Jonothan Meyer
Machine Learning
03/14/21
Homework 2

1. Select the pima diabetes dataset with binary target values from 'https://machinelearningmastery.com/standard-machine-learning-datasets/'

Downloaded the data set to my computer to be imported into python.

2. Use pandas to read CSV file as dataframe. (1pt)

```
col_names=['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedigree', 'age', 'labelvalue']
pima = pd.read_csv("C:/Users/Jonothan/Desktop/MSU-Spring2021/Machine
                    Learning/Assignments/HW2/pima-indians-diabetes-database.csv", header=None,
                    names = col_names)
```

After downloading the data set we import the data using the 'read_csv()' method in the pandas class and stored it in the variable 'pima'. Then we add a top row to the data set to give context to each column.

3. Select 5 (if not possible then select 4) features from the chosen dataset. (1pt) List all features you selected in your report.

```
feature_cols = ['pregnant', 'bp', 'age', 'insulin', 'glucose']
X = pima[feature_cols]
y = np.array(pima.labelvalue)
```

We chose the five variables listed in 'feature_cols' as our predictor variables to build our logistic regression model. These were chosen because after repeatedly testing different combinations of variables this combo consistently output the highest accuracy, above 70%. For our response variable 'labelvalue' was chosen. So overall the purpose of the model is to find which combination of five predictor variables is best for accurately determining if someone has diabetes (1 for does, 0 for doesn't).

4. Use "train _test_split" from sklearn.model_selection to split test and training data by 40% testing + 60% training. (1pt)

```
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.4)
```

Using 'sklearn.model' we partitioned the pima data into training and testing variables. 40% of the data was used for testing, and the remaining 60% was used for training. This was done using the 'train_test_split' function above.

5. Fit your model with training data and test your model after fitting.

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

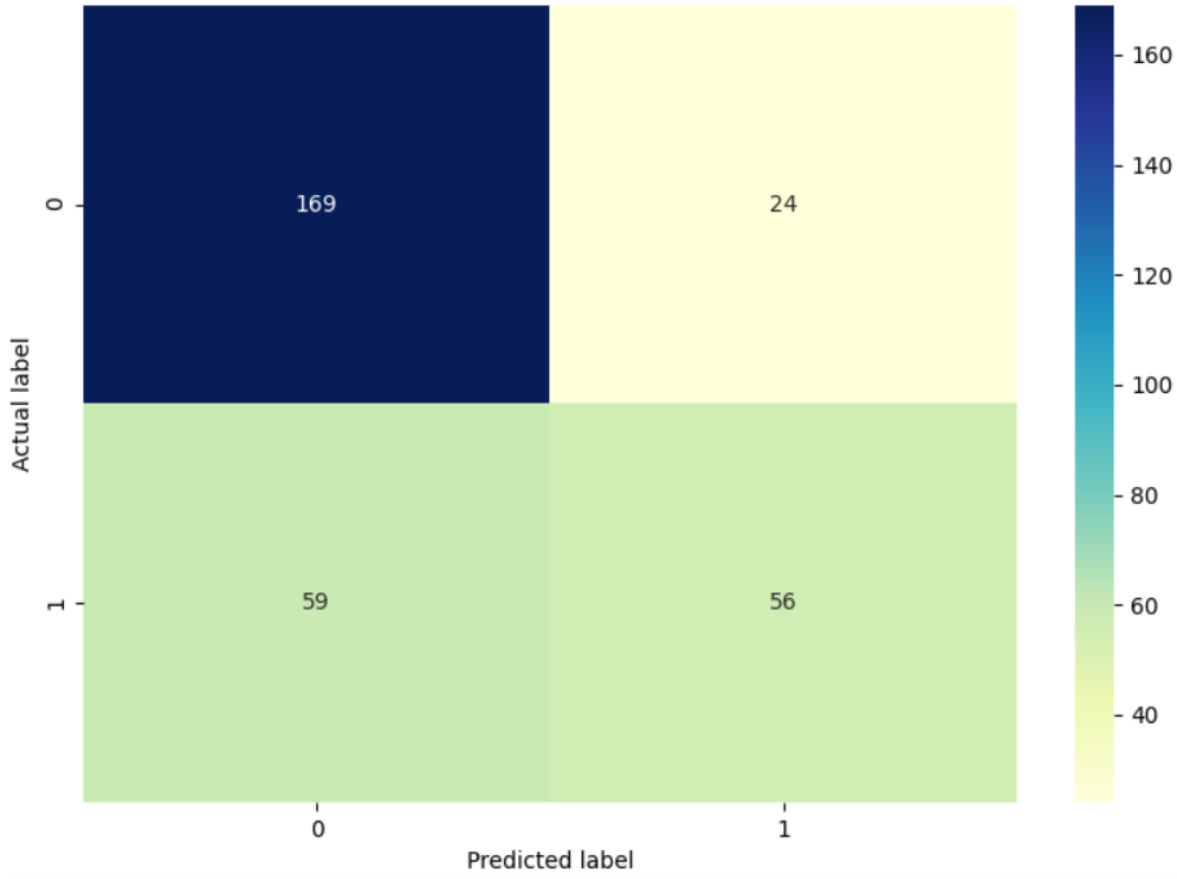
After importing the 'LogisticRegression' class from 'sklearn' we first instantiated our model called 'logreg'. Using the function 'fit' with the training data we fit our model and find our prediction 'y' points from the testing data.

6. Calculate and print out the confusion matrix (1pt), precision score, recall score, F score (3pts)

```
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
print(cnf_matrix) #Simplistic confusion matrix printout
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))
print(metrics.classification_report( y_test, y_pred)) #F-score found here!
```

Accuracy: .7305
Precision: .70
Recall: .4869
F score: 0 = .80, 1 = .57

Confusion Matrix



The code to print out the color shaded confusion matrix seen above can be found in the .py file.

7. Plot out the ROC curve and print out the ROC_AUC score (sklearn.metrics.roc_curve() and sklearn.metrics.roc_auc_score() can be used.) (3pts)

```
def plot_ROC_Curve(X_test, y_test):
    y_pred_proba = logreg.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) #calculates ROC_AUC score, find_accuracy()
                                                      #method also works
    plt.plot(fpr, tpr, label="data 1, auc=" + str(round(auc,4)))
    plt.legend(loc=4)
    plt.title('ROC Curve')
    plt.ylabel('True Positive Rate')
    plt.xlabel('False Positive Rate')
    plt.show()
```

ROC Score: .7637