ass5_mnish

February 5, 2026

[23]:
```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

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, precision_score,␣
 ↪recall_score
from sklearn.metrics import roc_curve, auc

%matplotlib inline
```

[25]:
```
dataset = pd.read_csv('Social_Network_Ads.csv')

dataset.columns = dataset.columns.str.strip()

print(dataset.head())

print(dataset.info())
```

```
    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
<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
None
```

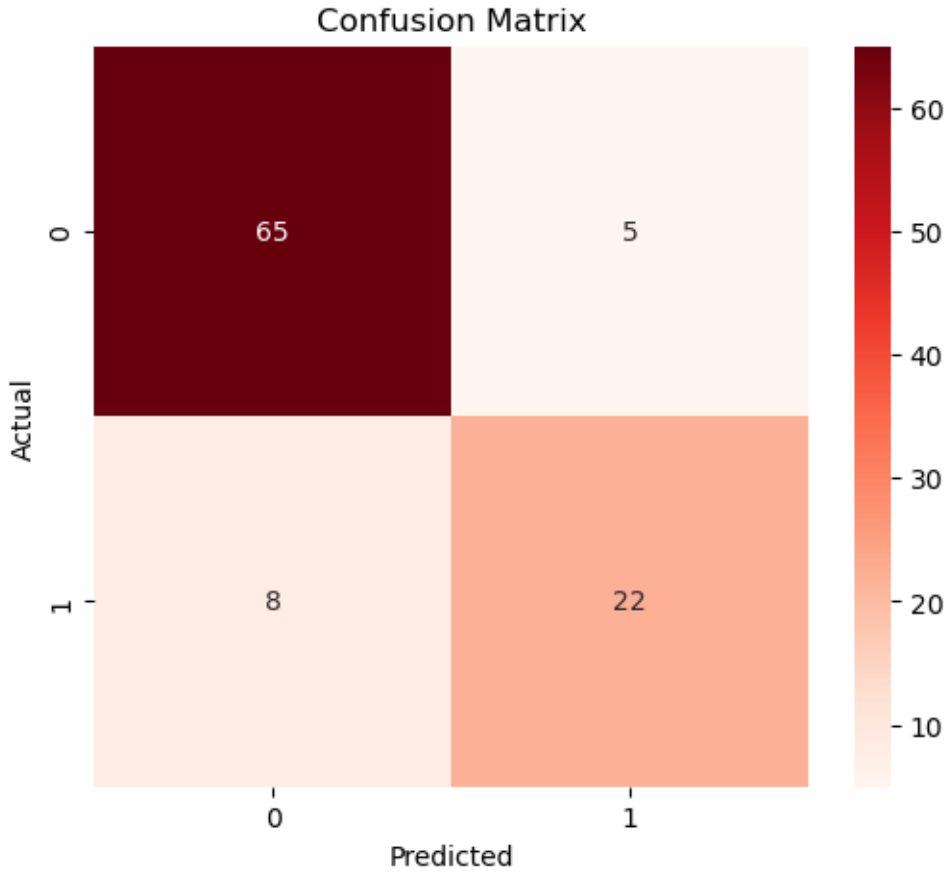
[27]:
```
dataset = dataset.drop(['User ID'], axis=1)

dataset['Gender'] = dataset['Gender'].map({'Male':1, 'Female':0})

X = dataset[['Gender', 'Age', 'EstimatedSalary']]
y = dataset['Purchased']


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,␣
 ↪random_state=4)


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

[29]:
```
model = LogisticRegression(random_state=0)

model.fit(X_train, y_train)

y_pred = model.predict(X_test)
```

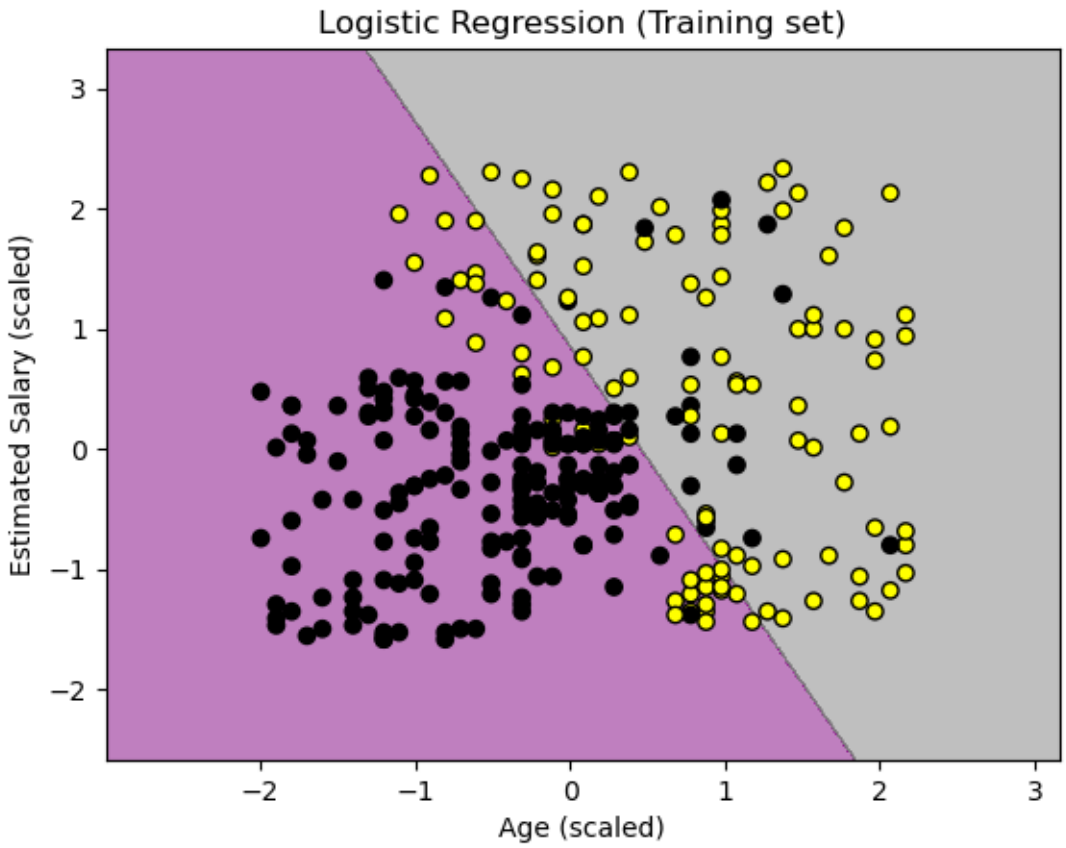
[31]:
```
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)

TN, FP, FN, TP = cm.ravel()
print(f"True Positive (TP): {TP}")
print(f"False Positive (FP): {FP}")
print(f"True Negative (TN): {TN}")
print(f"False Negative (FN): {FN}")

accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

error_rate = 1 - accuracy
print("Error Rate:", error_rate)

precision = precision_score(y_test, y_pred)
print("Precision:", precision)

recall = recall_score(y_test, y_pred)
print("Recall:", recall)
```

```
Confusion Matrix:
 [[65  5]
 [ 8 22]]
True Positive (TP): 22
False Positive (FP): 5
True Negative (TN): 65
False Negative (FN): 8
Accuracy: 0.87
Error Rate: 0.13
Precision: 0.8148148148148148
Recall: 0.7333333333333333
```

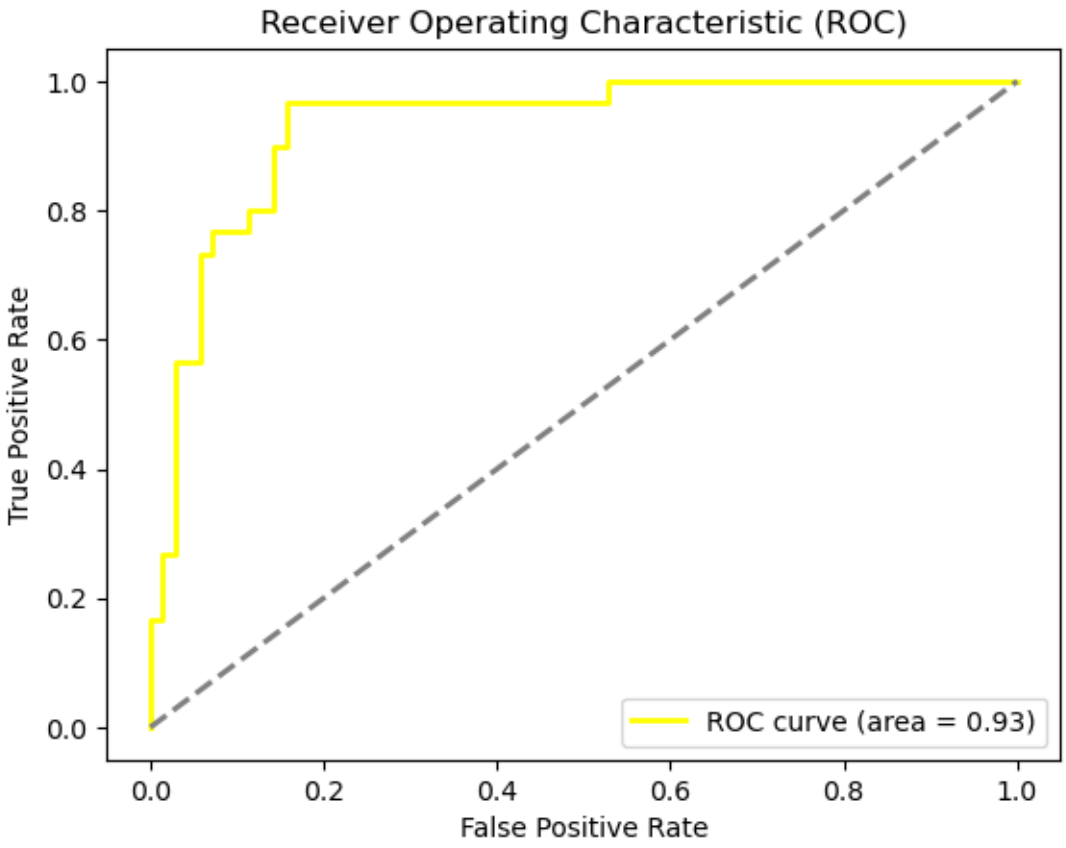
[33]:
```
plt.figure(figsize=(6,5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Reds')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```



[35]:
```
X_vis = dataset[['Age','EstimatedSalary']].values
y_vis = dataset['Purchased'].values

X_train_vis, X_test_vis, y_train_vis, y_test_vis = train_test_split(X_vis,␣
 ↪y_vis, test_size=0.25, random_state=0)
sc_vis = StandardScaler()
X_train_vis = sc_vis.fit_transform(X_train_vis)
X_test_vis = sc_vis.transform(X_test_vis)

model_vis = LogisticRegression(random_state=0)
model_vis.fit(X_train_vis, y_train_vis)

from matplotlib.colors import ListedColormap
X_set, y_set = X_train_vis, y_train_vis
X1, X2 = np.meshgrid(np.arange(start=X_set[:,0].min()-1, stop=X_set[:,0].
 ↪max()+1, step=0.01),
                     np.arange(start=X_set[:,1].min()-1, stop=X_set[:,1].
 ↪max()+1, step=0.01))
plt.contourf(X1, X2, model_vis.predict(np.array([X1.ravel(), X2.ravel()]).T).
 ↪reshape(X1.shape),
             alpha=0.5, cmap=ListedColormap(('purple', 'gray')))
plt.scatter(X_set[:,0], X_set[:,1], c=y_set,␣
 ↪cmap=ListedColormap(('black','yellow')), edgecolor='k')
plt.title('Logistic Regression (Training set)')
plt.xlabel('Age (scaled)')
plt.ylabel('Estimated Salary (scaled)')
plt.show()
```



[37]:
```
y_prob = model.predict_proba(X_test)[:,1]

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

plt.figure()
plt.plot(fpr, tpr, color='yellow', lw=2, label='ROC curve (area = %.2f)' %␣
 ↪roc_auc)
plt.plot([0,1], [0,1], color='gray', lw=2, linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)')
plt.legend(loc='lower right')
plt.show()
```


Receiver Operating Characteristic (ROC)
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.93)

[ ]:

[ ]:

[ ]: