**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 datast.**

**Import all the required Python Libraries.**

**import** numpy **as** np

**import** pandas **as** pd

**1) Implement logistic regression using Python/R to perform classification on**

**Social\_Network\_Ads.csv dataset.**

df **=** pd**.**read\_csv('Social\_Network\_Ads.csv')

df**.**head()

**User ID Gender Age EstimatedSalary Purchased**

**0** 15624510 Male 19.0 19000.0 0

**1** 15810944 Male 35.0 20000.0 0

**2** 15668575 Female 26.0 43000.0 0

**3** 15603246 Female 27.0 57000.0 0

**4** 15804002 Male 19.0 76000.0 0

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 float64

3 EstimatedSalary 400 non-null float64

4 Purchased 400 non-null int64

dtypes: float64(2), int64(2), 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

X **=** df[['Age', 'EstimatedSalary']]

Y **=** df['Purchased']

**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 **=** 0)

print(f'Train Dataset Size - X: {X\_train**.**shape}, Y: {Y\_train**.**shape}')

print(f'Test Dataset Size - X: {X\_test**.**shape}, Y: {Y\_test**.**shape}')

Train Dataset Size - X: (300, 2), Y: (300,)

Test Dataset Size - X: (100, 2), Y: (100,)

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**from** sklearn.linear\_model **import** LogisticRegression

lm **=** LogisticRegression(random\_state **=** 0, solver**=**'lbfgs' )

lm**.**fit(X\_train, Y\_train)

predictions **=** lm**.**predict(X\_test)

2) Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate,

Precision, Recall on the given datast.

Confusion matrix

**from** sklearn.metrics **import** classification\_report

cm **=** classification\_report(Y\_test, predictions)

print('Classification report : \n', cm)

Classification report :

precision recall f1-score support

0 0.89 0.96 0.92 68

1 0.89 0.75 0.81 32

accuracy 0.89 100

macro avg 0.89 0.85 0.87 100

weighted avg 0.89 0.89 0.89 100

**from** sklearn.metrics **import** classification\_report, confusion\_matrix

conf\_matrix **=** confusion\_matrix(Y\_test, predictions)

plt**.**figure(figsize**=**(8, 6))

sns**.**heatmap(conf\_matrix, annot**=True**, fmt**=**'d', cmap**=**'Blues', cbar**=False**)

plt**.**xlabel('Predicted labels')

plt**.**ylabel('True labels')

plt**.**title('Confusion Matrix')

plt**.**show()

y\_pred **=** lm**.**predict(X\_test)

cm **=** confusion\_matrix(Y\_test, y\_pred)

TN **=** cm[0, 0]

FP **=** cm[0, 1]

FN **=** cm[1, 0]

TP **=** cm[1, 1]

accuracy **=** (TP **+** TN) **/** float(TP **+** TN **+** FP **+** FN)

error\_rate **=** (FP **+** FN) **/** float(TP **+** TN **+** FP **+** FN)

precision **=** TP **/** float(TP **+** FP)

recall **=** TP **/** float(TP **+** FN)

print("\nPerformance Metrics:")

print("True Positives (TP):", TP)

print("False Positives (FP):", FP)

print("True Negatives (TN):", TN)

print("False Negatives (FN):", FN)

print("Accuracy:", accuracy)

print("Error Rate:", error\_rate)

print("Precision:", precision)

print("Recall:", recall)

Performance Metrics:

True Positives (TP): 24

False Positives (FP): 3

True Negatives (TN): 65

False Negatives (FN): 8

Accuracy: 0.89

Error Rate: 0.11

Precision: 0.8888888888888888

Recall: 0.75

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js