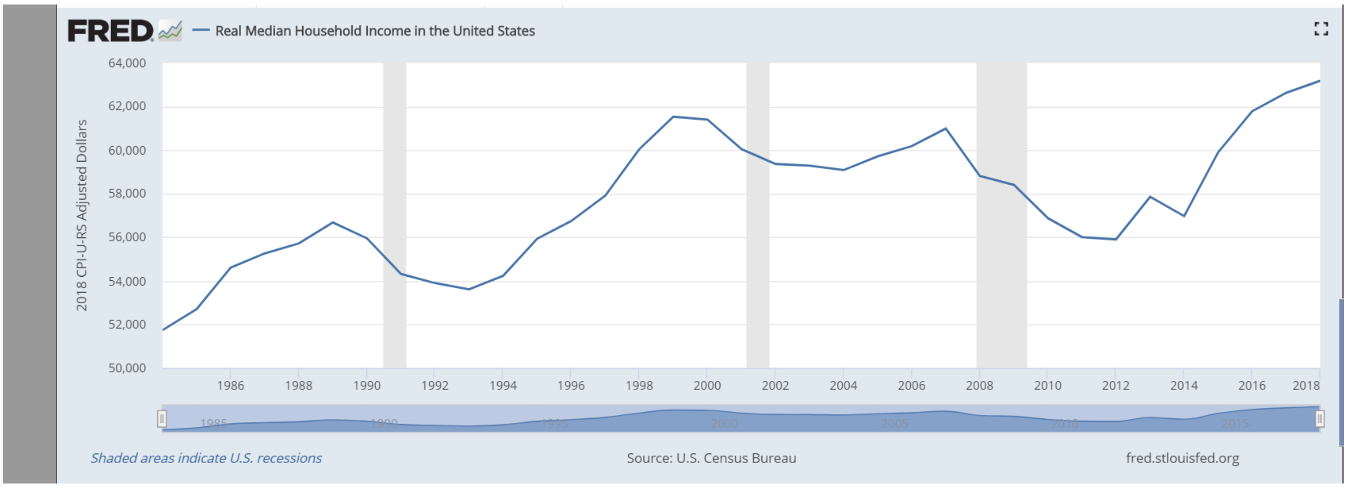

Figure 1: *Real Median Income in the United States* [1]

3 Case Study and Feature Selection

3.1 Raw Data: Initial Features

The original dataset is given by 42 categories with corresponding values:

1. *Age*: The age of the person/worker given by an integer in the range 0 to 90.

2. *Class of Worker:* A categorical value from the values: *Not in universe (meaning not applicable), Private, Self-employed-not incorporated, Local governement, State goverment, Self-employed-incorporated, Federal government, Never worked, Without pay.*
***Note that an incorporated business, or a corporation, is a separate entity from the

