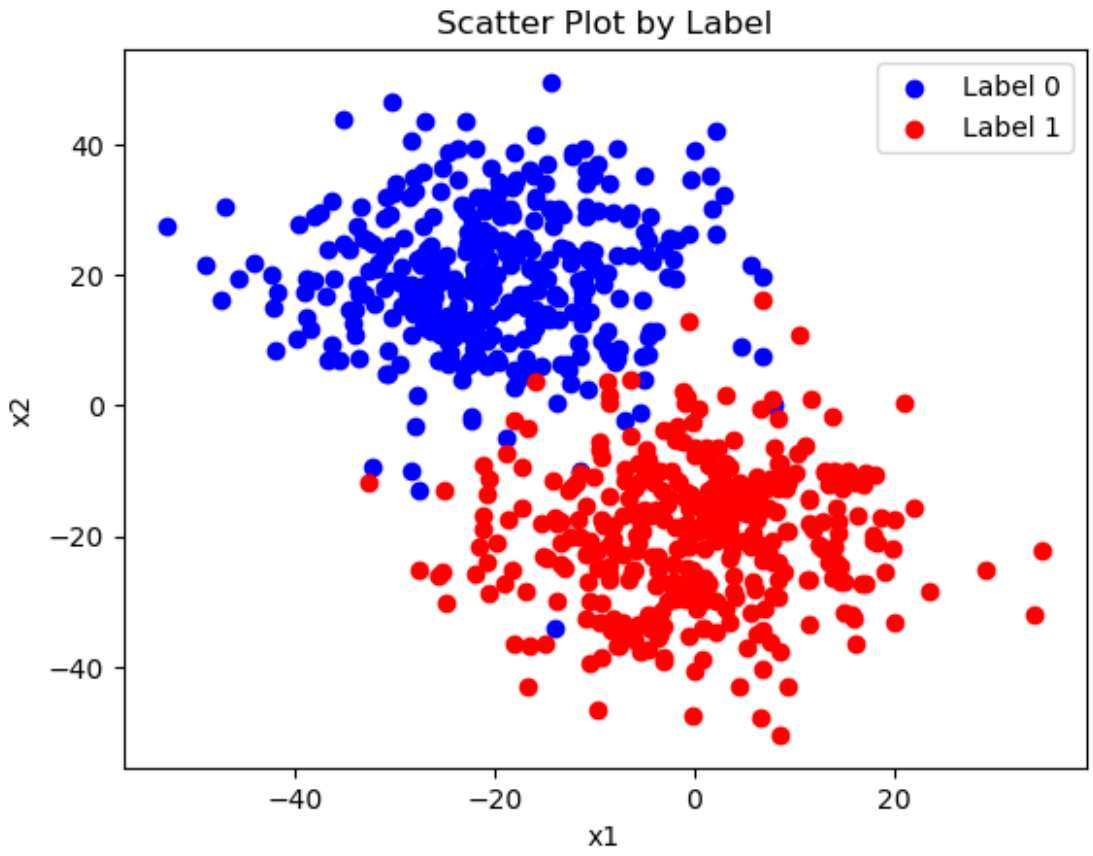
```
import numpy as np
import pandas as pd
import sklearn.linear_model as lm
import matplotlib.pyplot as plt
df = pd.read_csv("Logistic-Regression-data-2-class-v0.csv")
print(df.head())
```

```
          x1         x2  yclass
0 -12.304702   3.499240       0
1 -21.302900  17.983794       0
2  -6.320254  29.639092       0
3   2.259775  26.227155       0
4 -14.777150  19.536615       0
```

```
df_0=df[df['yclass']==0]
df_1=df[df['yclass']==1]
plt.scatter(df_0['x1'], df_0['x2'], label='Label 0', color='blue')
plt.scatter(df_1['x1'], df_1['x2'], label='Label 1', color='red')
plt.title('Scatter Plot by Label')
plt.xlabel('x1')
plt.ylabel('x2')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x210288ef700>
```



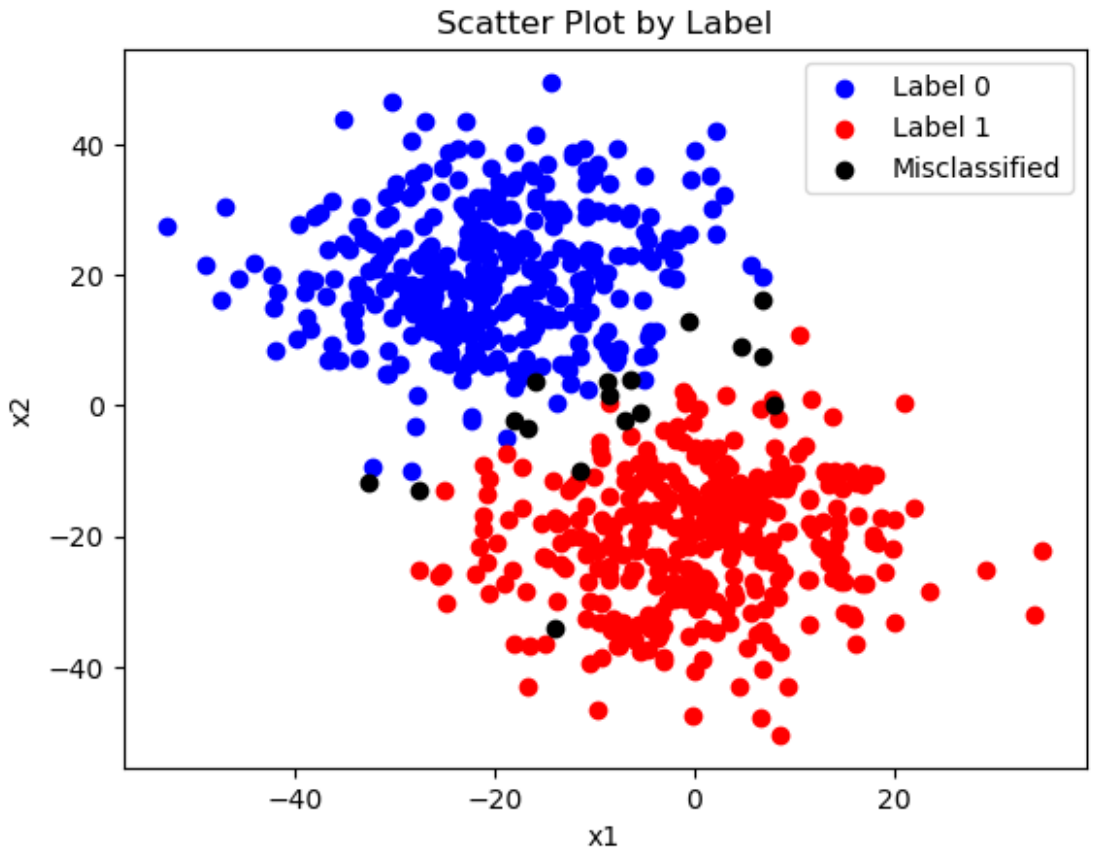
1. We can see that the two types of data (0 and 1) are clustered in different regions.

```
logist_regr = lm.LogisticRegression()
logist_regr.fit(df[['x1','x2']], df['yclass'])
```

```
LogisticRegression()
```

```
pred = logist_regr.predict(df[['x1','x2']])
wrong=np.where(pred!=df['yclass'])
wdf=df.iloc[wrong]
cdf=df.iloc[np.where(pred==df['yclass'])]
cdf_0=cdf[cdf['yclass']==0]
cdf_1=cdf[cdf['yclass']==1]
plt.scatter(cdf_0['x1'], cdf_0['x2'], label='Label 0', color='blue')
plt.scatter(cdf_1['x1'], cdf_1['x2'], label='Label 1', color='red')
plt.scatter(wdf['x1'], wdf['x2'], label='Misclassified',
color='black')
plt.title('Scatter Plot by Label')
plt.xlabel('x1')
plt.ylabel('x2')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x210289b2080>
```



1. The model seems to be of sufficiently good quality since it misclassifies only a small amount of data occuring at the boundaries of the two clusters.

```
tn=0
tp=0
fn=0
fp=0
for i in range(0,720):
    if df['yclass'][i]==1 and pred[i]==1:
        tp+=1
    elif df['yclass'][i]==0 and pred[i]==1:
        fp+=1
    elif df['yclass'][i]==0 and pred[i]==0:
        tn+=1
    else:
        fn+=1
con=[[tn,fp],[fn, tp]]
print("Confusion Matrix:")
print(con)
```

```
Confusion Matrix:
[[352, 8], [9, 351]]
```

```
prec=(tp/(tp+fp))
rec=(tp/(tp+fn))
f1=2*prec*rec/(prec+rec)
print(f"Precision:{prec}")
print(f"Recall:{rec}")
print(f"f1:{f1}")
print(f"TPR:{rec}")
FPR=fp/(tn+fp)
print(f"FPR:{FPR}")
```

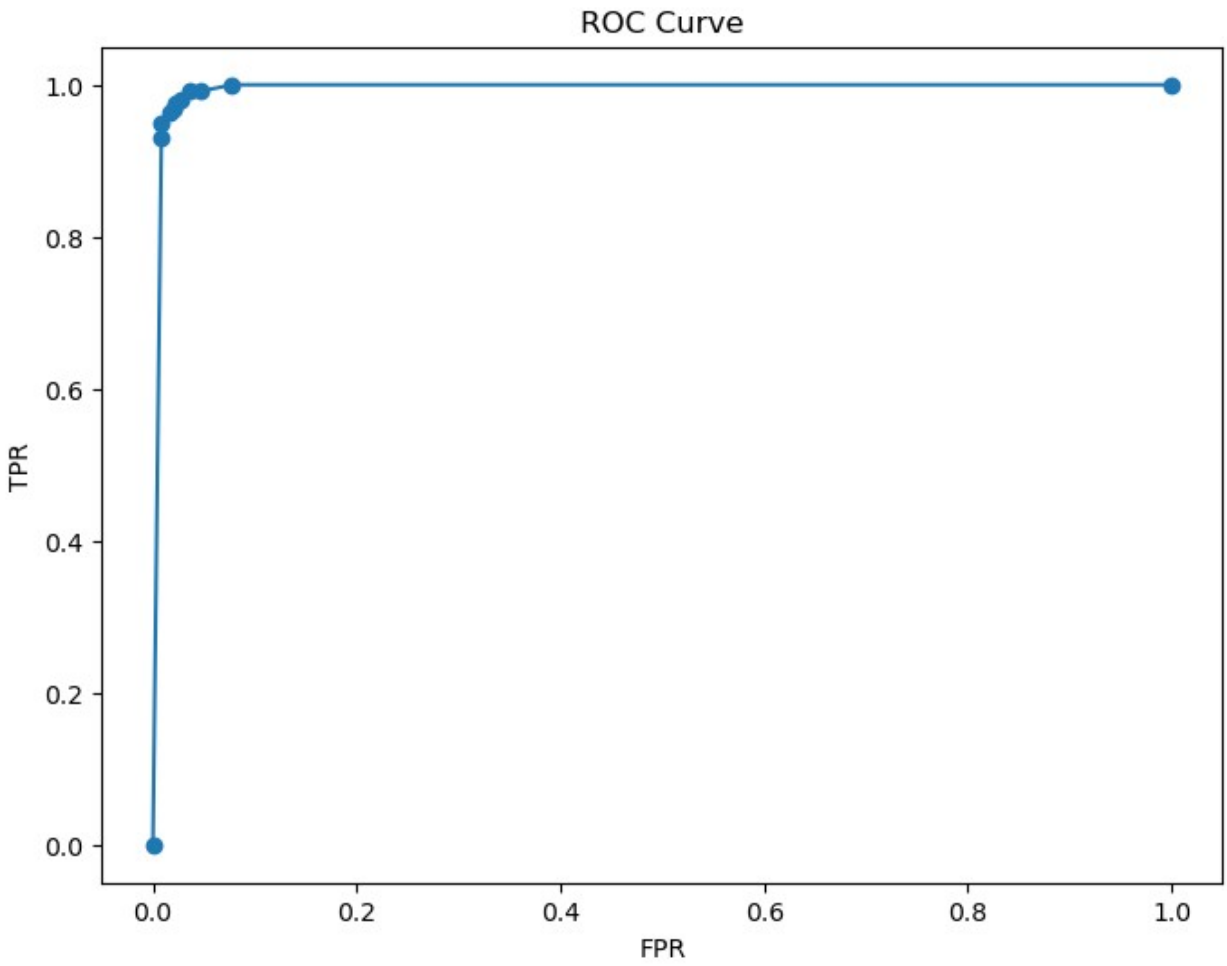
```
Precision:0.9777158774373259
Recall:0.975
f1:0.9763560500695411
TPR:0.975
FPR:0.022222222222222223
```

```
prob = logist_regr.predict_proba(df[['x1','x2']])
tprd=[0]*11
fprd=[0]*11
i=0
for threshold in np.arange(0, 1.1, 0.1):
    pred=(np.array(prob[:, 1])>=threshold).astype(int)
    tp=sum((df['yclass']==1) & (pred==1))
    fp=sum((df['yclass']==0) & (pred==1))
    tn=sum((df['yclass']==0) & (pred==0))
    fn=sum((df['yclass']==1) & (pred==0))
    tpr=tp/(tp+fn)
    fpr=fp/(fp+tn)
    tprd[i]=tpr
    fprd[i]=fpr
    i+=1
    print(f"Threshold = {threshold}")
    print("Confusion Matrix:")
    print(f"   {tn}   {fp}")
    print(f"   {fn}   {tp}")
    print(f"TPR = {tpr}")
    print(f"FPR = {fpr}")
    print()
```

```
Threshold = 0.0
Confusion Matrix:
   0   360
   0   360
TPR = 1.0
FPR = 1.0

Threshold = 0.1
Confusion Matrix:
   332   28
   0   360
```

```
TPR = 1.0
FPR = 0.07777777777777778

Threshold = 0.2
Confusion Matrix:
   343   17
   3   357
TPR = 0.9916666666666667
FPR = 0.04722222222222222

Threshold = 0.30000000000000004
Confusion Matrix:
   347   13
   3   357
TPR = 0.9916666666666667
FPR = 0.03611111111111111

Threshold = 0.4
Confusion Matrix:
   350   10
   7   353
TPR = 0.9805555555555555
FPR = 0.027777777777777776

Threshold = 0.5
Confusion Matrix:
   352   8
   9   351
TPR = 0.975
FPR = 0.022222222222222223

Threshold = 0.6000000000000001
Confusion Matrix:
   353   7
   11   349
TPR = 0.9694444444444444
FPR = 0.019444444444444445

Threshold = 0.7000000000000001
Confusion Matrix:
   354   6
   13   347
TPR = 0.9638888888888889
FPR = 0.016666666666666666

Threshold = 0.8
Confusion Matrix:
   357   3
   18   342
TPR = 0.95
```

```
FPR = 0.008333333333333333

Threshold = 0.9
Confusion Matrix:
   357   3
   25   335
TPR = 0.9305555555555556
FPR = 0.008333333333333333

Threshold = 1.0
Confusion Matrix:
   360   0
   360   0
TPR = 0.0
FPR = 0.0


plt.figure(figsize=(8,6))
plt.plot(fprd, tprd, marker='o', linestyle='-')
plt.title('ROC Curve')
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.show()
```



1. The ROC follows what we expect from a good logistic regression model, rising very steeply and finally fininshing at (1,1).

```
from sklearn.metrics import roc_auc_score
auc = roc_auc_score(df['yclass'], prob[:, 1] )  # Replace y_true and
y_scores with your actual data

# Print the AUC value
print(f"AUC: {auc}")
```

```
AUC: 0.9967361111111112
```

The value is very close to one, indicating it's a good regression model.