nir

June 2, 2022

0.1

```
1.                                                       .
2.                            .             ,                          .                        .
3.           ,                        .                     .             .                     ,
4.                          .                                                    .
5.                                 .                                    .
6.                                                 .                           ,
7.                                                  .
8.                (baseline)                                   .
9.                            .                       –     .
10.           8                              .                          baseline-    .
11.                                                  .
```

.   .

0.2 1. .

Diabetes Health Indicators Dataset (https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset).

, , .

, .

csv :

- Diabetes_binary - 1 - / 0 - .
- HighBP - 1 - / 0 - .
- HighChol - 1 - / 0 -
- CholCheck - 1 - 5- / 0 -
- BMI -
- Smoker - 100 ? - 1/ - 0
- Stroke - ? - 1/ - 0.
- HeartDiseaseorAttack - ( ) ( ) 0 = 1 =
- PhysActivity - 30 , 0 = 1 =
- Fruits - 1 0 = 1 =
- Veggies - 1 0 = 1 =
- HvyAlcoholConsumption - (, 14 , , 7) 0 =
- AnyHealthcare - - , , , HMO . . 0 = 1 =

- NoDocbcCost - 12 , , - ? 0 = 1 =
- GenHlth - , : 1 5 1 = 2 = 3 = 4 = 5 =
- MentHlth - , , , ,
- PhysHlth - , , 30
- DiffWalk - ? 0 = 1 =
- Sex - 0 = 1 =
- Age - 13- (_AGEG5YR, .) 1 = 18–24 9 = 60–64 13 = 80
- Education - (EDUCA .)
- Income - (INCOME2 .)

-

[ ]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from collections import Counter

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier,␣
  ↪GradientBoostingClassifier
from xgboost import XGBClassifier

from sklearn.metrics import accuracy_score, confusion_matrix, roc_auc_score,␣
  ↪ConfusionMatrixDisplay, precision_score, recall_score, f1_score,␣
  ↪classification_report, roc_curve, plot_roc_curve, auc,␣
  ↪precision_recall_curve, plot_precision_recall_curve, average_precision_score
from sklearn.model_selection import GridSearchCV

import warnings
warnings.filterwarnings('ignore')
```

[ ]:
```
data = pd.read_csv('./diabetes_binary_5050split_health_indicators_BRFSS2015.
  ↪csv')
```

```
[ ]: data.head()
```

```
[ ]:   Diabetes_binary  HighBP  HighChol  CholCheck   BMI  Smoker  Stroke  \
     0             0.0     1.0       0.0        1.0  26.0     0.0     0.0
     1             0.0     1.0       1.0        1.0  26.0     1.0     1.0
     2             0.0     0.0       0.0        1.0  26.0     0.0     0.0
     3             0.0     1.0       1.0        1.0  28.0     1.0     0.0
     4             0.0     0.0       0.0        1.0  29.0     1.0     0.0

        HeartDiseaseorAttack  PhysActivity  Fruits  …  AnyHealthcare  \
     0                   0.0           1.0     0.0  …            1.0
     1                   0.0           0.0     1.0  …            1.0
     2                   0.0           1.0     1.0  …            1.0
     3                   0.0           1.0     1.0  …            1.0
     4                   0.0           1.0     1.0  …            1.0

        NoDocbcCost  GenHlth  MentHlth  PhysHlth  DiffWalk  Sex   Age  Education  \
     0          0.0      3.0       5.0      30.0       0.0  1.0   4.0        6.0
     1          0.0      3.0       0.0       0.0       0.0  1.0  12.0        6.0
     2          0.0      1.0       0.0      10.0       0.0  1.0  13.0        6.0
     3          0.0      3.0       0.0       3.0       0.0  1.0  11.0        6.0
     4          0.0      2.0       0.0       0.0       0.0  0.0   8.0        5.0

        Income
     0     8.0
     1     8.0
     2     8.0
     3     8.0
     4     8.0

     [5 rows x 22 columns]
```

```
[ ]: print(f'             {data.shape[0]}       {data.shape[1]}     .')
```

```
                 70692      22     .
```

```
[ ]: data.info()
```

```
    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 70692 entries, 0 to 70691
    Data columns (total 22 columns):
     #   Column                Non-Null Count  Dtype
    ---  ------                --------------  -----
     0   Diabetes_binary       70692 non-null  float64
     1   HighBP                70692 non-null  float64
     2   HighChol              70692 non-null  float64
     3   CholCheck             70692 non-null  float64
     4   BMI                   70692 non-null  float64
```

```
 5   Smoker                70692 non-null  float64
 6   Stroke                70692 non-null  float64
 7   HeartDiseaseorAttack  70692 non-null  float64
 8   PhysActivity          70692 non-null  float64
 9   Fruits                70692 non-null  float64
 10  Veggies               70692 non-null  float64
 11  HvyAlcoholConsump     70692 non-null  float64
 12  AnyHealthcare         70692 non-null  float64
 13  NoDocbcCost           70692 non-null  float64
 14  GenHlth               70692 non-null  float64
 15  MentHlth              70692 non-null  float64
 16  PhysHlth              70692 non-null  float64
 17  DiffWalk              70692 non-null  float64
 18  Sex                   70692 non-null  float64
 19  Age                   70692 non-null  float64
 20  Education             70692 non-null  float64
 21  Income                70692 non-null  float64
dtypes: float64(22)
memory usage: 11.9 MB
```

[ ]:
```
data = data.astype('int')
```

0.3 2.

[ ]:
```
data.isnull().sum()
```

[ ]:
```
Diabetes_binary         0
HighBP                  0
HighChol                0
CholCheck               0
BMI                     0
Smoker                  0
Stroke                  0
HeartDiseaseorAttack    0
PhysActivity            0
Fruits                  0
Veggies                 0
HvyAlcoholConsump       0
AnyHealthcare           0
NoDocbcCost             0
GenHlth                 0
MentHlth                0
PhysHlth                0
DiffWalk                0
Sex                     0
Age                     0
Education               0
```

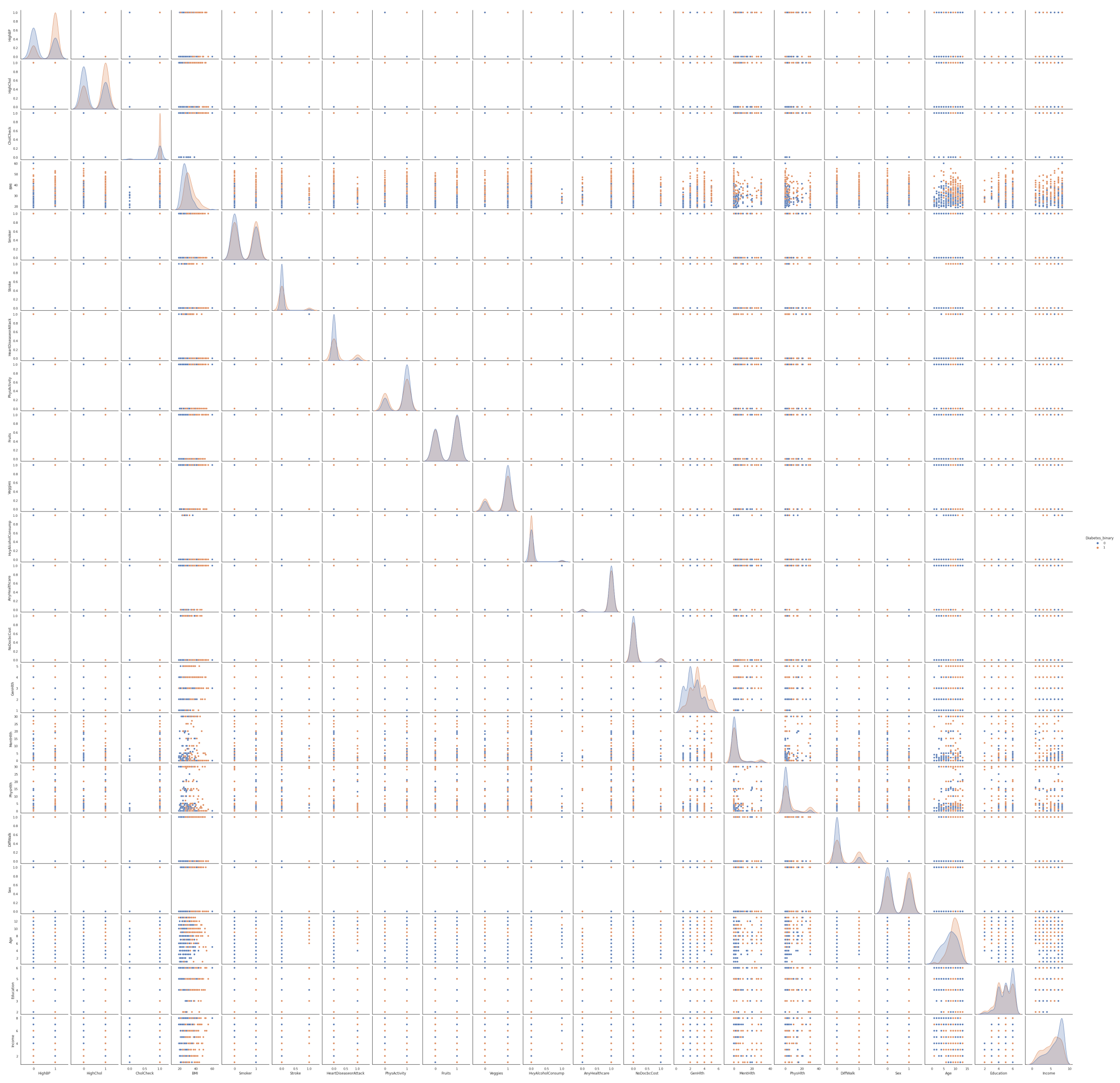
```
Income                  0
dtype: int64
```

```
[ ]: total = data.shape[0]
     class_0, class_1 = data['Diabetes_binary'].value_counts()
     print('    0        {}%,       1        {}%.'
           .format(round(class_0 / total, 4)*100, round(class_1 / total, 4)*100))
```

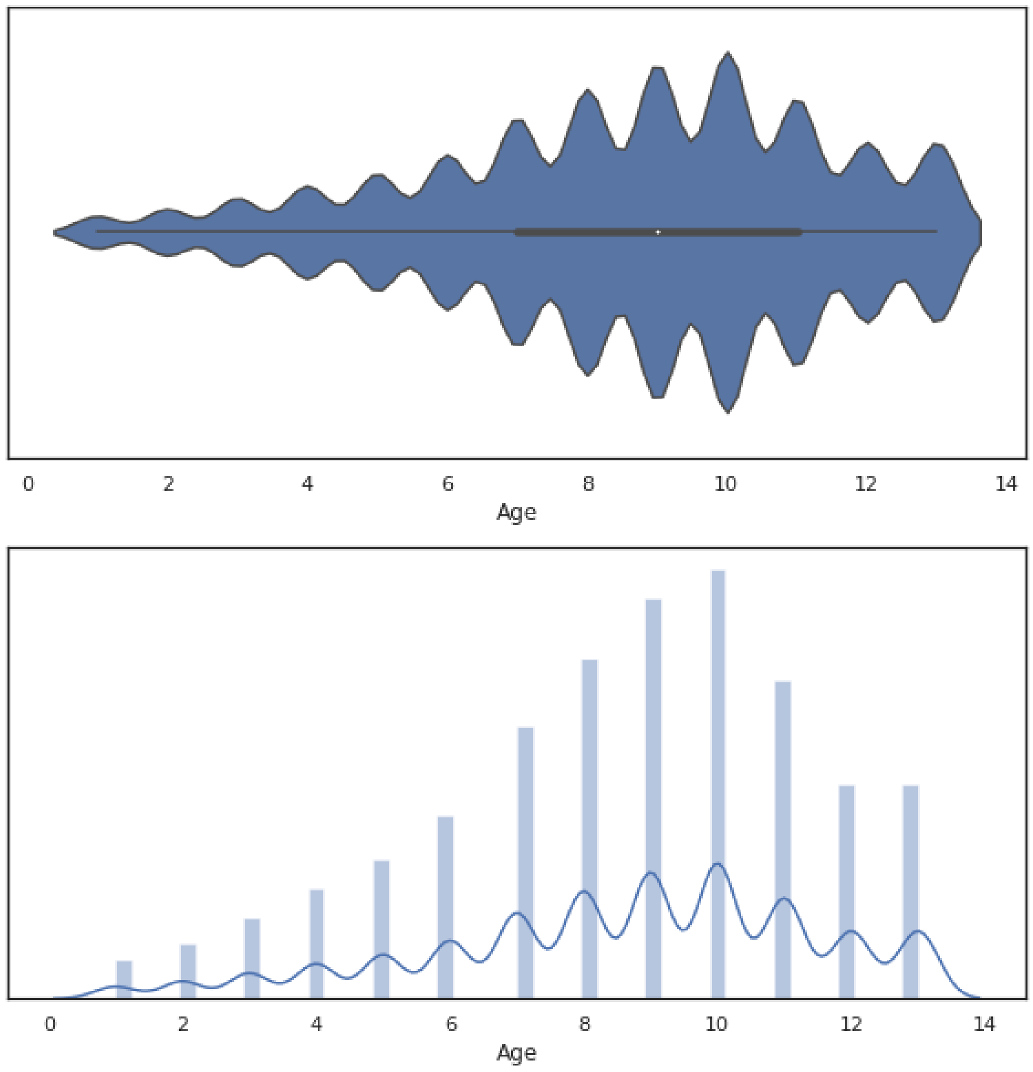
```
   0        50.0%,       1        50.0%.

