Project Title:

Analysis, Inference, & Predictive modeling of Drug use by the public of United States

TABLE OF CONTENT

ABSTRACT

This study employs decision trees, bagging and Random Forest techniques to delve into youth drug use, leveraging data from the National Survey on Drug Use and Health. Decision trees offer insight into complex relationships between demographic variables and drug use behaviors among youth. By applying bagging, an ensemble learning method, I enhanced predictive accuracy and robustness against overfitting. Through comprehensive analysis, I identified key predictors of youth drug use and quantified their impact. ‘Alcohol frequency past year’, ‘first used alcohol prior to age 18’, and ‘any tobacco ever used’ appear to be the most important variables for predicting youth drug use. This approach facilitates targeted interventions and policy recommendations aimed at mitigating substance abuse among adolescents. Insights gained from this study contribute to the development of effective prevention strategies and public health initiatives.

INTRO AND OVERVIEW

This report addresses decision trees, bagging and Random Forest techniques to find out about youth drug use, leveraging data from the National Survey on Drug Use and Health. It explains the techniques involved behind prediction of factors such as alcohol age of first use, marijuana ever used, and number of days of cigarettes in past month by a person. The methods used in the predictions of the target variables involve decision trees, bagging and Random Forest. The dataset involved for this project is the survey data from National Survey on Drug Use and Health. It is the leading source of statistical data on alcohol, tobacco, drug use, mental health, and other health-related issues in the United States among the general population. It consists of about 33,000 responses to nearly 3,000 questions about drug use and health. It consists of about 2890 potential variables to investigate and about 5500 entries.

THEORETICAL BACKGROUND

The methods used for the predictions include Decision Tree Regressor, Bagging, and Random Forest Classifier.

Decision Tree Regressor is used to predict a quantitative variable called ‘alcohol age of first use’. The Decision Tree Regressor is a powerful algorithm for supervised learning tasks in regression analysis. It constructs a tree-like structure where each internal node represents a decision based on a feature, and each leaf node represents the output (regression) value. The tree is built recursively by splitting the data into subsets based on features, aiming to minimize the variance of the target variable within each subset. The splits are determined by maximizing information gain or minimizing impurity measures like Gini impurity or entropy. The resulting model is interpretable and intuitive, making it suitable for understanding the relationships between features and the target variable. However, decision trees are prone to overfitting, especially with complex datasets, which can be mitigated using techniques like pruning or ensemble methods such as Random Forest. Overall, Decision Tree Regressor is versatile, easy to understand, and capable of capturing nonlinear relationships in the data. As the target vector is quantitative Decision Tree Regressor seems to be right fit involving regression. The tuning parameters used for building this model were selected and tuned using 5 fold cross validation.

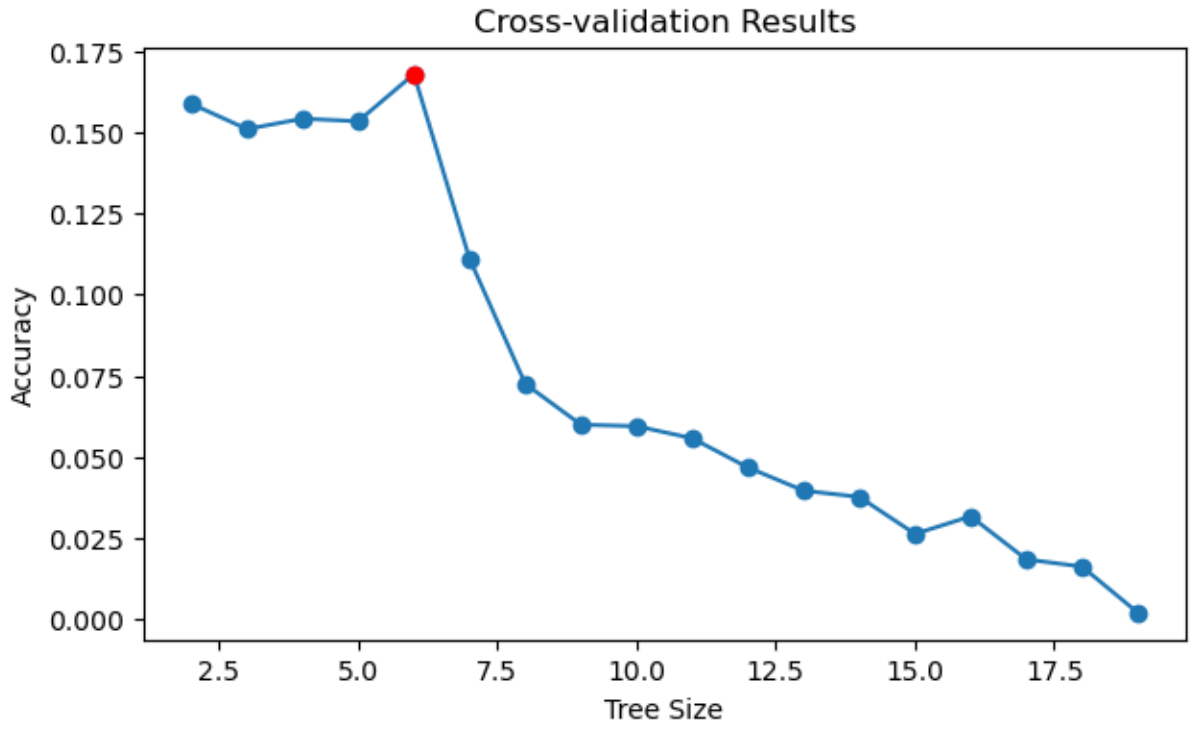
Bagging is used to predict a binary variable called ‘marijuana ever used’. Bagging, or Bootstrap Aggregating, is an ensemble learning technique that aims to improve the stability and accuracy of machine learning models. It works by training multiple models on different subsets of the training data, drawn with replacement (bootstrap samples). Each model is trained independently, typically using the same base learning algorithm. Bagging reduces overfitting and variance by combining predictions from multiple models through averaging or voting. It is particularly effective when the base models exhibit high variance. Popular bagging algorithms include Random Forest, where bagging is applied to Decision Trees, and Bagged Decision Trees, which employs bagging directly on Decision Trees. As the target vector is qualitative Random Forest Classifier seems to be right fit involving classification with bagging. The tuning parameters used for building this model are ‘max_features’ which was set to include all features and ‘n_estimators’ which refers to number of trees which was set to 500.

Random Forest Classifier is used to predict multi-class variable which is ‘number of days of cigarettes in past month’. Random Forest Classifier is an ensemble learning method based on decision trees. It operates by constructing a multitude of decision trees during training and outputting the mode of the classes (classification) predicted by individual trees. Each tree is trained on a random subset of the training data and a random subset of features, mitigating overfitting and improving robustness. The algorithm combines the predictions of multiple trees to produce more accurate and stable results than individual trees alone. It is effective for both classification and regression tasks, handling large datasets with high dimensionality. Random Forests are resistant to overfitting and perform well with minimal hyperparameter tuning. They are widely used in various applications, including bioinformatics, finance, and marketing, where high accuracy and interpretability are essential. Overall, Random Forest Classifier is a powerful and versatile algorithm suitable for a wide range of machine learning tasks. As the target vector is qualitative ,Random Forest Classifier seems to be right fit involving classification but with subset of variables considered at each split. The tuning parameters used for building this model are ‘max_features’ which was set based on the number of features which gave highest accuracy and ‘n_estimators’ which refers to number of trees which was set to 500.

METHODOLOGY

There is good amount of data processing carried out in order to clean the dataand make it suitable for applying the machine learning models. First of all, all the irrelevant variables were removed from the dataset. Then imputation was carried for certain columns in order to handle the missing values. Any rows having nulls were dropped. The duplicate rows were also dropped. The variables which were highly correlated with the target vector were dropped before applying each model.

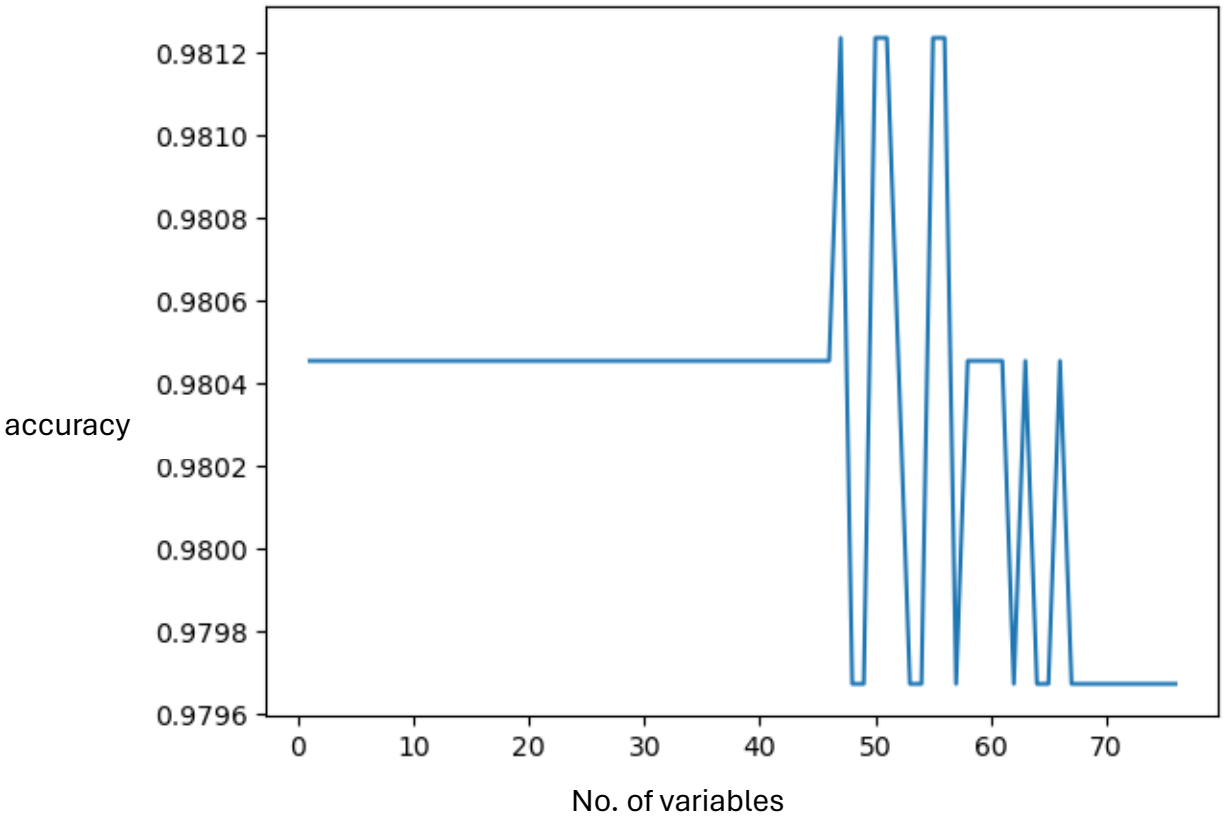
Decision Tree Regressor is used to predict a quantitative variable called ‘alcohol age of first use’. Here the target variable consisted of an encoded value which was just a category. So all observations having this category were removed. The features with its data were then converted to a matrix. The dataset was split into train and test sets with test size=0.3. The Decision Tree Regressor was then fit to the training set. Using GridSearchCV with 'max_leaf_nodes' in range of (2, 20) and cv=20 and grid search the best model was identified. The model achieved using grid search a test error of 4.02. The best tree size is 6 after pruning.



Bagging is used to predict a binary variable called ‘marijuana ever used’. The features with its data were converted to a matrix. The dataset was split into train and test sets with test size=0.3. The Random Forest Classifier was then fit to the training set with max_features=73, n_estimators which is number of trees=500. The model achieved a accuracy of 0.90.

Random Forest Classifier is used to predict multi-class variable which is ‘number of days of cigarettes in past month’. The features with its data were converted to a matrix. The dataset was split into train and test sets with test size=0.3. The Random Forest Classifier was then fit to the training set with max_features->selected by comparing accuracy of number of subset of variables of all sizes, and n_estimators=500. Here the model with 47 features achieved highest accuracy of 0.9812. The plot of number of variables against accuracy is given below.

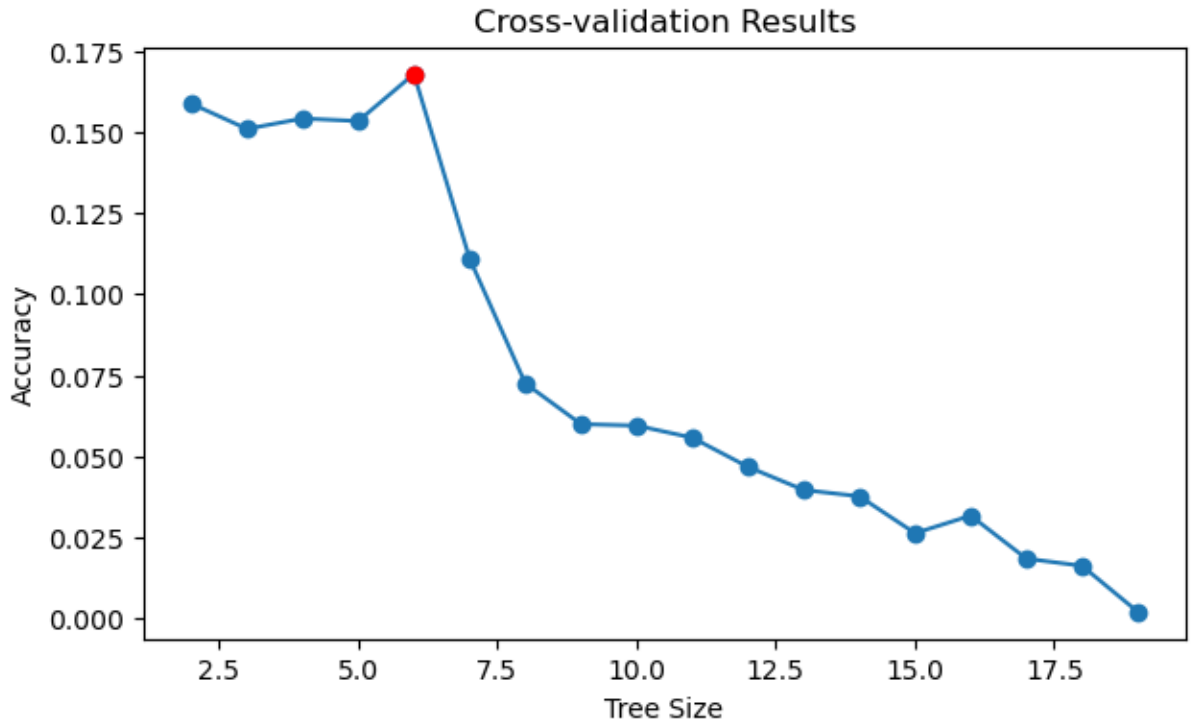
.



COMPUTATIONAL RESULTS

- Decision Tree Regressor

This model predicted the quantitative variable called ‘alcohol age of first use’ well efficiently. The model achieved using grid search a test error of 4.02. The best tree size is 6 after pruning.The graph of cross validation results is given below

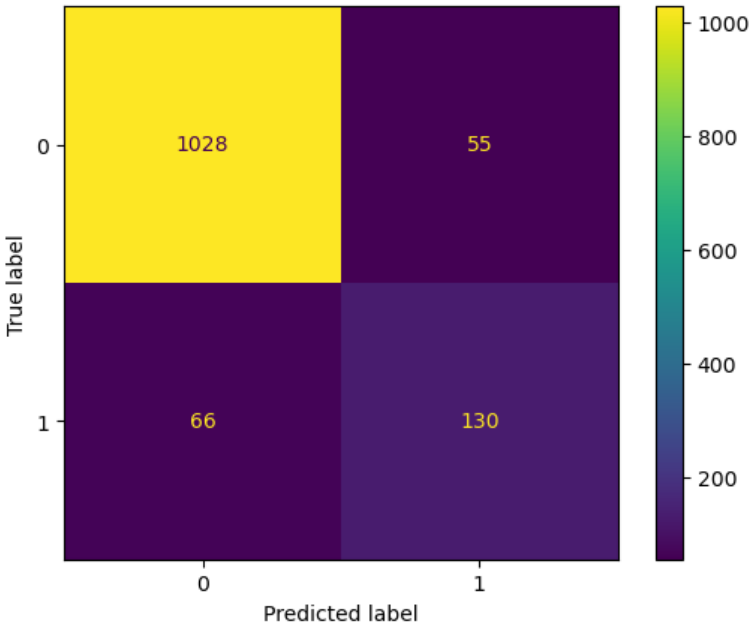


- Bagging

This model predicted a binary variable called ‘marijuana ever used’. The model achieved a accuracy of 0.90. The most important variables of the model are ‘first used alcohol prior to age 18’, ‘alcohol frequency past year’, ‘alcohol ever used’ .

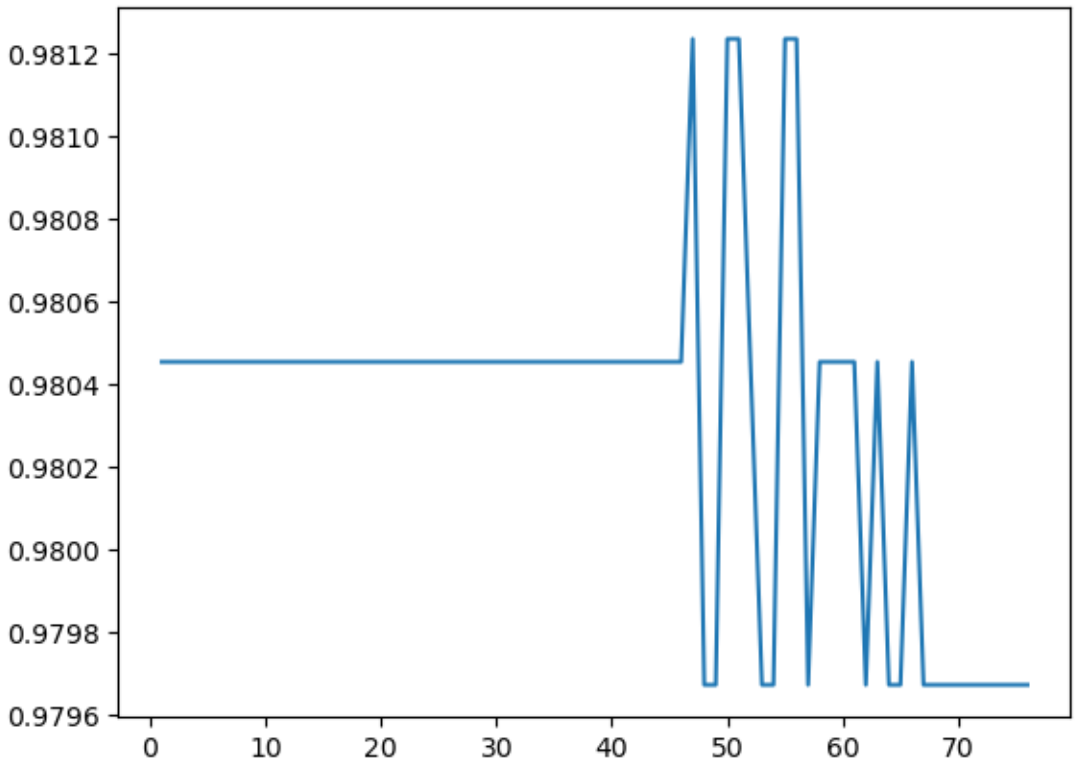
Confusion matrix for the test set:

Truth	0	1
Predicted		
0	1028	66
1	55	130



- Random Forest Classifier

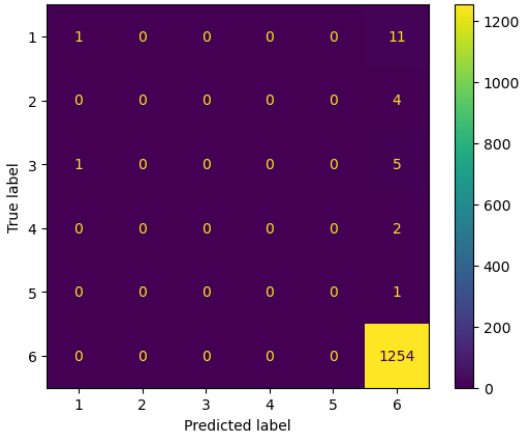
  This model predicted multi-class variable which is ‘number of days of cigarettes in past month’. The plot of number of variables against accuracy is given below.



Here the model with 47 features achieved highest accuracy of 0.9812. The important features of this model are: ‘any tobacco ever used’, ‘marijuana frequency past year’, ‘marijuana frequency past month’

The confusion matrix is given below for the test set

Truth	1	2	3	4	5	6
Predicted						
1	1	0	1	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0
6	11	4	5	2	1	1254

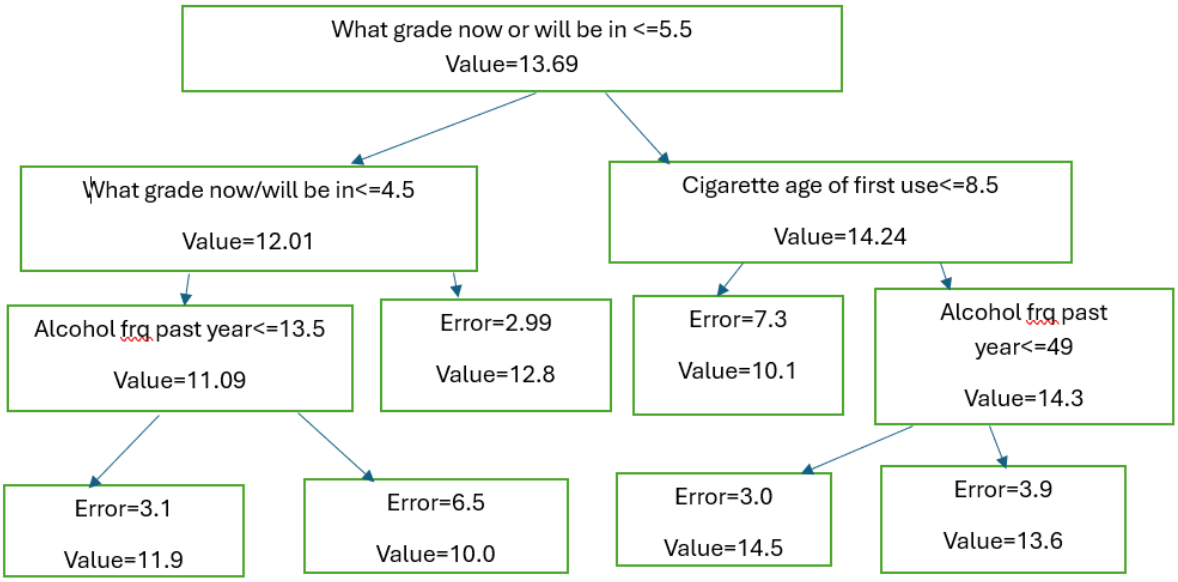

1 0 0 0 0 11
0 0 0 0 0 4
1 0 0 0 0 5
0 0 0 0 0 2
0 0 0 0 0 1
0 0 0 0 0 1254
True label
Predicted label
1 2 3 4 5 6
1200
1000
800
600
400
200
0

DISCUSSION

- Decision Tree Regressor

The model is able to predict unseen data very efficiently with a error of 4.02. Key finding is the most important variables of the model are ‘alcohol frequency past year’, ‘what grade in now/will be in’ , and ‘cigarette age of first use’. So ‘alcohol age of first use’ would surely depend on its frequency past year meaning if if higher frequency would mean the person is taking alcohol from very long time, and so age of first use could get predicted accurately. ‘What grade now or will be in’ also is useful as mindsets of teenager and adult differ. Similarly cigarette age of first use can determine about alcohol’s also as both are related and are part of drugs. The grade could determine if person is too young or not.

The model in terms of tree is represented below.



In the above regression tree, when value of ‘what grade now/will be in’ < =5.5, then the observation gets directed to node: ‘what grade now/will be in’ which when <=4.5 gets directed to ‘alcohol freq past year’ , which when <=13.5 then the observation gets assigned value of 11.9 which is avrage of all samples in that node. In the same fashion if observation has value >= 5.5 then the observation gets directed to node ‘Cigarette age of first use’ and if it has value lesss than 8.5 then it gets 10.1 as value assigned to it.

Lets say a new observation has its value for ‘what grade now/will be in’ > =5.5, so then it gets directed to node ‘Cigarette age of first use’ and if it has value lesss than or equal to

8.5 then it gets 10.1 as value assigned to it. So this 10.1 is average value of all samples in this node.

The selection of binary, ordinal, and numerical variables in decision trees and integration affects the predictions and insights derived from the data. Binary variables (such as gender or presence/absence of indicators) provide differences that decision trees can easily use for discrimination. By providing a direct definition of segmentation, they are suitable for processing categorical data with a binary structure. If coded correctly, the decision tree can use this code for effective partitioning, leading to better relationships between clusters. Ordinal variables are important if the data present a clear sequence or group order and help reveal order-based patterns . Decision trees can allocate different numbers to different locations, capturing the relationship between different predictions and different goals. They are ideal for data with regular behavior and allow for a detailed understanding of how changes in numbers affect predictions. Binary variables are suitable for clear yes/no scenarios, ordinal variables are suitable for ordered categories, and numerical variables are suitable for continuous data. Understanding the characteristics of the data and the needs of the problem can guide the selection of the most appropriate data to achieve accurate predictions and useful insights.

Improvements that could be by carrying out significant amount of Hyperparameter Tuning and Feature Engineering

Feature importances table:

Feature_name	importance
GRADE NOW/WILL	0.717
CIGARETTE FIRST USE	0.109
# DAYS ALCOHOL PAST YEAR	0.087

- Bagging

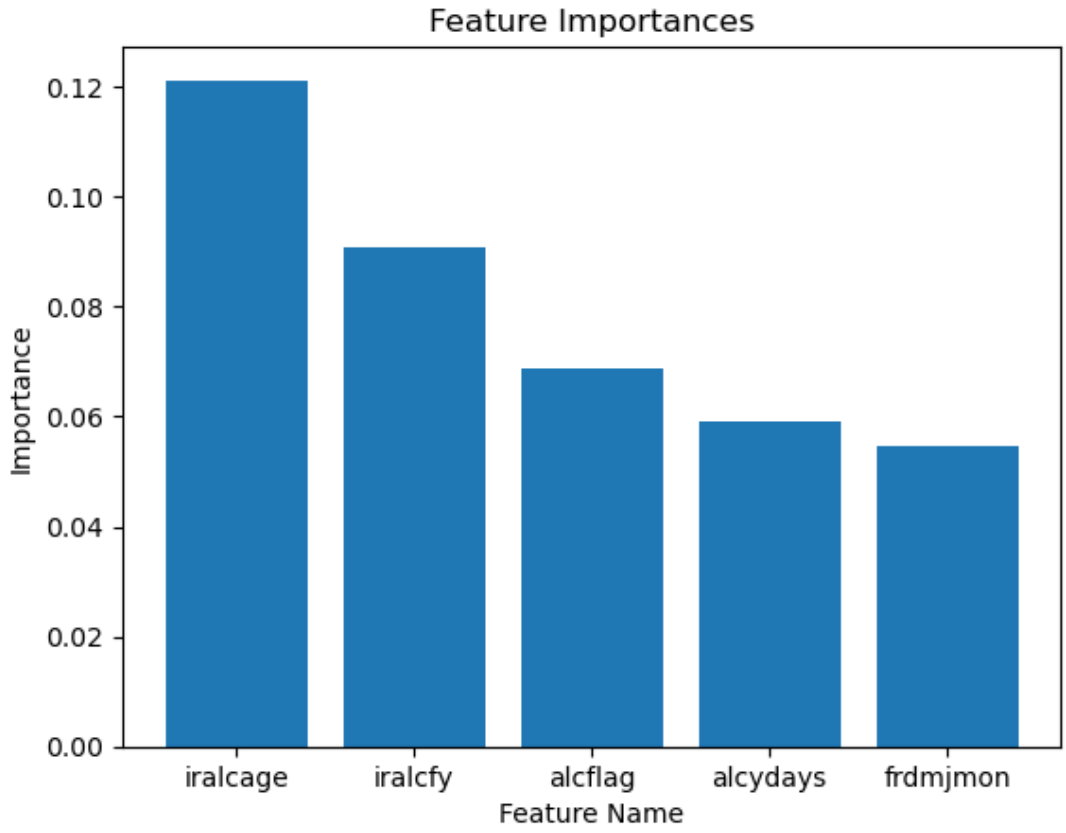
The model is able to predict unseen data very efficiently with a accuracy of 0.90. Key finding is the most important variables of the model are ‘first used alcohol prior to age 18’, ‘alcohol frequency past year’, ‘alcohol ever used’. It classified 1158 labels correctly and 121 labels incorrectly out of 1279 records. For predicting ‘marijuana ever used’, ‘first used alcohol prior to age 18’’ could determine if person had consumed any related drug like alcohol recently so his consumption status could become an important factor, similary ‘alcohol frequency past year’ could determine if person was into consumption of drugs recently, while alcohol ever used can determine if person has consumed a related drug ever.

Improvements that could be by carrying out significant amount of Hyperparameter Tuning and Feature Engineering.

Feature importances table:

Feature_name	importance
FIRST USED ALCOHOL PRIOR TO AGE 18	0.121
ALCOHOL FREQUENCY PAST YEAR	0.0907
ALCOHOL EVER USED	0.0687

Feature importances graph:



- Random Forest Classifier

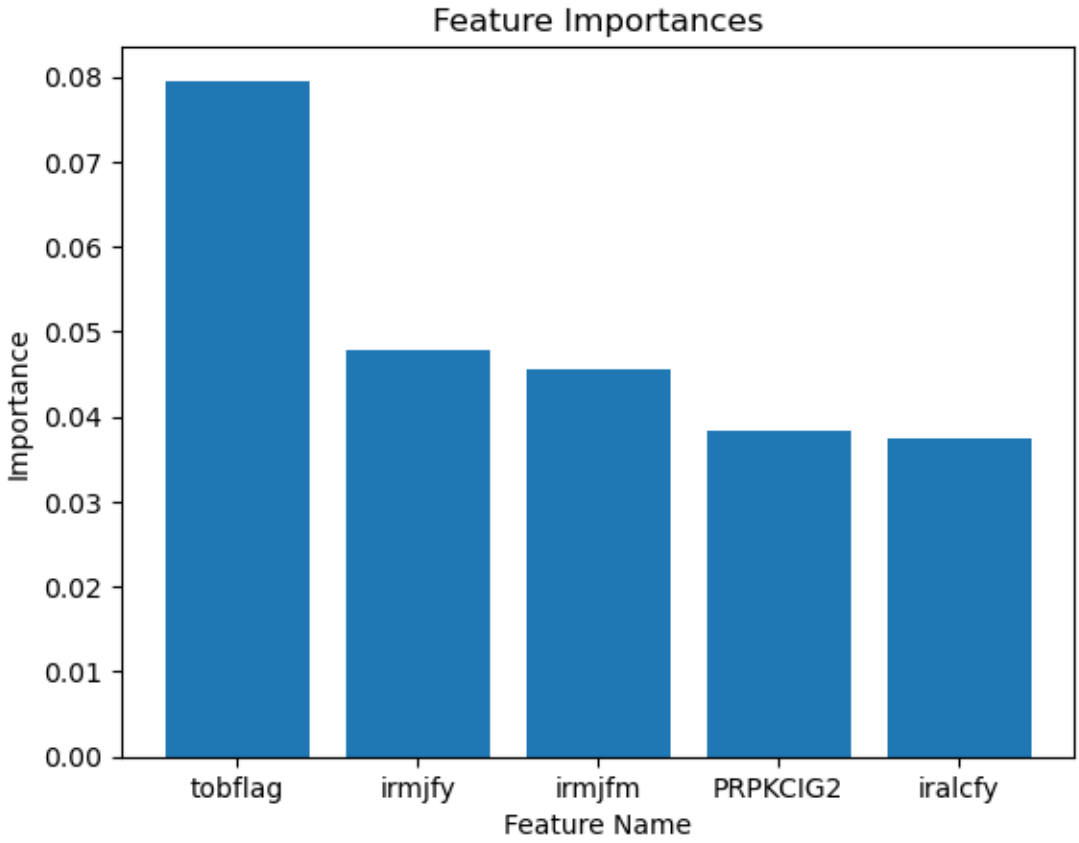
The model is able to predict unseen data very efficiently with a accuracy of 0.9812. Key finding is the important features of this model are: 'any tobacco ever used', 'marijuana frequency past year', 'marijuana frequency past month'. It classified 1255 labels correctly and 24 labels incorrectly out of 1279 records. For predicting 'number of days of cigarettes in past month' so as tobacco is a related drug it is a very good predictor in terms of usage, same goes marijuana consumption past year and past month.

Improvements that could be by carrying out Advanced Ensemble Methods like XGBoost and taking care of Parallelization and Optimization

Feature importances table:

Feature_name	importance
ANY TOBACCO EVER USED	0.079
# DAYS MARIJUANA PAST YEAR	0.047
# DAYS MARIJUANA PAST MONTH	0.045

Feature importances graph:



CONCLUSION

From the findings it is clear that ‘what grade in now/will be in’ forms a very important predictor to predict ‘alcohol age of first use’. Similarly ‘first used alcohol prior to age 18’ forms a very important predictor to predict ‘marijuana ever used’. Number of people who never used marijuana seem to be way higher than people who used it. Similarly ‘any tobacco ever used’ forms a very important predictor to predict ‘number of days of cigarettes in past month’. So ‘No Past Month Use’ seem to be way higher than any number of days used cigarettes in past month.

BIBLIOGRAPHY

[1] W3schools. URL: https://www.w3schools.com/
[2] An Introduction to Statistical Learning with Applications in Python/R, by James, Witten, Hastie, & Tibshirani. URL: https://www.statlearning.com/

In [1]:
```
import numpy as np
import pandas as pd
from matplotlib.pyplot import subplots
import sklearn.model_selection as skm
from ISLP import load_data, confusion_table
from ISLP.models import ModelSpec as MS
from sklearn.tree import (DecisionTreeClassifier as DTC,
                          DecisionTreeRegressor,
                          plot_tree,
                          export_text)
from sklearn.metrics import (accuracy_score,
                             log_loss)
from sklearn.ensemble import \
     (RandomForestRegressor as RF,
      GradientBoostingRegressor as GBR)
from ISLP.bart import BART
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import OneHotEncoder, LabelBinarizer
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import OneHotEncoder, LabelBinarizer
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

Data Preprocessing

In [2]:
```
#loading the csv file
df=pd.read_csv("youth_data.csv")
#df
```

In [3]:
```
#display columns and datatypes
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5500 entries, 0 to 5499
Data columns (total 79 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   iralcfy      5500 non-null   int64
 1   irmjfy       5500 non-null   int64
 2   ircigfm      5500 non-null   int64
 3   IRSMKLSS30N  5500 non-null   int64
 4   iralcfm      5500 non-null   float64
 5   irmjfm       5500 non-null   float64
 6   ircigage     5500 non-null   int64
 7   irsmklsstry  5500 non-null   int64
 8   iralcage     5500 non-null   int64
 9   irmjage      5500 non-null   int64
 10  mrjflag      5500 non-null   int64
 11  alcflag      5500 non-null   int64
 12  tobflag      5500 non-null   int64
 13  alcydays     5500 non-null   int64
 14  mrjydays     5500 non-null   int64
 15  alcmdays     5500 non-null   int64
 16  mrjmdays     5500 non-null   int64
 17  cigmdays     5500 non-null   int64
 18  smklsmdays   5500 non-null   int64
 19  schfelt      5500 non-null   int64
 20  tchgjob      5484 non-null   float64
 21  avggrade     5147 non-null   float64
 22  stndscig     5247 non-null   float64
 23  stndsmj      5241 non-null   float64
 24  stndalc      5258 non-null   float64
 25  stnddnk      5202 non-null   float64
 26  parchkhw     5481 non-null   float64
 27  parhlphw     5462 non-null   float64
 28  PRCHORE2     5482 non-null   float64
 29  PRLMTTV2     5458 non-null   float64
 30  parlmtsn     5368 non-null   float64
 31  PRGDJOB2     5478 non-null   float64
 32  PRPROUD2     5479 non-null   float64
 33  argupar      5413 non-null   float64
 34  YOFIGHT2     5477 non-null   float64
 35  YOGRPFT2     5471 non-null   float64
 36  YOHGUN2      5477 non-null   float64
 37  YOSELL2      5485 non-null   float64
 38  YOSTOLE2     5487 non-null   float64
 39  YOATTAK2     5487 non-null   float64
 40  PRPKCIG2     5443 non-null   float64
 41  PRMJEVR2     5440 non-null   float64
 42  prmjmo       5431 non-null   float64
 43  PRALDLY2     5445 non-null   float64
 44  YFLPKCG2     5448 non-null   float64
 45  YFLTMRJ2     5445 non-null   float64
 46  yflmjmo      5447 non-null   float64
 47  YFLADLY2     5448 non-null   float64
 48  FRDPCIG2     5413 non-null   float64
 49  FRDMEVR2     5416 non-null   float64
 50  frdmjmon     5411 non-null   float64
 51  FRDADLY2     5420 non-null   float64
 52  talkprob     5354 non-null   float64
 53  PRTALK3      5410 non-null   float64
 54  PRBSOLV2     5334 non-null   float64
 55  PREVIOL2     5421 non-null   float64
 56  PRVDRGO2     5433 non-null   float64
 57  GRPCNSL2     5434 non-null   float64
 58  PREGPGM2     5449 non-null   float64
 59  YTHACT2      5474 non-null   float64
 60  DRPRVME3     5411 non-null   float64
 61  ANYEDUC3     5422 non-null   float64
 62  rlgattd      5365 non-null   float64
 63  rlgimpt      5380 non-null   float64
 64  rlgdcsn      5369 non-null   float64
 65  rlgfrnd      5347 non-null   float64
 66  irsex        5500 non-null   int64
 67  NEWRACE2     5500 non-null   int64
 68  HEALTH2      5499 non-null   float64
 69  eduschlgo    5500 non-null   int64
```

