

Logistic Regression

```
# Logistic Regression on SUV dataset
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, classification_report, accuracy_score
```

```
data = pd.read_csv("suv dataset.csv")
data.head()
```

	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

```
data.shape
```

```
(400, 5)
```

```
data['Age'].unique()
```

```
array([19, 35, 26, 27, 32, 25, 20, 18, 29, 47, 45, 46, 48, 49, 31, 21, 28,
       33, 30, 23, 24, 22, 59, 34, 39, 38, 37, 42, 40, 36, 41, 58, 55, 52,
       60, 56, 53, 50, 51, 57, 44, 43, 54], dtype=int64)
```

```
for col in data.columns:
    print(f"Unique values in '{col}': {data[col].unique()}")
```

```
15595228 15782530 15592877 15651983 15746737 15774179 15667265 15655123
15595917 15668385 15709476 15711218 15798659 15663939 15694946 15631912
15768816 15682268 15684801 15636428 15809823 15699284 15786993 15709441
15710257 15582492 15575694 15756820 15766289 15593014 15584545 15675949
15672091 15801658 15706185 15789863 15720943 15697997 15665416 15660200
15619653 15773447 15739160 15689237 15679297 15591433 15642725 15701962
15811613 15741049 15724423 15574305 15678168 15697020 15610801 15745232
15722758 15792102 15675185 15801247 15725660 15638963 15800061 15578006
15668504 15687491 15610403 15741094 15807909 15666141 15617134 15783029
15622833 15746422 15750839 15749130 15779862 15767871 15679651 15576219
15699247 15619087 15605327 15610140 15791174 15602373 15762605 15598840
15744279 15670619 15599533 15757837 15697574 15578738 15762228 15614827
15789815 15579781 15587013 15570932 15794661 15581654 15644296 15614420
15609653 15594577 15584114 15673367 15685576 15774727 15694288 15603319
15759066 15814816 15724402 15571059 15674206 15715160 15730448 15662067
15779581 15662901 15689751 15667742 15738448 15680243 15745083 15708228
15628523 15708196 15735549 15809347 15660866 15766609 15654230 15794566
15800890 15697424 15724536 15735878 15707596 15657163 15622478 15779529
15636023 15582066 15666675 15732987 15789432 15663161 15694879 15593715
15575002 15622171 15795224 15685346 15691808 15721007 15794253 15694453
15813113 15614187 15619407 15646227 15660541 15753874 15617877 15772073
15701537 15736228 15780572 15769596 15586996 15722061 15638003 15775590
15730688 15753102 15810075 15723373 15795298 15584320 15724161 15750056
15609637 15794493 15569641 15815236 15811177 15680587 15672821 15767681
15600379 15801336 15721592 15581282 15746203 15583137 15680752 15688172
15791373 15589449 15692819 15727467 15734312 15764604 15613014 15759684
15609669 15685536 15750447 15663249 15638646 15734161 15631070 15761950
15649668 15713912 15586757 15596522 15625395 15760570 15566689 15725794
15673539 15705298 15675791 15747043 15736397 15678201 15720745 15637593
15598070 15787550 15603942 15733973 15596761 15652400 15717893 15622585
15733964 15753861 15747097 15594762 15667417 15684861 15742204 15623502
15774872 15611191 15674331 15619465 15575247 15695679 15713463 15785170
15796351 15639576 15693264 15589715 15769902 15587177 15814553 15601550
```

```
Unique values in 'Age': [19 35 26 27 32 25 20 18 29 47 45 46 48 49 31 21 28 33 30 23 24 22 59 34
39 38 37 42 40 36 41 58 55 52 60 56 53 50 51 57 44 43 54]
Unique values in 'EstimatedSalary': [ 19000 20000 43000 57000 76000 58000 84000 150000 33000 65000
80000 52000 86000 18000 82000 25000 26000 28000 29000 22000
49000 41000 23000 30000 74000 137000 16000 44000 90000 27000
72000 31000 17000 51000 108000 15000 79000 54000 135000 89000
32000 83000 55000 48000 117000 87000 66000 120000 63000 68000
113000 112000 42000 88000 62000 118000 85000 81000 50000 116000
123000 73000 37000 59000 149000 21000 35000 71000 61000 75000
53000 107000 96000 45000 47000 100000 38000 69000 148000 115000
34000 60000 70000 36000 39000 134000 101000 130000 114000 142000
78000 143000 91000 144000 102000 126000 133000 147000 104000 146000
122000 97000 95000 131000 77000 125000 106000 141000 93000 138000
119000 105000 99000 129000 46000 64000 139000]
Unique values in 'Purchased': [0 1]
```

```
data.isnull().sum()
```

```
User ID      0
Gender       0
Age          0
EstimatedSalary 0
Purchased    0
dtype: int64
```

```
data.duplicated().sum()
```

```
0
```

```
data.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