                                              .
```

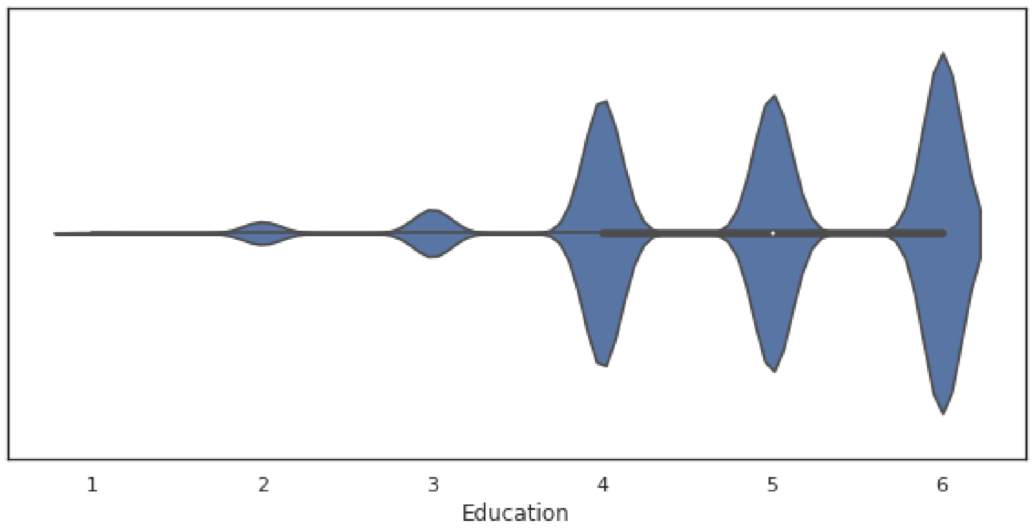
```
[ ]: mini_data = data.sample(frac=1)
     mini_data = mini_data[:500]
     sns.pairplot(mini_data, hue='Diabetes_binary')
```

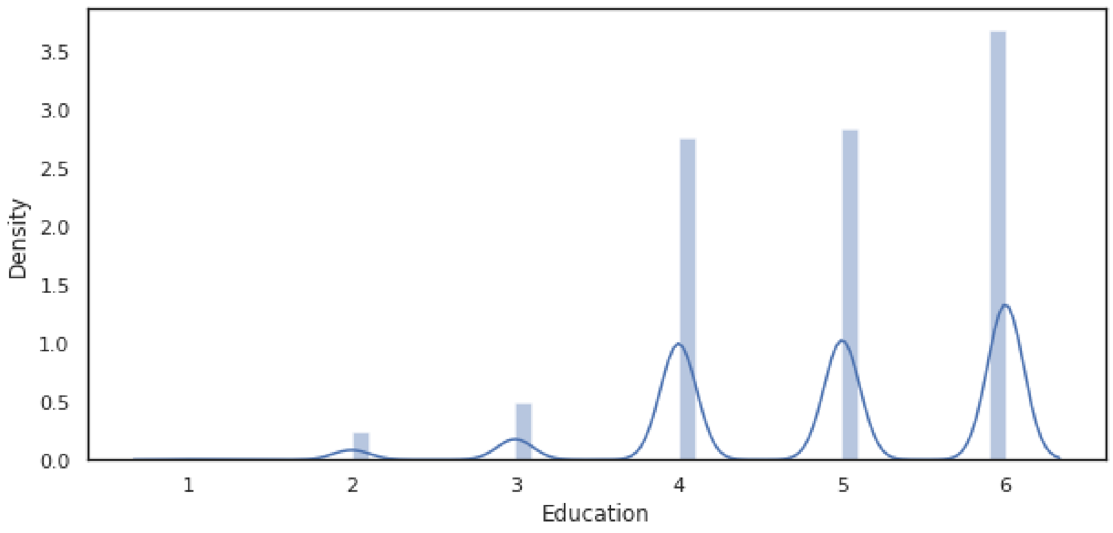
```
[ ]: <seaborn.axisgrid.PairGrid at 0x7f32494584c0>
```

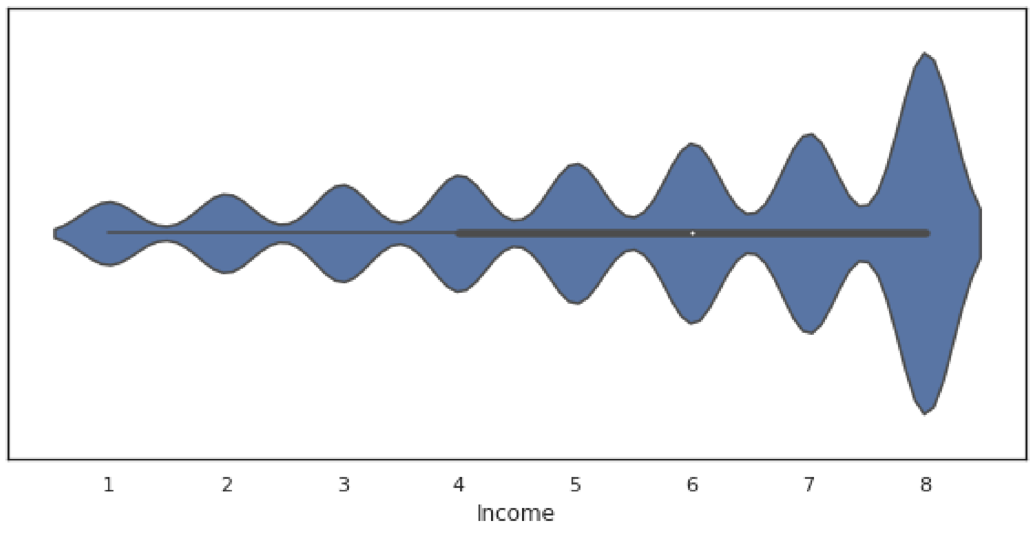


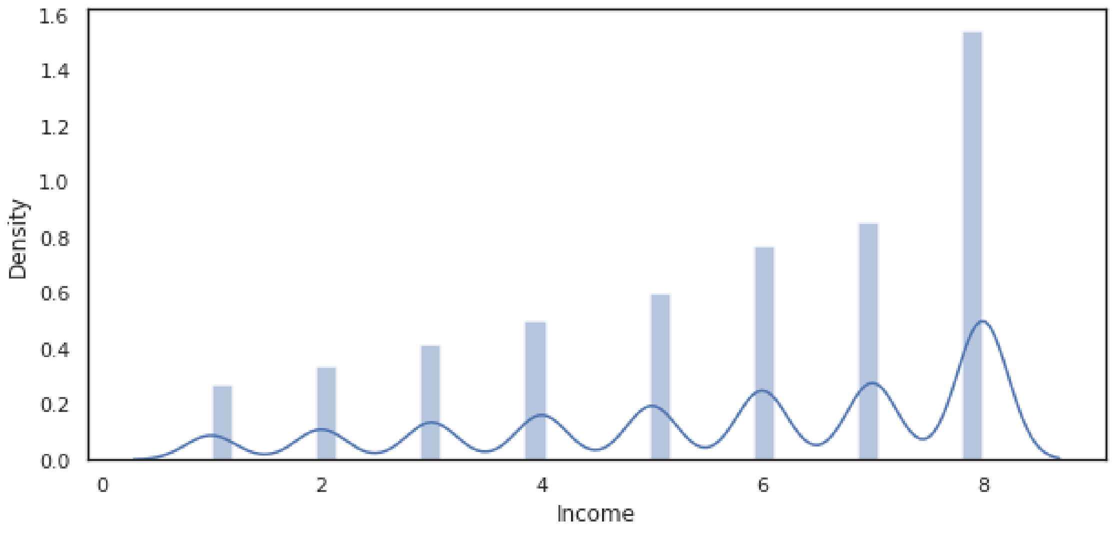
```
[ ]: for col in ['Age', 'Education', 'Income', 'BMI', 'PhysHlth']:
         fig, ax = plt.subplots(2, 1, figsize=(10,10))
         sns.violinplot(ax=ax[0], x=data[col])
         sns.distplot(data[col], ax=ax[1])
```

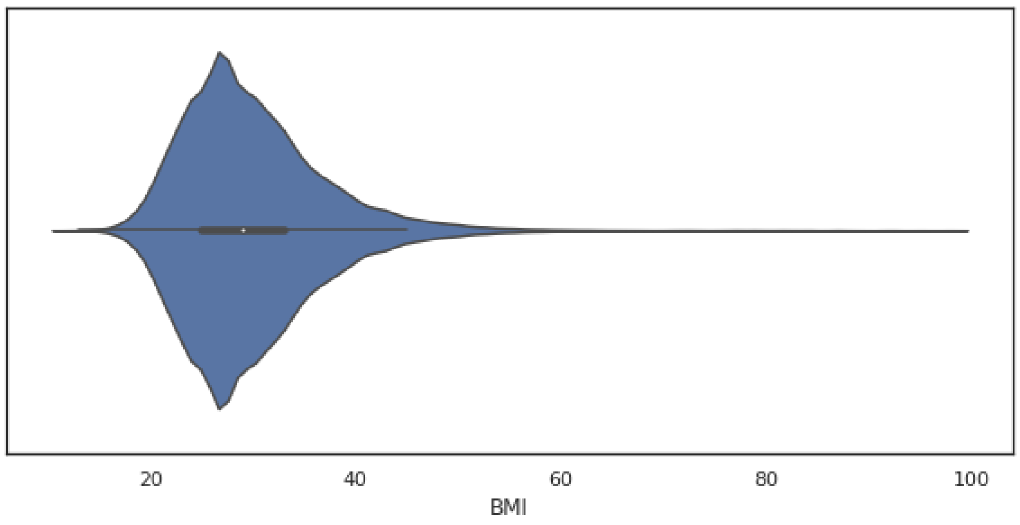


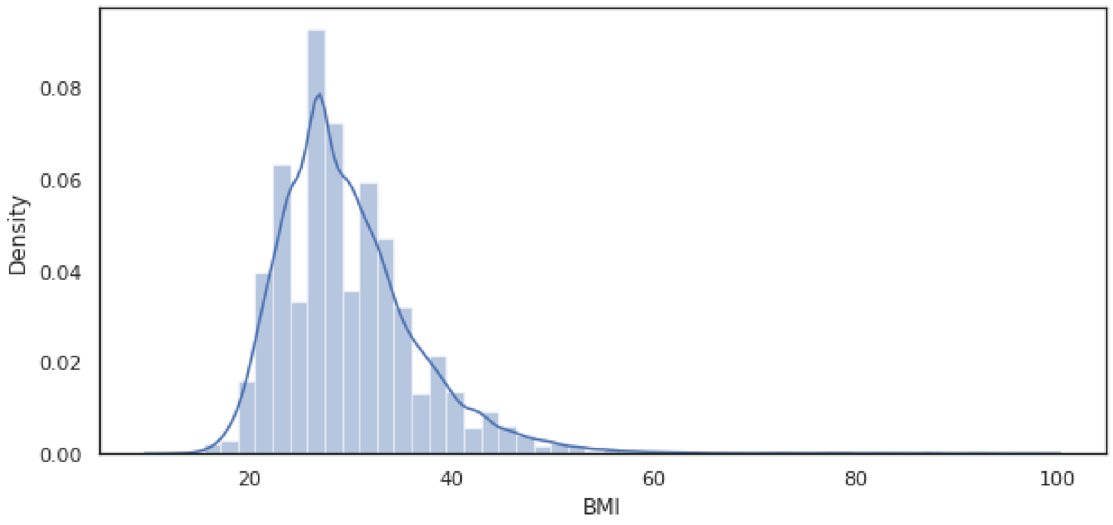

1
2
3
4
5
6
Education

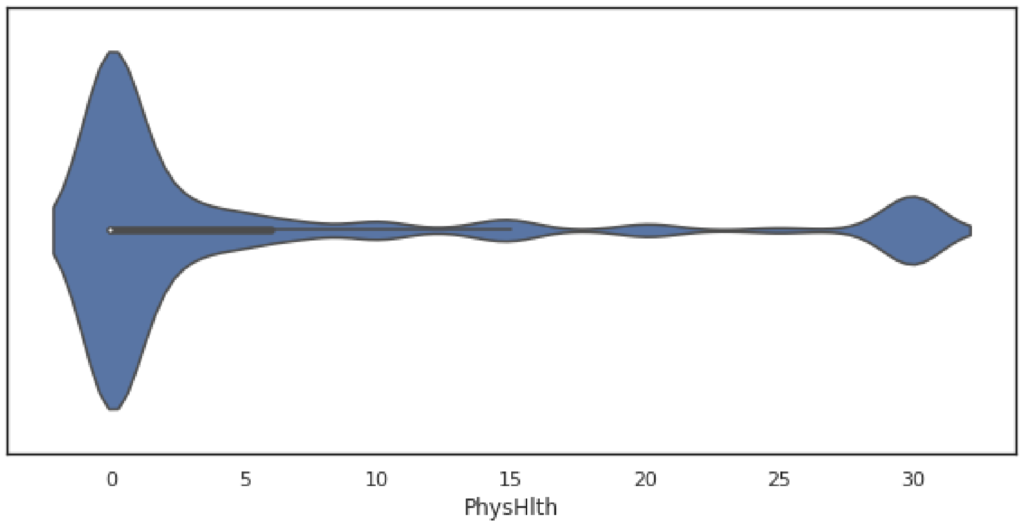

3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Density
1
2
3
4
5
6
Education

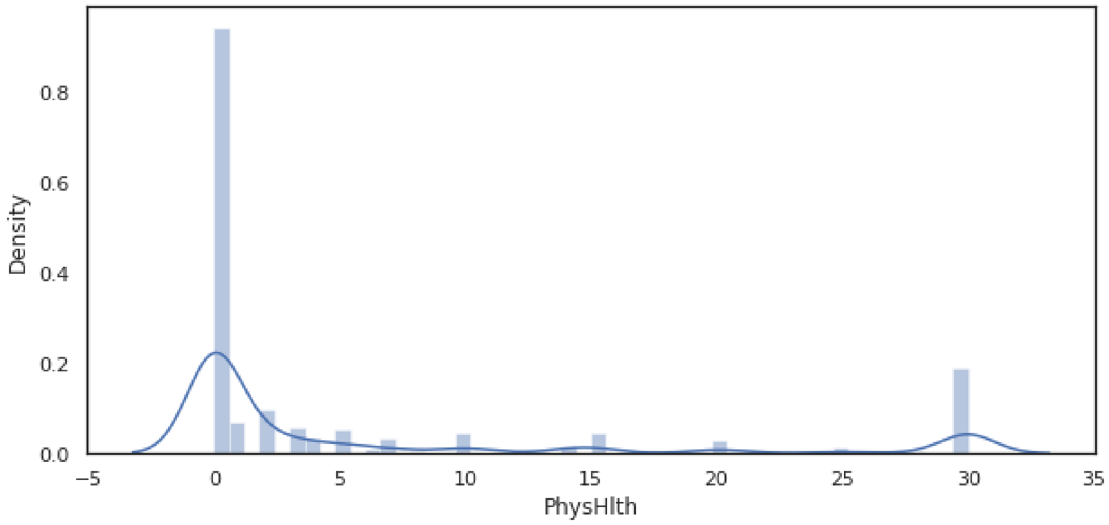

1
2
3
4
5
6
7
8
Income


1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Density
0
2
4
6
8
Income


20
40
60
80
100
BMI


0.08
0.06
0.04
0.02
0.00
Density
20
40
60
80
100
BMI

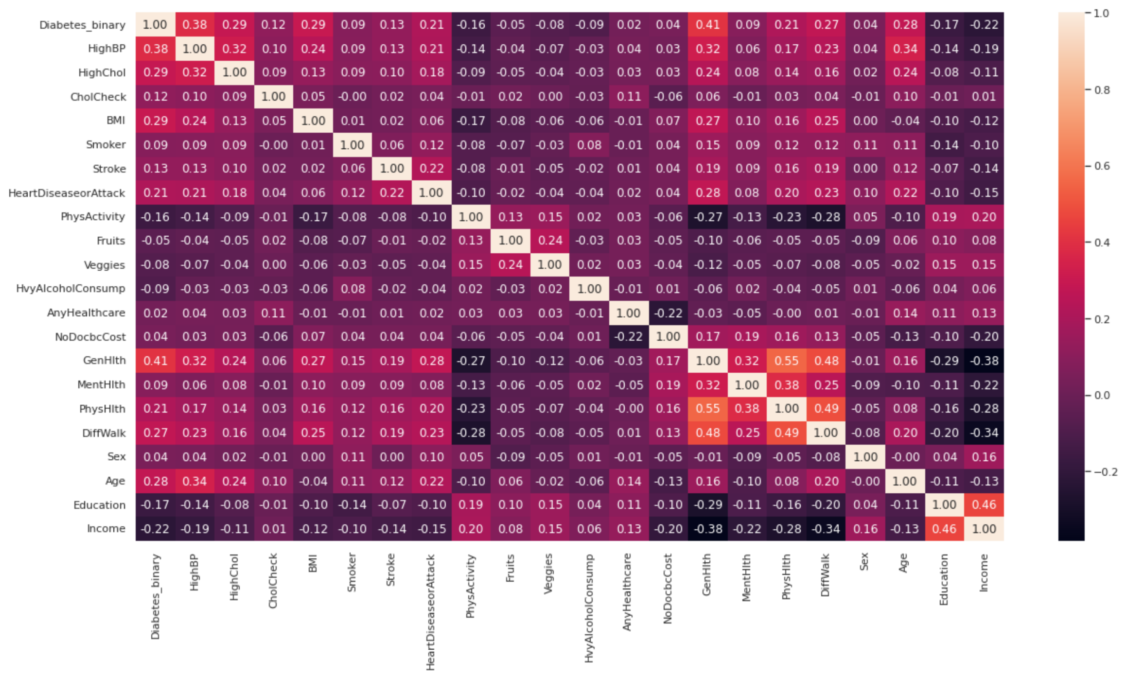




0.3.1 3. .

```
[ ]: fig, ax = plt.subplots(1, 1, sharex='col', sharey='row', figsize=(20,10))
     sns.heatmap(data.corr(), annot=True, fmt='.2f')
```

```
[ ]: <AxesSubplot:>
```



```
[ ]: def correlation_fun(ds,threshold):
         corr_col=set()
         corr_mat=ds.corr()
         for i in range(len(corr_mat.columns)):
             for j in range(i):
                 if abs(corr_mat.iloc[i,j])>threshold:
                     colname=corr_mat.columns[i]
                     corr_col.add(colname)
         return corr_col
```

```
[ ]: threshold = 0.5

     correlation_fun(data.drop("Diabetes_binary",axis=1),threshold)
```

```
[ ]: {'PhysHlth'}
```

, . , , , .
Sex, AnyHealthcare, NoDocbcCost, Fruits, PhysHlth.

```
[ ]: drop_columns = ['Sex','AnyHealthcare', 'NoDocbcCost', 'Fruits', 'PhysHlth']
```

```
[ ]: data=data.drop(drop_columns,axis=1)
```

```
[ ]: data.head()
[ ]:    Diabetes_binary  HighBP  HighChol  CholCheck  BMI  Smoker  Stroke  \
     0                0       1         0          1   26       0       0
     1                0       1         1          1   26       1       1
     2                0       0         0          1   26       0       0
     3                0       1         1          1   28       1       0
     4                0       0         0          1   29       1       0

        HeartDiseaseorAttack  PhysActivity  Veggies  HvyAlcoholConsump  GenHlth  \
     0                     0             1        1                  0        3
     1                     0             0        0                  0        3
     2                     0             1        1                  0        1
     3                     0             1        1                  0        3
     4                     0             1        1                  0        2

        MentHlth  DiffWalk  Age  Education  Income
     0         5         0    4          6       8
     1         0         0   12          6       8
     2         0         0   13          6       8
     3         0         0   11          6       8
     4         0         0    8          5       8
```

```
[ ]: data.shape
```

```
[ ]: (70692, 17)
```

0.3.2 .

:

, :

precision:

, "accuracy".

$$precision = \frac{TP}{TP + FP}$$

, , .

precision_score.

recall ( ):

$$recall = \frac{TP}{TP + FN}$$

, .

recall_score.

F1-

, precision recall F_β - ,
precision recall:

$$F_\beta = (1+\beta^2)\cdot\frac{precision \cdot recall}{precision + recall}$$

β .

F1- (F-) β=1:

$$F_1 = 2\cdot\frac{precision \cdot recall}{precision + recall}$$

f1_score.

ROC AUC

:

$$truePR = \frac{TP}{TP + FN}$$

True Positive Rate, . recall.

$$falsePR = \frac{FP}{FP + TN}$$

False Positive Rate, . .

ROC- (0,0)-(0,1)-(1,1), .

, .

- ROC AUC (Area Under the Receiver Operating Characteristic Curve). .

ROC AUC roc_auc_score.

0.3.3 .

: - - - - -

0.3.4 .

```
[ ]: X = data.drop('Diabetes_binary', axis=1)
     Y = data['Diabetes_binary']
```

```
[ ]: X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2,␣
     ↪random_state=1)
```

```
[ ]: X_train.shape
[ ]: (56553, 16)
```

```
[ ]: X_test.shape
```

```
[ ]: (14139, 16)
```

```
[ ]: sc = StandardScaler()

     X_train = sc.fit_transform(X_train)

     X_test = sc.transform(X_test)
```

0.3.5 .

0.3.6 .

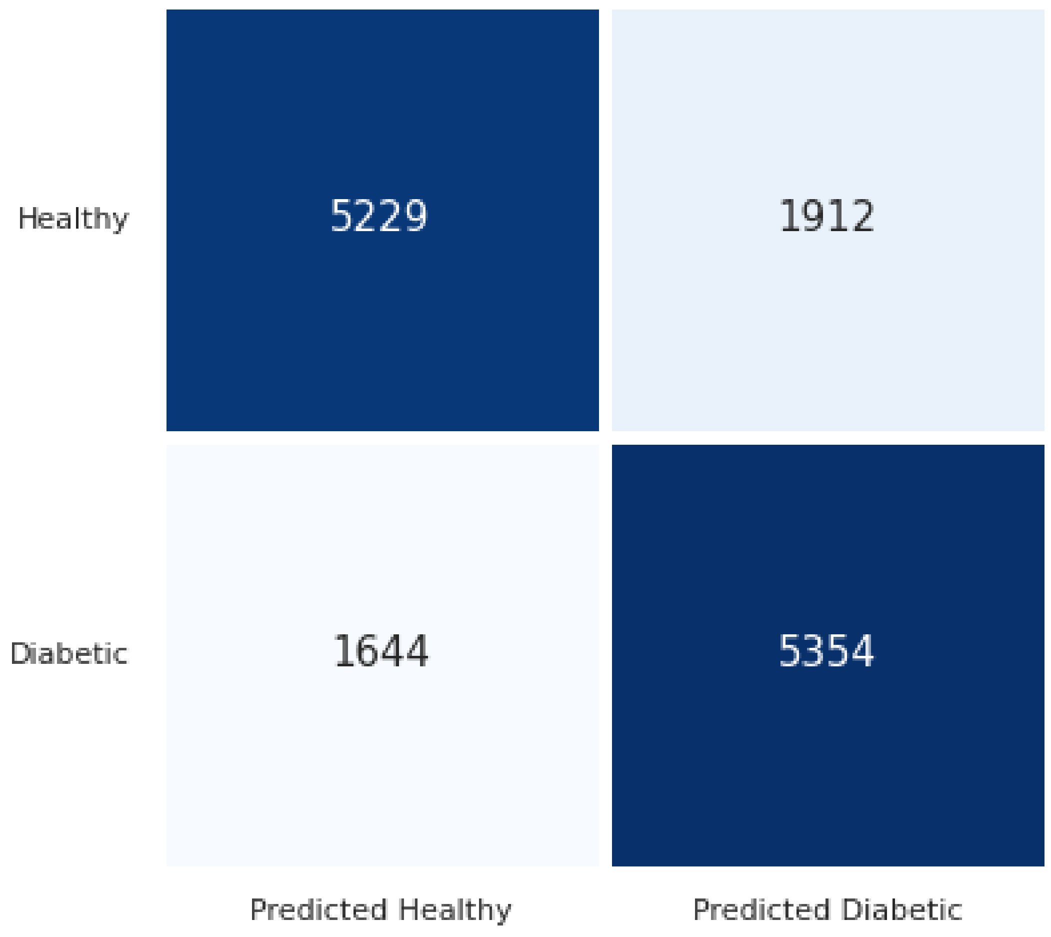
```
[ ]: models = {  'LogisticRegression': LogisticRegression(),
                 'KNearestNeighbors': KNeighborsClassifier(n_neighbors=5),
                 'DecisionTree': DecisionTreeClassifier(),
                 'RandomForest': RandomForestClassifier(),
                 'GradientBoost': GradientBoostingClassifier()}

     accuracies = {}
```

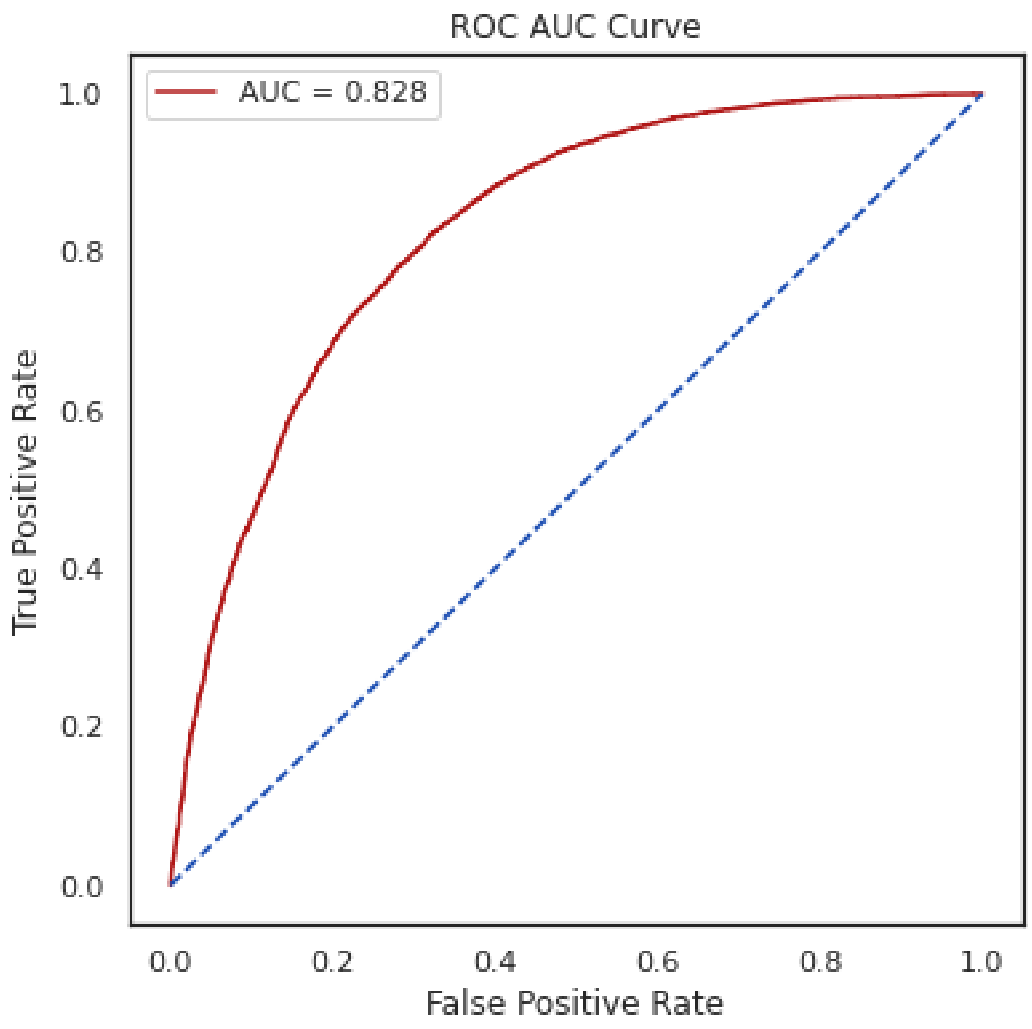
```
[ ]: def DrawGraphics(Y_test, y_pred):
         print("**************************************************************")
         print(model_name)
         print("**************************************************************")
         print(classification_report(Y_test, y_pred))
         print(f'ROC AUC score: {roc_auc_score(Y_test, y_prob)}')
         print('Accuracy Score: ',accuracy_score(Y_test, y_pred))

         plt.figure(figsize = (6, 6))
         sns.heatmap(cm, cmap = 'Blues', annot = True, fmt = 'd', linewidths = 5,␣
      ↪cbar = False, annot_kws = {'fontsize': 15},
                 yticklabels = ['Healthy', 'Diabetic'], xticklabels = ['Predicted␣
      ↪Healthy', 'Predicted Diabetic'])
         plt.yticks(rotation = 0)
         plt.show()

         false_positive_rate, true_positive_rate, thresholds = roc_curve(Y_test,␣
      ↪y_prob)
         roc_auc = auc(false_positive_rate, true_positive_rate)

         sns.set_theme(style = 'white')
         plt.figure(figsize = (6, 6))
```

```
    plt.plot(false_positive_rate,true_positive_rate, color = '#b01717', label =␣
↪'AUC = %0.3f' % roc_auc)
    plt.legend(loc = 'lower right')
    plt.plot([0, 1], [0, 1], linestyle = '--', color = '#174ab0')
    plt.axis('tight')
    plt.ylabel('True Positive Rate')
    plt.xlabel('False Positive Rate')
    plt.title('ROC AUC Curve')
    plt.legend()
    plt.show()
```

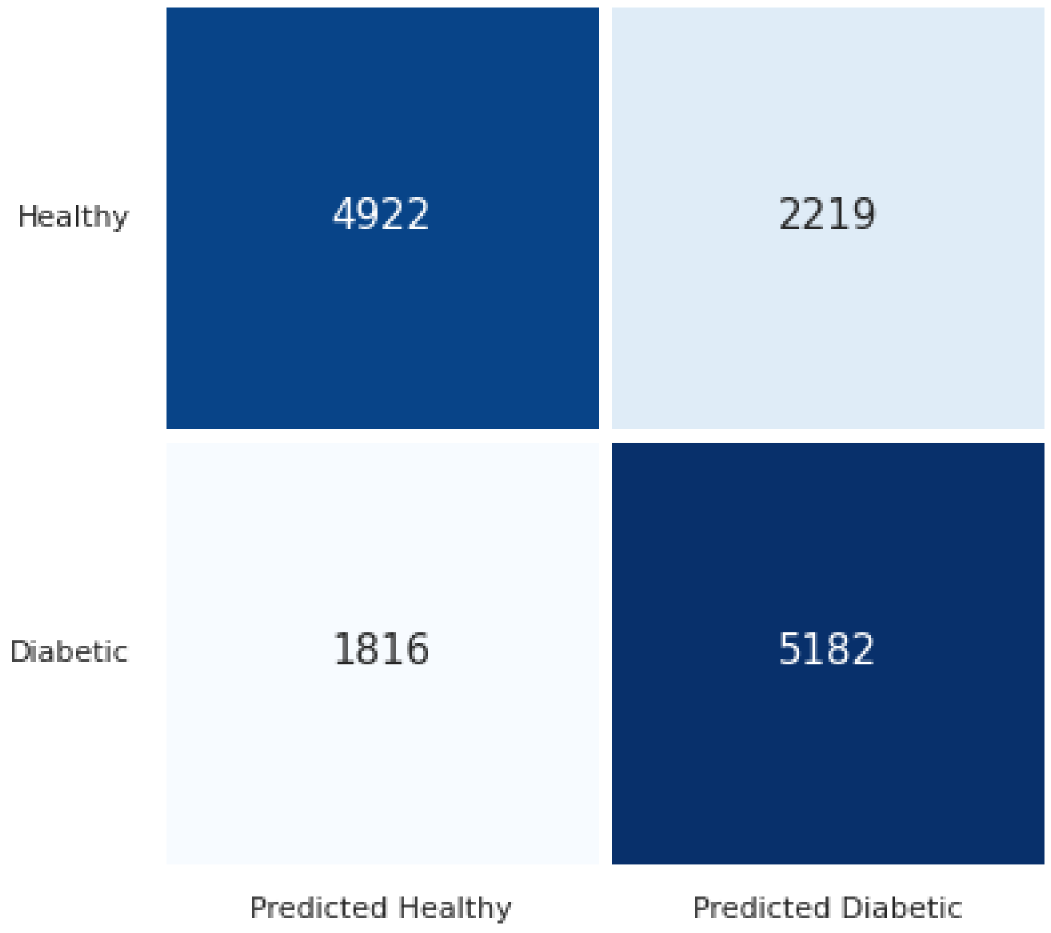
[ ]:
```
for model_name, model in models.items():
    model.fit(X_train, Y_train)
    y_pred = model.predict(X_test)
    y_prob = model.predict_proba(X_test)[:,1]
    cm = confusion_matrix(Y_test, y_pred)

    DrawGraphics(Y_test, y_pred)

    acc = accuracy_score(Y_test, y_pred)*100
    accuracies[model_name] = acc
```

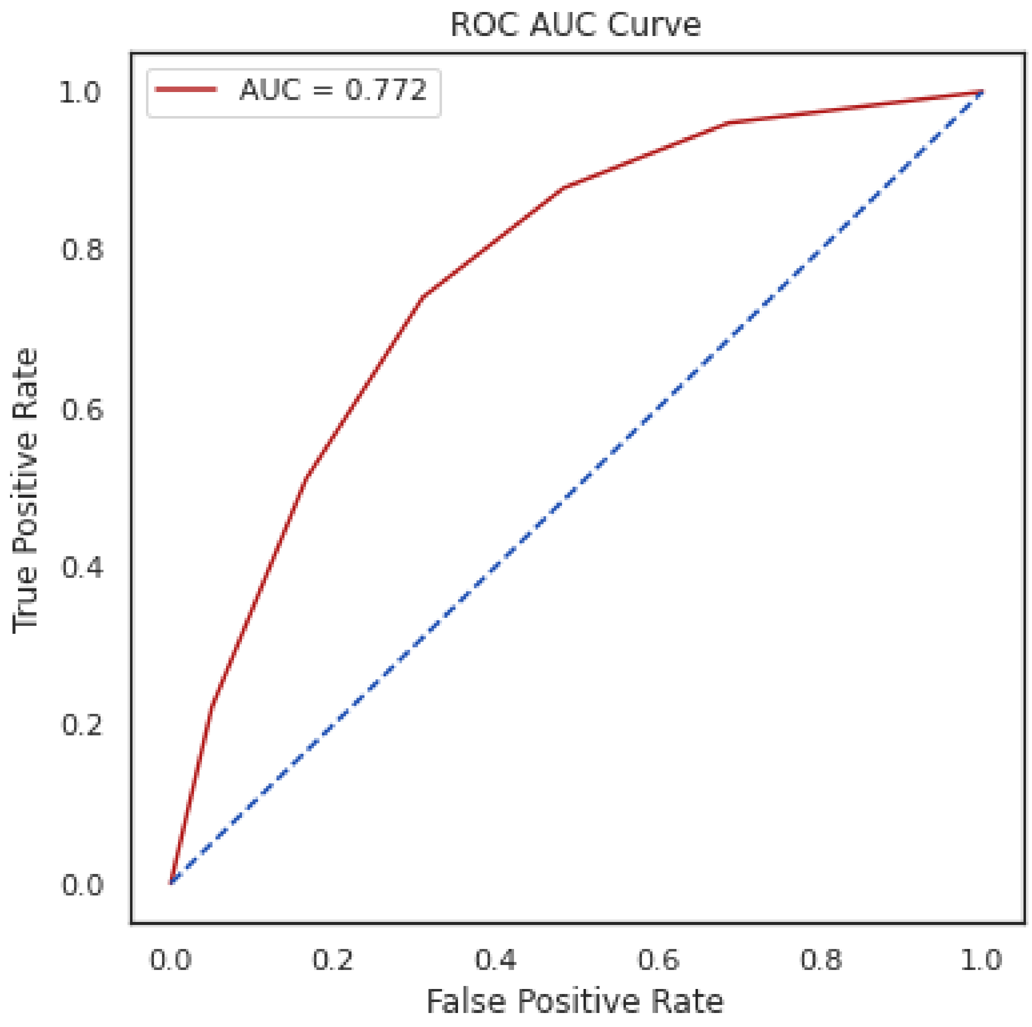
```
**************************************************************
LogisticRegression
**************************************************************
              precision    recall  f1-score   support

           0       0.76      0.73      0.75      7141
           1       0.74      0.77      0.75      6998

    accuracy                           0.75     14139
   macro avg       0.75      0.75      0.75     14139
weighted avg       0.75      0.75      0.75     14139

ROC AUC score: 0.8276305683433108
Accuracy Score:  0.7484970648560718
```


Healthy
Diabetic
5229
1912
1644
5354
Predicted Healthy
Predicted Diabetic

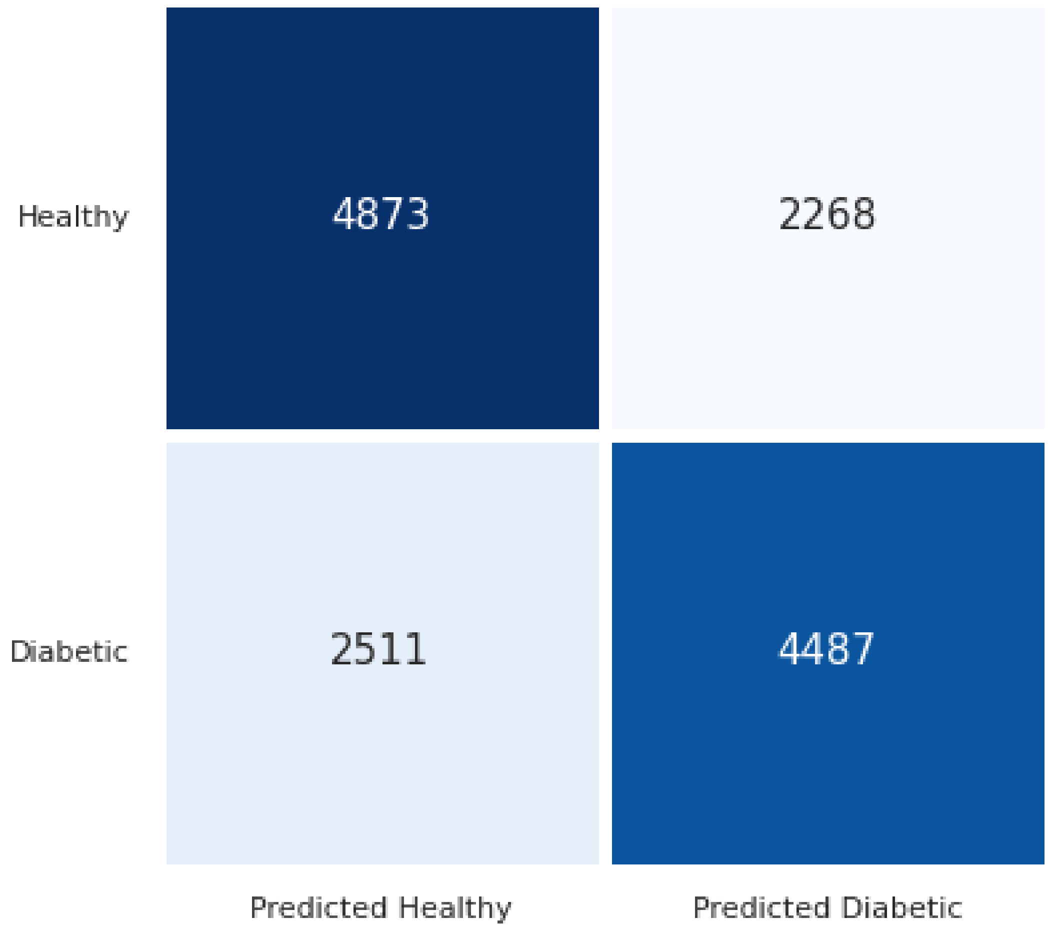


```
************************************************************
KNearestNeighbors
************************************************************
              precision    recall  f1-score   support

           0       0.73      0.69      0.71      7141
           1       0.70      0.74      0.72      6998

    accuracy                           0.71     14139
   macro avg       0.72      0.71      0.71     14139
weighted avg       0.72      0.71      0.71     14139

ROC AUC score: 0.7720235069063084
Accuracy Score:  0.7146191385529387
```

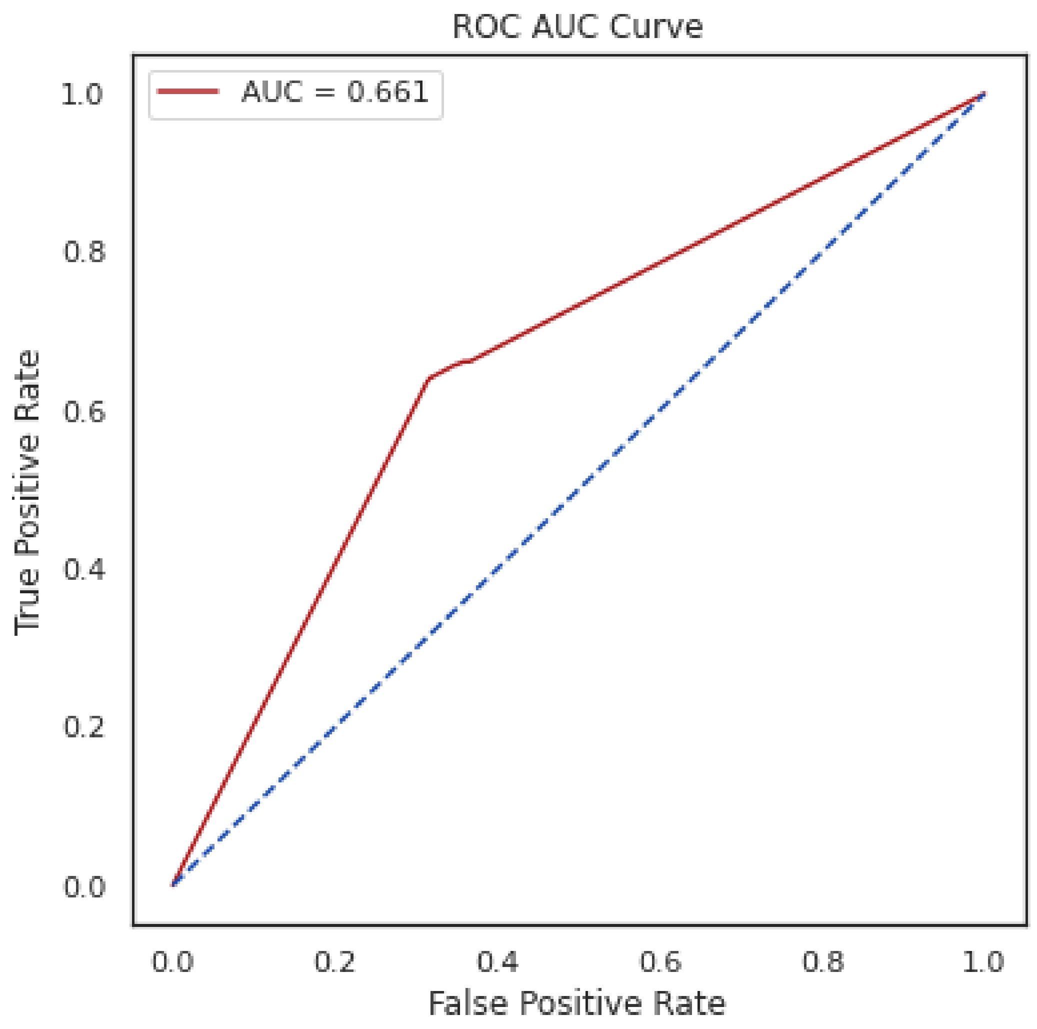

Healthy
Diabetic
4922
2219
1816
5182
Predicted Healthy
Predicted Diabetic



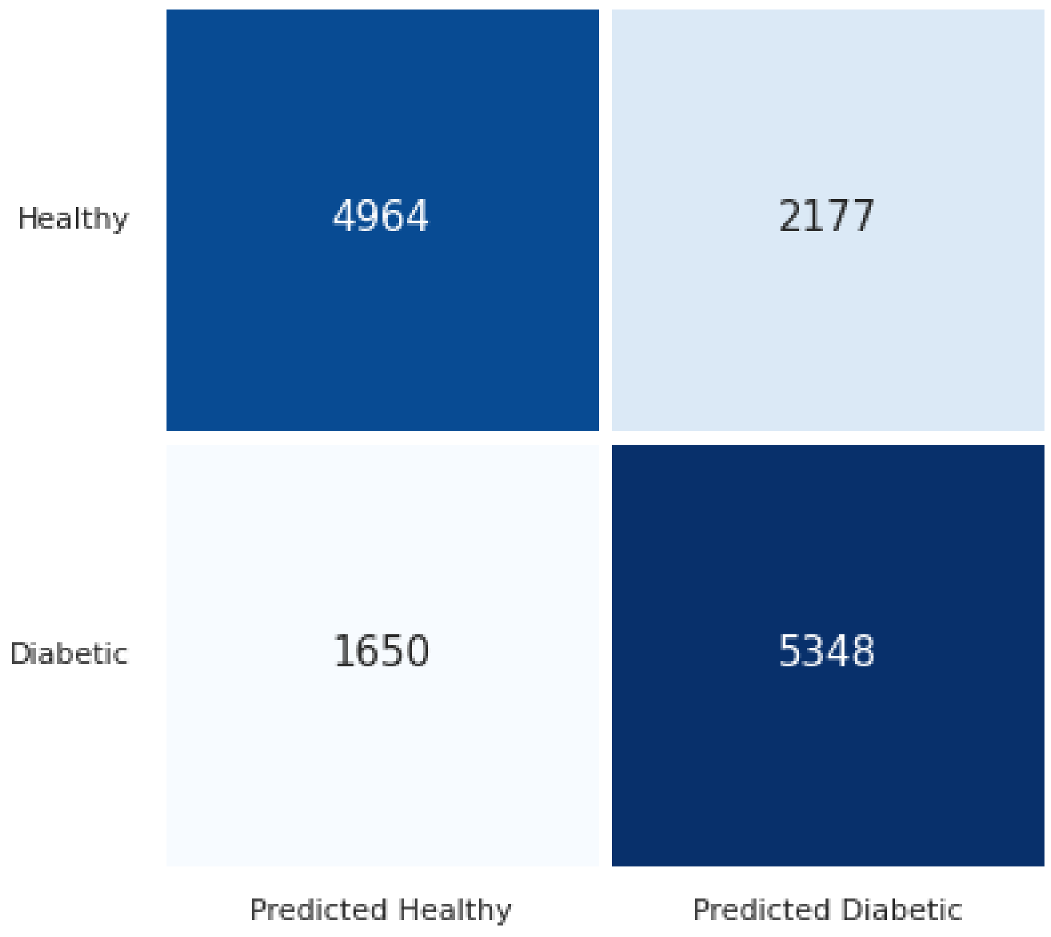
```
**************************************************************
DecisionTree
**************************************************************
              precision    recall  f1-score   support

           0       0.66      0.68      0.67      7141
           1       0.66      0.64      0.65      6998

    accuracy                           0.66     14139
   macro avg       0.66      0.66      0.66     14139
weighted avg       0.66      0.66      0.66     14139

ROC AUC score: 0.6607924047677376
Accuracy Score:  0.6619987269255252
```

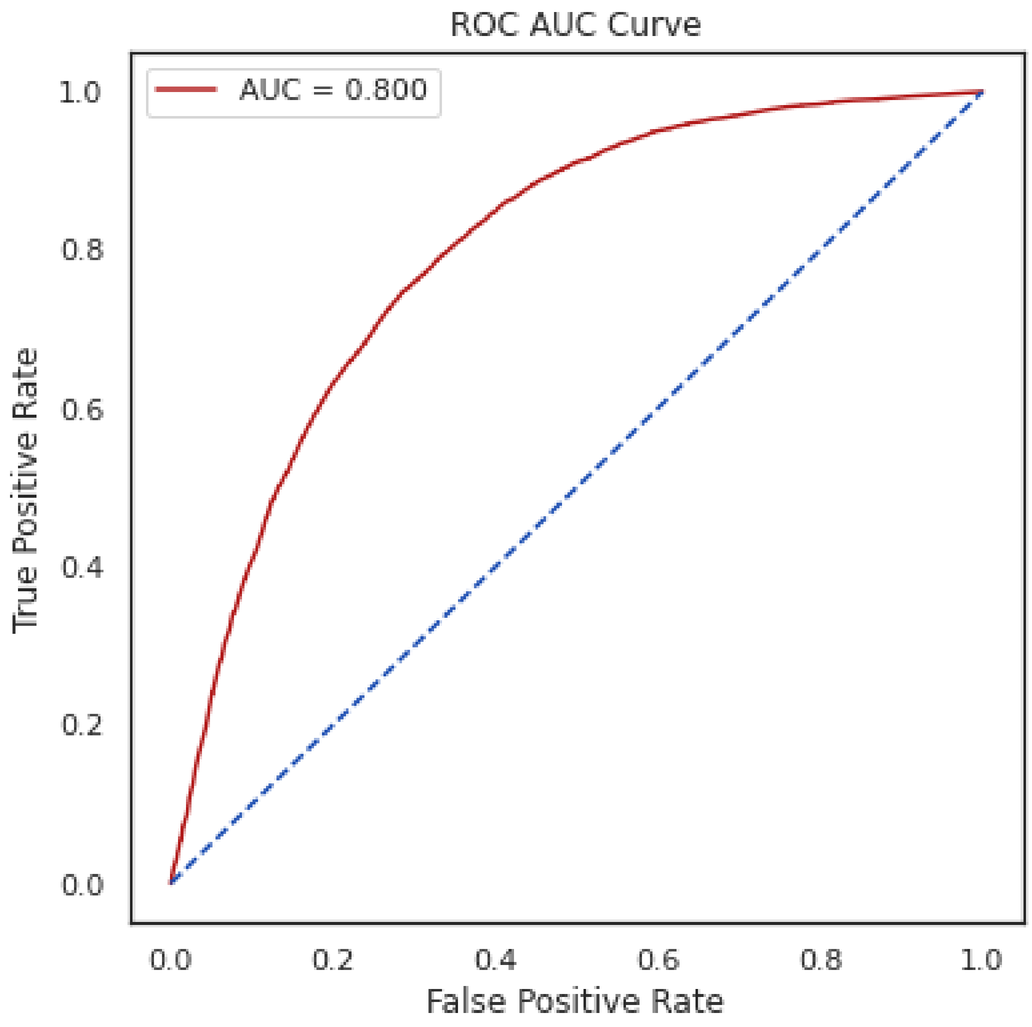


Healthy
Diabetic
4873
2268
2511
4487
Predicted Healthy
Predicted Diabetic

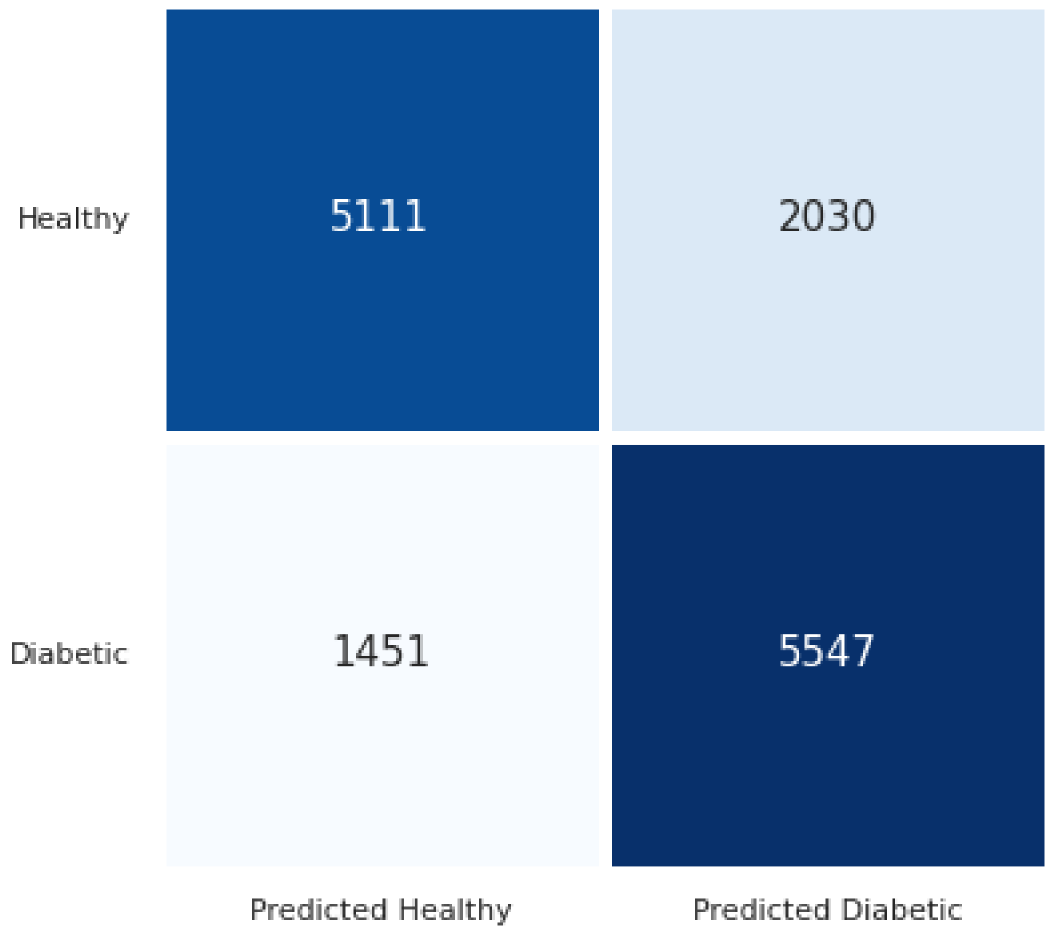


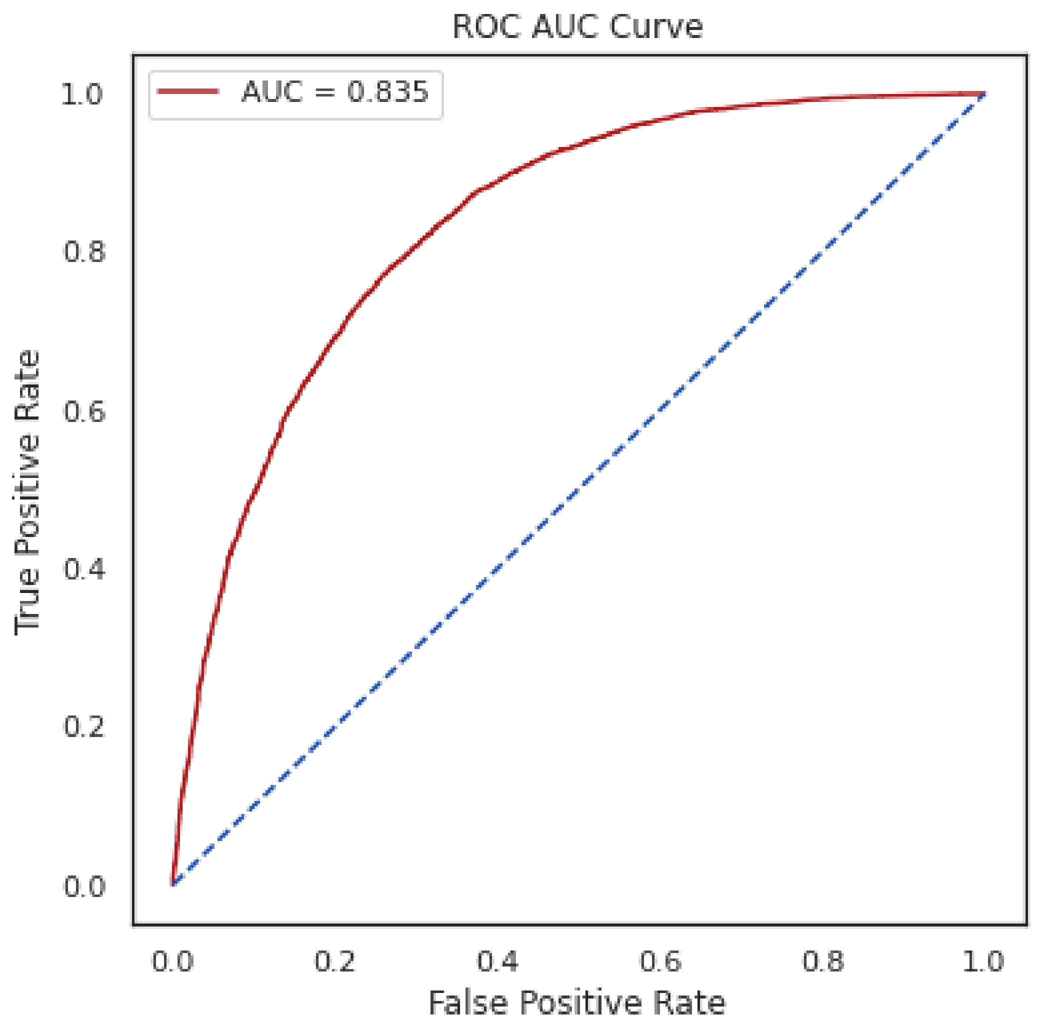
```
************************************************************
RandomForest
************************************************************
              precision    recall  f1-score   support

           0       0.75      0.70      0.72      7141
           1       0.71      0.76      0.74      6998

    accuracy                           0.73     14139
   macro avg       0.73      0.73      0.73     14139
weighted avg       0.73      0.73      0.73     14139

ROC AUC score: 0.7999799910823342
Accuracy Score:  0.7293302213735059
```


Healthy
Diabetic
4964
2177
1650
5348
Predicted Healthy
Predicted Diabetic



```
************************************************************
GradientBoost
************************************************************
              precision    recall  f1-score   support

           0       0.78      0.72      0.75      7141
           1       0.73      0.79      0.76      6998

    accuracy                           0.75     14139
   macro avg       0.76      0.75      0.75     14139
weighted avg       0.76      0.75      0.75     14139

ROC AUC score: 0.8347393171610158
Accuracy Score:  0.7538015418346418
```


Healthy
Diabetic
5111
2030
1451
5547
Predicted Healthy
Predicted Diabetic



```
[ ]: n_range_list = list(range(0,250,50))
     n_range_list[0] = 1
     n_range_list
```

```
[ ]: [1, 50, 100, 150, 200]
```

```
[ ]: grid_models = [(LogisticRegression(),[{'C':[0.25,0.5,0.75,1],'random_state':
       ↪[0]}]),
                    (KNeighborsClassifier(),[{'n_neighbors':n_range_list}]),
                    (DecisionTreeClassifier(),[{'criterion':
       ↪['gini','entropy'],'random_state':[0]}]),
                    (RandomForestClassifier(),[{'n_estimators':
       ↪n_range_list,'criterion':['gini','entropy'],'random_state':[0]}]),
                    (GradientBoostingClassifier(),[{'n_estimators':
       ↪n_range_list,'criterion':['friedman_mse','mse'],'loss':
       ↪['deviance','exponential'],'learning_rate':[0.1, 0.5, 0.8, 1],'random_state':
       ↪[0]}]))
```

```
[ ]: for i,j in grid_models:
         grid = GridSearchCV(estimator=i,param_grid = j, scoring = 'accuracy',cv=2)
         grid.fit(X_train, Y_train)
         best_accuracy = grid.best_score_
         best_param = grid.best_params_
         print('{}:\nBest Accuracy : {:.2f}%'.format(i,best_accuracy*100))
         print('Best Parameters : ',best_param)
         print('')
         print('----------------')
         print('')
```

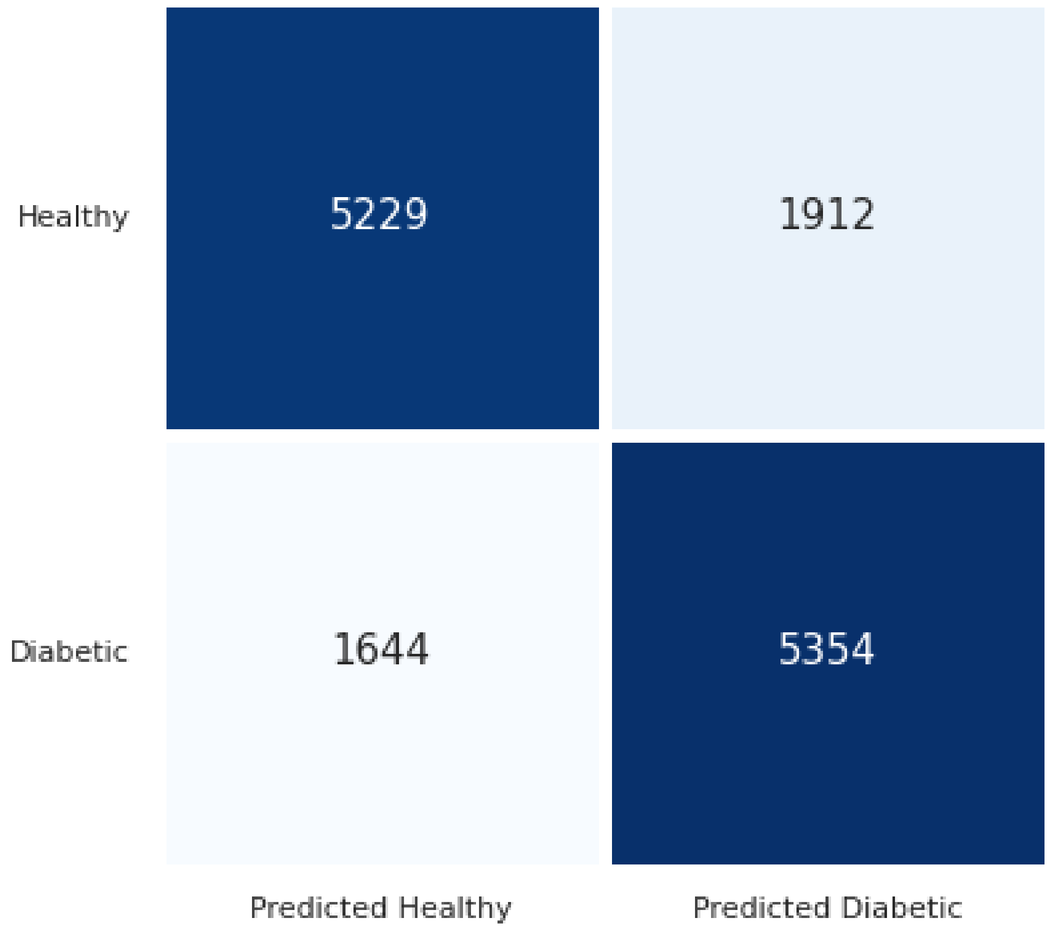
```
LogisticRegression():
Best Accuracy : 74.67%
Best Parameters :  {'C': 0.5, 'random_state': 0}

----------------

KNeighborsClassifier():
Best Accuracy : 74.08%
Best Parameters :  {'n_neighbors': 150}

----------------

DecisionTreeClassifier():
Best Accuracy : 65.79%
Best Parameters :  {'criterion': 'entropy', 'random_state': 0}

----------------

RandomForestClassifier():
Best Accuracy : 72.84%
Best Parameters :  {'criterion': 'entropy', 'n_estimators': 200, 'random_state':
0}

----------------

GradientBoostingClassifier():
Best Accuracy : 75.01%
Best Parameters :  {'criterion': 'friedman_mse', 'learning_rate': 0.1, 'loss':
'exponential', 'n_estimators': 200, 'random_state': 0}

----------------
```

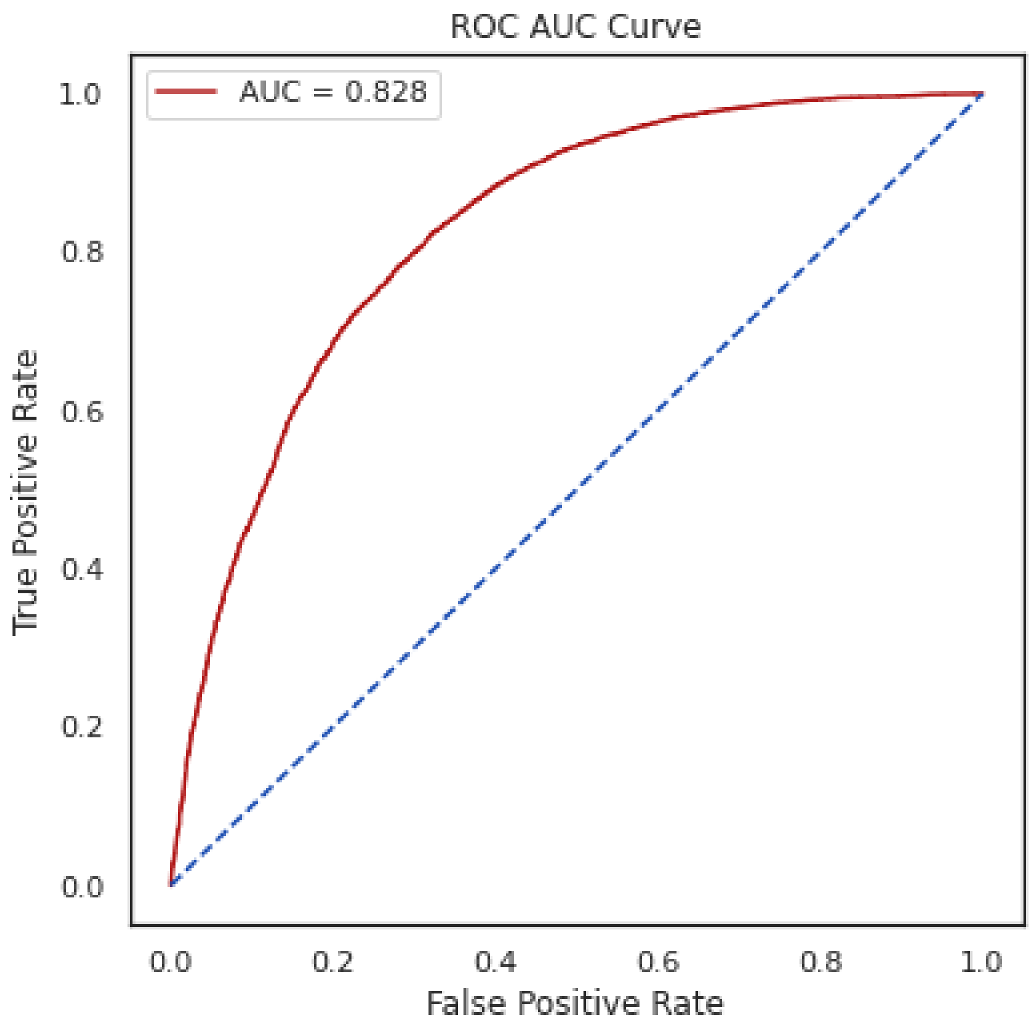
[ ]:
```
params_models = {  'LogisticRegression': LogisticRegression(C = 0.5,␣
↪random_state= 0),
             'KNearestNeighbors': KNeighborsClassifier(n_neighbors=150),
             'DecisionTree':␣
↪DecisionTreeClassifier(criterion='entropy',random_state=0),
             'RandomForest':␣
↪RandomForestClassifier(criterion='gini',n_estimators=200,random_state=0),
             'GradientBoost':␣
↪GradientBoostingClassifier(criterion='friedman_mse',learning_rate=0.
↪1,loss='exponential',n_estimators=200,random_state=0)}

params_accuracies = {}
params_precision = {}
params_recall = {}
params_f1 = {}
params_roc_auc = {}
```

[ ]:
```
for model_name, model in params_models.items():
    model.fit(X_train, Y_train)
    y_pred = model.predict(X_test)
    y_prob = model.predict_proba(X_test)[:,1]
    cm = confusion_matrix(Y_test, y_pred)

    DrawGraphics(Y_test, y_pred)

    param_acc = accuracy_score(Y_test, y_pred)*100
    params_accuracies[model_name] = param_acc
    params_precision[model_name] = precision_score(Y_test, y_pred)
    params_recall[model_name] = recall_score(Y_test, y_pred)
    params_f1[model_name] = f1_score(Y_test, y_pred)
    params_roc_auc[model_name] = roc_auc_score(Y_test,y_pred)
```

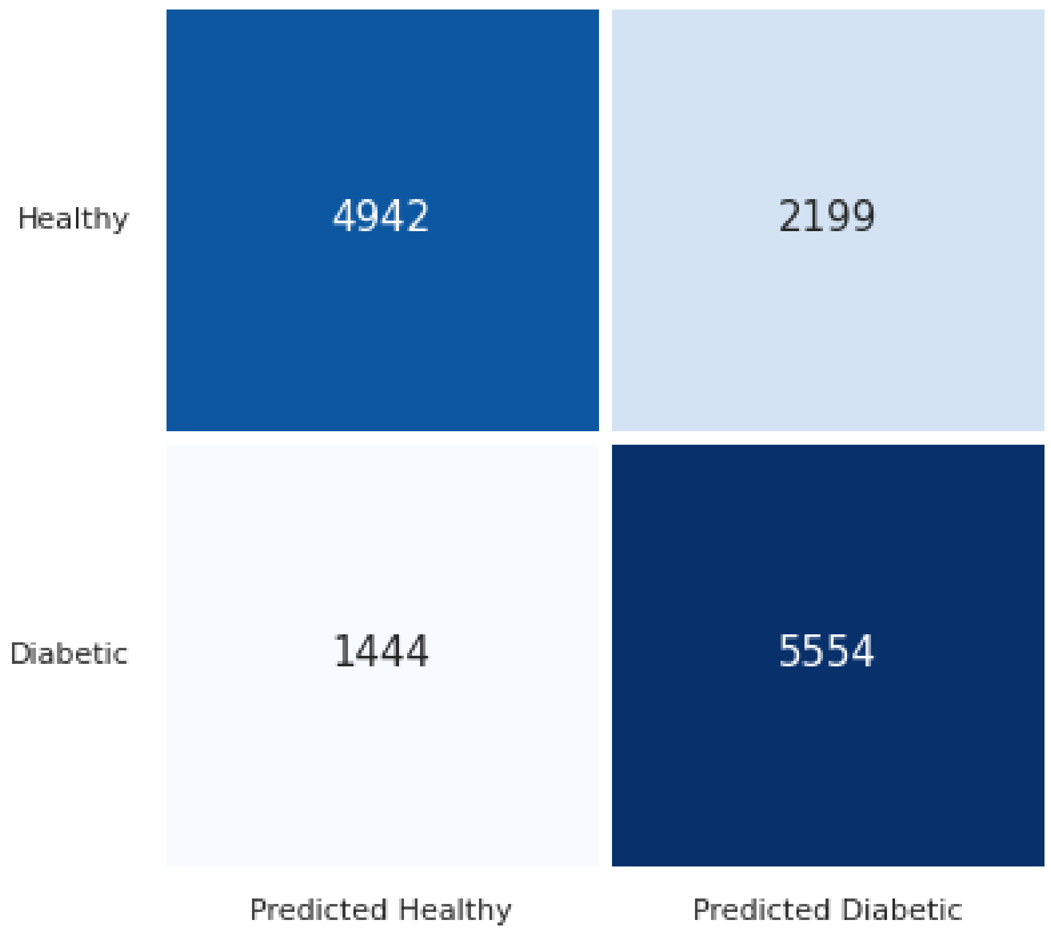
```
**************************************************************
LogisticRegression
**************************************************************
              precision    recall  f1-score   support

           0       0.76      0.73      0.75      7141
           1       0.74      0.77      0.75      6998

    accuracy                           0.75     14139
   macro avg       0.75      0.75      0.75     14139
weighted avg       0.75      0.75      0.75     14139

ROC AUC score: 0.827631468834655
```

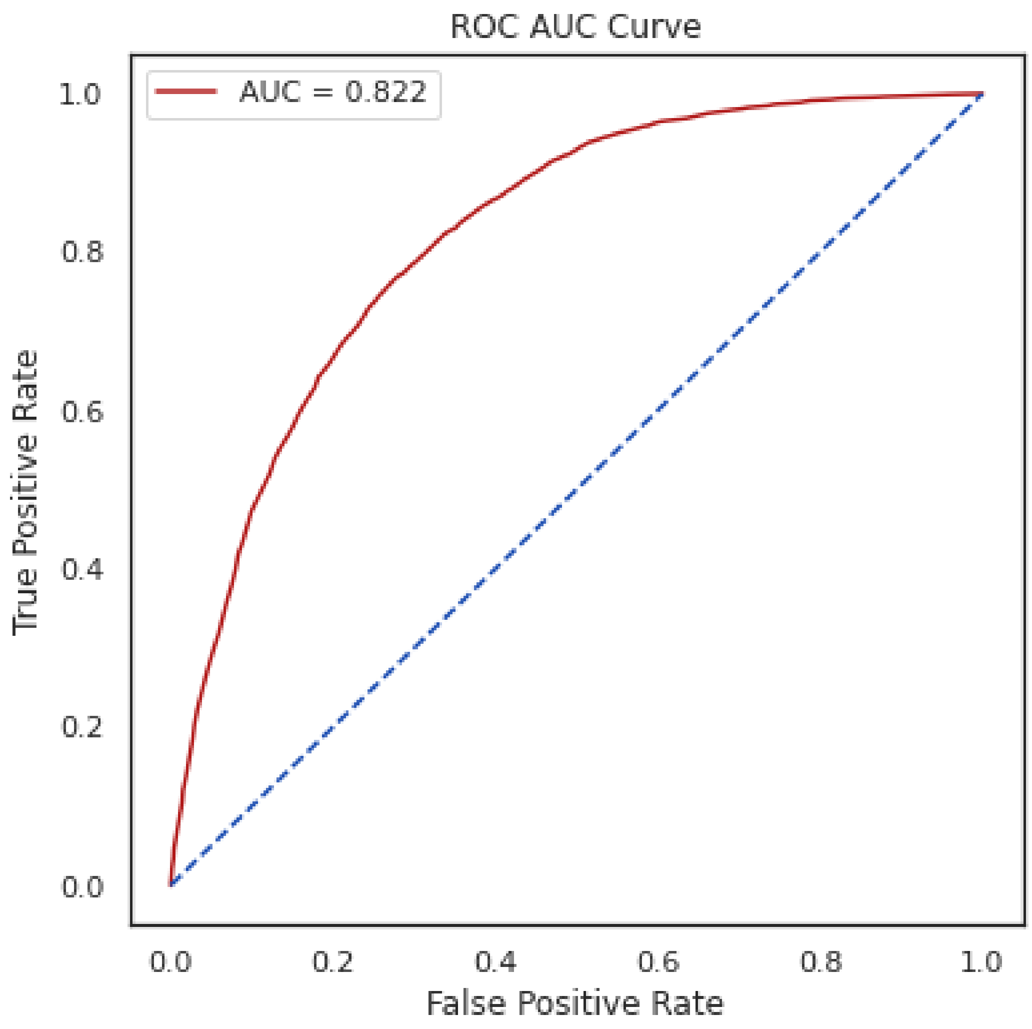
```
Accuracy Score:  0.7484970648560718
```



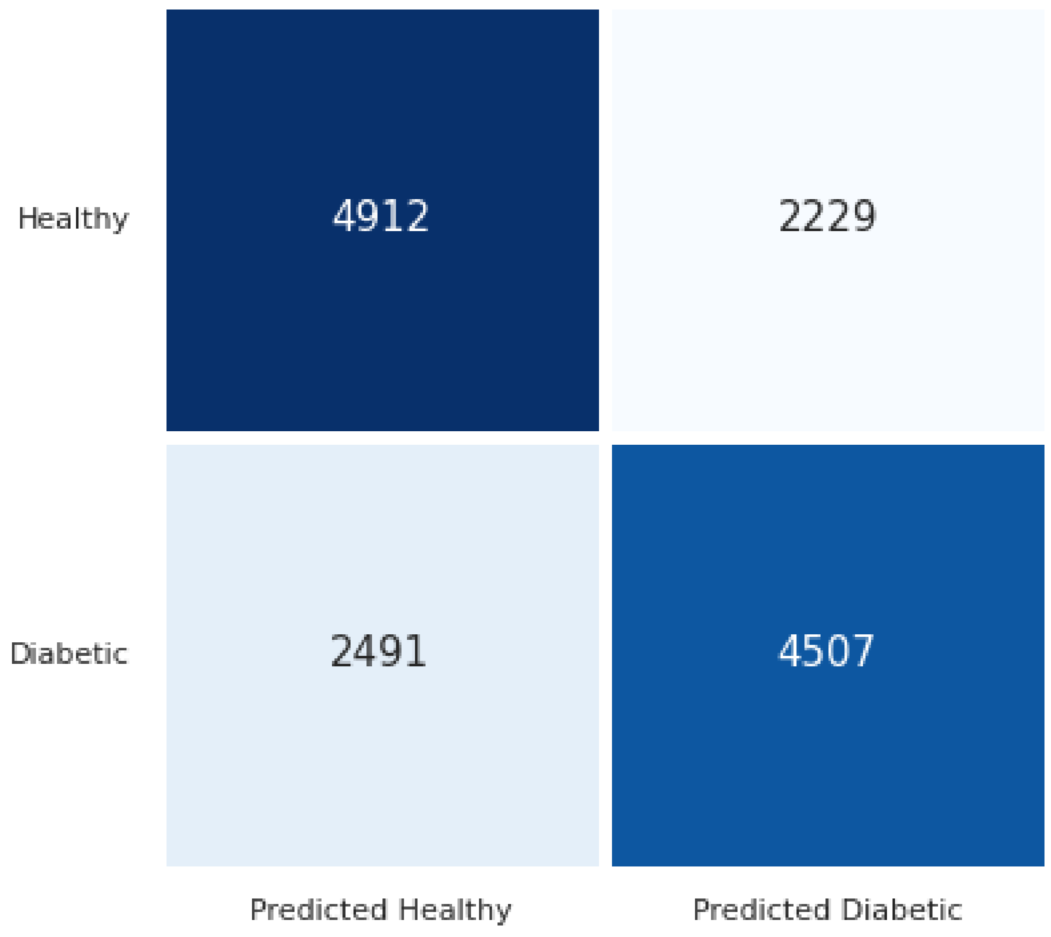


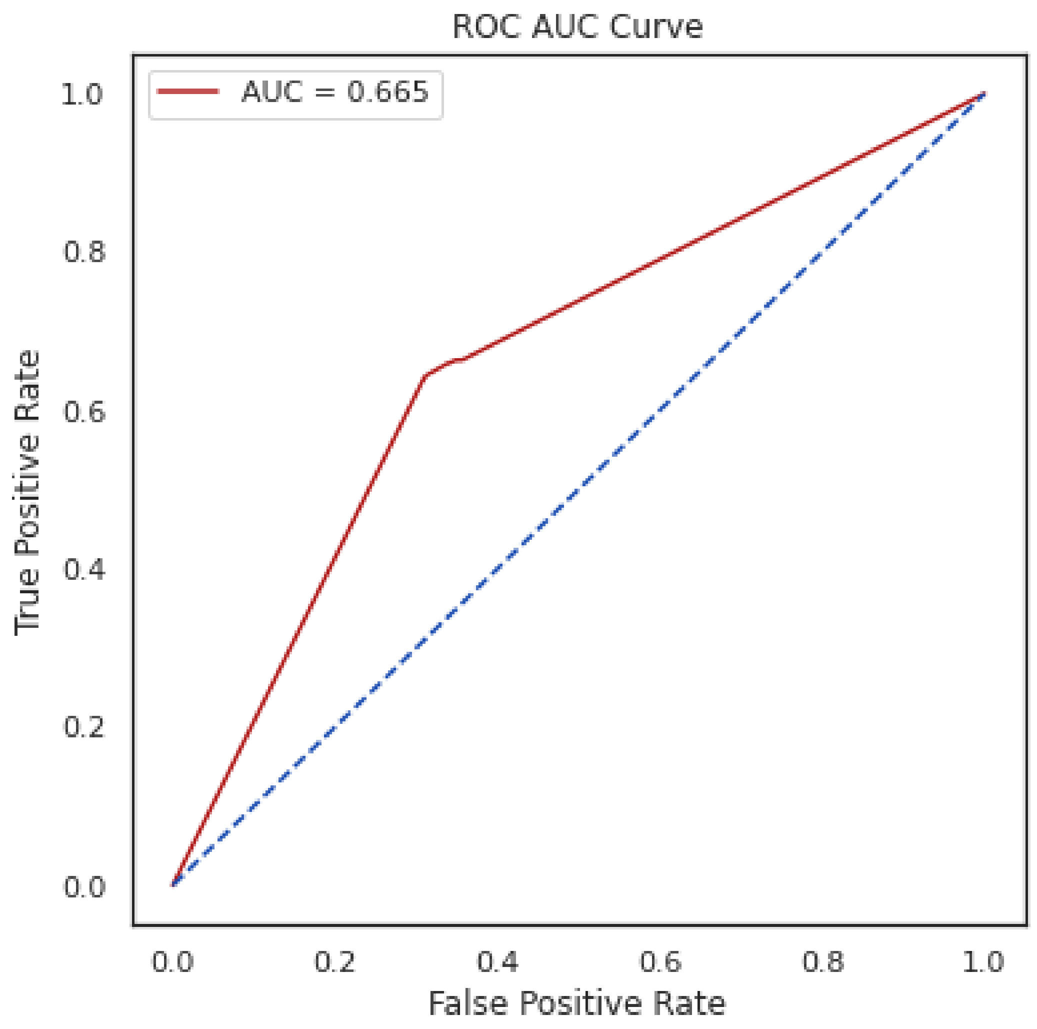
```
*************************************************************
KNearestNeighbors
*************************************************************
              precision    recall  f1-score   support

           0       0.77      0.69      0.73      7141
           1       0.72      0.79      0.75      6998

    accuracy                           0.74     14139
   macro avg       0.75      0.74      0.74     14139
weighted avg       0.75      0.74      0.74     14139

ROC AUC score: 0.8216396794747087
Accuracy Score:  0.7423438715609307
```


Healthy
Diabetic
4942
2199
1444
5554
Predicted Healthy
Predicted Diabetic

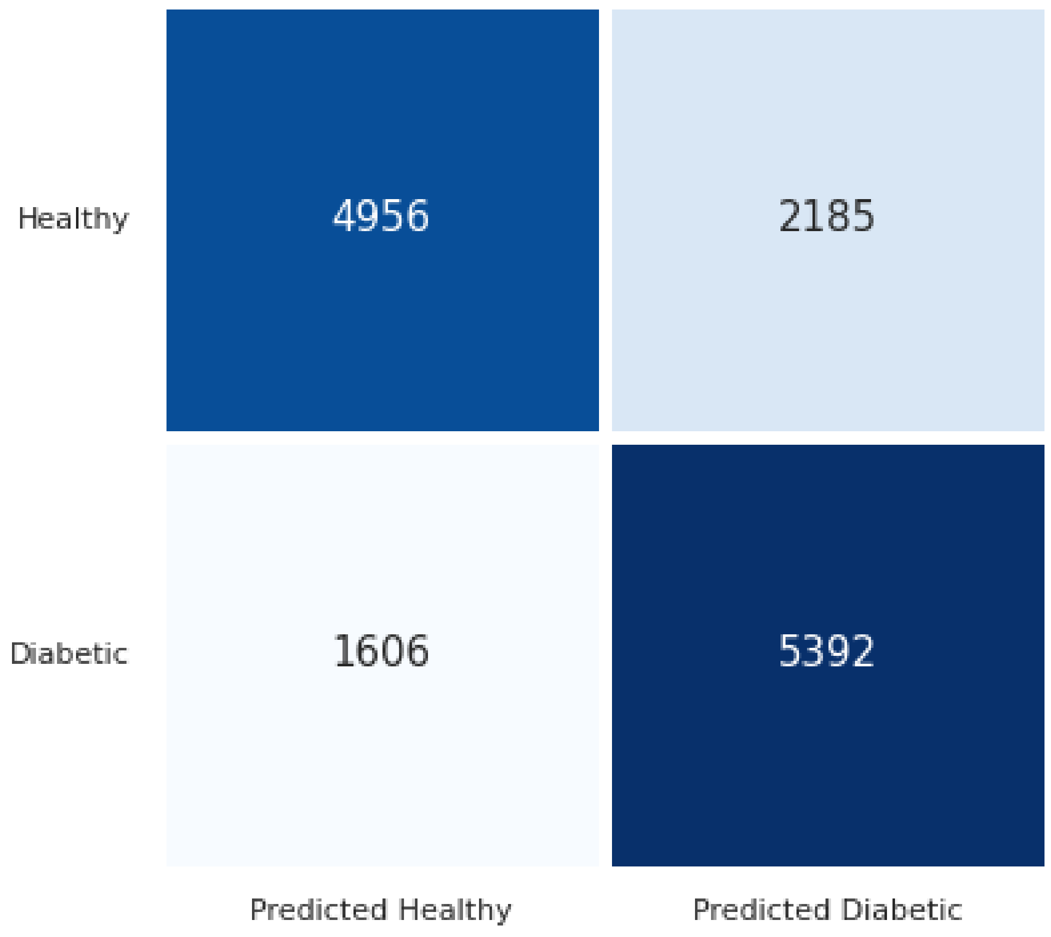


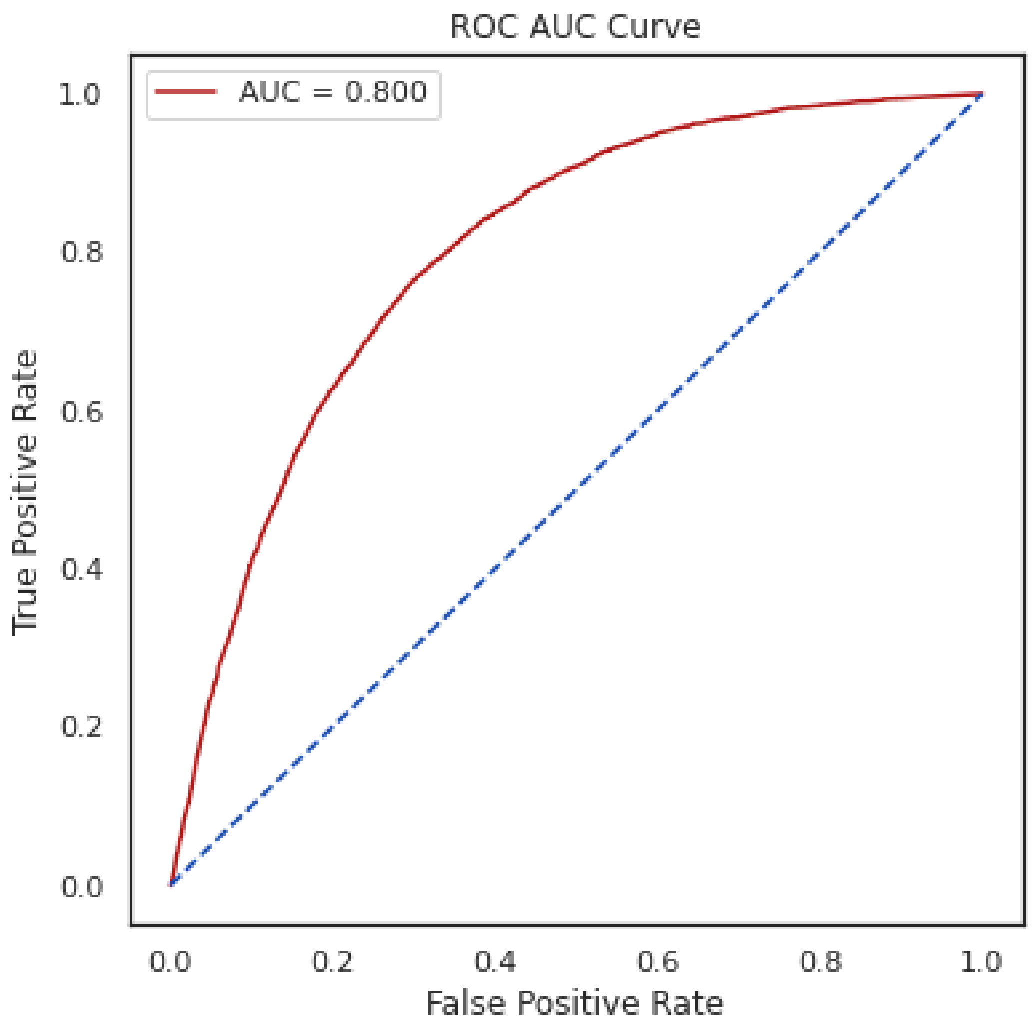
```
**************************************************************
DecisionTree
**************************************************************
              precision    recall  f1-score   support

           0       0.66      0.69      0.68      7141
           1       0.67      0.64      0.66      6998

    accuracy                           0.67     14139
   macro avg       0.67      0.67      0.67     14139
weighted avg       0.67      0.67      0.67     14139

ROC AUC score: 0.6652005200117392
Accuracy Score:  0.6661715821486668
```


Healthy
Diabetic
4912
2229
2491
4507
Predicted Healthy
Predicted Diabetic

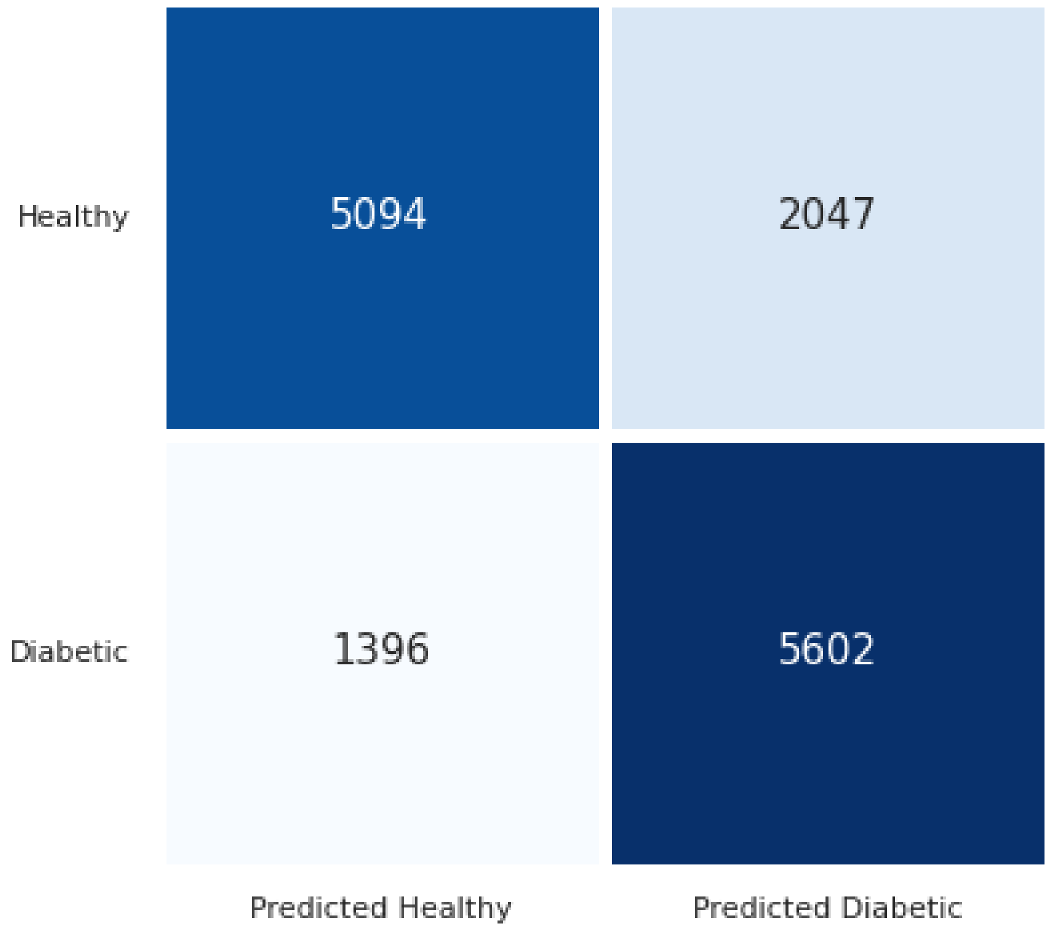


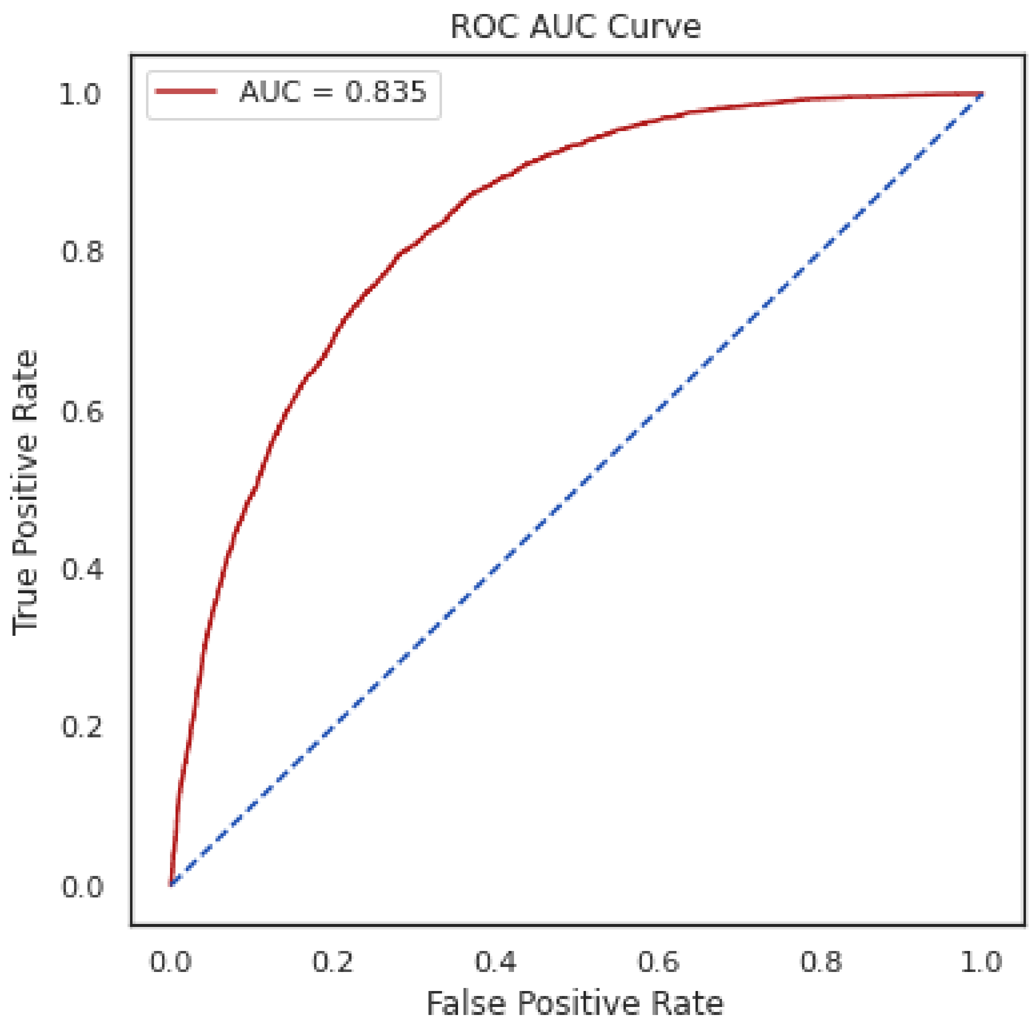
```
**************************************************************
RandomForest
**************************************************************
              precision    recall  f1-score   support

           0       0.76      0.69      0.72      7141
           1       0.71      0.77      0.74      6998

    accuracy                           0.73     14139
   macro avg       0.73      0.73      0.73     14139
weighted avg       0.73      0.73      0.73     14139

ROC AUC score: 0.800058894134996
Accuracy Score:  0.7318763703232195
```


Healthy
4956
2185
Diabetic
1606
5392
Predicted Healthy
Predicted Diabetic

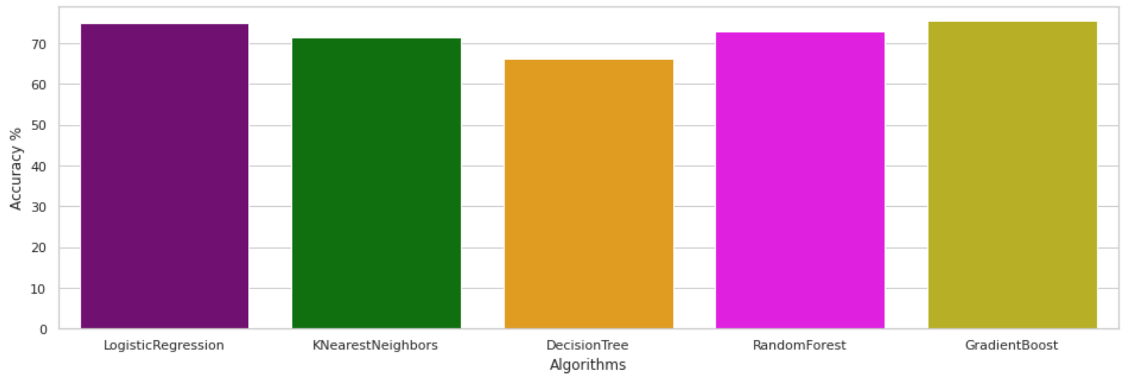


```
************************************************************
GradientBoost
************************************************************
              precision    recall  f1-score   support

           0       0.78      0.71      0.75      7141
           1       0.73      0.80      0.76      6998

    accuracy                           0.76     14139
   macro avg       0.76      0.76      0.76     14139
weighted avg       0.76      0.76      0.76     14139

ROC AUC score: 0.8349524634621635
Accuracy Score:  0.7564891435037838
```

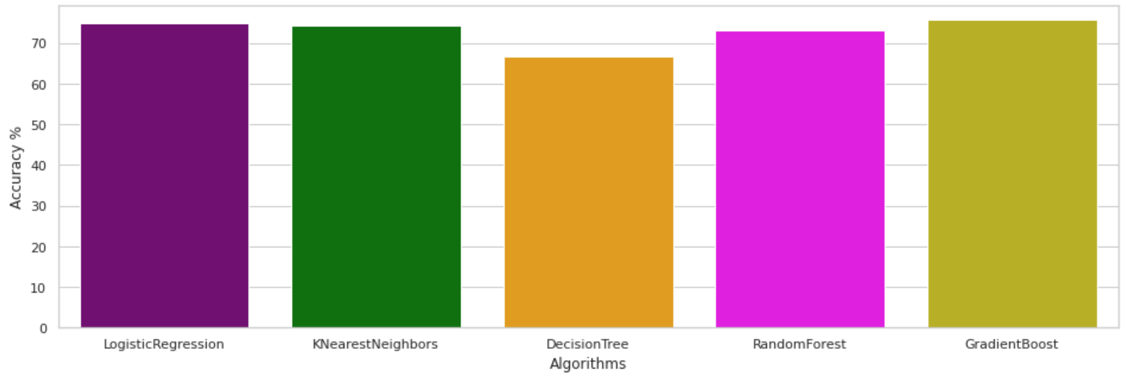

Healthy
Diabetic
5094
2047
1396
5602
Predicted Healthy
Predicted Diabetic



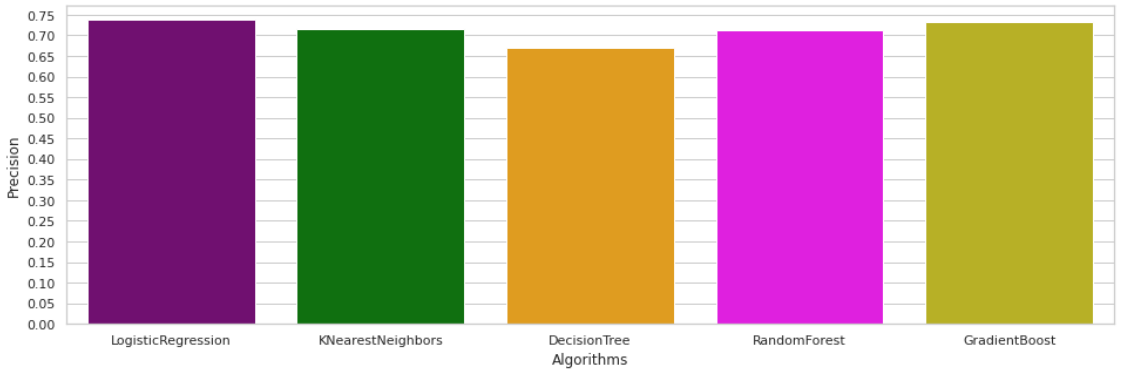
```
[ ]: colors = ["purple", "green", "orange",␣
      ↪"magenta","#CFC60E","#0FBBAE",'#417D7A','#066163','#4D4C7D']

     sns.set_style("whitegrid")
     plt.figure(figsize=(16,5))
     plt.yticks(np.arange(0,100,10))
     plt.ylabel("Accuracy %")
     plt.xlabel("Algorithms")
     sns.barplot(x=list(accuracies.keys()), y=list(accuracies.values()),␣
      ↪palette=colors)
     plt.show()
```



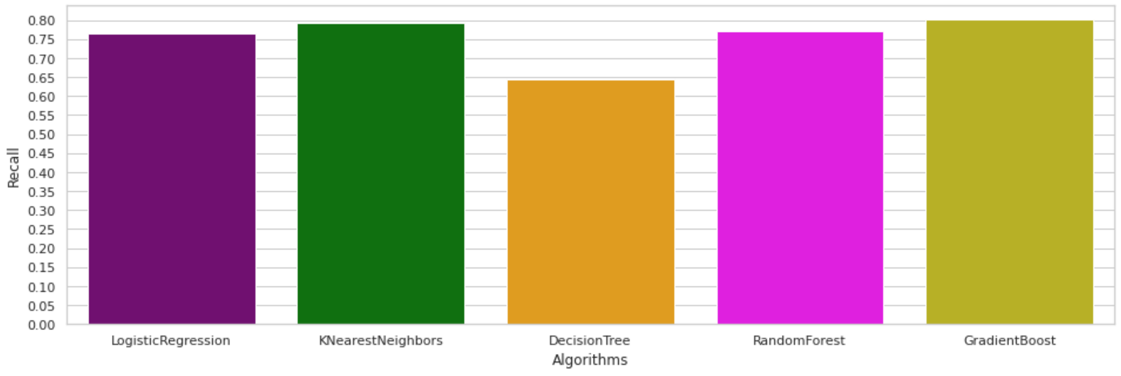
```
[ ]: colors = ["purple", "green", "orange",␣
       ↪"magenta","#CFC60E","#0FBBAE",'#417D7A','#066163','#4D4C7D']

     sns.set_style("whitegrid")
     plt.figure(figsize=(16,5))
     plt.yticks(np.arange(0,100,10))
     plt.ylabel("Accuracy %")
     plt.xlabel("Algorithms")
     sns.barplot(x=list(params_accuracies.keys()), y=list(params_accuracies.
       ↪values()), palette=colors)
     plt.show()
```



```
[ ]: colors = ["purple", "green", "orange",␣
       ↪"magenta","#CFC60E","#0FBBAE",'#417D7A','#066163','#4D4C7D']

     sns.set_style("whitegrid")
     plt.figure(figsize=(16,5))
     plt.yticks(np.arange(0,1,0.05))
     plt.ylabel("Precision")
     plt.xlabel("Algorithms")
     sns.barplot(x=list(params_accuracies.keys()), y=list(params_precision.
       ↪values()), palette=colors)
     plt.show()
```

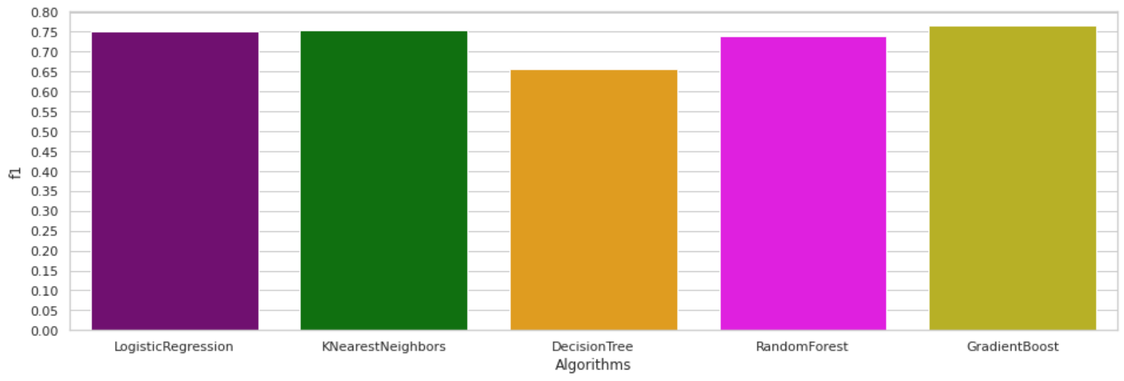


```
[ ]: colors = ["purple", "green", "orange",␣
       ↪"magenta","#CFC60E","#0FBBAE",'#417D7A','#066163','#4D4C7D']

     sns.set_style("whitegrid")
     plt.figure(figsize=(16,5))
     plt.yticks(np.arange(0,1,0.05))
     plt.ylabel("Recall")
     plt.xlabel("Algorithms")
     sns.barplot(x=list(params_accuracies.keys()), y=list(params_recall.values()),␣
       ↪palette=colors)
     plt.show()
```

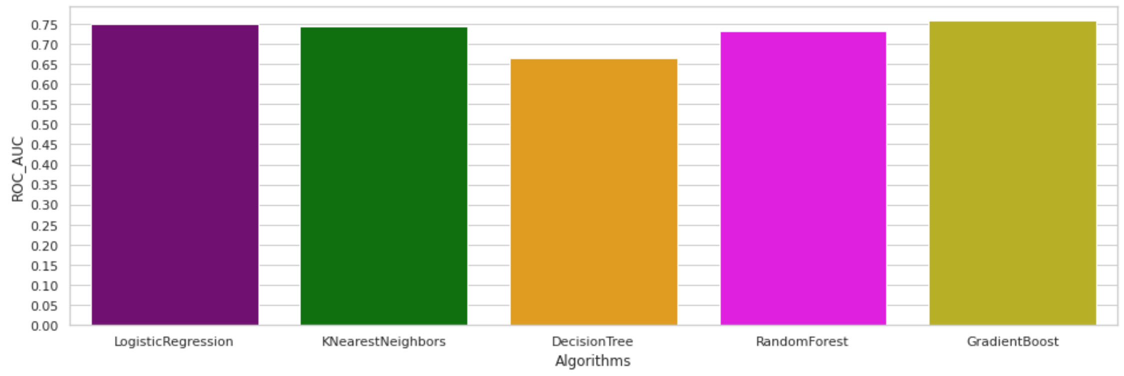


```
[ ]: colors = ["purple", "green", "orange",␣
       ↪"magenta","#CFC60E","#0FBBAE",'#417D7A','#066163','#4D4C7D']
```

```
sns.set_style("whitegrid")
plt.figure(figsize=(16,5))
plt.yticks(np.arange(0,1,0.05))
plt.ylabel("f1")
plt.xlabel("Algorithms")
sns.barplot(x=list(params_accuracies.keys()), y=list(params_f1.values()),␣
    ↪palette=colors)
plt.show()
```



[ ]:
```
colors = ["purple", "green", "orange",␣
    ↪"magenta","#CFC60E","#0FBBAE",'#417D7A','#066163','#4D4C7D']

sns.set_style("whitegrid")
plt.figure(figsize=(16,5))
plt.yticks(np.arange(0,1,0.05))
plt.ylabel("ROC_AUC")
plt.xlabel("Algorithms")
sns.barplot(x=list(params_accuracies.keys()), y=list(params_roc_auc.values()),␣
    ↪palette=colors)
plt.show()
```



:   .