



```
In [1]: import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_r
```

```
In [2]: df = pd.read_csv("C:\\Users\\DELL\\OneDrive\\Desktop\\Varsha\\Admission_Predic
```

```
In [3]: df.columns
```

```
Out[3]: Index(['Serial No.', 'GRE Score', 'TOEFL Score', 'University Rating', 'SOP',
              'LOR ', 'CGPA', 'Research', 'Chance of Admit '],
              dtype='object')
```

```
In [ ]: Create Target Column (Binary)
```

```
In [4]: df['Admit_Class'] = (df['Chance of Admit '] >= 0.5).astype(int)
```

Drop Unneeded Columns

```
In [5]: df = df.drop(['Serial No.', 'Chance of Admit '], axis=1)
```

Convert Yes/No If Present

```
In [6]: df = df.replace({'Yes': 1, 'No': 0})
```

C:\Users\DELL\AppData\Local\Temp\ipykernel_912\3534361578.py:1: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a future version. To retain the old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to the future behavior, set `pd.set_option('future.no_silent_downcasting', True)`

```
df = df.replace({'Yes': 1, 'No': 0})
```

Convert Text Columns to Numeric Automatically

```
In [7]: from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

for col in df.columns:
    if df[col].dtype == 'object':
        df[col] = le.fit_transform(df[col])
```

Split Features (X) and Target (y)

```
In [8]: X = df.drop('Admit_Class', axis=1)
y = df['Admit_Class']
```

Train/Test Split

```
In [9]: from sklearn.model_selection import train_test_split

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

Scaling the Data (Very Important for Logistic Regression)

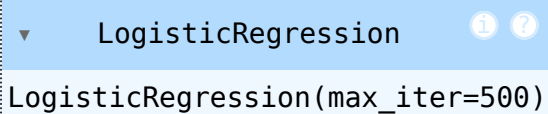
```
In [10]: from sklearn.preprocessing import StandardScaler

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

Train the Logistic Regression Model

```
In [11]: from sklearn.linear_model import LogisticRegression

model = LogisticRegression(max_iter=500)
model.fit(X_train, y_train)
```

```
Out[11]: 
LogisticRegression(max_iter=500)
```

Make Predictions

```
In [12]: y_pred = model.predict(X_test)
```

Check Accuracy, Confusion Matrix & Report

```
In [13]: print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

Accuracy: 0.9375

Confusion Matrix:

```
[[ 5  5]
 [ 0 70]]
```

Classification Report:

	precision	recall	f1-score	support
0	1.00	0.50	0.67	10
1	0.93	1.00	0.97	70
accuracy			0.94	80
macro avg	0.97	0.75	0.82	80
weighted avg	0.94	0.94	0.93	80

In []: