Lab - Logistic Regression Classifier: Diabetes Prediction

Importing Important Libraries

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import metrics
from sklearn.metrics import mean_squared_error, r2_score
import pandas as pd
import io
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
from sklearn import preprocessing
plt.rc("font", size = 14)
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
sns.set(style="white")
sns.set(style="whitegrid", color_codes = True)
```

Load and Explore the data

The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset.

Our main focuse will be on "Outcome" column. Outcome: 1 if diabetes, 0 if no diabetes.

```
data = pd.read_csv('/content/diabetes.csv')
```

```
data.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedi
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	

```
data.shape
```

```
(768, 9)
```

Here we perform the selection of the features and the spliting of the data into train and test sets.

```
X = data.drop('Outcome', axis=1) # Features
y = data['Outcome'] # Target values used for predicting Outcome (diabetes 0/1)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
X.columns
```

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

```
y_test.head()
```

```
668    0
324    0
624    0
690    0
473    0
Name: Outcome, dtype: int64
```

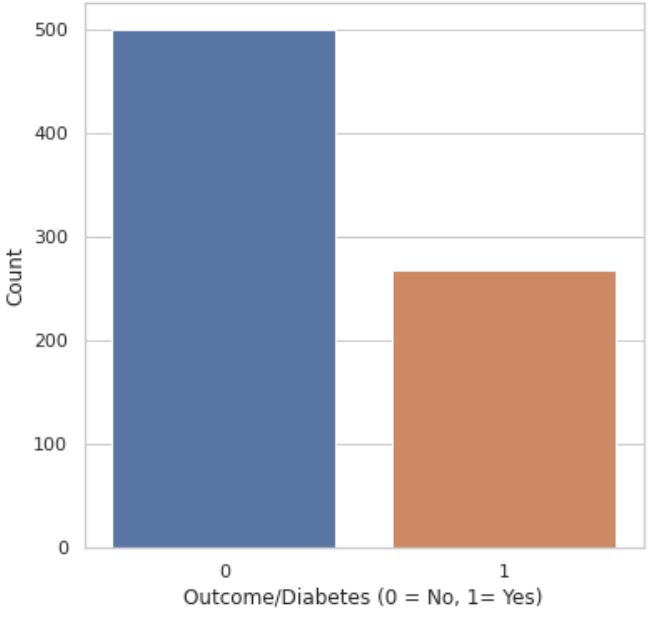
```
y_train.value_counts()
```

```
0    401
1    213
Name: Outcome, dtype: int64
```

Here we plot the countplot for Outcome/Diabetes (0 = No and 1 = Yes)

We can notice from the plot that there are less people who have diabetes compared to those who don't.

```
plt.figure(figsize=(6,6))
sns.countplot(x='Outcome', data=data)
plt.xlabel("Outcome/Diabetes (0 = No, 1= Yes)", fontsize=12)
plt.ylabel("Count", fontsize=12)

plt.show()
```



Model Development and Prediction

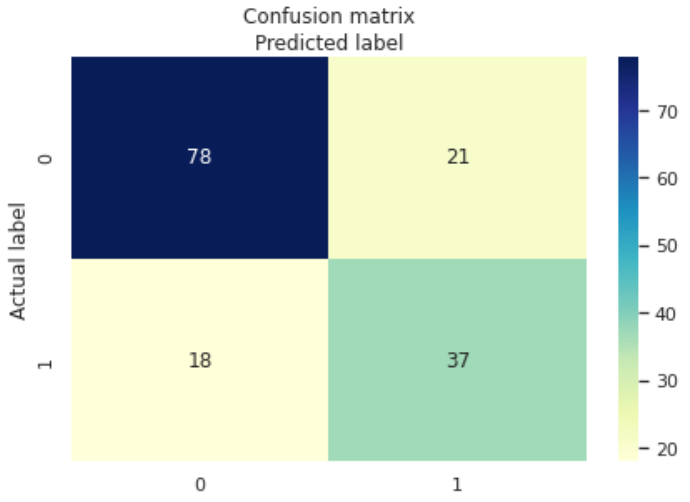
Here we create and apply our Logistic Regression Model.

Then we visualize the results of the model using Heatmap on the confusion matrix.

```
# Logistic Regression Model

logRegModel = LogisticRegression()
logRegModel.fit(X_train, y_train)
y_pred = logRegModel.predict(X_test)
```

```
# Visualizing Confusion Matrix using Heatmap

cnf_matrix = metrics.confusion_matrix(y_test, y_pred)

class_names=[0,1] # name  of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

```
Text(0.5, 257.44, 'Predicted label')
```



CFM evaluation metrics Accuracy, Precision, Recall

```
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
print("Precision:",metrics.precision_score(y_test, y_pred))
print("Recall:",metrics.recall_score(y_test, y_pred))
```

```
Accuracy: 0.7467532467532467
Precision: 0.6379310344827587
Recall: 0.6727272727272727
```

Observations:

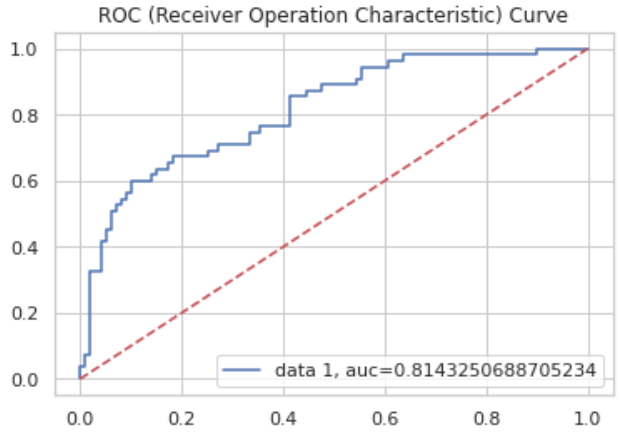
- 78 People have been correctly predicted to have diabetes
- 37 People have been correctly predicted to not have diabetes
- 18 People have been incorrectly predicted to have diabetes (False Positives)
- 21 People have been incorrectly predicted to not have diavetes (False Negatives)

The model **accuracy with Logistic Regression is 75% (rounded 0.7467532467532467)**, which is not to bad, but can be improved.

Here we plot the Receiver Operation Characteristic Curve (ROC).

The graph illustrates the true positive rate against the false positive rate. The further away the blue line (Classifier) is from the red line, the "better" it is.

```
y_pred_proba = logRegModel.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)
plt.plot(fpr,tpr,label="data 1, auc="+str(auc))
plt.plot([0,1],[0,1],"r--")
plt.legend(loc=4)
plt.title("ROC (Receiver Operation Characteristic) Curve")
plt.show()
```



References Used:

https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database

https://colab.research.google.com/github/diannekrouse/LRPython/blob/master/LogisticRegression.ipynb#scrollTo=6FOw7SPCGTlg

---