Name	**Sujal Sandeep Dingankar**
UID no.	**2024301005**
Experiment No.	**9**

AIM:	**Logistic Regression**
Program 1	
PROBLEM STATEMENT :	Logistic Regression and Regularization. • Get a data set which requires logistic regression. • Show overfitting using MSE, bias and variance. • Use L1 regularization to solve overfitting. • Use L2 regularization to solve overfitting. • Use Elastic Net to solve overfitting. • (You may Use different data set for show overfitting and regularization)
PROGRAM:	

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split,
cross_val_score

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import mean_squared_error,
accuracy_score, roc_auc_score

# Import OneHotEncoder for encoding categorical features

from sklearn.preprocessing import OneHotEncoder
# Load your dataset

data = pd.read_csv("/content/logistic regression dataset-
Social_Network_Ads.csv")

X = data.drop("Purchased", axis=1)

y = data["Purchased"]
```

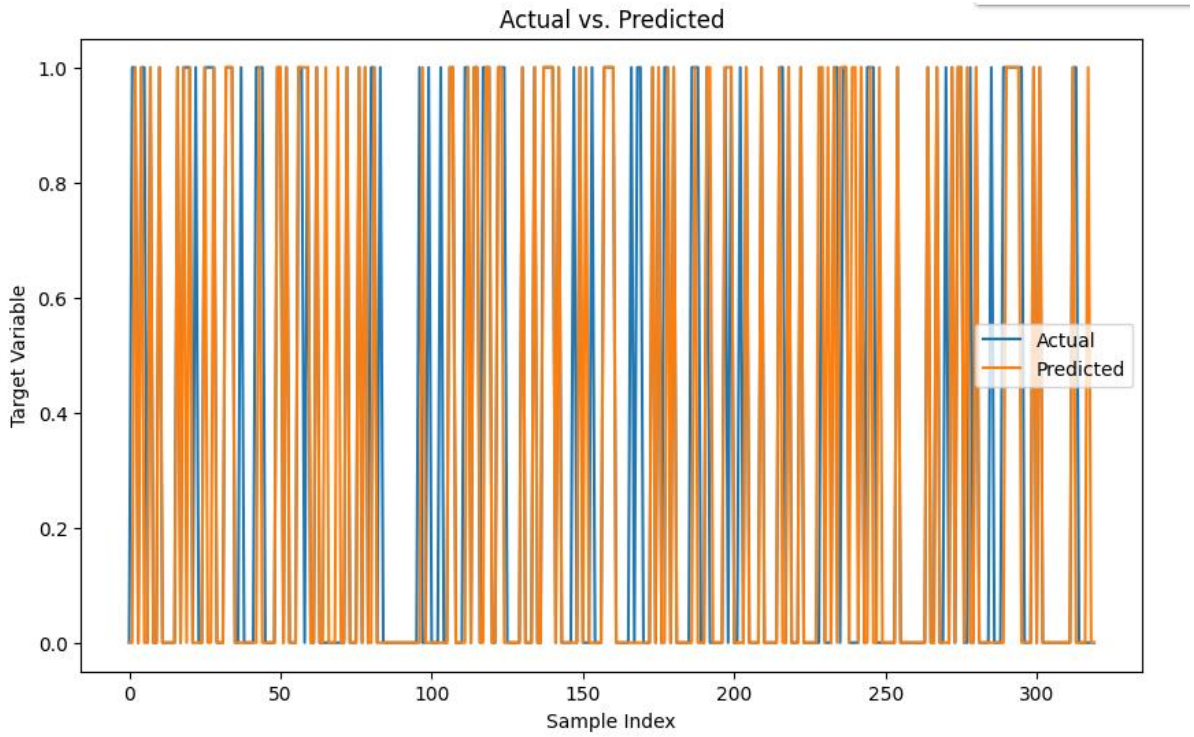
```
encoder = OneHotEncoder(sparse_output=False,
handle_unknown='ignore')

# Fit the encoder on the categorical features and
transform them

# Assuming 'Gender' is the only categorical column, adjust
if needed

categorical_features = ['Gender']

encoded_features =
encoder.fit_transform(X[categorical_features])

