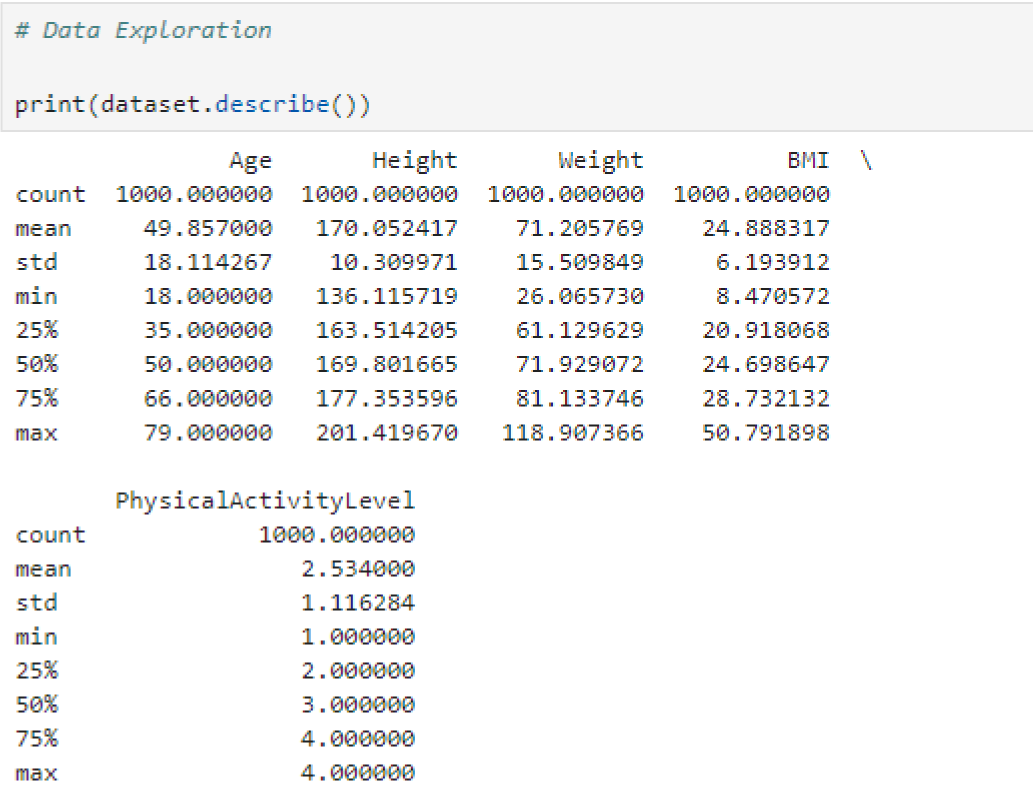
Obesity prediction

The Kaggle link of this dataset is https://www.kaggle.com/datasets/mrsimple07/obesity-prediction and the usability rating is 10.

In this project we are trying to predict the Obesity category of a person based on their gender, age, height, weight, BMI and physical activity level. The target is a categorical variable with values normal weight, obese, overweight and underweight. An individual based on their features described earlier will be classified to fall under one of this target categories. The total records in the dataset are 1000.

Data Exploration

- Data preparation issues: As gender is categorical, this needs to be encoded to be numerical for comparison.

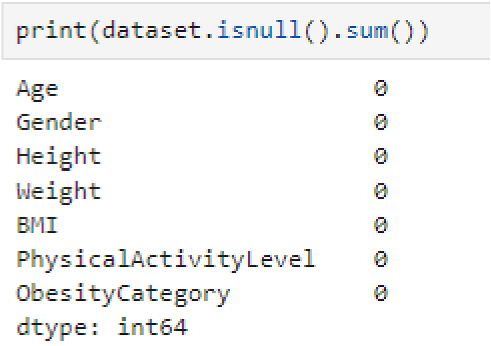


```
# Data Exploration

print(dataset.describe())
```

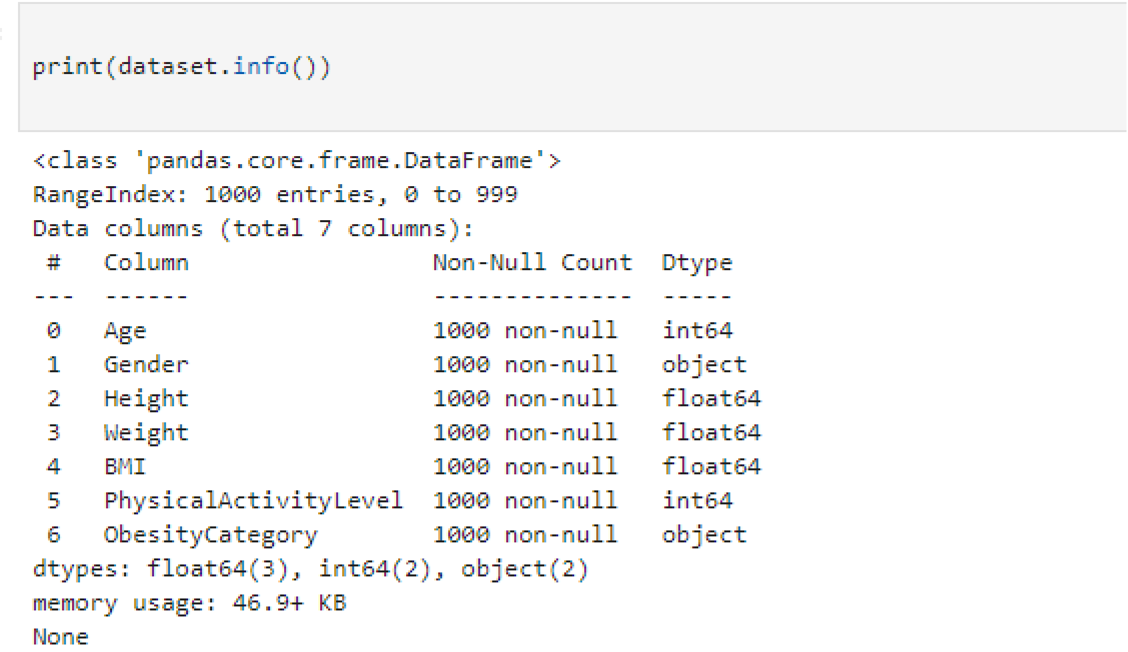
```
               Age       Height       Weight          BMI  \
count  1000.000000  1000.000000  1000.000000  1000.000000
mean     49.857000   170.052417    71.205769    24.888317
std      18.114267    10.309971    15.509849     6.193912
min      18.000000   136.115719    26.065730     8.470572
25%      35.000000   163.514205    61.129629    20.918068
50%      50.000000   169.801665    71.929072    24.698647
75%      66.000000   177.353596    81.133746    28.732132
max      79.000000   201.419670   118.907366    50.791898

       PhysicalActivityLevel
count            1000.000000
mean                2.534000
std                 1.116284
min                 1.000000
25%                 2.000000
50%                 3.000000
75%                 4.000000
max                 4.000000
```



```
print(dataset.isnull().sum())
```

```
Age                     0
Gender                  0
Height                  0
Weight                  0
BMI                     0
PhysicalActivityLevel   0
ObesityCategory         0
dtype: int64
```

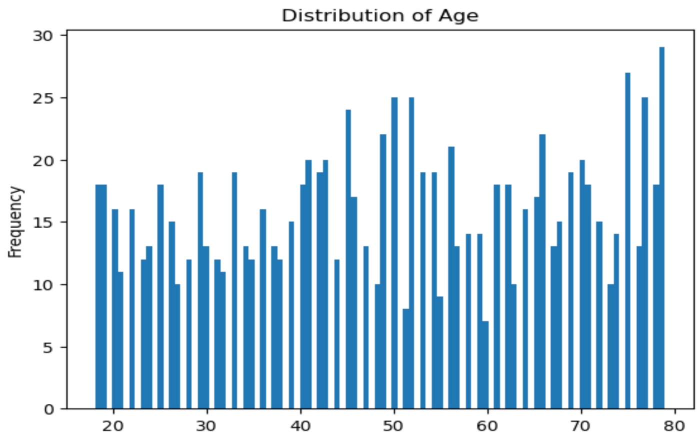


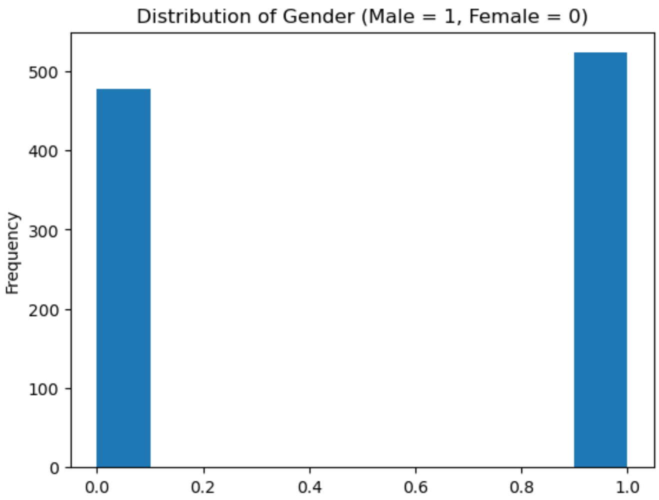
```
print(dataset.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 7 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Age                    1000 non-null   int64
 1   Gender                 1000 non-null   object
 2   Height                 1000 non-null   float64
 3   Weight                 1000 non-null   float64
 4   BMI                    1000 non-null   float64
 5   PhysicalActivityLevel  1000 non-null   int64
 6   ObesityCategory        1000 non-null   object
dtypes: float64(3), int64(2), object(2)
memory usage: 46.9+ KB
None
```

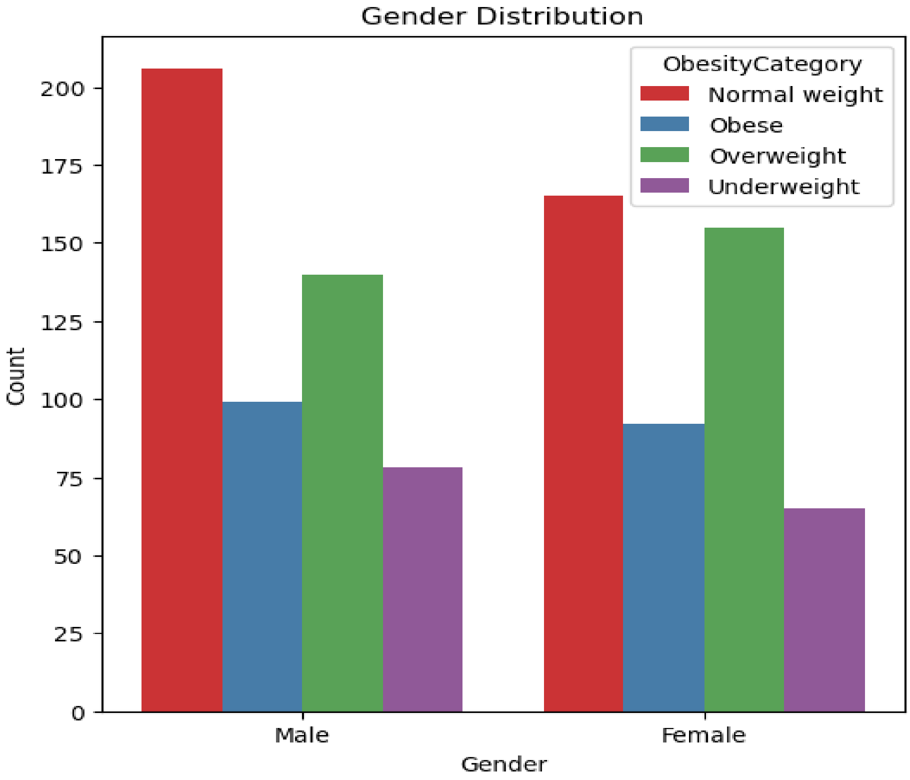
Feature Analysis

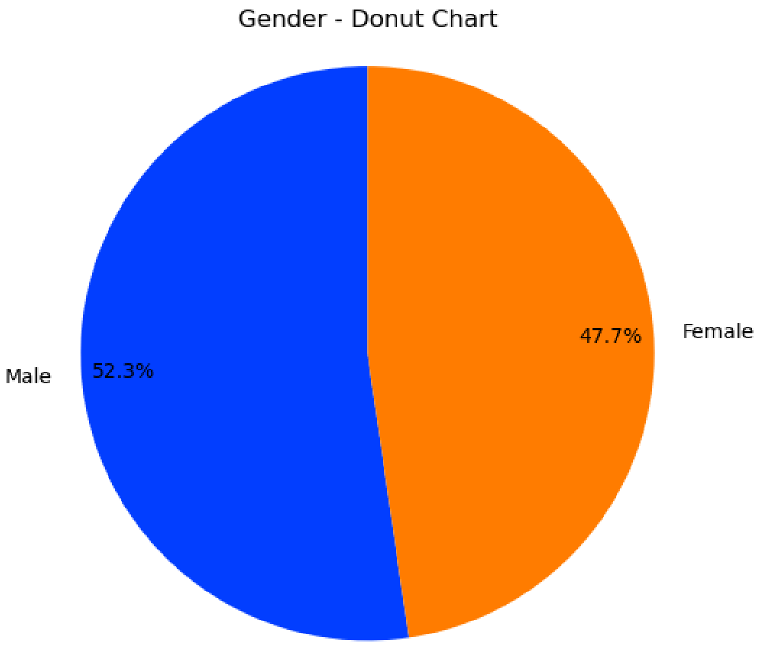
Below are some summary statistics of the dataset. Firstly, we would like to understand if the dataset has a good representation of the population. To this end, we start by looking at the histograms of age and gender.



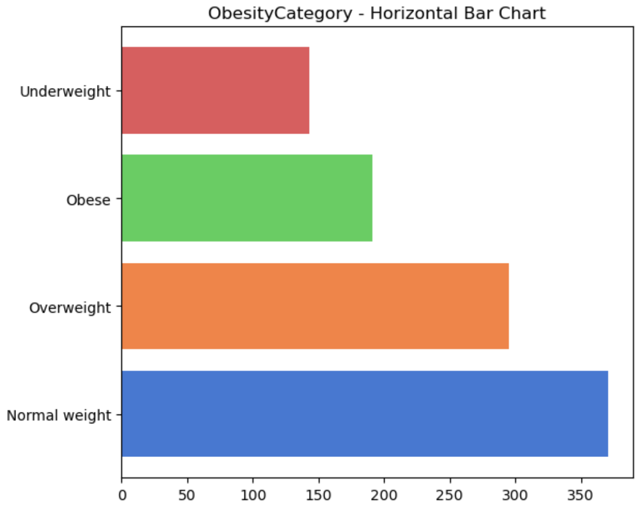


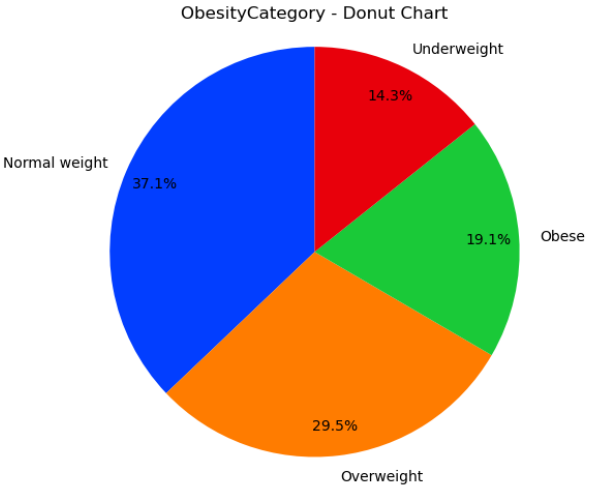
Based on the gender, the data is equally distributed.



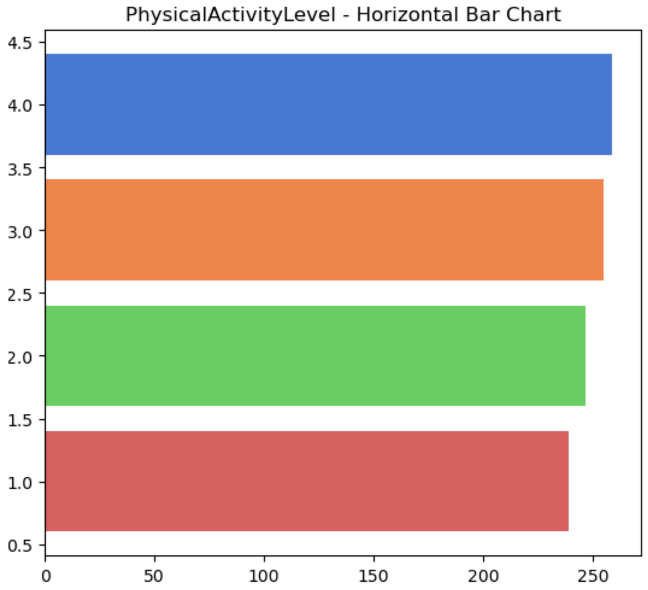


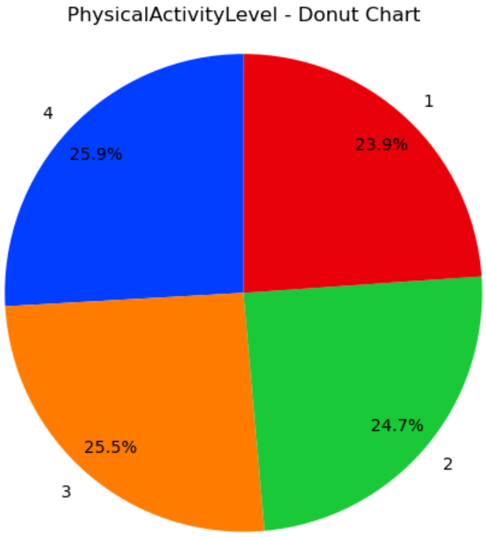
Plotting the number of people under each target label of obesity category.



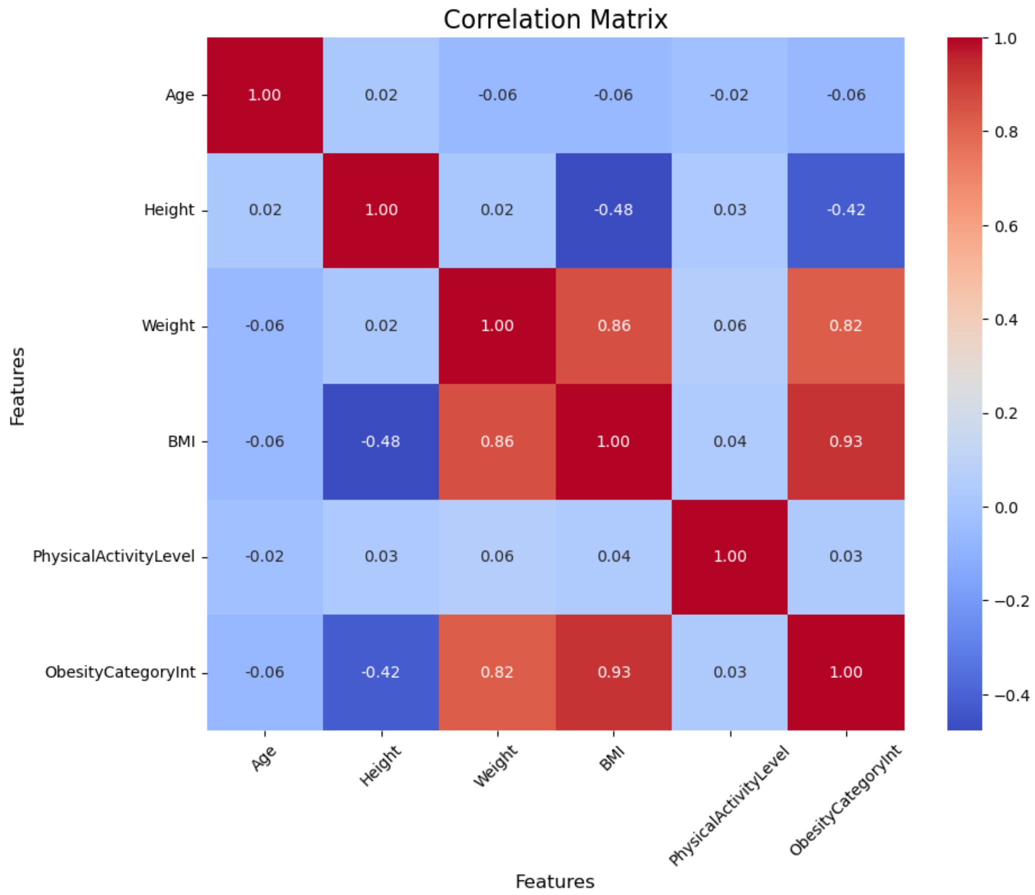


Based on physical activity level,





As BMI is based on height and weight, which is evident from the below correlation map. Hence those features can be dropped.



To be able to plot obesity category along with the other variables we convert obesity category into the following ordered integer values.

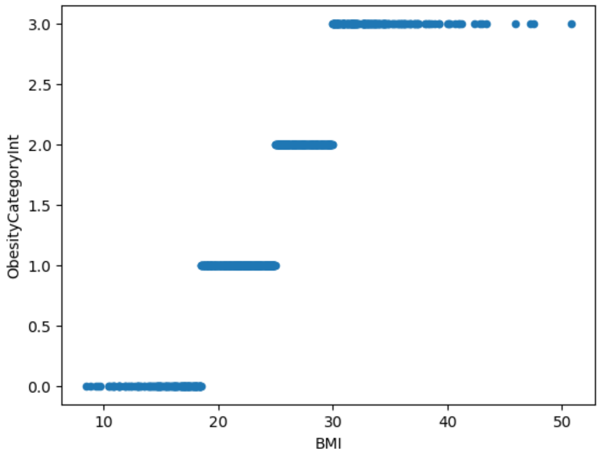
Underweight = 0

Normal weight = 1

Overweight = 2

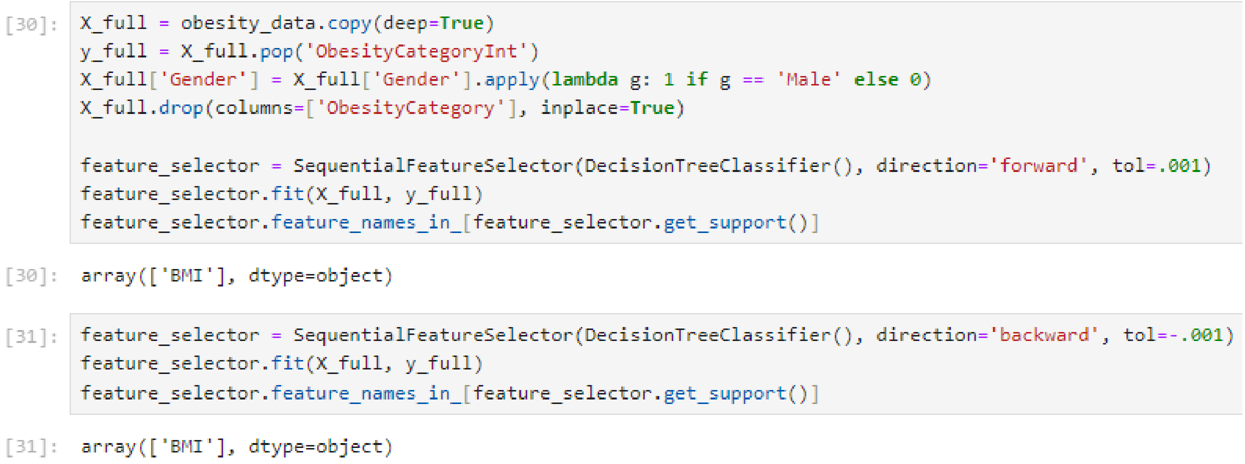
Obese = 3

We see the following by plotting obesity category vs. BMI.



Attribute selection

The relationship between BMI and features such as age, gender, height and weight are more interesting. Typically, BMI is calculated based on these features. The obesity category itself then is typically based on the range for the BMI. Therefore, if we know BMI of a person then we can in principle ignore all the other features to predict the obesity category. And this is evident from the below.



```
[30]: X_full = obesity_data.copy(deep=True)
      y_full = X_full.pop('ObesityCategoryInt')
      X_full['Gender'] = X_full['Gender'].apply(lambda g: 1 if g == 'Male' else 0)
      X_full.drop(columns=['ObesityCategory'], inplace=True)

      feature_selector = SequentialFeatureSelector(DecisionTreeClassifier(), direction='forward', tol=.001)
      feature_selector.fit(X_full, y_full)
      feature_selector.feature_names_in_[feature_selector.get_support()]
```

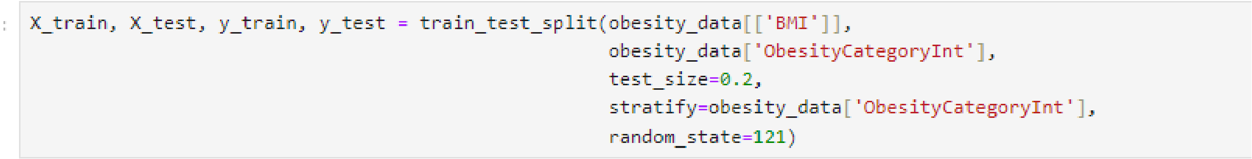
```
[30]: array(['BMI'], dtype=object)
```

```
[31]: feature_selector = SequentialFeatureSelector(DecisionTreeClassifier(), direction='backward', tol=-.001)
      feature_selector.fit(X_full, y_full)
      feature_selector.feature_names_in_[feature_selector.get_support()]
```

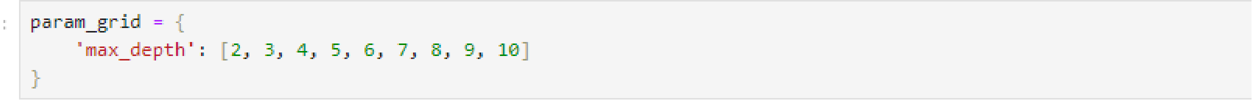
```
[31]: array(['BMI'], dtype=object)
```

Our Model- Decision Tree with Grid Search cross validation

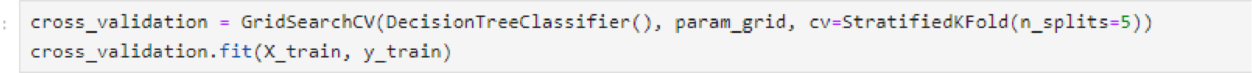
It appears that it is sufficient to find a model that maps BMI to Obesity Category. Given the strong split between categories, we could consider a decision tree. We indeed see a very strong performance even with this simple model.



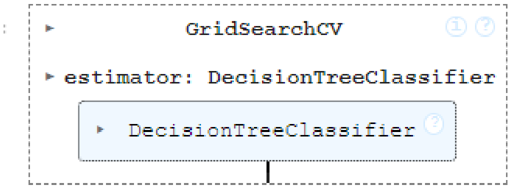
```
X_train, X_test, y_train, y_test = train_test_split(obesity_data[['BMI']],
                                                    obesity_data['ObesityCategoryInt'],
                                                    test_size=0.2,
                                                    stratify=obesity_data['ObesityCategoryInt'],
                                                    random_state=121)
```



```
param_grid = {
    'max_depth': [2, 3, 4, 5, 6, 7, 8, 9, 10]
}
```

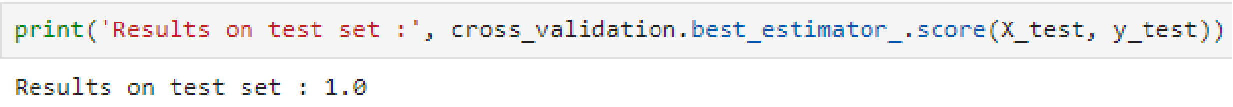


```
cross_validation = GridSearchCV(DecisionTreeClassifier(), param_grid, cv=StratifiedKFold(n_splits=5))
cross_validation.fit(X_train, y_train)
```



```
▸        GridSearchCV
▸ estimator: DecisionTreeClassifier
    ▸ DecisionTreeClassifier
```

And we are getting 100% accuracy



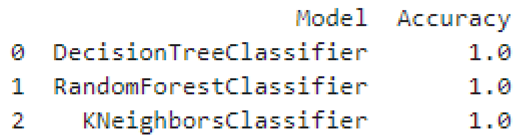
```
print('Results on test set :', cross_validation.best_estimator_.score(X_test, y_test))

Results on test set : 1.0
```

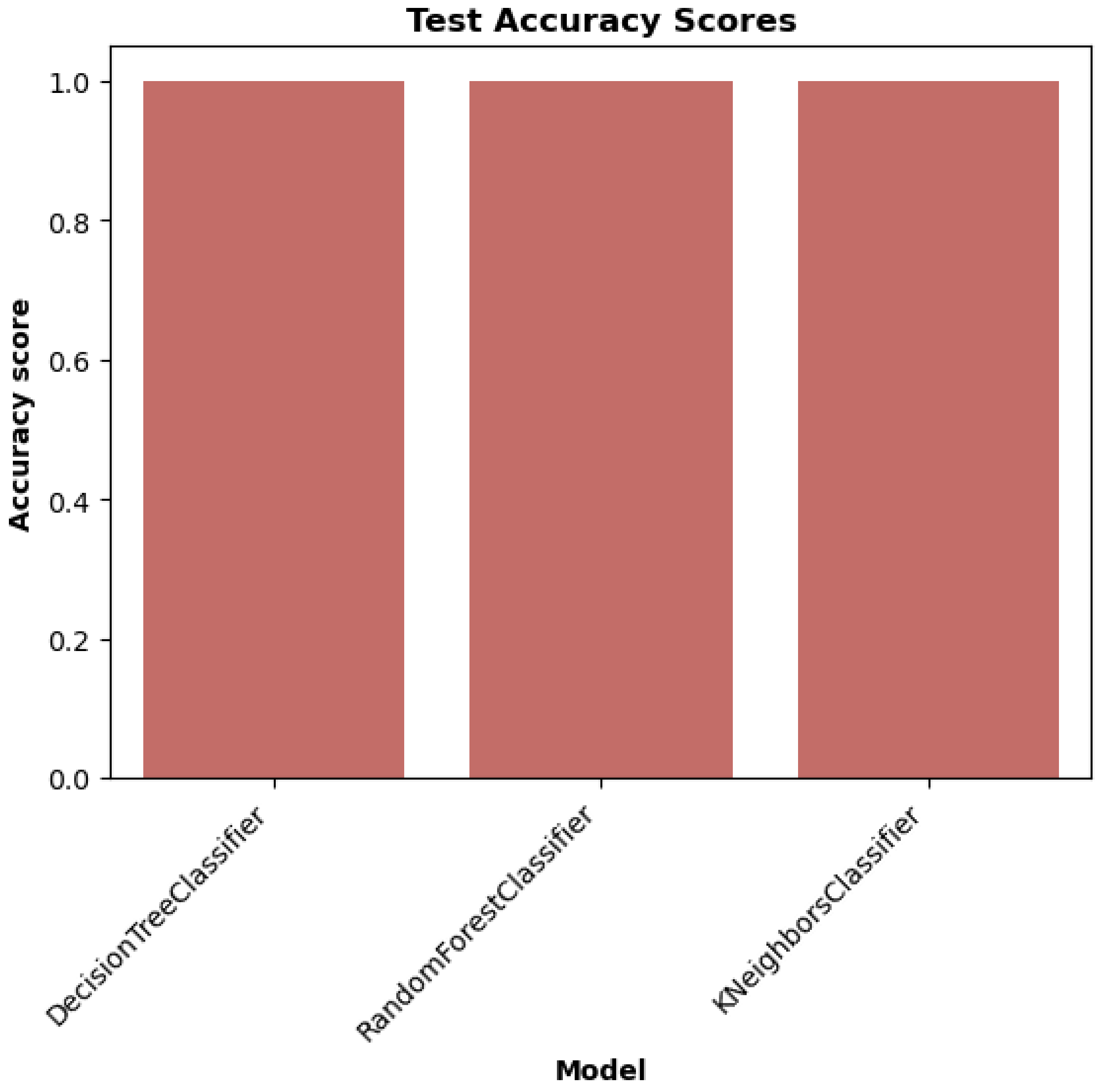
We also did a comparison with KNN and Random Forest and got 100% accuracy score.

Comparison with different classifiers

The dataset was run through different classifiers and below the comparison chart of the accuracy score.

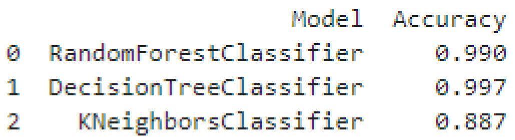


```
                    Model  Accuracy
0  DecisionTreeClassifier       1.0
1  RandomForestClassifier       1.0
2    KNeighborsClassifier       1.0
```

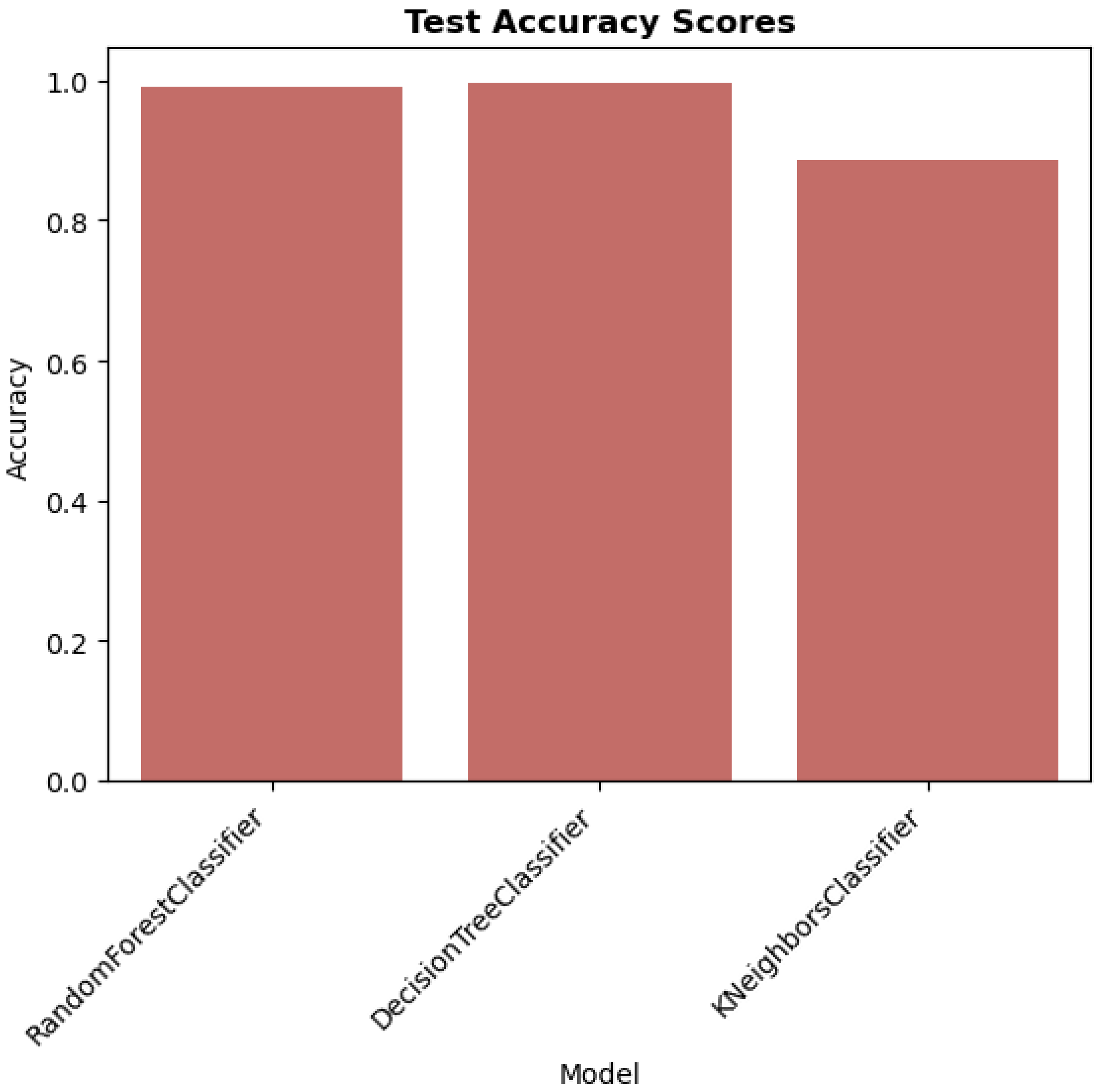


Contribution by other Kaggle participants

All the Kaggle participants seem to miss the fact that BMI alone is a sufficient predictor of obesity category. Most of the participants have used a mixture of different models to achieve an accuracy that is not any better than the model Decision Tree/Random Forest/KNN with Grid Search. We also tried to implement the 3 models using all features like other Kaggle contributors and obtained the below accuracy score.



```
                    Model  Accuracy
0  RandomForestClassifier     0.990
1  DecisionTreeClassifier     0.997
2    KNeighborsClassifier     0.887
```



Code implementation is present in the Github link – Obesity Prediction

Project Team – Sheema Selvam & Jayasree Jandhyala

Responsibilities of the team – Analyzing the Kaggle dataset, feature selection, comparing different Machine Learning models for classification

Sheema – Classification based on all features and comparison with different models with stratified k-fold validation

Jayasree – Classification based on only BMI and comparison with different models with stratified k-fold validation and Grid Search