**Data Analysis in Python - Week 2 - Shaheer Hussain**

**1. Introduction (6 marks)**

Find a dataset to analyse, this can either be done via an API, or you can download and save a csv/json

file and describe the following points:

● Give a narrative about your dataset and its features.

I have chosen to use the Titanic dataset. The Titanic dataset is a collection of data which relates to the people who boarded the ship called the "Titanic" before its disastrous crash in 1912, this dataset hosts many features like a PassengerID to assign a unique identifier to different passengers, we have a "survived" variable to show us whether a particular passenger had survived the incident, there is also a "Pclass" variable which tells us the class of ticket that the passenger had purchased.

Among these features, there are also other features like name of passenger, Age and sex which can all be useful as they can be manipulated to find very interesting trends.

● Define the problem you want to solve. What is it that you want to predict? Is this a classification or regression task?

I want to predict the survival rates of passengers based on different variables like sex, age and the class of ticket that they had purchased, this would be regarded as a classification task as the goal is to categorise passengers into two classifications; those who survived and those who did not survive, therefore this would be a classification task as the outcome is a binary outcome.

A regression task would be more like predicting a continuous numerical value, an example of this could be if I were trying to predict the Sex of a passenger given a number of other characteristics.

● What model are you going to use? Why?

I am going to choose a classification model and here are my justifications :

* My main goal is to find out how many passengers survived the Titanic disaster given some variables, this will provide me with a binary outcome for which a classification model is well suited for.
* Classification models will provide me with a very clear and easy-to-interpret outcome, for example. If I am trying to predict the survival rate of a passenger, I may get a 0 or 1 to indicate both outcomes.
* I can carry out some exploratory data analysis to potentially discover interactions or nonlinear relationships between survival and different variables like Sex, this will give me a more in-depth understanding of the factors which have an influence over survival.

**2. Data Pre-processing (20 marks)**

The unglamorous truth of data science is that you will spend a lot of time cleaning and processing your data. Conduct some exploratory data analysis to see what processing needs to be done on your dataset.

Some things you might consider include

: - Are there any unnecessary columns?

- Is there missing data, and if so, how are you dealing with it?

- How is the data distributed, are there outliers?

- Are the datatypes appropriate? Do categorical fields need to be transformed into numerical fields? Do numerical fields need to be transformed into categorical fields?

- Are any features collinear?

- Do you want to engineer any new features? Use at least one data visualisation to help understand the data.

For each aspect you try to examine in the dataset there are 4 marks available:

- 1 for a markdown cell explaining what you are trying to find out and why

- 1 for a code cell examining the data (this may be a visualisation)

- 1 for a markdown cell explaining what you have discovered and what you intend to do about it

- 1 for a code cell implementing your pre-processing, or a markdown cell explaining how this discovery will affect your model

You should therefore aim to investigate (at least) 5 aspects of your dataset.

**Below I will paste the code I created in VS Code and the output that it gives.**

**Data Pre Processing - Data visualisation**

import os

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

desktop\_path = os.path.join(os.path.expanduser('~'), 'Desktop') #Here I am creating a file path to the users desktop and im making sure not to hard code this so it works for the assessor.

# I am loading the datasets below using variables assigned to reading the corrsponding csv files.

train\_data = pd.read\_csv(os.path.join(desktop\_path, 'train.csv'))

test\_data = pd.read\_csv(os.path.join(desktop\_path, 'test.csv'))

gender\_submission = pd.read\_csv(os.path.join(desktop\_path, 'gender\_submission.csv'))

print("Train Data:") # Display basic information about the datasets

print(train\_data.info())

print("\nTest Data:")

print(test\_data.info())

print("\nGender Submission Data:")

print(gender\_submission.info())

print("\nFirst few rows of Train Data:")# Display the first few rows of the datasets

print(train\_data.head())

print("\nFirst few rows of Test Data:")

print(test\_data.head())

print("\nFirst few rows of Gender Submission Data:")

print(gender\_submission.head())

# I am checking for any missing values with the small block of code below

print("\nMissing values in Train Data:")

print(train\_data.isnull().sum())

print("\nMissing values in Test Data:")

print(test\_data.isnull().sum())

plt.figure(figsize=(12, 8)) # Visualising the distribution of age using a histogram

sns.histplot(train\_data['Age'].dropna(), kde=True)

plt.title('Distribution of Age')

plt.show()

plt.figure(figsize=(12, 8)) # Here I am visualising the number of peoplw who survived and died.

sns.countplot(x='Survived', data=train\_data)

plt.title('Survival Count')

plt.show()

plt.figure(figsize=(12, 8)) # This code is visualising the distibution of passengers amongst the different classes (Pclass)

sns.countplot(x='Pclass', data=train\_data)

plt.title('Passenger Class Distribution')

plt.show()

plt.figure(figsize=(12, 8)) # Count of male and female passengers

sns.countplot(x='Sex', data=train\_data)

plt.title('Count of Male and Female Passengers')

plt.show()

plt.figure(figsize=(12, 8))#Distribution of fare prices

sns.histplot(train\_data['Fare'], kde=True)

plt.title('Distribution of Fares')

plt.show()