```
 70  EDUSCHGRD2   5500 non-null   int64
 71  eduskpcom    5500 non-null   int64
 72  imother      5500 non-null   int64
 73  ifather      5500 non-null   int64
 74  income       5500 non-null   int64
 75  govtprog     5500 non-null   int64
 76  POVERTY3     5500 non-null   int64
 77  PDEN10       5500 non-null   int64
 78  COUTYP4      5500 non-null   int64
dtypes: float64(49), int64(30)
memory usage: 3.3 MB
```

In [4]:
```
#dropping duplicate rows
df.drop_duplicates(inplace=True)
```

In [5]:
```
#dropping rows having null values
df=df.dropna()
```

In [6]:
```
df.isna().sum()
```

Out[6]:
```
iralcfy        0
irmjfy         0
ircigfm        0
IRSMKLSS30N    0
iralcfm        0
              ..
income         0
govtprog       0
POVERTY3       0
PDEN10         0
COUTYP4        0
Length: 79, dtype: int64
```

Regression

iralcage --alcohol age of first use

RANGE = 1 - 66

Method: Decision Tree Regressor

In [7]:
```
df.corr()
```

Out[7]:

	iralcfy	irmjfy	ircigfm	IRSMKLSS30N	iralcfm	irmjfm	ircigage	irsmklsstry	iralcage	irmjage	...
iralcfy	1.000000	0.505847	0.200616	0.104151	0.582621	0.402221	0.344839	0.215347	0.871142	0.524586	...
irmjfy	0.505847	1.000000	0.211089	0.101352	0.409150	0.663708	0.377101	0.225436	0.500621	0.893295	...
ircigfm	0.200616	0.211089	1.000000	0.202205	0.188892	0.269435	0.437386	0.264104	0.186367	0.233165	...
IRSMKLSS30N	0.104151	0.101352	0.202205	1.000000	0.105629	0.077804	0.187793	0.487343	0.109571	0.120407	...
iralcfm	0.582621	0.409150	0.188892	0.105629	1.000000	0.400261	0.291421	0.219283	0.510764	0.420731	...
...	...	...	...	...	...	...	...	...	...	...	...
income	-0.062033	-0.009531	0.011648	-0.008260	-0.055055	-0.021057	0.048654	-0.017285	-0.047337	0.000812	...
govtprog	-0.031855	-0.004190	0.015006	-0.025880	-0.017476	-0.003245	0.036572	-0.032159	-0.023999	0.014217	...
POVERTY3	-0.075172	-0.011591	0.001156	-0.014336	-0.054141	-0.019101	0.027760	-0.021881	-0.064840	-0.002196	...
PDEN10	-0.017435	-0.013408	-0.062501	-0.045583	-0.004731	0.007493	-0.080876	-0.059250	-0.013851	-0.024033	...
COUTYP4	-0.027651	-0.021574	-0.064109	-0.048503	-0.017672	-0.004051	-0.088261	-0.060455	-0.031565	-0.030360	...

79 rows × 79 columns

In [8]:
```
df.shape
```

Out[8]:
```
(4262, 79)
```

In [ ]:
```
```

In [9]:
```
df['iralcage'].unique()
```

Out[9]:
```
array([ 12, 991,  13,  15,  16,  14,   8,   2,  10,  11,   9,  17,   6,
         3,   7,   5,   1,   4], dtype=int64)
```

In [10]:
```
#removing the rows that have iralcage value as a category
data = df.loc[(df['iralcage'] != 991), :]
```

In [11]:
```
data.corr()
```

Out[11]:

	iralcfy	irmjfy	ircigfm	IRSMKLSS30N	iralcfm	irmjfm	ircigage	irsmklsstry	iralcage	irmjage	...
iralcfy	1.000000	0.187299	0.083596	0.019681	0.325872	0.172410	0.034831	0.054999	-0.284825	0.143117	...
irmjfy	0.187299	1.000000	0.144037	0.051000	0.235923	0.601214	0.262400	0.143918	-0.062753	0.859082	...
ircigfm	0.083596	0.144037	1.000000	0.195213	0.116908	0.228935	0.421081	0.253995	0.010502	0.174203	...
IRSMKLSS30N	0.019681	0.051000	0.195213	1.000000	0.064305	0.034720	0.159828	0.481201	0.005596	0.076717	...
iralcfm	0.325872	0.235923	0.116908	0.064305	1.000000	0.281214	0.136934	0.142132	-0.055220	0.232560	...
...	...	...	...	...	...	...	...	...	...	...	...
income	-0.085082	0.004523	0.002882	-0.025990	-0.072388	-0.030492	0.103482	-0.018625	0.113888	0.025967	...
govtprog	-0.044047	-0.002211	-0.007464	-0.050087	-0.011960	0.005874	0.073199	-0.048815	0.062507	0.037788	...
POVERTY3	-0.078431	0.018246	-0.005257	-0.032764	-0.050486	-0.012675	0.085364	-0.017253	0.073315	0.049017	...
PDEN10	-0.021092	-0.012966	-0.120251	-0.084730	0.005113	0.031257	-0.148152	-0.096132	-0.053126	-0.046704	...
COUTYP4	-0.000700	-0.009178	-0.119206	-0.091178	-0.003441	0.029113	-0.143394	-0.094071	-0.033322	-0.041702	...

79 rows × 79 columns

In [ ]:
```
```

In [12]:
```
data.drop(columns=['iralcage','alcydays'],axis=1).columns
```

Out[12]:
```
Index(['iralcfy', 'irmjfy', 'ircigfm', 'IRSMKLSS30N', 'iralcfm', 'irmjfm',
       'ircigage', 'irsmklsstry', 'irmjage', 'mrjflag', 'alcflag', 'tobflag',
       'mrjydays', 'alcmdays', 'mrjmdays', 'cigmdays', 'smklsmdays', 'schfelt',
       'tchgjob', 'avggrade', 'stndscig', 'stndsmj', 'stndalc', 'stnddnk',
       'parchkhw', 'parhlphw', 'PRCHORE2', 'PRLMTTV2', 'parlmtsn', 'PRGDJOB2',
       'PRPROUD2', 'argupar', 'YOFIGHT2', 'YOGRPFT2', 'YOHGUN2', 'YOSELL2',
       'YOSTOLE2', 'YOATTAK2', 'PRPKCIG2', 'PRMJEVR2', 'prmjmo', 'PRALDLY2',
       'YFLPKCG2', 'YFLTMRJ2', 'yflmjmo', 'YFLADLY2', 'FRDPCIG2', 'FRDMEVR2',
       'frdmjmon', 'FRDADLY2', 'talkprob', 'PRTALK3', 'PRBSOLV2', 'PREVIOL2',
       'PRVDRGO2', 'GRPCNSL2', 'PREGPGM2', 'YTHACT2', 'DRPRVME3', 'ANYEDUC3',
       'rlgattd', 'rlgimpt', 'rlgdcsn', 'rlgfrnd', 'irsex', 'NEWRACE2',
       'HEALTH2', 'eduschlgo', 'EDUSCHGRD2', 'eduskpcom', 'imother', 'ifather',
       'income', 'govtprog', 'POVERTY3', 'PDEN10', 'COUTYP4'],
      dtype='object')
```

In [13]:
```
#creating feature matrix
model = MS(data.drop(columns=['iralcage','alcydays'],axis=1).columns, intercept=False)
D = model.fit_transform(data)
feature_names = list(D.columns)
X = np.asarray(D)
#list(D.columns)
```

In [14]:
```
#splitting dataset into train and test sets
(X_train,
 X_test,
 y_train,
 y_test) = skm.train_test_split(X,
                                data['iralcage'],
                                test_size=0.3,
```

```
                                    random_state=0)
X_test.shape
```

Out[14]:
```
(347, 77)
```

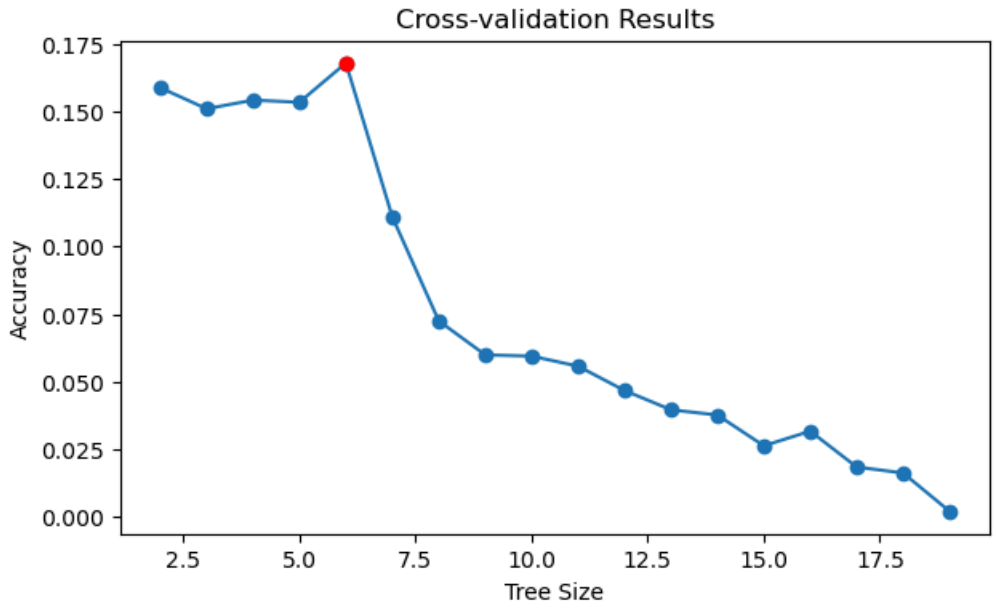
In [15]:
```
# #applying decision tree regressor model
# reg = DTR()
# reg.fit(X_train, y_train)
```

In [16]:
```
# #use the cost_complexity_pruning_path() method of clf to extract cost-complexity values and extracting an opti
# ccp_path = reg.cost_complexity_pruning_path(X_train, y_train)
# kfold = skm.KFold(5,
#                   shuffle=True,
#                   random_state=10)
# grid = skm.GridSearchCV(reg,
#                         {'ccp_alpha': ccp_path.ccp_alphas},
#                         refit=True,
#                         cv=kfold,
#                         scoring='neg_mean_squared_error')
# G = grid.fit(X_train, y_train)
```

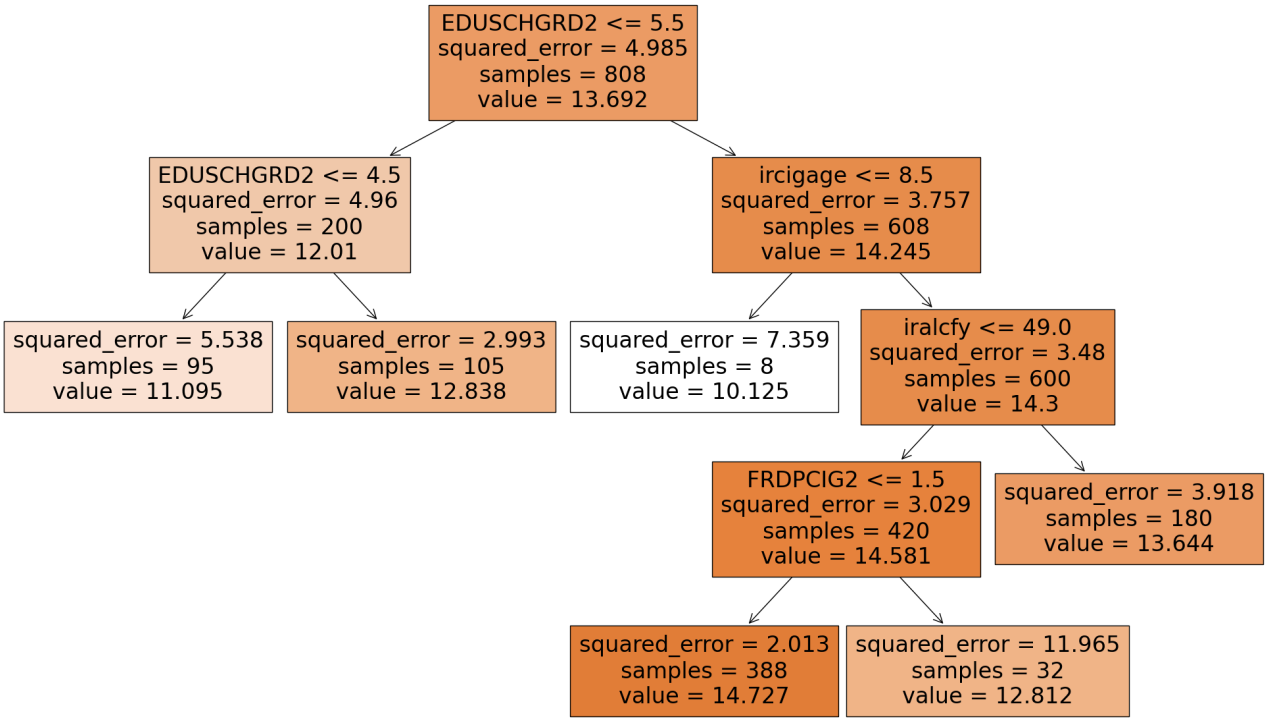
In [17]:
```
# #calculate MSE
# best_ = grid.best_estimator_
# np.mean((y_test - best_.predict(X_test))**2)
```

In [ ]:
```

```

In [18]:
```
# fit decision tree model
tree_ = DecisionTreeRegressor(random_state = 1)
tree_.fit(X_train, y_train)

# cross-validation to determine optimal tree size
params = {'max_leaf_nodes': range(2, 20)}
cv_ = GridSearchCV(tree_, params, cv=20)
cv_.fit(X_train, y_train)
cv_results = cv_.cv_results_

# find the best score for max leaf nodes
best_size = cv_.best_params_['max_leaf_nodes']
best_score = cv_.best_score_

# plot results of cross-validation
plt.figure(figsize=(7, 4))
plt.plot(cv_results["param_max_leaf_nodes"].data, cv_results["mean_test_score"], 'o-')
plt.plot(best_size, best_score, 'ro-')
plt.xlabel('Tree Size')
plt.ylabel('Accuracy')
plt.title('Cross-validation Results');
```



In [19]:
```
# prune tree using optimal size
prune_ = DecisionTreeRegressor(random_state = 1, max_leaf_nodes = best_size)
prune_.fit(X_train, y_train)

# plot pruned tree
plt.figure(figsize=(25,15))
plt.title('Pruned Tree')
plot_tree(prune_, feature_names=feature_names, filled=True);
```



In [ ]:
```
```

In [20]:
```
# Calculate MSE
MSE = ((y_test - prune_.predict(X_test))**2).mean()
print(MSE)
```

```
4.029972396173291
```

In [ ]:
```
```

In [ ]:
```
```

In [21]:
```
#extract features and their importances
importances = pd.DataFrame({'feature_name': feature_names, 'importance': prune_.feature_importances_})
importances = importances.sort_values('importance', ascending=False).reset_index(drop=True)
print(importances)
```

```
   feature_name  importance
0    EDUSCHGRD2    0.717067
1      ircigage    0.109229
2       iralcfy    0.087716
3      FRDPCIG2    0.085989
4      YFLPKCG2    0.000000
..          ...         ...
72     parhlphw    0.000000
73     parchkhw    0.000000
74      stnddnk    0.000000
75      stndalc    0.000000
76      COUTYP4    0.000000

[77 rows x 2 columns]
```

In [22]:
```
tree_summary = export_text(prune_, feature_names=feature_names)
print(tree_summary)
```

