# Logistic Regression

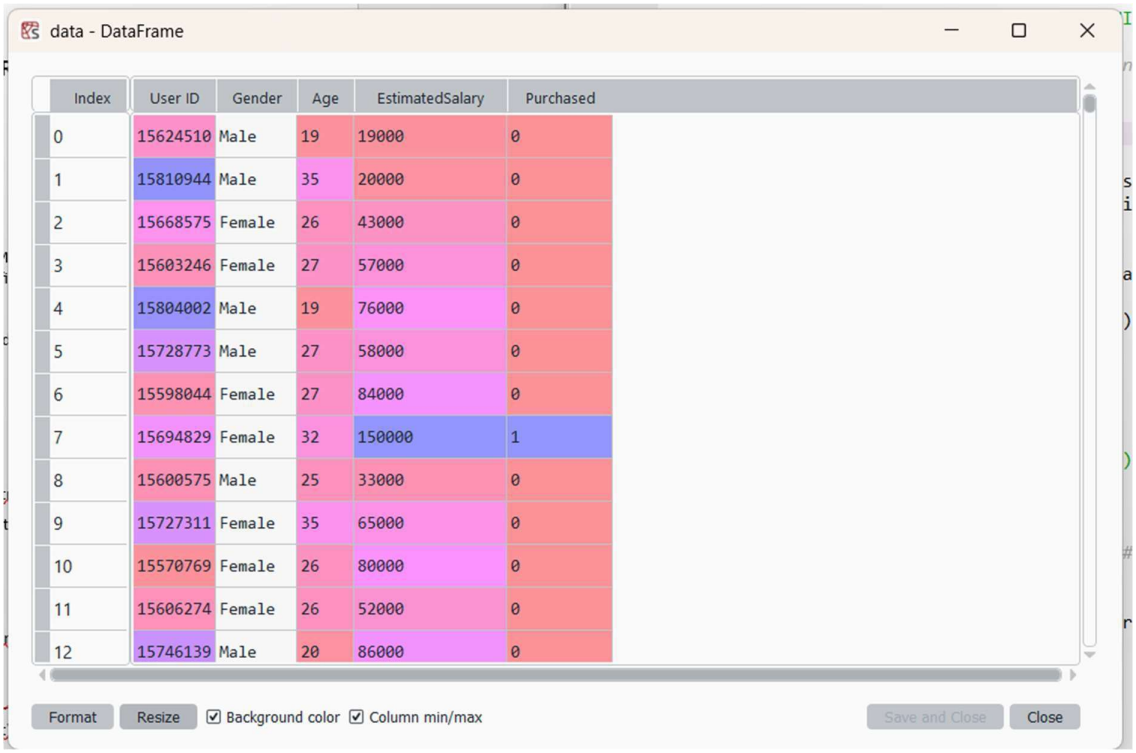
**#lets import libraries**

import pandas as pd

import matplotlib.pyplot as plt

**#lets read the dataset**

data=pd.read_csv(r"C:\Users\TharunMahendra\NIT\6.Algorithms\2.Classification\1.LogisticRegression\LogisticClassificationData.csv")