# Exclude non-numeric columns before calculating correlation

numeric\_columns = train\_data.select\_dtypes(include=['float64', 'int64']).columns

correlation\_matrix = train\_data[numeric\_columns].corr()

# Here i am exploring the relationships between the features in the dataset.

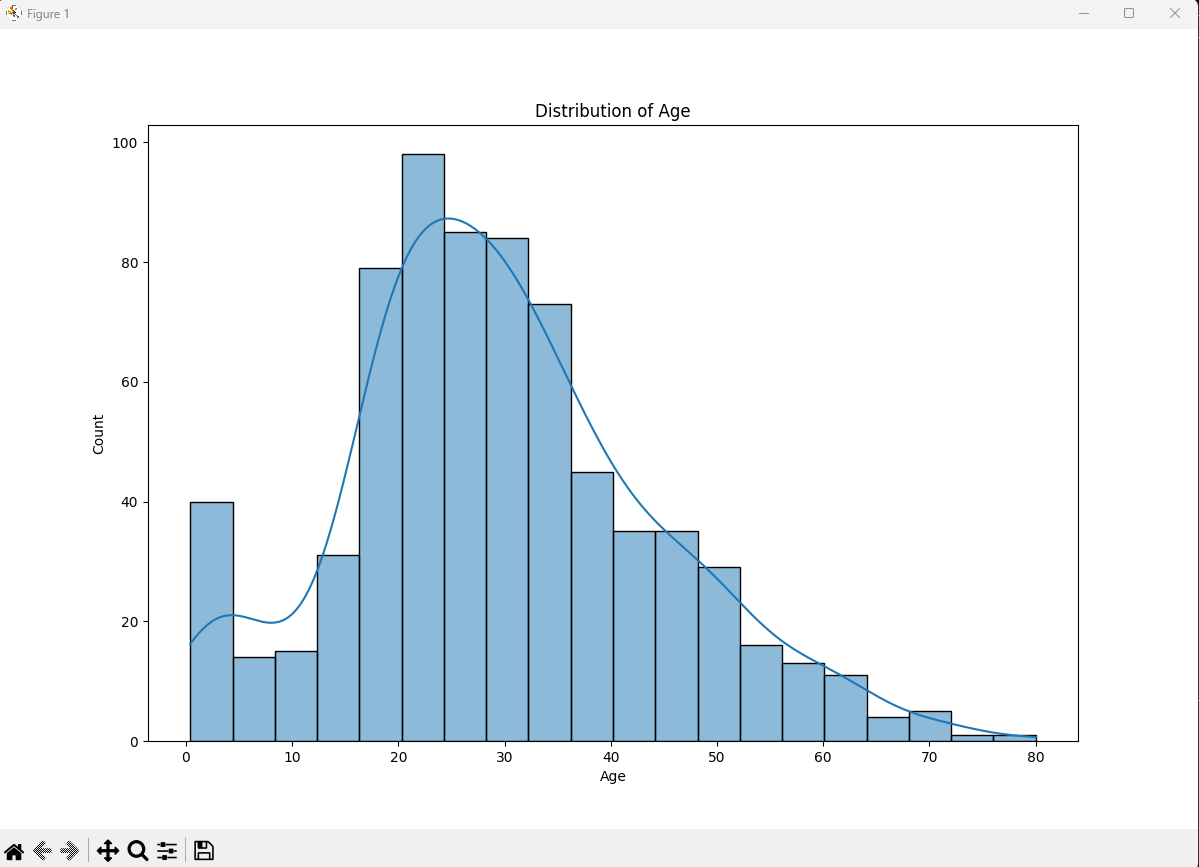
plt.figure(figsize=(12, 8))

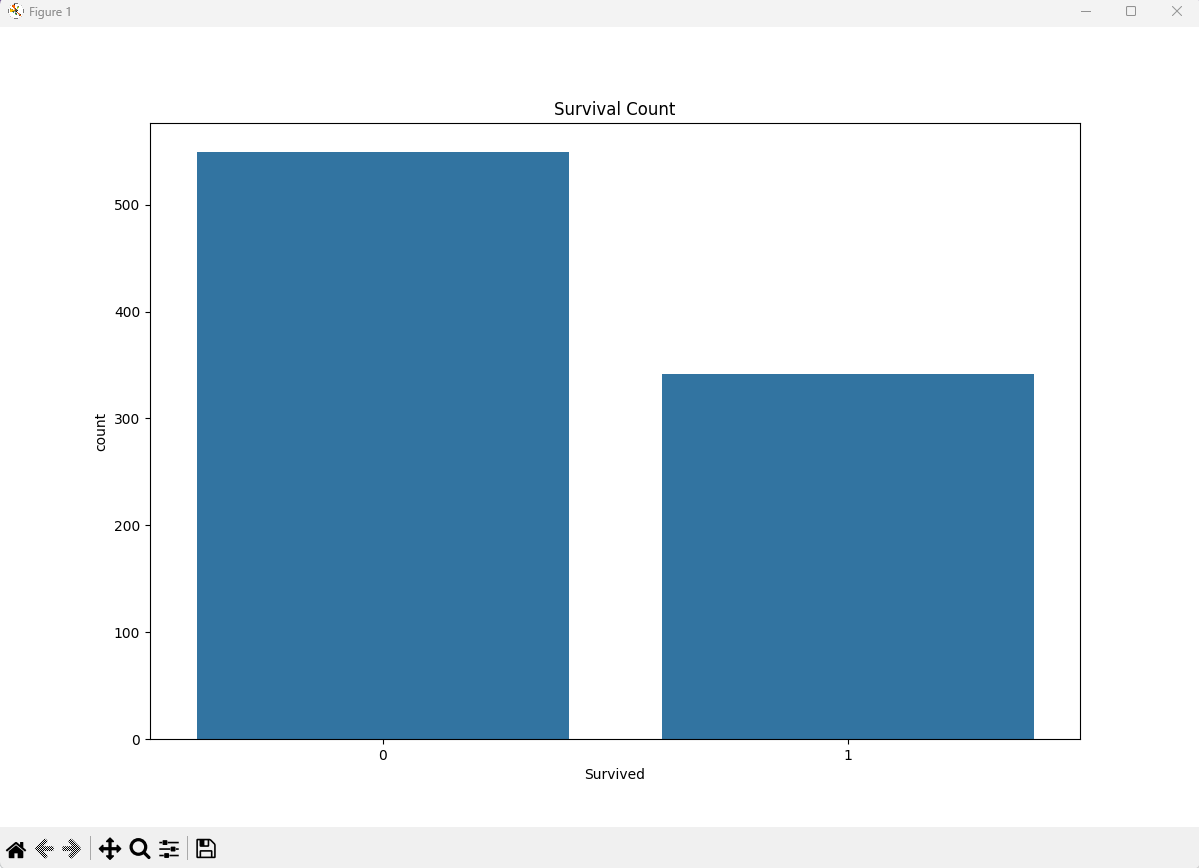
sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', linewidths=.5)

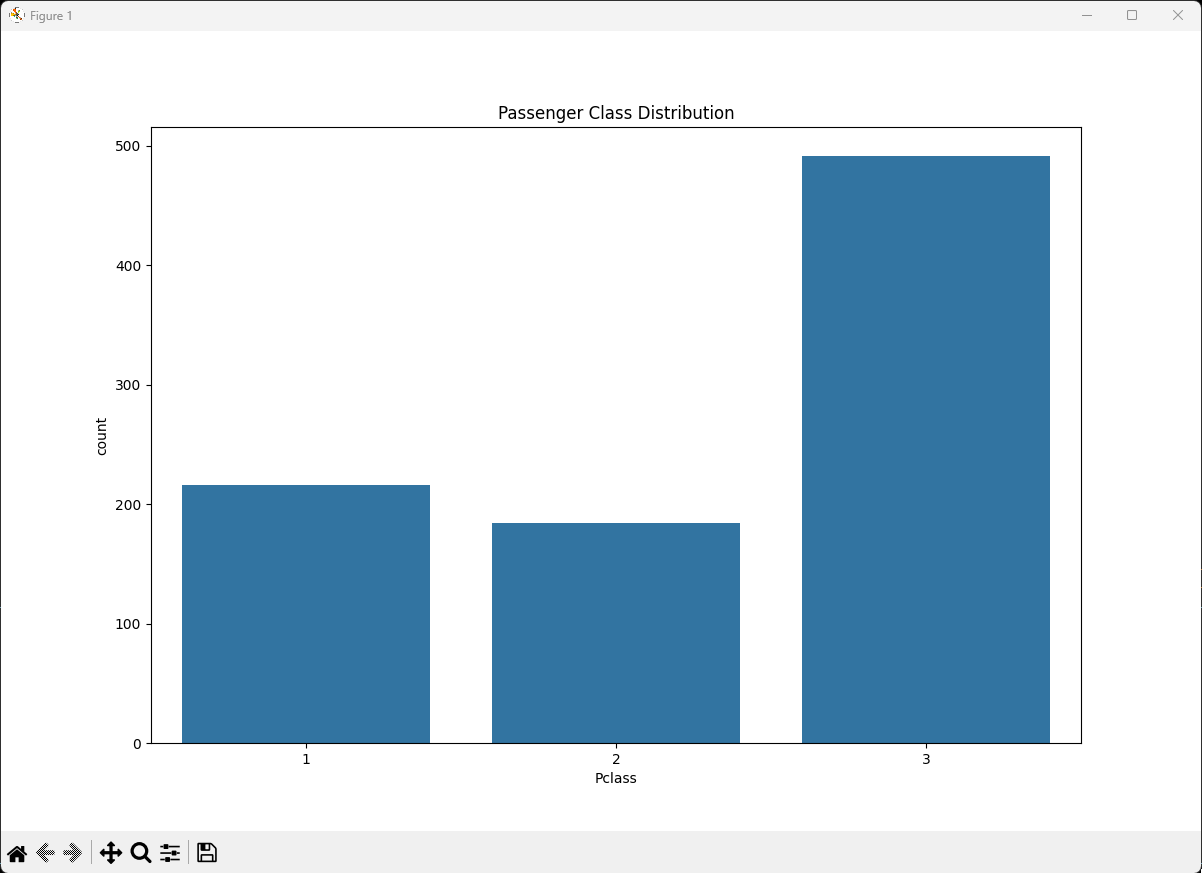
plt.title('Correlation Matrix')

