PRAC 5

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score,
f1_score



 Importing DataSet and take a look at Data
df = pd.read_csv("socialnetwork_ads.csv")
df.head()

# Select relevant features and target
X = df[['Age', 'EstimatedSalary']]
y = df['Purchased']

# Split the data into training and test sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)



# Create logistic regression model
log_model = LogisticRegression()

# Fit the model
log_model.fit(X_train, y_train)

# Predict on test data
y_pred = log_model.predict(X_test)



# Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Extract TP, FP, TN, FN
TP = conf_matrix[1, 1]  # True Positive
FP = conf_matrix[0, 1]  # False Positive
TN = conf_matrix[0, 0]  # True Negative
FN = conf_matrix[1, 0]  # False Negative

# Calculate Accuracy, Error rate, Precision, Recall
accuracy = accuracy_score(y_test, y_pred)
error_rate = 1 - accuracy
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)



# Print the results
```

```
print(f"Confusion Matrix:\n{conf_matrix}")
print(f"True Positive (TP): {TP}")
print(f"False Positive (FP): {FP}")
print(f"True Negative (TN): {TN}")
print(f"False Negative (FN): {FN}")
print(f"Accuracy: {accuracy:.4f}")
print(f"Error Rate: {error_rate:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")



# Create a meshgrid with reduced resolution to prevent memory overload
x_min, x_max = X['Age'].min() - 1, X['Age'].max() + 1
y_min, y_max = X['EstimatedSalary'].min() - 1, X['EstimatedSalary'].max() + 1

# Increase the step size to reduce the grid size
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.5),  # Change step size from 0.1 to 0.5
                     np.arange(y_min, y_max, 0.5))  # Change step size from 0.1 to 0.5

# Predict the class for each point in the meshgrid
Z = log_model.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

# Plot the decision boundary and the points
plt.contourf(xx, yy, Z, alpha=0.8, cmap='coolwarm')
plt.scatter(X['Age'], X['EstimatedSalary'], c=y, cmap='coolwarm', s=30, edgecolor='k')
plt.title('Logistic Regression Decision Boundary')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.show()
```