logistic-regression

July 8, 2023

#INTERNSHIP PROJECT #TITLE : LOGISTIC REGRESSION

[1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import␣
 ↪accuracy_score,confusion_matrix,classification_report
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
df= pd.read_csv('diabetes2.csv')
df.head() # Display the first few rows of the dataset
```

[2]:

	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	

	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

[3]:
```
df.info() # Get information about the dataset, including missing values
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
```

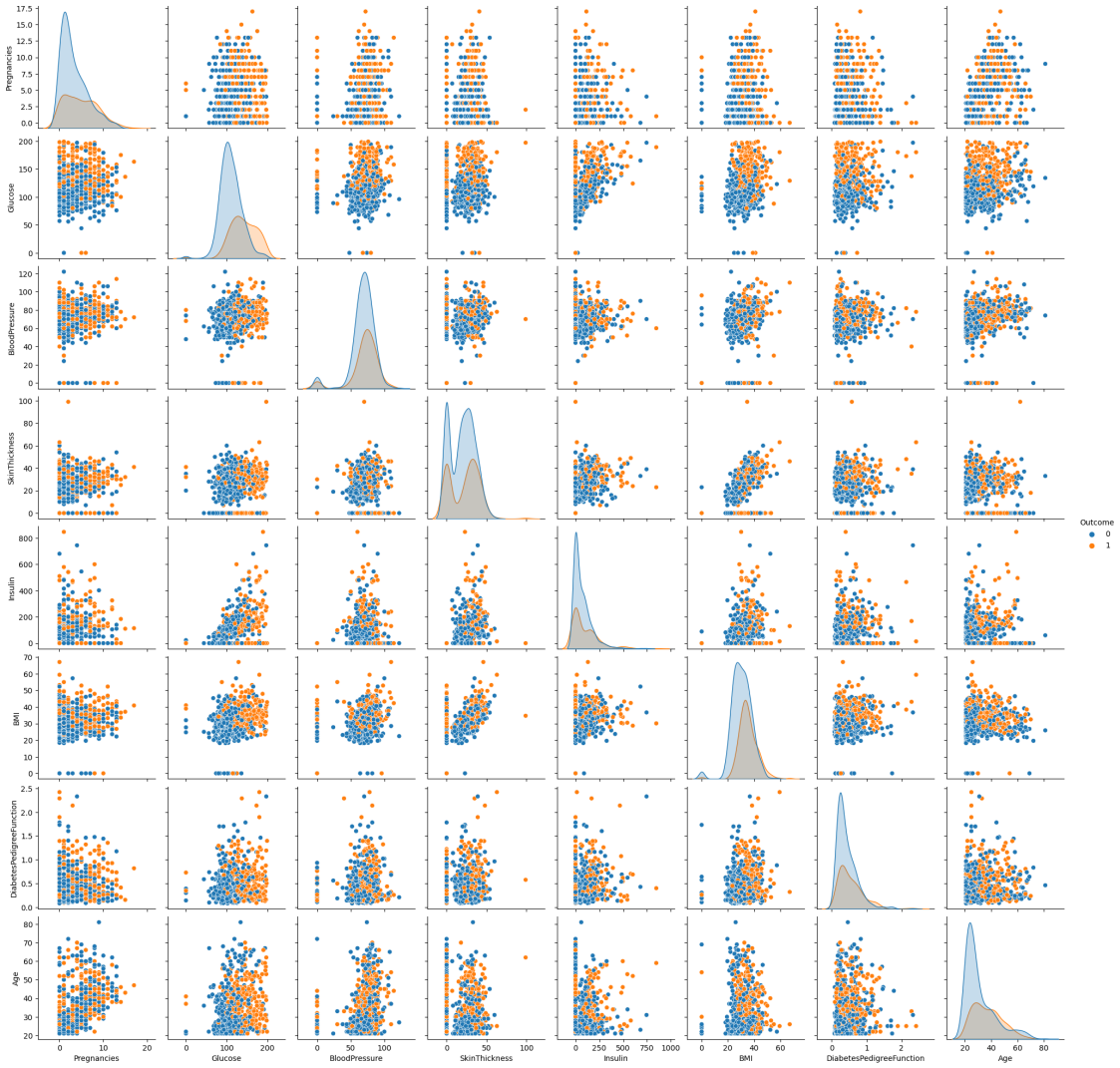
```
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

[12]:
```
df.isnull().sum()
```

[12]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

[4]:
```
sns.pairplot(df, hue="Outcome")
plt.show
```

[4]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



[5]:
```
x = df.drop("Outcome", axis=1)
y = df["Outcome"]
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25,␣
 ↪random_state=42)
```

[6]:
```
model = LogisticRegression()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
```

```
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[7]:
```
x_new=pd.DataFrame({'Pregnancies':[2],'Glucose':[130],'BloodPressure':
  ↪[60],'SkinThickness':[35],'Insulin':[95],'BMI':
  ↪[35],'DiabetesPedigreeFunction':[2.00],'Age':[19]})
```

[8]:
```
print(model.predict(x_new))
```

```
[1]
```

[9]:
```
y_pred_train = model.predict(x)
accuracy = accuracy_score(y, y_pred_train)
print("Accuracy:",accuracy)
```

```
Accuracy: 0.765625
```

[10]:
```
confusion_mat = confusion_matrix(y, y_pred_train)
print("confusion matrix:")
print(confusion_mat)
```

```
confusion matrix:
[[431  69]
 [111 157]]
```

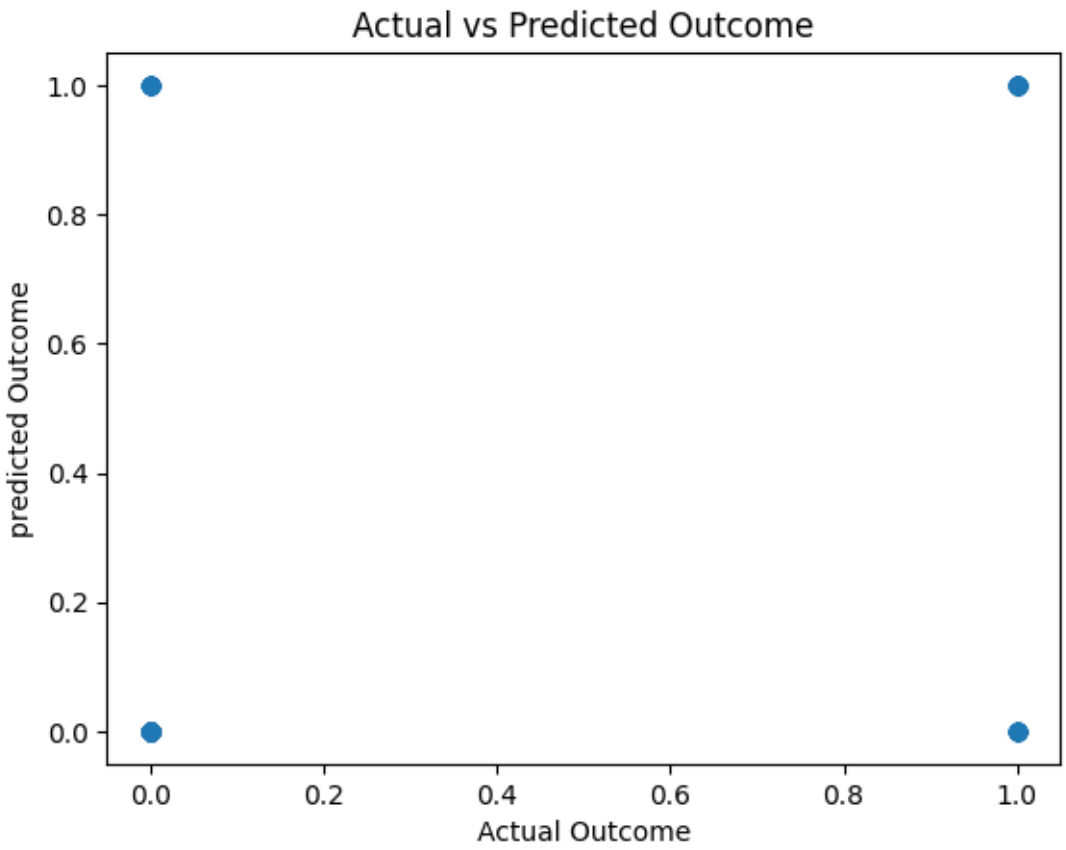
[11]:
```
report = classification_report(y, y_pred_train)
print("Classification Report:")
print(report)
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.86      0.83       500
           1       0.69      0.59      0.64       268

    accuracy                           0.77       768
   macro avg       0.74      0.72      0.73       768
weighted avg       0.76      0.77      0.76       768
```

[16]:
```
plt.scatter(y_test,y_pred)
plt.xlabel("Actual Outcome")
plt.ylabel("predicted Outcome")
plt.title("Actual vs Predicted Outcome")
plt.show()
```


Actual vs Predicted Outcome
predicted Outcome
Actual Outcome
0.0
0.2
0.4
0.6
0.8
1.0