Suraj Sawant TEB-38

Practical A5:-Logistic Regression (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.

```
# https://www.kaggle.com/datasets/akram24/social-network-ads/data
```

```
# Dataset on Social media ads describing users, whether users have purchased a product by
```

```
import pandas as pd
```

```
df = pd.read_csv("/content/Social_Network_Ads - Social_Network_Ads.csv")
df.head(10)
```

	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

```
df.shape
```

```
(400, 5)
```

```
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
```

```
df.describe()
```

	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

```
df.drop(['User ID'],axis=1,inplace=True)# Drop the unwanted column User ID
```

```
df.Purchased.value_counts()
```

Purchased	count
0	257
1	143

dtype: int64

```
from sklearn.preprocessing import LabelEncoder  # Import LabelEncoder
label_encoder = LabelEncoder()  # Create an instance of LabelEncoder
df['Gender'] = label_encoder.fit_transform(df['Gender']) # Male=1, Female=0
```

```
df.head()
```

	Gender	Age	EstimatedSalary	Purchased
0	1	19	19000	0
1	1	35	20000	0
2	0	26	43000	0
3	0	27	57000	0
4	1	19	76000	0

```
x=df[['Gender','Age','EstimatedSalary']]
```

```
x.head()
# x contains the independent features (Gender, Age, EstimatedSalary)
```

	Gender	Age	EstimatedSalary
0	1	19	19000
1	1	35	20000
2	0	26	43000
3	0	27	57000
4	1	19	76000

```
y=df['Purchased']
y.head() # y contains the dependent variable
"Purchased")
```

	Purchased
0	0
1	0
2	0
3	0
4	0

dtype: int64

```
from sklearn.model_selection import train_test_split xtrain,xtest, ytrain,
ytest=train_test_split(x,y,test_size=0.2,random_state=42)
#20% of the data is reserved for testing, while 80% is used for training.
# y contains the dependent variable  "Purchased")


xtrain.shape
```

```
(320, 3)
```

```
xtest.shape
```

```
(80, 3)
```

```
ytrain.shape
```

```
(320,)
```

```
ytest.shape
```

```
(80,)
```

```
from sklearn.linear_model import LogisticRegression #import
model=LogisticRegression() #Create model.fit(xtrain,ytrain)
#Train
```

```
LogisticRegression()
```

```
ypred=model.predict(xtest) # model is generated for the test dataset
```

```
ypred #Predicted Values
```

```
array([0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0,
       0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0])
```

```
ytest #Actual Values
```

	Purchased
209	0
280	1
33	0
210	1
93	0
...	...
246	0
227	1
369	1
176	0
289	1

80 rows × 1 columns

dtype: int64

```
# Measure the Performance of the Model
from sklearn.metrics import accuracy_score
accuracy=accuracy_score(ytest, ypred)
accuracy
```

```
0.8875
```

```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(ytest,ypred) cm
```

```
array([[50,  2],
       [ 7, 21]])
```

```
from sklearn.metrics import classification_report
report=classification_report(ytest, ypred) print(report)
```

```
              precision    recall  f1-score   support

           0       0.88      0.96      0.92        52
           1       0.91      0.75      0.82        28

    accuracy                           0.89
80     macro avg       0.90      0.86      0.87
80  weighted avg       0.89      0.89      0.88
80  # With this Classification Report, Practical No.
    A-5 is Over
```

```
# The Program is extended further just to understand the implementaion of Naive Bayes Cla
# with respect to the same example, not to be included for practical exam

#Using Naive Bayes Classifier


from sklearn.naive_bayes import GaussianNB #algorithm based on Bayes' Theorem.


nb_model = GaussianNB()# GaussianNB() initializes the Naïve Bayes classifier.
nb_model.fit(xtrain, ytrain) #it trains model using x train and y train
```

```
GaussianNB
GaussianNB()
```

```
nb_ypred = nb_model.predict(xtest)


nb_report = classification_report(ytest, nb_ypred)
print(nb_report)
```

```
              precision    recall  f1-
score   support

           0       0.94      0.96      0.95        52
      1       0.93      0.89      0.91        28

    accuracy                           0.94        80
   macro avg       0.93      0.93      0.93        80
```