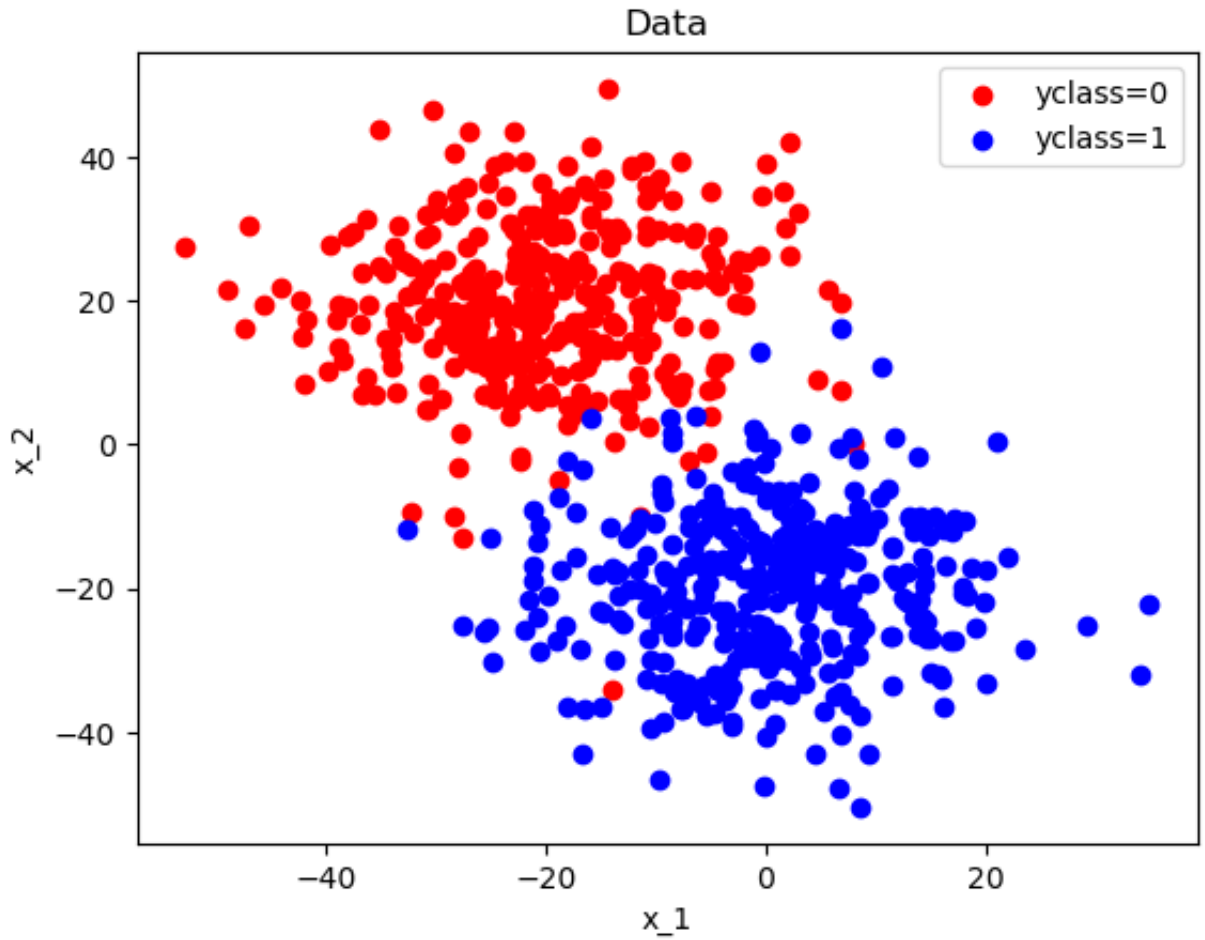
In [14]:
```
import numpy as np
import pandas as pd
import sklearn.linear_model as lm
from sklearn.metrics import confusion_matrix, precision_score, recall_sco
import matplotlib.pyplot as plt
```

In [15]:
```
df = pd.read_csv("Sixth_assignment.csv")
```

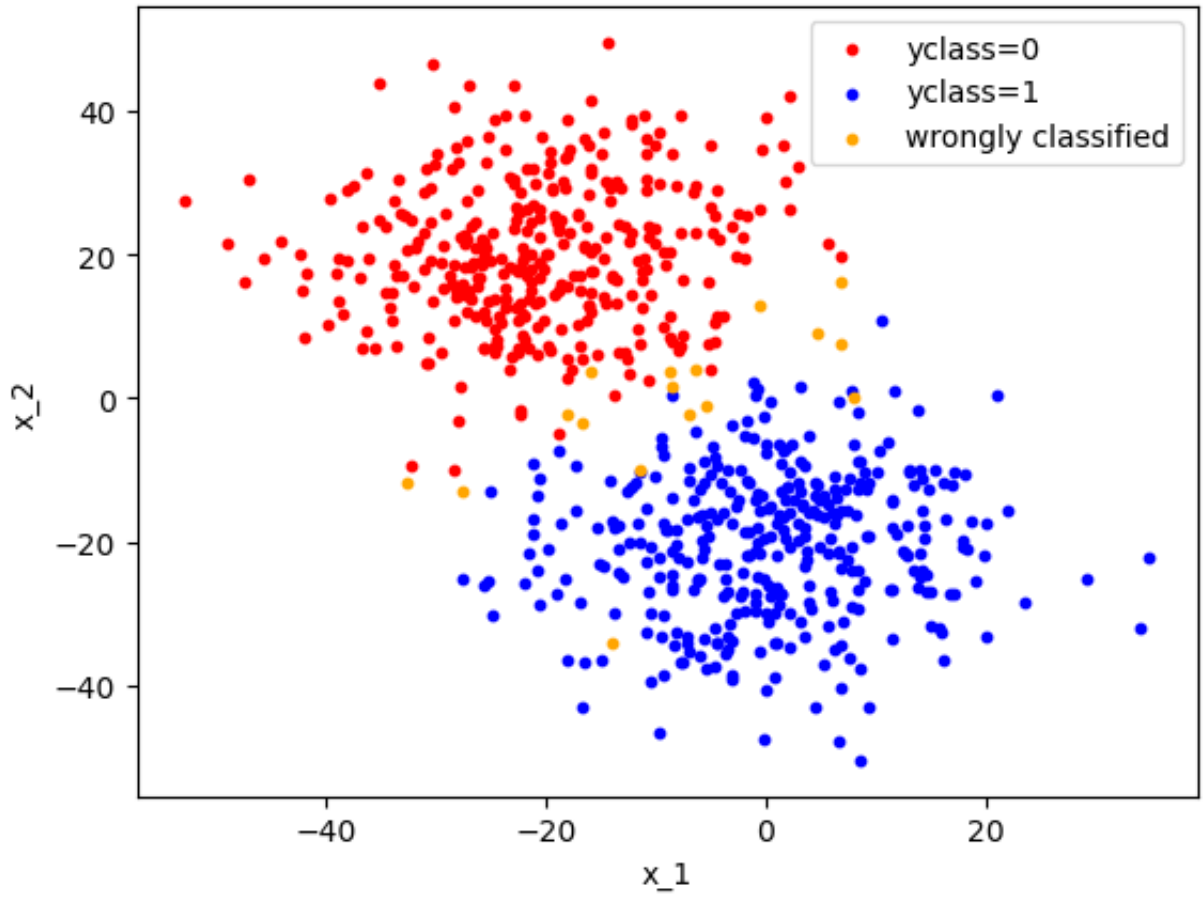
In [26]:
```
plt.scatter(df[df['yclass'] == 0]['x1'], df[df['yclass'] == 0]['x2'], col
plt.scatter(df[df['yclass'] == 1]['x1'], df[df['yclass'] == 1]['x2'], col
plt.xlabel('x_1')
plt.ylabel('x_2')
plt.title('Data')
plt.legend()
plt.show()
```



It is observed that there exist points that lie between 0 and 1 that makes it tough to classify them to any of them.

