Data 5322 Statistical Machine Learning 2

Decision Trees

Nidhi Trivedi

Abstract

This report states the study factors correlated with youth drug use using data from the National Survey on Drug Use and Health (NSDUH). The dataset contains detailed information on respondents' demographics, youth experiences, and use of various drugs. Our analysis employs decision trees, including ensemble methods, to build predictive models for three problem types: binary classification, multi-class classification and regression. This study sheds light on the complex relationship between various factors and youth drug use, providing insights into predictive modeling using decision trees. Throughout the report, we present visual representations of decision trees, providing a human-readable depiction of the decision-making process within the models. Furthermore, we discuss the flow of selected decision trees, tracing the paths from root to leaf nodes and elucidating the implications of noteworthy end nodes.

Introduction

Lots of young people use drugs, which is a big problem for their health and the community. To understand why this happens, we're using information from a big survey called the National Survey on Drug Use and Health (NSDUH). In this report, we're using "decision trees" to figure out why young people use alcohol and how often they use them. Through binary and multi-class classification, as well as regression analyses, we aim to distinguish between users and non-users of specific substances, ascertain the frequency of alochol use, and predict alcohol consumption among youth populations. Furthermore, we examine the influence of various types of variables on predictive accuracy and address ethical considerations in data interpretation.

Overview

This report takes a deep dive into the world of youth drug use, using data from the National Survey on Drug Use and Health (NSDUH) to uncover the driving forces behind this behavior.

We explore three key dimensions: binary classification, multi-class classification, and regression, to understand who uses drugs, how often they use them, and predict alcohol consumption among youths. By leveraging decision trees, we strive to provide clear and actionable insights into youth drug use dynamics. Additionally, we investigate the impact of different types of variables on our predictions and ensure ethical considerations are addressed in our analysis. Our ultimate goal is to not only understand but also effectively address youth drug use, contributing to the well-being of young people and society as a whole.

Theoretical Background

In our study we have used Decision tree , Regression analysis and Random Forests, which aggregate predictions of multiple decision trees, are also employed in this study. Regression analysis predicts continuous variables, such as the frequency of alcohol consumption among youth respondents. The National Survey on Drug Use and Health (NSDUH) provides detailed information on respondents' demographics, drug use history, and other factors. Decision trees offer advantages, including handling both numerical and categorical data, robustness to outliers and missing values, and interpretability.

Techniques such as pruning, limiting the maximum tree depth, and setting minimum samples per leaf node mitigate overfitting. Random Forests mitigate overfitting by aggregating predictions of

multiple decision trees. Run time for decision trees and Random Forests varies with the dataset size and tree complexity.

Methodology

The dataset used for this analysis contains 79 columns, including details on youth experiences, demographics, and substance use-related features. To prepare the data for the required classification and regression techniques, several data cleaning and preprocessing steps were performed.

Data Cleaning and Preprocessing

Handling Missing Values: All null, NA, or blank values were removed from the dataset as they did not contribute any meaningful information for the analysis.

Data Separation: Separate dataframes were created for binary classification, multi-level classification, and regression tasks to facilitate the respective analyses.

Feature Selection: Only the relevant columns required for each analysis were retained, and the remaining columns were discarded.

Data Transformation: Several columns, such as 'iralcfy', 'eduschgrd2', 'eduschlgo', and 'eduskpcom', were categorized or encoded to facilitate their use in the models.

Column Renaming: For ease of use and consistency, all column names were converted to lowercase.

Model Building and Evaluation

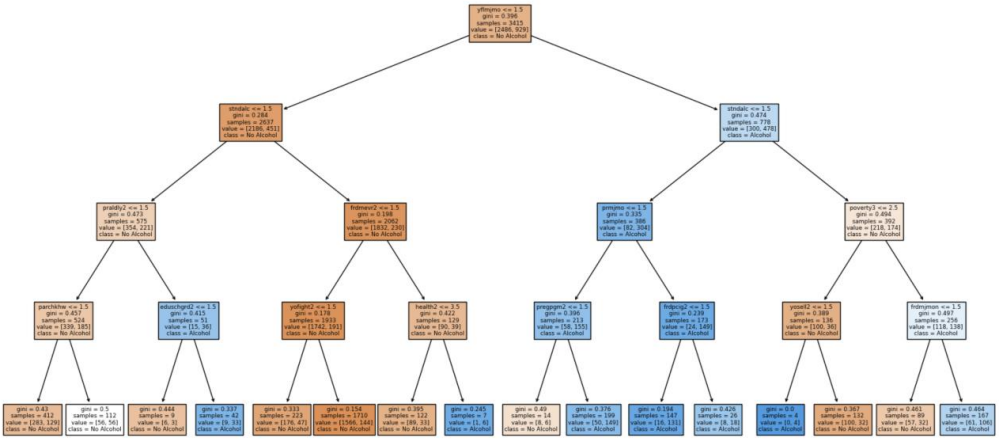
Binary Classification: A binary classification model was built to predict whether a youth has or has not used a particular substance (e.g., cigarettes). Accuracy and confusion matrix were computed to evaluate the model's performance.

Multi-level Classification: A multi-class classification model was developed to differentiate between different levels of substance use (e.g., seldom, sometimes, and frequent marijuana use). Accuracy and confusion matrix were calculated to assess the model's performance.

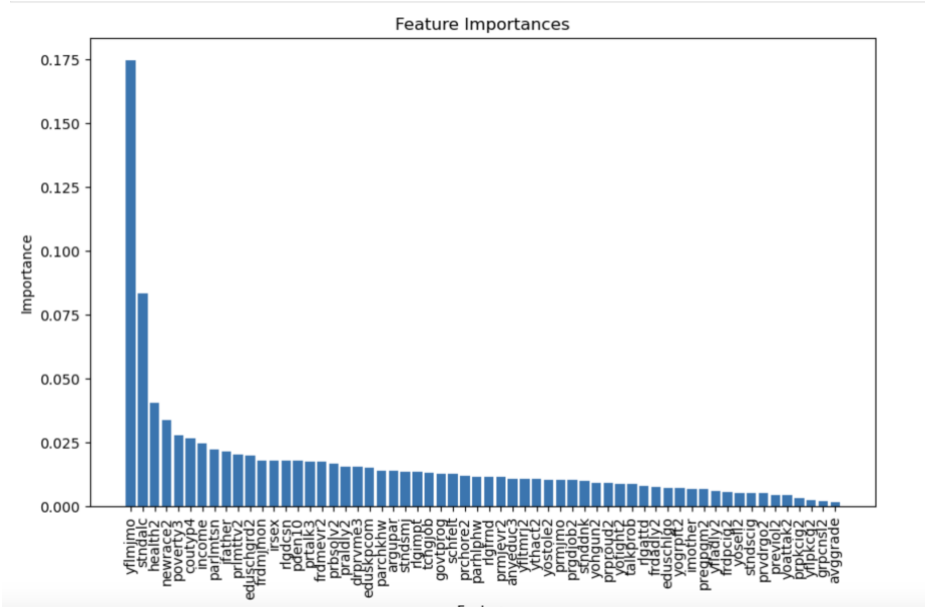
Regression: A regression model was trained to predict the number of days per year a person has used a particular substance (e.g., alcohol). Appropriate regression metrics were computed to evaluate the model's performance.

Computational Results

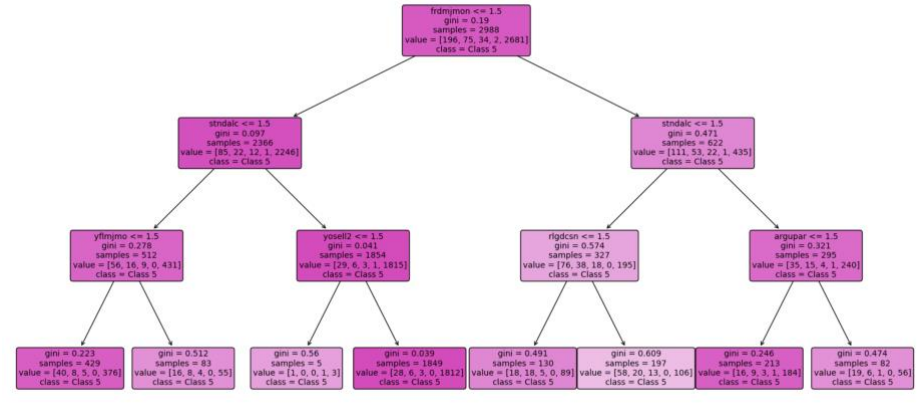
BELOW IS DECISION TREE FOR BINARY CLASSIFICATION



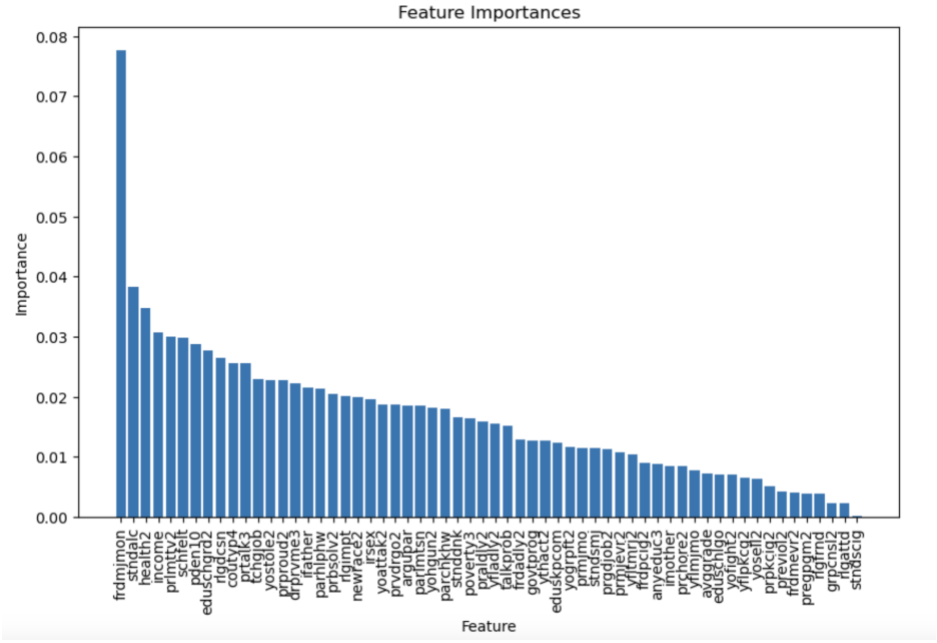
IMPORTANT FEATURES



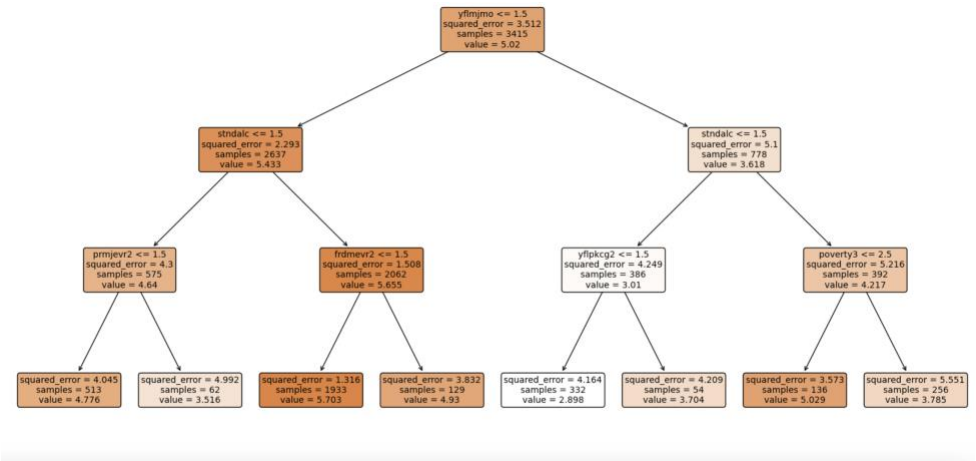
DECISION TREE FOR MULTI CLASSIFIER



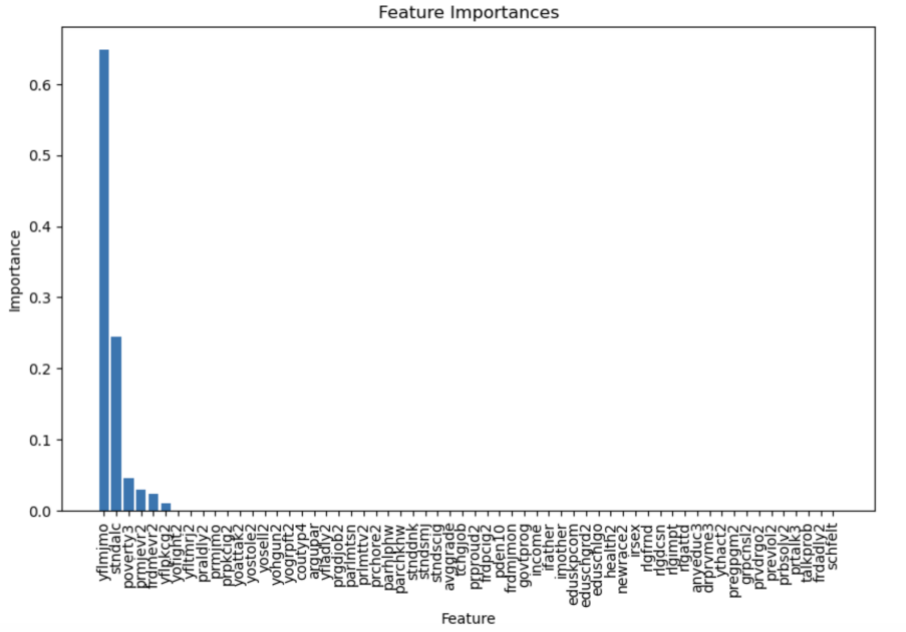
IMPORTANT FEATURES



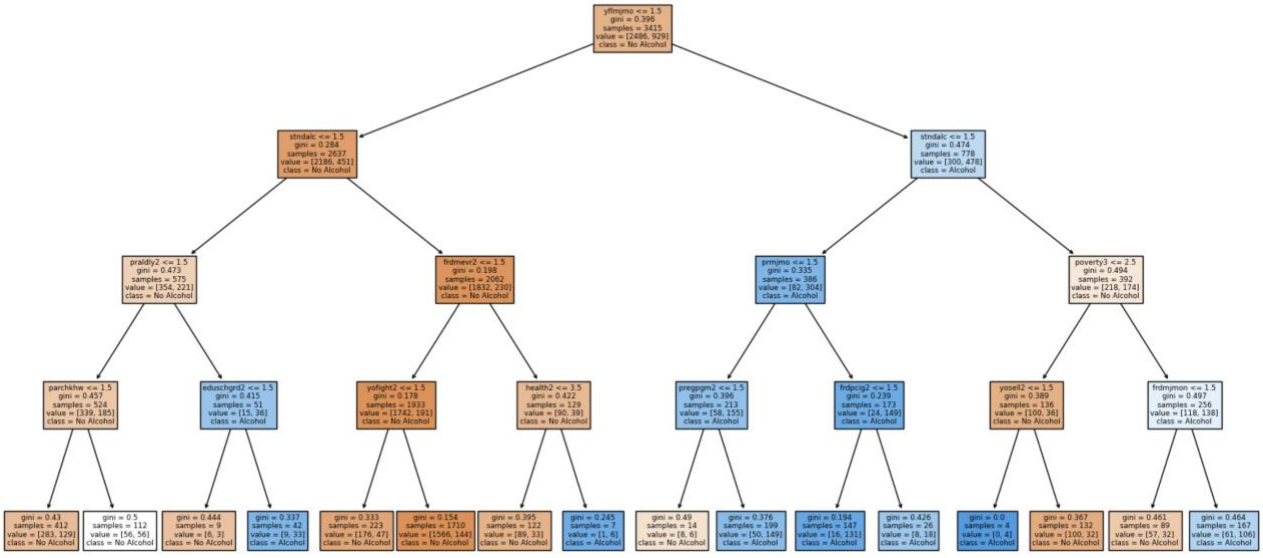
REGRESSION DECISION TREE



IMPORTANT FEATURES



Discussion



The decision tree starts with the class = No alcohol. If it is No , it moves to the left side and if it is Yes, it moves to the right branch. Let's consider right branch now and reach towards the leaf node. Here it checks for the value of attribute yfltmrj2 <= 1.5 and class alcohol, then it navigates to right or else left. Following this path, we reach the "class Alcohol" node. This indicates that the decision tree predicts someone with a high value for "yfitmrj2" (potentially indicating higher potential for drinking) is likely to consume alcohol.

The use of different data types, including binary, ordinal (categorical and ordered), and numerical variables, can have significant implications for predictions in youth drug use analysis. Binary variables, such as 'alcflag,' provide straightforward information about whether an individual engages in alcohol use, making them useful for binary classification tasks. Ordinal variables, like 'iralcfy' (alcohol usage frequency), offer insights into the frequency of alcohol use, allowing for multi-level classification. Numerical variables, such as age and income, provide continuous data that can be valuable for regression analysis, particularly in predicting the frequency of alcohol use.

In our analysis, variables related to alcohol usage frequency ('iralcfy'), educational background ('eduschgrd2'), and demographic factors like age and income tend to be significant predictors. Interpretation of these variables involves

recognizing their impact on drug use behaviors. For example, higher levels of alcohol usage frequency and lower levels of education may indicate increased risk of alcohol use.

Interpretations of computational results

For Binary classification :

- Decision tree accuracy: 70.49%
- Random Forest accuracy: 79.04%
- Bagging model accuracy: 78.22%
- Random Forest performed the best in terms of accuracy among the three models.
- Decision tree had the lowest accuracy score, indicating its limitations in capturing complex relationships in the data.
- Bagging model showed competitive performance, but slightly lower accuracy compared to Random Forest.