# Create a DataFrame from the encoded features

encoded_df = pd.DataFrame(encoded_features,
columns=encoder.get_feature_names_out(categorical_feature
s))

# Drop the original categorical features and concatenate
the encoded features

X = X.drop(categorical_features, axis=1)

X = pd.concat([X, encoded_df], axis=1)

# Split the data into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Create a logistic regression model

model = LogisticRegression()

# Train the model

model.fit(X_train, y_train)

# Evaluate the model on the training and testing sets

y_pred_train = model.predict(X_train)

y_pred_test = model.predict(X_test)

mse_train = mean_squared_error(y_train, y_pred_train)
```

```
mse_test = mean_squared_error(y_test, y_pred_test)

accuracy_train = accuracy_score(y_train, y_pred_train)

accuracy_test = accuracy_score(y_test, y_pred_test)

auc_train = roc_auc_score(y_train,
model.predict_proba(X_train)[:, 1])

auc_test = roc_auc_score(y_test,
model.predict_proba(X_test)[:, 1])

print("MSE (Training):", mse_train)

print("MSE (Testing):", mse_test)

print("Accuracy (Training):", accuracy_train)

print("Accuracy (Testing):", accuracy_test)

print("AUC (Training):", auc_train)

print("AUC (Testing):", auc_test)

# Visualize the results (optional)

plt.figure(figsize=(10, 6))

plt.plot(range(len(y_train)), y_train, label="Actual")

plt.plot(range(len(y_train)), y_pred_train,
label="Predicted")

plt.title("Actual vs. Predicted")

plt.xlabel("Sample Index")

plt.ylabel("Target Variable")

plt.legend()

plt.show()

# Implement regularization

models = {"Original": model,
"L1 Regularization": LogisticRegression(penalty="l1",
solver="liblinear"),
```

```
"L2 Regularization": LogisticRegression(penalty="l2"),
"Elastic Net": LogisticRegression(penalty="elasticnet",
solver="saga", l1_ratio=0.5)
}

for name, model in models.items():

    model.fit(X_train, y_train)


    # Evaluate the model using cross-validation

    cv_scores = cross_val_score(model, X, y, cv=5,
scoring="accuracy")

    print(f"\n{name}:")

    print(f"Cross-Validation Accuracy:
{cv_scores.mean():.3f} ± {cv_scores.std():.3f}")

    # Evaluate the model on the testing set

    y_pred_test = model.predict(X_test)

    mse_test = mean_squared_error(y_test, y_pred_test)

    accuracy_test = accuracy_score(y_test, y_pred_test)

    auc_test = roc_auc_score(y_test,
model.predict_proba(X_test)[:, 1])

    print(f"MSE (Testing): {mse_test:.3f}")

    print(f"Accuracy (Testing): {accuracy_test:.3f}")

    print(f"AUC (Testing): {auc_test:.3f}")
```

RESULT:

```
MSE (Training): 0.159375
MSE (Testing): 0.1125
Accuracy (Training): 0.840625
Accuracy (Testing): 0.8875
AUC (Training): 0.9146129374337222
AUC (Testing): 0.9690934065934066
```




```
Original:
Cross-Validation Accuracy: 0.807 ± 0.116
MSE (Testing): 0.113
Accuracy (Testing): 0.887
AUC (Testing): 0.969

 L1 Regularization:
 Cross-Validation Accuracy: 0.815 ± 0.114
 MSE (Testing): 0.113
 Accuracy (Testing): 0.887
 AUC (Testing): 0.969

 L2 Regularization:
 Cross-Validation Accuracy: 0.807 ± 0.116
 MSE (Testing): 0.113
 Accuracy (Testing): 0.887
 AUC (Testing): 0.969

Elastic Net:
Cross-Validation Accuracy: 0.643 ± 0.006
MSE (Testing): 0.350
Accuracy (Testing): 0.650
AUC (Testing): 0.641
```

CONCLUSION:	I implemented logistic regression and used L1, L2, and Elastic Net regularization to reduce overfitting. L1 created sparse models, L2 balanced complexity, and Elastic Net handled correlated features. Regularization improved generalization and model performance.