ML Experiment 2

ML Experiment 2 DATE:

Kareena Shah
60004210243
C'32

Aim : To implement & understand the basics of logistic regression

Theory :

Logistic Regression is a supervised machine learning algorithm used for classification tasks where the goal is to predict the probability that an instance belongs to a given class or not. It is a statistical algorithm which analyze the relationship between two data factors

Types of logistic Regression :

(1) Binomial
(2) Multinomial
(3) Ordinal

$z = w.X + b$

X : independent inputs
b : bias term (intercept)
w : weights

Sigmoid function

The output (z) is mapped to get the probability

$$\sigma(z) = \frac{1}{1 - e^{-z}}$$

DATE:

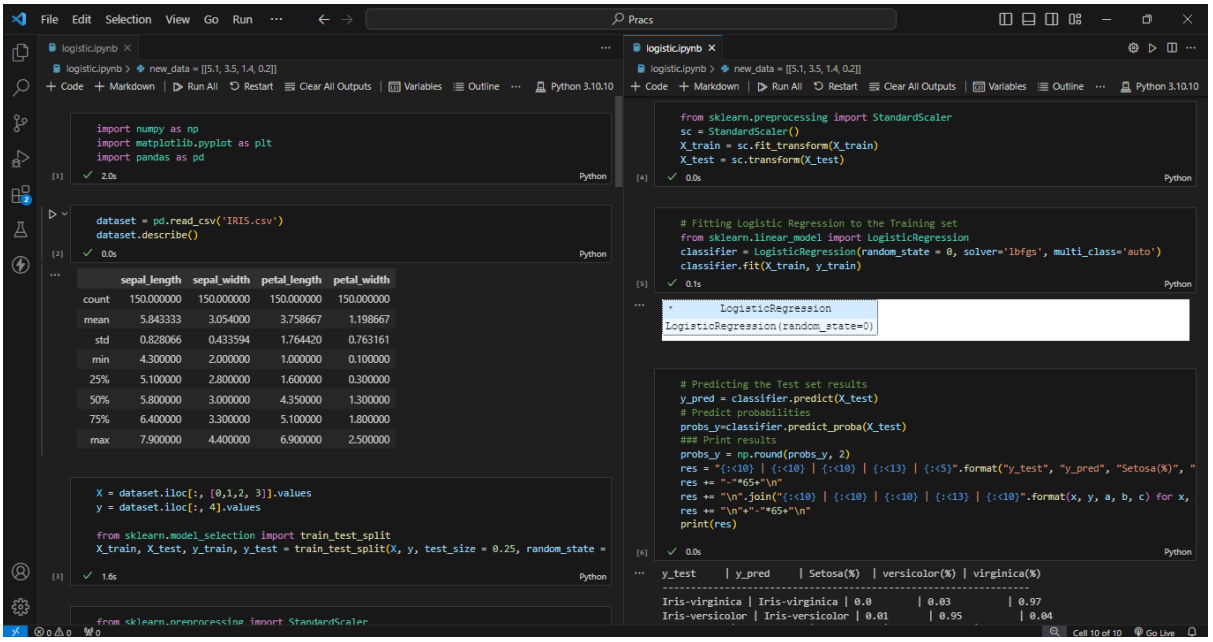
Logistic Regression Equation

$$P(X, b, w) = \frac{1}{1 + e^{-w \cdot X + b}}$$

Evaluating model methods :

(1) Accuracy
(2) Precision
(3) Recall
(4) F1 score

Conclusion : Thus, we implemented logistic regression.



```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

```
dataset = pd.read_csv('IRIS.csv')
dataset.describe()
```

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

```
X = dataset.iloc[:, [0,1,2, 3]].values
y = dataset.iloc[:, 4].values

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 =
```

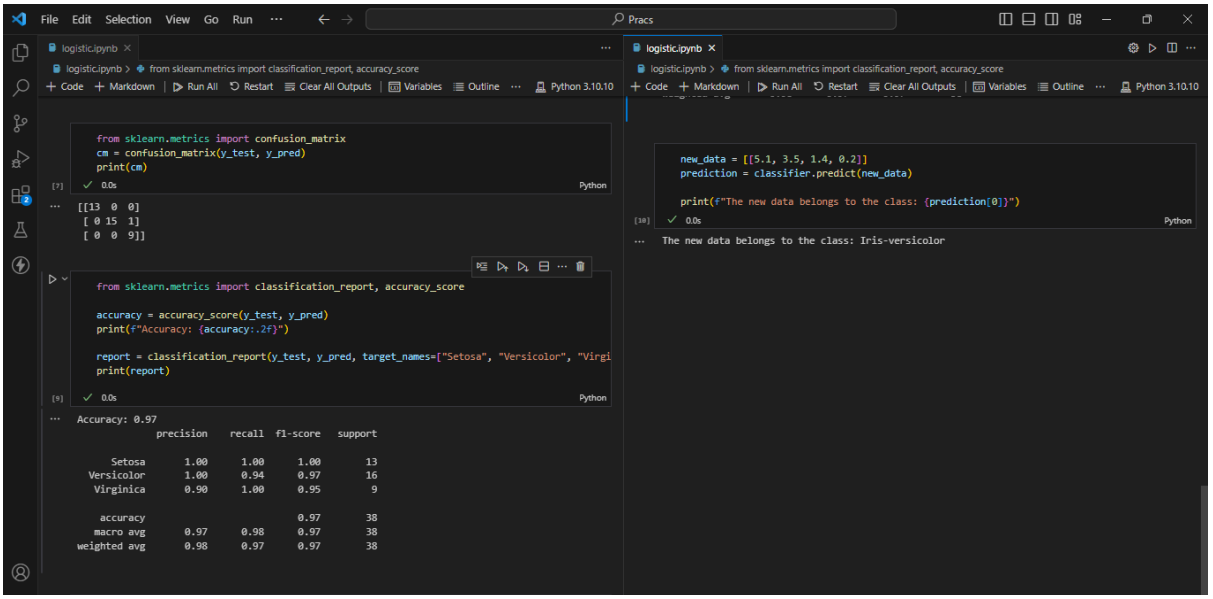
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
# Fitting Logistic Regression to the Training set
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0, solver='lbfgs', multi_class='auto')
classifier.fit(X_train, y_train)
```

```
LogisticRegression(random_state=0)
```

```
# Predicting the Test set results
y_pred = classifier.predict(X_test)
# Predict probabilities
probs_y=classifier.predict_proba(X_test)
### Print results
probs_y = np.round(probs_y, 2)
res = "{:<10} | {:<10} | {:<10} | {:<13} | {:<5}".format("y_test", "y_pred", "Setosa(%)", "
res += "-"*65+"\n"
res += "\n".join("{:<10} | {:<10} | {:<10} | {:<13} | {:<10}".format(x, y, a, b, c) for x,
res += "\n"+"-"*65+"\n"
print(res)
```

```
y_test     | y_pred     | Setosa(%)  | versicolor(%) | virginica(%)
-----------------------------------------------------------------
Iris-virginica | Iris-virginica | 0.0        | 0.03          | 0.97
Iris-versicolor | Iris-versicolor | 0.01       | 0.95          | 0.04
```



```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[13  0  0]
 [ 0 15  1]
 [ 0  0  9]]
```

```
from sklearn.metrics import classification_report, accuracy_score

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

report = classification_report(y_test, y_pred, target_names=["Setosa", "Versicolor", "Virgi
print(report)
```

```
Accuracy: 0.97
              precision    recall  f1-score   support

      Setosa       1.00      1.00      1.00        13
  Versicolor       1.00      0.94      0.97        16
   Virginica       0.90      1.00      0.95         9

    accuracy                           0.97        38
   macro avg       0.97      0.98      0.97        38
weighted avg       0.98      0.97      0.97        38
```

```
new_data = [[5.1, 3.5, 1.4, 0.2]]
prediction = classifier.predict(new_data)

print(f"The new data belongs to the class: {prediction[0]}")
```

```
The new data belongs to the class: Iris-versicolor
```