pratical-5

April 17, 2024

---

1 Data Analytics II

---

1. Implement logistic regression using Python/R to perform classification on Social_Network_Ads.csv dataset.
2. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.

[ ]:
```
# # Import required libraries
# import numpy as np
# import pandas as pd
# from sklearn.model_selection import train_test_split
# from sklearn.preprocessing import StandardScaler
# from sklearn.linear_model import LogisticRegression
# from sklearn.metrics import confusion_matrix,classification_report
# import matplotlib.pyplot as plt
# %matplotlib inline

# # Load the dataset
# df = pd.read_csv("Social_Network_Ads.csv")
# df.head(10)

# df.info()

# df.describe()

# X = df.iloc[:,[2,3]].values
# y = df.iloc[:,4].values

# # Split the dataset into train and test
# X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25,␣
 ↪random_state = 0)

# # Preprocessing
# # Standard Scalar
# sc = StandardScaler()
```

```
# X_train = sc.fit_transform(X_train)
# X_test = sc.transform(X_test)

# classifier = LogisticRegression(random_state=0)
# classifier.fit(X_train, y_train)

# y_pred = classifier.predict(X_test)
# y_pred

# # Confusion Matrix
# cm = confusion_matrix(y_test, y_pred)
# cm

# c1_report = classification_report(y_test, y_pred)
# c1_report

# tp, fn, fp, tn = confusion_matrix(y_test, y_pred, labels=[0,1]).reshape(-1)

# print(f"Outcome Values:\nTrue Positive = {tp}\nFalse Negative = {fn}\nFalse Positive = {fp}\nTrue Negative = {tn}")

# accuracy_cm = (tp+tn)/(tp+fp+tn+fn)
# precision_cm = tp/(tp+fp)
# recall_cm = tp/(tp+fn)
# f1_score = 2/((1/recall_cm)+(1/precision_cm))

# print(f"Accuracy   : {accuracy_cm}")
# print(f"Precision  : {precision_cm}")
# print(f"Recall     : {recall_cm}")
# print(f"F1-Score   : {f1_score}")
```

[1]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix,classification_report
import matplotlib.pyplot as plt
%matplotlib inline
```

[2]:
```
df = pd.read_csv("Social_Network_Ads.csv")
df.head(10)
```

[2]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0

```
3  15603246  Female   27            57000           0
4  15804002    Male   19            76000           0
5  15728773    Male   27            58000           0
6  15598044  Female   27            84000           0
7  15694829  Female   32           150000           1
8  15600575    Male   25            33000           0
9  15727311  Female   35            65000           0
```

[3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   User ID          400 non-null    int64
 1   Gender           400 non-null    object
 2   Age              400 non-null    int64
 3   EstimatedSalary  400 non-null    int64
 4   Purchased        400 non-null    int64
dtypes: int64(4), object(1)
memory usage: 15.8+ KB
```

[4]:
```
df.describe()
```

[4]:

	User ID	Age	EstimatedSalary	Purchased
count	4.000000e+02	400.000000	400.000000	400.000000
mean	1.569154e+07	37.655000	69742.500000	0.357500
std	7.165832e+04	10.482877	34096.960282	0.479864
min	1.556669e+07	18.000000	15000.000000	0.000000
25%	1.562676e+07	29.750000	43000.000000	0.000000
50%	1.569434e+07	37.000000	70000.000000	0.000000
75%	1.575036e+07	46.000000	88000.000000	1.000000
max	1.581524e+07	60.000000	150000.000000	1.000000

[5]:
```
X = df.iloc[:,[2,3]].values
y = df.iloc[:,4].values
```

[6]:
```
# Split the dataset into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25,␣
 ↪random_state = 0)
```

[7]:
```
# Preprocessing

# Standard Scalar
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
```

```
X_test = sc.transform(X_test)
```

[8]:
```
classifier = LogisticRegression(random_state=0)
classifier.fit(X_train, y_train)
```

[8]:
```
LogisticRegression(random_state=0)
```

[9]:
```
y_pred = classifier.predict(X_test)
y_pred
```

[11]:
```
# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
cm
```

[13]:
```
c1_report = classification_report(y_test, y_pred)
c1_report
```

[15]:
```
tp, fn, fp, tn = confusion_matrix(y_test, y_pred, labels=[0,1]).reshape(-1)
```

[16]:
```
print(f"Outcome Values:\nTrue Positive = {tp}\nFalse Negative = {fn}\nFalse␣
↪Positive = {fp}\nTrue Negative = {tn}")
```

```
Outcome Values:
True Positive = 65
False Negative = 3
False Positive = 8
True Negative = 24
```

[17]:
```
accuracy_cm = (tp+tn)/(tp+fp+tn+fn)
precision_cm = tp/(tp+fp)
recall_cm = tp/(tp+fn)
f1_score = 2/((1/recall_cm)+(1/precision_cm))
```

[18]:
```
print(f"Accuracy   : {accuracy_cm}")
print(f"Precision  : {precision_cm}")
print(f"Recall     : {recall_cm}")
print(f"F1-Score   : {f1_score}")
```

```
Accuracy   : 0.89
Precision  : 0.8904109589041096
Recall     : 0.9558823529411765
F1-Score   : 0.9219858156028368
```

[ ]:
```

```