CSE422 - Artificial Intelligence

Project Report

on

Estimation on Obesity levels

Prepared By

Name: Mustafina Joarder
ID: 22101215

Name: Ahmed Riasat
ID: 24141253

Group 7

Section - 13

Table of Contents

Introduction:

The aim of our project is to develop a machine learning model that can predict obesity levels. There are many aspects that affect a person's obesity level, such ase - age, weight, gender, height, family history of obesity, eating habits, smoking habits.

Nowadays, life has become more digital, people are more connected to devices. The normal mobility that a person needs to stay fit has reduced significantly. People are less concerned about their fitness. As a result, obesity has become a major health challenge linked to numerous chronic diseases. Through this project we are trying to identify individuals at risk of obesity based on their lifestyle and other parameters so it can help in early intervention reducing a long term loss.

Our motivation behind this project is the growing rate of obesity which highlights the urgent need of prevention. By using machine learning to analyze lifestyle data, we want to provide a data driven approach to promote healthier living and encourage individuals to make decisions about their health.

Dataset Description:

1. Source:

- **Link:**https://archive.ics.uci.edu/dataset/544/estimation+of+obesity+levels+based+on+eating+habits+and+physical+condition
- **Reference:**Estimation of Obesity Levels Based On Eating Habits and Physical Condition [Dataset]. (2019). UCI Machine Learning Repository. https://doi.org/10.24432/C5H31Z.

2. Dataset Description:

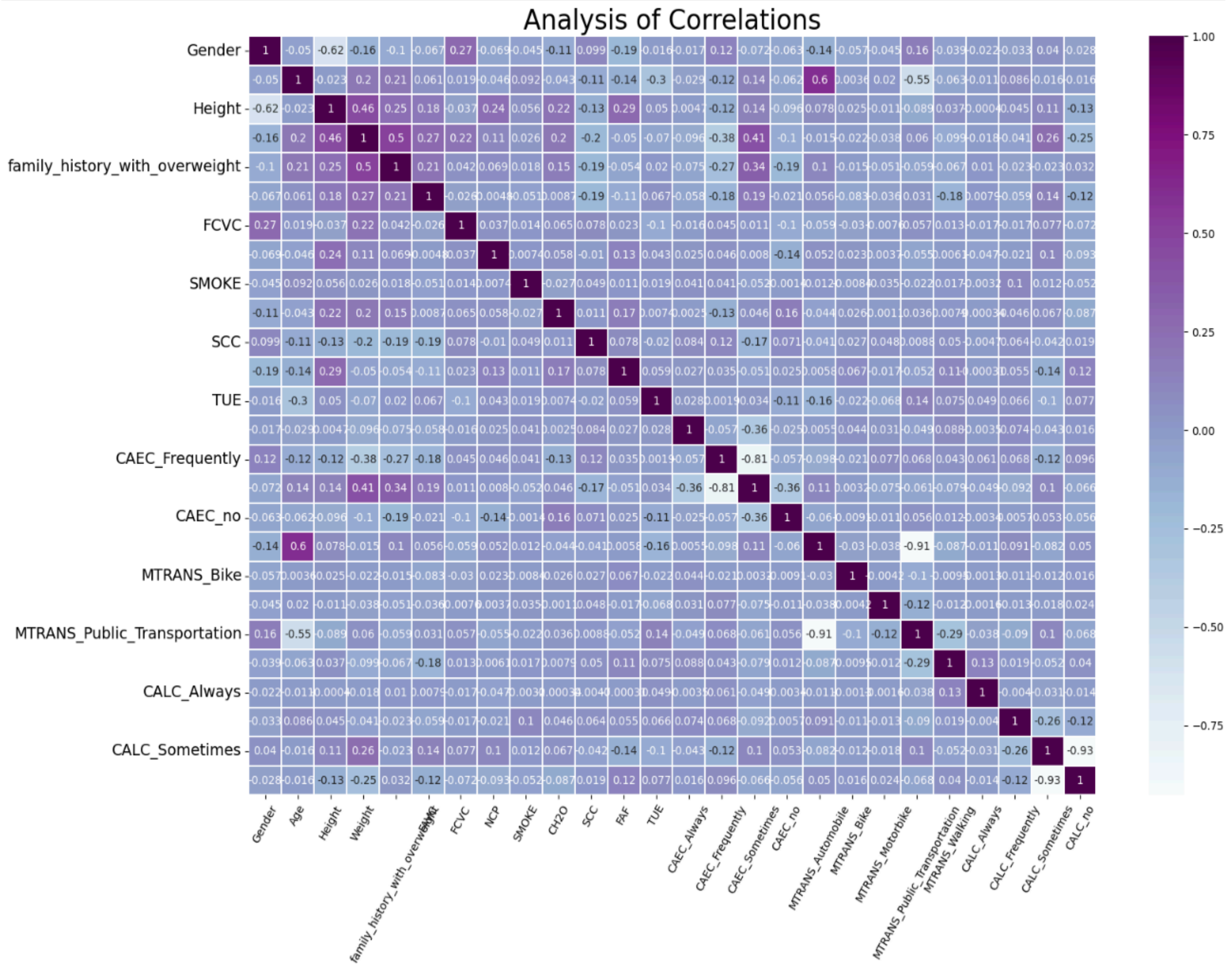
- **Features:** There are a total of 16 features in this dataset.
- **Classification or regression problem:** This dataset has classification, clustering and regression problems in it.

  This dataset includes obesity levels as categorical variables such as - underweight, normal weight, overweight and obese. These variables can be classified as they are discrete and mutually exclusive.

This dataset has features like - age, weight, height etc. By using these, BMI can be calculated which is a continuous variable. Predicting a continuous variable is a regression problem.

This dataset has a variety of features that can be used to group individuals. Clustering is useful when you want to define a specific pattern which is not present in the dataset. For example- people with similar eating habits can be grouped together.

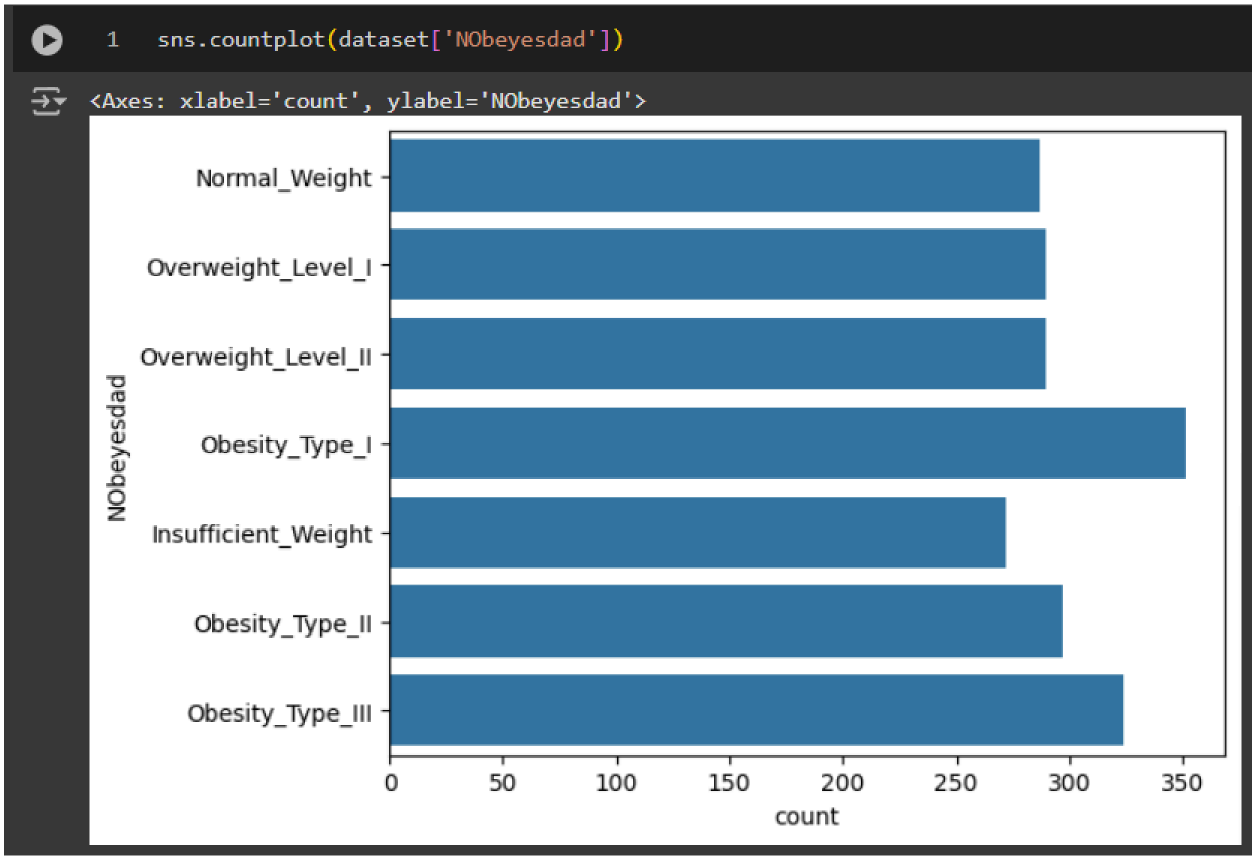
- **Data Points:** This dataset has 2112 data points (row).
- **Kind of features:** This dataset has 4 types of features - categorical, continuous, binary, integer.
- **Correlation of all features:**



The heat map of the correlation between the features

3. Imbalance Dataset:

This dataset is imbalanced as all the features do not have the same number of values. For the feature which gives the decision that a person is obese or not, we can see this from a chart.

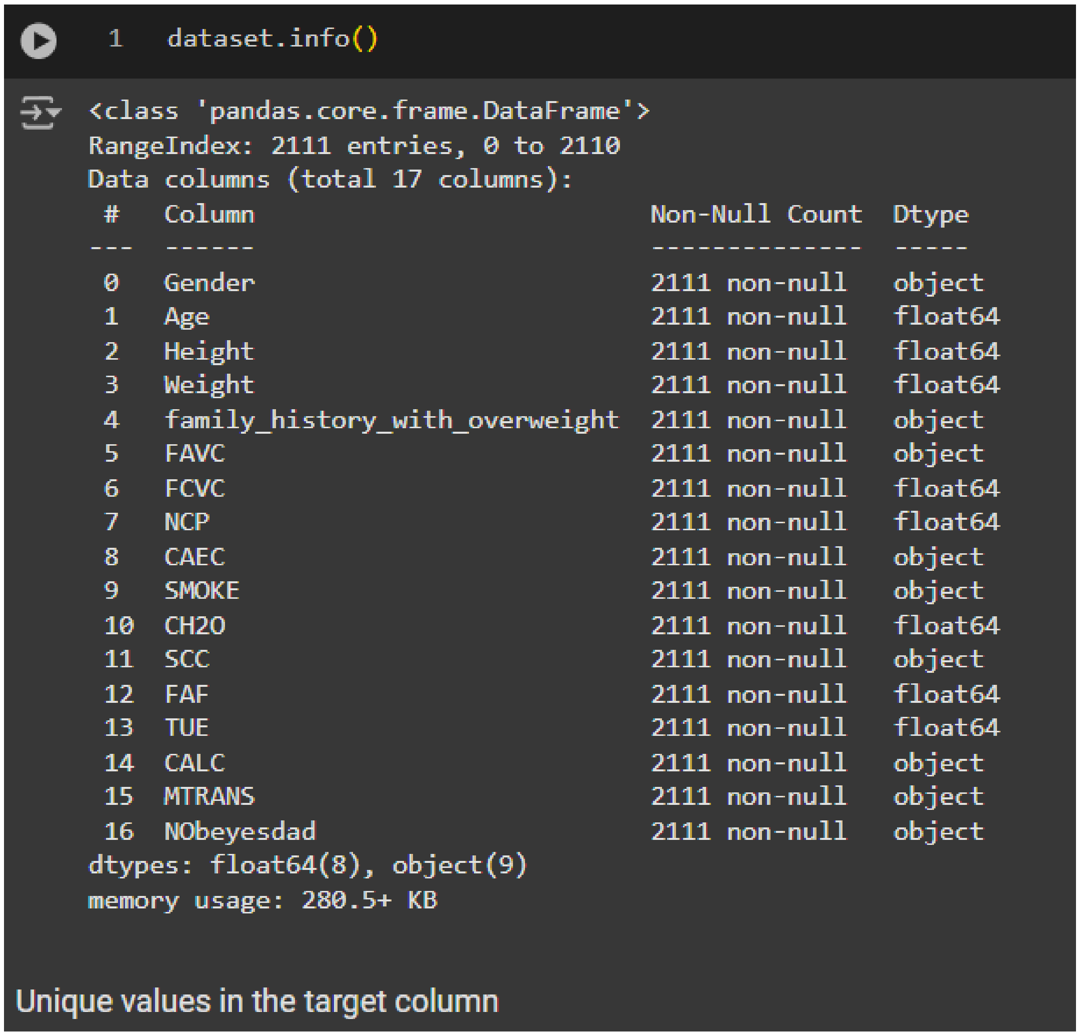


```
sns.countplot(dataset['NObeyesdad'])
```

```
<Axes: xlabel='count', ylabel='NObeyesdad'>
```

Dataset pre processing:

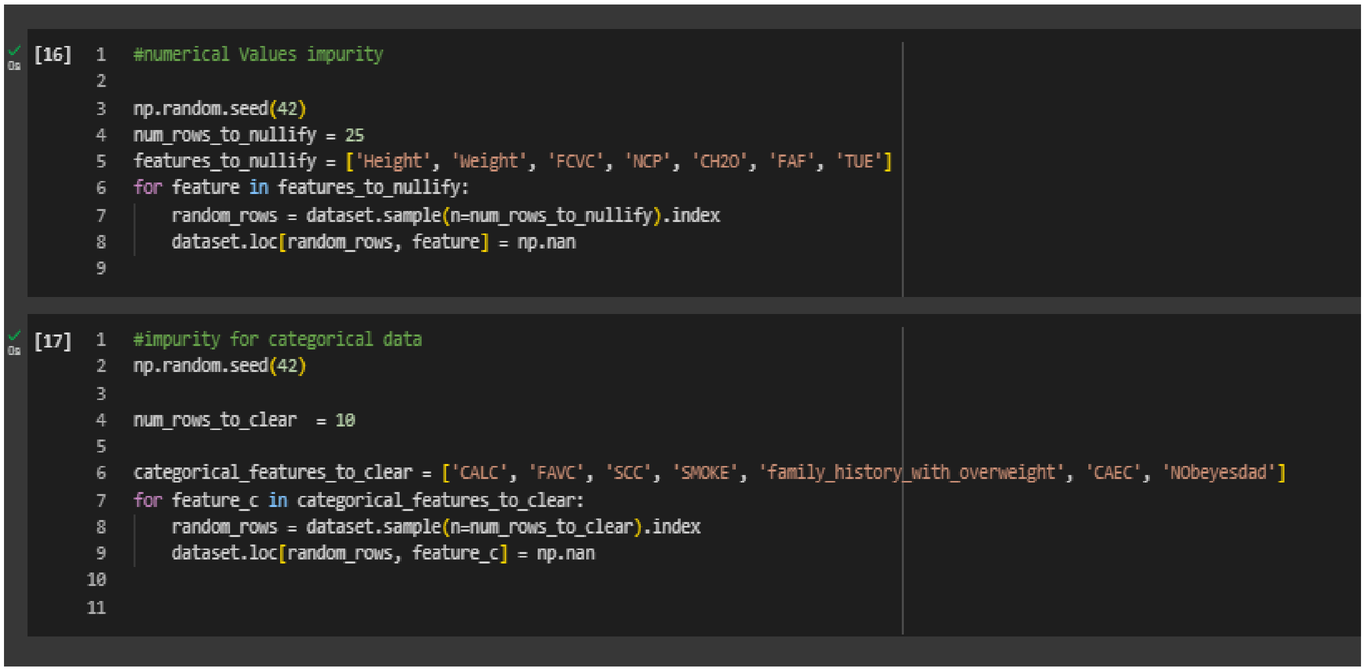
- **Null Values:** Our chosen dataset had no null values, therefore we had to delete some values manually. For features having numerical values, we deleted 25 data per column. And for features having categorical values, we deleted 10 data per column.



```
dataset.info()
```

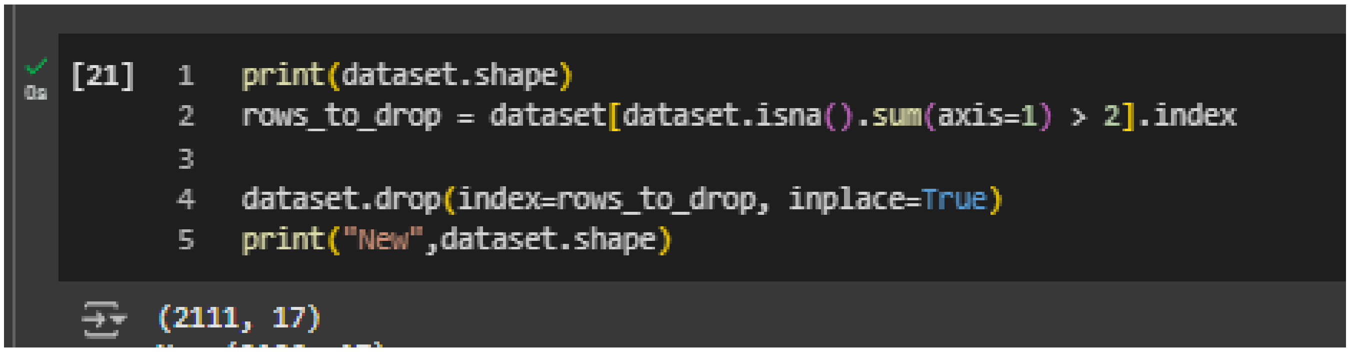
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2111 entries, 0 to 2110
Data columns (total 17 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   Gender                          2111 non-null   object 
 1   Age                             2111 non-null   float64
 2   Height                          2111 non-null   float64
 3   Weight                          2111 non-null   float64
 4   family_history_with_overweight  2111 non-null   object 
 5   FAVC                            2111 non-null   object 
 6   FCVC                            2111 non-null   float64
 7   NCP                             2111 non-null   float64
 8   CAEC                            2111 non-null   object 
 9   SMOKE                           2111 non-null   object 
 10  CH2O                            2111 non-null   float64
 11  SCC                             2111 non-null   object 
 12  FAF                             2111 non-null   float64
 13  TUE                             2111 non-null   float64
 14  CALC                            2111 non-null   object 
 15  MTRANS                          2111 non-null   object 
 16  NObeyesdad                      2111 non-null   object 
dtypes: float64(8), object(9)
memory usage: 280.5+ KB
```

Unique values in the target column


```
[16]
#numerical Values impurity

np.random.seed(42)
num_rows_to_nullify = 25
features_to_nullify = ['Height', 'Weight', 'FCVC', 'NCP', 'CH2O', 'FAF', 'TUE']
for feature in features_to_nullify:
    random_rows = dataset.sample(n=num_rows_to_nullify).index
    dataset.loc[random_rows, feature] = np.nan
```

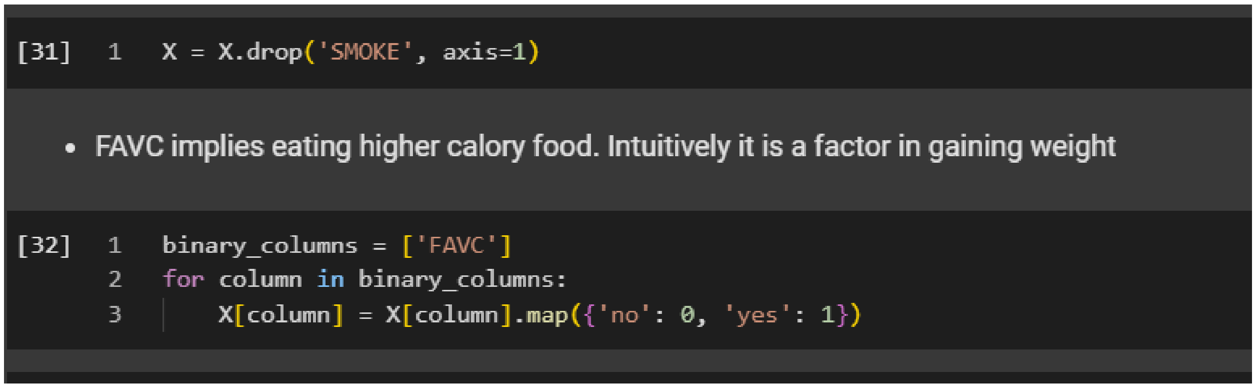
```
[17]
#impurity for categorical data
np.random.seed(42)

num_rows_to_clear  = 10

categorical_features_to_clear = ['CALC', 'FAVC', 'SCC', 'SMOKE', 'family_history_with_overweight', 'CAEC', 'NObeyesdad']
for feature_c in categorical_features_to_clear:
    random_rows = dataset.sample(n=num_rows_to_clear).index
    dataset.loc[random_rows, feature_c] = np.nan
```

After doing this, we had some “null” values in our dataset. If any specific row had more than 2 null values, we dropped that row from the dataset.


```
[21]
print(dataset.shape)
rows_to_drop = dataset[dataset.isna().sum(axis=1) > 2].index

dataset.drop(index=rows_to_drop, inplace=True)
print("New",dataset.shape)
```

```
(2111, 17)
```

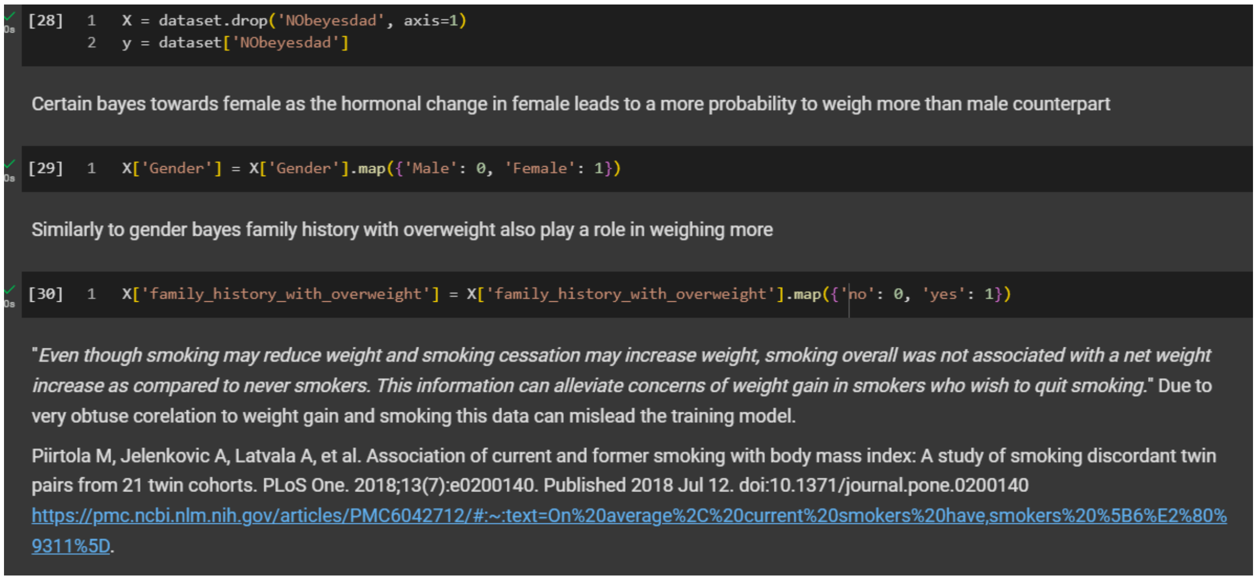
- **Categorical Values:** The dataset has categorical features - ‘CALC’, ‘FAVC’,’SCC’, ‘Smoke’, ‘family history with overweight’, ‘CAEC’, ‘NObetesdad’. We labeled these data as 0 & 1.



```
[31] X = X.drop('SMOKE', axis=1)
```

- FAVC implies eating higher calory food. Intuitively it is a factor in gaining weight

```
[32] binary_columns = ['FAVC']
     for column in binary_columns:
         X[column] = X[column].map({'no': 0, 'yes': 1})
```



```
[28] X = dataset.drop('NObeyesdad', axis=1)
     y = dataset['NObeyesdad']
```

Certain bayes towards female as the hormonal change in female leads to a more probability to weigh more than male counterpart

```
[29] X['Gender'] = X['Gender'].map({'Male': 0, 'Female': 1})
```

Similarly to gender bayes family history with overweight also play a role in weighing more

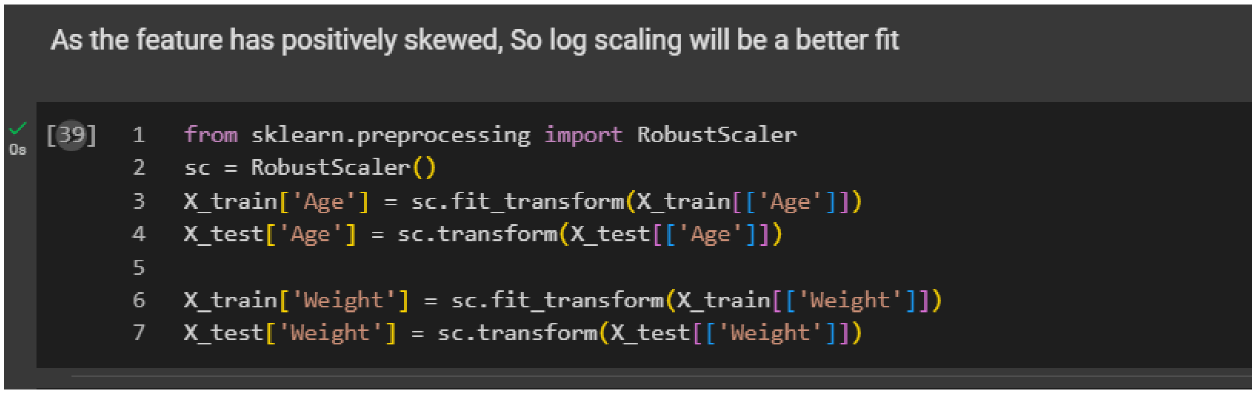
```
[30] X['family_history_with_overweight'] = X['family_history_with_overweight'].map({'no': 0, 'yes': 1})
```

"*Even though smoking may reduce weight and smoking cessation may increase weight, smoking overall was not associated with a net weight increase as compared to never smokers. This information can alleviate concerns of weight gain in smokers who wish to quit smoking.*" Due to very obtuse corelation to weight gain and smoking this data can mislead the training model.

Piirtola M, Jelenkovic A, Latvala A, et al. Association of current and former smoking with body mass index: A study of smoking discordant twin pairs from 21 twin cohorts. PLoS One. 2018;13(7):e0200140. Published 2018 Jul 12. doi:10.1371/journal.pone.0200140 https://pmc.ncbi.nlm.nih.gov/articles/PMC6042712/#:~:text=On%20average%2C%20current%20smokers%20have,smokers%20%5B6%E2%80%9311%5D.

Feature Scaling:

The features of our dataset has positively skewed. So, log scaling will be a better fit.

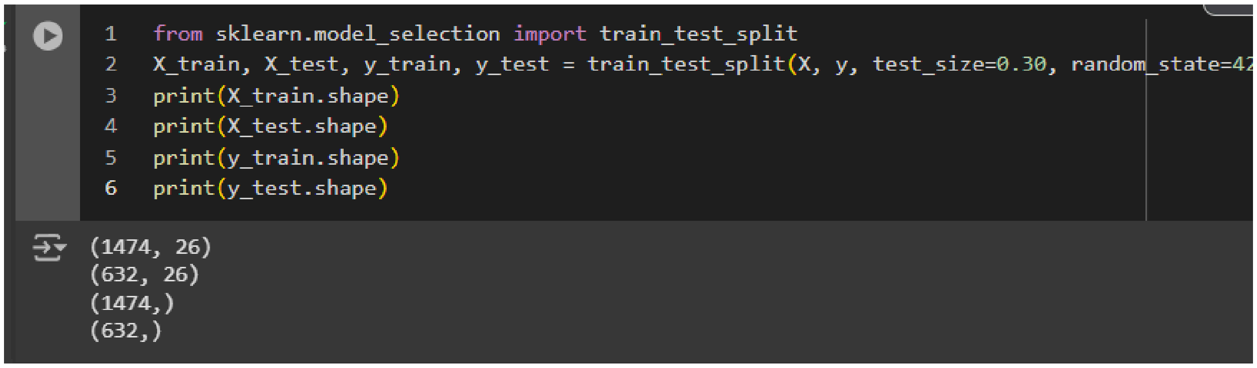


As the feature has positively skewed, So log scaling will be a better fit

```
from sklearn.preprocessing import RobustScaler
sc = RobustScaler()
X_train['Age'] = sc.fit_transform(X_train[['Age']])
X_test['Age'] = sc.transform(X_test[['Age']])

X_train['Weight'] = sc.fit_transform(X_train[['Weight']])
X_test['Weight'] = sc.transform(X_test[['Weight']])
```

Dataset Splitting:

For train-test splitting, we split the features using test_size = 0.3 meaning 70% data were used for training and 30% for testing. As a result, we have enough data trained to accurately predict the testing samples



```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=42
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(1474, 26)
(632, 26)
(1474,)
(632,)
```

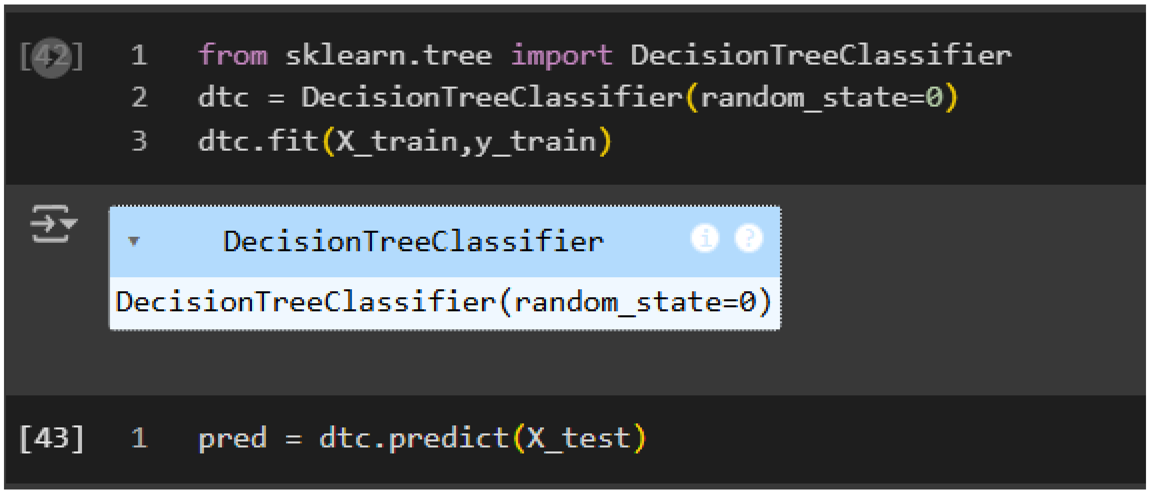
Model Training and testing:

We have applied

- Decision Tree
- Logistic Regression
- KNeighbors
- Random Forest
- Naive Bayes

Decision Tree:

It is a tree structured algorithm where it splits data into subsets based on features creating a flowchart-like structure. Each node represents a feature and each branch a decision and each leaf an output. It optimizes splits using impurity measures like ‘Gini’(by default) or ‘Entropy’.



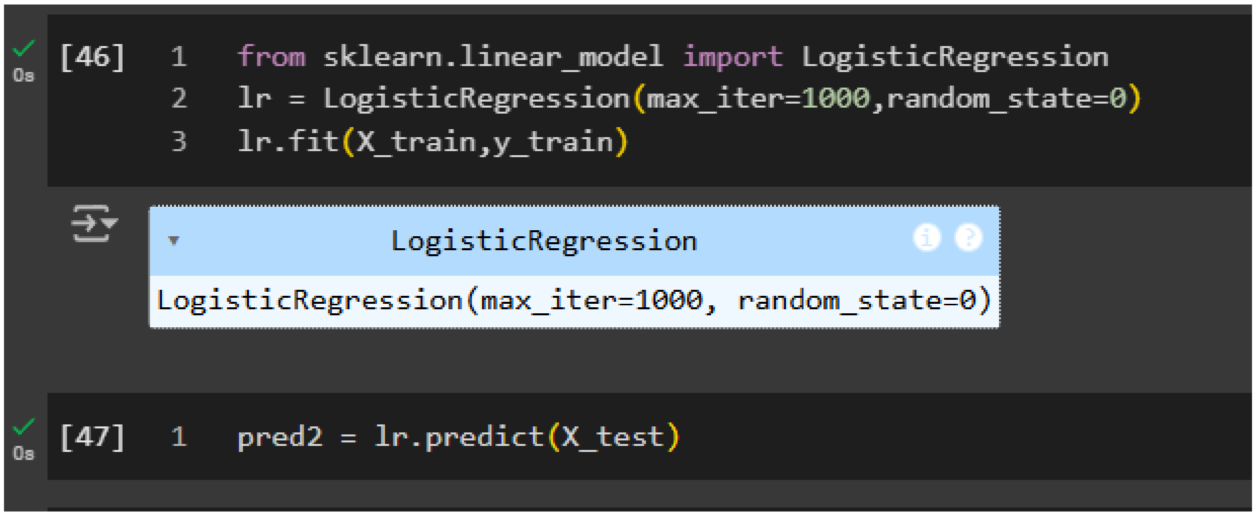
```
[42]
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier(random_state=0)
dtc.fit(X_train,y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(random_state=0)
```

```
[43]
pred = dtc.predict(X_test)
```

Logistic Regression:

It is a statistical ML algorithm used for mainly binary classification but can be extended to multinomial or ordinal regression for multi-class tasks. It uses the sigmoid logistic function to map predicted values and finds a linear relationship between input features and the target label.



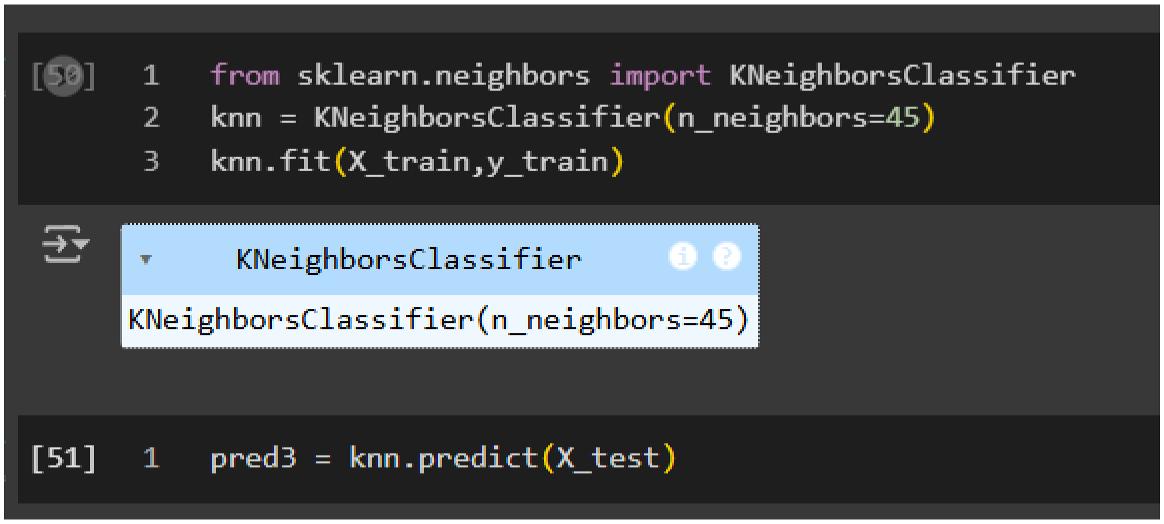
```
[46]
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(max_iter=1000,random_state=0)
lr.fit(X_train,y_train)
```

```
LogisticRegression
LogisticRegression(max_iter=1000, random_state=0)
```

```
[47]
pred2 = lr.predict(X_test)
```

KNeighbors:

It is a simple non-parametric ML algorithm for classification and regression tasks. It uses instance based prediction by storing the training data and predicting the testing data by finding nearest neighbors(K amount). For classification problems, it assigns the class most common among the neighbors.



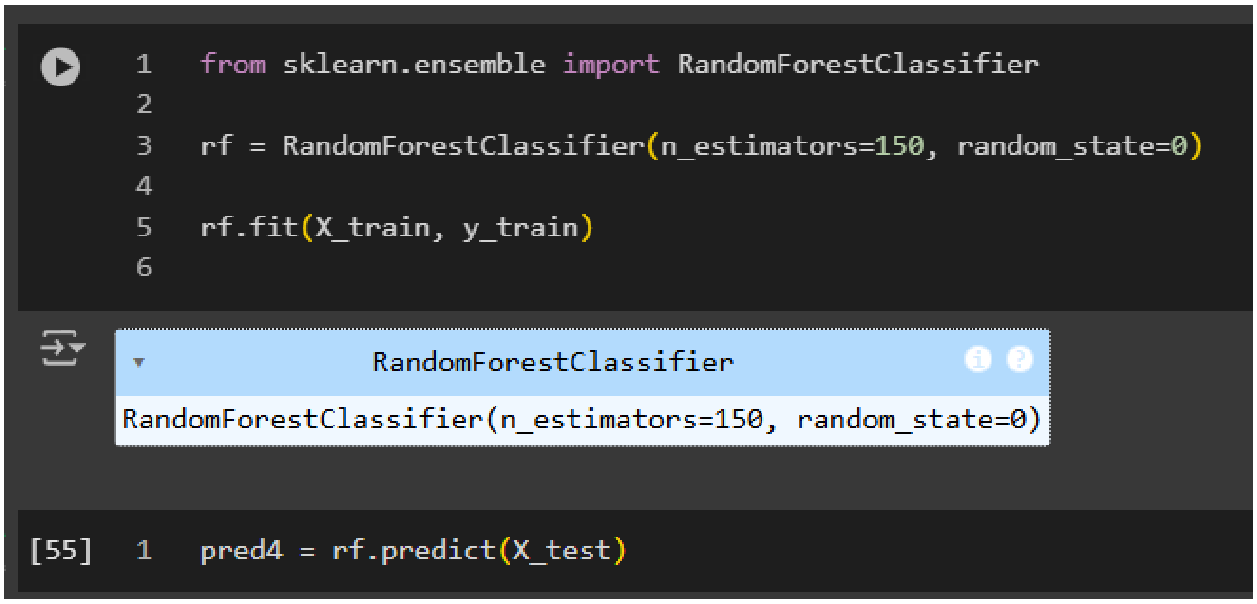
```
[50]
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=45)
knn.fit(X_train,y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=45)
```

```
[51]
pred3 = knn.predict(X_test)
```

Random Forest:

It is a machine learning algorithm that combines multiple decision trees to improve classification. It works by aggregating predictions from individual trees to enhance accuracy and reduce overfitting.



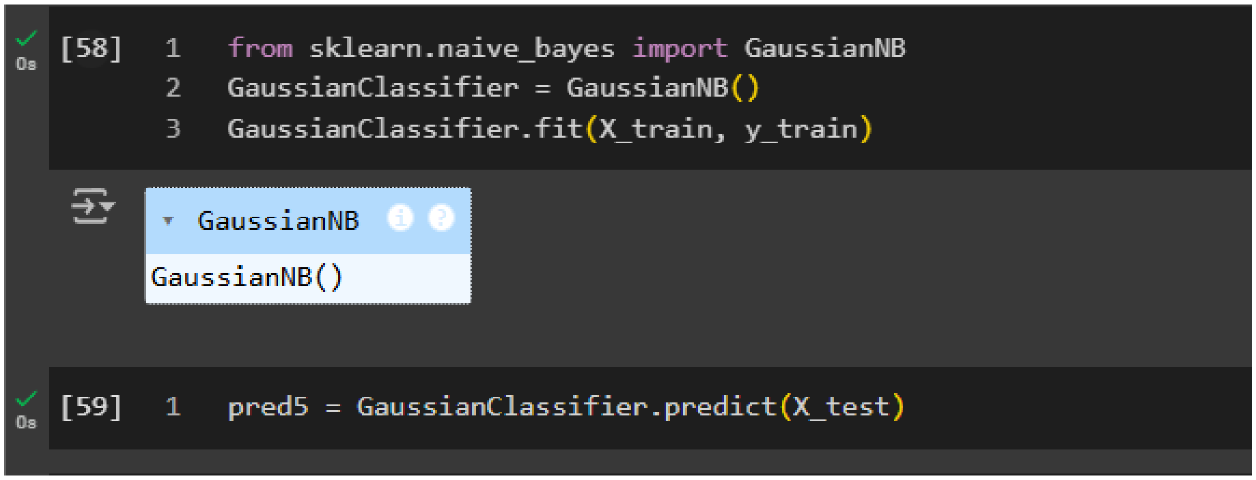
```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(n_estimators=150, random_state=0)

rf.fit(X_train, y_train)

```

```
RandomForestClassifier
RandomForestClassifier(n_estimators=150, random_state=0)
```

```
[55]
pred4 = rf.predict(X_test)
```

Gaussian Naive Bayes:

It is a probabilistic classifier based on Bayes' Theorem, assuming features are independent and follows a gaussian distribution. It calculates class probabilities and selects the class with the highest probability for that feature variable.



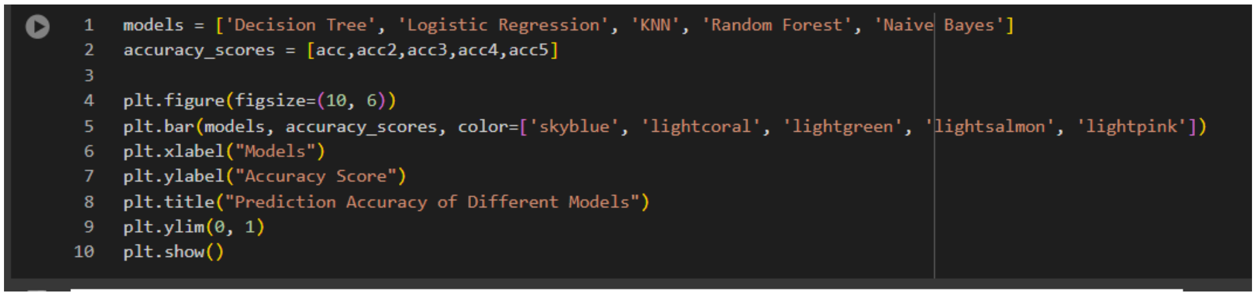
```
[58] from sklearn.naive_bayes import GaussianNB
     GaussianClassifier = GaussianNB()
     GaussianClassifier.fit(X_train, y_train)
```

```
GaussianNB
GaussianNB()
```

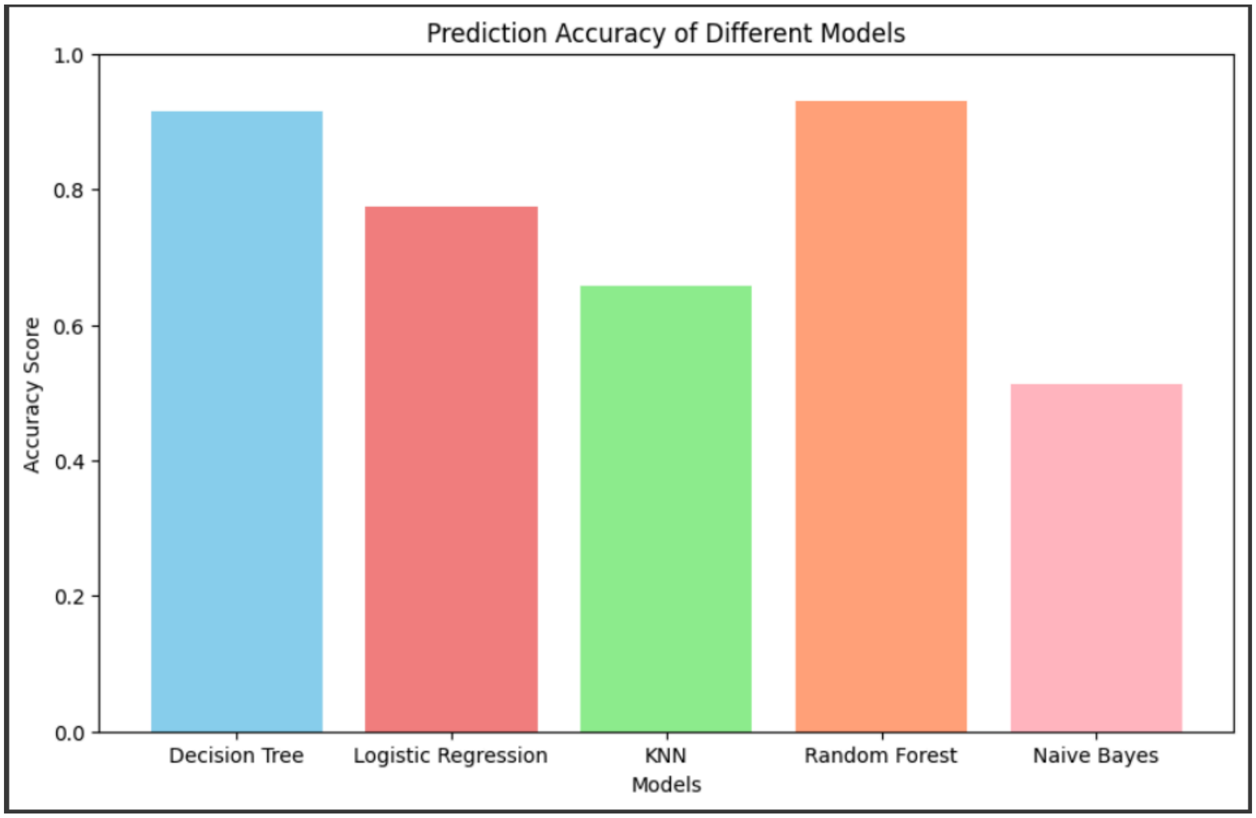
```
[59] pred5 = GaussianClassifier.predict(X_test)
```

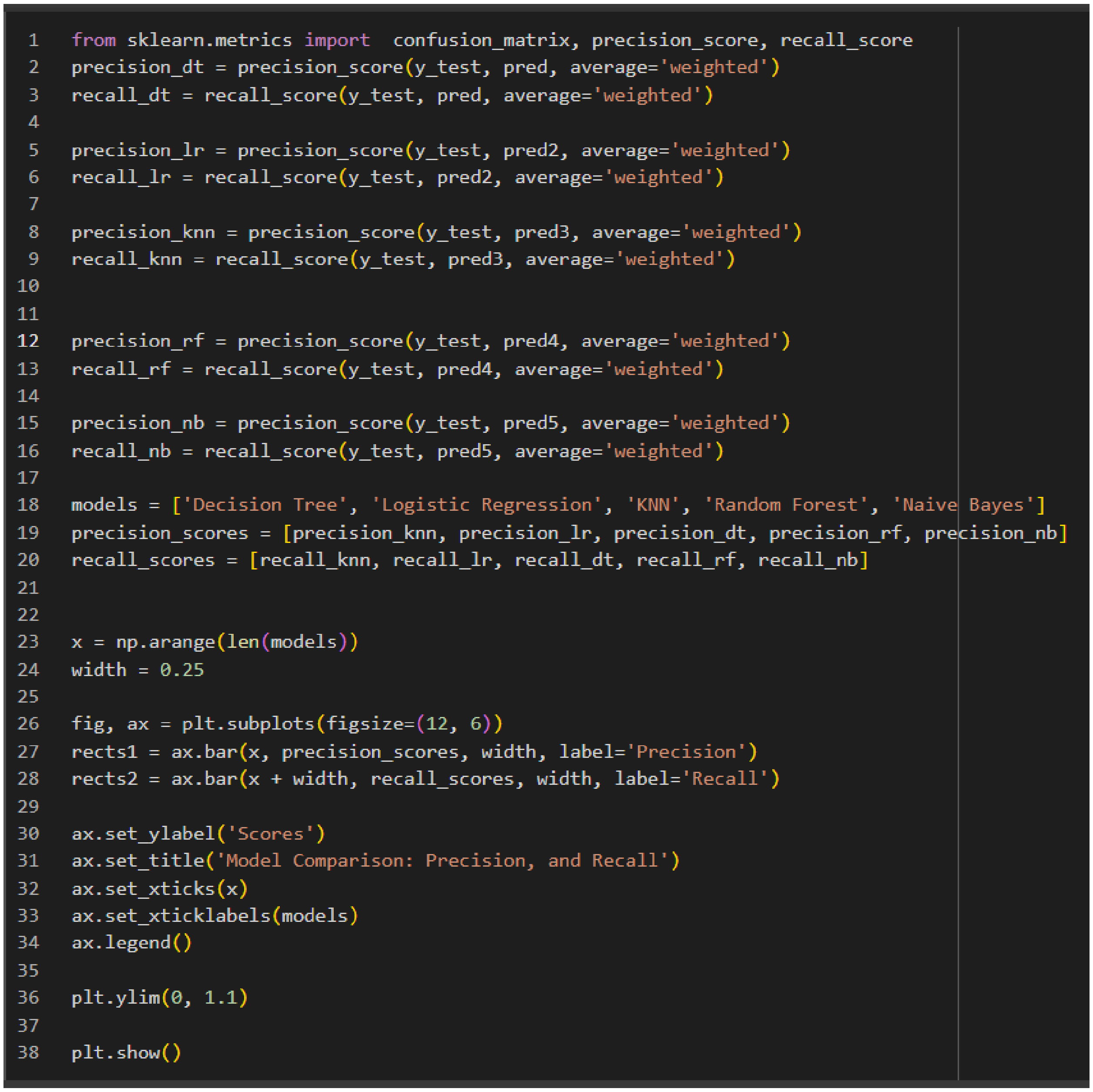
Model Selection/ Comparison Analysis:

- Barchart showcasing accuracy of the applied models.

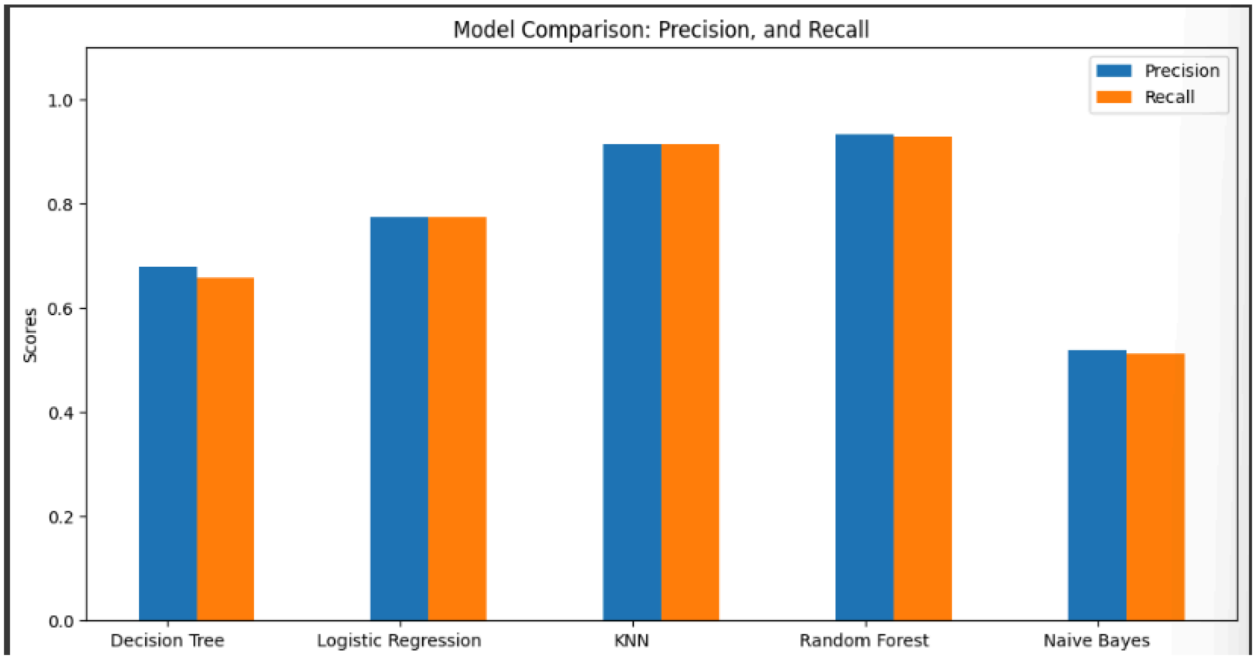


```python
models = ['Decision Tree', 'Logistic Regression', 'KNN', 'Random Forest', 'Naive Bayes']
accuracy_scores = [acc,acc2,acc3,acc4,acc5]

plt.figure(figsize=(10, 6))
plt.bar(models, accuracy_scores, color=['skyblue', 'lightcoral', 'lightgreen', 'lightsalmon', 'lightpink'])
plt.xlabel("Models")
plt.ylabel("Accuracy Score")
plt.title("Prediction Accuracy of Different Models")
plt.ylim(0, 1)
plt.show()
```

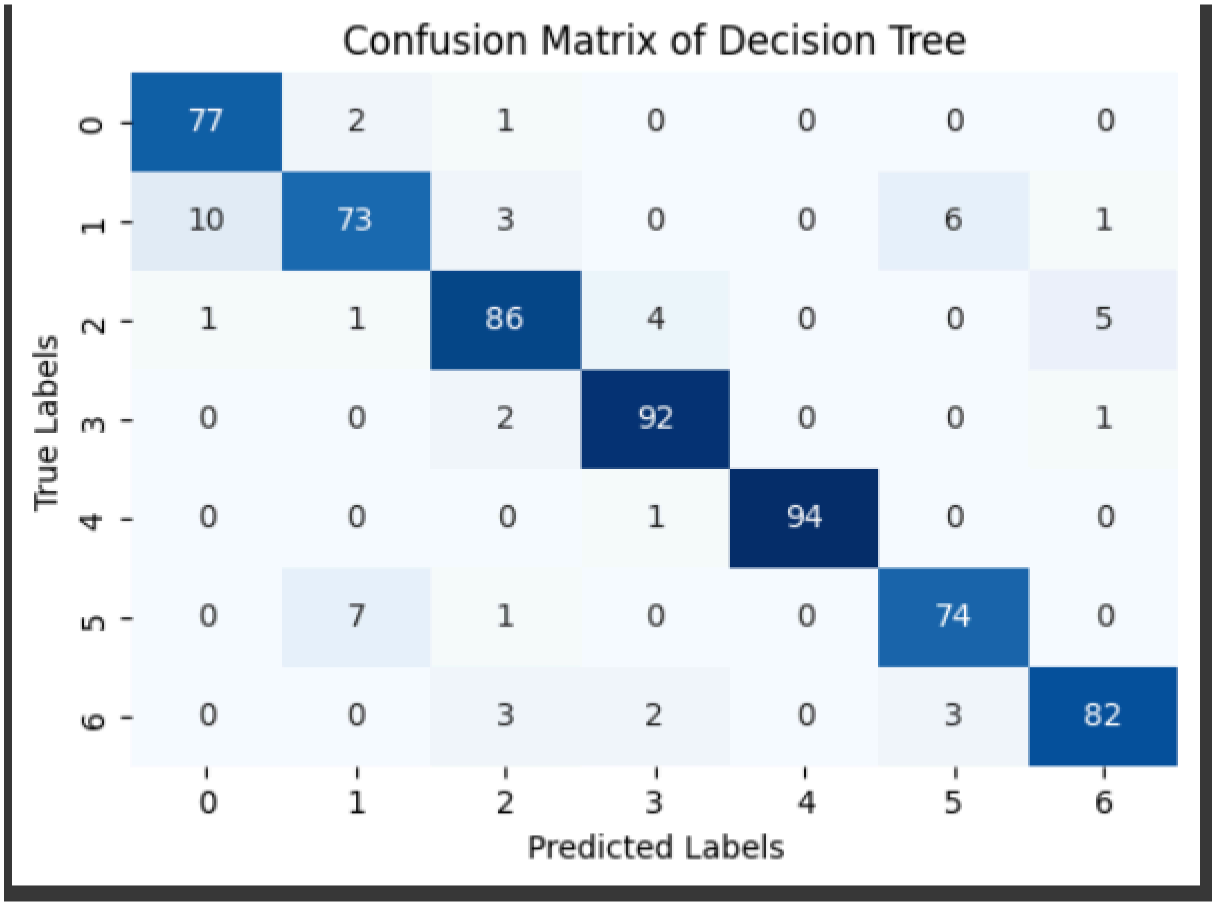




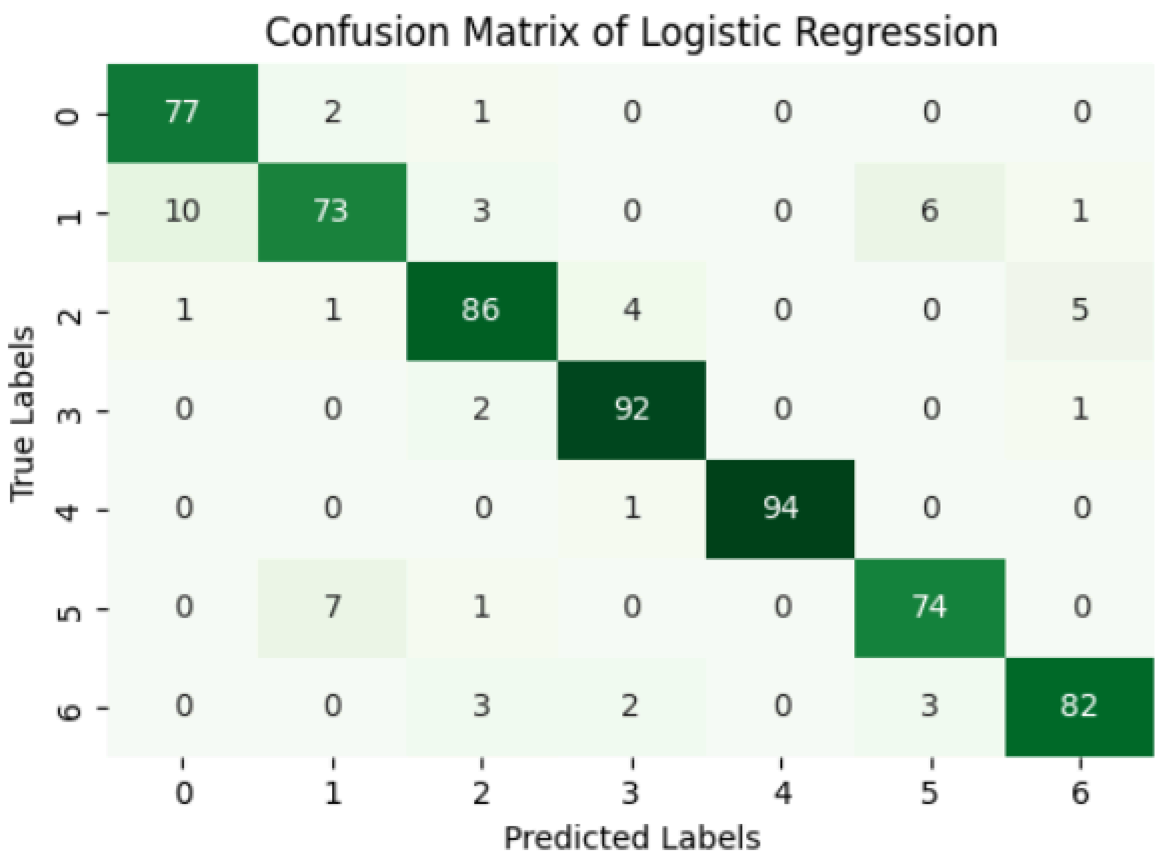
```python
from sklearn.metrics import  confusion_matrix, precision_score, recall_score
precision_dt = precision_score(y_test, pred, average='weighted')
recall_dt = recall_score(y_test, pred, average='weighted')

precision_lr = precision_score(y_test, pred2, average='weighted')
recall_lr = recall_score(y_test, pred2, average='weighted')

precision_knn = precision_score(y_test, pred3, average='weighted')
recall_knn = recall_score(y_test, pred3, average='weighted')


precision_rf = precision_score(y_test, pred4, average='weighted')
recall_rf = recall_score(y_test, pred4, average='weighted')

precision_nb = precision_score(y_test, pred5, average='weighted')
recall_nb = recall_score(y_test, pred5, average='weighted')

models = ['Decision Tree', 'Logistic Regression', 'KNN', 'Random Forest', 'Naive Bayes']
precision_scores = [precision_knn, precision_lr, precision_dt, precision_rf, precision_nb]
recall_scores = [recall_knn, recall_lr, recall_dt, recall_rf, recall_nb]


x = np.arange(len(models))
width = 0.25

fig, ax = plt.subplots(figsize=(12, 6))
rects1 = ax.bar(x, precision_scores, width, label='Precision')
rects2 = ax.bar(x + width, recall_scores, width, label='Recall')

ax.set_ylabel('Scores')
ax.set_title('Model Comparison: Precision, and Recall')
ax.set_xticks(x)
ax.set_xticklabels(models)
ax.legend()

plt.ylim(0, 1.1)

plt.show()
```

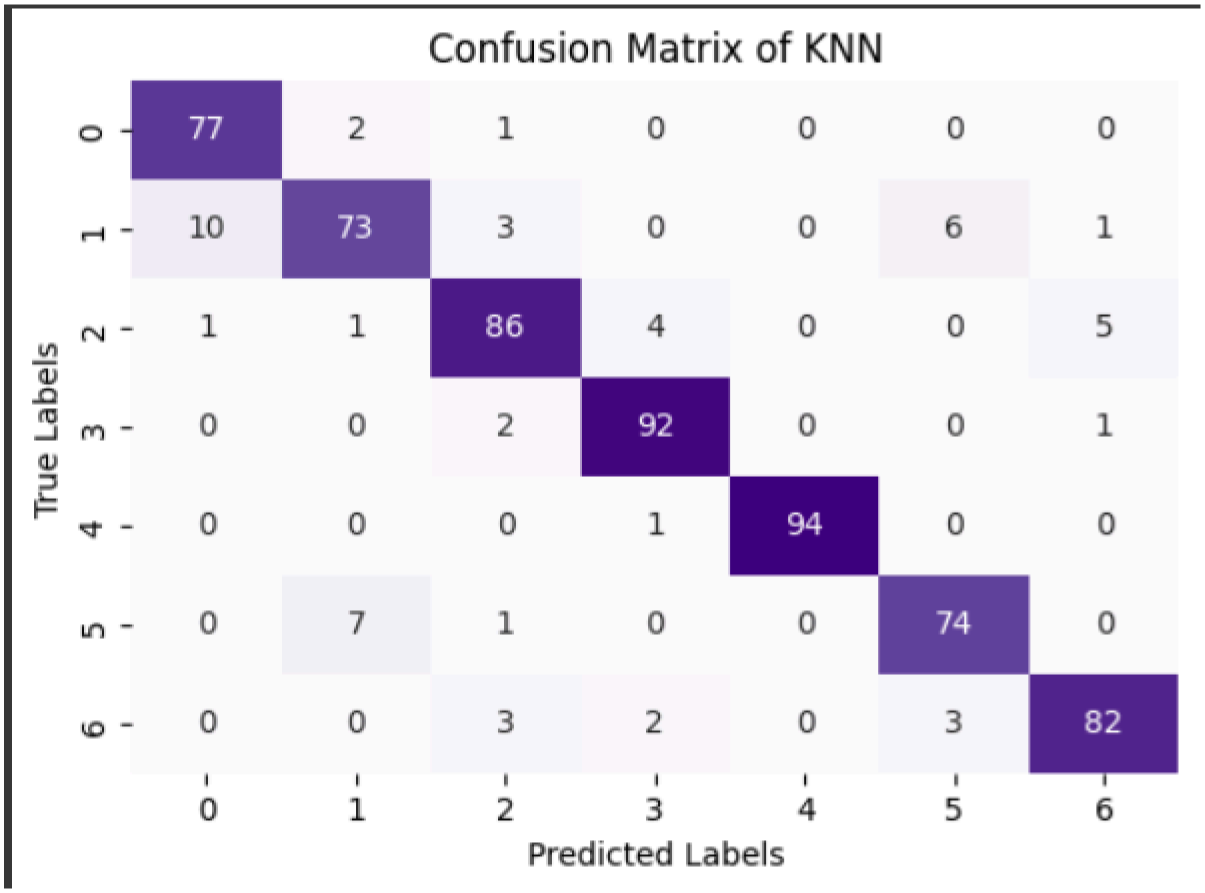


- **Confusion Matrix for each model:**

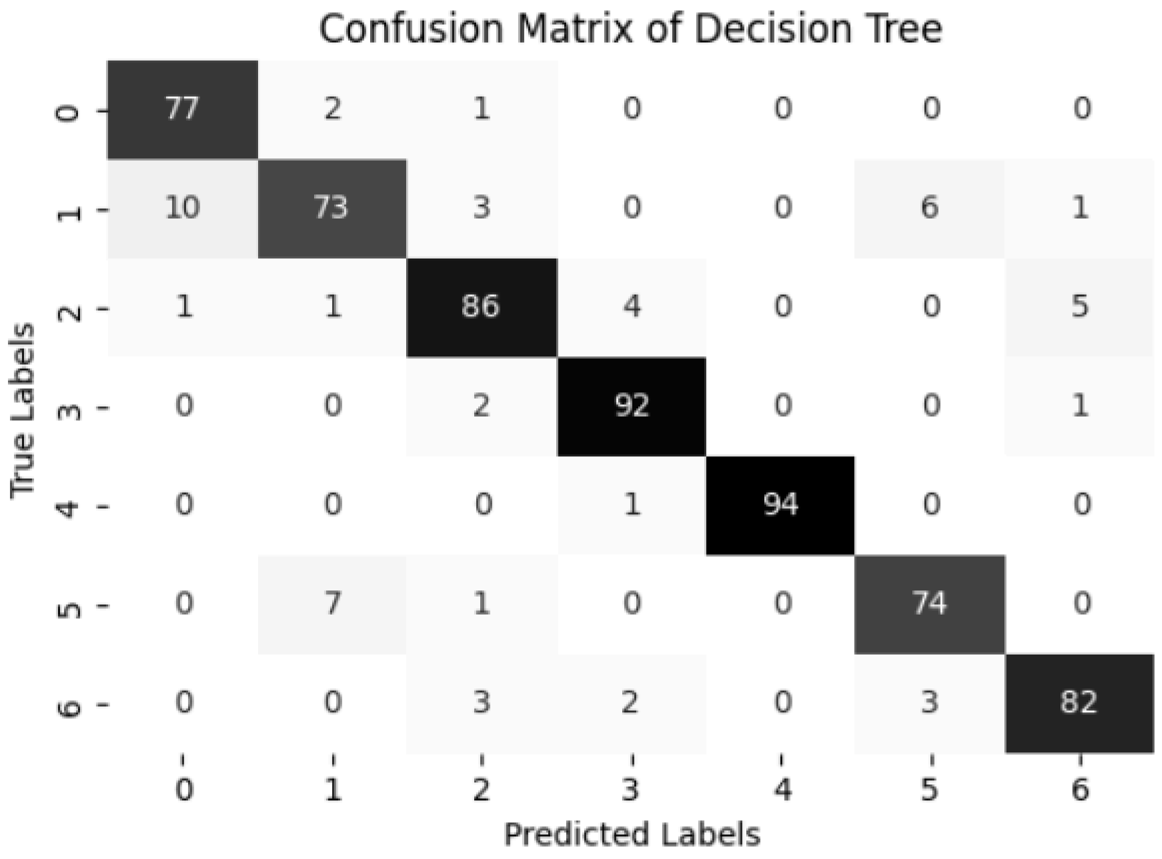


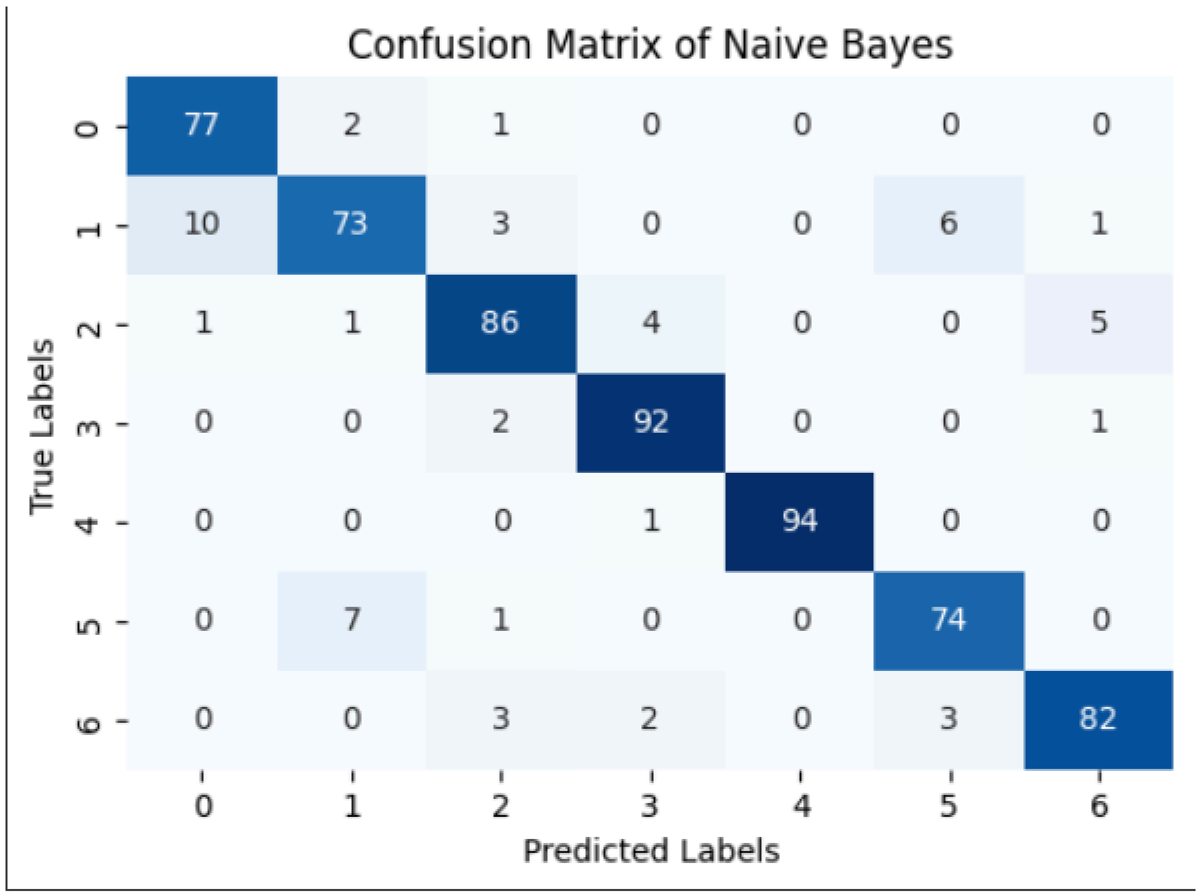
True \ Predicted	0	1	2	3	4	5	6
0	77	2	1	0	0	0	0
1	10	73	3	0	0	6	1
2	1	1	86	4	0	0	5
3	0	0	2	92	0	0	1
4	0	0	0	1	94	0	0
5	0	7	1	0	0	74	0
6	0	0	3	2	0	3	82


Confusion Matrix of Logistic Regression
True Labels
Predicted Labels
0 1 2 3 4 5 6
77 2 1 0 0 0 0
10 73 3 0 0 6 1
1 1 86 4 0 0 5
0 0 2 92 0 0 1
0 0 0 1 94 0 0
0 7 1 0 0 74 0
0 0 3 2 0 3 82



	0	1	2	3	4	5	6
0	77	2	1	0	0	0	0
1	10	73	3	0	0	6	1
2	1	1	86	4	0	0	5
3	0	0	2	92	0	0	1
4	0	0	0	1	94	0	0
5	0	7	1	0	0	74	0
6	0	0	3	2	0	3	82


Confusion Matrix of Decision Tree
True Labels
Predicted Labels
0 1 2 3 4 5 6
77 2 1 0 0 0 0
10 73 3 0 0 6 1
1 1 86 4 0 0 5
0 0 2 92 0 0 1
0 0 0 1 94 0 0
0 7 1 0 0 74 0
0 0 3 2 0 3 82



	0	1	2	3	4	5	6
0	77	2	1	0	0	0	0
1	10	73	3	0	0	6	1
2	1	1	86	4	0	0	5
3	0	0	2	92	0	0	1
4	0	0	0	1	94	0	0
5	0	7	1	0	0	74	0
6	0	0	3	2	0	3	82

Conclusion:

This project successfully demonstrates the application of machine learning to predict obesity levels based on eating habits, physical activity, and demographic data. By analyzing the dataset, we explored various features and addressed preprocessing steps like handling missing values, scaling, and encoding categorical data.

Through training and testing multiple models — Decision Tree, Logistic Regression, K-Nearest Neighbors, Random Forest, and Gaussian Naive Bayes — we compared their performance using metrics like accuracy and confusion matrices. These models provide valuable insights into identifying individuals at risk of obesity and highlight the potential of data-driven methods to support early intervention.

The project emphasizes the importance of lifestyle factors in determining obesity levels, advocating for actionable insights to promote healthier living. Our results demonstrate the ability of machine learning to provide accurate and interpretable predictions, showcasing its role in addressing real-world health challenges like obesity.