For Multi-level classification :

- Multi-level Decision Tree accuracy: 83.22%
- Random Forest accuracy: 91.80%
- Random Forest achieved the highest accuracy among the two models.
- Multi-level Decision Tree showed strong performance but had a lower accuracy compared to Random Forest.
- The higher accuracy of Random Forest suggests its ability to capture complex patterns in the data and make more accurate predictions.

Using Regression :

- Mean Squared Error (MSE) using linear regression: 2.61
- Decision Tree Regressor MSE: 2.74
- The linear regression model achieved a lower MSE compared to the Decision Tree Regressor.
- A lower MSE indicates that the linear regression model provided better predictions, with smaller errors on average.
- While the Decision Tree Regressor performed reasonably well, it had a slightly higher MSE, suggesting that it may have struggled to capture certain patterns in the data compared to linear regression.

Conclusion

Our comprehensive analysis of youth alcohol use behaviors using various machine learning techniques has yielded valuable insights into the factors influencing drug consumption patterns among young individuals. By leveraging decision trees, ensemble methods like Random Forests and bagging, as well as regression analysis, we have gained a deeper understanding of the complex dynamics surrounding youth drug use.

Through this analysis, the report aims to contribute to a better understanding of the factors influencing youth alcohol use and demonstrate the application of decision tree models in extracting insights from complex survey data. The findings can inform targeted interventions, educational programs, and policy decisions to address the issue of youth drug use effectively.

Bibliography/References

- https://www.samhsa.gov/data/sites/default/files/NSDUH-PairManual-2015.pdf
- https://www.cdc.gov/nchs/hus/sources-definitions/nsduh.htm

Appendix:

Importing all the necessary libraries and loading data

In [ ]:

```
import pandas as pd

import numpy as np

import pandas as pd

import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.metrics import confusion_matrix, accuracy_score

import matplotlib.pyplot as plt

from sklearn.tree import DecisionTreeClassifier, plot_tree

from sklearn.ensemble import BaggingClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy_score

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, confusion_matrix

from sklearn.linear_model import LogisticRegression

from sklearn.linear_model import Ridge, Lasso

from sklearn.ensemble import RandomForestRegressor

from sklearn.ensemble import BaggingRegressor
```

```
from sklearn.linear_model import Ridge, Lasso
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LinearRegression
```

In [ ]:

```
youth_data = pd.read_csv('/Users/nidhitrivedi/Downloads/youth_data.csv')
```

In [ ]:

```
#to view sample data
youth_data.head()
```

In [ ]:

```
youth_data.shape
```

In [ ]:

```
youth_data.info()
```

First, let's check the data for any missing values. Then, we'll get remove those missing values because they won't help us analyze the data.

In [ ]:

```
null_values = youth_data.isnull().sum()
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
null_values
```

In [ ]:

```
youth_data = youth_data.dropna()
```

In [ ]:

```
null_values = youth_data.isnull().sum()
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)


null_values
```

As our key variable for binary classification is 'alcflag' it's essential to examine its value distribution to check the number of entries classified as 1 and 0.

In [ ]:

```
youth_data['alcflag'].value_counts()
```

In [ ]:

```
(distinct, counts) = np.unique(youth_data['alcflag'], return_counts=True)
print(distinct, counts)
```

```
sns.barplot(x=distinct, y=counts)

plt.xlabel("alcflag")

plt.ylabel("count")

plt.xticks()

plt.title("Target variable count in dataset")

plt.show()
```

Below code is just to verify if there are any nulls in alcflag.

In [ ]:

```
null_count = youth_data['alcflag'].isnull().sum()

print("Number of null values in 'alcflag' column:", null_count)
```

We'll examine each feature to identify the unique values it contains and categorize them based on our needs.

In [ ]:

```
for column_name in youth_data.columns:

    print(f"Column: {column_name}")

    print(youth_data[column_name].unique())
```

To maintain the consistency for ease of use, lets convert all the columns to lowercase

In [ ]:

```
youth_data.columns = youth_data.columns.str.lower()


print(youth_data.columns)
```

In [ ]:

```
#considering below columns to make changes for multi - level

columns_to_change = ['iralcfy', 'eduschgrd2', 'eduschlgo', 'eduskpcom']


for column_name in columns_to_change:

    print(f"Column: {column_name}")

    print(youth_data[column_name].unique())
```

IRALCFY provides information on alcohol usage frequency in the past year, including specific days of usage as well as indicators for never and no usage. Range from 1 to 365: Indicates the number of days alcohol was used in the past year. 991: Indicates "NEVER USED ALCOHOL." 993: Indicates "DID NOT USE ALCOHOL PAST YEAR."

Assigning 991 and 993 to 0 and futher dividing the range of days used into groups.

In [ ]:

```
youth_data['iralcfy'] = [0 if value in [991, 993] else (1 if value < 90 else (2 if
value < 180 else 3)) for value in youth_data['iralcfy']]

print(youth_data['iralcfy'].value_counts())
```

eduschgrd2 provides information about the What grade or year of school are you now attending? What grade or year of school will you be attending when your vacation is over? As 98 and 99 are legitimate skip we are assigning them to 0, considering 1,2,3 will not drink as they are below 7th grade - assigning them 0 , making seperate class for 4-8 as 1

In [ ]:

```
youth_data['eduschgrd2'] = [0 if value in [99, 98] else (1 if value < 5 else (2 if
value < 9 else 3)) for value in youth_data['eduschgrd2']]

print(youth_data['eduschgrd2'].value_counts())
```

eduschlgo provides information if respondents originally reported that they were not enrolled in school. Assigned to 1 if they are now going to school otherwise 2. EDUSCHLGO was assigned a code of 11 if they are enrolled but there is uncertainty about their enrollment status. Converting 85, 94, 97, 98 to 0 as that is bad data and 11 to 1.

In [ ]:

```
youth_data['eduschlgo'] = [0 if value in [85, 94, 97, 98] else (1 if value == 11 else
value) for value in youth_data['eduschlgo']]

print(youth_data['eduschlgo'].value_counts())
```

eduskpcom provides information about how many days the respondant missed schol from skipping 94, 97, 98, 99 is assigned to 0 as it is bad data.

In [ ]:

```
youth_data['eduskpcom'] = [0 if value in [94, 97, 98, 99] else (1 if value <= 7 else (2
if value <= 14 else (3 if value <= 21 else 4))) for value in youth_data['eduskpcom']]

print(youth_data['eduskpcom'].value_counts())
```

Creating a new DataFrame containing only the necessary variables for binary classification, including 'iralcfy' and all demographic and youth-related variables.

In [ ]:

```
required_columns = ['alcflag', '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']


youth_data_selected = youth_data[required_columns]


youth_data_selected.head(5)
```

Lets split the data into train and test.

In [ ]:

```
X = youth_data_selected.drop('alcflag', axis=1)

y = youth_data_selected['alcflag']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

tree_model = DecisionTreeClassifier()

tree_model.fit(X_train, y_train)
```

In [ ]:

```
ypredict = tree_model.predict(X_test)

test_accuracy = accuracy_score(y_test, ypredict)

print("Accuracy:", test_accuracy)
```

In [ ]:

```
conf_matrix = confusion_matrix(y_test, ypredict)

print("Confusion Matrix:")

print(conf_matrix)
```

In [ ]:

```
plt.figure(figsize=(20, 10))

plot_tree(tree_model, filled=True, feature_names=X.columns.tolist(), class_names=['No
Alcohol', 'Alcohol'])

plt.show()
```

In [ ]:

```
pruned_tree = DecisionTreeClassifier(max_depth=4)

pruned_tree.fit(X_train, y_train)

plt.figure(figsize=(20, 10))

plot_tree(pruned_tree, filled=True, feature_names=X.columns.tolist(), class_names=['No
Alcohol', 'Alcohol'])

plt.show()
```

In [ ]:

```
importances = tree_model.feature_importances_

feature_names = X.columns

indices = np.argsort(importances)[::-1]

plt.figure(figsize=(10, 6))

plt.title("Feature Importances")

plt.bar(range(X.shape[1]), importances[indices], align="center")
```

```
plt.xticks(range(X.shape[1]), feature_names[indices], rotation=90)

plt.xlabel("Feature")

plt.ylabel("Importance")

plt.show()
```

We will do it now using bagging

In [ ]:

```
tree_model = DecisionTreeClassifier()

bagging_classifier = BaggingClassifier(tree_model, n_estimators=10, random_state=42)

bagging_classifier.fit(X_train, y_train)
```

In [ ]:

```
bagging_predict = bagging_classifier.predict(X_test)
```

In [ ]:

```
accuracy_bagging = accuracy_score(y_test, bagging_predict)

print('Accuracy with bagging model:', accuracy_bagging)
```

In [ ]:

```
conf_matrix_bagging = confusion_matrix(y_test, bagging_predict)

print('Confusion Matrix:')

print(conf_matrix_bagging)
```

In [ ]:

```
tree_model = DecisionTreeClassifier()

tree_model.fit(X, y)


plt.figure(figsize=(20, 10))

plot_tree(tree_model, filled=True, feature_names=X.columns.tolist(), class_names=['No
Alcohol', 'Alcohol'])

plt.show()
```

In [ ]:

```
pruned_tree = DecisionTreeClassifier(max_depth=3)

classifier_pruned = BaggingClassifier(pruned_tree, n_estimators=10, random_state=42)

classifier_pruned.fit(X_train, y_train)

pruned_tree = classifier_pruned.estimators_[0]

plt.figure(figsize=(20, 10))

plot_tree(pruned_tree, filled=True, feature_names=list(X.columns), class_names=['No
Alcohol', 'Alcohol'])

plt.show()
```

In [ ]:

```
importances = tree_model.feature_importances_

feature_names = X.columns

indices = np.argsort(importances)[::-1]

plt.figure(figsize=(10, 6))

plt.title("Feature Importances")

plt.bar(range(X.shape[1]), importances[indices], align="center")

plt.xticks(range(X.shape[1]), feature_names[indices], rotation=90)

plt.xlabel("Feature")

plt.ylabel("Importance")

plt.show()
```

Lets continue the same using random forest

In [ ]:

```
random_forest_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

random_forest_classifier.fit(X_train, y_train)

y_pred_random_forest = random_forest_classifier.predict(X_test)
```

In [ ]:

```
accuracy_random_forest = accuracy_score(y_test, y_pred_random_forest)

print('Accuracy (Random Forest):', accuracy_random_forest)
```

In [ ]:

```
conf_matrix_random_forest = confusion_matrix(y_test, y_pred_random_forest)

print('Confusion Matrix (Random Forest):')

print(conf_matrix_random_forest)
```

In [ ]:

```
random_forest_pruned = RandomForestClassifier(n_estimators=100, max_depth=3,
random_state=42)

random_forest_pruned.fit(X_train, y_train)

plt.figure(figsize=(20, 10))

plot_tree(random_forest_pruned.estimators_[0], filled=True,
feature_names=list(X.columns), class_names=['No Alcohol', 'Alcohol'])

plt.show()
```

In [ ]:

```
feature_importances = random_forest_classifier.feature_importances_

feature_names = X.columns

indices = np.argsort(feature_importances)[::-1]
```

```
plt.figure(figsize=(10, 6))

plt.title("Feature Importance")

plt.bar(range(X.shape[1]), feature_importances[indices], align="center")

plt.xticks(range(X.shape[1]), [feature_names[i] for i in indices], rotation=90)

plt.xlim([-1, X.shape[1]])

plt.tight_layout()

plt.show()
```

Multiclass Classifier

For multiclass we will be using ALCMDAYS

RC-# OF DAYS USED ALCOHOL IN PAST MONTH</br> Freq Pct</br> 1 = 1-2 Days (IRALCFM=1-2)</br> 2 = 3-5 Days (IRALCFM=3-5)</br> 3 = 6-19 Days (IRALCFM=6-19)</br> 4 = 20-30 Days (IRALCFM=20-30)</br> 5 = Non User or No Past Month Use (IRALCFM=91,93)</br>

Creating a new dataframe with required features.

In [ ]:

```
required_columns = ['alcmdays', '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']


dataframe_multilevel = youth_data[required_columns]


dataframe_multilevel.head(5)
```

In [ ]:

```
X = dataframe_multilevel.drop('alcmdays', axis=1)

y = dataframe_multilevel['alcmdays']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)
```

In [ ]:

```
multi_classifier_tree = DecisionTreeClassifier()

multi_classifier_tree.fit(X_train, y_train)


multi_predictions = multi_classifier_tree.predict(X_test)


multi_accuracy = accuracy_score(y_test, multi_predictions)

print("Multi-level Decision Tree Accuracy:", multi_accuracy)
```

In [ ]:

```
plt.figure(figsize=(20, 10))

plot_tree(multi_classifier_tree, filled=True, feature_names=X.columns.tolist(),
class_names=['Class 1', 'Class 2', 'Class 3', 'Class 4', 'Class 5'])

plt.show()
```

In [ ]:

```
multi_classifier_pruned = DecisionTreeClassifier(max_depth=3)

multi_classifier_pruned.fit(X_train, y_train)


plt.figure(figsize=(20, 10))

plot_tree(multi_classifier_pruned, filled=True, feature_names=list(X.columns),
class_names=['Class 1', 'Class 2', 'Class 3', 'Class 4', 'Class 5'], rounded=True,
fontsize=10)

plt.show()
```

