Logistic regression is a supervised learning algorithm used to predict a dependent categorical target variable. In essence, if you have a large set of data that you want to categorize, logistic regression may be able to help. • For example, if you were given a dog and an orange and you wanted to find out whether each of these items was an animal or not, the desired result would be for the dog to end up classified as an animal, and for the orange to be categorized as not an animal. • Animal is your target; it is dependent on your data in order to be able to classify the item correctly. In this example, there are only two possible answers (binary logistic regression), animal or not an animal. However, it is also possible to set up your logistic regression with more than two possible categories (multinomial logistic regression).

```
import pandas as pd
import random

# Create a DataFrame with the initial data
data = {
    'User ID': [1, 2, 3, 4, 5, 395, 396, 397, 398, 399],
    'Gender': ['Male', 'Male', 'Female', 'Female', 'Male', 'Female', 'Female', 'Male', 'Female', 'Male'],
    'Age': [19, 35, 26, 27, 19, 46, 50, 36, 49, 51],
    'EstimatedSalary': [15624510, 15810944, 15603246, 15804002, 15668575, 15691863, 15706071, 15654296, 15755018, 15594041],
    'Purchased': [0, 0, 0, 0, 0, 1, 1, 1, 0, 1]
}

df = pd.DataFrame(data)

# Generate additional 390 rows
for i in range(390):
    gender = random.choice(['Male', 'Female'])
    age = random.randint(18, 60)
    estimated_salary = random.randint(10000, 100000) * 1000
    purchased = random.choice([0, 1])

    df.loc[len(df)] = [len(df), gender, age, estimated_salary, purchased]

# Save the DataFrame to a CSV file
df.to_csv('Ads_dat.csv', index=False)
```

```
df= pd.read_csv('/content/Ads_dat.csv')
df
```

	User ID	Gender	Age	EstimatedSalary	Purchased
0	1	Male	19	15624510	0
1	2	Male	35	15810944	0
2	3	Female	26	15603246	0
3	4	Female	27	15804002	0
4	5	Male	19	15668575	0
...	...	...	...	...	...
395	395	Female	27	38376000	0
396	396	Female	50	76002000	1
397	397	Female	30	37288000	1
398	398	Female	39	55253000	1
399	399	Male	56	47878000	1

400 rows × 5 columns

Next steps: View recommended plots

```
 #input data
x=df[['Age','EstimatedSalary']]

#output data
y=df['Purchased']
```

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
x_scaled = scaler.fit_transform(x)

#cross. validation

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_scaled,y,random_state=0,test_size=0.2)
```

```
x_train
```

```
       [0.14285714, 0.13627908],
       [0.02380952, 0.57019156],
       [0.76190476, 0.21811191],
       [0.4047619 , 0.23857576],
       [0.16666667, 0.6832219 ],
       [0.5952381 , 0.81839604],
       [0.5       , 0.93240729],
       [0.57142857, 0.8711285 ],
       [0.04761905, 0.90861737],
       [0.26190476, 0.07697338],
       [0.5       , 0.91198854],
       [0.95238095, 0.83662747],
       [0.33333333, 0.41886056],
       [0.45238095, 0.64341042],
       [0.14285714, 0.35937447],
       [0.61904762, 0.40083208],
       [0.66666667, 0.74271927],
       [0.54761905, 0.30806264],
       [0.19047619, 0.99673029],
       [0.73809524, 0.43337129],
       [1.        , 0.69525216],
       [0.26190476, 0.45726269],
       [1.        , 0.71739596],
       [0.71428571, 0.30521011],
       [0.38095238, 0.22087425],
       [0.61904762, 0.93276809],
       [0.33333333, 0.80425738],
       [0.78571429, 0.09385183],
       [0.47619048, 0.50741321],
       [0.        , 0.25147419],
       [0.21428571, 0.16711578],
       [0.16666667, 0.56908662],
       [0.19047619, 0.58511946],
       [0.14285714, 0.18671147],
       [0.95238095, 0.25114722],
       [0.85714286, 0.56314478],
       [0.85714286, 0.42126211],
       [0.61904762, 0.76069137],
       [0.11904762, 0.65236264],
       [0.11904762, 0.28269424],
       [0.07142857, 0.32273122],
       [0.78571429, 0.05116572],
       [0.78571429, 0.41774435],
       [0.61904762, 0.74074617],
       [0.35714286, 0.20315019],
       [0.73809524, 0.49893453],
       [0.73809524, 0.75672263],
       [0.69047619, 0.21474073],
       [0.73809524, 0.15073343],
       [0.57142857, 0.33555072]])
```

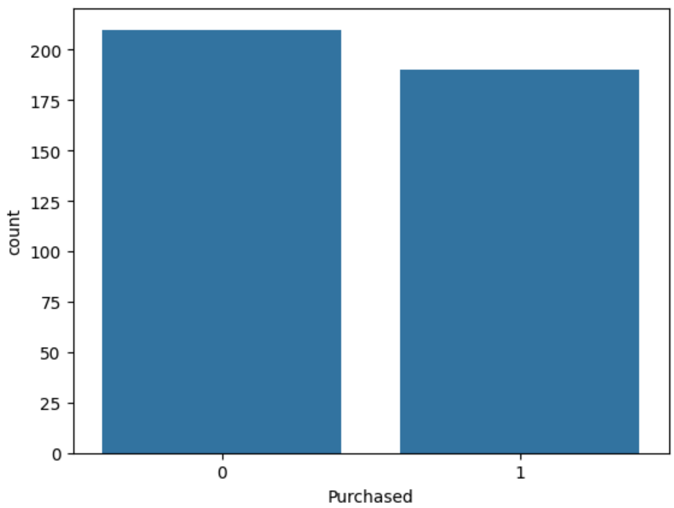
```
y_train
```

```
336    0
64     0
55     0
106    0
300    0
      ..
323    0
192    0
117    0
47     0
172    1
Name: Purchased, Length: 320, dtype: int64
```

```
from sklearn.linear_model import LogisticRegression
```

```
import seaborn as sns
sns.countplot(x=y)
```

```
<Axes: xlabel='Purchased', ylabel='count'>
```



```
y.value_counts()
```

```
0    210
1    190
Name: Purchased, dtype: int64
```

```
 #creat the object
classifier = LogisticRegression()
```

```
classifier.fit(x_train,y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
 #predication
y_pred = classifier.predict(x_test)
y_train.shape
```

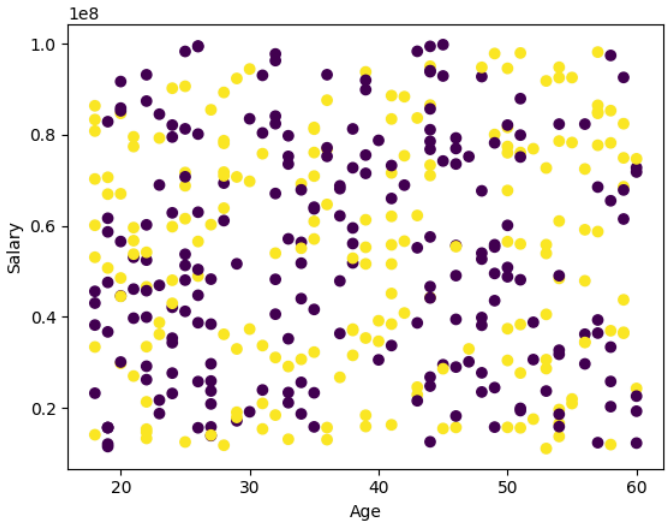
```
(320,)
```

```
x_train.shape
```

```
(320, 2)
```

```
import matplotlib.pyplot as plt
plt.xlabel('Age')
plt.ylabel('Salary')
plt.scatter(x['Age'],x['EstimatedSalary'],c=y)
```

```
<matplotlib.collections.PathCollection at 0x7a4403cebb50>
```



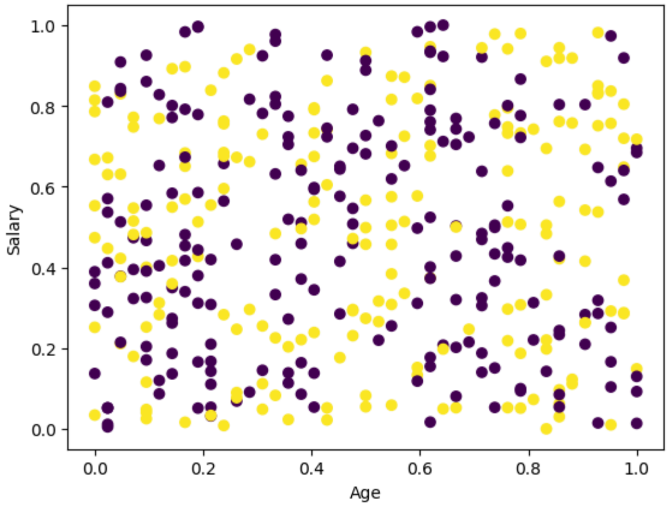

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
x_scaled = scaler.fit_transform(x)
```

```
pd.DataFrame(x_scaled).describe()
```

	0	1
count	400.000000	400.000000
mean	0.475417	0.477639
std	0.302803	0.286456
min	0.000000	0.000000
25%	0.190476	0.224206
50%	0.476190	0.481115
75%	0.738095	0.730557
max	1.000000	1.000000

```
plt.xlabel('Age')
plt.ylabel('Salary')
plt.scatter(x_scaled[:,0],x_scaled[:,1],c=y)
```

```
<matplotlib.collections.PathCollection at 0x7a4403d3f730>
```




```
y_test.value_counts()
```

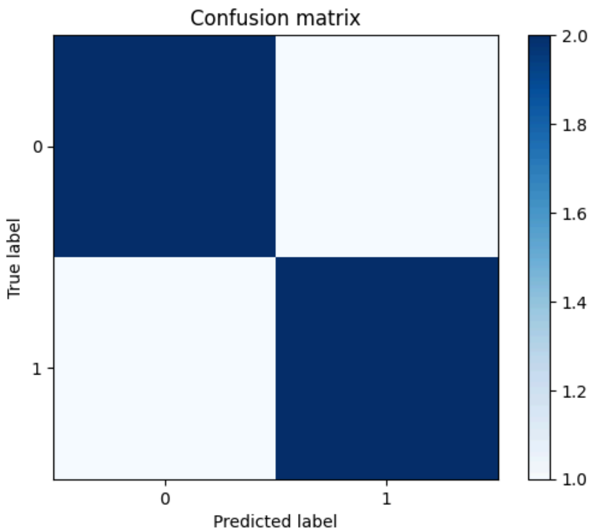
```
0    40
1    40
Name: Purchased, dtype: int64
```

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

```
# Assuming y_true and y_pred are your true and predicted labels
y_true = [0, 1, 0, 1, 0, 1]
y_pred = [0, 1, 1, 1, 0, 0]

# Compute confusion matrix
conf_matrix = confusion_matrix(y_true, y_pred)

# Plot confusion matrix
plt.imshow(conf_matrix, interpolation='nearest', cmap=plt.cm.Blues)
plt.title('Confusion matrix')
plt.colorbar()
tick_marks = np.arange(len(conf_matrix))
plt.xticks(tick_marks, tick_marks)
plt.yticks(tick_marks, tick_marks)
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()
```




```
from sklearn.metrics import accuracy_score, classification_report

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

# Classification report
print(classification_report(y_test, y_pred))

# New data for prediction
new1 = [[26, 34000]]
new2 = [[57, 138000]]
```

```
# Predicting on new data
prediction_new1 = classifier.predict(scaler.transform(new1))
prediction_new2 = classifier.predict(scaler.transform(new2))

print("Prediction for new data point 1:", prediction_new1)
print("Prediction for new data point 2:", prediction_new2)
```

```
Accuracy: 0.5625
              precision    recall  f1-score   support

           0       0.53      0.97      0.69        40
           1       0.86      0.15      0.26        40

    accuracy                           0.56        80
   macro avg       0.70      0.56      0.47        80
weighted avg       0.70      0.56      0.47        80

Prediction for new data point 1: [0]
Prediction for new data point 2: [0]
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but MinMaxScaler was fitted with fea
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but MinMaxScaler was fitted with fea
  warnings.warn(
```

```
# Get feature importance
feature_importance = classifier.coef_[0]
print("Feature Importance:")
for i, feature in enumerate(x.columns):
    print(feature, ":", feature_importance[i])
```

```
Feature Importance:
Age : 0.2343040777752449
EstimatedSalary : 0.07722544174946042
```