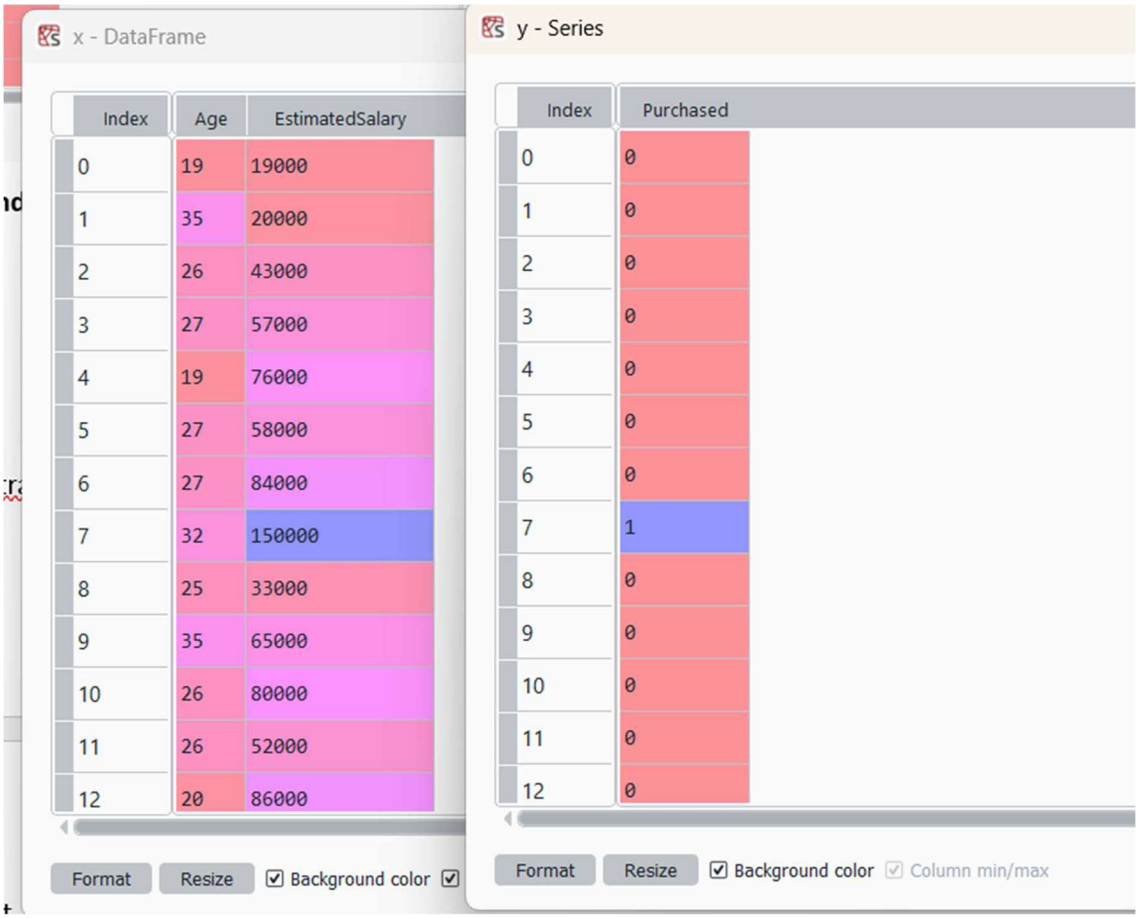



| Index | User ID | Gender | Age | EstimatedSalary | Purchased |
|---|---|---|---|---|---|
| 0 | 15624510 | Male | 19 | 19000 | 0 |
| 1 | 15810944 | Male | 35 | 20000 | 0 |
| 2 | 15668575 | Female | 26 | 43000 | 0 |
| 3 | 15603246 | Female | 27 | 57000 | 0 |
| 4 | 15804002 | Male | 19 | 76000 | 0 |
| 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 |
| 10 | 15570769 | Female | 26 | 80000 | 0 |
| 11 | 15606274 | Female | 26 | 52000 | 0 |
| 12 | 15746139 | Male | 20 | 86000 | 0 |

**#lets divide them into dependent & independent**

x=data.iloc[:,2:4] #age,salary

y=data.iloc[:,-1] #purchased




**#splitting data**

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,
random_state=0)
```

x_test - DataFrame

| Index | Age | EstimatedSalary |
|---|---|---|
| 132 | 30 | 87000 |
| 309 | 38 | 50000 |
| 341 | 35 | 75000 |
| 196 | 30 | 79000 |
| 246 | 35 | 50000 |
| 60 | 27 | 20000 |
| 155 | 31 | 15000 |
| 261 | 36 | 144000 |
| 141 | 18 | 68000 |
| 214 | 47 | 43000 |
| 37 | 30 | 49000 |
| 134 | 28 | 55000 |
| 113 | 37 | 55000 |

x_train - DataFrame

| Index | Age | EstimatedSalary |
|---|---|---|
| 336 | 58 | 144000 |
| 64 | 59 | 83000 |
| 55 | 24 | 55000 |
| 106 | 26 | 35000 |
| 300 | 58 | 38000 |
| 229 | 42 | 80000 |
| 122 | 40 | 75000 |
| 373 | 59 | 130000 |
| 395 | 46 | 41000 |
| 325 | 41 | 60000 |
| 380 | 42 | 64000 |
| 253 | 37 | 146000 |
| 56 | 23 | 48000 |

y_test - Series

| Index | Purchased |
|---|---|
| 132 | 0 |
| 309 | 0 |
| 341 | 0 |
| 196 | 0 |
| 246 | 0 |
| 60 | 0 |
| 155 | 0 |
| 261 | 1 |
| 141 | 0 |
| 214 | 0 |
| 37 | 0 |
| 134 | 0 |
| 113 | 0 |

y_train - Series

| Index | Purchased |
|---|---|
| 336 | 1 |
| 64 | 0 |
| 55 | 0 |
| 106 | 0 |
| 300 | 1 |
| 229 | 1 |
| 122 | 0 |
| 373 | 1 |
| 395 | 1 |
| 325 | 0 |
| 380 | 0 |
| 253 | 1 |
| 56 | 0 |

**#feature scaling**

from sklearn.preprocessing import **StandardScaler** **#range between-> -3to3**

featurescaling=StandardScaler()

x_train=featurescaling.fit_transform(x_train)

x_test=featurescaling.transform(x_test)

x_test - NumPy object array

| | 0 | 1 |
|---|---|---|
| 0 | -0.798951 | 0.494608 |
| 1 | -0.0212649 | -0.577359 |
| 2 | -0.312897 | 0.146943 |
| 3 | -0.798951 | 0.262831 |
| 4 | -0.312897 | -0.577359 |
| 5 | -1.09058 | -1.44652 |
| 6 | -0.70174 | -1.59138 |
| 7 | -0.215686 | 2.14602 |
| 8 | -1.96548 | -0.0558618 |
| 9 | 0.853632 | -0.780164 |
| 10 | -0.798951 | -0.606331 |
| 11 | -0.993372 | -0.432499 |
| 12 | -0.118476 | -0.432499 |

x_train - NumPy object array

| | 0 | 1 |
|---|---|---|
| 0 | 1.92295 | 2.14602 |
| 1 | 2.02016 | 0.378719 |
| 2 | -1.38222 | -0.432499 |
| 3 | -1.18779 | -1.01194 |
| 4 | 1.92295 | -0.925024 |
| 5 | 0.367578 | 0.291803 |
| 6 | 0.173157 | 0.146943 |
| 7 | 2.02016 | 1.74041 |
| 8 | 0.756421 | -0.838108 |
| 9 | 0.270367 | -0.287638 |
| 10 | 0.367578 | -0.17175 |
| 11 | -0.118476 | 2.20396 |
| 12 | -1.47943 | -0.635303 |

**#model building**

```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(x_train,y_train)
```

**#prediction**