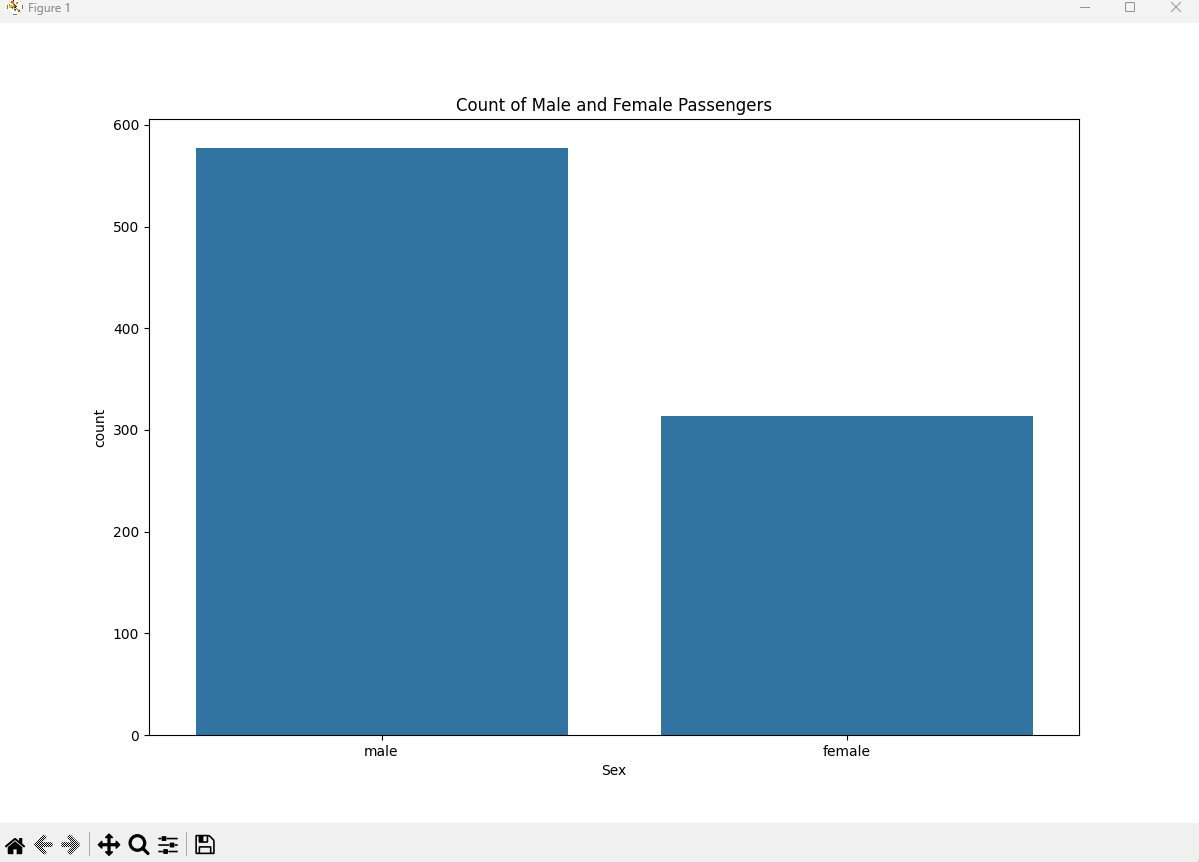
plt.show()