In [ ]:

```
importances = multi_classifier_tree.feature_importances_

feature_names = X.columns

indices = np.argsort(importances)[::-1]

plt.figure(figsize=(10, 6))

plt.title("Feature Importances")

plt.bar(range(X.shape[1]), importances[indices], align="center")

plt.xticks(range(X.shape[1]), feature_names[indices], rotation=90)

plt.xlabel("Feature")

plt.ylabel("Importance")

plt.show()
```

In [ ]:

```
X = dataframe_multilevel.drop('alcmdays', axis=1)

y = dataframe_multilevel['alcmdays']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)


random_forest_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

random_forest_classifier.fit(X_train, y_train)

y_pred_random_forest = random_forest_classifier.predict(X_test)
```

In [ ]:

```
accuracy_random_forest = accuracy_score(y_test, y_pred_random_forest)

print('Accuracy (Random Forest):', accuracy_random_forest)
```

In [ ]:

```
conf_matrix_random_forest = confusion_matrix(y_test, y_pred_random_forest)

print('Confusion Matrix (Random Forest):')

print(conf_matrix_random_forest)
```

In [ ]:

```
random_forest_pruned = RandomForestClassifier(n_estimators=100, max_depth=3,
random_state=42)

random_forest_pruned.fit(X_train, y_train)

plt.figure(figsize=(20, 10))

plot_tree(random_forest_pruned.estimators_[0], filled=True,
feature_names=list(X.columns), class_names=['Class 1', 'Class 2', 'Class 3', 'Class 4',
'Class 5'])

plt.show()
```

In [ ]:

```
feature_importances = random_forest_classifier.feature_importances_

feature_names = X.columns

indices = np.argsort(feature_importances)[::-1]

plt.figure(figsize=(10, 6))

plt.title("Feature Importance")

plt.bar(range(X.shape[1]), feature_importances[indices], align="center")

plt.xticks(range(X.shape[1]), [feature_names[i] for i in indices], rotation=90)

plt.xlim([-1, X.shape[1]])

plt.tight_layout()

plt.show()
```

Lets go ahead with regression models now, here we are selecting alcydays (Number of days a person has used alcohol in the past year) for accuracy

In [ ]:

```
required_columns = ['alcydays', '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']


dataframe_regression = youth_data[required_columns]


dataframe_regression.head(5)
```

In [ ]:

```
X = dataframe_regression.drop(columns=['alcydays'])
y = dataframe_regression['alcydays']


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

In [ ]:

```
dt_regressor = DecisionTreeRegressor(max_depth = 3, random_state=42)
dt_regressor.fit(X_train, y_train)


y_predict = linear_regression_model.predict(X_test)


mse = mean_squared_error(y_test, y_predict)
print("Mean Squared Error:", mse)
```

In [ ]:

```
plt.figure(figsize=(20, 10))
plot_tree(dt_regressor, filled=True, feature_names=list(X.columns), rounded=True,
fontsize=10)
plt.show()
```

In [ ]:

```
importances = dt_regressor.feature_importances_

feature_names = X.columns

indices = np.argsort(importances)[::-1]

plt.figure(figsize=(10, 6))

plt.title("Feature Importances")

plt.bar(range(X.shape[1]), importances[indices], align="center")

plt.xticks(range(X.shape[1]), feature_names[indices], rotation=90)

plt.xlabel("Feature")

plt.ylabel("Importance")

plt.show()
```

In [ ]:

```
X = dataframe_regression.drop(columns=['alcydays'])

y = dataframe_regression['alcydays']


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)


rf_regressor = RandomForestRegressor(n_estimators=100, random_state=42)

rf_regressor.fit(X_train, y_train)


y_pred = linear_regression_model.predict(X_test)
```

In [ ]:

```
rf_predictions = rf_regressor.predict(X_test)

dt_mse = mean_squared_error(y_test, rf_predictions)

print(f"Decision Tree Regressor MSE: {dt_mse:.2f}")
```

In [ ]:

```
plt.figure(figsize=(20, 10))

plot_tree(dt_regressor, filled=True, feature_names=list(X.columns), rounded=True,
fontsize=10)

plt.show()
```

In [ ]:

```
importances = rf_regressor.feature_importances_

feature_names = X.columns

indices = np.argsort(importances)[::-1]

plt.figure(figsize=(10, 6))
```

```
plt.title("Feature Importances")

plt.bar(range(X.shape[1]), importances[indices], align="center")

plt.xticks(range(X.shape[1]), feature_names[indices], rotation=90)

plt.xlabel("Feature")

plt.ylabel("Importance")

plt.show()
```

In [ ]: