```
#importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

sns.set()


from pandas.plotting import scatter_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import classification_report
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline

#reading the dataset
diabetes_df = pd.read_excel('diabetes.xlsx')
diabetes_df.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```
#columns

diabetes_df.columns
```

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness',
'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

```
#information about the dataset
diabetes_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

```
#more about dataset
diabetes_df.describe()
```

```
       Pregnancies     Glucose  BloodPressure  SkinThickness
Insulin  \
count   768.000000  768.000000     768.000000     768.000000
768.000000
mean      3.845052  120.894531      69.105469      20.536458
79.799479
std       3.369578   31.972618      19.355807      15.952218
115.244002
min       0.000000    0.000000       0.000000       0.000000
0.000000
25%       1.000000   99.000000      62.000000       0.000000
0.000000
50%       3.000000  117.000000      72.000000      23.000000
30.500000
75%       6.000000  140.250000      80.000000      32.000000
127.250000
max      17.000000  199.000000     122.000000      99.000000
846.000000

              BMI  DiabetesPedigreeFunction         Age     Outcome
count  768.000000                768.000000  768.000000  768.000000
mean    31.992578                  0.471876   33.240885    0.348958
std      7.884160                  0.331329   11.760232    0.476951
```

```
min      0.000000                  0.078000   21.000000    0.000000
25%     27.300000                  0.243750   24.000000    0.000000
50%     32.000000                  0.372500   29.000000    0.000000
75%     36.600000                  0.626250   41.000000    1.000000
max     67.100000                  2.420000   81.000000    1.000000
```

```
#dataset with Transpose
diabetes_df.describe().T
```

```
                          count        mean         std     min
25%  \
Pregnancies               768.0    3.845052    3.369578   0.000
1.00000
Glucose                   768.0  120.894531   31.972618   0.000
99.00000
BloodPressure             768.0   69.105469   19.355807   0.000
62.00000
SkinThickness             768.0   20.536458   15.952218   0.000
0.00000
Insulin                   768.0   79.799479  115.244002   0.000
0.00000
BMI                       768.0   31.992578    7.884160   0.000
27.30000
DiabetesPedigreeFunction  768.0    0.471876    0.331329   0.078
0.24375
Age                       768.0   33.240885   11.760232  21.000
24.00000
Outcome                   768.0    0.348958    0.476951   0.000
0.00000

                               50%        75%     max
Pregnancies                  3.0000    6.00000   17.00
Glucose                    117.0000  140.25000  199.00
BloodPressure               72.0000   80.00000  122.00
SkinThickness               23.0000   32.00000   99.00
Insulin                     30.5000  127.25000  846.00
BMI                         32.0000   36.60000   67.10
DiabetesPedigreeFunction     0.3725    0.62625    2.42
Age                         29.0000   41.00000   81.00
Outcome                      0.0000    1.00000    1.00
```

```
#dataset have nullvalues or not
diabetes_df.isnull().head(10)
```

```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin    BMI
\
0        False    False          False          False    False  False

1        False    False          False          False    False  False

2        False    False          False          False    False  False
```

```
3        False    False          False          False    False  False

4        False    False          False          False    False  False

5        False    False          False          False    False  False

6        False    False          False          False    False  False

7        False    False          False          False    False  False

8        False    False          False          False    False  False

9        False    False          False          False    False  False
```

```
   DiabetesPedigreeFunction    Age  Outcome
0                     False  False    False
1                     False  False    False
2                     False  False    False
3                     False  False    False
4                     False  False    False
5                     False  False    False
6                     False  False    False
7                     False  False    False
8                     False  False    False
9                     False  False    False
```

```
#number of nullvalues
diabetes_df.isnull().sum()
```

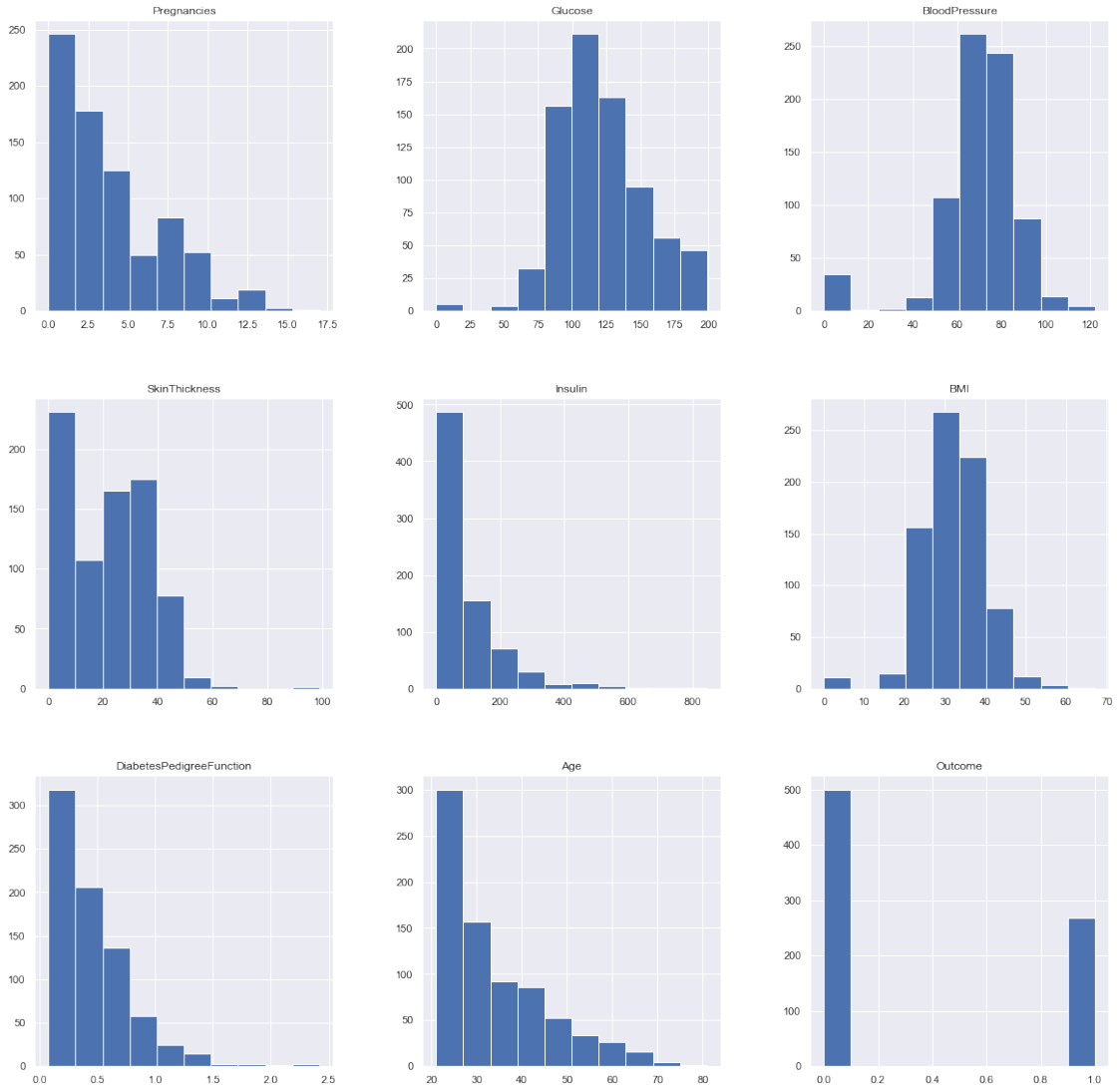
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

```
diabetes_df_copy = diabetes_df.copy(deep = True)
diabetes_df_copy[['Glucose','BloodPressure','SkinThickness','Insulin',
'BMI']] =
diabetes_df_copy[['Glucose','BloodPressure','SkinThickness','Insulin',
'BMI']].replace(0,np.NaN)

# Showing the Count of NANs
print(diabetes_df_copy.isnull().sum())
```

```
Pregnancies                   0
Glucose                       5
BloodPressure                35
SkinThickness               227
Insulin                     374
BMI                          11
DiabetesPedigreeFunction      0
Age                           0
Outcome                       0
dtype: int64
```

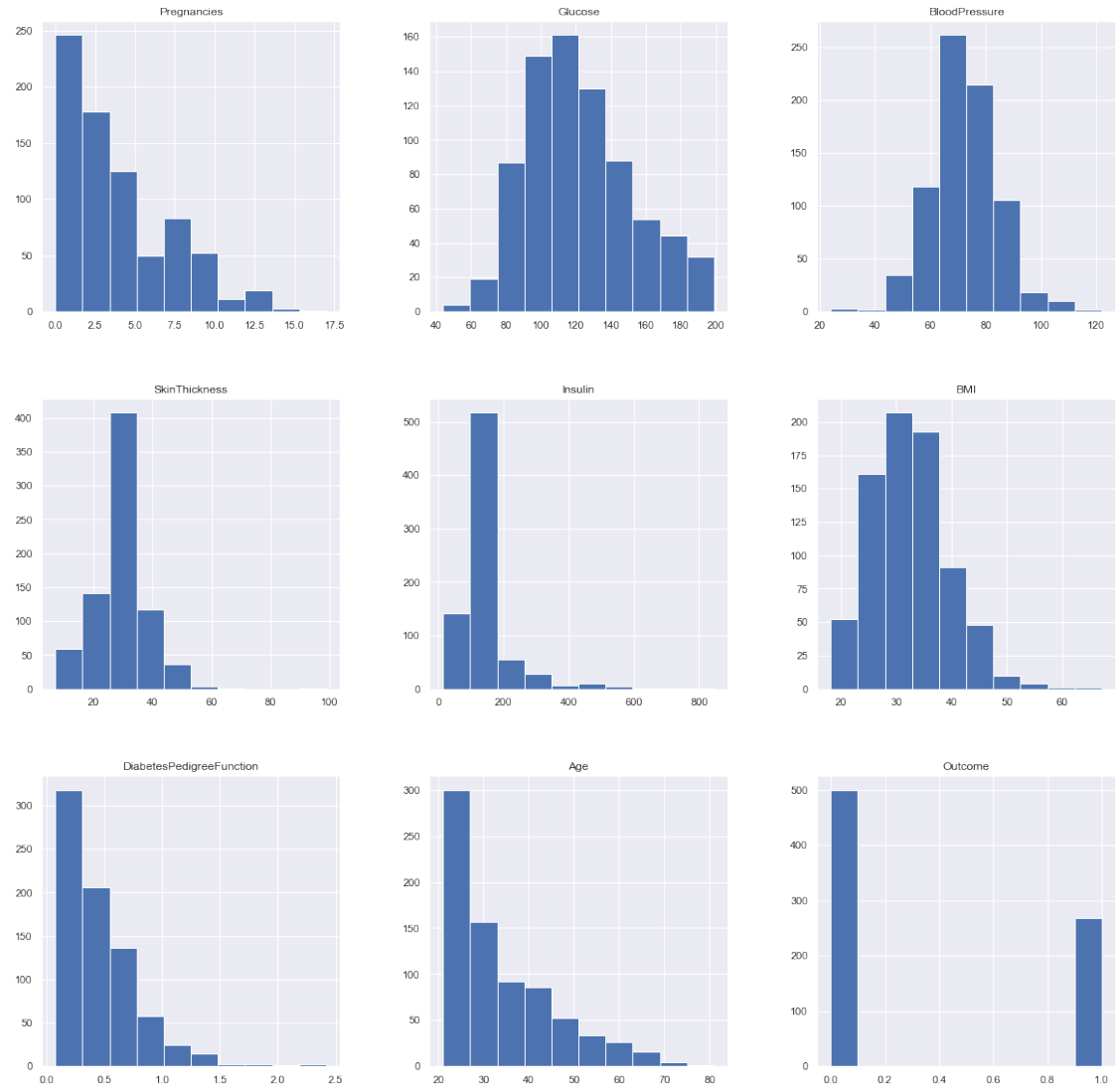
```
#DATA VISUALIZATION
p = diabetes_df.hist(figsize = (20,20))
```




```
diabetes_df_copy['Glucose'].fillna(diabetes_df_copy['Glucose'].mean(),
inplace = True)
diabetes_df_copy['BloodPressure'].fillna(diabetes_df_copy['BloodPressu
re'].mean(), inplace = True)
```

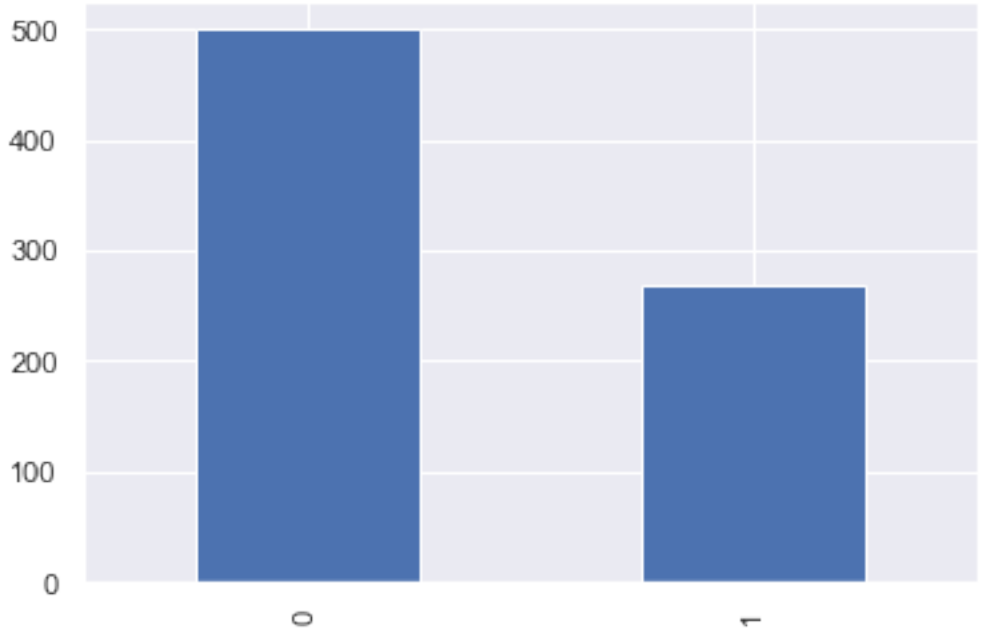
```
diabetes_df_copy['SkinThickness'].fillna(diabetes_df_copy['SkinThickness'].median(), inplace = True)
diabetes_df_copy['Insulin'].fillna(diabetes_df_copy['Insulin'].median(), inplace = True)
diabetes_df_copy['BMI'].fillna(diabetes_df_copy['BMI'].median(), inplace = True)

#removing NAN values
p = diabetes_df_copy.hist(figsize = (20,20))
```

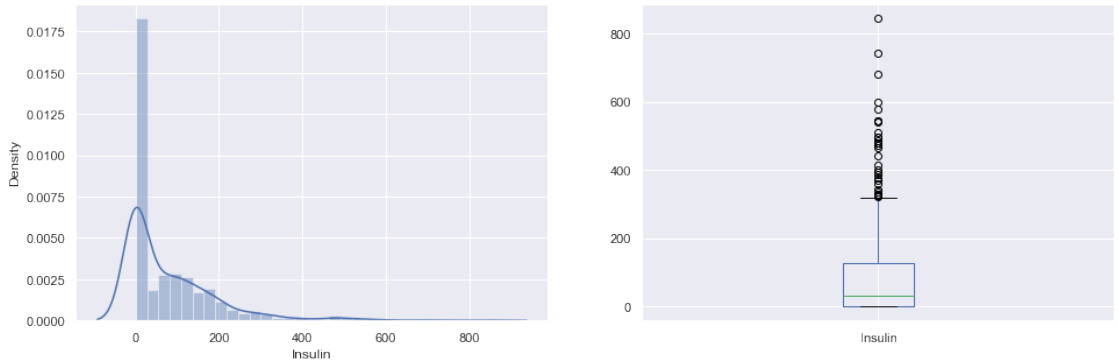



```
color_wheel = {1: "#0392cf", 2: "#7bc043"}
colors = diabetes_df["Outcome"].map(lambda x: color_wheel.get(x + 1))
print(diabetes_df.Outcome.value_counts())
p=diabetes_df.Outcome.value_counts().plot(kind="bar")
```

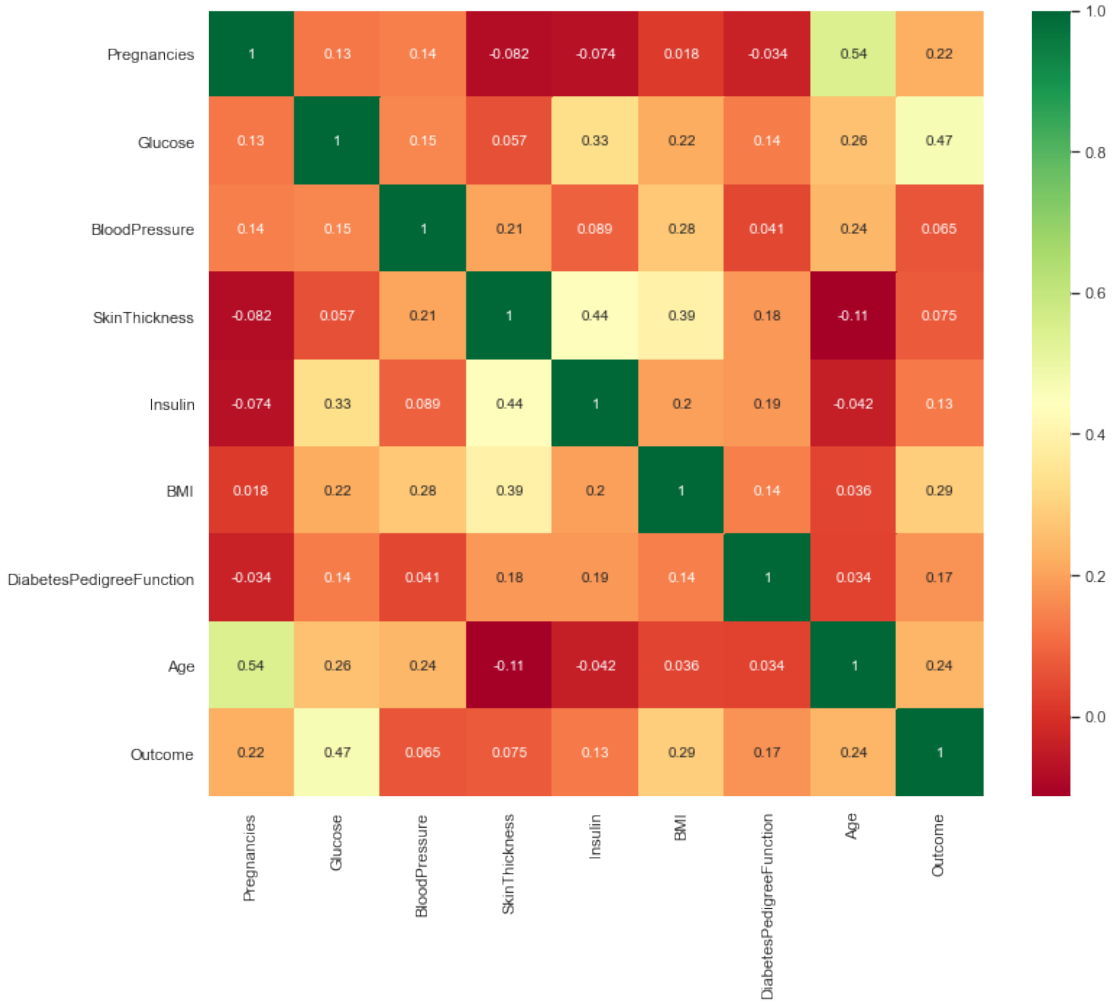
```
0    500
1    268
Name: Outcome, dtype: int64
```




```
plt.subplot(121), sns.distplot(diabetes_df['Insulin'])
plt.subplot(122), diabetes_df['Insulin'].plot.box(figsize=(16,5))
plt.show()
```




```
#CORRELATION BETWEEN ALL THE FEATURES
plt.figure(figsize=(12,10))
# seaborn has an easy method to showcase heatmap
p = sns.heatmap(diabetes_df.corr(), annot=True,cmap ='RdYlGn')
```




```
#SCALING THE DATA
diabetes_df_copy.head()
```

```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin
BMI  \
0            6    148.0           72.0           35.0    125.0  33.6

1            1     85.0           66.0           29.0    125.0  26.6

2            8    183.0           64.0           29.0    125.0  23.3

3            1     89.0           66.0           23.0     94.0  28.1

4            0    137.0           40.0           35.0    168.0  43.1


   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
```

```
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
```

```
#after Standard scaling
sc_X = StandardScaler()
X =
pd.DataFrame(sc_X.fit_transform(diabetes_df_copy.drop(["Outcome"],axis
= 1),), columns=['Pregnancies',
'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI',
'DiabetesPedigreeFunction', 'Age'])
X.head()
```

```
   Pregnancies   Glucose  BloodPressure  SkinThickness   Insulin
BMI  \
0     0.639947  0.865108      -0.033518       0.670643 -0.181541
0.166619
1    -0.844885 -1.206162      -0.529859      -0.012301 -0.181541 -
0.852200
2     1.233880  2.015813      -0.695306      -0.012301 -0.181541 -
1.332500
3    -0.844885 -1.074652      -0.529859      -0.695245 -0.540642 -
0.633881
4    -1.141852  0.503458      -2.680669       0.670643  0.316566
1.549303

   DiabetesPedigreeFunction       Age
0                  0.468492  1.425995
1                 -0.365061 -0.190672
2                  0.604397 -0.105584
3                 -0.920763 -1.041549
4                  5.484909 -0.020496
```

```
y = diabetes_df_copy.Outcome
y
```

```
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

```
#MODEL BUILDING
X = diabetes_df.drop('Outcome', axis=1)
y = diabetes_df['Outcome']

#USING train_test_split function
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,y,
test_size=0.33,
                                                    random_state=7)

#RANDOM FOREST

from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(n_estimators=200)
rfc.fit(X_train, y_train)

RandomForestClassifier(n_estimators=200)

#ACCURACY
rfc_train = rfc.predict(X_train)
from sklearn import metrics

print("Accuracy_Score =", format(metrics.accuracy_score(y_train,
rfc_train)))

Accuracy_Score = 1.0

from sklearn import metrics

predictions = rfc.predict(X_test)
print("Accuracy_Score =", format(metrics.accuracy_score(y_test,
predictions)))

Accuracy_Score = 0.7755905511811023

#CLASSIFICATION REPORT AND CONFUSION MATRIX
from sklearn.metrics import classification_report, confusion_matrix

print(confusion_matrix(y_test, predictions))
print(classification_report(y_test,predictions))

[[136  26]
 [ 31  61]]
              precision    recall  f1-score   support

           0       0.81      0.84      0.83       162
           1       0.70      0.66      0.68        92

    accuracy                           0.78       254
```

```
   macro avg       0.76      0.75      0.75       254
weighted avg       0.77      0.78      0.77       254
```

```
#DECISION TREE

from sklearn.tree import DecisionTreeClassifier

dtree = DecisionTreeClassifier()
dtree.fit(X_train, y_train)

DecisionTreeClassifier()

#ACCURACY
from sklearn import metrics

predictions = dtree.predict(X_test)
print("Accuracy Score =",
format(metrics.accuracy_score(y_test,predictions)))

Accuracy Score = 0.7047244094488189

from sklearn.metrics import classification_report, confusion_matrix

print(confusion_matrix(y_test, predictions))
print(classification_report(y_test,predictions))

[[125  37]
 [ 38  54]]
              precision    recall  f1-score   support

           0       0.77      0.77      0.77       162
           1       0.59      0.59      0.59        92

    accuracy                           0.70       254
   macro avg       0.68      0.68      0.68       254
weighted avg       0.70      0.70      0.70       254


#SUPPORT VECTOR MACHINE(SVM)

from sklearn.svm import SVC

svc_model = SVC()
svc_model.fit(X_train, y_train)

SVC()

svc_pred = svc_model.predict(X_test)
```

```
#accuracy
from sklearn import metrics

print("Accuracy Score =", format(metrics.accuracy_score(y_test,
svc_pred)))
```

```
Accuracy Score = 0.7480314960629921
```

```
#classification report and confusion matrix of the svm classifier
from sklearn.metrics import classification_report, confusion_matrix

print(confusion_matrix(y_test, svc_pred))
print(classification_report(y_test,svc_pred))
```

```
[[145  17]
 [ 47  45]]
              precision    recall  f1-score   support

           0       0.76      0.90      0.82       162
           1       0.73      0.49      0.58        92

    accuracy                           0.75       254
   macro avg       0.74      0.69      0.70       254
weighted avg       0.74      0.75      0.73       254
```

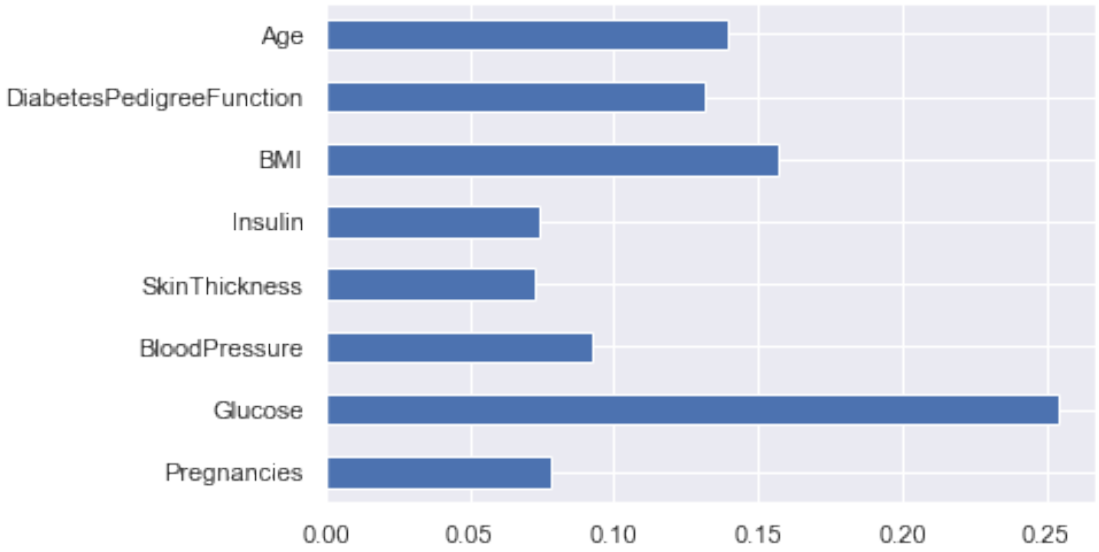
```
#FEATURE IMPORTANCE

rfc.feature_importances_
```

```
array([0.07783765, 0.25400916, 0.09262098, 0.07263453, 0.07448972,
       0.15732518, 0.13119862, 0.13988417])
```

```
#PLOTTING
(pd.Series(rfc.feature_importances_,
index=X.columns).plot(kind='barh'))
```

```
<AxesSubplot:>
```




```
#SAVING MODEL-RANDOM FOREST

import pickle

# Firstly we will be using the dump() function to save the model using
pickle
saved_model = pickle.dumps(rfc)

# Then we will be loading that saved model
rfc_from_pickle = pickle.loads(saved_model)

# lastly, after loading that model we will use this to make
predictions
rfc_from_pickle.predict(X_test)
```

```
array([0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0,
0,
       1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1,
0,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1,
1,
       0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
0,
       1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1,
       0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
1,
       0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
0,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0,
0,
       1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1,
```

```
0,
       0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0,
0,
       0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0,
1,
       1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1], dtype=int64)
```

```
diabetes_df.head()
```

```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin
BMI  \
0            6      148             72             35        0  33.6

1            1       85             66             29        0  26.6

2            8      183             64              0        0  23.3

3            1       89             66             23       94  28.1

4            0      137             40             35      168  43.1


   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
```

```
diabetes_df.tail()
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI
\
763           10      101             76             48      180  32.9

764            2      122             70             27        0  36.8

765            5      121             72             23      112  26.2

766            1      126             60              0        0  30.1

767            1       93             70             31        0  30.4


     DiabetesPedigreeFunction  Age  Outcome
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0
```

```
rfc.predict([[0,137,40,35,168,43.1,2.228,33]]) #4th patient
```

```
array([1], dtype=int64)
```