Classify Meister Midterm Evaluation

GROUP 19

- Chetan
- Mohit Dhakad
- Harsukh Singh Sagri
- Nitin Babu

Problem Statement

- To predict death and survival in the Titanic ship sinking, developing a model that can accurately classify passengers as either survivors or non-survivors based on various features and data available about the passengers .
- The Titanic dataset contains information about passengers such as their age, gender, class, fare, and whether they survived or not . We aim to understand the factors that influenced survival and create a model that can predict survival outcomes for new or unseen data .
- Analyzing the model's results to understand which factors had the most significant influence on survival predictions and gaining insights into the dynamics of the Titanic disaster . Overall, the main goal of this ML project would be to create a reliable predictive model that can accurately classify passengers into survivors and non-survivors based on historical data, contributing to a better understanding of the factors that influenced survival rates during the Titanic sinking.

OBJECTIVES

- The main goal of this ML project would be to create a reliable predictive model that can accurately classify passengers into survivors and non-survivors based on historical data, contributing to a better understanding of the factors that influenced survival rates during the Titanic sinking.

LOGISTIC REGRESSION

- Logistic Regression is a machine learning classification algorithm that is used to predict the probability of a categorical dependent variable.
- In Logistic Regression, the dependent variable is a binary variable that contains data coded as 1 (yes , success etc.) or 0 (no , failure etc.).
- It is based on the concept of the logistic function (sigmoid function) , which maps any real valued number to a value between between 0 and 1. This function is used to estimate the probability of the binary outcome.

Dataset

We have 11 features

- Survival: 0 = No, 1 = Yes
- Pclass: Passenger Class
- Ticket Class 1 = 1st, 2 = 2nd, 3 = 3rd
- Sex: Gender
- Age: Age in Years
- SibSp: Number of Siblings / spouses abroad the Titanic
- Parch: Number of Parents / children
a broad the Titanic
- Ticket: Ticket Number
- Fare: Passenger Fare
- Cabin: Cabin Number
- Embarked: Port of Embarkation C = Cherboug, Q = Queenstown, S = Southampton

Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	3	Braund, Mr	male	22	1	0	A/5 21171	7.25		S
1	1	Cumings, M	female	38	1	0	PC 17599	71.2833	C85	C
1	3	Heikkinen, I	female	26	0	0	STON/O2. 3	7.925		S
1	1	Futrelle, Mr	female	35	1	0	113803	53.1	C123	S
0	3	Allen, Mr. V	male	35	0	0	373450	8.05		S
0	3	Moran, Mr.	male		0	0	330877	8.4583		Q
0	1	McCarthy, I	male	54	0	0	17463	51.8625	E46	S
0	3	Palsson, Ma	male	2	3	1	349909	21.075		S

Titanic Disaster Survival Using Logistic Regression

Importing Libraries

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

✓ 0.4s

Load the Data

```
titanic_data = pd.read_csv("train.csv")
```

✓ 0.1s

DESCRIBING THE DATA

```
titanic_data.describe()
```

✓ 0.0s

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

```
titanic_data.columns
```

✓ 0.0s

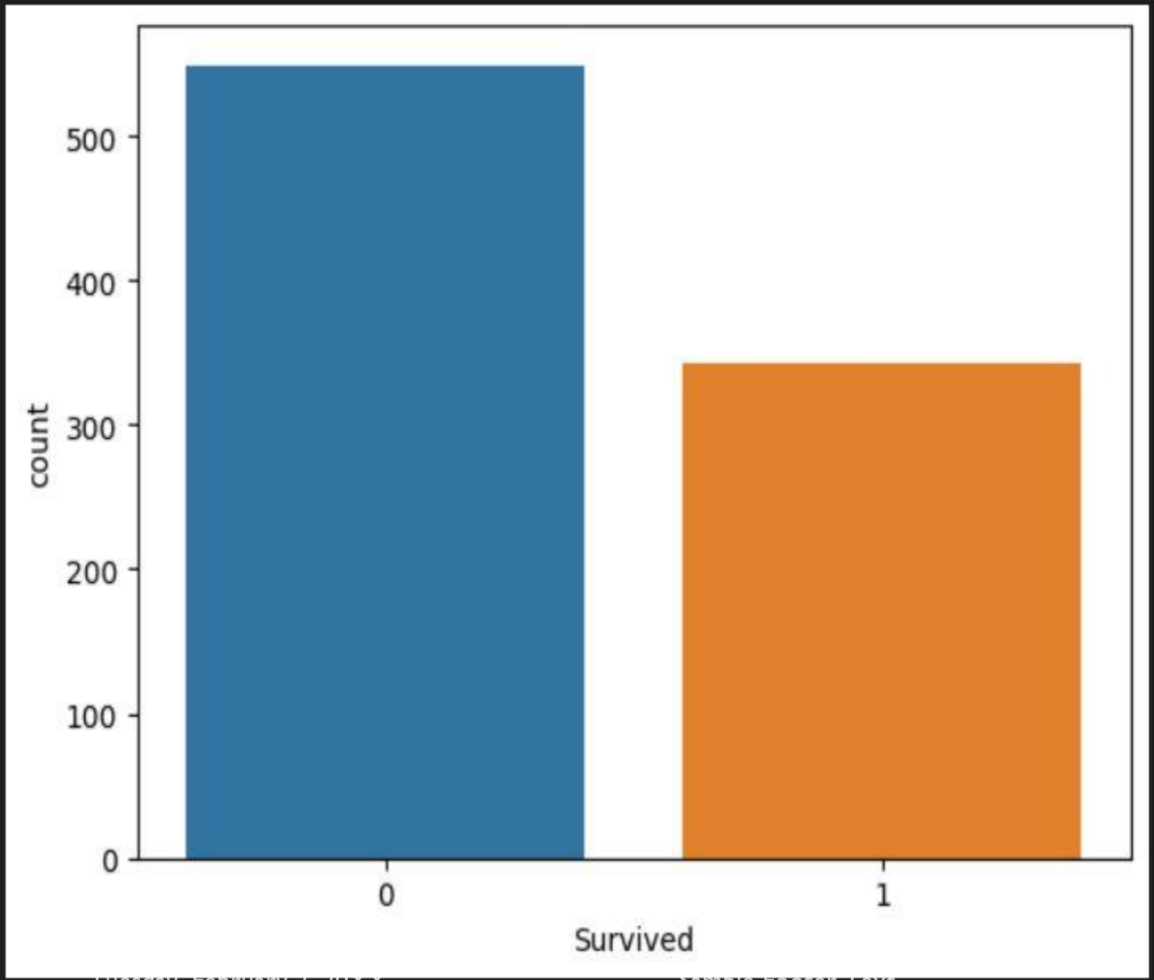
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

Countplot of survived vs not survived

```
sns.countplot(x = "Survived", data = titanic_data)
```

✓ 0.2s

<Axes: xlabel='Survived', ylabel='count'>

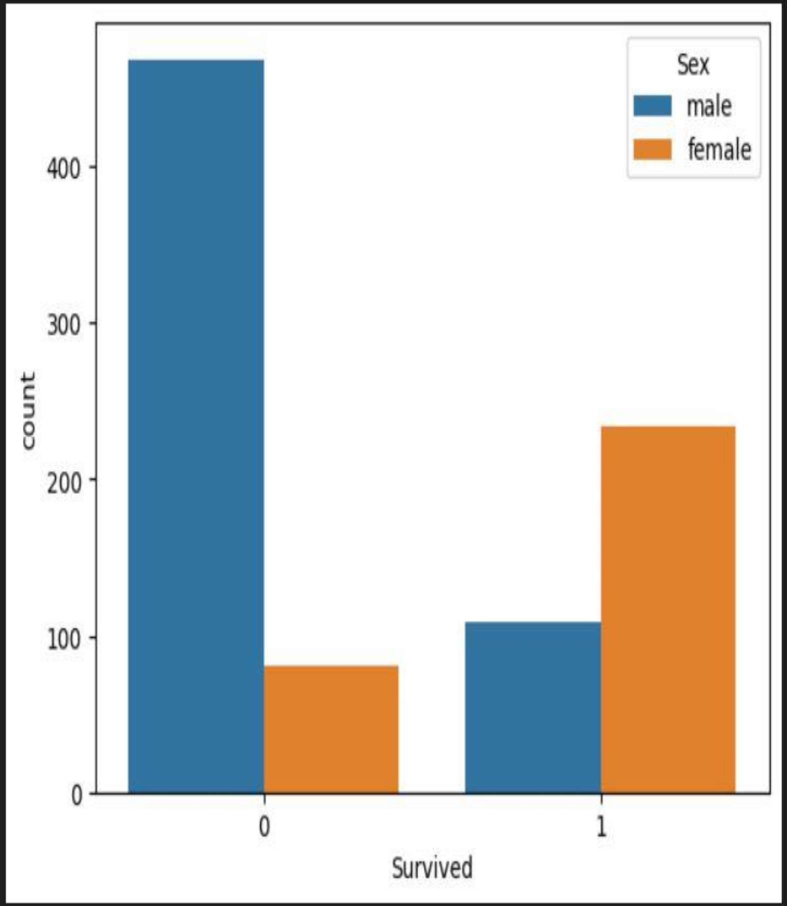


Male vs Female Survival

```
sns.countplot(x = "Survived", data = titanic_data, hue = "Sex")
```

✓ 0.2s

<Axes: xlabel='Survived', ylabel='count'>

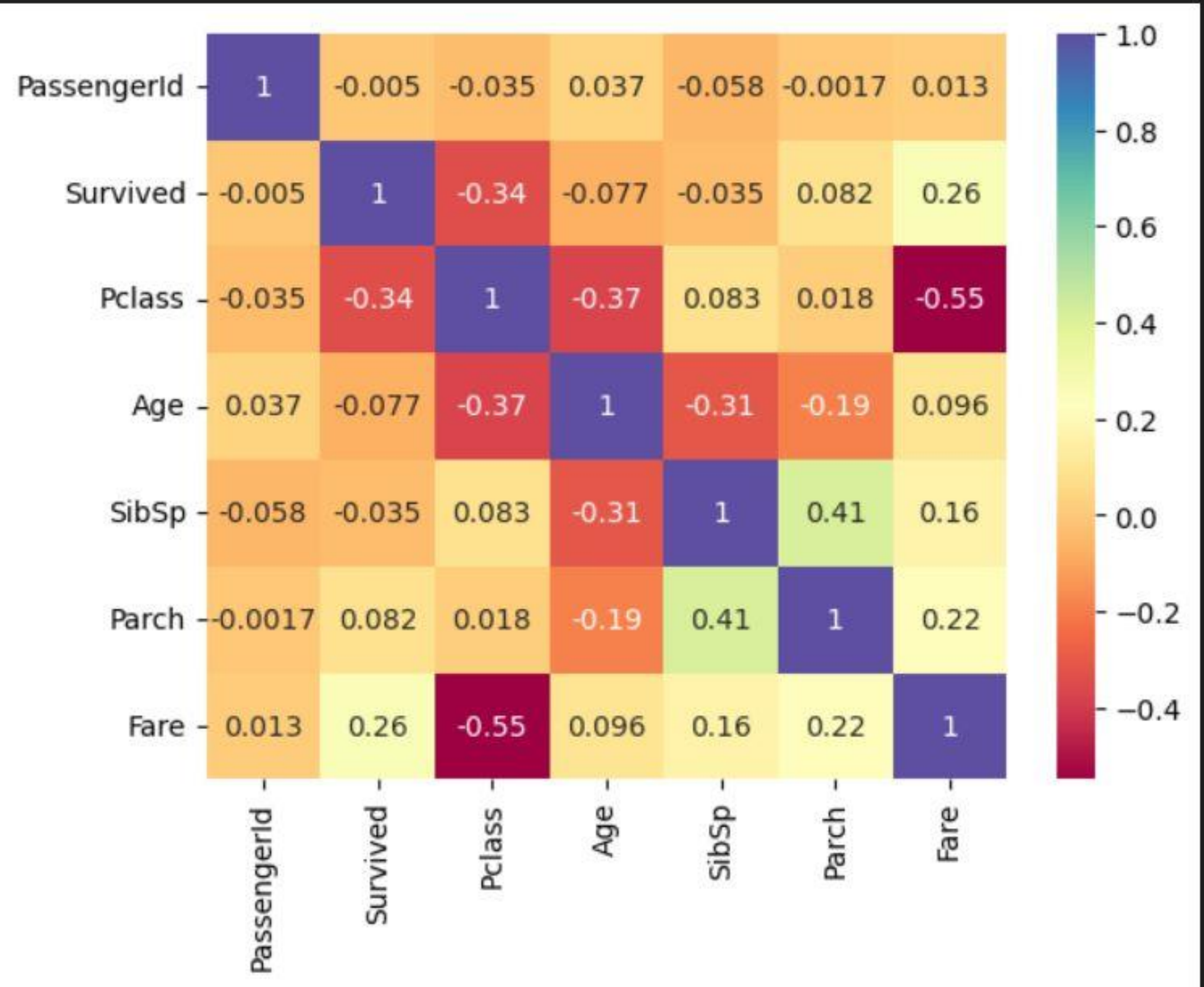


As we can see through histogram the women survival rate is much higher than men survival rate

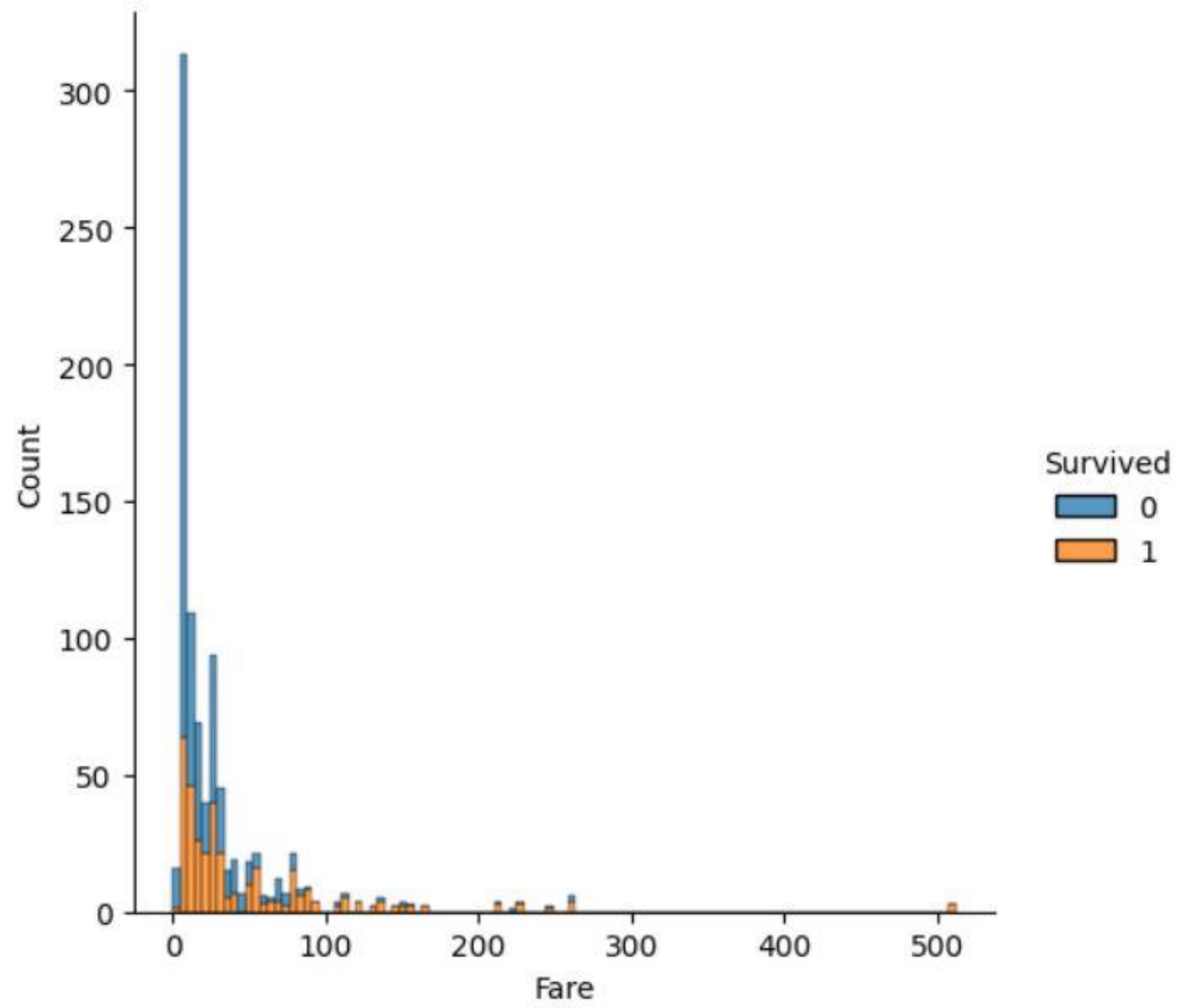
```python
print("\n" +'\033[1m','\033[94m', "Correlation matrix for the numerical columns:", '\033[0m' + "\n")
corr = titanic_data.corr(numeric_only = True)
sns.heatmap(corr, annot = True, cmap="Spectral")
plt.show()
```

✓ 0.4s

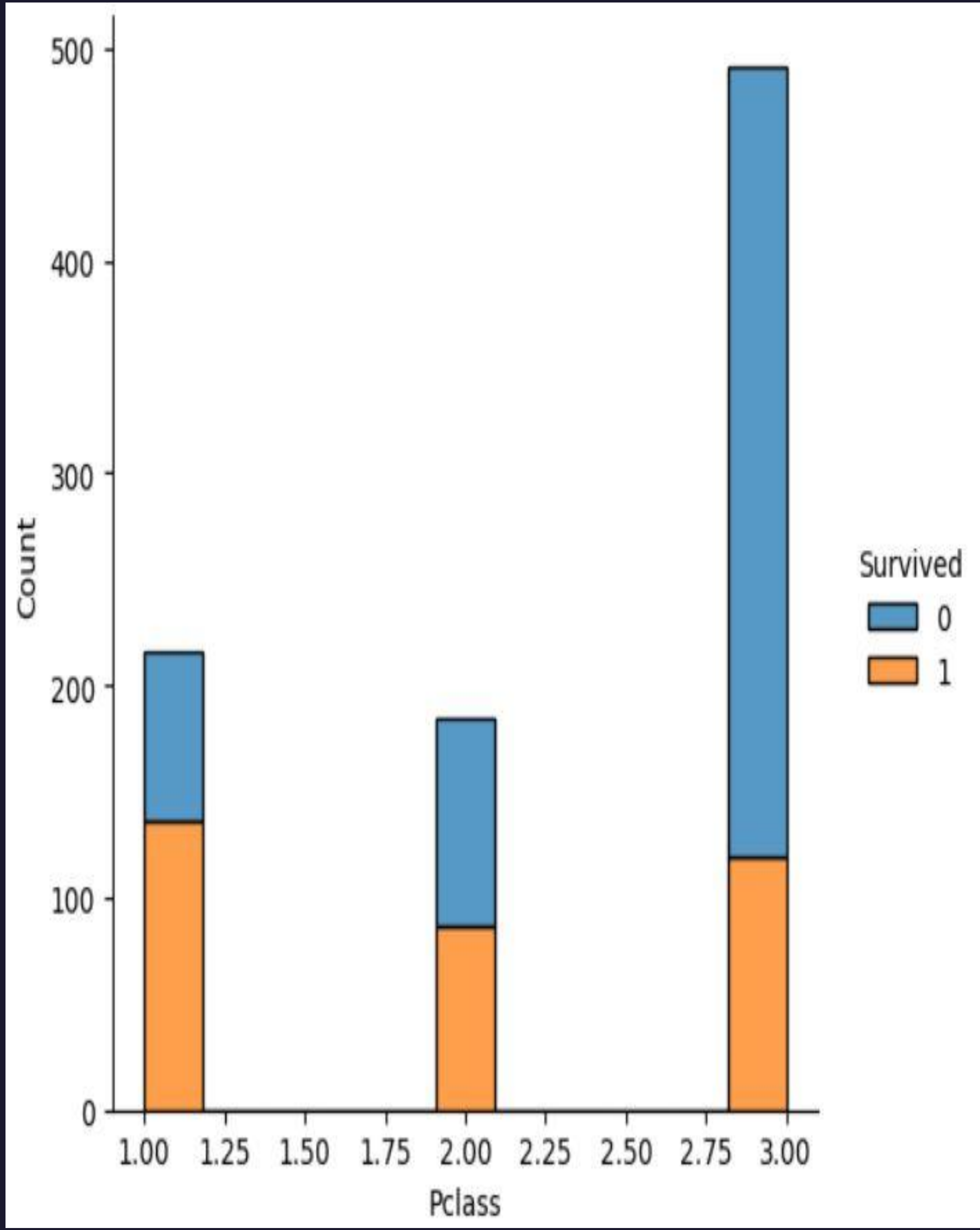
Correlation matrix for the numerical columns:



Correlation Matrix



Count
300
250
200
150
100
50
0
0
100
200
300
400
500
Fare
Survived
0
1



Count
500
400
300
200
100
0
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
Pclass
Survived
0
1

Model Selection

Data Modelling

+ Code + Markdown

Building Model using Logistic Regression

```
# import train test split method
```

✓ 0.2s

```
from sklearn.model_selection import train_test_split
```

✓ 1.6s

```
# train test split
```

✓ 0.0s

```
X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size = 0.33, random_state = 42)
```

✓ 0.2s

```
# import Logistic Regression
```

✓ 0.1s

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

✓ 0.3s

```
# Fit Logistic Regression
```
✓ 0.1s

```
logisticRegression = LogisticRegression()
```
✓ 0.1s

```
logisticRegression.fit(X_train, Y_train)
```
✓ 0.2s

```
c:\Python311\Lib\site-packages\sklearn\linear_model\_logistic.py:458: Converger
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
  n_iter_i = _check_optimize_result(
```

```
▾ LogisticRegression
LogisticRegression()
```

```
# Predict
```
✓ 0.0s

```
predict = logisticRegression.predict(X_test)
```
✓ 0.0s

Fitting and predicting the model

Testing and Accuracy Score

Testing

```
# Print Confusion matrix
```
✓ 0.1s

```
from sklearn.metrics import confusion_matrix
```
✓ 0.0s

```
pd.DataFrame(confusion_matrix(Y_test, predict), columns = ["Predicted No", "Predicted Yes"], index = ["Actual No", "Actual Yes"])
```
✓ 0.0s

	Predicted No	Predicted Yes
Actual No	151	24
Actual Yes	37	83

```
# import Classification Report
```
✓ 0.0s

```
from sklearn.metrics import classification_report
```
✓ 0.0s

```
print(classification_report(Y_test, predict))
```
✓ 0.1s

```
              precision    recall  f1-score   support

           0       0.80      0.86      0.83       175
           1       0.78      0.69      0.73       120

    accuracy                           0.79       295
   macro avg       0.79      0.78      0.78       295
weighted avg       0.79      0.79      0.79       295
```

Accuracy = 79%

F1 Score = 78%

Thank You

Mohit Dhakad

Chetan

Harsukh Singh Sagri

Nitin Babu