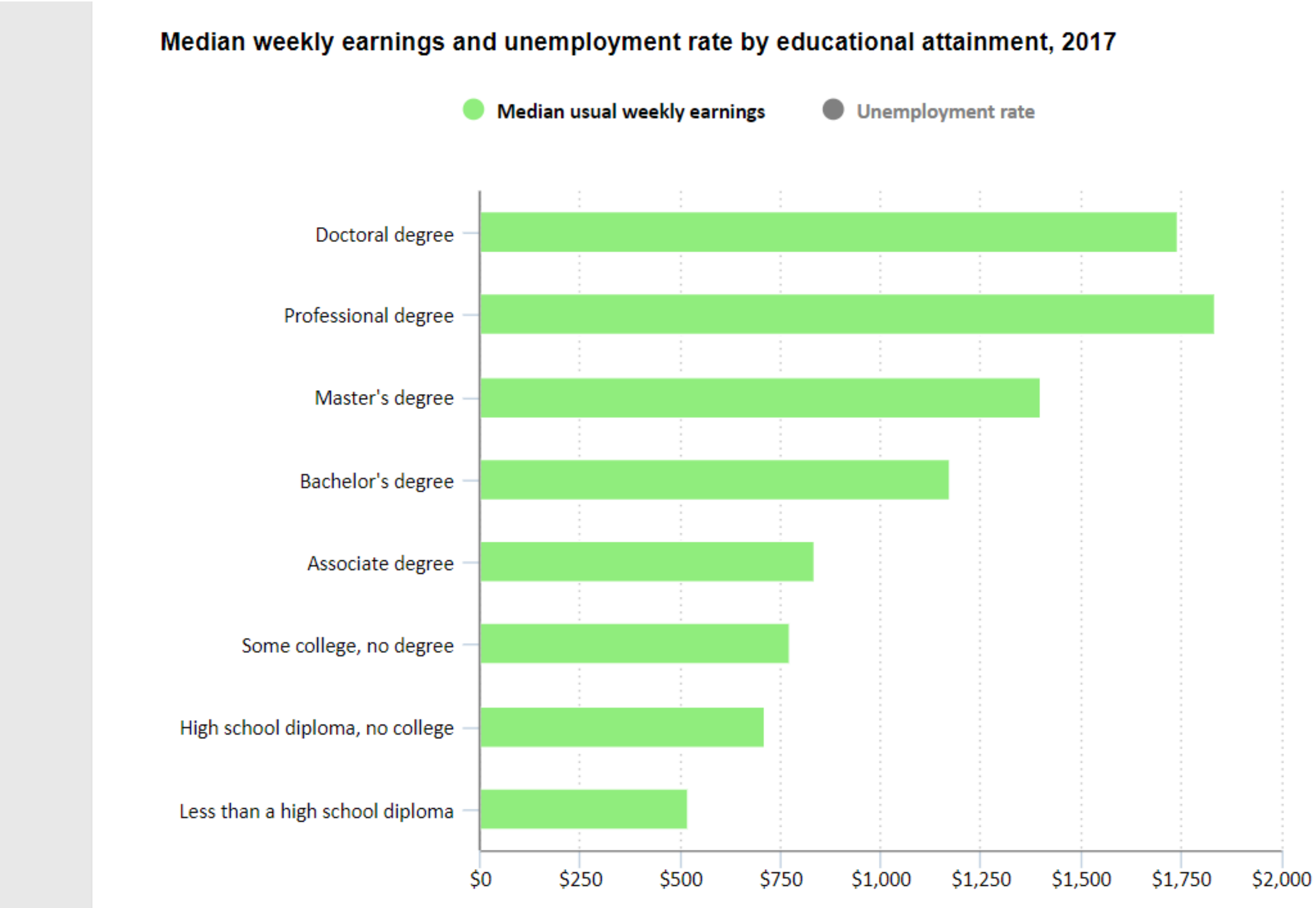

Figure 2: *Median weekly earnings by Education Level* [3]

business owner and has legal rights.

3. *Industry Code:* A two-digit code corresponding to the industry of the worker (0 if not applicable).

4. *Occupation Code:* A two-digit code corresponding to the occupation the worker (0 if not applicable).

5. *Education level:* A categorical value from the values: *Children, Less than 1st grade, 1st 2nd 3rd or 4th grade, 5th or 6th grade, 7th and 8th grade, 9th grade, 10th grade, 11th grade, 12th grade no diploma, High school graduate, Some college but no degree, Associates degree-academic program, Associates degree-occup /vocational, Bachelors degree(BA AB BS), Masters degree(MA MS MEng MEd MSW MBA), Prof school degree (MD DDS DVM LLB JD), Doctorate degree(PhD EdD).*

6. *Wage per hour:* A real number corresponding to 100 times the hourly wage paid. For example, a value of 1200 would correspond to 12 dollars/hr.

7. *Enrolled in educational institution last week:* Categorical value describing the current enrollemnt status of the person with values: *Not in universe, High school, College or university.*

8. *Marital Status:* Categorical value with values: *Never married, Married-civilian spouse present, Divorced, Widowed, Separated, Married-spouse absent, Married-A F spouse present.*

9. *Major Industry:* A categorical value describing the industry of the worker. Examples values include: Mining, Education, Retail Trade, etc.

10. *Major Occupation:* A categorical value describing the occupation of the worker. Example values include: Executive admin and managerial, machine operators assemblers and inspectors, transportation and material moving, etc.

11. *Race:* A categorical value indicating the race of the person from the values: *White, Black, Asian or Pacific Islander, Other, Amer Indian Aleut or Eskimo.*

12. *Hispanic Origin* A categorical value describing the origin, if applicable, of a hispanic person from the values: *All other, Mexican-American, Mexican (Mexicano), Central or South American, Puerto Rican, Other Spanish, Cuban, NA, Do not know, Chicano.*

13. *Sex:* A categorical value with values: *Male, Female.*

14. *Member of a Labour Union:* A categorical value from: *Not in universe, No, Yes.*

15. *Unemployment Reason:* A categorical value describing the reason, if applicable, for unemployment from the values: *Not in universe, Other job loser, Re-entrant, Job loser - on layoff, Job leaver, New entrant.*

16. *Full or Part-Time employment status:* A categorical value describing the current employment status of the person with values: *Children or Armed Forces, Full-time schedules, Not in labor force, PT for non-econ reasons usually FT, Unemployed full-time, PT for econ reasons usually PT, Unemployed part- time, PT for econ reasons usually FT.*
***Note that PT = part time, FT = full time.

17. *Capital Gains:* A numerical value typically denoting a profit from the sale of property or an investment during one fiscal year.

18. *Capital Losses:* A numerical value typically denoting a loss from the sale of property or an investment during one fiscal year.

19. *Stock Dividends:* A numerical value denoting the stock dividends received during one fiscal year. Note that a stock dividend is a dividend payment made in the form of additional shares rather than a cash payout where a dividend is a sum of money paid regularly (typically quarterly) by a company to its shareholders out of its profits.

20. *Tax Filer Status:* A categorical value describing the tax filer status of the person

given by the values: *Nonfiler, Joint both under 65, Single, Joint both 65+, Head of household, Joint one under 65 & one 65+*.

21. *Region of Previous Residence:* A categorical value from: *Not in universe, South, West, Midwest, Northeast, Abroad.*

22. *State of Previous Residence:* The state where the person previously resided if applicable. A question mark (?) is given if the value is unknown.

23. *Detailed Household and Family Status:* A categorical value describing the household status of the worker. Examples include: *Child 18+ ever marr Not in a subfamily, Child under 18 of RP of unrel subfamily, etc..*

24. *Detailed Household Summary:* A categorical value giving the household status of the person. Values are from: *Householder, Child under 18 never married, Spouse of householder, Child 18 or older, Other relative of householder, Nonrelative of householder, Group Quarters- Secondary individual, Child under 18 ever married.*

25. *Instance Weight:* A real number ascribing a weight to instances due to stratified sampling, more about this in the next section.