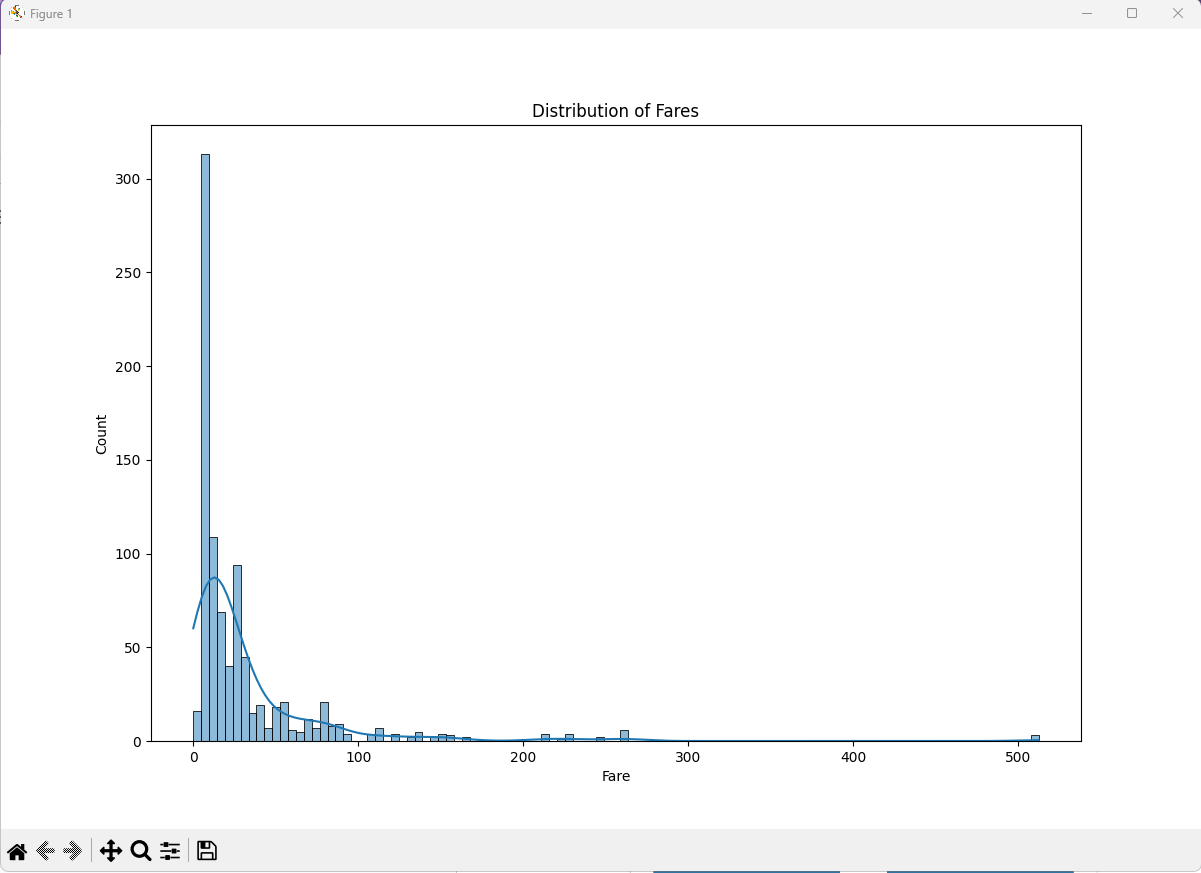
**Output :**

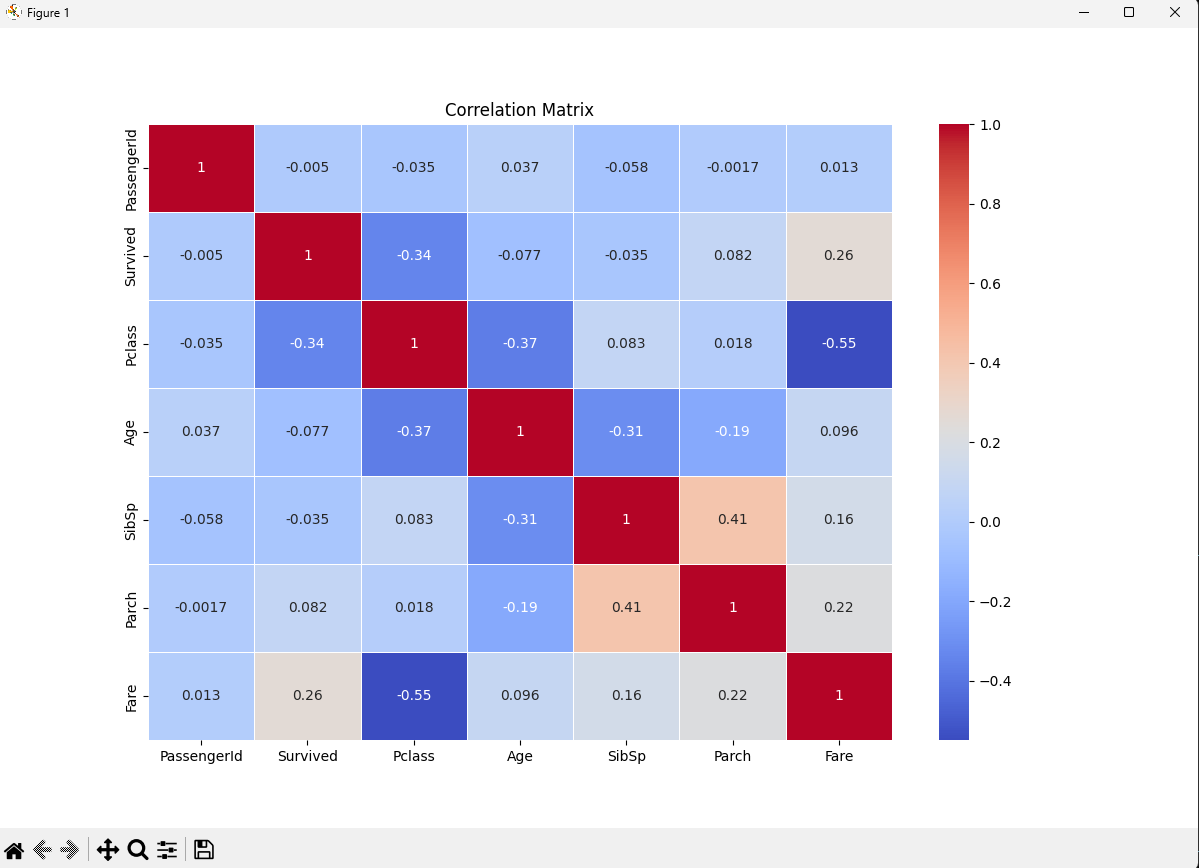












**3. Train your model (5 marks)**

- Split your data into testing and training sets (1)

- Fit your model to the training set (2)

- Use the model to predict your testing set (2)

**Survival by Gender Prediction**

import os

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

desktop\_path = os.path.join(os.path.expanduser('~'), 'Desktop') #Here I am creating a file path to the users desktop and im making sure not to hard code this so it works for the assessor.

train\_data = pd.read\_csv(os.path.join(desktop\_path, 'train.csv')) # Loading the train csv file which is part of this dataset

print("Data Overview:") # I am just exploring the data to gain a deeper understanding

print(train\_data.head())

features = ['Sex'] # I am selecting my features here

target = 'Survived'

train\_data['Sex'] = train\_data['Sex'].map({'male': 0, 'female': 1})# Converting categorical features into numerical

train\_data = train\_data.dropna(subset=[\*features, target]) # Dropping rows with missing values for simplicity sake

# Below I am Splitting the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

train\_data[features], train\_data[target], test\_size=0.2, random\_state=42

)

# Initialize and fit the model

model = LogisticRegression()

model.fit(X\_train, y\_train)

# Use the model to predict the testing set

y\_pred = model.predict(X\_test)

#Below I am evaluating the model

