diabetes-prediction-model

0.0.1 Diabetes Prediction Model using Stacking Algorithm

```
[ ]:
```

```
[1]: # importing the data.

import pandas as pd
diabetes_data = pd.read_csv(r"C:\Users\DELL\Documents\MACHINE LEARNING\diabetes.
 ↪csv")
print(diabetes_data)
```

```
     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
..           …        …             …             …      …    …
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
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
..                        …  …      …
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

[768 rows x 9 columns]
```

[2]:
```
# to check whether data is balanced or not?

diabetes_data['Outcome'].value_counts()
```

[2]:
```
Outcome
0    500
1    268
Name: count, dtype: int64
```

Hence, it's an imbalanced dataset, imbalanced towards class 0 i.e. No diabetes cases. So now we'll perform a sampling technique to balance the dataset.

[3]:
```
yes_diabetes=diabetes_data[diabetes_data.Outcome==1]
yes_diabetes.head()
```

[3]:
```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
2            8      183             64              0        0  23.3
4            0      137             40             35      168  43.1
6            3       78             50             32       88  31.0
8            2      197             70             45      543  30.5

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
2                     0.672   32        1
4                     2.288   33        1
6                     0.248   26        1
8                     0.158   53        1
```

[4]:
```
yes_diabetes.shape
```

[4]:
```
(268, 9)
```

[5]:
```
no_diabetes=diabetes_data[diabetes_data.Outcome==0]
no_diabetes.head()
```

[5]:
```
    Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
1             1       85             66             29        0  26.6
3             1       89             66             23       94  28.1
5             5      116             74              0        0  25.6
7            10      115              0              0        0  35.3
10            4      110             92              0        0  37.6

    DiabetesPedigreeFunction  Age  Outcome
1                      0.351   31        0
3                      0.167   21        0
5                      0.201   30        0
```

```
7                        0.134   29         0
10                       0.191   30         0
```

[6]:
```
no_diabetes.shape
```

[6]:
```
(500, 9)
```

[7]:
```
# upsampling the minority class i.e. outcome=1 cases.

from sklearn.utils import resample
yes_diabetes_upsampled = resample(yes_diabetes, replace=True, n_samples=470)
yes_diabetes_upsampled.shape
```

[7]:
```
(470, 9)
```

[33]:
```
# combining the upsampled data with no diabetes case from original data.

diabetes_data_new=pd.concat([yes_diabetes_upsampled,no_diabetes])
diabetes_data_new.shape
```

[33]:
```
(970, 9)
```

[37]:
```
# Shuffling the new combined data.

from sklearn.utils import shuffle
diabetes_data_new=shuffle(diabetes_data_new)
diabetes_data_new.head()
```

[37]:
```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
314            7      109             80             31        0  35.9
520            2       68             70             32       66  25.0
250            9      106             52              0        0  31.2
261            3      141              0              0        0  30.0
372            0       84             64             22       66  35.8

     DiabetesPedigreeFunction  Age  Outcome
314                     1.127   43        1
520                     0.187   25        0
250                     0.380   42        0
261                     0.761   27        1
372                     0.545   21        0
```

[38]:
```
# Selecting target variable

y = diabetes_data_new['Outcome']
y.head()
```

[38]:
```
314    1
520    0
250    0
261    1
372    0
Name: Outcome, dtype: int64
```

[39]:
```
# Selecting features.

x = diabetes_data_new.drop('Outcome', axis=1)
x.head()
```

[39]:
```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
314            7      109             80             31        0  35.9
520            2       68             70             32       66  25.0
250            9      106             52              0        0  31.2
261            3      141              0              0        0  30.0
372            0       84             64             22       66  35.8

     DiabetesPedigreeFunction  Age
314                     1.127   43
520                     0.187   25
250                     0.380   42
261                     0.761   27
372                     0.545   21
```

[40]:
```
#splitting the data into training and testing.

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.2,␣
 ↪random_state=42)
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

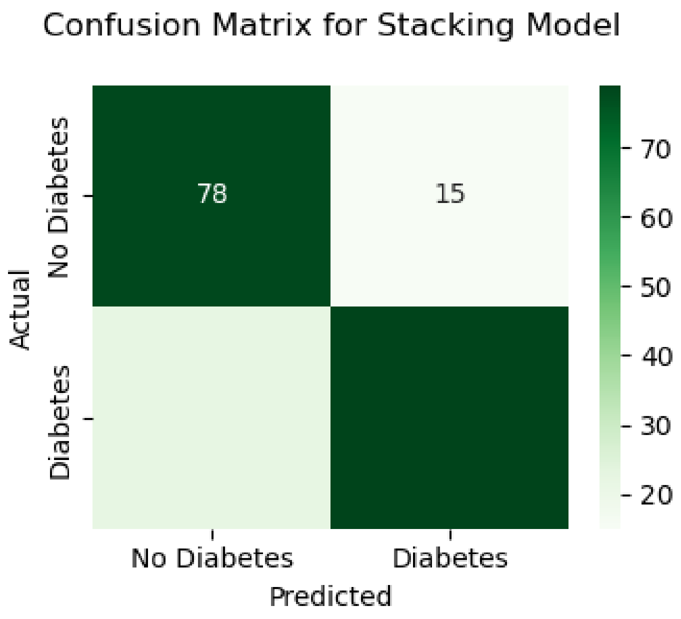
[40]:
```
((776, 8), (194, 8), (776,), (194,))
```

[41]:
```
# creating stacking model.

from sklearn.ensemble import AdaBoostClassifier, RandomForestClassifier,␣
 ↪StackingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report,confusion_matrix,roc_auc_score


# Defining base estimators.
ada = AdaBoostClassifier(algorithm='SAMME',n_estimators=50, random_state=42)
rf = RandomForestClassifier(n_estimators=100, random_state=42)
```

```python
# Defining final estimator.
dt = DecisionTreeClassifier(random_state=42)

# Defining the stacking classifier model.
stacking_model = StackingClassifier(
    estimators=[('ada', ada), ('rf', rf)],
    final_estimator=dt,
    cv=5
)

# Training the stacking classifier model.
stacking_model.fit(x_train, y_train)

# Making predictions.
y_pred = stacking_model.predict(x_test)

# Generating classification report
print('STACKING MODEL - CLASSIFICATION REPORT\n')
report = classification_report(y_test, y_pred, target_names=['No Diabetes',␣
 ↪'Diabetes'])
print(report)
cm=confusion_matrix(y_test,y_pred)
print('STACKING MODEL - CONFUSION MATRIX\n')
print(cm)
print('')
score=roc_auc_score(y_test,y_pred)
print('STACKING MODEL - ROC AUC SCORE: ',score)
```

```
STACKING MODEL - CLASSIFICATION REPORT

              precision    recall  f1-score   support

 No Diabetes       0.78      0.84      0.81        93
    Diabetes       0.84      0.78      0.81       101

    accuracy                           0.81       194
   macro avg       0.81      0.81      0.81       194
weighted avg       0.81      0.81      0.81       194

STACKING MODEL - CONFUSION MATRIX

[[78 15]
 [22 79]]

STACKING MODEL - ROC AUC SCORE:  0.8104439476205685
```

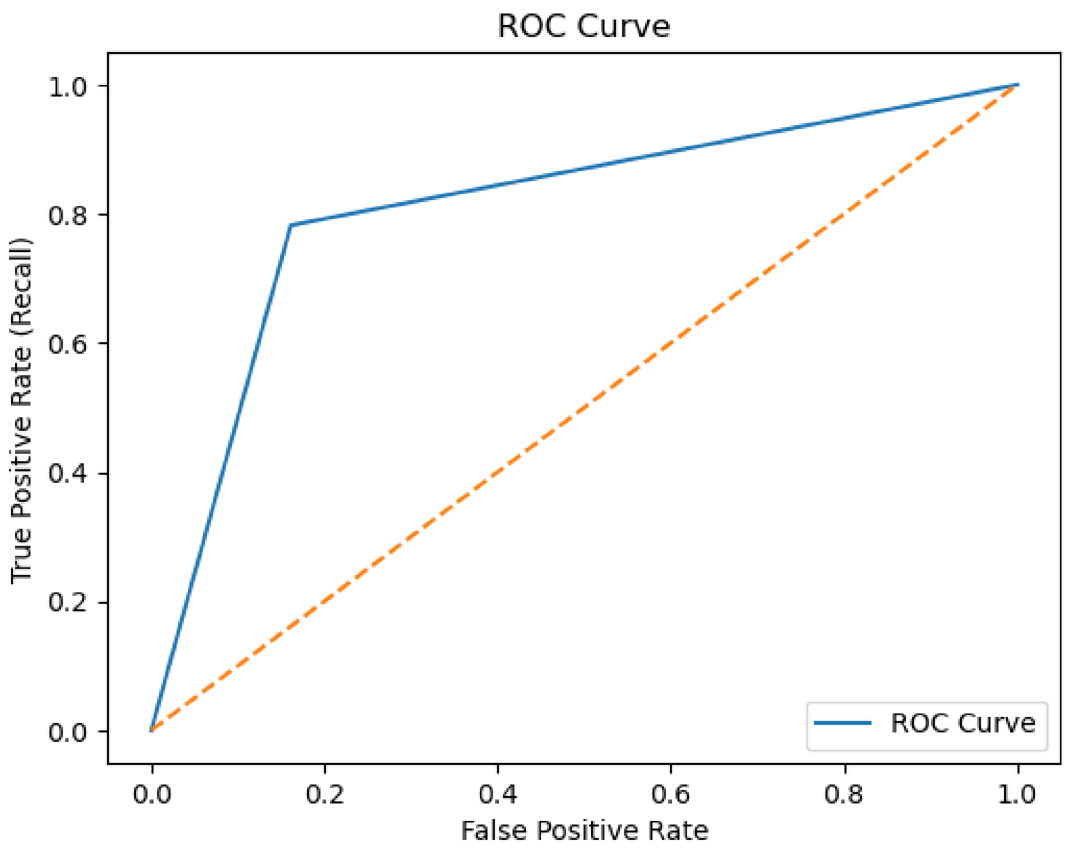
[44]:
```
# Visualization of confusion matrix.