26. *Migration Code: Change in MSA*: A categorical value giving information about the person's migration to the United States within the last fiscal year given by the values: *?, Nonmover, MSA to MSA, NonMSA to nonMSA, Not in universe, MSA to nonMSA, NonMSA to MSA, Abroad to MSA, Not identifiable, Abroad to nonMSA*.
**Note that a metropolitan statistical area (MSA) is a geographical region with a relatively high population density at its core and close economic ties throughout the area.

27. *Migration Code: Change in Region:* A categorical value giving information about the person's migration between regions within the United States during the last fiscal year given by the values: *? Nonmover, Same county, Different county same state, Not in universe, Different region, Different state same division, Abroad, Different division same region.*

28. *Migration Code Within Region:* A categorical value giving information about the person's migration within a given region in the United States during the last fiscal year, given by the values: *?, Nonmover, Same county, Different county same state, Not in universe, Different state in South, Different state in West, Different state in Midwest, Abroad, Different state in Northeast.*

29. *Lived in the same house one year ago:* A categorical value giving describing if the person lived in the same house as one year ago from values: *Not in universe, under 1 year old, Yes, No.*

30. *Migration - previous residence in sunbelt:* A categorical value describing if the person's residence is in the sunbelt from values: *?, Not in universe, No, Yes.*
***Note that the Sun Belt is a region of the United States that stretches across the Southeast and Southwest.

31. *Number of Persons Working for Current Employer:* A numeric value depicting the number of employees working for the current employer of the person.

32. *Family members under 18:* A categorical value describing the presence of parents to a person under 18 given by the values: *Not in universe, Both parents present, Mother only present, Father only present Neither parent present.*

33. *Birth Country of the Father:* A categorical value giving the birth country of the father of the person.

34. *Birth Country of the Mother:* A categorical value giving the birth country of the mother of the person.

35. *Birth Country of the Person:* A categorical value giving the birth country of the person.

36. *Citizenship:* A categorical value describing the U.S.A citizenship status of the person given by the values: *Native- Born in the United States, Foreign born- Not a citizen of US, Foreign born- U S citizen by naturalization, Native- Born abroad of American Parent(s), Native- Born in Puerto Rico or U S Outlying.*

37. *Own a business or self-employed:* A numeric value indicating if the person is self employed or owns a business, 1 for yes, 0 for no.

38. *Completed Questionnaire for Veterans Administration:* Categorical value depending on if the person completed the survey given by the values: *Not in universe, No, Yes.*

39. *Veterans Benefits:* A numeric value indicating whether the individual gets veterans benefits or not.

40. *Weeks Worked:* An integer representing the number of weeks worked in a year.

41. *Year the Survey was Completed:* An integer representing the year the survey was completed, either 1994 or 1994.

42. *Income Category:* A string representing whether or not the person makes more than 50 thousand USD a year with values: - *50000.* for no, and *50000 +.* for yes.

3.2 Examining the Raw Data

Upon first examination, the dataset appears to have many incomplete values and extraneous information that will need to be cleaned up. Additionally 41 categories represents a lot of information, and so one has to caution that there may be redundancy. The data has the following properties:

- The total number of instances in the training data is 199523
- The total number of instances in the testing data is 99762

- Many attributes have missing entries consisting of the value “ ? ”. The training set has 415717 missing entries, and the testing data set has 208379 missing entries. To account for missing values, the most common value is substituted for a missing entry, in particular *USA* is entered for the birth country of the parents as well as the person since the survey takes place in the United States.

- The instances are given a *weight* feature

- The data has instance weight conflicts where two instances with the exact same values have a difference weight, which I solve by replacing two instances with one instance with weight correspodning to the mean of the values (which is done with possibly more than 2 instances)

- The instance weight feature corresponds to the number of people that are represented by a particular instance due to stratified sampling, as per the authors description of the data. The instance weights are in the range $(0, 20000)$ As a result, I initally duplicated instances according to the formula

instance count $=$ ((instance weight)/20000) $*$ 10) $+$ 1, i.e. duplicate the instance according to its weight.

However, upon testing, ignoring the weights altogether yielded significantly better results, and so the weights were removed.

- The data contains duplicate entries (not counting the number of instance weight conflicts): 3229 in the training set, and 883 in the test set; the duplicates in the training set were removed.

- Before accounting for duplicates, there are 12382 instances in the training set that make more than $50K$ a year, and 187141 who do not, and hence we have a heavily skewed data set.

- The test data (which also contains duplicates which are not removed) contains 6186 instances that make more than $50K$ a year, and 94576 that do not.

3.3 Feature Filtering

Since the data contains 41 different attribute values per instance, it is easy to see why feature filtering may simplify the training, testing, and reading of the results for the outputs of the models. Moreover some of the categories give irrelevant or redundant information (for example Occupation Code and Major Occupation are redundant), and hence I will employ feature evaluation methods to try and deduce which data categories will be most useful, and how to modify them accordingly.

Three techniques, alongside common sense, will be employed to determine which features give useful information for our classification task. First, the *CfSubset Eval* method in Weka is used alongside the *Greedy Stepwise* search method. This method evaluates the worth of a subset of attributes by considering the predictive ability of each feature along with a measure of redundancy. The resulting feature subset is: *Occupation Code, Education, Sex, Capital Gains, Capital Losses, Stock Dividends, Weeks Worked.*

Next the *Information Gain* attribute evaluator is used, again in Weka, alongside a ranker, to rank the attributes according to the information gain they provide according to the formula InfoGain(Class, Attribute)$= H(\text{class}) - H(\text{class}|\text{Attribute})$, where H is the information entropy of a class, and $H(\text{class}|\text{Attribute})$ is the conditional entropy. The results are summarized in the following table.

Information Gain	Attribute
0.094795	Occupation Code
0.077738	Education
0.073776	Major Occupation
0.046931	WeeksWorked
0.04684	Industry Code
0.044809	CapitalGains
0.044487	Age
0.042934	Major Industry
0.040657	HouseHoldFamily
0.038557	TaxFiler
0.037571	StockDividends
0.03681	HouseholdSummary
0.034823	ClassOfWorker
0.032059	NumPeopleWorkingWithEmployers
0.02423	Marital Status
0.023158	Sex
0.020464	CapitalLosses
0.014295	FamilySituation
0.013763	EmploymentStatus
0.009991	VetBenefits
0.008045	EnrolledEducation
0.006929	HispanicOrigin
0.006558	CountryOfBirthOfFather
0.006188	CountryOfBirthOfMother
0.005658	Wage
0.004701	Race
0.004503	CountryOfBirthOfPerson
0.003071	MigrationCode
0.002945	MigrationMove
0.002937	StateOfResidence
0.002937	MigrationChangeRegion
0.00261	RegionOfResidence
0.0026	Self-Employed
0.002503	LivingInHouse
0.002311	Citizenship
0.00202	UnemploymentReason
0.001594	Union
0.001208	MigrationPreviousResidence
0.000381	YearCompleted
0.00033	SurveryCompleted

Finally, we exam the Pearson correlation coefficient in regards to the correlation of some numerical variables tested against the target class. Notably:
- 0.23626 correlation with the *capital gains* attribute
- 0.22324 correlation with *weeks worked* attribute
- 0.16868 correlation with the *stock dividends* attribute

- 0.10294 correlation with the *age* attribute.

As per these results, alongside my own judgement, the following features will be used, and the data will be modified in the code accordingly. In some cases the categories are simplified to reflect less cases, see the code for details.

Age: Integer from 0 to 90
Class of Worker: Not Employed, Private, Self-Employed, Local Government, State Government, Federal Government
Industry Code: Non-negative integer
Occupation Code: Non-negative integer
Education: Less than high school, High School Graduate, College, Bachelors, Masters, Prof Degree, Doctorate
Wage: Real valued
Enrolled Education: Not Enrolled, High School, College or University
Married: Not Married, Married, Divorced, Widowed
Race: Asian or Pacific Islander, White, Other, Amer Indian Aleut or Eskimo, Black
Sex: Male, Female
Employment Status: Not Employed, Part Time, Full Time
Capital Gains: Real valued
Capital Losses: Real valued
Stock Dividends: Real valued
Tax Filer Status: Head of Household, Joint, Single, Nonfiler
Country of Birth Parents: USA (if at least one parent is born in the USA), Both not USA.
Country of Birth of Person: American, Not American
Weeks Worked: Non-negative integer

Income Category: 0 for a person making less than or equal to $50K$ a year, 1 for a person making more than $50K$ a year.

Finally there is one more layer of feature transformations applied before model construction.

3.4 Feature Transformations

The categorical features from the reduced/simplified feature set will be transformed into numerical features with the use of *one hot-encoding.* This will apply to the attributes: *class of worker, education, enrolled education, married, race, employment status, and tax filer status.*
For example from the possible values for the Tax Filer attribute, given by the set $\{'HeadofHousehold','Joint','Single','Nonfiler'\}$, an instance with value *Joint* will be given a vector value of $(0, 1, 0, 0)$.

The features sex, country of birth of parents (at least one born in America?), country of birth of person (American?), as well as the income category, are binarized (i.e. set to 0 or 1).

The occupation and industry codes have too many possible values to employ one-hot encoding without adding too many dimensions to the data set, and so feature calibration is employed.

Suppose for a given occupation or industry code x that we have m of n people in our dataset that work in this occuption/industry who make more than $50K$, then the posterior odds are given by $\frac{m}{n-m}$, and the likelihood ratio is given by $\frac{m}{c(n-m)}$ where c is the prior odds of any person belonging to the income category that makes more than $50K$ a year. From the US. Census bureau, it is estimated that roughly 30 percent of Americans make more than $50K$ a year, and so $c = \frac{0.3}{1-.3} \approx 0.43$.

Therefore by tallying the number of people in our dataset from each occupation/industry code that make more than $50K$ a year versus the total number of people that have that code, we can convert the likelihood ratio to a probability and compute $P(x) = \frac{\frac{m}{c(n-m)}}{\frac{m}{c(n-m)}+1} = \frac{m}{m+c(n-m)}$. This approach is adopted from the feature transformation section (10.2) of the course textbook written by Peter Flach. [5].

The remainder of the numerical features are normalized using the formula
$N(x) = \frac{x - min}{max - min}$.

Next we describe the experimental setup.

3.5 Dealing with Class Imbalance

Since the data is heavily inbalanced in favor of the majority class, several strategies were attempted to increase minority class recall upon training and testing of the various models. In particular the following were attempted:
- Undersampling the Majoirty Class by various amounts
- Oversampling the Minority Class by various amounts
- Balanced sampling (undersampling followed by oversampling)
- The use of Smote and AdaSyn techniques

In the end the result chosen was to use the random over sampler class from the imbalanced learn package alongside the random undersampler from the same package, to achieve a ratio of 20000 minority class instances and 50000 majority class instances. Duplicating minority class instances further had diminshing returns for the performance metrics of the models on the validation set, as did undersampling the majority class.

4 Experimental Setup

For this project, five models will be used for our binary classification task. Implemented in scikit-learn, a machine learning library for Python, the models are:

• Tree Based Model: Decision Tree Classifier. This is the standard decision tree used in scikit learn, and so I have chosen to use it to become more familiar with its hyper-parameters.
• Distance Based Model: K-nearest neighbors classifier. This is a very popular distance based model and so I have chosen it to experiment with its hyper-parameters.
• Rule-Based Model: Skope Rules classifier, an experimental rules approach that works on top of scikit-learn.
• Linear Model: A semi-supervised neural network implemented in Keras. Note that Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow to enable fast experimentation with deep neural networks. This model is chosen in order to explore both semi-supervised learning, and neural networks.
• Ensemble Model: The Ada Boost Classifier, chosen for its reputation as being robust to outliers.
• A majority weighted vote of all the models, just for fun!

The training data is split, as per the authors choice, in a 2/3-train, 1/3 test fashion. The hyperparameters of each model are validated on 90 percent of the training data, and tested on the remainder of the training set, as per a standard machine learning approach

Cross validation with 10-folds is done on each model (except the neural network and the overall model), after which a paired T-test is used for pairwise performance analysis of the models using the accuracy and minority class recall metrics. The recall metric is used to see which models perform better at not missing positive instances, as the data is heavily imbalanced. The paired T-test is used to compare different models pairwise to determine if the differences in accuracy (or in our case recall), are significant according to an assumption that the differences per fold are normally distributed.

After training, the models' predictions on the testing set are computed, upon which the results are evaluated. The performance metrics chosen will be accuracy, minority class recall, f1 score, precision, and the confusion matrix. In particular the f1 score and the confusion matrix demonstrate which model does the best job of balancing the recall-precision tradeoff. Finally the ROC curve is shown for a visualization of the diagnostic ability of our binary classifiers as their discrimination thresholds are varied, where the area under the curve represents the ranking accuracy of the models.

4.1 Decision Tree Model

The decision tree classifier is a non-parametric model that uses a splitting criterion to classify data according to a probability assigned to the leaf that an instance will belong to after traversal. There are many decision tree algorithms in the literature, however scikit-learn uses an optimized version of the **Cart** (Classification and Regression Trees) algorithm which constructs binary trees using the features and thresholds that yield the largest information gain at each node. The time complexity of this algorithm is roughly $O(n_{features}n_{samples}^2log(n_{samples}))$ [6].

The hyper-parameters considered in the implementation are the criterion, Gini or

Entropy, the maximum depth of the tree, and the minimum samples split. Limiting the maximum depth of the tree was crucial to prevent overfitting and increase minority class recall. The final parameters for the chosen model after validation are
max depth = 8, criterion = gini, min samples split = 2.

4.2 Distance-Based Model

The nearest neighbours classifier in scikit learn, with class name KNeighboursClassifier, is the distance based model chosen. The running time of this model was by far the longest of any of the chosen models, as the time complexity of this approach is roughly $O(dimension_{data} \cdot n_{samples} \cdot min\{n_{samples}, n_{features}\})$ [7].

The hyperparameter chosen to be tested were:
- the number of neighbours, $1, 3, 5$ respectively
- weighted votes or non-weighted votes
- the distance metric: Manhattan or Euclidean

The model chosen after validation has hyper-parameters: 5 neighbours, weighted votes, with Euclidean distance. I suspect the class overlap present in the data was a primary factor in the improved performance from weighting the votes in this algorithm.

4.3 Rule-Based Model

The rule based model will be the *Skope-Rules* package which is built on top of scikit learn. According to the authors, Skope-rules aims at learning logical, interpretable rules for scoping a target class, and offers a trade-off between the interpretability of a Decision Tree and the modelization power of a Random Forest. This method uses a bagging estimator consisting of decision trees and regression trees in order to construct logical rules for prediction, see Figure 3 [8].

The hyper-parameters tuned for validation of this method are the number of estimators, the minimum precision, and the minimum recall. The hyper-parameters for the model chosen are number of estimators = 50, min precision = 0.2 and min recall= 0.2. With larger recall and precision minimums, the model did not output very many rules (less than 2 rules for minimum value 0.4), and with high enough minimums, the model would only predict the majority class, and so tuning these hyper-parameters was essential. Additionally a large number of estimators resulted in much longer training time with diminishing performance returns.

4.4 Linear Model: A Semi-Supervised Neural Network

Semi-Supervised learning utilizes both labelled training data, alongside unlabelled data to perform classifaction. In our scenario, we first construct a neural network using Keras for classification with an hourglass shape:

```
neural_network = Sequential()
neural_network.add(Dense(8, input_dim=42, activation='relu'))
```

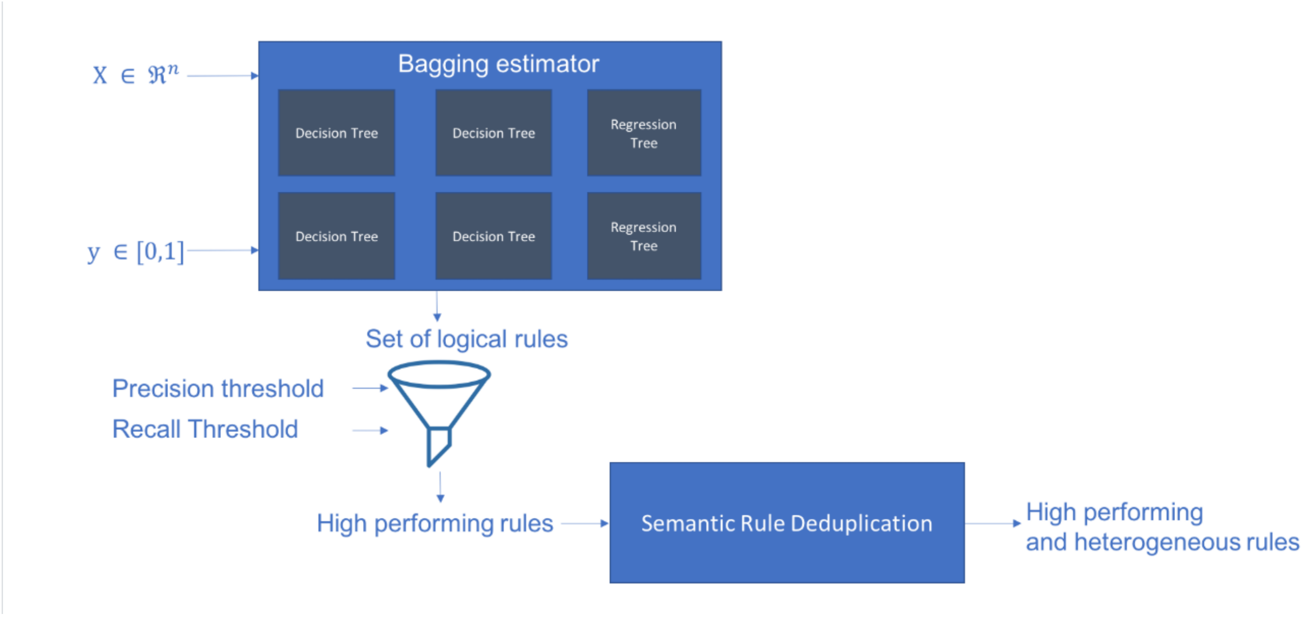

Figure 3: A High level description of The *Skope-Rules* Classifier [8]

```
neural_network.add(Dense(16, activation='relu'))
neural_network.add(Dense(8, activation='relu'))
neural_network.add(Dense(4, activation='relu'))
neural_network.add(Dense(1, activation='sigmoid'))
```

There are 4 hidden layers and one output layer. The hidden layers use the *ReLu* activation function, and the outer layer with one node uses the *sigmoid* function to give a probability of belonging to class 0 or 1 (higher probabilities for class 1, lower probabilities for class 0). The semi-supervised method works as follows

1. Split the training data into 4 partitions
2. Train the data on the first partition
3. For each remaining partition, split it in half.
4. Train the network with the first half
5. Make predictions using the network on the second half, and for all predictions that have a high probability of belonging to a certain class (computed using a threshold function), use the resulting class labels to train the network
6. After all training, evaluate the network on the test data set.

The loss function utilized is binary cross-entropy using the Adam optimizer. The threshold for the probabilities was the primary hyper-parameter tweaked for this model. Too high of a threshold for the positive class, say above 0.65, would result in almost no instances being classified as positive, while too low a threshold for the majority class, say below 0.15, would result in many false positives. An appropriate balance was struck as per the code:

```
# Get the indices which have a high probability for training the network
# only on high probability unsupervised labellings
def get_indices_of_high_probability(pred_array):
    indices = []
```

```
count = 0
for i in range(0, len(pred_array)):
    if pred_array[i] > 0.625:
        count = count + 1
        indices.append(i)
    elif pred_array[i] < 0.175:
        indices.append(i)
return indices
```

4.5 Ensemble Model

The ensemble model chosen is the AdaBoost classifier, which is described in the scikit-learn documentation as a meta-estimator that works by first fitting a classifier on the original dataset. The model subsequently fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted. This ensures that later classifiers focus more on difficult cases [9], and so have a higher probability of classifying these cases correctly. This method is known to be robust to outlier detection. The hyper-paramters considered for this model are:
- the number of estimators
- the learning rate

The model chosen after validation has 100 estimators, with learning rate = 0.2. Similar to Skope Rules, increasing the number of estimators resulted in an increase in training time with diminishing performance returns. The learning rate was adjusted with a manual trial and error approach.

5 Results

5.1 Cross-Validation

The accuracy and recall cross-validation of the models (excluding the neural network and overall predictive model) are printed below:

DT = Decision Tree, KNN = K nearest neighbours, SR = Skope rules, AB = Ada Boost classifier, A = Accuracy, R = minority class recall.