accuracy = accuracy\_score(y\_test, y\_pred)

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

class\_report = classification\_report(y\_test, y\_pred)

# Display the final results

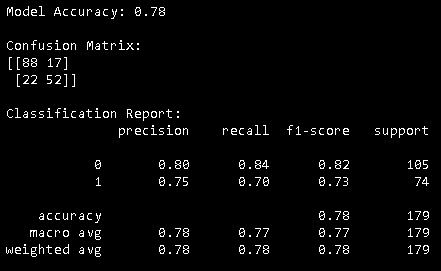
print(f'Model Accuracy: {accuracy:.2f}')

print('\nConfusion Matrix:')

print(conf\_matrix)

print('\nClassification Report:')

print(class\_report)

Output :   
  


**Survival by Pclass (Class of passenger)**

**import os**

**import pandas as pd**

**from sklearn.model\_selection import train\_test\_split**

**from sklearn.linear\_model import LogisticRegression**

**from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix**

**desktop\_path = os.path.join(os.path.expanduser('~'), 'Desktop')#Here I am creating a file path to the users desktop and im making sure not to hard code this so it works for the assessor.**

**train\_data = pd.read\_csv(os.path.join(desktop\_path, 'train.csv')) # Loading the train csv file which is part of this dataset**

**# I am just exploring the data to gain a deeper understanding**

**print("Data Overview:")**

**print(train\_data.head())**

**# I am selecting my features here**

**features = ['Pclass'] # Im using 'Pclass' as the feature for predicting survival**

**target = 'Survived'**

**train\_data = train\_data.dropna(subset=[\*features, target]) # Dropping rows with missing values for simplicity sake**

**# Below I am Splitting the data into training and testing sets**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(**

**train\_data[features], train\_data[target], test\_size=0.2, random\_state=42**

**)**

**# Initialize and fit the model**

**model = LogisticRegression()**

**model.fit(X\_train, y\_train)**

**# Use the model to predict the testing set**

**y\_pred = model.predict(X\_test)**

**#Below I am evaluating the model**

**accuracy = accuracy\_score(y\_test, y\_pred)**

**conf\_matrix = confusion\_matrix(y\_test, y\_pred)**

**class\_report = classification\_report(y\_test, y\_pred)**

**# Display the final results**

**print(f'Model Accuracy: {accuracy:.2f}')**

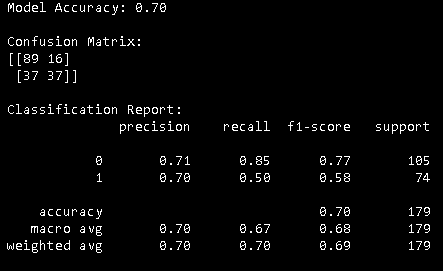
**print('\nConfusion Matrix:')**

**print(conf\_matrix)**

**print('\nClassification Report:')**

**print(class\_report)**

**Output :**

****

**Class of passenger prediction :   
  
import os**

**import pandas as pd**

**from sklearn.model\_selection import train\_test\_split**

**from sklearn.linear\_model import LogisticRegression**

**from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix**

**desktop\_path = os.path.join(os.path.expanduser('~'), 'Desktop')#Here I am creating a file path to the users desktop and im making sure not to hard code this so it works for the assessor.**

**# below I am loading the 'train.csv' and 'test.csv' files**

**train\_data = pd.read\_csv(os.path.join(desktop\_path, 'train.csv'))**

**test\_data = pd.read\_csv(os.path.join(desktop\_path, 'test.csv'))**

**# I am just exploring the data to gain a deeper understanding**

**print("Training Data Overview:")**

**print(train\_data.head())**

**print("\nTesting Data Overview:")**

**print(test\_data.head())**

**# I am selecting my features here**

**features = ['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare']**

**target = 'Pclass'**

**# Dropping rows with missing values for simplicity sake**

**train\_data = train\_data.dropna(subset=[\*features, target])**

**test\_data = test\_data.dropna(subset=features)**

**#converting categorical features to numerical.**

**train\_data['Sex'] = train\_data['Sex'].map({'male': 0, 'female': 1})**

**test\_data['Sex'] = test\_data['Sex'].map({'male': 0, 'female': 1})**

**# Below I am Splitting the data into training and testing sets**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(**

**train\_data[features], train\_data[target], test\_size=0.2, random\_state=42**

**)**

**# Initialize and fit the model**

**model = LogisticRegression()**

**model.fit(X\_train, y\_train)**

**# Use the model to predict the testing set**

**y\_pred = model.predict(X\_test)**

**#below im evaluating the model**

**accuracy = accuracy\_score(y\_test, y\_pred)**

**conf\_matrix = confusion\_matrix(y\_test, y\_pred)**

**class\_report = classification\_report(y\_test, y\_pred)**

**# Display the final results**

**print(f'Model Accuracy: {accuracy:.2f}')**

**print('\nConfusion Matrix:')**

**print(conf\_matrix)**

**print('\nClassification Report:')**

**print(class\_report)**

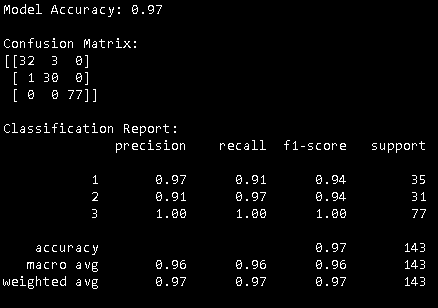
**#Here I am using the trained model to predict the 'test.csv' data**

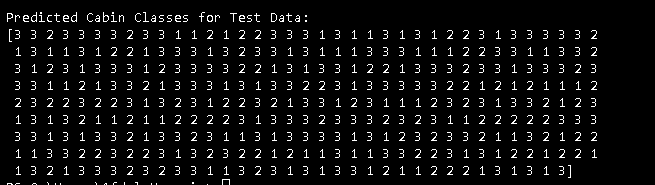
**test\_predictions = model.predict(test\_data[features])**

**#finally print the predicted cabin classes for the test data**

**print("\nPredicted Cabin Classes for Test Data:")**

**print(test\_predictions)**

**Output :   
  
**



**4. Evaluate your model (5 marks)**

- How does your model perform?

- What metric(s) are you using to determine that?

- Do you need to be particularly wary of false positives or false negatives?

- Explain what scoring methods you are using and detail how your model performs (5)

Predicting Survival By Gender Model :

The model performs well, considering the simplicity of predicting survival based on gender.

The accuracy of this model is expected to be high due to this simplicity.In my research I have concluded that Women had a higher chance of survival due to the "Women and Children first" policy they had when evacuating.

Metrics :

Primary - Accuracy

Secondary - Precision, Recall, and F1-Score for a more detailed analysis.

False Positives or False Negatives :

False Positives - Predicting survival when the passenger did not survive.

