# CS 634-Data mining

# Final Project

Topic: Supervised Data mining (Classification)

Data: Predicting whether an individual is obese or not based on their eating habits and physical condition

Name: Veena Chaudhari
Email: vac38@njit.edu
Github: https://github.com/vac38/Classification_of_obesity.git

# Table of contents

# INTRODUCTION

Machine learning is a branch of Artificial intelligence that involves training an algorithm to identify patterns and make decisions with minimal human intervention.

Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain. It consists of several layers of networks called neural networks. Deep learning algorithms can handle complex classification tasks such as image recognition, speech recognition ,music generation , time series analysis, etc.

Option chosen: Option 1
Task: classification
Machine learning models:

1) logistic regression
2) Decision tree
3) Random forest

Deep learning model : LSTM
Evaluation metrics: 10-fold cross validation
Models evaluated based on :

1) Accuracy,
2) Sensitivity,
3) Specificity,
4) Precision,
5) F1_score,
6) Error rate,
7) Negative predictive value,
8) False positive rate,
9) False Discovery Rate,
10) False negative rate,
11) Balanced Accuracy(BACC),
12) True Skills Statistics(TSS),
13) Heidke Skill Score (HSS)

## REQUIREMENTS

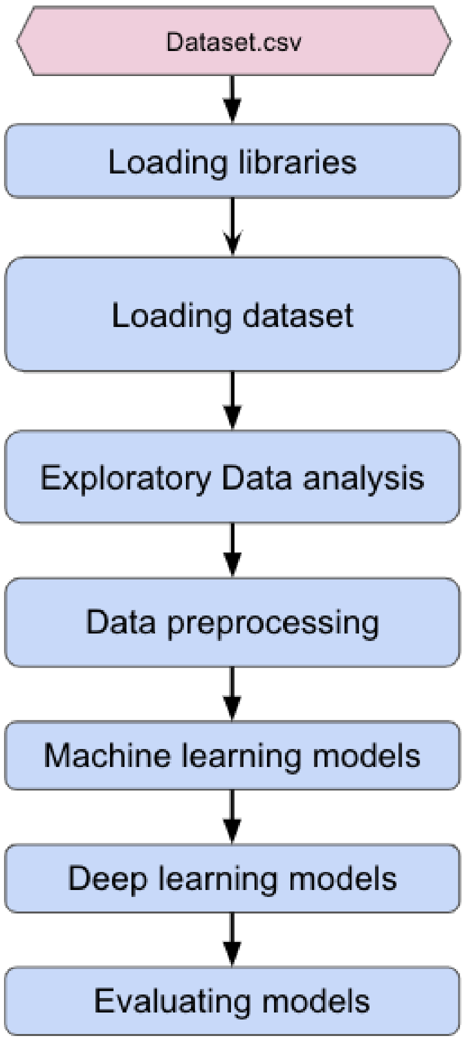
### Software:

Python 3.8 with Python libraries installed:  Pandas, numpy, seaborn, matplotlib, scipy, sklearn and tensorflow .

### Hardware:

MacBook Air 13-inch, Dual-Core Intel Core i5 with 8 GB RAM

## OVERALL FLOWCHART FOR IMPLEMENTATION



Flowchart for overall process

## DATASET

The data set used in this project is predicting whether a person is obese or not based on their eating habits and physical conditions.
The link to dataset is
https://archive.ics.uci.edu/ml/datasets/Estimation+of+obesity+levels+based+on+eating+habits+and+physical+condition+

The Data contains 16 features and 2117 records.The Label column is 'Obesity.
The data is collected from a survey that was conducted for 2117 participants, in which each participant was asked 16 questions and their responses were collected.

All of the 16 features ,along with the question as for each feature  and the possible answer categories are listed in Table 1

Table 1: All 16 features in Obesity data set with the question as for each feature  and the possible answer categories

| Sr.No | Feature | Questions | Categories/Answers |
|---|---|---|---|
| 1 | Gender | What is your gender? | Male or Female |
| 2 | Age | What is your age? | Numeric value |
| 3 | Height | What is your height? | Numeric value in meters |
| 4 | Weight | What is your weight? | Numeric value in kilograms |
| 5 | family history of obesity | Has a family member suffered or suffers from being overweight? | Yes,No |
| 6 | Frequent consumption of high caloric food (FAVC) | Do you eat high caloric food frequently? | Yes,No |
| 7 | Frequency of consumption of vegetables (FCVC) | Do you usually eat vegetables in your meals? | Never, Sometimes,always |

| | | | |
|---|---|---|---|
| 8 | Number of main meals (NCP) | How many main meals do you have daily? | 1, 2, 3 or 4 meals a day |
| 9 | Consumption of food between meals (CAEC) | Do you eat any food between meals? | No,Sometimes,Frequently, always |
| 10 | SMOKE | Do you smoke? | Yes,No |
| 11 | Consumption of water daily (CH20) | How much water do you drink daily? | Less than a liter, Between 1 and 2 L, more than 2 L |
| 12 | Calories consumption monitoring (SCC) | Do you monitor the calories you eat daily? | Yes, No |
| 13 | Physical activity frequency (FAF) | What is your frequency of physical activity? | I do not, 1 or 2 days, 2 or 4 days, 4 or 5 days |
| 14 | Time using technology devices (TUE) | How much time do you spend using technological devices such as cell phones, video games, television, computers and others? | 0–2 hours, 3–5 hours, More than 5 hours |
| 15 | Consumption of alcohol (CALC) | How often do you drink alcohol? | I do not drink,Sometimes, Frequently, Always |
| 16 | Transportation used (MTRANS) | Which transportation do you usually use? | Automobile, Motorbike, Bike, Public Transportation, Walking |

# EXPLORATORY DATA ANALYSIS FOR OBESITY DATA

Exploratory data analysis is performed on the Obesity data to determine the null values, correlation between features  and understand the distribution of features influencing obesity.

1) **Renaming the column names for better  understanding:** Columns in dataset were labeled to Gender, Age,Height, ,Weight family_history_with_overweight, high_caloric_food,vegetables_consumption,main_mealsfood_between_meals, SMOKE,Daily_water,Calories_consumption,physical_activity,technology_devices, Alcohol_consumption, Transportation_used and Obesity.

2) **Converting class within the labels for each record into binary values:**

In the original dataset each record is classified using the classes Insufficient Weight, Normal Weight, Overweight Level I, Overweight Level II, Obesity Type I, Obesity Type II and Obesity Type III.  Since the task for this project is to perform binary classification, the labels were categorized in to Normal or Obese using the following stratergy:

- Insufficient Weight, Normal Weight, Overweight Level I, Overweight Level II → Categorized to class  'NORMAL'
- Obesity Type II and Obesity Type III →  categorized to class 'OBESE'

The count for each class within the label: Normal and obese are checked  to determine if there is imbalanced data. Imbalanced data is data where the classes are not represented equally. Using imbalance data for training a model will create a biased model , which will increase the testing error.

3) **Null values**: Checking for missing values/null values using isnull() function and plotting the results using heatmap from seaborn. Null values in data should be handled as machine learning algorithms cannot perform on data with missing data.
4) **Plotting important features to determine their distribution**: Age, Gender , height and weight are plotted from the list of features.
   - The plot for age determines the age range of records in data
   - Plot for height, weight and gender tells about the distribution of  height and weight for the males and females.

- Gender , height and weight are plotted against obesity to determine how height and weight influence obesity in the different genders. These 3 features are considered beacuse the body mass index is a important factor for categorizing a person as obese. The body mass index is a measure of body fat based on height and weight that applies to adult men and women. It is calculated by taking the ratio of weight in kg to square of height in meters. A BMI of greater than 30 is considered as obese.

## DATA PREPROCESSING

1) **Label Encoding**: classification models can handle machine readable forms, therefore labels in data are converted to numerical forms using label encoding. Label encoder class from sklearn is used to perform label encoding. The features with modified feature names (for simplicity) and the labels after encoding are in the table .

Note: 1- yes means value yes was labeled as '1'.

Table 2: All columns with legend for label encoding

| Sr.No | Feature | Categories/Answers |
|---|---|---|
| 1 | Gender | 1- Male<br>0 -Female |
| 2 | Age | Numeric value |
| 3 | Height | Numeric value in meters |
| 4 | Weight | Numeric value in kilograms |
| 5 | family_history_with_overweight | 1-Yes<br>0-No |
| 6 | high_caloric_food | 1-Yes<br>0-No |
| 7 | Vegetable consumption | 1 = never<br>2 = sometimes<br>3 = always |
| 8 | Main meals | 1, 2, 3 or 4 meals a day |
| 9 | food between meals | 1=no<br>2=sometimes<br>3=frequently<br>4=always |
| 10 | SMOKE | 1-Yes<br>0-No |
| 11 | Daily water | 1 = less than a liter<br>2 = 1–2 liters |

| | | |
|---|---|---|
| | | 3 = more than 2 liters |
| 12 | calorie_consumption | 1-Yes<br>0-No |
| 13 | physical_activity | 0 = none<br>1 = 1 to 2 days<br>2= 2 to 4 days<br>3 = 4 to 5 days |
| 14 | Technology devices | 0 = 0–2 hours<br>1 = 3–5 hours<br>2 = more than 5 hours |
| 15 | Alcohol_consumption | 1= never<br>2 = sometimes<br>3 = frequently<br>4 = always |
| 16 | Transportation used | 1 = automobile<br>2 = motorbike<br>3 = bike<br>4 = public transportation<br>5= walking |
| 17 | Obesity | 0 - Normal<br>1- Obese |

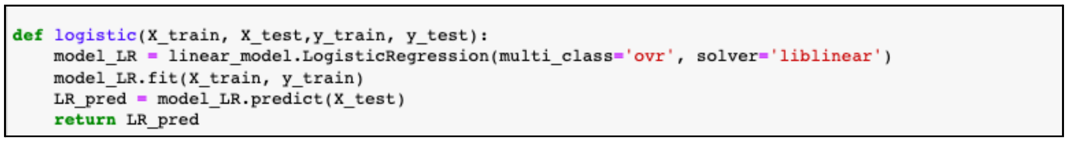
2) **Correlation between features**: Correlation between features is plotted using a heatmap to determine if two features are closely related. This step is a part of feature selection. Highly correlated features are determined and removed to increase the speed and accuracy of the model.

3) **Normalization of Data**: The range of values for each feature are different. For example weight ranges from 39 kgs to 173 kgs and gender has only two values: 0 and 1. Therefore to convert all feature values between 0 and 1 , normalization is performed.

# CLASSIFICATION MODELS

## Machine learning models used for predicting Obesity

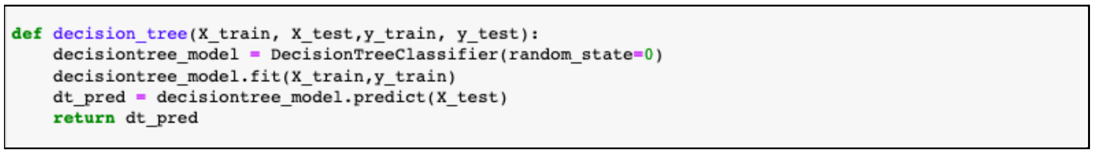
1) Logistic Regression: Logistic regression is performed on the data using the following function
    -Syntax: logistic(X_train, X_test,y_train, y_test)
    -Input parameters: training and testing data for X and Y
    -output Parameters: Predicted values



```
def logistic(X_train, X_test,y_train, y_test):
    model_LR = linear_model.LogisticRegression(multi_class='ovr', solver='liblinear')
    model_LR.fit(X_train, y_train)
    LR_pred = model_LR.predict(X_test)
    return LR_pred
```

Figure 1: Function for logistic regression

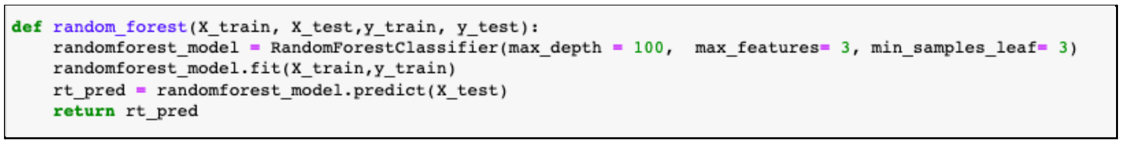
2) Decision Tree: Decision tree is performed on the data using the following function
    -Syntax: decision_tree(X_train, X_test,y_train, y_test):
    -Input parameters: training and testing data for X and Y
    -output Parameters: Predicted values



```
def decision_tree(X_train, X_test,y_train, y_test):
    decisiontree_model = DecisionTreeClassifier(random_state=0)
    decisiontree_model.fit(X_train,y_train)
    dt_pred = decisiontree_model.predict(X_test)
    return dt_pred
```

Figure 2: Function for Decision tree

3) Random forest: Random Forest is performed on the data using the following function
    -Syntax: random_forest(X_train, X_test,y_train, y_test):
    -Input parameters: training and testing data for X and Y
    -output Parameters: Predicted values

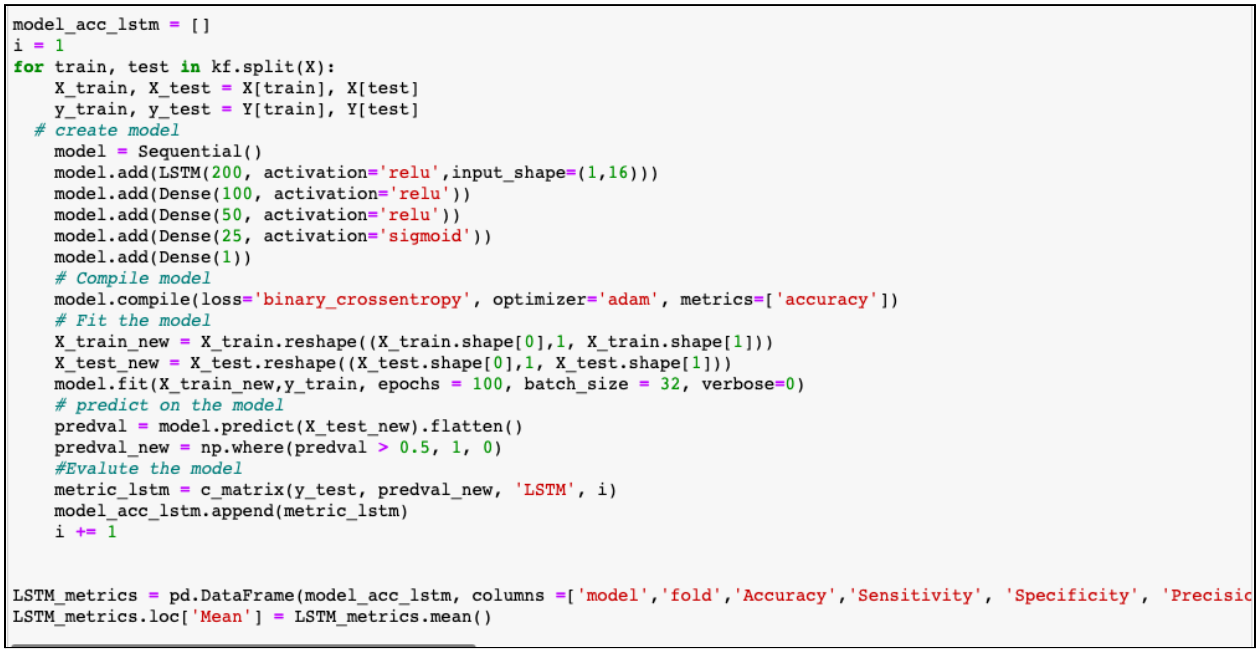


```
def random_forest(X_train, X_test,y_train, y_test):
    randomforest_model = RandomForestClassifier(max_depth = 100,  max_features= 3, min_samples_leaf= 3)
    randomforest_model.fit(X_train,y_train)
    rt_pred = randomforest_model.predict(X_test)
    return rt_pred
```

Figure 3: Function for Random forest

## Deep learning models for predicting Obesity

Long short term memory (LSTM) is performed on data using following code

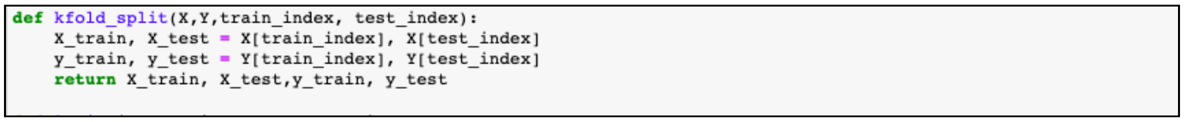


```
model_acc_lstm = []
i = 1
for train, test in kf.split(X):
    X_train, X_test = X[train], X[test]
    y_train, y_test = Y[train], Y[test]
  # create model
    model = Sequential()
    model.add(LSTM(200, activation='relu',input_shape=(1,16)))
    model.add(Dense(100, activation='relu'))
    model.add(Dense(50, activation='relu'))
    model.add(Dense(25, activation='sigmoid'))
    model.add(Dense(1))
    # Compile model
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    # Fit the model
    X_train_new = X_train.reshape((X_train.shape[0],1, X_train.shape[1]))
    X_test_new = X_test.reshape((X_test.shape[0],1, X_test.shape[1]))
    model.fit(X_train_new,y_train, epochs = 100, batch_size = 32, verbose=0)
    # predict on the model
    predval = model.predict(X_test_new).flatten()
    predval_new = np.where(predval > 0.5, 1, 0)
    #Evalute the model
    metric_lstm = c_matrix(y_test, predval_new, 'LSTM', i)
    model_acc_lstm.append(metric_lstm)
    i += 1


LSTM_metrics = pd.DataFrame(model_acc_lstm, columns =['model','fold','Accuracy','Sensitivity', 'Specificity', 'Precisic
LSTM_metrics.loc['Mean'] = LSTM_metrics.mean()
```

Figure 4: Code for LSTM

## EVALUATION METRICS

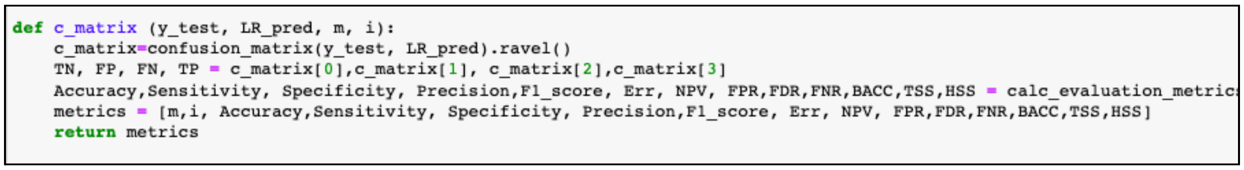
14) The evaluation metrics used is 10 fold cross validation.: The 10 fold are created using the KFold function in sklearn.



```python
def kfold_split(X,Y,train_index, test_index):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = Y[train_index], Y[test_index]
    return X_train, X_test,y_train, y_test
```

Figure 5: Function for creating X and Y training data

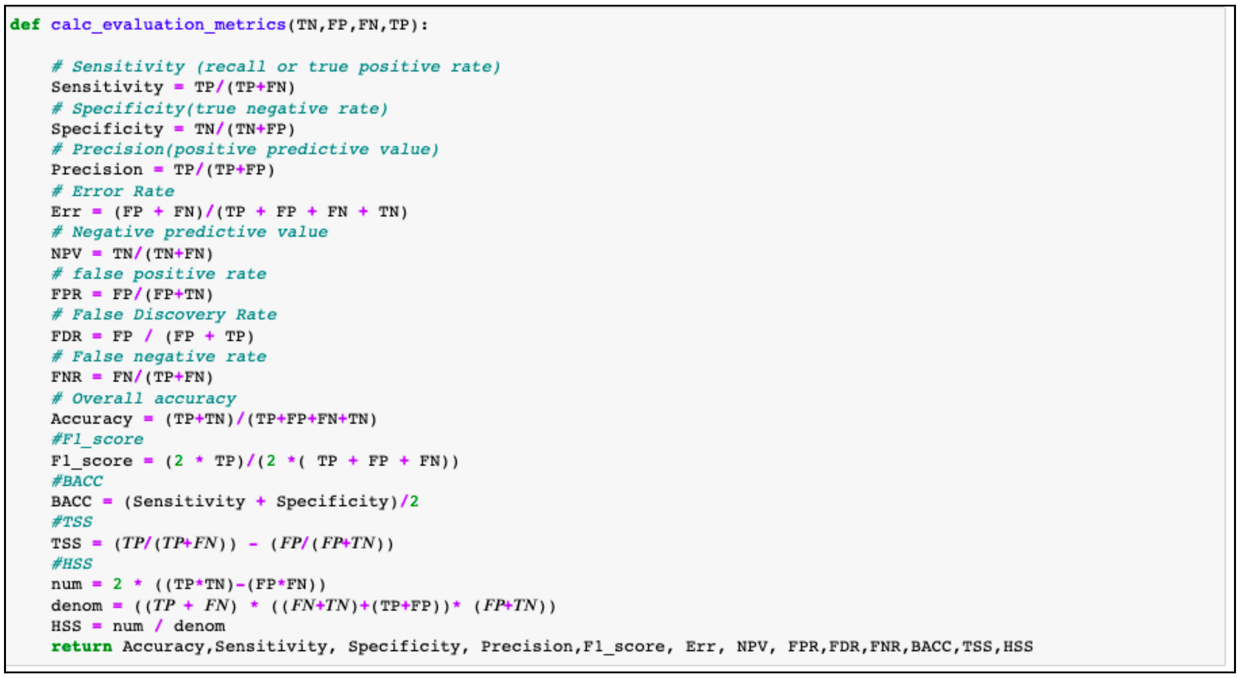
15) For calculating metrics, true positives(TP) ,true negative(TN) , False negative(FN) and False positive(FP) are calculated using a confusion matrix. (Function used: c_matrix)



```python
def c_matrix (y_test, LR_pred, m, i):
    c_matrix=confusion_matrix(y_test, LR_pred).ravel()
    TN, FP, FN, TP = c_matrix[0],c_matrix[1], c_matrix[2],c_matrix[3]
    Accuracy,Sensitivity, Specificity, Precision,F1_score, Err, NPV, FPR,FDR,FNR,BACC,TSS,HSS = calc_evaluation_metric
    metrics = [m,i, Accuracy,Sensitivity, Specificity, Precision,F1_score, Err, NPV, FPR,FDR,FNR,BACC,TSS,HSS]
    return metrics
```

Figure 6: Function for creating confusion matrix and obtaining TP, TN, FN, FP

16) To calculate Accuracy, Sensitivity, Specificity, Precision, F1_score, Error rate, Negative predictive value, False positive rate, False Discovery Rate, False negative rate, Balanced Accuracy(BACC), True Skills Statistics(TSS), Heidke Skill Score (HSS), formulae and function in figure 7 are used.



```python
def calc_evaluation_metrics(TN,FP,FN,TP):

    # Sensitivity (recall or true positive rate)
    Sensitivity = TP/(TP+FN)
    # Specificity(true negative rate)
    Specificity = TN/(TN+FP)
    # Precision(positive predictive value)
    Precision = TP/(TP+FP)
    # Error Rate
    Err = (FP + FN)/(TP + FP + FN + TN)
    # Negative predictive value
    NPV = TN/(TN+FN)
    # false positive rate
    FPR = FP/(FP+TN)
    # False Discovery Rate
    FDR = FP / (FP + TP)
    # False negative rate
    FNR = FN/(TP+FN)
    # Overall accuracy
    Accuracy = (TP+TN)/(TP+FP+FN+TN)
    #F1_score
    F1_score = (2 * TP)/(2 *( TP + FP + FN))
    #BACC
    BACC = (Sensitivity + Specificity)/2
    #TSS
    TSS = (TP/(TP+FN)) - (FP/(FP+TN))
    #HSS
    num = 2 * ((TP*TN)-(FP*FN))
    denom = ((TP + FN) * ((FN+TN)+(TP+FP))* (FP+TN))
    HSS = num / denom
    return Accuracy,Sensitivity, Specificity, Precision,F1_score, Err, NPV, FPR,FDR,FNR,BACC,TSS,HSS
```

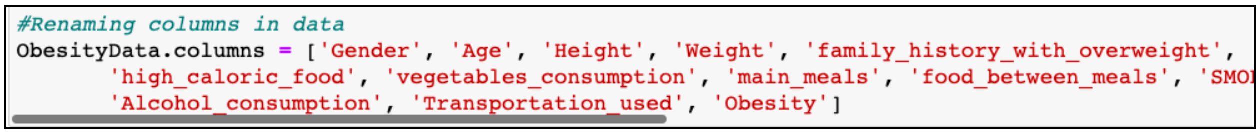
Figure 7: Function for calculating metrics value

## IMPLEMENTATION OF CODE AND OUTPUT

1. **Loading dataset**: Data set is loaded using csv reader using pandas dataframe

### 2. Exploratory Data analysis

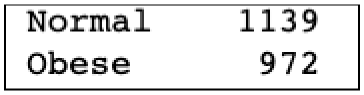
a. For better understanding, columns are renamed according to table 2.



```
#Renaming columns in data
ObesityData.columns = ['Gender', 'Age', 'Height', 'Weight', 'family_history_with_overweight',
        'high_caloric_food', 'vegetables_consumption', 'main_meals', 'food_between_meals', 'SMOI
        'Alcohol_consumption', 'Transportation_used', 'Obesity']
```

Figure 8: Renaming columns

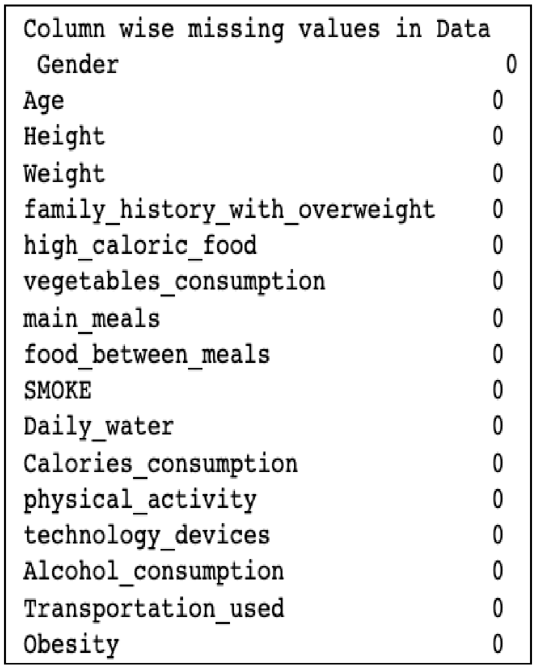
b. converting classes within dataset to binary: The distribution of each class with the labels shows that the data is not balanced since 1139 records belong to 'Normal' class and 972 to 'Obese' class and their ratio is ~1.17



```
Normal      1139
Obese        972
```

Figure 9: value counts for label

c. Checking for null values: No null values seen from the figure 10, 11. Uniform colour in heatmap indicated no null/ missing values in the data



```
Column wise missing values in Data
 Gender                            0
Age                               0
Height                            0
Weight                            0
family_history_with_overweight    0
high_caloric_food                 0
vegetables_consumption            0
main_meals                        0
food_between_meals                0
SMOKE                             0
Daily_water                       0
Calories_consumption              0
physical_activity                 0
technology_devices                0
Alcohol_consumption               0
Transportation_used               0
Obesity                           0
```

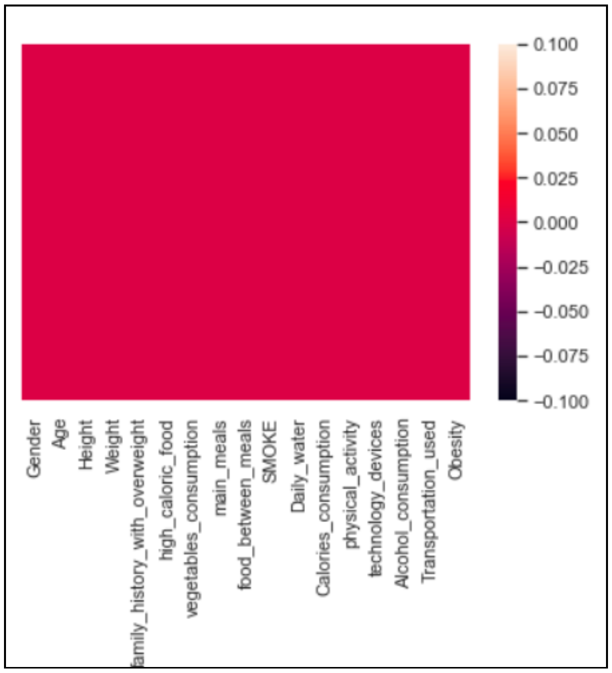
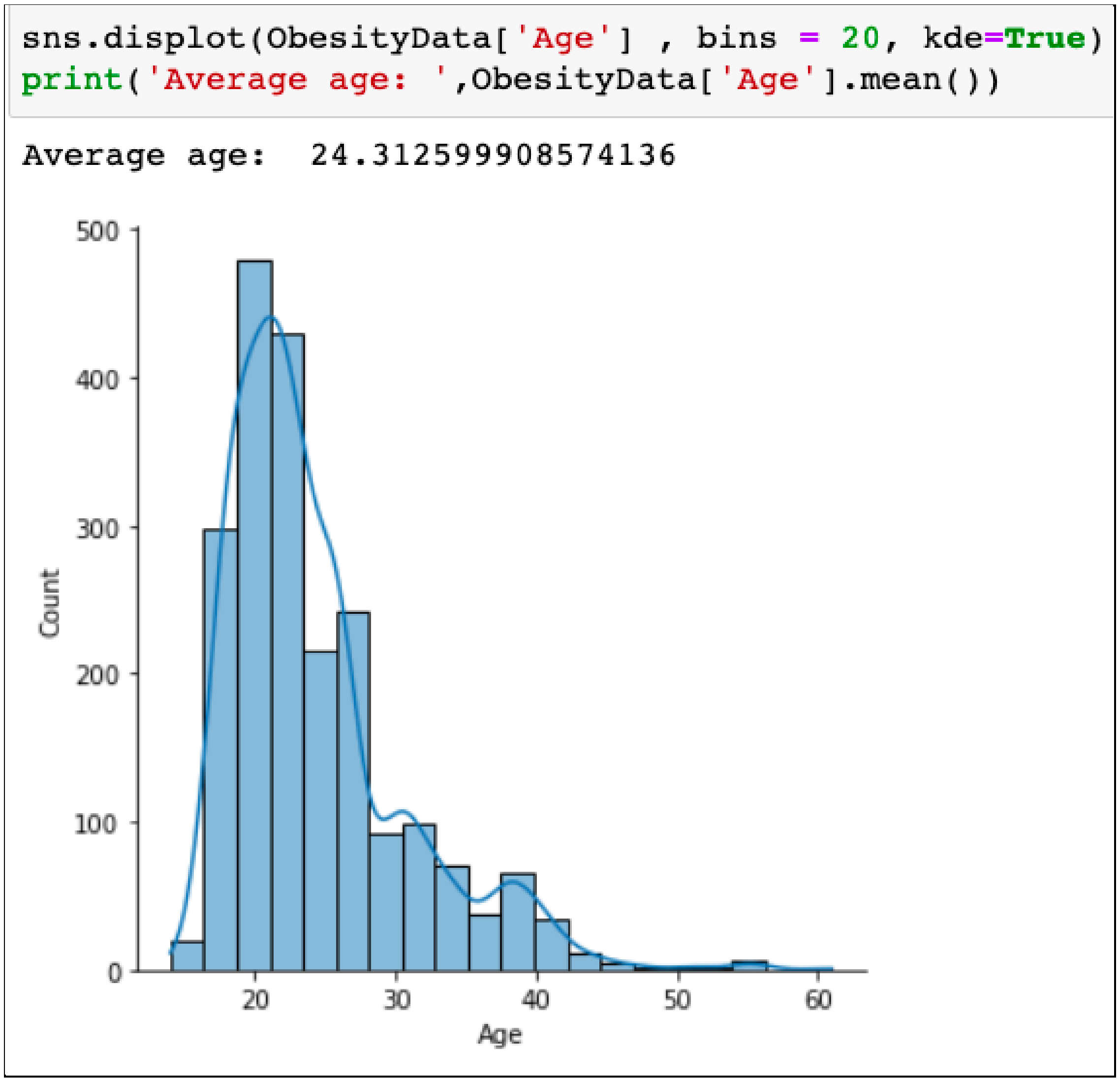