import matplotlib.pyplot as plt
import seaborn as sns

cm=confusion_matrix(y_test,y_pred)

labels = ['No Diabetes', 'Diabetes']
plt.figure(figsize=(4, 3))
sns.heatmap(cm, annot=True, fmt='d', cmap='Greens', xticklabels=labels,␣
 ↪yticklabels=labels)
plt.title('Confusion Matrix for Stacking Model\n')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```



[ ]:
```

```

[76]:
```
# Plotting ROC Curve.

import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, roc_auc_score

y_pred = stacking_model.predict(x_test)
```

```
fpr, tpr,_= roc_curve(y_test, y_pred)
plt.plot(fpr,tpr);
plt.plot([0,1],[0,1],linestyle='--');
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate (Recall)')
plt.title('ROC Curve')
plt.legend(['ROC Curve'],loc='lower right')
plt.show()
```



0.1 Analysis.

0.1.1 Analysis of Classification report.

An accuracy of 0.81 means that 81% of the overall predictions by the model are correct.

The precision for predicting No Diabetes is 0.78 means that 78% of the time the model predicting a case as No Diabetes is correct.

The precision for predicting Diabetes is 0.84 means that 84% of the time the model predicting a case as Diabetes is correct.

Recall for No Diabetes is 0.84 means that the model can recognize 84% of all the No Diabetes cases.

Recall for Diabetes is 0.78 means that the model can recognize 78% of all the Diabetes cases.

F-1 scores for No diabetes and Diabetes is 0.81, indicating a good balance between precision and recall for both classes & a balanced model's prediction for both case types.

Support of 93 and 101 samples for No Diabetes and Diabetes classes respectively indicates that the dataset is relatively balanced between both classes.

Hence, the model is performing effectively in predicting both cases.

0.1.2 Analysis of Confusion Matrix.

True Positive(TP)- 79, it's the number of times a diabetes case is correctly predicted as diabetes.

True Negative(TN)- 78, it's the number of times a no diabetes case is correctly predicted as no diabetes.

False Positive(FP)- 15, it's the number of times a no diabetes case is incorrectly predicted as diabetes.

False Negative(FN)- 22, it's the number of times a diabetes case is incorrectly predicted no diabetes

The above values from the confusion matrix indicate that-

The model correctly predicts 79 instances of diabetes and 78 instances of no diabetes.

It incorrectly identifies 15 no diabetes cases as diabetes and 22 diabetes cases as no diabetes.

Hence, the model is effective in predicting both cases but still there is some room for improvement and false positive and false negative cases can be reduced further.

0.1.3 ROC-AUC(Receiver Operating Characteristics-Area Under the Curve) Score-

ROC-AUC Score of 0.8104439476205685 indicates that a model has a good ability to distinguish between both the Diabetes and No Diabetes classes. As ROC-AUC Score ranges from 0 to 1, with 1 indicating perfect classification and 0.5 or below indicating no discriminative power.

Hence, overall the stacking model has good performance and is very effective in predicting both classes, with a slight scope for further improvement.

```
[ ]:
```