Fold	DT - A	DT - R	KNN - A	KNN - R	SR A	SR R	AB - A	AB - R
1	85.51	65.80	88.77	84.85	76.73	79.43	85.84	69.25
2	84.77	66.00	88.03	84.00	77.94	78.43	85.59	68.80
3	84.56	65.75	88.16	83.70	81.27	66.72	85.49	68.55
4	85.07	67.30	87.76	83.85	79.93	71.24	85.56	68.80
5	85.41	67.90	87.67	84.40	80.96	71.29	85.89	71.15
6	84.37	65.75	88.47	84.15	80.04	67.02	85.10	67.80
7	84.93	69.65	87.67	82.90	76.16	77.32	85.33	69.05
8	84.97	63.35	88.94	86.45	78.57	76.89	85.14	67.30
9	84.97	67.85	88.69	85.15	80.73	71.84	85.56	69.70
10	84.96	65.05	88.37	84.05	80.66	68.45	85.61	68.70

5.2 Paired T-test

For each pair of the algorithms in the cross-validation above, the results of the paired T-test is computed for accuracy and recall for the significance level $\alpha = 0.05$. The results are as follows. $S =$ significance level threshold met, $A = accuracy, R = Recall.$

Data Type	DT vs KNN	DT vs SR	DT vs AB	KNN vs Rule	KNN vs AB	SR vs AB
A T-stat	-19.99	10.05	-3.78	15.56	20.39	-11.15
S (Y/N)	Y	Y	Y	Y	Y	Y
R T-stat	-39.60	-5.47	-4.72	8.00	35.40	3.68
S (Y/N)	Y	Y	Y	Y	Y	Y

Overall we see that there is a significant difference in accuracy and recall for each pair of algorithms cross-validated on the training data set with 10 folds.

5.3 Rules

For the decision tree approach, the corresponding tree rules are available in the Tree Rules.txt file on the github page. An example rule is that if the individual is over the age of 52, has a calibrated industry code probability of over 0.47, has capital gains over 1000 dollars, has a calibrated occupation code above 0.30, and has worked more than 82% of the weeks during the year, then the decision tree will predict that the individual makes more than $50K$ a year.

The rules from Skope rules are are available in the Skope Rules.txt file on the github page, and are given with their corresponding recall and precision (the result of applying that single rule to the data set). An example rule is that if Age $>$ 0.30 and the calibrated occupation code $>$ 0.22 and the person's stock dividends $>$ 0, then predict the positive class.

5.4 Overall Results and Discussion

5.4.1 Overall Results

The results for predictions on the test set of the five models is printed with regards to accuracy, recall, $F1$ score, precision, and the confusion matrix.

Metric	Decision Tree	KNN	Neural Network	Skope Rules	Ada Boost	Overall
Accuracy	92.75	90.71	90.29	87.51	92.58	92.37
Recall	64.14	64.13	74.50	70.01	69.29	69.25
F1	52.23	46.11	48.42	41.01	53.67	52.97
Precision	44.18	36.00	36.10	29.00	43.80	42.88

Next we print the confusion matrices for each algorithm.

Decision Tree

$$\begin{bmatrix} 88563 & 5013 \\ 2218 & 3968 \end{bmatrix}$$

K- Nearest Neighbours

$$\begin{bmatrix} 86524 & 7052 \\ 2219 & 3967 \end{bmatrix}$$

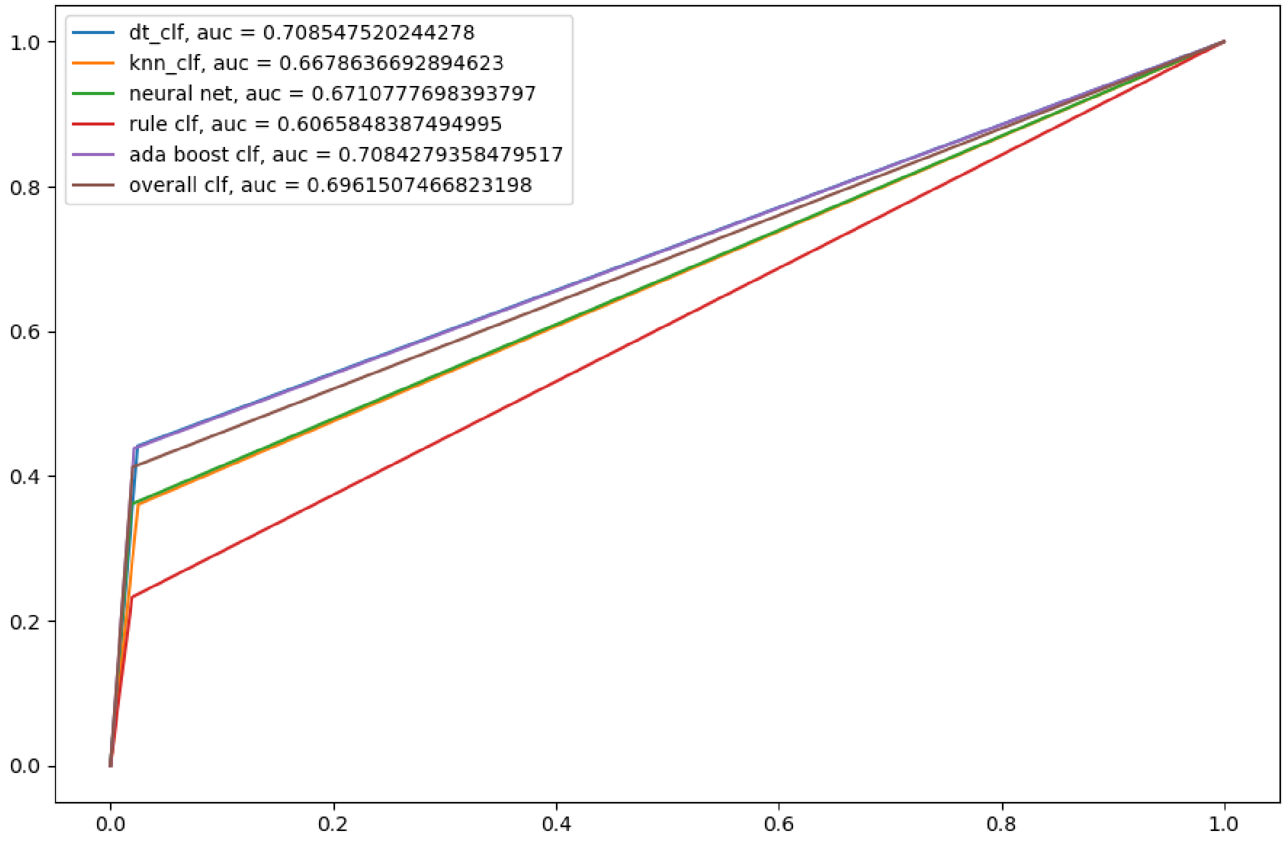

Figure 4: ROC curves for each model

Semi-Supervised Neural Network

$$\begin{bmatrix} 86688 & 6888 \\ 1821 & 4365 \end{bmatrix}$$

Skope Rules

$$\begin{bmatrix} 82973 & 10603 \\ 1855 & 4331 \end{bmatrix}$$

Ada Boost Classifier

$$\begin{bmatrix} 88076 & 5500 \\ 1900 & 4286 \end{bmatrix}$$

Overall Model:

$$\begin{bmatrix} 87870 & 5706 \\ 1902 & 4284 \end{bmatrix}$$

5.4.2 Discussion

As discussed in the introduction, the performance of each model is a clear example of the trade-off between precision and recall common in machine learning. In each model, the precision did not exceed 45%, while each model had a recall of at least 64%; the models had a difficult time increasing predictions for the minority class (those who make more than $50K$) a year without a corresponding increase in false positives. I believe this is due to several factors:

- There is class overlap, i.e. instances that are very similar that belong to both classes.
- There is missing information in the data. For instance salary earners are given a wage of 0, which if converted to a single indicator i.e. wage/salary, would make the results more clear.
- The probabilities given by the feature calibration of the occupation code and industry code are sometimes in the ambiguous range (not low enough or high enough). In other words in general the industry and occupation of a worker helps to determine income, but certain industries have large variance in terms of salaries/wages.

The high accuracy can be explained by noting that the survey considered age categories of individuals highly unlikely to be in the labour market, i.e. children, adolescents, and the elderly. Moreover as discussed in the feature transformation section, the prior odds of an individual making more than $50K$ a year are roughly 0.43, and so models that tend to classify instances in the negative will be accurate (not to mention the class imbalance).

Overall, the method that balanced this trade-off the best was the Ada Boost Classifier, which may have been expected given that this model is sensitive to noisy data and outliers, and is known to be less susceptible to overfitting. The model that performed the worst overall was the Rule-Based model; by viewing the rules it is clear that the class overlap resulted in rules with very low precision, and that were overly specific.

6 Conclusion

In conclusion, while the data set contained noise, class imbalance, incomplete values, and other challenges, the models were able to achieve high accuracy, with the glaring issue being precision and minority class recall lagging in behind. However, as per the goal of determining which factors are predictive in terms of the dataset some clear trends emerge:

- Individuals in the age ranges most common to full-time laborers tended to have a higher income
- Individuals who worked in occupations and industries known to have average salaries that exceeded $50K$ a year, unsurprisingly, were more likely to make $50K$ a year.
- Individuals who had the capital to invest in stocks, and to engage in capital transfers were more likely to have a higher income.
- Men, in general, have higher incomes than women (keep in mind this was $1994-1995$, and so I expect this imbalance to have decreased).
- Individuals with education levels higher than a high school degree, who are not currently enrolled, in general, had higher incomes.

Hence it follows that the task of predicting income is dependent on the *quality* of the data in terms of its most accurate representation of the individual. In particular from this data, we can conclude with certainty that the industry and occupation of a worker is a powerful predictive factor for income predictions. I would like to thank

Dr. Herna Viktor, and teaching assistant Rabaa Saleh A Alabdulrahman for their assistance with this project.

7 References

[1] U.S. Census Bureau, *Real Median Household Income in the United States* , retrieved from FRED, Federal Reserve Bank of St. Louis, `https://fred.stlouisfed.org/series/MEHOINUSA672N`, accessed November 23, 2019.

[2] *Census Long Form Definition.* United States Department of Housing and Urban Development. July 30, 2009. Archived from the original on October 8, 2012.

[3] Elka Torpey, *Measuring the value of education.* Career Outlook, U.S. Bureau of Labor Statistics, April 2018.

[4] T. Lane, R. Kohavi. *US Census Bureau Data from 1994-1995.* UCI repository of machine learning databases. Retrived from `www.ics.uci.edu/~mlearn/MLRepository.html`, accessed November 23, 2019.

[5] Flach, P. *Machine Learning: The Art and Science of Algorithms that Make Sense of Data*, Cambridge University Press, New York, NY, 2012

[6] *1.10. Decision Trees. (n.d.).* Retrieved from `https://scikit-learn.org/stable/modules/tree.html`, accessed November 23, 2019.

[7] *Nearest Neighbours.* Retrieved from `https://scikit-learn.org/stable/modules/neighbors.html`, accessed November 23, 2019.

[8] *Skope-Rules Github Home Page* (2019, October 1). F Gardin, R.Gautier, N. Goix, B. Ndiaye, J-M Shertzer. Retrieved from `https://github.com/scikit-learn-contrib/skope-rules`,
acessed November 23, 2019.

[9] *Ada Boost Classifier*, Retrieved from `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html`,
accessed November 23, 2019.