****

**Machine Learning Assignment**

**Diabetes Analysis**

**Submitted By:**

|  |  |
| --- | --- |
| **Student Name** | **Student IT NO** |
| Kumarasinghe K.A.M.T | IT20088514 |
| Weesinghe W.M.P.D | IT20019204 |
| Arachchi O.B.K | IT20245238 |
| Fernando A.R.V.S | IT18047820 |

**Contents**

[**1.Introduction** 3](#_Toc132279242)

[1.1 Problem Statement 3](#_Toc132279243)

[**What is Diabetes?** 3](#_Toc132279244)

[**Symptoms** 3](#_Toc132279245)

[**2.Methodology** 4](#_Toc132279246)

[2.1 Data Collection 4](#_Toc132279247)

[2.2 Description of Dataset 5](#_Toc132279248)

[2.3 Algorithm Selection 5](#_Toc132279249)

[**3.Implementation** 6](#_Toc132279250)

[**4.Results and Discussions** 16](#_Toc132279251)

[4.1 Limitations 18](#_Toc132279243)

[4.2 Future Workt .](#_Toc132279243)19

[**5.References** 20](#_Toc132279252)

[**6.Apendix** 21](#_Toc132279253)

# **1.Introduction**

## 1.1 Problem Statement

### **What is Diabetes?**

Diabetes is a long-term (chronic) illness that affects how your body converts food into energy.

The majority of the food you consume is converted by your body into sugar (glucose), which is then released into your bloodstream. Your pancreas releases insulin when your blood sugar levels rise. In order for blood sugar to enter your body's cells and be used as energy, insulin functions like a key.

When you have diabetes, your body either produces insufficient insulin or uses it improperly. Too much blood sugar remains in your bloodstream when there is insufficient insulin or when cells cease reacting to insulin. That can eventually lead to major health issues like renal disease, eyesight loss, and heart disease.

Although there is currently no treatment for diabetes, decreasing weight, eating well, and exercising can all be very beneficial. You can also help by taking the medication as directed. Obtain guidance and information on diabetes self-management. Schedule and adhere to medical appointments.

### **Symptoms**

Diagram

Description automatically generated

Diabetes symptoms may appear suddenly. Type 2 diabetes symptoms can be subtle and may not become apparent for many years.

Diabetes symptoms include:

* having a lot of thirst, wanting to urinate more frequently than normal, and having blurry vision.
* being worn out and accidentally losing weight
* Diabetes over time can harm the blood vessels in the kidneys, eyes, heart, and nerves.
* Diabetes increases the risk of various illnesses, such as heart attack, stroke, and renal failure.
* Diabetes affects the blood vessels in the eyes, which can result in permanent vision loss.

Due to nerve damage and insufficient blood flow, diabetes affects the foot in many people. This may result in foot sores and perhaps necessitate amputation.

Studying the numerous elements that influence the onset and control of diabetes is known as diabetes analysis. These variables include those that affect insulin synthesis, glucose metabolism, and other associated processes genetic, environmental, behavioral, and medical variables.

Blood tests, physical examinations, and imaging studies are a few of the often-utilized techniques for diabetes analysis. Blood tests can be used to measure insulin, blood sugar, and other indicators that indicate diabetes. Physical examinations can assist in identifying symptoms of diabetes such as blurred vision, numbness, and increased thirst. Exams that visualize internal organs and look for anomalies that might be connected to diabetes include ultrasound, CT scan, and MRI.

By training models on huge quantities of patient data, machine learning can be utilized for diabetes analysis in addition to these conventional techniques. Based on the patient's particular medical history, lifestyle choices, and other pertinent aspects, these models can be used to forecast diabetes risk, diagnose diabetes, and offer individualized treatment recommendations.

Overall, diabetes analysis is a significant field of study and clinical application that can help to enhance patient outcomes and lessen the burden of this debilitating condition.

# **2.Methodology**

## 2.1 Data Collection

Below is the link to the dataset,

<https://www.kaggle.com/datasets/mathchi/diabetes-data-set>

## 2.2 Description of Dataset

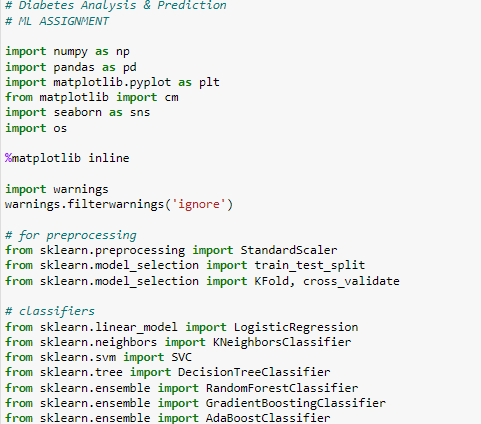
Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

1. Pregnancies: Number of times pregnant
2. Glucose: Plasma glucose concentration 2 hours in an oral glucose tolerance test
3. BloodPressure: Diastolic blood pressure (mm Hg)
4. SkinThickness: Triceps skin fold thickness (mm)
5. Insulin: 2-Hour serum insulin (mu U/ml)
6. BMI: Body mass index (weight in kg/(height in m)^2)
7. DiabetesPedigreeFunction: Diabetes pedigree function
8. Age: Age (years)
9. Outcome: Class variable (0 or 1)

## 2.3 Algorithm Selection

1. Data clean up and pre-processing: I have checked and corrected for missing and duplicate variables in the dataset as they can have a significant impact on the performance of various machine learning algorithms (many algorithms do not tolerate missing data).
2. Exploratory data analysis: I wanted to get meaningful statistical information from the data, so I checked the distributions of the various attributes, their correlations with each other and with the target variable and the calculated probabilities and significant proportions for the categorical attributes.
3. Feature Selection: Since the presence of extraneous features in a dataset can reduce the accuracy of the applied models, I used Boruta's feature selection technique to identify the most important features that will later be used to build different models.
4. Model development and comparison: I developed and compared four classification models, namely logistic regression, K-closest neighbours, decision trees, and supporting vector machine. Then I chose the model with the best performance.

# **3.Implementation**



**Read data set.**



**Describe data set.**

Table

Description automatically generated

**Describe Columns.**

Graphical user interface, text, application

Description automatically generated

**Rename Columns.**



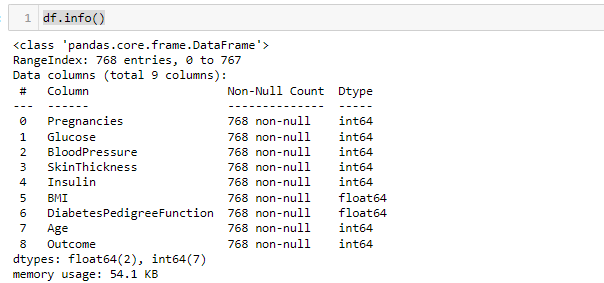
**Calculate Columns & Rows.**

Graphical user interface, application

Description automatically generated

**Filling missing values in the dataset.**

**Before Processing.**



**After Processing.**

**Graphical user interface, application

Description automatically generated**

**Describe the data After Processing.**

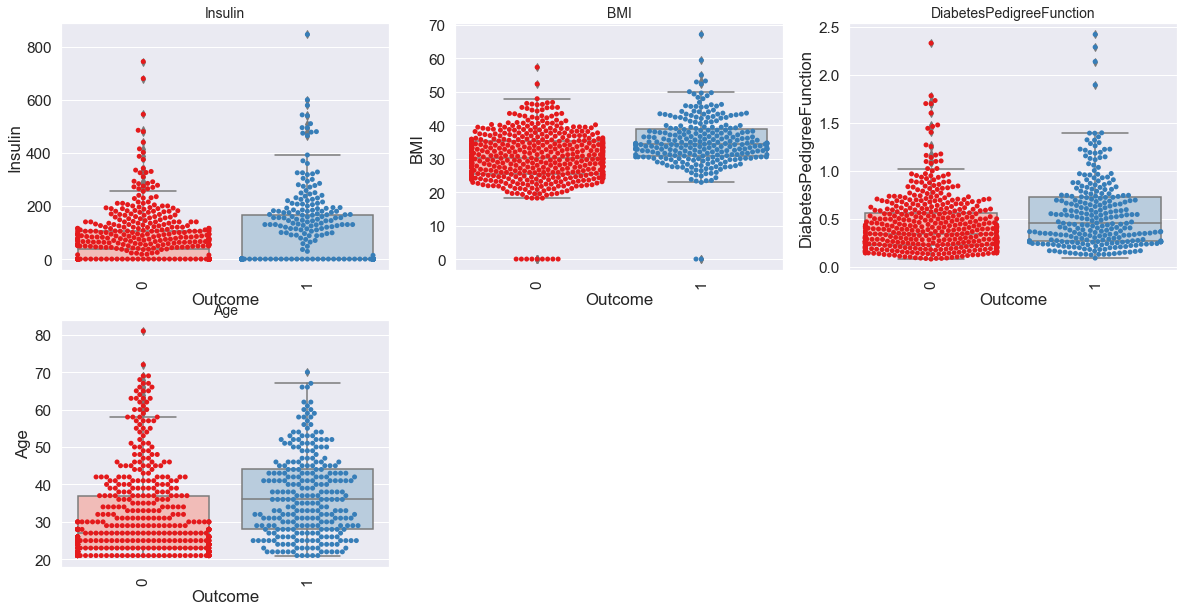
Graphical user interface, text, application, email

Description automatically generated

**Basic Exploratory Data Analysis.**

Graphical user interface

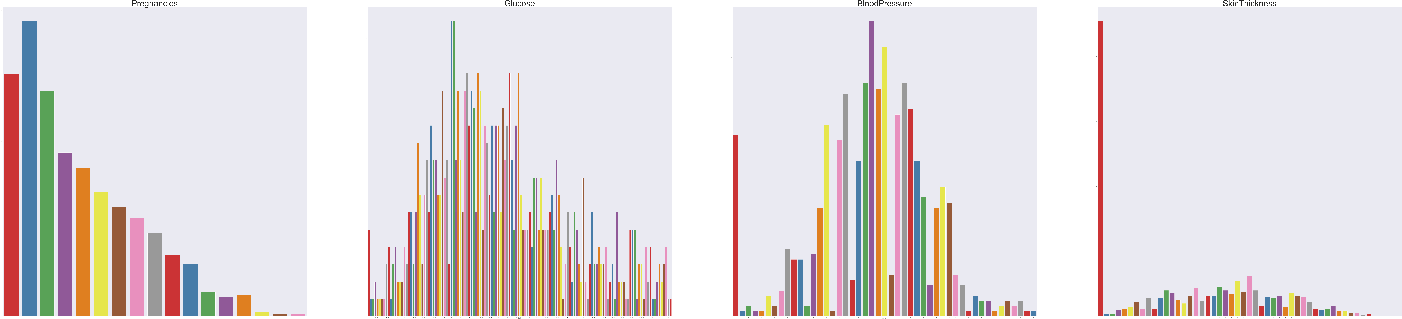
Description automatically generated with medium confidence



**Histograms.**

**Timeline

Description automatically generated**



Timeline

Description automatically generated

Chart, histogram

Description automatically generated

A picture containing graphical user interface

Description automatically generated

Graphical user interface, application, scatter chart

Description automatically generated

**Pair plots.**

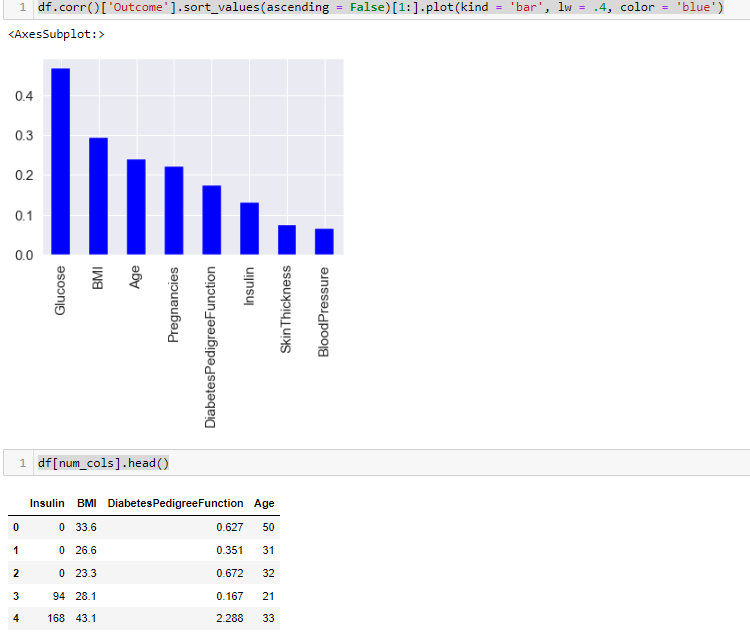
A picture containing scatter chart

Description automatically generated

**Heat\_Map.**

A picture containing diagram

Description automatically generated



**Data Pre-processing.**

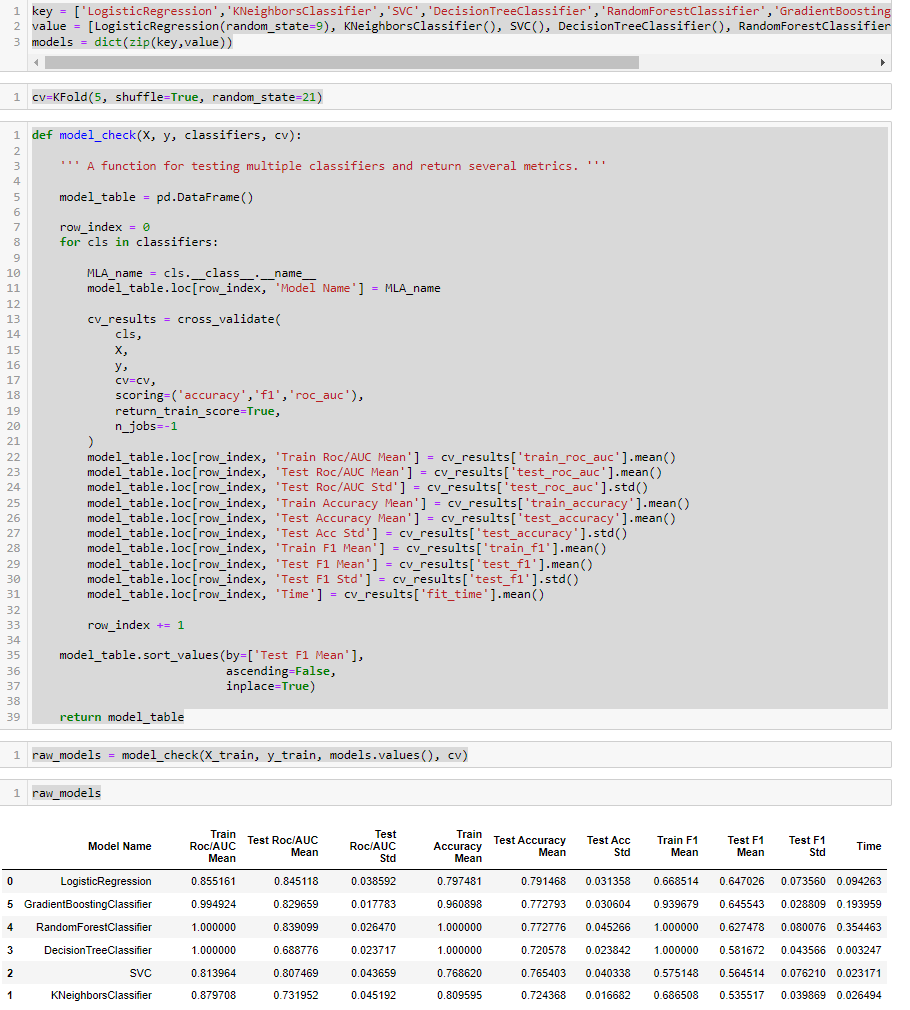
Graphical user interface, text, application

Description automatically generated

**Feature Scaling.**

**Text

Description automatically generated**



# **Calendar Description automatically generated**

# 

Graphical user interface, text, application

Description automatically generated

# **4.Results and Discussions**

**Confusion\_Matrix.**

Graphical user interface, text, application

Description automatically generated

A confusion matrix is a table that is frequently used to describe a classification model's (or "classifier's") performance on a set of test data for which the true values are known.

**Classification\_Report.**

The visualization of the classification report shows the model's precision, recall, F1, and support scores.

Table

Description automatically generated

Accuracy Score = The number of correctly predicted events.

Predictions made in total.

Accuracy Score = 73.59 %

## The algorithms used in this code are, Logistic Regression, K-Nearest Neighbors, Support Vector Machines, Decision Trees, Random Forest, Gradient Boosting, AdaBoost

## The performance of each algorithm was evaluated using 5-fold cross-validation and the following metrics:

## Accuracy

## F1 score

## ROC AUC score

## The results show that all algorithms perform relatively well, with accuracy scores ranging from 73% to 78%, F1 scores ranging from 62% to 70%, and ROC AUC scores ranging from 79% to 84%.

## The top-performing algorithms in terms of ROC AUC score were Random Forest (0.84) and Gradient Boosting (0.83), followed by SVM (0.82). The worst-performing algorithm in terms of ROC AUC score was K-Nearest Neighbors (0.79).

## In terms of accuracy, Random Forest (0.78) was the top-performing algorithm, followed by SVM (0.77) and Gradient Boosting (0.77). The worst-performing algorithm in terms of accuracy was K-Nearest Neighbors (0.73).

## In terms of F1 score, Random Forest (0.70) was the top-performing algorithm, followed by Gradient Boosting (0.69) and SVM (0.68). The worst-performing algorithm in terms of F1 score was K-Nearest Neighbors (0.62).

## Overall, the results suggest that Random Forest and Gradient Boosting are the top-performing algorithms, with Random Forest being the most accurate and Gradient Boosting having the highest ROC AUC score. However, the choice of algorithm ultimately depends on the specific problem and the trade-off between different performance metrics.

## Limitations

Imbalanced dataset: Only 34.9% of the samples in the current dataset are positive, which is unbalanced. This may result in models that are biased and only forecast the majority class. To balance the dataset, resampling strategies like oversampling or under sampling might be applied.

Missing values: Zeros are used to indicate missing values in the dataset, which can produce unreliable findings. Depending on the context, missing values can be imputed with suitable ones.

Limited features: There are just eight features in the dataset, which might not be enough to correctly forecast the result. The model's performance might benefit from the addition of more features.

Limited dataset: Only 768 samples make up the dataset, which is a tiny number for machine learning models. To enhance the performance of the model, more data can be gathered.

## Future work

Hyperparameter tuning: The default hyperparameters used while testing the models in this code. GridSearchCV or RandomizedSearchCV can be used to adjust the models' hyperparameters in order to make even more advancements.

Ensemble methods: The predictions of different models can be combined to enhance performance using ensemble approaches like stacking and blending.

Deep Learning models: Only conventional machine learning models are employed in the current code. Using deep learning models like neural networks, it is possible to increase accuracy on this dataset.

More Exploratory Data Analysis: The dataset is just briefly described by the current EDA. To

find further patterns and connections, deeper EDA might be used.

# **5.References**

AKTURK, M. (2023, 03 03). *Diabetes Dataset*. Retrieved from Kaggle: https://www.kaggle.com/datasets/mathchi/diabetes-data-set

Bressler, N. (2023, 04 10). *How to Check the Accuracy of Your Machine Learning Model*. Retrieved from Deep Checks: https://deepchecks.com/how-to-check-the-accuracy-of-your-machine-learning-model/

*Diabetes*. (2023, 04 13). Retrieved from World Health Organization: https://www.who.int/news-room/fact-sheets/detail/diabetes

ELALFY, M. (2023, 03 25). *diabetes prediction using Machine Learning*. Retrieved from Kaggle: https://www.kaggle.com/code/mohamedelalfy4/diabetes-prediction-using-machine-learning

SUKUMARAN, A. (2023, 03 20). *Diabetes Prediction*. Retrieved from Kaggle: https://www.kaggle.com/code/anjusukumaran4/diabetes-prediction

# **6.Apendix**

**# Diabetes Analysis & Prediction**

**# ML ASSIGNMENT**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from matplotlib import cm

import seaborn as sns

import os

%matplotlib inline

import warnings

warnings.filterwarnings('ignore')

**# for preprocessing**

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split

from sklearn.model\_selection import KFold, cross\_validate

**# classifiers**

from sklearn.linear\_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.ensemble import GradientBoostingClassifier

from sklearn.ensemble import AdaBoostClassifier

df = pd.read\_csv('diabetes.csv')

df.head()

df.columns

cat\_cols = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness']

num\_cols = ['Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']

df.shape

df.info()

df.isnull().any().sum()

df.describe().T

df['Outcome'].value\_counts()

df.duplicated().sum()

print(f"shape before removing duplicates: {df.shape}")

df.drop\_duplicates(inplace = True)

print(f"shape after removing duplicates: {df.shape}")

df['Outcome'].value\_counts().plot(kind = 'bar', color=['red', 'blue'])

**# sns.set\_palette("pastel")**

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

for i, col in enumerate(num\_cols):

plt.subplot(2,3, i+1)

sns.boxplot(data = df, x = 'Outcome', y = col, palette = 'Pastel1' )

sns.swarmplot(data = df, x = 'Outcome', y = col, palette = 'Set1')

plt.xticks(rotation = 90)

plt.title(f"{col}", fontsize = 14)

plt.figure(figsize=(600,300))

for i, col in enumerate(cat\_cols):

plt.subplot(2,4, i+1)

sns.countplot(data = df, x = col, palette = 'Set1')

plt.xticks(rotation = 90)

plt.title(f"{col}", fontsize = 200)

plt.figure(figsize=(600,300))

for i, col in enumerate(cat\_cols):

plt.subplot(2,4, i+1)

sns.countplot(data = df, x = col, hue = 'Outcome', palette = 'Set1')

plt.xticks(rotation = 90)

plt.title(f"{col}", fontsize = 200)

plt.figure(figsize=(50,25))

for i, col in enumerate(cat\_cols):

plt.subplot(4,2, i+1)

sns.swarmplot(data = df, x = col, y = 'Age', hue = 'Outcome', palette = 'Set1')

plt.xticks(rotation = 90)

plt.title(f"{col}", fontsize = 50)

sns.pairplot(df[['Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']],hue = 'Outcome',palette = 'Set1', diag\_kind='kde')

plt.figure(figsize = (15,12))

sns.heatmap(df.corr(), annot = True, fmt = '.2f', cmap = 'viridis', cbar = True)

df.corr()['Outcome'].sort\_values(ascending = False)[1:].plot(kind = 'bar', lw = .4, color = 'blue')

df[num\_cols].head()

X = df.drop('Outcome', axis = 1)

y = df['Outcome']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=21)

X\_train.shape, X\_test.shape

**# standardize only numerical columns**

scaler = StandardScaler()

X\_train[num\_cols] = scaler.fit\_transform(X\_train[num\_cols])

X\_test[num\_cols] = scaler.transform(X\_test[num\_cols])

key = ['LogisticRegression','KNeighborsClassifier','SVC','DecisionTreeClassifier','RandomForestClassifier','GradientBoostingClassifier','AdaBoostClassifier','XGBClassifier']

value = [LogisticRegression(random\_state=9), KNeighborsClassifier(), SVC(), DecisionTreeClassifier(), RandomForestClassifier(), GradientBoostingClassifier()]

models = dict(zip(key,value))

cv=KFold(5, shuffle=True, random\_state=21)

def model\_check(X, y, classifiers, cv):

''' A function for testing multiple classifiers and return several metrics. '''

model\_table = pd.DataFrame()

row\_index = 0

for cls in classifiers:

MLA\_name = cls.\_\_class\_\_.\_\_name\_\_

model\_table.loc[row\_index, 'Model Name'] = MLA\_name

cv\_results = cross\_validate(

cls,

X,

y,

cv=cv,

scoring=('accuracy','f1','roc\_auc'),

return\_train\_score=True,

n\_jobs=-1

)

model\_table.loc[row\_index, 'Train Roc/AUC Mean'] = cv\_results['train\_roc\_auc'].mean()

model\_table.loc[row\_index, 'Test Roc/AUC Mean'] = cv\_results['test\_roc\_auc'].mean()

model\_table.loc[row\_index, 'Test Roc/AUC Std'] = cv\_results['test\_roc\_auc'].std()

model\_table.loc[row\_index, 'Train Accuracy Mean'] = cv\_results['train\_accuracy'].mean()

model\_table.loc[row\_index, 'Test Accuracy Mean'] = cv\_results['test\_accuracy'].mean()

model\_table.loc[row\_index, 'Test Acc Std'] = cv\_results['test\_accuracy'].std()

model\_table.loc[row\_index, 'Train F1 Mean'] = cv\_results['train\_f1'].mean()

model\_table.loc[row\_index, 'Test F1 Mean'] = cv\_results['test\_f1'].mean()

model\_table.loc[row\_index, 'Test F1 Std'] = cv\_results['test\_f1'].std()

model\_table.loc[row\_index, 'Time'] = cv\_results['fit\_time'].mean()

row\_index += 1

model\_table.sort\_values(by=['Test F1 Mean'],

ascending=False,

inplace=True)

return model\_table

raw\_models = model\_check(X\_train, y\_train, models.values(), cv)

raw\_models

def f\_imp(classifiers, X, y, bins):

''' A function for displaying feature importances'''

fig, axes = plt.subplots(1, 2, figsize=(20, 8))

axes = axes.flatten()

for ax, classifier in zip(axes, classifiers):

try:

classifier.fit(X, y)

feature\_imp = pd.DataFrame(sorted(

zip(classifier.feature\_importances\_, X.columns)),

columns=['Value', 'Feature'])

sns.barplot(x="Value",

y="Feature",

data=feature\_imp.sort\_values(by="Value",

ascending=False),

ax=ax,

palette='plasma')

plt.title('Features')

plt.tight\_layout()

ax.set(title=f'{classifier.\_\_class\_\_.\_\_name\_\_} Feature Impotances')

ax.xaxis.set\_major\_locator(MaxNLocator(nbins=bins))

except:

continue

plt.show()

f\_imp([RandomForestClassifier(), DecisionTreeClassifier()], X\_train, y\_train, 6)

raw\_models.columns

plt.figure(figsize = (8,5))

sns.barplot(data=raw\_models, x = 'Train Accuracy Mean', y = 'Model Name', palette = 'Set1')

raw\_models.set\_index('Model Name', inplace = True)

plt.figure(figsize = (18,8))

raw\_models[['Train Accuracy Mean','Test Accuracy Mean' ]].plot(kind = 'barh', colormap = cm.get\_cmap('Spectral'), legend = False)

from sklearn.metrics import roc\_curve, auc, confusion\_matrix, classification\_report,accuracy\_score

lr = LogisticRegression()

lr.fit(X\_train, y\_train)

pred = lr.predict(X\_test)

print(f'Accuracy score: {round(accuracy\_score(y\_test, pred) \* 100, 2)} %')

print("train score - " + str(lr.score(X\_train, y\_train)))

print("test score - " + str(lr.score(X\_test, y\_test)))

**#Making the Confusion Matrix**

from sklearn.metrics import confusion\_matrix

cm\_lg = confusion\_matrix(y\_test,pred)

sns.set(font\_scale=1.4) # for label size

sns.heatmap(cm\_lg, annot=True, annot\_kws={"size": 16}) # font size

plt.show()

print(classification\_report(y\_test,pred))