EXPERIMENT- 6

AIM
Implement a classification problem based on logistic regression.

SOFTWARE USED
Google Colab Platform - Python Programming Language

PROGRAM CODE

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt # data visualization
import seaborn as sns # statistical data visualization
%matplotlib inline


import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# Any results you write to the current directory are saved as output.

data = '/kaggle/input/weather-dataset-rattle-package/weatherAUS.csv'

df = pd.read_csv(data)

# view dimensions of dataset

df.shape

# preview the dataset

df.head()

col_names = df.columns

col_names

# find categorical variables

categorical = [var for var in df.columns if df[var].dtype=='O']

print('There are {} categorical variables\n'.format(len(categorical)))

print('The categorical variables are :', categorical)

# view the categorical variables
```

```
df[categorical].head()

X = df.drop(['RainTomorrow'], axis=1)

y = df['RainTomorrow']

# 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 = 0.2, random_state = 0)

# check the shape of X_train and X_test

X_train.shape, X_test.shape

# train a logistic regression model on the training set
from sklearn.linear_model import LogisticRegression


# instantiate the model
logreg = LogisticRegression(solver='liblinear', random_state=0)


# fit the model
logreg.fit(X_train, y_train)

y_pred_test = logreg.predict(X_test)

y_pred_test

# probability of getting output as 0 - no rain

logreg.predict_proba(X_test)[:,0]

# probability of getting output as 1 - rain

logreg.predict_proba(X_test)[:,1]

from sklearn.metrics import accuracy_score

print('Model accuracy score: {0:0.4f}'. format(accuracy_score(y_test, y_pred_test)))

y_pred_train = logreg.predict(X_train)

y_pred_train

print('Training-set accuracy score: {0:0.4f}'. format(accuracy_score(y_train, y_pred_train)))
```

OUTPUT

logistic-regression-classifier-tutorial.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

Connect Editing

Import libraries

Table of Contents

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load in

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt # data visualization
import seaborn as sns # statistical data visualization
%matplotlib inline

# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# Any results you write to the current directory are saved as output.
```

```
/kaggle/input/weather-dataset-rattle-package/weatherAUS.csv
```

Import dataset

Table of Contents

```
[ ] data = '/kaggle/input/weather-dataset-rattle-package/weatherAUS.csv'

    df = pd.read_csv(data)
```

logistic-regression-classifier-tutorial.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

Connect Editing

```
[ ] data = '/kaggle/input/weather-dataset-rattle-package/weatherAUS.csv'

    df = pd.read_csv(data)
```

Exploratory data analysis

Table of Contents

Now, I will explore the data to gain insights about the data.

```
[ ] # view dimensions of dataset

    df.shape
```

```
(142193, 24)
```

We can see that there are 142193 instances and 24 variables in the data set.

```
# preview the dataset

df.head()
```

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Ter
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	NaN	W	44.0	W	...	22.0	1007.7	1007.1	8.0	NaN	16.9	
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	NaN	WNW	44.0	NNW	...	25.0	1010.6	1007.8	NaN	NaN	17.2	
2	2008-12-03	Albury	12.9	25.7	0.0	NaN	NaN	WSW	46.0	W	...	30.0	1007.6	1008.7	NaN	2.0	21.0	
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	NaN	NE	24.0	SE	...	16.0	1017.6	1012.8	NaN	NaN	18.1	

logistic-regression-classifier-tutorial.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

Connect Editing

2	2008-12-03	Albury	12.9	25.7	0.0	NaN	NaN	WSW	46.0	W	...	30.0	1007.6	1008.7	NaN	2.0	21.0
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	NaN	NE	24.0	SE	...	16.0	1017.6	1012.8	NaN	NaN	18.1
4	2008-12-05	Albury	17.5	32.3	1.0	NaN	NaN	W	41.0	ENE	...	33.0	1010.8	1006.0	7.0	8.0	17.8

5 rows × 24 columns

```
col_names = df.columns

col_names
```

```
Index(['Date', 'Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation',
       'Sunshine', 'WindGustDir', 'WindGustSpeed', 'WindDir9am', 'WindDir3pm',
       'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm',
       'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am',
       'Temp3pm', 'RainToday', 'RISK_MM', 'RainTomorrow'],
      dtype='object')
```

Types of variables

In this section, I segregate the dataset into categorical and numerical variables. There are a mixture of categorical and numerical variables in the dataset. Categorical variables have data type object. Numerical variables have data type float64.

First of all, I will find categorical variables.

```
# find categorical variables

categorical = [var for var in df.columns if df[var].dtype=='O']

print('There are {} categorical variables\n'.format(len(categorical)))

print('The categorical variables are :', categorical)
```

logistic-regression-classifier-tutorial.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

Connect Editing

```
print('The categorical variables are :', categorical)
```

```
There are 7 categorical variables

The categorical variables are : ['Date', 'Location', 'WindGustDir', 'WindDir9am', 'WindDir3pm', 'RainToday', 'RainTomorrow']
```

```
# view the categorical variables

df[categorical].head()
```

	Date	Location	WindGustDir	WindDir9am	WindDir3pm	RainToday	RainTomorrow
0	2008-12-01	Albury	W	W	WNW	No	No
1	2008-12-02	Albury	WNW	NNW	WSW	No	No
2	2008-12-03	Albury	WSW	W	WSW	No	No
3	2008-12-04	Albury	NE	SE	E	No	No
4	2008-12-05	Albury	W	ENE	NW	No	No

Summary of categorical variables

- There is a date variable. It is denoted by `Date` column.
- There are 6 categorical variables. These are given by `Location`, `WindGustDir`, `WindDir9am`, `WindDir3pm`, `RainToday` and `RainTomorrow`.
- There are two binary categorical variables - `RainToday` and `RainTomorrow`.
- `RainTomorrow` is the target variable.

Declare feature vector and target variable

Table of Contents

logistic-regression-classifier-tutorial.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

```
X = df.drop(['RainTomorrow'], axis=1)

y = df['RainTomorrow']
```

Split data into separate training and test set

Table of Contents

```
# 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 = 0.2, random_state = 0)
```

```
# check the shape of X_train and X_test

X_train.shape, X_test.shape
```

```
((113754, 24), (28439, 24))
```

Model training

Table of Contents

```
# train a logistic regression model on the training set
from sklearn.linear_model import LogisticRegression


# instantiate the model
logreg = LogisticRegression(solver='liblinear', random_state=0)
```

logistic-regression-classifier-tutorial.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

```
# fit the model
logreg.fit(X_train, y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=0, solver='liblinear', tol=0.0001, verbose=0,
                   warm_start=False)
```

Predict results

Table of Contents

```
y_pred_test = logreg.predict(X_test)

y_pred_test
```

```
array(['No', 'No', 'No', ..., 'No', 'No', 'Yes'], dtype=object)
```

predict_proba method

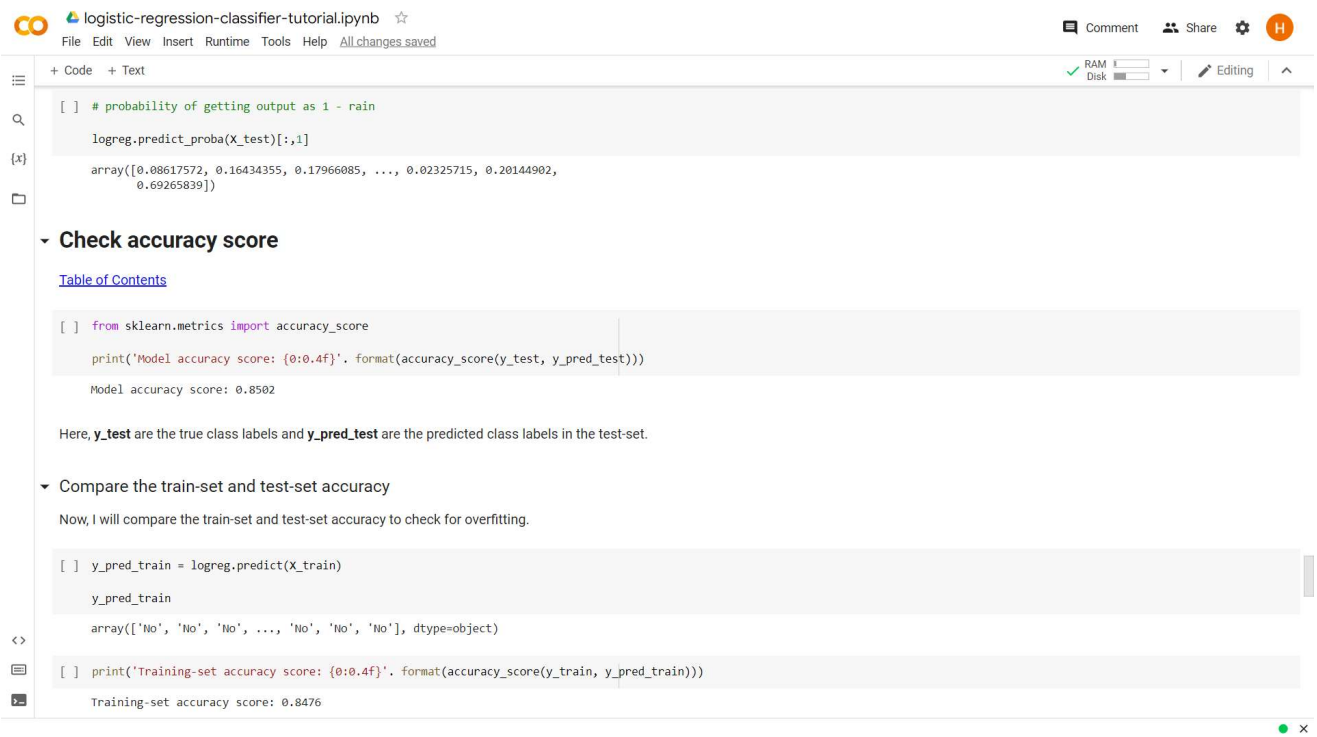
predict_proba method gives the probabilities for the target variable(0 and 1) in this case, in array form.

`0 is for probability of no rain` and `1 is for probability of rain.`

```
# probability of getting output as 0 - no rain

logreg.predict_proba(X_test)[:,0]
```

```
array([0.91382428, 0.83565645, 0.82033915, ..., 0.97674285, 0.79855098,
       0.30734161])
```



logistic-regression-classifier-tutorial.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

```
[ ] # probability of getting output as 1 - rain

    logreg.predict_proba(X_test)[:,1]
```

```
array([0.08617572, 0.16434355, 0.17966085, ..., 0.02325715, 0.20144902,
       0.69265839])
```

Check accuracy score

Table of Contents

```
[ ] from sklearn.metrics import accuracy_score

    print('Model accuracy score: {0:0.4f}'. format(accuracy_score(y_test, y_pred_test)))
```

```
Model accuracy score: 0.8502
```

Here, **y_test** are the true class labels and **y_pred_test** are the predicted class labels in the test-set.

Compare the train-set and test-set accuracy

Now, I will compare the train-set and test-set accuracy to check for overfitting.

```
[ ] y_pred_train = logreg.predict(X_train)

    y_pred_train
```

```
array(['No', 'No', 'No', ..., 'No', 'No', 'No'], dtype=object)
```

```
[ ] print('Training-set accuracy score: {0:0.4f}'. format(accuracy_score(y_train, y_pred_train)))
```

```
Training-set accuracy score: 0.8476
```

DISCUSSION and CONCLUSION

The logistic regression model has been applied and executed successfully on the classification problem over the weather dataset of Australia.

CRITERIA	TOTAL MARKS	MARKS OBTAINED	COMMENTS
Concept (A)	2		
Implementation (B)	2		
Performance (C)	2		
Total	6		