In [17]:
```
lr = lm.LogisticRegression()
lr.fit(df[['x1','x2']], df['yclass'])
pred = lr.predict(df[['x1','x2']])
df['pred_yclass']=pred
```

In [27]:
```
plt.scatter(df[(df['yclass'] == 0) & (df['pred_yclass'] == 0)]['x1'], df[
plt.scatter(df[(df['yclass'] == 1) & (df['pred_yclass'] == 1)]['x1'], df[
plt.scatter(df[df['yclass'] != df['pred_yclass']]['x1'], df[df['yclass']
plt.xlabel('x_1')
plt.ylabel('x_2')
plt.legend()
plt.show()
```



In [19]:
```
lr.score(df[['x1','x2']], df['yclass'])
```

Out[19]:
```
0.9763888888888889
```

The regression can classify 97.6% of the data correctly .

In [20]:
```
conf_matrix = confusion_matrix(df['yclass'], pred)
print("Confusion Matrix:")
print(conf_matrix)
```

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

In [21]:
```
precision = precision_score(df['yclass'], pred)
recall = recall_score(df['yclass'], pred)
f1 = f1_score(df['yclass'], pred)
fpr, tpr, thresholds = roc_curve(df['yclass'], pred)
print("Precision:", precision)
print("Recall:", recall)
print("F1-Score:", f1)
print("TPR:", tpr[1])
print("FPR:", fpr[1])
```

```
Precision: 0.9777158774373259
Recall: 0.975
F1-Score: 0.9763560500695411
TPR: 0.975
FPR: 0.022222222222222223
```

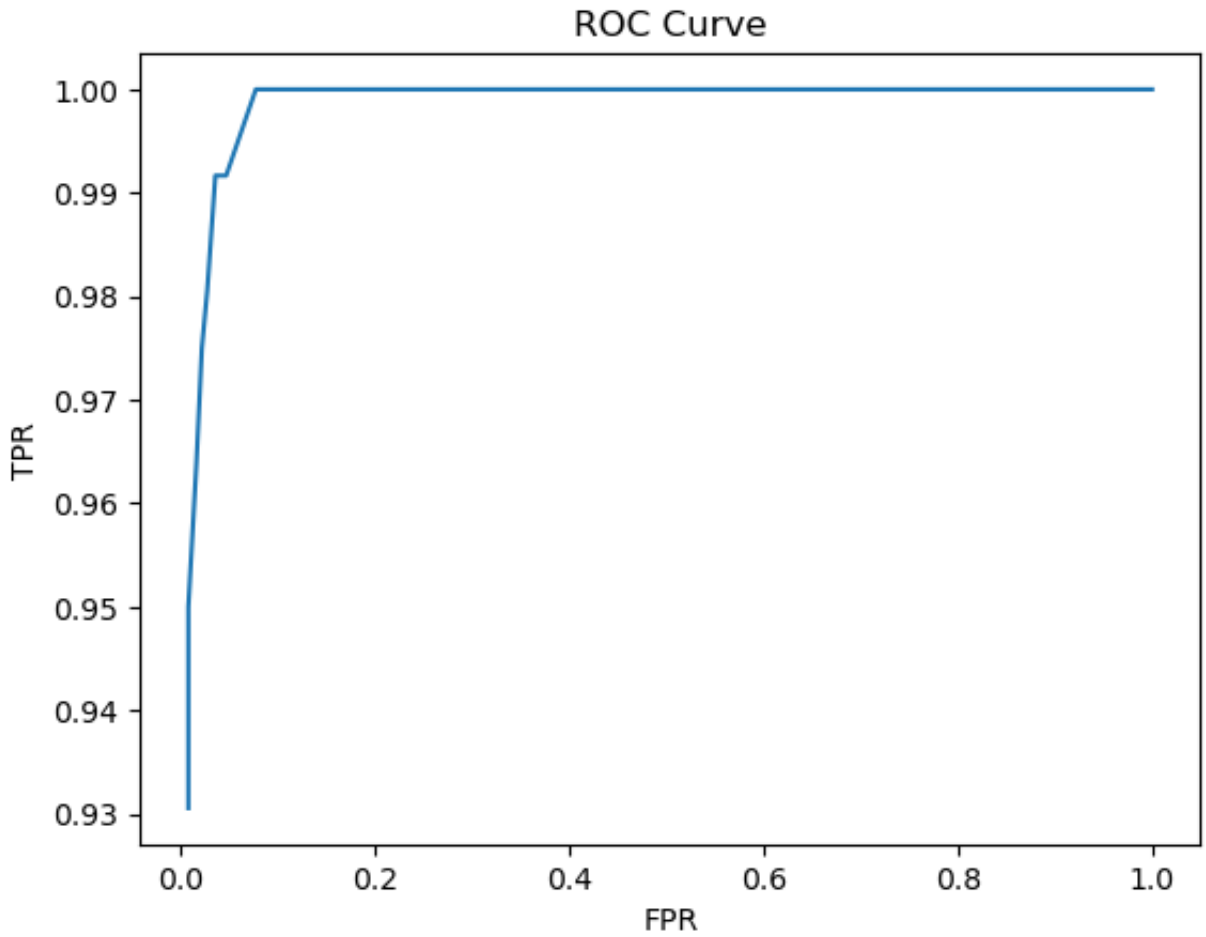
The value of Precision, Recall, F1-Score, TPR are all close to one and hence the prediction made by the model is quite satisfactory. Also the value of FPR is low as expected from a good model.

In [22]:
```
prob = lr.predict_proba(df[['x1','x2']])
```

In [23]:
```
threshold_prob=np.arange(0,1,0.1)
prob_yclass1=prob[:,1]
TPR=[]
FPR=[]
for i in threshold_prob:
    pred_new=(prob_yclass1>=i).astype(int)
    conf_matrix = confusion_matrix(df['yclass'], pred_new)
    fpr, tpr, thresholds = roc_curve(df['yclass'], pred_new)
    print('threshold=',i)
    print("Confusion Matrix:")
    print(conf_matrix,'\n')
    TPR.append(tpr[1])
    FPR.append(fpr[1])
```

```
threshold= 0.0
Confusion Matrix:
[[  0 360]
 [  0 360]]

threshold= 0.1
Confusion Matrix:
[[332  28]
 [  0 360]]

threshold= 0.2
Confusion Matrix:
[[343  17]
 [  3 357]]

threshold= 0.30000000000000004
Confusion Matrix:
[[347  13]
 [  3 357]]

threshold= 0.4
Confusion Matrix:
[[350  10]
 [  7 353]]

threshold= 0.5
Confusion Matrix:
[[352   8]
 [  9 351]]

threshold= 0.6000000000000001
Confusion Matrix:
[[353   7]
 [ 11 349]]

threshold= 0.7000000000000001
Confusion Matrix:
[[354   6]
 [ 13 347]]

threshold= 0.8
Confusion Matrix:
[[357   3]
 [ 18 342]]

threshold= 0.9
Confusion Matrix:
[[357   3]
 [ 25 335]]
```

In [24]:
```
plt.plot(FPR,TPR)
plt.title('ROC Curve')
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.show()
```



Here, we can see that the value of TPR is close to one while the value of FPR is close to zero. Hnece our model has high performace and correctly classifies the data into two classes.

In [25]:
```
auc = roc_auc_score(df['yclass'], pred)
print(auc)
```

```
0.976388888888889
```

AUC=0.97 means that it is almost a perfect model.