Diabetic_classification

October 20, 2020

[26]:
```
import pandas as pd

# Takes the file
filepath = r"diabetes.csv"

# read the CSV file
df = pd.read_csv(filepath)

# print the data set
print(df)
```

```
     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]
```

[7]:
```
print(df['Outcome'].value_counts())
```

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

[11]:
```
# The data set is in-balanced

df_suffle = df.sample(frac=1,random_state=4)

# Get all the Diabetic records
Diabetic_df = df_suffle.loc[df_suffle['Outcome'] == 1]

# Get 270 non-diabetic records
Nondiabetic_df = df_suffle.loc[df_suffle['Outcome'] == 0].
 ↪sample(n=270,random_state=25)

# Concatenate both dataframes again
Balanced_df = pd.concat([Diabetic_df, Nondiabetic_df])

print(Balanced_df)
```

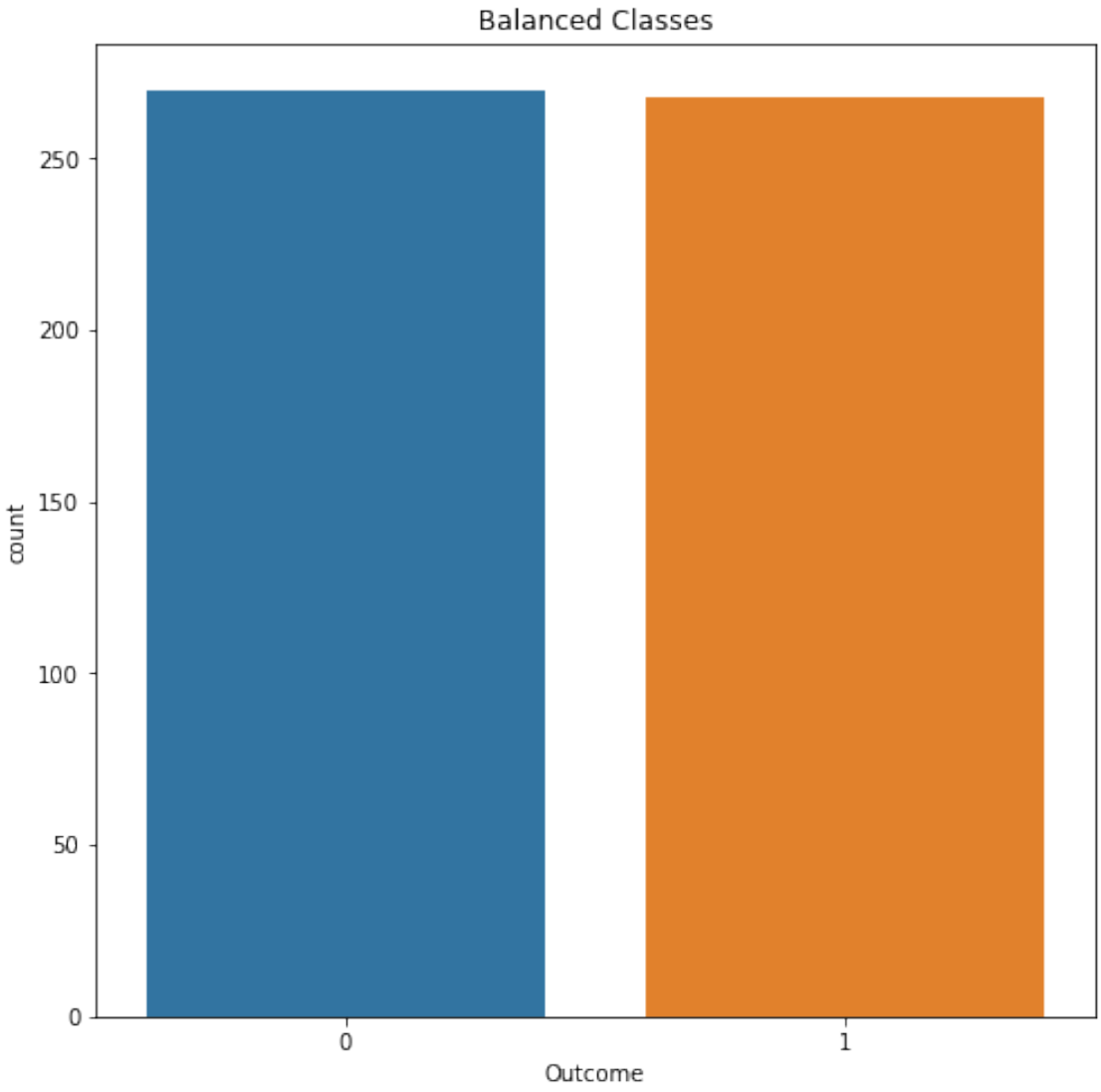
```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
328            2      102             86             36      120  45.5
72            13      126             90              0        0  43.4
110            3      171             72             33      135  33.3
254           12       92             62              7      258  27.6
215           12      151             70             40      271  41.8
..           ...      ...            ...            ...      ...  ...
272            3      122             78              0        0  23.0
463            5       88             78             30        0  27.6
763           10      101             76             48      180  32.9
307            0      137             68             14      148  24.8
573            2       98             60             17      120  34.7

     DiabetesPedigreeFunction  Age  Outcome
328                     0.127   23        1
72                      0.583   42        1
110                     0.199   24        1
254                     0.926   44        1
215                     0.742   38        1
..                        ...  ...      ...
272                     0.254   40        0
463                     0.258   37        0
763                     0.171   63        0
307                     0.143   21        0
573                     0.198   22        0
```

```
[538 rows x 9 columns]
```