```
y_pred=model.predict(x_test)
```

y_test - Series

| Index | Purchased |
|---|---|
| 132 | 0 |
| 309 | 0 |
| 341 | 0 |
| 196 | 0 |
| 246 | 0 |
| 60 | 0 |
| 155 | 0 |
| 261 | 1 |
| 141 | 0 |
| 214 | 0 |
| 37 | 0 |
| 134 | 0 |
| 113 | 0 |

y_pred - NumPy object array

| | 0 |
|---|---|
| 0 | 0 |
| 1 | 0 |
| 2 | 0 |
| 3 | 0 |
| 4 | 0 |
| 5 | 0 |
| 6 | 0 |
| 7 | 1 |
| 8 | 0 |
| 9 | 0 |
| 10 | 0 |
| 11 | 0 |
| 12 | 0 |

**#confusion Matrix**

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

| | 0 | 1 |
|---|---|---|
| 0 | 57 | 1 |
| 1 | 5 | 17 |

from sklearn.metrics import accuracy_score

ac=**accuracy_score**(y_test,y_pred) ->**0.925**

from sklearn.metrics import classification_report

cr=classification_report(y_test,y_pred)

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.92 | 0.98 | 0.95 | 58 |
| 1 | 0.94 | 0.77 | 0.85 | 22 |
| accuracy | | | 0.93 | 80 |
| macro avg | 0.93 | 0.88 | 0.90 | 80 |
| weighted avg | 0.93 | 0.93 | 0.92 | 80 |

**bias**=model.score(x_train,y_train) **-> 0.821875**

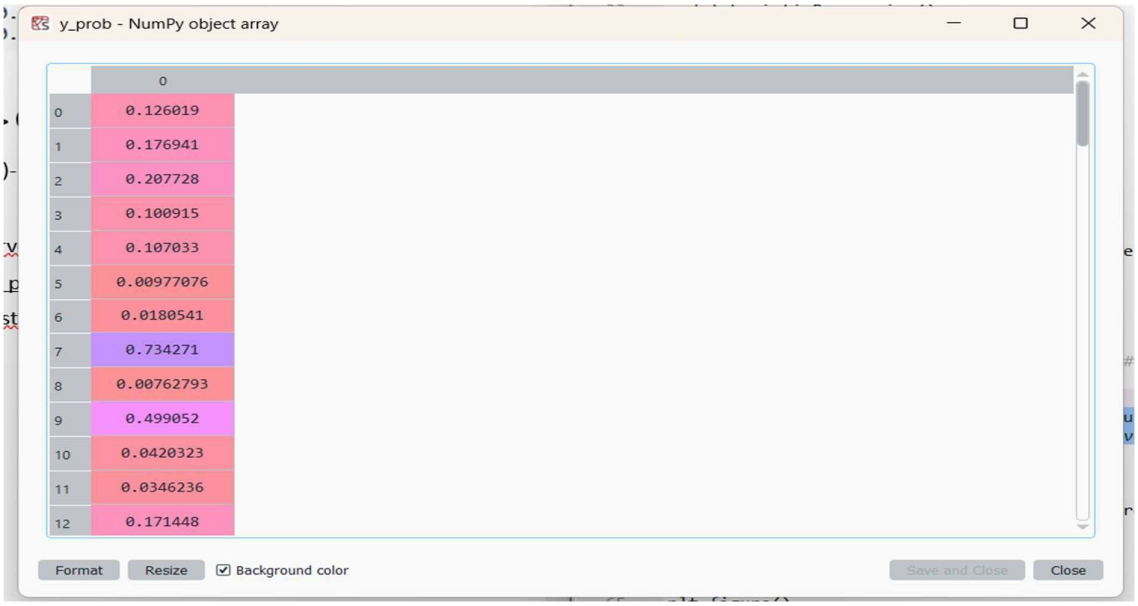
**variance**=model.score(x_test,y_test)-**> 0.925**

**#plotting graph**

from sklearn.metrics import roc_curve, roc_auc_score

# Get predicted probabilities for the positive class (usually class 1)

y_prob = model.predict_proba(x_test)[:, 1]



**# Compute ROC curve and AUC**

fpr, tpr, thresholds = roc_curve(y_test, y_prob)

auc_score = roc_auc_score(y_test, y_prob)

print("AUC Score:", auc_score) **AUC Score: 0.9764890282131661**

**# Plot ROC curve**

plt.figure()

plt.plot(fpr, tpr, color='blue', label='ROC curve (area = %0.2f)' % auc_score)

plt.plot([0, 1], [0, 1], color='gray', linestyle='--') # Diagonal line

plt.xlabel('False Positive Rate')

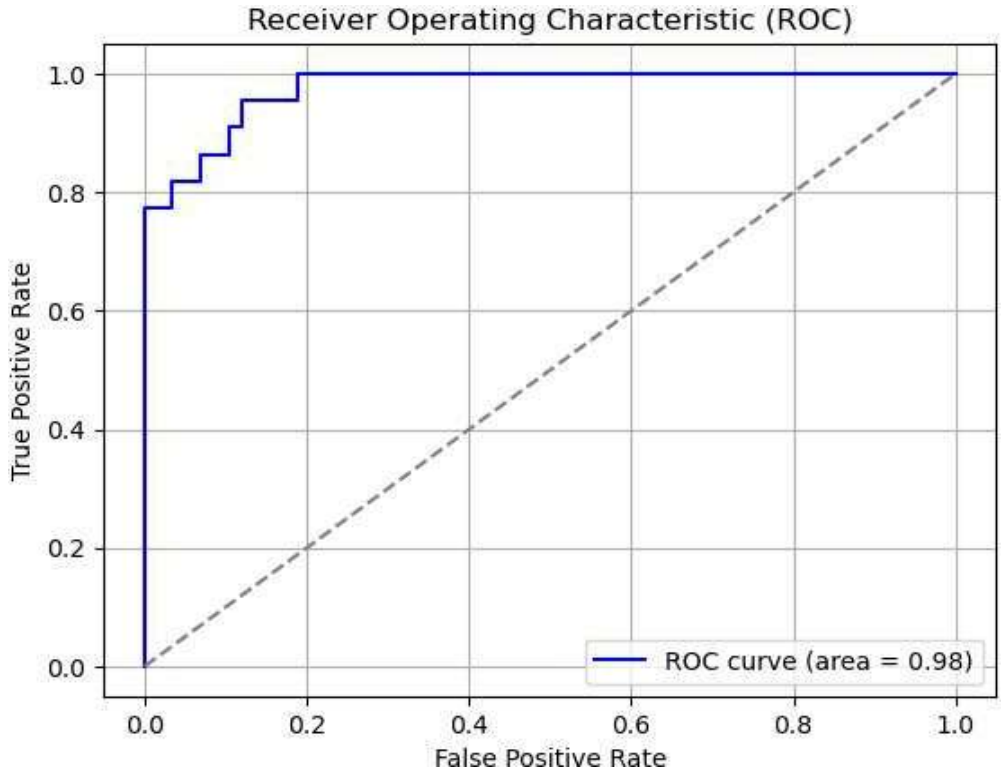
plt.ylabel('True Positive Rate')

plt.title('Receiver Operating Characteristic (ROC)')

plt.legend()

plt.grid()

plt.show()



**#feature scaling**

from sklearn.preprocessing import **Normalizer** **#range between-> 0to1**

featurescaling=Normalizer()

x_train=featurescaling.fit_transform(x_train)

x_test=featurescaling.transform(x_test)

x_test - NumPy object array

| | 0 | 1 |
|---|---|---|
| 0 | 0.000344828 | 1 |
| 1 | 0.00076 | 1 |
| 2 | 0.000466667 | 1 |
| 3 | 0.000379747 | 1 |
| 4 | 0.0007 | 1 |
| 5 | 0.00135 | 0.999999 |
| 6 | 0.00206666 | 0.999998 |
| 7 | 0.00025 | 1 |
| 8 | 0.000264706 | 1 |
| 9 | 0.00109302 | 0.999999 |
| 10 | 0.000612245 | 1 |
| 11 | 0.000509091 | 1 |
| 12 | 0.000672727 | 1 |

x_train - NumPy object array

| | 0 | 1 |
|---|---|---|
| 0 | 0.000402778 | 1 |
| 1 | 0.000710843 | 1 |
| 2 | 0.000436364 | 1 |
| 3 | 0.000742857 | 1 |
| 4 | 0.00152631 | 0.999999 |
| 5 | 0.000525 | 1 |
| 6 | 0.000533333 | 1 |
| 7 | 0.000453846 | 1 |
| 8 | 0.00112195 | 0.999999 |
| 9 | 0.000683333 | 1 |
| 10 | 0.00065625 | 1 |
| 11 | 0.000253425 | 1 |
| 12 | 0.000479167 | 1 |

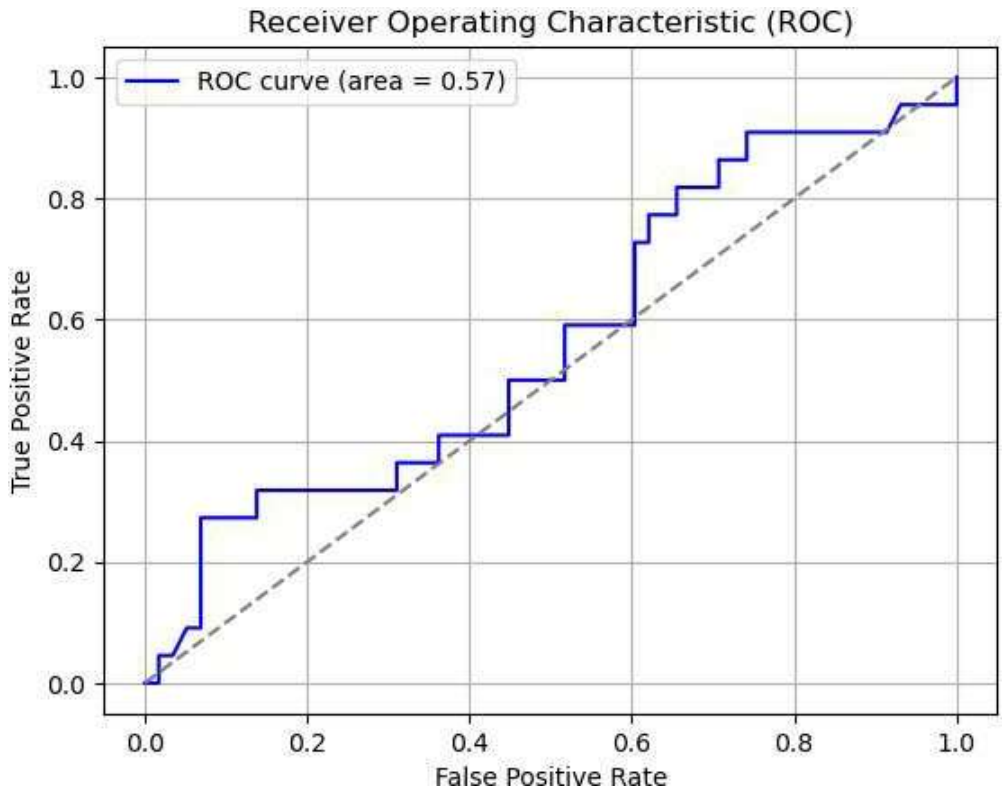
**#Confusion Matrix**

| | 0 | 1 |
|---|---|---|
| 0 | 58 | 0 |
| 1 | 22 | 0 |

Accuracy -> **0.725**

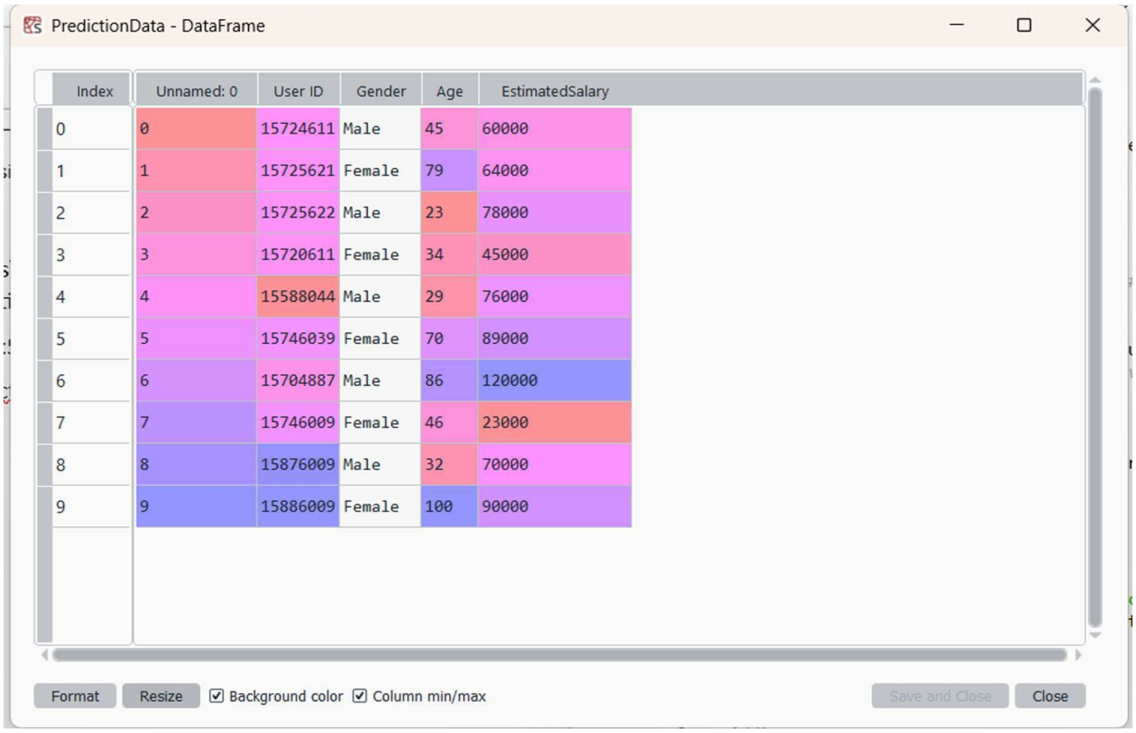
Bias-> **0.621875**

Variance-> **0.725**




**#Let's Predict Future Based on Data**

PredictionData=pd.read_csv(r"C:\Users\TharunMahendra\NIT\6.Algorithms\2. Classification\1.LogisticRegression\TestingData.csv")




| Index | Unnamed: 0 | User ID | Gender | Age | EstimatedSalary |
|---|---|---|---|---|---|
| 0 | 0 | 15724611 | Male | 45 | 60000 |
| 1 | 1 | 15725621 | Female | 79 | 64000 |
| 2 | 2 | 15725622 | Male | 23 | 78000 |
| 3 | 3 | 15720611 | Female | 34 | 45000 |
| 4 | 4 | 15588044 | Male | 29 | 76000 |
| 5 | 5 | 15746039 | Female | 70 | 89000 |
| 6 | 6 | 15704887 | Male | 86 | 120000 |
| 7 | 7 | 15746009 | Female | 46 | 23000 |
| 8 | 8 | 15876009 | Male | 32 | 70000 |
| 9 | 9 | 15886009 | Female | 100 | 90000 |

FutureData=PredictionData.copy()
PredictionData=PredictionData.iloc[:,3:5]

PredictionData - DataFrame

| Index | Age | EstimatedSalary |
|---|---|---|
| 0 | 45 | 60000 |
| 1 | 79 | 64000 |
| 2 | 23 | 78000 |
| 3 | 34 | 45000 |
| 4 | 29 | 76000 |
| 5 | 70 | 89000 |
| 6 | 86 | 120000 |
| 7 | 46 | 23000 |
| 8 | 32 | 70000 |
| 9 | 100 | 90000 |

PredictionData=featurescaling.fit_transform(PredictionData)

PredictionData - NumPy object array

| | 0 | 1 |
|---|---|---|
| 0 | -0.364945 | -0.457862 |
| 1 | 0.955068 | -0.298606 |
| 2 | -1.21907 | 0.258792 |
| 3 | -0.792008 | -1.05507 |
| 4 | -0.986127 | 0.179163 |
| 5 | 0.605653 | 0.696747 |
| 6 | 1.22684 | 1.93098 |
| 7 | -0.326121 | -1.93098 |
| 8 | -0.869655 | -0.0597212 |
| 9 | 1.77037 | 0.736561 |

```
FuturePrediction=pd.DataFrame()
FutureData['FuturePrediction']=model.predict(PredictionData)
FutureData.to_csv('PredictedData')
```

FutureData - DataFrame

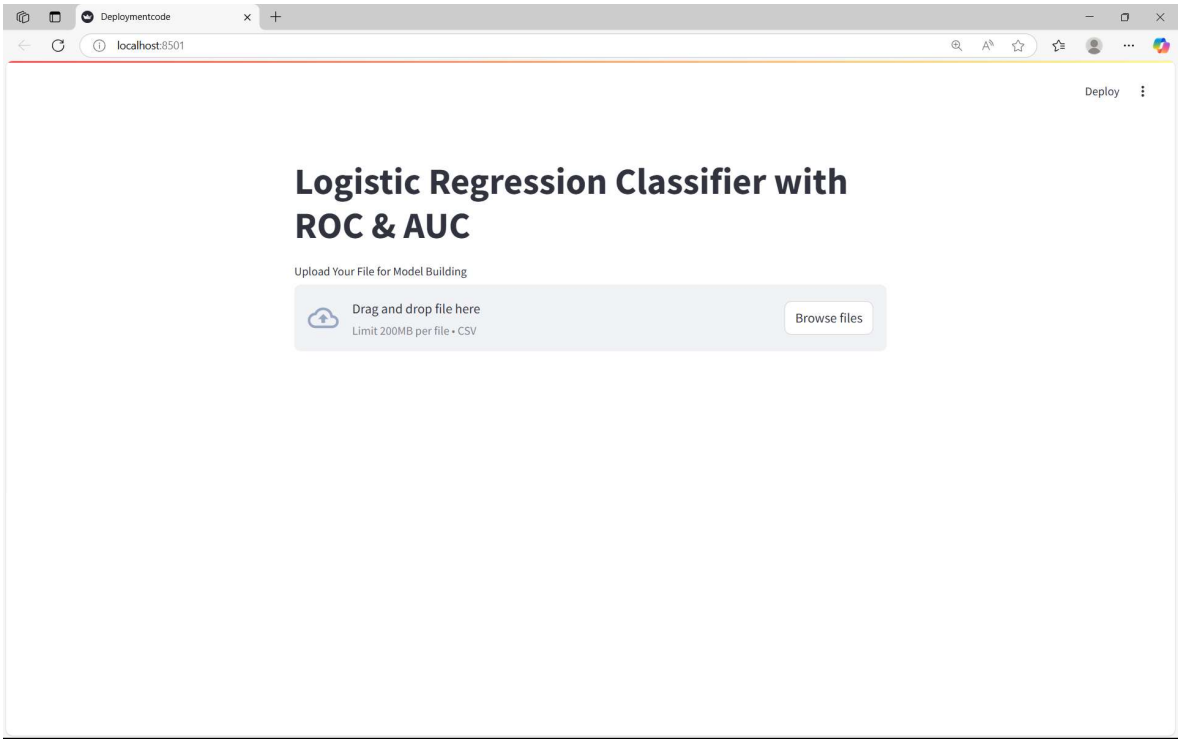
| Index | Unnamed: 0 | User ID | Gender | Age | EstimatedSalary | FuturePrediction |
|---|---|---|---|---|---|---|
| 0 | 0 | 15724611 | Male | 45 | 60000 | 0 |
| 1 | 1 | 15725621 | Female | 79 | 64000 | 1 |
| 2 | 2 | 15725622 | Male | 23 | 78000 | 0 |
| 3 | 3 | 15720611 | Female | 34 | 45000 | 0 |
| 4 | 4 | 15588044 | Male | 29 | 76000 | 0 |
| 5 | 5 | 15746039 | Female | 70 | 89000 | 1 |
| 6 | 6 | 15704887 | Male | 86 | 120000 | 1 |
| 7 | 7 | 15746009 | Female | 46 | 23000 | 0 |
| 8 | 8 | 15876009 | Male | 32 | 70000 | 0 |
| 9 | 9 | 15886009 | Female | 100 | 90000 | 1 |

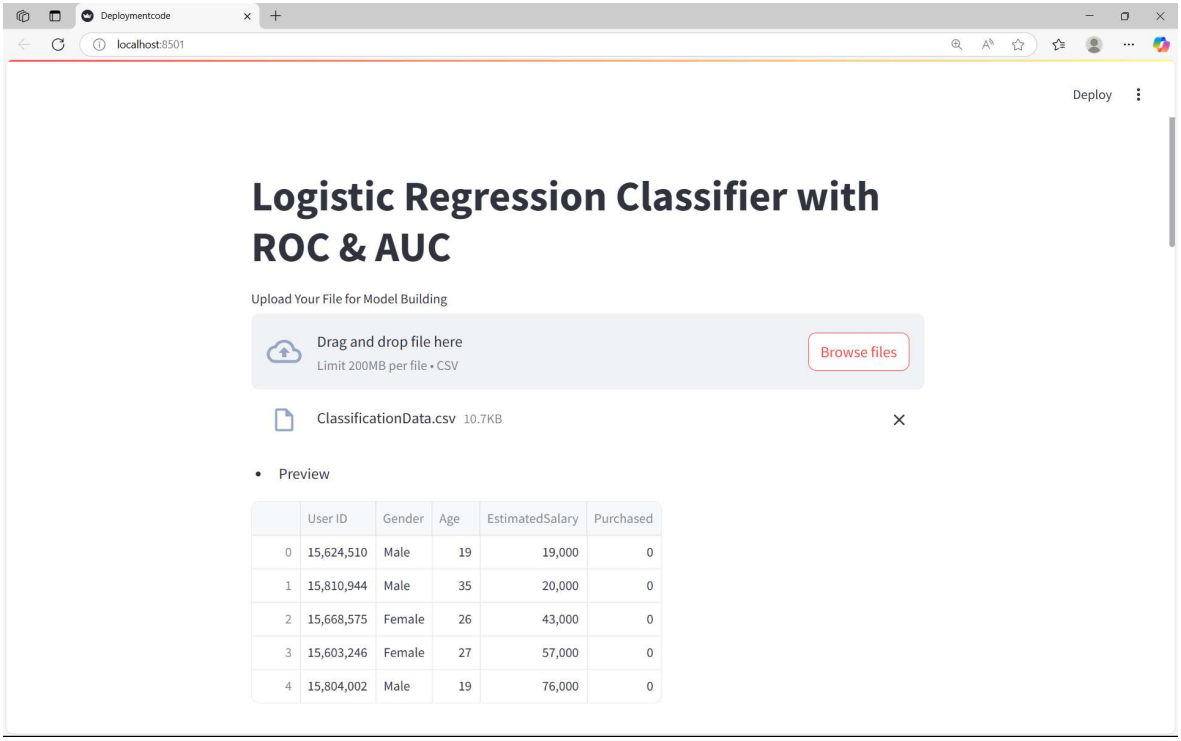
# Deployment Code

```
# importing libraries
import streamlit as st
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
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,accuracy_score,classification_report,roc_curve,roc_auc_score)
import seaborn as sns

st.title("Logistic Regression Classifier with ROC & AUC")

# Uploading File
file=st.file_uploader('Upload Your File for Model Building',type=['csv'])
if file is not None:
    # Load
    data=pd.read_csv(file)
    st.write('- Preview')
    st.dataframe(data.head())

    # FeatureSelection
    x=data.iloc[:,2:4]
    y=data.iloc[:,-1]
```

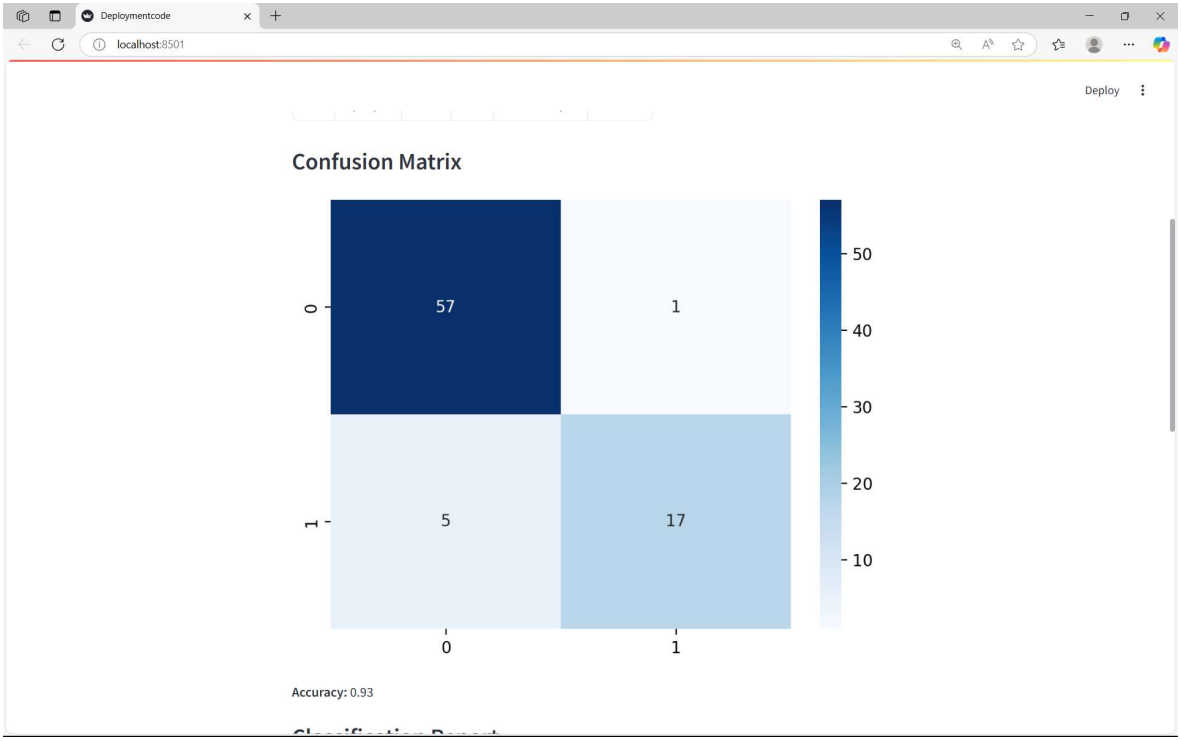
```
# SplittingData
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)

# feature scaling
featurescaling=StandardScaler()
x_train=featurescaling.fit_transform(x_train)
x_test=featurescaling.transform(x_test)

# model building
model=LogisticRegression(penalty='l2',solver='saga')
model.fit(x_train,y_train)

# prediction
y_pred=model.predict(x_test)
y_prob=model.predict_proba(x_test)[:,1]

# Metrics
st.subheader("Confusion Matrix")
cm = confusion_matrix(y_test, y_pred)
fig_cm, ax = plt.subplots()
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", ax=ax)
st.pyplot(fig_cm)

ac = accuracy_score(y_test, y_pred)
st.write(f"**Accuracy:** {ac:.2f}")
```

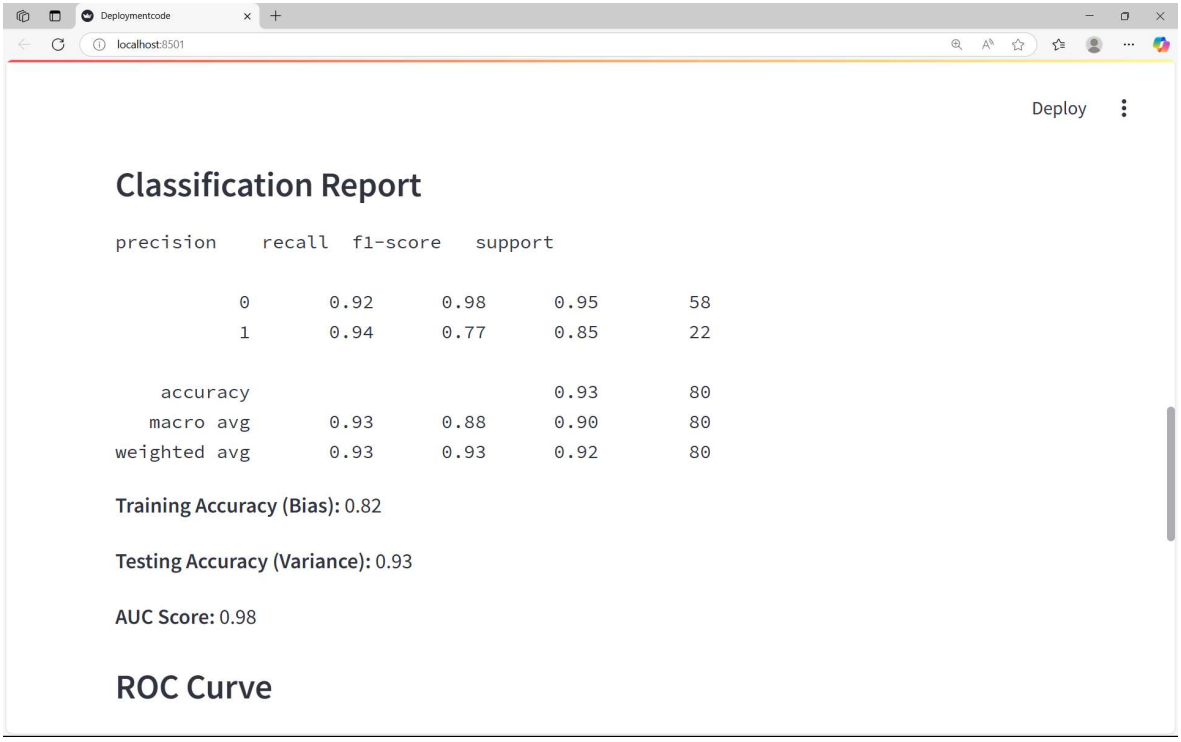
```
    st.subheader("Classification Report")
    st.text(classification_report(y_test, y_pred))

    st.write(f"**Training Accuracy (Bias):** {model.score(x_train, y_train):.2f}")
    st.write(f"**Testing Accuracy (Variance):** {model.score(x_test, y_test):.2f}")

    # ROC Curve and AUC
    fpr, tpr, _ = roc_curve(y_test, y_prob)
    auc_score = roc_auc_score(y_test, y_prob)
    st.write(f"**AUC Score:** {auc_score:.2f}")

    st.subheader("ROC Curve")
    fig_roc, ax = plt.subplots()
    ax.plot(fpr, tpr, color="blue", label=f"ROC curve (AUC = {auc_score:.2f})")
    ax.plot([0, 1], [0, 1], color="gray", linestyle="--")
    ax.set_xlabel("False Positive Rate")
    ax.set_ylabel("True Positive Rate")
    ax.set_title("ROC Curve")
    ax.legend(loc="lower right")
    st.pyplot(fig_roc)
```

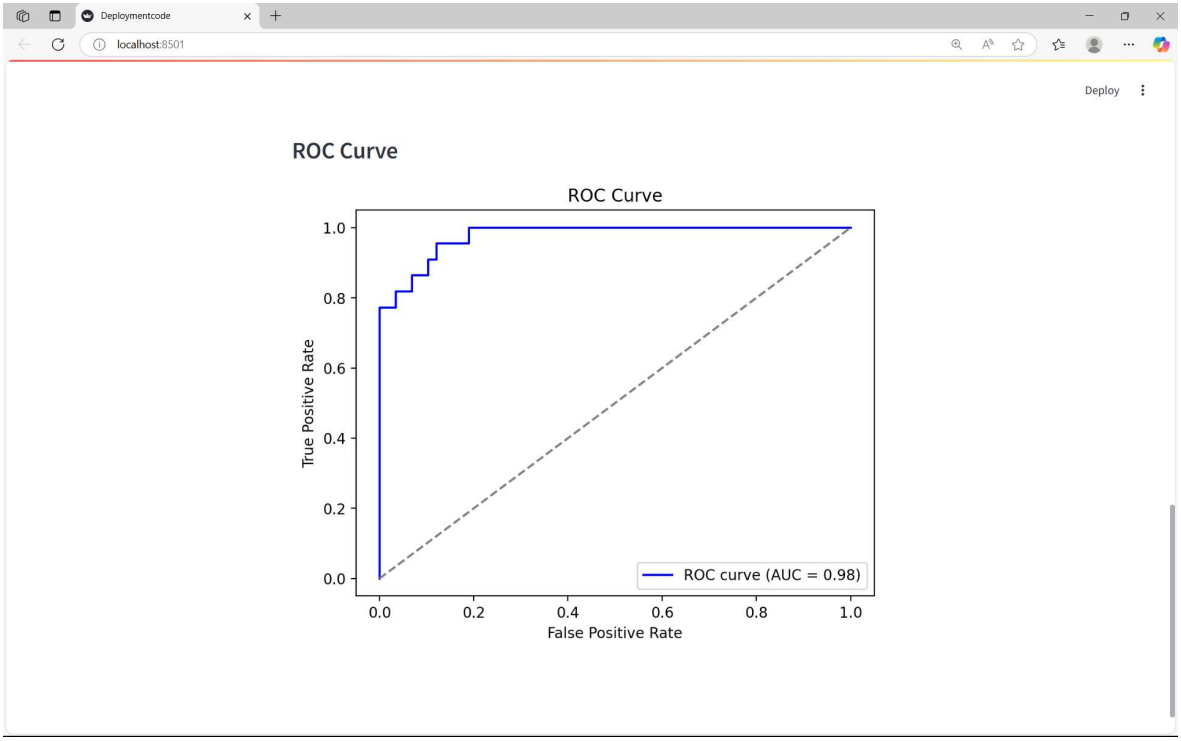



| | User ID | Gender | Age | EstimatedSalary | Purchased |
|---|---|---|---|---|---|
| 0 | 15,624,510 | Male | 19 | 19,000 | 0 |
| 1 | 15,810,944 | Male | 35 | 20,000 | 0 |
| 2 | 15,668,575 | Female | 26 | 43,000 | 0 |
| 3 | 15,603,246 | Female | 27 | 57,000 | 0 |
| 4 | 15,804,002 | Male | 19 | 76,000 | 0 |


Deployment code
localhost:8501
Deploy
Confusion Matrix
57
1
5
17
0
1
0
1
50
40
30
20
10
Accuracy: 0.93


Deployment code
localhost:8501
Deploy
Classification Report
precision recall f1-score support
0 0.92 0.98 0.95 58
1 0.94 0.77 0.85 22
accuracy 0.93 80
macro avg 0.93 0.88 0.90 80
weighted avg 0.93 0.93 0.92 80
Training Accuracy (Bias): 0.82
Testing Accuracy (Variance): 0.93
AUC Score: 0.98
ROC Curve


Deploymentcode
localhost:8501
Deploy
ROC Curve
ROC Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (AUC = 0.98)