6000 4210210
Amartya Mishra
6 Or COMPS-C31

ML- Experiment 2

Aim: To Implement logistic regression

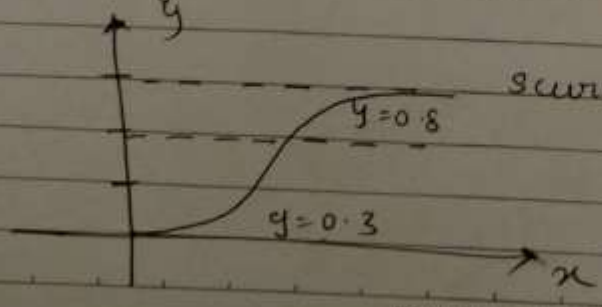
Theory: It is one of most popular ML algo which comes under supervised learning

It is used for predicate, categorical dependent variable using given set of independent variable
It predicts output of categorical dependent variable
∴ outcome must be categorical or discrete value
It can either be Yes or No 0, 1, True or false
But instead of exact values as 0, 1 it gives probabilistic values which lie between 0 & 1

Logistic Regression is used for solving categorical & thus classification problem.

In logistic regression instead of fitting a straight line as in linear regression, a function with curved shape is fit onto the data.

logistic regression can be visualised as



Conclusion

We implemented logistic regression for Data which has category as output rather than continous values

Code And Output:

[12]:
```
import pandas as pd
col_names = ['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedigree',␣
 ↪'age', 'label']

df = pd.read_csv("diabetes.csv",header=None, names=col_names)
```

[13]:
```
df.head()
```

[13]:

	pregnant	glucose	bp	skin	insulin	bmi	pedigree	age	label
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

[14]:
```
#split dataset in features and target variable
feature_cols = ['pregnant', 'insulin', 'bmi', 'age','glucose','bp','pedigree']
X = df[feature_cols] # Features
y = df.label # Target variable
```

[15]:
```
X
```

[15]:

	pregnant	insulin	bmi	age	glucose	bp	pedigree
0	6	0	33.6	50	148	72	0.627
1	1	0	26.6	31	85	66	0.351
2	8	0	23.3	32	183	64	0.672
3	1	94	28.1	21	89	66	0.167
4	0	168	43.1	33	137	40	2.288
..	…	…	…	…	…	..	…
763	10	180	32.9	63	101	76	0.171
764	2	0	36.8	27	122	70	0.340
765	5	112	26.2	30	121	72	0.245

[768 rows x 7 columns]

[16]:
```
766         1        0 30.1   47      126 60     0.349
767         1        0 30.4   23       93 70     0.315
```

```
y
```

[16]:
```
0      1
1      0
2      1
3      0
4      1
..
763    0
764    0
765    0
766    1
767    0
Name: label, Length: 768, dtype: int64
```

[17]:
```
# split X and y into training and testing sets
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,␣
 ↪random_state=16)
```

[20]:
```
from sklearn.linear_model import LogisticRegression


logreg = LogisticRegression(random_state=16 ,max_iter=2000)

logreg.fit(X_train, y_train)

y_pred = logreg.predict(X_test)
```

[21]:
```
# import the metrics class
from sklearn import metrics

cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix
```

[21]:
```
array([[115,  10],
       [ 25,  42]], dtype=int64)
```

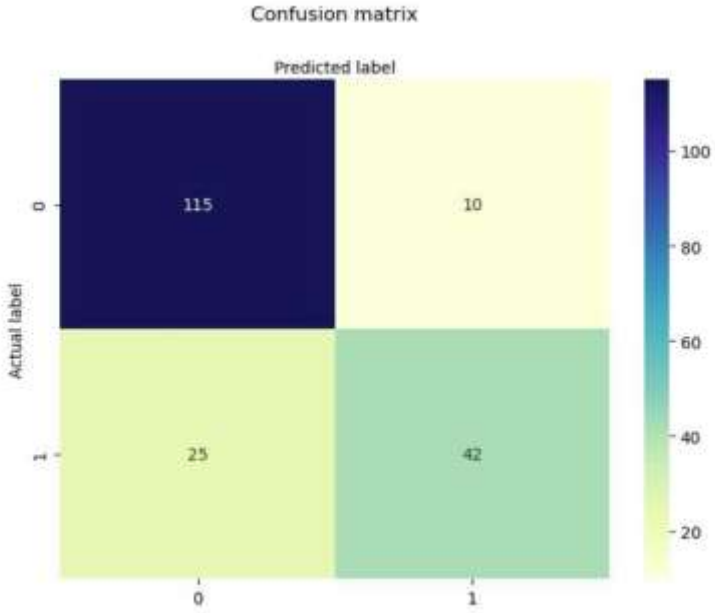
[22]:
```
# import required modules
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

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',  =1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')

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

```
---------------------------------------------------------------------------
-------
NameError                                 Traceback (most recent call last)
Cell In[22], line 19
     16 plt.ylabel('Actual label')
     17 plt.xlabel('Predicted label')
---> 19 Text(0.5,257.44,'Predicted label')

NameError: name 'Text' is not defined
```



[53]:
```
from sklearn.metrics import
classification_report,accuracy_score target_names = ['without
diabetes', 'with diabetes']
                    print(classification_report(y_test, y_pred,
                                               target_names=target_names))
```

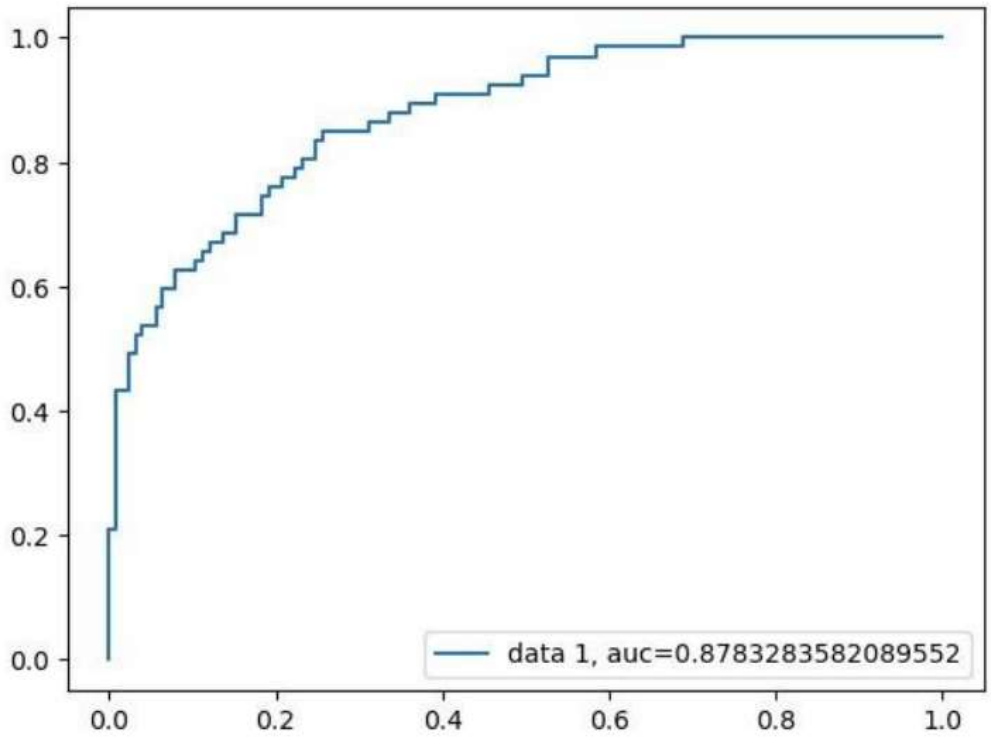
```
                  precision    recall  f1-score   support

    without diabetes       0.82      0.92      0.87       125
       with diabetes       0.81      0.63      0.71        67

            accuracy                           0.82       192
           macro avg       0.81      0.77      0.79       192
        weighted avg       0.82      0.82      0.81       192
```

[54]:
```
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)
plt.plot(fpr,tpr,label="data 1, auc="+str(auc))
plt.legend(loc=4)
plt.show()
```



[55]:
```
acc_score=accuracy_score(y_test,y_pred)
round(acc_score,2)
```

[55]: 0.82