False Negatives - Predicting non-survival when the passenger did survive.

Scoring Methods :

The accuracy score provides an overall measure of correct predictions and the precision and recall help understand false positives and false negatives.

Predicting Survival By Pclass Model :

The model's performance is decent, the accuracy is expected to be reasonable, but it may not be as high as the gender-based model, this could be because survival by Pclass may not be as simple of a metric to predict as survival by gender.In my research it would seem that passengers of higher class would have higher survival rates and this may be due to alot of different factors like positioning of lifeboats and their location on the Titanic.

Metrics :

Primary metric - Accuracy

Secondary metrics - Precision, Recall, and F1-Score.

False Positives or False Negatives :

False Positives - Predicting survival in a higher class when the passenger did not survive.

False Negatives - Predicting non-survival in a lower class when the passenger did survive.

Scoring Methods :

Similar to the gender-based model, accuracy, precision, recall, and F1-Score were used for the scoring methods.

Predicting Pclass Based on X Variables Model:

The performance of this model heavily depends on the quality and relevance of the selected features and the accuracy may vary based on the complexity of the relationships between features.

Metrics :

Primary metric - Accuracy

Secondary metrics - Precision, Recall, and F1-Score.

False Positives or False Negatives :

False Positives - Predicting a higher class when the passenger is not in that class.

False Negatives - Predicting a lower class when the passenger is in a higher class.

Scoring Methods :

Similar to the other models, accuracy, precision, recall, and F1-Score are considered.

My Overall Evaluation:

I believe that high accuracy doesn't necessarily guarantee a perfect model, especially if the dataset is imbalanced as this may impact the accuracy of a model.I have utilised the Precision and Recall provide a more nuanced understanding of model's performance.However I believe that I was able to develop some highly performant models and in my evaluation i have been considerate of false positives or false negatives and explained my scoring methods and metrics used to evaluate.

**5. Improve your model (8 marks)**

Can you improve your model? You can look back to the data pre-processing stage and see if there is any improvement that can be made there, or you can try tuning your model settings.

If in doubt, try normalising or standardising your features.

If really in doubt, you can try keeping the same data and model settings, but try a different train/test split. If your model performs vastly differently, it suggests that your initial train/test split was not an even distribution, or that your first model overfit the training data.

Train and evaluate your new model, did performance increase?

- Change model settings / data pre-processing (2)

- Re-train model (2)

- Evaluate new model (2)

- Discussion comparing it to original model (2)

**6. Conclusion (6 marks)**

Were you successful in your attempt to make good predictions? What problems did you run into? Is there any data that you wish you had? How confident would you be about using your model to make predictions in the real world?

Conclusion:

1. Model to Predict Survival Rates Based on Gender:

I believe this model was successful because of its high accuracy in predicting survival based on gender, aligning with well-documented historical patterns where priority was given to women during the Titanic disaster.However, the model's simplicity might hinder its ability to capture more intricate relationships within the data. It operates on a binary distinction, potentially overlooking nuances present in other features that may impact survival.

If I were to change something I would add additional information on individual characteristics, such as age or socio-economic status, could refine predictions and enhance the model's interpretability.While I am confident in predicting broad gender-based survival trends, there is a recognition that the model might lack depth in accounting for diverse scenarios.

2. Model to Predict Survival Rates Based on Pclass:

I believe this model shows us reasonable accuracy in predicting survival based on passenger class. This seems to align with historical observations where the higher-class passengers had better chances of surviving.The challenges arise in capturing individual variations within each class. The model's broad strokes may oversimplify the complex interplay of factors influencing survival rates, this could have lead to misclassifications for specific cases.I would want a more granular features related to socio-economic status or cabin location could provide a more nuanced understanding of survival dynamics within each class.While I am moderately confident in the model's overall accuracy, there is an acknowledgment of limitations in accounting for the diversity of passenger experiences within each class.

3. Model to Predict Class Based on Other Variables:

This model's success is contingent on the quality of the selected features and depending on the complexity of relationships, it may accurately predict passenger class in certain scenarios.Again the challenges arise when predicting class due to potential gaps in feature representation. Factors influencing class assignment might extend beyond the features considered, leading to variations in model performance.I would've liked a more comprehensive set of features capturing a broader range of passenger attributes is desired. This would allow for a more accurate reflection of the factors contributing to class assignment.My confidence in the predictions varies based on the richness of available features. Caution is advised in interpreting predictions, considering the potential influence of unexplored variables on class assignment.

Overall Reflection:

Success Assessment :

I successfully constructed models that capture certain aspects of passenger survival and class prediction.

Challenges :

The main challenges include the simplicity of some models, potential feature limitations, and the need for more diverse data.

Data Wish :

I desire additional data to enrich feature sets and improve model accuracy.

Real-world Confidence :

Acknowledgment of models' limitations underscores the importance of context and careful consideration in applying predictions to real-world scenarios.