Figure 10 and 11: Checking null values (Right) column wise and (left) plotting results using heatmap

d. Age group of people in Dataset: The Age group of most of the individuals in this study is 15 to 28 years with average age of 24 years

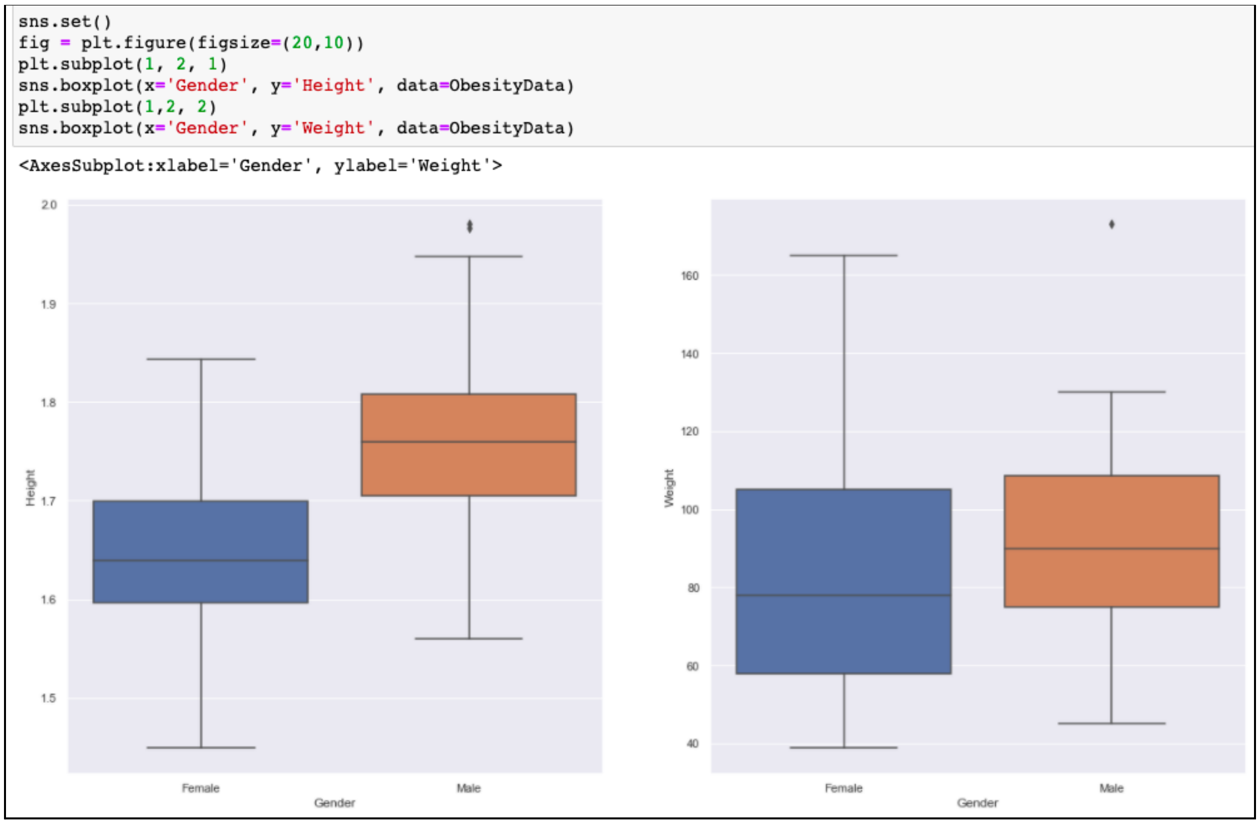


```
sns.displot(ObesityData['Age'] , bins = 20, kde=True)
print('Average age: ',ObesityData['Age'].mean())
```

```
Average age:  24.312599908574136
```

Figure 12: plot for age groups of individuals in data

e. Average height and weight for the males and females: The box plots show that average height for males is more than female and Average weight of males is more than that of females.

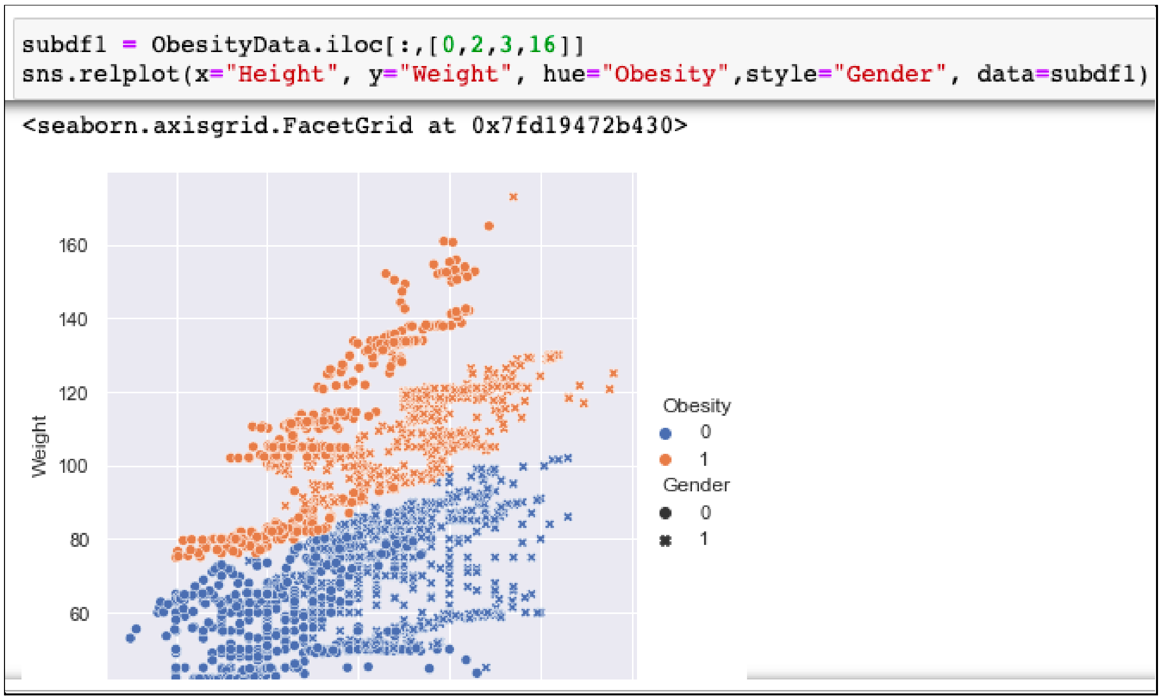


```
sns.set()
fig = plt.figure(figsize=(20,10))
plt.subplot(1, 2, 1)
sns.boxplot(x='Gender', y='Height', data=ObesityData)
plt.subplot(1,2, 2)
sns.boxplot(x='Gender', y='Weight', data=ObesityData)
```

```
<AxesSubplot:xlabel='Gender', ylabel='Weight'>
```

Figure 13: Average height and weight for the males and females

f. Relation plot for weight, height, genders and obesity :From the plot below we can say that people with higher weights tend to be more obese and obesity is determined by ratio of height and weight. ( Check point 4 in exploratory data analysis)



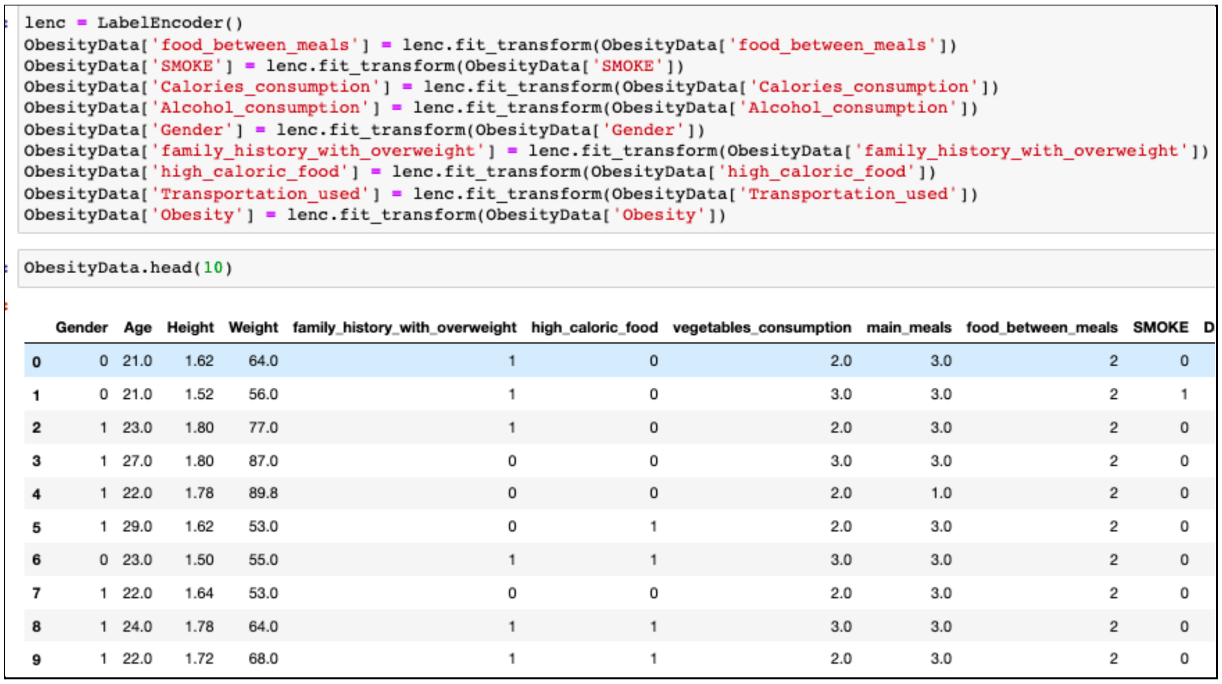
```
subdf1 = ObesityData.iloc[:,[0,2,3,16]]
sns.relplot(x="Height", y="Weight", hue="Obesity",style="Gender", data=subdf1)
```

```
<seaborn.axisgrid.FacetGrid at 0x7fd19472b430>
```

Figure 14: Average height and weight for the males and females

## 3. Data Preprocessing:

### a. Label Encoding



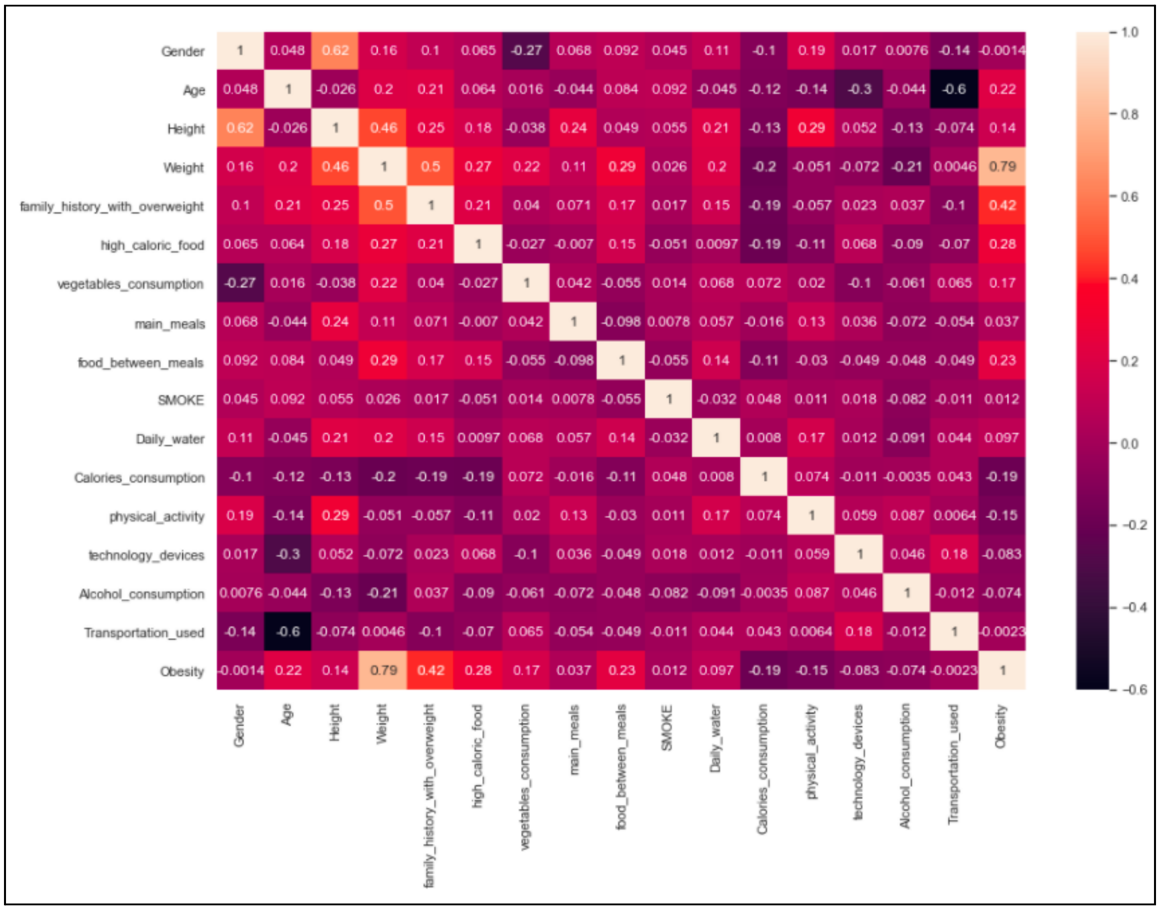
```
lenc = LabelEncoder()
ObesityData['food_between_meals'] = lenc.fit_transform(ObesityData['food_between_meals'])
ObesityData['SMOKE'] = lenc.fit_transform(ObesityData['SMOKE'])
ObesityData['Calories_consumption'] = lenc.fit_transform(ObesityData['Calories_consumption'])
ObesityData['Alcohol_consumption'] = lenc.fit_transform(ObesityData['Alcohol_consumption'])
ObesityData['Gender'] = lenc.fit_transform(ObesityData['Gender'])
ObesityData['family_history_with_overweight'] = lenc.fit_transform(ObesityData['family_history_with_overweight'])
ObesityData['high_caloric_food'] = lenc.fit_transform(ObesityData['high_caloric_food'])
ObesityData['Transportation_used'] = lenc.fit_transform(ObesityData['Transportation_used'])
ObesityData['Obesity'] = lenc.fit_transform(ObesityData['Obesity'])
```

```
ObesityData.head(10)
```

| | Gender | Age | Height | Weight | family_history_with_overweight | high_caloric_food | vegetables_consumption | main_meals | food_between_meals | SMOKE | D |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 21.0 | 1.62 | 64.0 | 1 | 0 | 2.0 | 3.0 | 2 | 0 | |
| 1 | 0 | 21.0 | 1.52 | 56.0 | 1 | 0 | 3.0 | 3.0 | 2 | 1 | |
| 2 | 1 | 23.0 | 1.80 | 77.0 | 1 | 0 | 2.0 | 3.0 | 2 | 0 | |
| 3 | 1 | 27.0 | 1.80 | 87.0 | 0 | 0 | 3.0 | 3.0 | 2 | 0 | |
| 4 | 1 | 22.0 | 1.78 | 89.8 | 0 | 0 | 2.0 | 1.0 | 2 | 0 | |
| 5 | 1 | 29.0 | 1.62 | 53.0 | 0 | 1 | 2.0 | 3.0 | 2 | 0 | |
| 6 | 0 | 23.0 | 1.50 | 55.0 | 1 | 1 | 3.0 | 3.0 | 2 | 0 | |
| 7 | 1 | 22.0 | 1.64 | 53.0 | 0 | 0 | 2.0 | 3.0 | 2 | 0 | |
| 8 | 1 | 24.0 | 1.78 | 64.0 | 1 | 1 | 3.0 | 3.0 | 2 | 0 | |
| 9 | 1 | 22.0 | 1.72 | 68.0 | 1 | 1 | 2.0 | 3.0 | 2 | 0 | |

Figure 15: label encoding

b. Correlation between features: As we can see from the correlation matrix in figure 16, No two features are highly correlated



| | Gender | Age | Height | Weight | family_history_with_overweight | high_caloric_food | vegetables_consumption | main_meals | food_between_meals | SMOKE | Daily_water | Calories_consumption | physical_activity | technology_devices | Alcohol_consumption | Transportation_used | Obesity |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Gender | 1 | 0.048 | 0.62 | 0.16 | 0.1 | 0.065 | -0.27 | 0.068 | 0.092 | 0.045 | 0.11 | -0.1 | 0.19 | 0.017 | 0.0076 | -0.14 | -0.0014 |
| Age | 0.048 | 1 | -0.026 | 0.2 | 0.21 | 0.064 | 0.016 | -0.044 | 0.084 | 0.092 | -0.045 | -0.12 | -0.14 | -0.3 | -0.044 | -0.6 | 0.22 |
| Height | 0.62 | -0.026 | 1 | 0.46 | 0.25 | 0.18 | -0.038 | 0.24 | 0.049 | 0.055 | 0.21 | -0.13 | 0.29 | 0.052 | -0.13 | -0.074 | 0.14 |
| Weight | 0.16 | 0.2 | 0.46 | 1 | 0.5 | 0.27 | 0.22 | 0.11 | 0.29 | 0.026 | 0.2 | -0.2 | -0.051 | -0.072 | -0.21 | 0.0046 | 0.79 |
| family_history_with_overweight | 0.1 | 0.21 | 0.25 | 0.5 | 1 | 0.21 | 0.04 | 0.071 | 0.17 | 0.017 | 0.15 | -0.19 | -0.057 | 0.023 | 0.037 | -0.1 | 0.42 |
| high_caloric_food | 0.065 | 0.064 | 0.18 | 0.27 | 0.21 | 1 | -0.027 | -0.007 | 0.15 | -0.051 | 0.0097 | -0.19 | -0.11 | 0.068 | -0.09 | -0.07 | 0.28 |
| vegetables_consumption | -0.27 | 0.016 | -0.038 | 0.22 | 0.04 | -0.027 | 1 | 0.042 | -0.055 | 0.014 | 0.068 | 0.072 | 0.02 | -0.1 | -0.061 | 0.065 | 0.17 |
| main_meals | 0.068 | -0.044 | 0.24 | 0.11 | 0.071 | -0.007 | 0.042 | 1 | -0.098 | 0.0078 | 0.057 | -0.016 | 0.13 | 0.036 | -0.072 | -0.054 | 0.037 |
| food_between_meals | 0.092 | 0.084 | 0.049 | 0.29 | 0.17 | 0.15 | -0.055 | -0.098 | 1 | -0.055 | 0.14 | -0.11 | -0.03 | -0.049 | -0.048 | -0.049 | 0.23 |
| SMOKE | 0.045 | 0.092 | 0.055 | 0.026 | 0.017 | -0.051 | 0.014 | 0.0078 | -0.055 | 1 | -0.032 | 0.048 | 0.011 | 0.018 | -0.082 | -0.011 | 0.012 |
| Daily_water | 0.11 | -0.045 | 0.21 | 0.2 | 0.15 | 0.0097 | 0.068 | 0.057 | 0.14 | -0.032 | 1 | 0.008 | 0.17 | 0.012 | -0.091 | 0.044 | 0.097 |
| Calories_consumption | -0.1 | -0.12 | -0.13 | -0.2 | -0.19 | -0.19 | 0.072 | -0.016 | -0.11 | 0.048 | 0.008 | 1 | 0.074 | -0.011 | -0.0035 | 0.043 | -0.19 |
| physical_activity | 0.19 | -0.14 | 0.29 | -0.051 | -0.057 | -0.11 | 0.02 | 0.13 | -0.03 | 0.011 | 0.17 | 0.074 | 1 | 0.059 | 0.087 | 0.0064 | -0.15 |
| technology_devices | 0.017 | -0.3 | 0.052 | -0.072 | 0.023 | 0.068 | -0.1 | 0.036 | -0.049 | 0.018 | 0.012 | -0.011 | 0.059 | 1 | 0.046 | 0.18 | -0.083 |
| Alcohol_consumption | 0.0076 | -0.044 | -0.13 | -0.21 | 0.037 | -0.09 | -0.061 | -0.072 | -0.048 | -0.082 | -0.091 | -0.0035 | 0.087 | 0.046 | 1 | -0.012 | -0.074 |
| Transportation_used | -0.14 | -0.6 | -0.074 | 0.0046 | -0.1 | -0.07 | 0.065 | -0.054 | -0.049 | -0.011 | 0.044 | 0.043 | 0.0064 | 0.18 | -0.012 | 1 | -0.0023 |
| Obesity | -0.0014 | 0.22 | 0.14 | 0.79 | 0.42 | 0.28 | 0.17 | 0.037 | 0.23 | 0.012 | 0.097 | -0.19 | -0.15 | -0.083 | -0.074 | -0.0023 | 1 |

Figure 16: heatmap for correlation matrix between features

c. Splitting the data into features(X) and Labels(Y) : Data is split into X and Y for further training and testing.

```
X_n = ObesityData[['Gender', 'Age', 'Height', 'Weight', 'family_history_with_overweight',
        'high_caloric_food', 'vegetables_consumption', 'main_meals', 'food_between_meals', 'SMOKE', 'Daily_water', 'Cal
        'Alcohol_consumption','Transportation_used']].values
Y = ObesityData['Obesity']
```

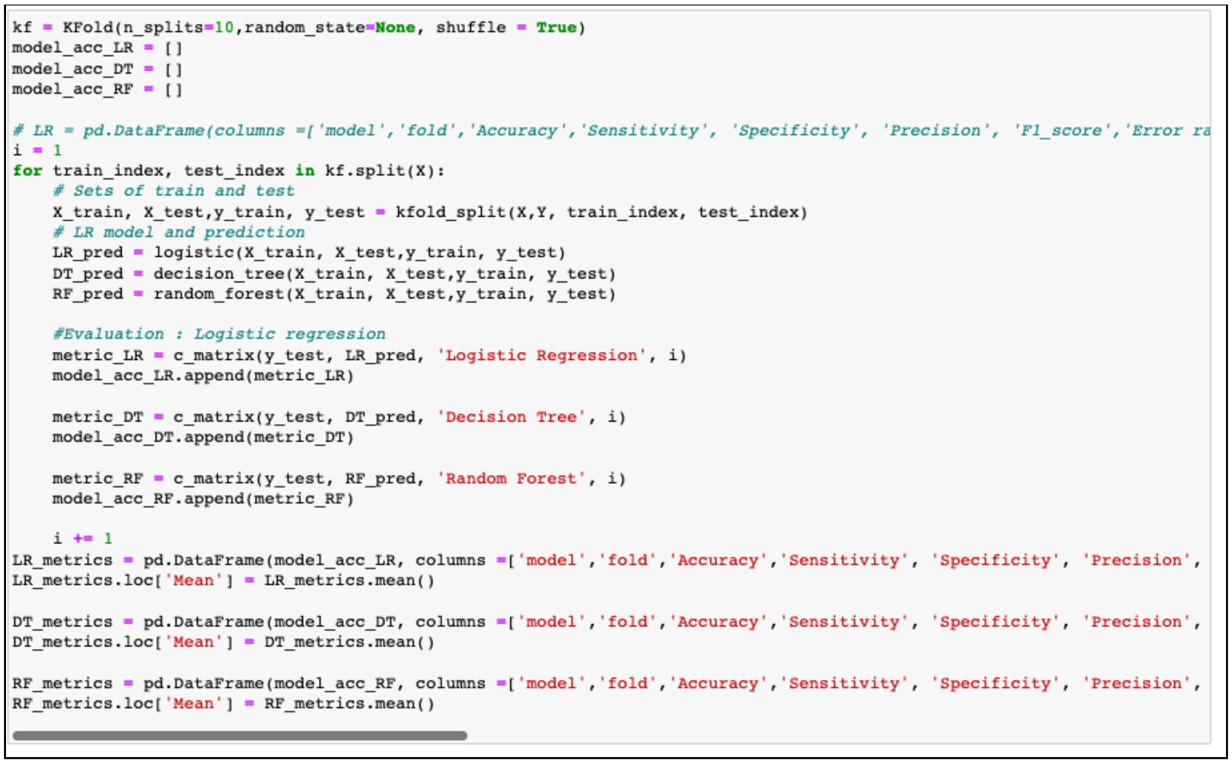
Figure 17: splitting data into features and labels

d. Normalization of Data: converted all feature values between 0 and 1

```
#returns a numpy array with normalized values for X
min_max_scaler = preprocessing.MinMaxScaler()
X = min_max_scaler.fit_transform(X_n)
```

Figure 18: Normalizing data

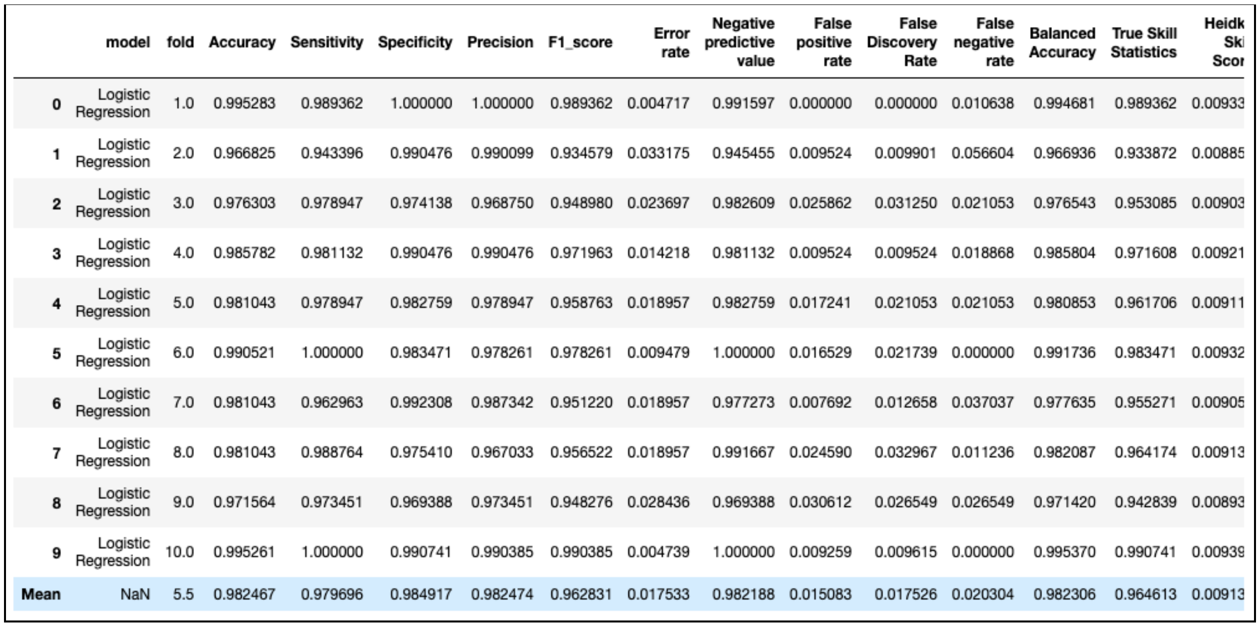
4. **Machine Learning models:** Three machine learning models are trained and tested on Obesity Data using k-fold cross validation with the number of folds being 10.



```
kf = KFold(n_splits=10,random_state=None, shuffle = True)
model_acc_LR = []
model_acc_DT = []
model_acc_RF = []

# LR = pd.DataFrame(columns =['model','fold','Accuracy','Sensitivity', 'Specificity', 'Precision', 'F1_score','Error ra
i = 1
for train_index, test_index in kf.split(X):
    # Sets of train and test
    X_train, X_test,y_train, y_test = kfold_split(X,Y, train_index, test_index)
    # LR model and prediction
    LR_pred = logistic(X_train, X_test,y_train, y_test)
    DT_pred = decision_tree(X_train, X_test,y_train, y_test)
    RF_pred = random_forest(X_train, X_test,y_train, y_test)

    #Evaluation : Logistic regression
    metric_LR = c_matrix(y_test, LR_pred, 'Logistic Regression', i)
    model_acc_LR.append(metric_LR)

    metric_DT = c_matrix(y_test, DT_pred, 'Decision Tree', i)
    model_acc_DT.append(metric_DT)

    metric_RF = c_matrix(y_test, RF_pred, 'Random Forest', i)
    model_acc_RF.append(metric_RF)

    i += 1
LR_metrics = pd.DataFrame(model_acc_LR, columns =['model','fold','Accuracy','Sensitivity', 'Specificity', 'Precision',
LR_metrics.loc['Mean'] = LR_metrics.mean()

DT_metrics = pd.DataFrame(model_acc_DT, columns =['model','fold','Accuracy','Sensitivity', 'Specificity', 'Precision',
DT_metrics.loc['Mean'] = DT_metrics.mean()

RF_metrics = pd.DataFrame(model_acc_RF, columns =['model','fold','Accuracy','Sensitivity', 'Specificity', 'Precision',
RF_metrics.loc['Mean'] = RF_metrics.mean()
```

Figure 19: code for Machine learning models on obesity data

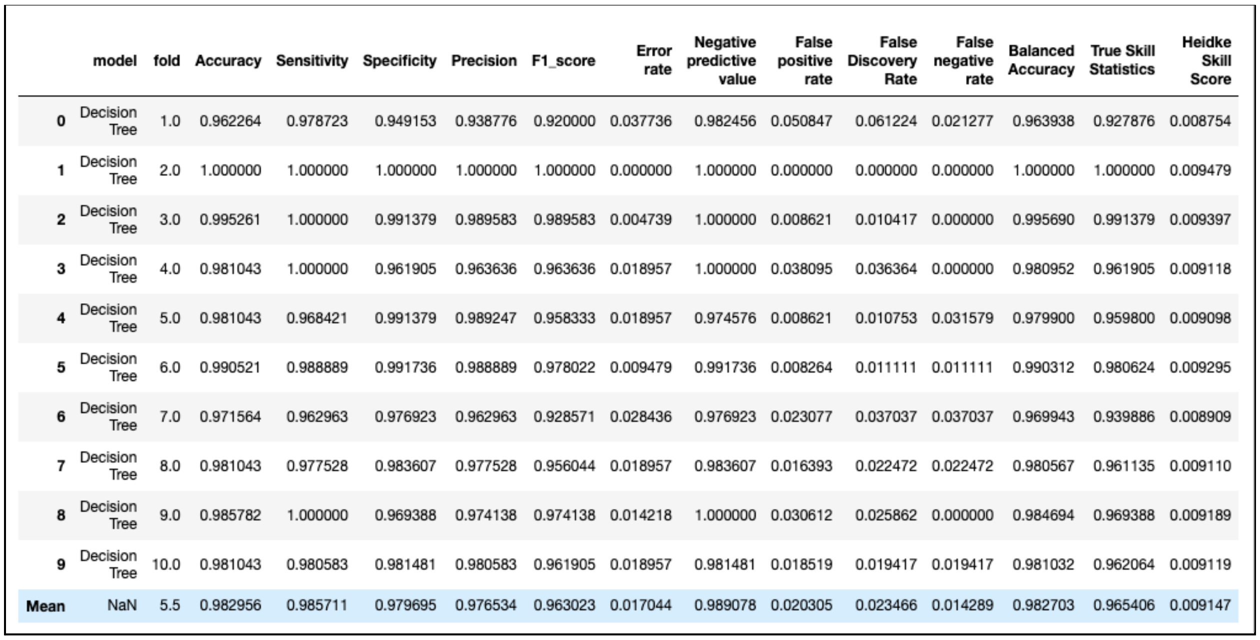
a. Table for Logistic regression with overall accuracy of **98.24 %**



| | model | fold | Accuracy | Sensitivity | Specificity | Precision | F1_score | Error rate | Negative predictive value | False positive rate | False Discovery Rate | False negative rate | Balanced Accuracy | True Skill Statistics | Heidk Ski Scor |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Logistic Regression | 1.0 | 0.995283 | 0.989362 | 1.000000 | 1.000000 | 0.989362 | 0.004717 | 0.991597 | 0.000000 | 0.000000 | 0.010638 | 0.994681 | 0.989362 | 0.00933 |
| **1** | Logistic Regression | 2.0 | 0.966825 | 0.943396 | 0.990476 | 0.990099 | 0.934579 | 0.033175 | 0.945455 | 0.009524 | 0.009901 | 0.056604 | 0.966936 | 0.933872 | 0.00885 |
| **2** | Logistic Regression | 3.0 | 0.976303 | 0.978947 | 0.974138 | 0.968750 | 0.948980 | 0.023697 | 0.982609 | 0.025862 | 0.031250 | 0.021053 | 0.976543 | 0.953085 | 0.00903 |
| **3** | Logistic Regression | 4.0 | 0.985782 | 0.981132 | 0.990476 | 0.990476 | 0.971963 | 0.014218 | 0.981132 | 0.009524 | 0.009524 | 0.018868 | 0.985804 | 0.971608 | 0.00921 |
| **4** | Logistic Regression | 5.0 | 0.981043 | 0.978947 | 0.982759 | 0.978947 | 0.958763 | 0.018957 | 0.982759 | 0.017241 | 0.021053 | 0.021053 | 0.980853 | 0.961706 | 0.00911 |
| **5** | Logistic Regression | 6.0 | 0.990521 | 1.000000 | 0.983471 | 0.978261 | 0.978261 | 0.009479 | 1.000000 | 0.016529 | 0.021739 | 0.000000 | 0.991736 | 0.983471 | 0.00932 |
| **6** | Logistic Regression | 7.0 | 0.981043 | 0.962963 | 0.992308 | 0.987342 | 0.951220 | 0.018957 | 0.977273 | 0.007692 | 0.012658 | 0.037037 | 0.977635 | 0.955271 | 0.00905 |
| **7** | Logistic Regression | 8.0 | 0.981043 | 0.988764 | 0.975410 | 0.967033 | 0.956522 | 0.018957 | 0.991667 | 0.024590 | 0.032967 | 0.011236 | 0.982087 | 0.964174 | 0.00913 |
| **8** | Logistic Regression | 9.0 | 0.971564 | 0.973451 | 0.969388 | 0.973451 | 0.948276 | 0.028436 | 0.969388 | 0.030612 | 0.026549 | 0.026549 | 0.971420 | 0.942839 | 0.00893 |
| **9** | Logistic Regression | 10.0 | 0.995261 | 1.000000 | 0.990741 | 0.990385 | 0.990385 | 0.004739 | 1.000000 | 0.009259 | 0.009615 | 0.000000 | 0.995370 | 0.990741 | 0.00939 |
| **Mean** | NaN | 5.5 | 0.982467 | 0.979696 | 0.984917 | 0.982474 | 0.962831 | 0.017533 | 0.982188 | 0.015083 | 0.017526 | 0.020304 | 0.982306 | 0.964613 | 0.00913 |

Figure 20: Table with all metrics and their average for logistic regression

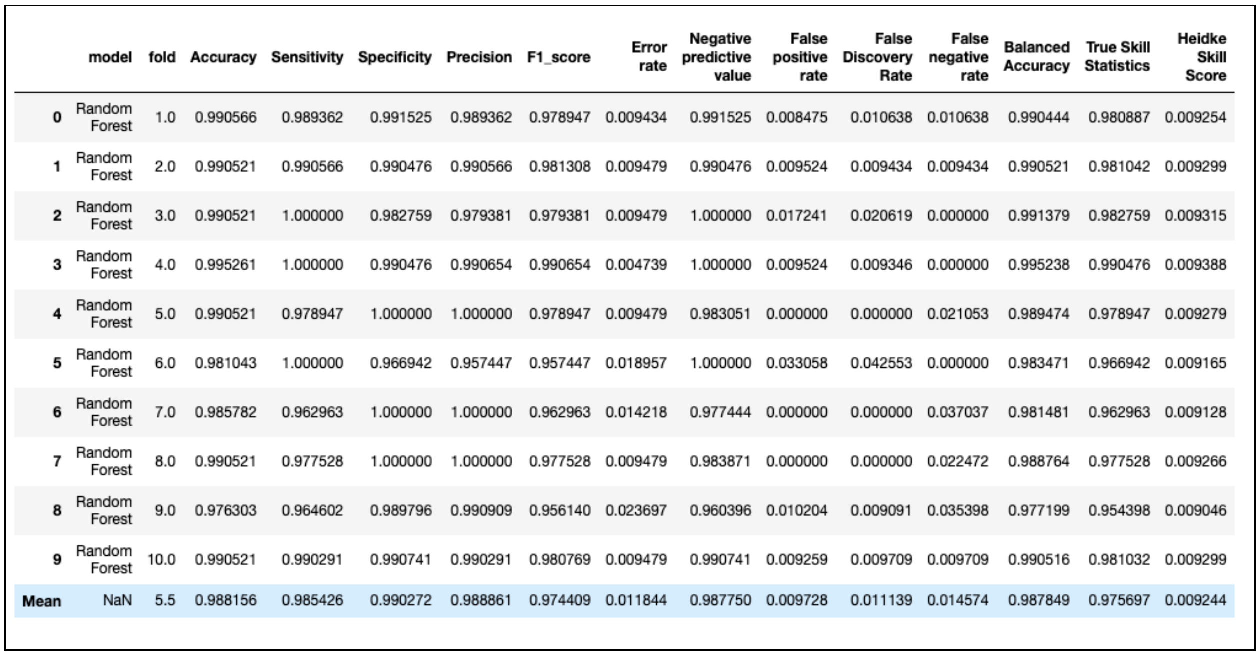
b. Table for Decision Tree with overall accuracy of **98.29 %**



| | model | fold | Accuracy | Sensitivity | Specificity | Precision | F1_score | Error rate | Negative predictive value | False positive rate | False Discovery Rate | False negative rate | Balanced Accuracy | True Skill Statistics | Heidke Skill Score |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Decision Tree | 1.0 | 0.962264 | 0.978723 | 0.949153 | 0.938776 | 0.920000 | 0.037736 | 0.982456 | 0.050847 | 0.061224 | 0.021277 | 0.963938 | 0.927876 | 0.008754 |
| 1 | Decision Tree | 2.0 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 0.000000 | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 1.000000 | 0.009479 |
| 2 | Decision Tree | 3.0 | 0.995261 | 1.000000 | 0.991379 | 0.989583 | 0.989583 | 0.004739 | 1.000000 | 0.008621 | 0.010417 | 0.000000 | 0.995690 | 0.991379 | 0.009397 |
| 3 | Decision Tree | 4.0 | 0.981043 | 1.000000 | 0.961905 | 0.963636 | 0.963636 | 0.018957 | 1.000000 | 0.038095 | 0.036364 | 0.000000 | 0.980952 | 0.961905 | 0.009118 |
| 4 | Decision Tree | 5.0 | 0.981043 | 0.968421 | 0.991379 | 0.989247 | 0.958333 | 0.018957 | 0.974576 | 0.008621 | 0.010753 | 0.031579 | 0.979900 | 0.959800 | 0.009098 |
| 5 | Decision Tree | 6.0 | 0.990521 | 0.988889 | 0.991736 | 0.988889 | 0.978022 | 0.009479 | 0.991736 | 0.008264 | 0.011111 | 0.011111 | 0.990312 | 0.980624 | 0.009295 |
| 6 | Decision Tree | 7.0 | 0.971564 | 0.962963 | 0.976923 | 0.962963 | 0.928571 | 0.028436 | 0.976923 | 0.023077 | 0.037037 | 0.037037 | 0.969943 | 0.939886 | 0.008909 |
| 7 | Decision Tree | 8.0 | 0.981043 | 0.977528 | 0.983607 | 0.977528 | 0.956044 | 0.018957 | 0.983607 | 0.016393 | 0.022472 | 0.022472 | 0.980567 | 0.961135 | 0.009110 |
| 8 | Decision Tree | 9.0 | 0.985782 | 1.000000 | 0.969388 | 0.974138 | 0.974138 | 0.014218 | 1.000000 | 0.030612 | 0.025862 | 0.000000 | 0.984694 | 0.969388 | 0.009189 |
| 9 | Decision Tree | 10.0 | 0.981043 | 0.980583 | 0.981481 | 0.980583 | 0.961905 | 0.018957 | 0.981481 | 0.018519 | 0.019417 | 0.019417 | 0.981032 | 0.962064 | 0.009119 |
| **Mean** | NaN | 5.5 | 0.982956 | 0.985711 | 0.979695 | 0.976534 | 0.963023 | 0.017044 | 0.989078 | 0.020305 | 0.023466 | 0.014289 | 0.982703 | 0.965406 | 0.009147 |

Figure 21: Table with all metrics and their average for decision tree

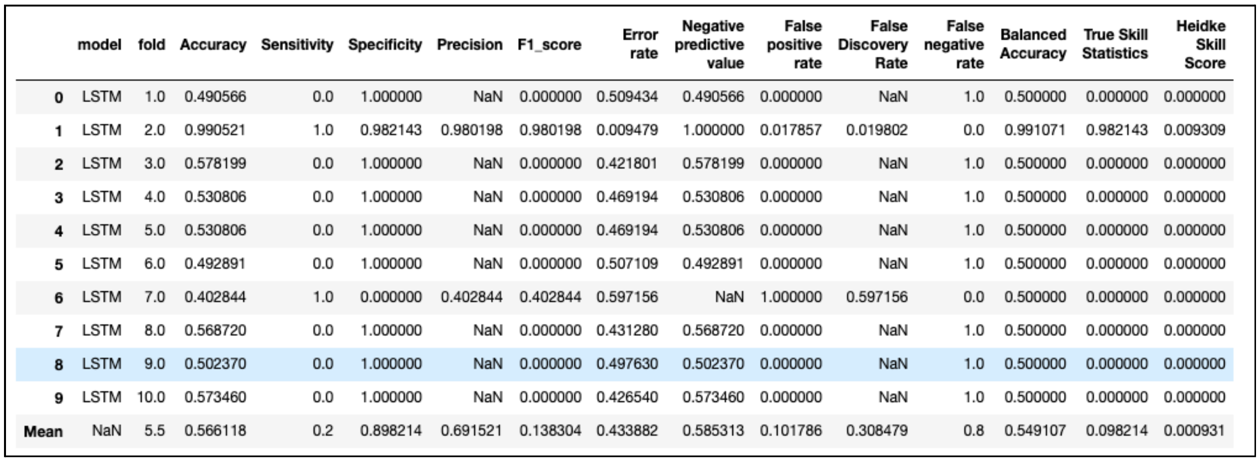
c. Table for Random Forest regression with overall accuracy of **98.88 %**



| | model | fold | Accuracy | Sensitivity | Specificity | Precision | F1_score | Error rate | Negative predictive value | False positive rate | False Discovery Rate | False negative rate | Balanced Accuracy | True Skill Statistics | Heidke Skill Score |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Random Forest | 1.0 | 0.990566 | 0.989362 | 0.991525 | 0.989362 | 0.978947 | 0.009434 | 0.991525 | 0.008475 | 0.010638 | 0.010638 | 0.990444 | 0.980887 | 0.009254 |
| 1 | Random Forest | 2.0 | 0.990521 | 0.990566 | 0.990476 | 0.990566 | 0.981308 | 0.009479 | 0.990476 | 0.009524 | 0.009434 | 0.009434 | 0.990521 | 0.981042 | 0.009299 |
| 2 | Random Forest | 3.0 | 0.990521 | 1.000000 | 0.982759 | 0.979381 | 0.979381 | 0.009479 | 1.000000 | 0.017241 | 0.020619 | 0.000000 | 0.991379 | 0.982759 | 0.009315 |
| 3 | Random Forest | 4.0 | 0.995261 | 1.000000 | 0.990476 | 0.990654 | 0.990654 | 0.004739 | 1.000000 | 0.009524 | 0.009346 | 0.000000 | 0.995238 | 0.990476 | 0.009388 |
| 4 | Random Forest | 5.0 | 0.990521 | 0.978947 | 1.000000 | 1.000000 | 0.978947 | 0.009479 | 0.983051 | 0.000000 | 0.000000 | 0.021053 | 0.989474 | 0.978947 | 0.009279 |
| 5 | Random Forest | 6.0 | 0.981043 | 1.000000 | 0.966942 | 0.957447 | 0.957447 | 0.018957 | 1.000000 | 0.033058 | 0.042553 | 0.000000 | 0.983471 | 0.966942 | 0.009165 |
| 6 | Random Forest | 7.0 | 0.985782 | 0.962963 | 1.000000 | 1.000000 | 0.962963 | 0.014218 | 0.977444 | 0.000000 | 0.000000 | 0.037037 | 0.981481 | 0.962963 | 0.009128 |
| 7 | Random Forest | 8.0 | 0.990521 | 0.977528 | 1.000000 | 1.000000 | 0.977528 | 0.009479 | 0.983871 | 0.000000 | 0.000000 | 0.022472 | 0.988764 | 0.977528 | 0.009266 |
| 8 | Random Forest | 9.0 | 0.976303 | 0.964602 | 0.989796 | 0.990909 | 0.956140 | 0.023697 | 0.960396 | 0.010204 | 0.009091 | 0.035398 | 0.977199 | 0.954398 | 0.009046 |
| 9 | Random Forest | 10.0 | 0.990521 | 0.990291 | 0.990741 | 0.990291 | 0.980769 | 0.009479 | 0.990741 | 0.009259 | 0.009709 | 0.009709 | 0.990516 | 0.981032 | 0.009299 |
| **Mean** | NaN | 5.5 | 0.988156 | 0.985426 | 0.990272 | 0.988861 | 0.974409 | 0.011844 | 0.987750 | 0.009728 | 0.011139 | 0.014574 | 0.987849 | 0.975697 | 0.009244 |

Figure 22: Table with all metrics and their average for Random forest

## 5. LSTM model:

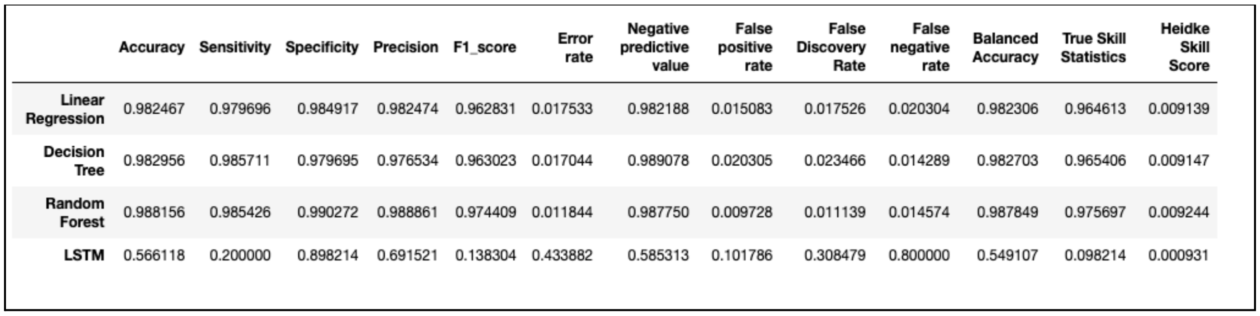


| | model | fold | Accuracy | Sensitivity | Specificity | Precision | F1_score | Error rate | Negative predictive value | False positive rate | False Discovery Rate | False negative rate | Balanced Accuracy | True Skill Statistics | Heidke Skill Score |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | LSTM | 1.0 | 0.490566 | 0.0 | 1.000000 | NaN | 0.000000 | 0.509434 | 0.490566 | 0.000000 | NaN | 1.0 | 0.500000 | 0.000000 | 0.000000 |
| **1** | LSTM | 2.0 | 0.990521 | 1.0 | 0.982143 | 0.980198 | 0.980198 | 0.009479 | 1.000000 | 0.017857 | 0.019802 | 0.0 | 0.991071 | 0.982143 | 0.009309 |
| **2** | LSTM | 3.0 | 0.578199 | 0.0 | 1.000000 | NaN | 0.000000 | 0.421801 | 0.578199 | 0.000000 | NaN | 1.0 | 0.500000 | 0.000000 | 0.000000 |
| **3** | LSTM | 4.0 | 0.530806 | 0.0 | 1.000000 | NaN | 0.000000 | 0.469194 | 0.530806 | 0.000000 | NaN | 1.0 | 0.500000 | 0.000000 | 0.000000 |
| **4** | LSTM | 5.0 | 0.530806 | 0.0 | 1.000000 | NaN | 0.000000 | 0.469194 | 0.530806 | 0.000000 | NaN | 1.0 | 0.500000 | 0.000000 | 0.000000 |
| **5** | LSTM | 6.0 | 0.492891 | 0.0 | 1.000000 | NaN | 0.000000 | 0.507109 | 0.492891 | 0.000000 | NaN | 1.0 | 0.500000 | 0.000000 | 0.000000 |
| **6** | LSTM | 7.0 | 0.402844 | 1.0 | 0.000000 | 0.402844 | 0.402844 | 0.597156 | NaN | 1.000000 | 0.597156 | 0.0 | 0.500000 | 0.000000 | 0.000000 |
| **7** | LSTM | 8.0 | 0.568720 | 0.0 | 1.000000 | NaN | 0.000000 | 0.431280 | 0.568720 | 0.000000 | NaN | 1.0 | 0.500000 | 0.000000 | 0.000000 |
| **8** | LSTM | 9.0 | 0.502370 | 0.0 | 1.000000 | NaN | 0.000000 | 0.497630 | 0.502370 | 0.000000 | NaN | 1.0 | 0.500000 | 0.000000 | 0.000000 |
| **9** | LSTM | 10.0 | 0.573460 | 0.0 | 1.000000 | NaN | 0.000000 | 0.426540 | 0.573460 | 0.000000 | NaN | 1.0 | 0.500000 | 0.000000 | 0.000000 |
| **Mean** | NaN | 5.5 | 0.566118 | 0.2 | 0.898214 | 0.691521 | 0.138304 | 0.433882 | 0.585313 | 0.101786 | 0.308479 | 0.8 | 0.549107 | 0.098214 | 0.000931 |

Figure 23: Table with all metrics and their average for LSTM

# EVALUATION OF MODELS

Average metrics for each of the classification models using 10-fold cross validation



| | Accuracy | Sensitivity | Specificity | Precision | F1_score | Error rate | Negative predictive value | False positive rate | False Discovery Rate | False negative rate | Balanced Accuracy | True Skill Statistics | Heidke Skill Score |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Linear Regression** | 0.982467 | 0.979696 | 0.984917 | 0.982474 | 0.962831 | 0.017533 | 0.982188 | 0.015083 | 0.017526 | 0.020304 | 0.982306 | 0.964613 | 0.009139 |
| **Decision Tree** | 0.982956 | 0.985711 | 0.979695 | 0.976534 | 0.963023 | 0.017044 | 0.989078 | 0.020305 | 0.023466 | 0.014289 | 0.982703 | 0.965406 | 0.009147 |
| **Random Forest** | 0.988156 | 0.985426 | 0.990272 | 0.988861 | 0.974409 | 0.011844 | 0.987750 | 0.009728 | 0.011139 | 0.014574 | 0.987849 | 0.975697 | 0.009244 |
| **LSTM** | 0.566118 | 0.200000 | 0.898214 | 0.691521 | 0.138304 | 0.433882 | 0.585313 | 0.101786 | 0.308479 | 0.800000 | 0.549107 | 0.098214 | 0.000931 |

Figure 24: Table with all metrics and their average for all models

# CONCLUSION

On comparing the accuracy, it is evident that Random forest outperforms all other models and therefore the best model for predicting obesity given this dataset.

The Random forest Algorithm performs better than all the other models, because random forest can handle classification tasks with all kinds of input features and also with minimal preprocessing.