```
# ---- STEP 2: Select Features and Target ----
X = data[['Age', 'EstimatedSalary']] # Features
y = data['Purchased'] # Target (0 = No, 1 = Yes)
```

```
# ---- STEP 3: Split Dataset ----
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=42
)
```

```
# ---- STEP 4: Feature Scaling ----
# -3 to +3
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
X_train_scaled
```

```
[ -0.109954 , -0.28918908],  
[ 1.794392 ,  0.99609572],  
[ -1.05332971, -0.34761112],  
[ -1.34792161, -1.10709759],  
[ -0.36594861,  0.06134314],  
[ -0.95513241, -1.10709759],  
[ 1.2052082 , -1.4576298 ],  
[ -0.46414591, -0.84419842],  
[ -0.85693511, -0.66893231],  
[ -1.54431621, -1.51605184],  
[ -0.75873781,  1.90163728],  
[ 0.9106163 ,  1.02530673],  
[ 0.812419 ,  0.26582026],  
[ 1.0088136 ,  1.87242626],  
[ 0.9106163 , -0.61051028],  
[ 1.1070109 ,  2.07690339],  
[ -0.56234321,  0.87925164],  
[ 0.0268406 ,  0.03213212],  
[ -1.83890811, -1.28236369],  
[ -0.0713567 ,  0.20739823],  
[ 0.9106163 , -0.55208824],  
[ 0.2232352 , -0.37682213],  
[ -0.169554 ,  1.6095271 ],  
[ -1.74071081,  0.11976517],  
[ -0.66054051, -0.34761112],  
[ 0.3214325 , -0.72735435],  
[ 0.4196298 , -0.46445519],  
[ -0.95513241, -0.96104249],  
[ 0.1250379 ,  0.00055416],  
[ -0.95513241,  0.44108637],  
[ 0.0268406 , -0.55208824],  
[ 0.9106163 , -0.78577639],  
[ -0.0713567 ,  0.06134314],  
[ 1.1070109 , -0.99025351],  
[ 0.7142217 , -1.39920777],  
[ -0.2677513 ,  0.06134314],  
[ -1.34792161, -1.25315268],  
[ -1.15152701, -1.01946453],  
[ 0.5178271 ,  1.84321524],  
[ 0.1250379 ,  0.20739823],  
[ -0.56234321,  0.47029739]])
```

```
# ---- STEP 5: Train Logistic Regression Model ----  
model = LogisticRegression()  
model.fit(X_train_scaled, y_train)
```

▼ LogisticRegression ⓘ ⓘ
► Parameters

```
# ---- STEP 6: Predictions ----  
y_pred = model.predict(X_test_scaled)
```

```
# ---- STEP 7: Evaluation ----  
cm = confusion_matrix(y_test, y_pred)  
print("Confusion Matrix:\n", cm)  
print("\nClassification Report:\n", classification_report(y_test, y_pred))  
accuracy = accuracy_score(y_test, y_pred)  
print(f"\nAccuracy: {accuracy:.2f}")
```

Confusion Matrix:
[[61 2]
 [12 25]]

	precision	recall	f1-score	support
0	0.84	0.97	0.90	63
1	0.93	0.68	0.78	37
accuracy			0.86	100
macro avg	0.88	0.82	0.84	100
weighted avg	0.87	0.86	0.85	100

Accuracy: 0.86

```
import matplotlib.pyplot as plt  
import numpy as np  
  
# Use test set features  
X_set, y_set = X_test_scaled, y_test  
  
# Create mesh grid directly from feature values
```

```

X1, X2 = np.meshgrid(
    np.arange(start=X_set[:, 0].min(), stop=X_set[:, 0].max(), step=0.01),
    np.arange(start=X_set[:, 1].min(), stop=X_set[:, 1].max(), step=0.01)
)

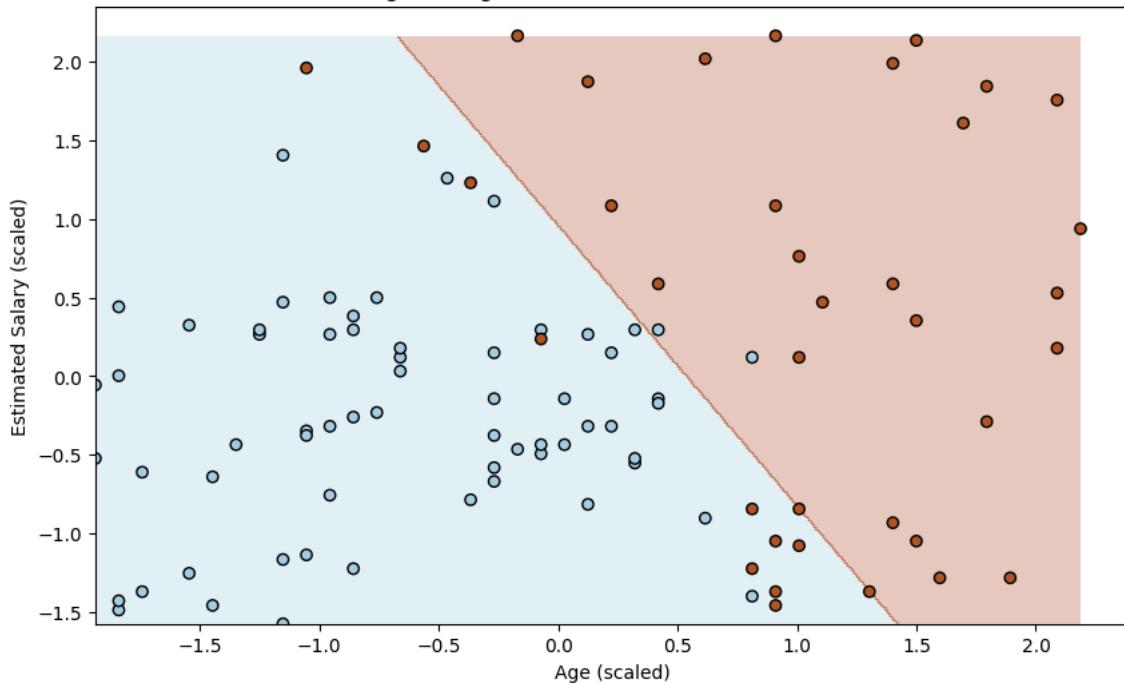
# Predict on the grid
Z = model.predict(np.array([X1.ravel(), X2.ravel()]).T)
Z = Z.reshape(X1.shape)

# Plot decision boundary
plt.figure(figsize=(10, 6))
plt.contourf(X1, X2, Z, alpha=0.3, cmap=plt.cm.Paired)

# Plot test points
plt.scatter(X_set[:, 0], X_set[:, 1], c=y_set, cmap=plt.cm.Paired, edgecolors='k')
plt.title("Logistic Regression - SUV Purchase Prediction")
plt.xlabel("Age (scaled)")
plt.ylabel("Estimated Salary (scaled)")
plt.show()

```

Logistic Regression - SUV Purchase Prediction



▼ Test data

```

test_data = pd.DataFrame({
    "Age": [25, 40, 50],
    "EstimatedSalary": [30000, 70000, 90000]
})

```

```

# Scale and predict
test_scaled = scaler.transform(test_data)
predictions = model.predict(test_scaled)
probs = model.predict_proba(test_scaled)

```

```

print("\n--- Test Data Predictions ---")
for i, row in test_data.iterrows():
    print(f"Age: {row['Age']}, Salary: {row['EstimatedSalary']} -> "
          f"{'{Buys SUV' if predictions[i] else 'No Purchase'}' "
          f"(Prob: {probs[i][1]*100:.1f}%)")

```

```

--- Test Data Predictions ---
Age: 25, Salary: 30000 -> No Purchase (Prob: 0.9%)
Age: 40, Salary: 70000 -> No Purchase (Prob: 35.5%)
Age: 50, Salary: 90000 -> Buys SUV (Prob: 87.2%)

```

```

# Simple plot with legend + probability annotations (on top)
plt.figure(figsize=(8, 5))

```

```
# Decision boundary
plt.contourf(
    X1, X2,
    model.predict(np.c_[X1.ravel(), X2.ravel()]).reshape(X1.shape),
    alpha=0.3, cmap=plt.cm.Paired
)

# Original test set points
plt.scatter(X_set[:, 0], X_set[:, 1], c=y_set, cmap=plt.cm.Paired, alpha=0.5, label="Test set points")

# Test points (new data)
plt.scatter(
    test_scaled[:, 0], test_scaled[:, 1], c=predictions,
    cmap=plt.cm.coolwarm, marker='X', s=200, edgecolors='black', label="New test data"
)

plt.title("Logistic Regression - Test Data with Probabilities")
plt.xlabel("Age (scaled)")
plt.ylabel("Salary (scaled)")
plt.legend()
plt.show()
```