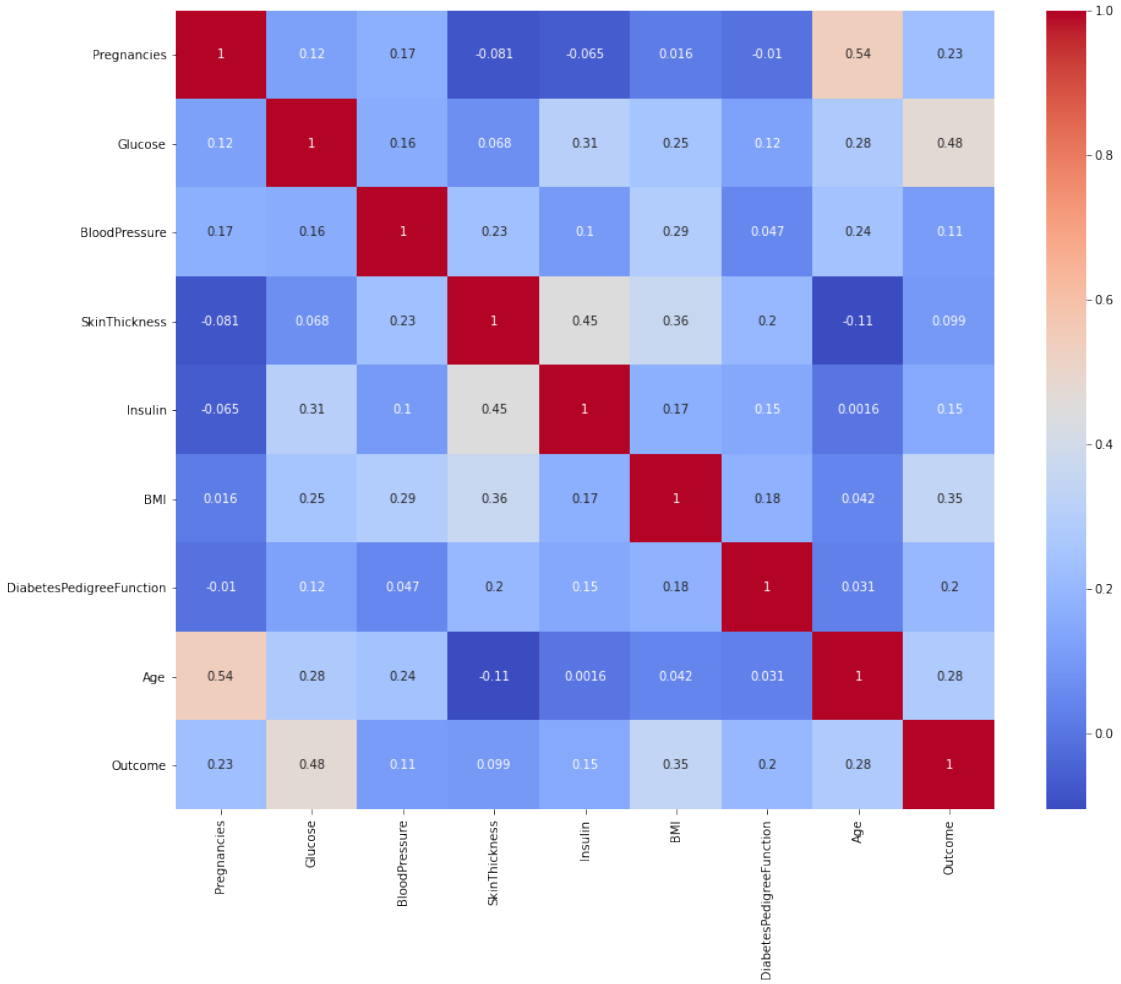
[16]:
```
#plot the dataset
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(8, 8))
sns.countplot('Outcome', data=Balanced_df)
plt.title('Balanced Classes')
plt.show()
```

```
/home/deepak/my_jupyter_project_dir/my_jupyter_project_env/lib/python3.6/site-
packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable
as a keyword arg: x. From version 0.12, the only valid positional argument will
be `data`, and passing other arguments without an explicit keyword will result
in an error or misinterpretation.
  FutureWarning
```



[17]:
```
# Now it is required to find the retationship between two attribute, if two␣
 ↪attributes are highly related
# we can drop one to reduce the computational complexity(but in this data set␣
 ↪we have less records still checking)

#Plot heatmap and show case correlation of each features
correlation_matrix = Balanced_df.corr()
plt.figure(figsize=(15,12))
sns.heatmap(correlation_matrix , annot = True, cmap = "coolwarm")
plt.show()
```



[ ]:
```
# From the plot it is observerd that no attribute is highly related with any␣
 ↪other attribute except dogonal value
# no value is close to .3.Therefore, all attributes will contribute.
```

[21]:
```
#Assigning training and test data
from sklearn.model_selection import train_test_split
X =␣
 ↪Balanced_df[['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','Diabe
Y = Balanced_df.Outcome


X_train, X_test, Y_train,Y_test = train_test_split(X, Y, test_size = 0.25,␣
 ↪random_state = 0)
```

[24]:
```
#Random forest
from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
rf_classifier = RandomForestClassifier()
rf_classifier.fit(X_train, Y_train)
Y_pred = rf_classifier.predict(X_test)
print("Accuracy of the model:",accuracy_score(Y_test, Y_pred))
print(classification_report(Y_test, Y_pred))
```

```
Accuracy of the model: 0.725925925925926
              precision    recall  f1-score   support

           0       0.74      0.73      0.74        71
           1       0.71      0.72      0.71        64

    accuracy                           0.73       135
   macro avg       0.73      0.73      0.73       135
weighted avg       0.73      0.73      0.73       135
```

[25]:
```
from sklearn.svm import SVC
svm_classifier = SVC()
svm_classifier.fit(X_train, Y_train)
Y_pred = svm_classifier.predict(X_test)
print("Accuracy of the model:",accuracy_score(Y_test, Y_pred))
print(classification_report(Y_test, Y_pred))
```

```
Accuracy of the model: 0.6962962962962963
              precision    recall  f1-score   support

           0       0.73      0.68      0.70        71
           1       0.67      0.72      0.69        64

    accuracy                           0.70       135
   macro avg       0.70      0.70      0.70       135
weighted avg       0.70      0.70      0.70       135
```

[ ]:
```

```