```
|--- EDUSCHGRD2 <= 5.50
|   |--- EDUSCHGRD2 <= 4.50
|   |   |--- value: [11.09]
|   |--- EDUSCHGRD2 >  4.50
|   |   |--- value: [12.84]
|--- EDUSCHGRD2 >  5.50
|   |--- ircigage <= 8.50
|   |   |--- value: [10.12]
|   |--- ircigage >  8.50
|   |   |--- iralcfy <= 49.00
|   |   |   |--- FRDPCIG2 <= 1.50
|   |   |   |   |--- value: [14.73]
|   |   |   |--- FRDPCIG2 >  1.50
|   |   |   |   |--- value: [12.81]
|   |   |--- iralcfy >  49.00
|   |   |   |--- value: [13.64]
```

Binary classification

mrjflag -- marijuana ever used (0=never, 1=ever)

Method: Bagging

In [23]:
```
df.corr()
```

Out[23]:

	iralcfy	irmjfy	ircigfm	IRSMKLSS30N	iralcfm	irmjfm	ircigage	irsmklsstry	iralcage	irmjage	...
iralcfy	1.000000	0.505847	0.200616	0.104151	0.582621	0.402221	0.344839	0.215347	0.871142	0.524586	...
irmjfy	0.505847	1.000000	0.211089	0.101352	0.409150	0.663708	0.377101	0.225436	0.500621	0.893295	...
ircigfm	0.200616	0.211089	1.000000	0.202205	0.188892	0.269435	0.437386	0.264104	0.186367	0.233165	...
IRSMKLSS30N	0.104151	0.101352	0.202205	1.000000	0.105629	0.077804	0.187793	0.487343	0.109571	0.120407	...
iralcfm	0.582621	0.409150	0.188892	0.105629	1.000000	0.400261	0.291421	0.219283	0.510764	0.420731	...
...	...	...	...	...	...	...	...	...	...	...	...
income	-0.062033	-0.009531	0.011648	-0.008260	-0.055055	-0.021057	0.048654	-0.017285	-0.047337	0.000812	...
govtprog	-0.031855	-0.004190	0.015006	-0.025880	-0.017476	-0.003245	0.036572	-0.032159	-0.023999	0.014217	...
POVERTY3	-0.075172	-0.011591	0.001156	-0.014336	-0.054141	-0.019101	0.027760	-0.021881	-0.064840	-0.002196	...
PDEN10	-0.017435	-0.013408	-0.062501	-0.045583	-0.004731	0.007493	-0.080876	-0.059250	-0.013851	-0.024033	...
COUTYP4	-0.027651	-0.021574	-0.064109	-0.048503	-0.017672	-0.004051	-0.088261	-0.060455	-0.031565	-0.030360	...

79 rows × 79 columns

In [24]:
```
#creating feature matrix
#split into train and test sets
data=df.drop(columns=['mrjflag','irmjfy','irmjfm','irmjage','mrjydays','mrjmdays'])
model = MS(data.columns, intercept=False)
D = model.fit_transform(data)
feature_names = list(D.columns)
X = np.asarray(D)

(X_train,
 X_test,
 y_train,
 y_test) = skm.train_test_split(X,
                                df['mrjflag'],
                                test_size=0.3,
                                random_state=50)
```

In [25]:
```
#fit RandomForestClassifier
clf = RandomForestClassifier(max_features=X_train.shape[1],n_estimators=500, random_state=80)
clf.fit(X_train, y_train)
```

Out[25]:

RandomForestClassifier

RandomForestClassifier(max_features=73, n_estimators=500, random_state=80)

In [26]:
```
print("Number of trees:", clf.n_estimators)
print("Number of features tried at each split:",clf.max_features)
print("Training score: {:.2f}%".format(clf.score(X_train,y_train)*100))
```

```
Number of trees: 500
Number of features tried at each split: 73
Training score: 100.00%
```

In [27]:
```
y_hat_bag = clf.predict(X_test)
```

In [28]:
```
#calculate MSE
accuracy_score(y_test, y_hat_bag)
```

Out[28]:
```
0.9053948397185301
```

In [29]:
```
print("Accuracy: {:.2f}%".format(accuracy_score(y_test, y_hat_bag)*100))
```

```
Accuracy: 90.54%
```

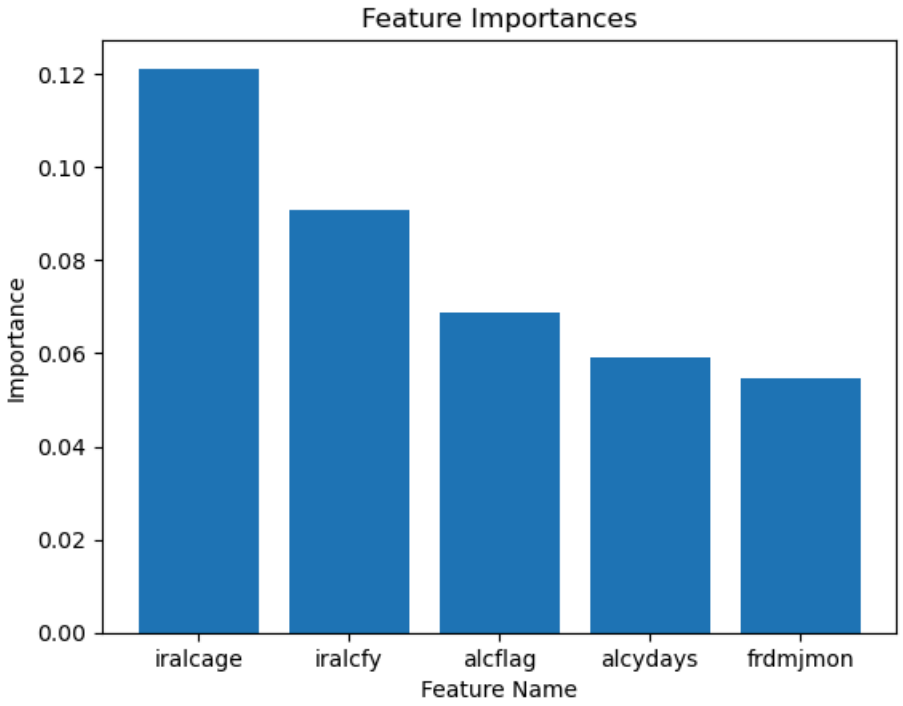
In [30]:
```
#extract features and their importances
feature_imp = pd.DataFrame(
    {'importance':clf.feature_importances_},
    index=feature_names)
features=feature_imp.sort_values(by='importance', ascending=False)
features
```

Out[30]:

	importance
iralcage	0.121223
iralcfy	0.090752
alcflag	0.068737
alcydays	0.059024
frdmjmon	0.054589
...	...
PREVIOL2	0.002479
PREGPGM2	0.002451
eduschlgo	0.002271
cigmdays	0.001656
smklsmdays	0.000529

73 rows × 1 columns

In [31]:
```
# bar plot of feature importances
plt.bar(features.index[:5], features.importance[:5])
plt.xlabel('Feature Name')
plt.ylabel('Importance')
plt.title('Feature Importances')
plt.show()
```



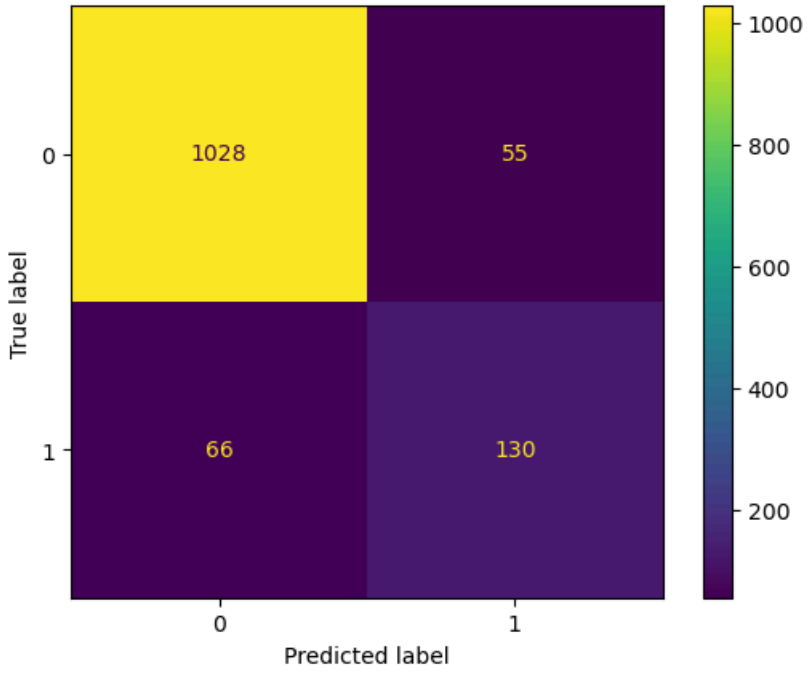
In [32]:
```
#printing confusion table
confusion_table(y_hat_bag,y_test)
```

Out[32]:

Truth	0	1
Predicted		
0	1028	66
1	55	130

In [33]:
```
ConfusionMatrixDisplay.from_predictions(y_test, y_hat_bag)
```

Out[33]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2684580a590>
```



Multi-class classification

cigmdays -- number of days of cigarettes in past month (1-5 categories, 6=none)

Method: Random Forest Classifier

In [34]:
```
df.corr()
```

Out[34]:

	iralcfy	irmjfy	ircigfm	IRSMKLSS30N	iralcfm	irmjfm	ircigage	irsmklsstry	iralcage	irmjage	...
iralcfy	1.000000	0.505847	0.200616	0.104151	0.582621	0.402221	0.344839	0.215347	0.871142	0.524586	...
irmjfy	0.505847	1.000000	0.211089	0.101352	0.409150	0.663708	0.377101	0.225436	0.500621	0.893295	...
ircigfm	0.200616	0.211089	1.000000	0.202205	0.188892	0.269435	0.437386	0.264104	0.186367	0.233165	...
IRSMKLSS30N	0.104151	0.101352	0.202205	1.000000	0.105629	0.077804	0.187793	0.487343	0.109571	0.120407	...
iralcfm	0.582621	0.409150	0.188892	0.105629	1.000000	0.400261	0.291421	0.219283	0.510764	0.420731	...
...	...	...	...	...	...	...	...	...	...	...	...
income	-0.062033	-0.009531	0.011648	-0.008260	-0.055055	-0.021057	0.048654	-0.017285	-0.047337	0.000812	...
govtprog	-0.031855	-0.004190	0.015006	-0.025880	-0.017476	-0.003245	0.036572	-0.032159	-0.023999	0.014217	...
POVERTY3	-0.075172	-0.011591	0.001156	-0.014336	-0.054141	-0.019101	0.027760	-0.021881	-0.064840	-0.002196	...
PDEN10	-0.017435	-0.013408	-0.062501	-0.045583	-0.004731	0.007493	-0.080876	-0.059250	-0.013851	-0.024033	...
COUTYP4	-0.027651	-0.021574	-0.064109	-0.048503	-0.017672	-0.004051	-0.088261	-0.060455	-0.031565	-0.030360	...

79 rows × 79 columns

In [35]:
```
data=df.drop(columns=['cigmdays','ircigfm','ircigage'])
```

In [36]:
```
#creating feature matrix and split data into train and test
model = MS(data.columns, intercept=False)
D = model.fit_transform(data)
feature_names = list(D.columns)
X = np.asarray(D)

(X_train,
 X_test,
 y_train,
 y_test) = skm.train_test_split(X,
                                df['cigmdays'],
                                test_size=0.3,
                                random_state=90)
```

In [37]:
```
X_train.shape
```

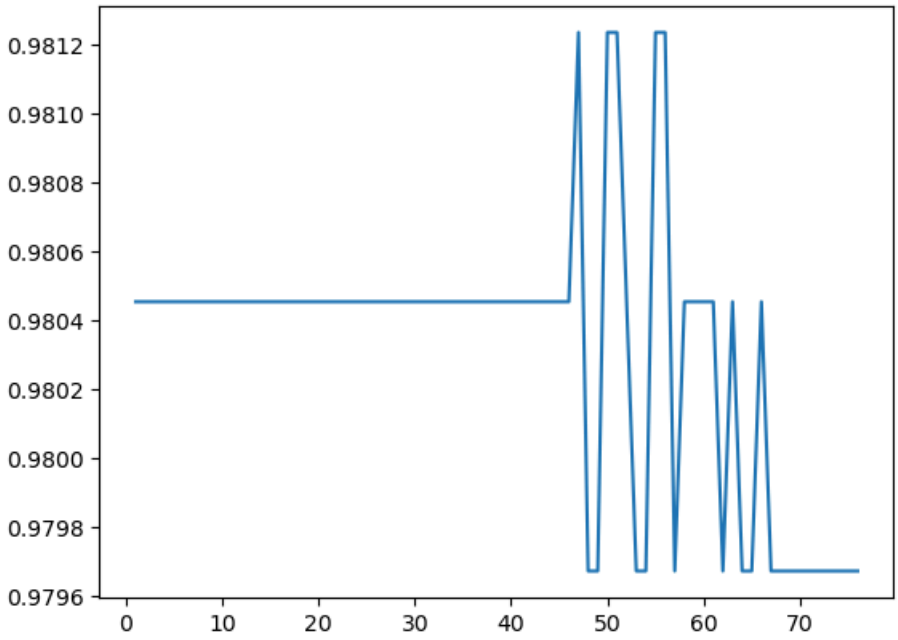
Out[37]:
```
(2983, 76)
```

In [38]:
```
np.arange(1,77)
```

Out[38]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
       35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51,
       52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68,
       69, 70, 71, 72, 73, 74, 75, 76])
```

In [39]:
```
#finding out optimal number of max_features through test accuracy
li=[]
for i in range(1,77):
    RF = RandomForestClassifier(max_features=i,n_estimators=500,
                   random_state=0).fit(X_train, y_train)
    y_hat_RF = RF.predict(X_test)
    li.append(accuracy_score(y_test, y_hat_RF))
plt.plot(np.arange(1,77),li)
```

Out[39]:
```
[<matplotlib.lines.Line2D at 0x268487fb0d0>]
```



In [40]:
```
np.max(li[0:49])
```

Out[40]:
```
0.9812353401094606
```

In [41]:
```
li.index(np.max(li[0:49]))
```

Out[41]:
```
46
```

In [42]:
```
#fitting RandomForestClassifier with max_features that has highest test accuracy
RF = RandomForestClassifier(max_features=47,n_estimators=500,
                 random_state=0).fit(X_train, y_train)
y_hat_RF = RF.predict(X_test)
accuracy_score(y_test, y_hat_RF)
```

Out[42]:
```
0.9812353401094606
```

In [43]:
```
print("Accuracy: {:.2f}%".format(accuracy_score(y_test, y_hat_RF)*100))
```

```
Accuracy: 98.12%
```

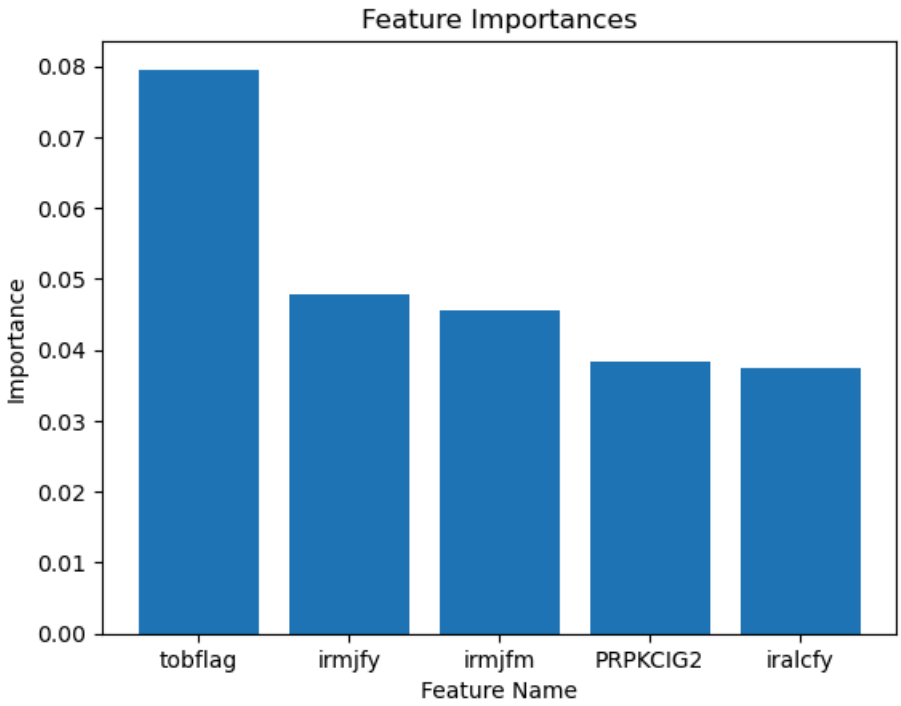
In [44]:
```
#printing features and its importances
feature_imp = pd.DataFrame(
    {'importance':RF.feature_importances_},
    index=feature_names)
features=feature_imp.sort_values(by='importance', ascending=False)
features
```

Out[44]:

	importance
tobflag	0.079515
irmjfy	0.047906
irmjfm	0.045658
PRPKCIG2	0.038287
iralcfy	0.037453
...	...
YOSTOLE2	0.003646
PREVIOL2	0.003053
mrjflag	0.002299
alcflag	0.001556
PREGPGM2	0.000694

76 rows × 1 columns

In [45]:
```
# bar plot of feature importances
plt.bar(features.index[:5], features.importance[:5])
plt.xlabel('Feature Name')
plt.ylabel('Importance')
plt.title('Feature Importances')
plt.show()
```



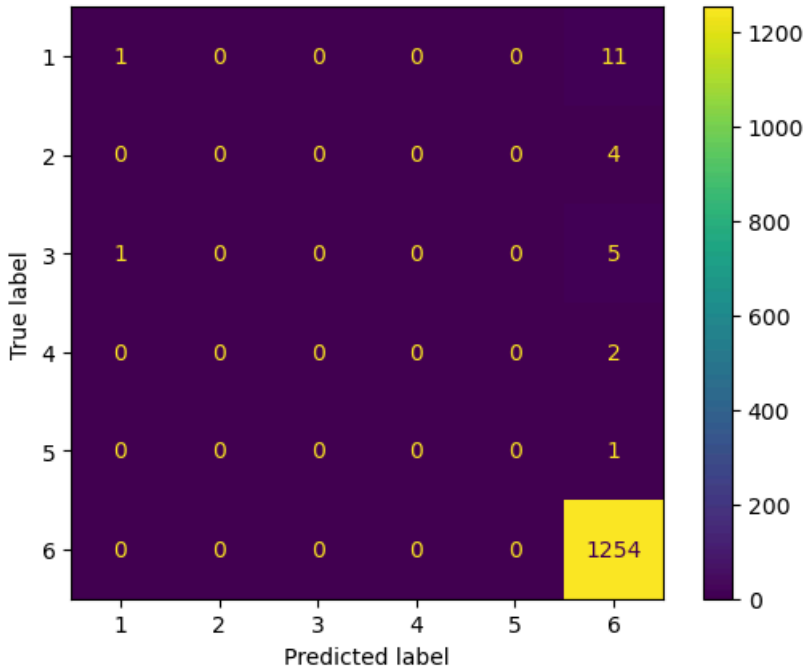
In [46]:
```
#printing confusion table
confusion_table(y_hat_RF,y_test)
```

Out[46]:

Truth	1	2	3	4	5	6
Predicted						
1	1	0	1	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0
6	11	4	5	2	1	1254

In [47]:
```
ConfusionMatrixDisplay.from_predictions(y_test, y_hat_RF)
```

Out[47]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x26847b32d10>
```



In [ ]: