FINDINGS REPORT

"Exploratory Data Analysis – Titanic Dataset"

Tejitha.ch
internship/Project Title: "Data Analyst Internship – Task 5"

Objective:

The goal of this task was to perform Exploratory Data Analysis (EDA) on the Titanic dataset to discover patterns, trends, and relationships that impact passenger survival.

Dataset Description

Source: Kaggle Titanic Dataset
Columns Overview:

PassengerId , Survived, Pclass, Name, Sex, Age, SibSp, Parch, Ticket, Fare, Cabin, Embarked

Initial Data Exploration

.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  418 non-null    int64  
 1   Pclass       418 non-null    int64  
 2   Name         418 non-null    object 
 3   Sex          418 non-null    object 
 4   Age          332 non-null    float64
 5   SibSp        418 non-null    int64  
 6   Parch        418 non-null    int64  
 7   Ticket       418 non-null    object 
 8   Fare         417 non-null    float64
 9   Cabin        91 non-null     object 
 10  Embarked     418 non-null    object 
dtypes: float64(2), int64(4), object(5)
memory usage: 36.1+ KB
None
```

```
print(train_df.describe())
print(test_df.describe())
```

```
       PassengerId    Survived      Pclass         Age       SibSp  \
count   891.000000  891.000000  891.000000  714.000000  891.000000   
mean    446.000000    0.383838    2.308642   29.699118    0.523008   
std     257.353842    0.486592    0.836071   14.526497    1.102743   
min       1.000000    0.000000    1.000000    0.420000    0.000000   
25%     223.500000    0.000000    2.000000   20.125000    0.000000   
50%     446.000000    0.000000    3.000000   28.000000    0.000000   
75%     668.500000    1.000000    3.000000   38.000000    1.000000   
max     891.000000    1.000000    3.000000   80.000000    8.000000   

            Parch        Fare  
count  891.000000  891.000000  
mean     0.381594   32.204208  
std      0.806057   49.693429  
min      0.000000    0.000000  
25%      0.000000    7.910400  
50%      0.000000   14.454200  
75%      0.000000   31.000000  
max      6.000000  512.329200  
       PassengerId      Pclass         Age       SibSp       Parch        Fare
count   418.000000  418.000000  332.000000  418.000000  418.000000  417.000000
mean   1100.500000    2.265550   30.272590    0.447368    0.392344   35.627188
std     120.810458    0.841838   14.181209    0.896760    0.981429   55.907576
min     892.000000    1.000000    0.170000    0.000000    0.000000    0.000000
25%     996.250000    1.000000   21.000000    0.000000    0.000000    7.895800
50%    1100.500000    3.000000   27.000000    0.000000    0.000000   14.454200
75%    1204.750000    3.000000   39.000000    1.000000    0.000000   31.500000
max    1309.000000    3.000000   76.000000    8.000000    9.000000  512.329200
```

Data cleaning

Data cleaning is an essential step in the EDA process to ensure the dataset is consistent, complete, and ready for analysis or modelling. In this task, we identified and handled missing values, removed irrelevant features, and encoded categorical variables to prepare the Titanic dataset for analysis and machine learning.

Steps Performed:

Dropped Irrelevant or Non-Numeric Columns

We dropped columns that were not useful for analysis or modelling:

- Cabin – Contains too many missing values (over 75%) and is not directly helpful.
- Ticket – Categorical with high cardinality and no predictive pattern.
- Name – Unstructured data, not useful without NLP techniques.
- Passenger Id – Only dropped for model training, kept for final prediction file.

Code used:

```
cols_to_drop = ['Cabin', 'Ticket', 'Name']
if drop_passenger_id:
  cols_to_drop.append('PassengerId')
df.drop(columns=[col for col in cols_to_drop if col in df.columns], inplace=True)
```

Filled Missing Values

To avoid errors and retain data, we filled missing values using appropriate statistical methods:

- **Age**: Replaced missing values with the **median age**.
- **Fare**: Filled missing fares with the **median fare**.
- **Embarked**: Filled missing ports with the **most frequent (mode)** value.

Code used:

```
df['Age'] = df['Age'].fillna(df['Age'].median())
df['Fare'] = df['Fare'].fillna(df['Fare'].median())
df['Embarked'] = df['Embarked'].fillna(df['Embarked'].mode()[0])
```

Converted Categorical Columns into Numeric

Machine learning models can't work with string values, so we encoded them:

- **Sex**:
  - 'male' → 0
  - 'female' → 1
- **Embarked**: Converted to dummy variables using one-hot encoding and dropped the first category to avoid multicollinearity.

Code used:

```
df['Sex'] = df['Sex'].map({'male': 0, 'female': 1})
df = pd.get_dummies(df, columns=['Embarked'], drop_first=True)
```

Ensured Consistent Features Across Train and Test Data

We used .reindex() to make sure the test dataset has the **same columns in the same order** as the training dataset, and filled any missing columns with 0.

Code used:

```
X_test = test_clean.drop('PassengerId', axis=1)
X_test = X_test.reindex(columns=X.columns, fill_value=0)
```

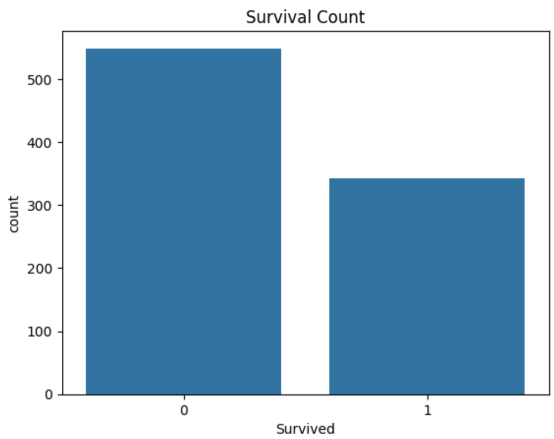
Handled Remaining Missing Values Using Imputer

To avoid NaN errors during model training and prediction, we applied a **SimpleImputer** with the median strategy on both training and test data.

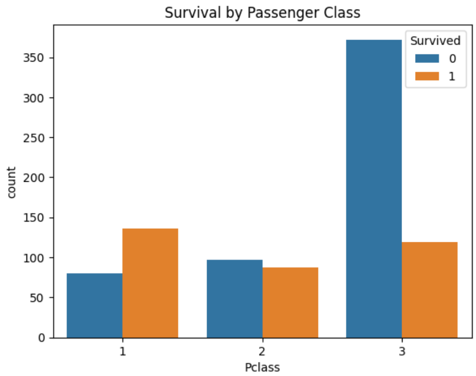
Code used:

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy='median')
X = pd.DataFrame(imputer.fit_transform(X), columns=X.columns)
X_test = pd.DataFrame(imputer.transform(X_test), columns=X.columns)
```

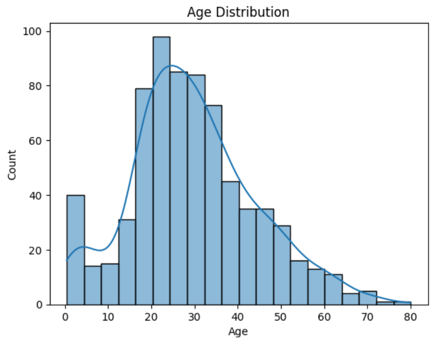
Visual Explorations and Observations



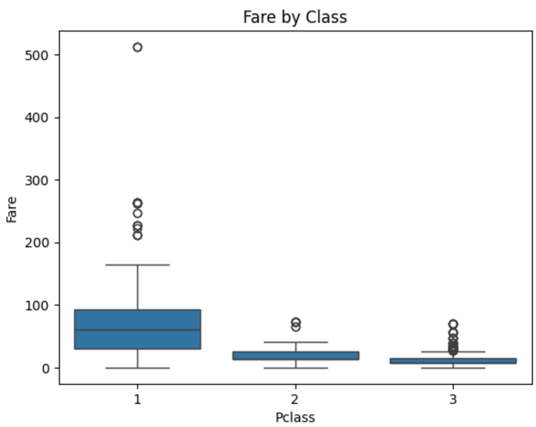
- Shows how many passengers survived (1) vs didn't (0).
  Observation: There were more non-survivors than survivors, showing **class imbalance** in the target variable.



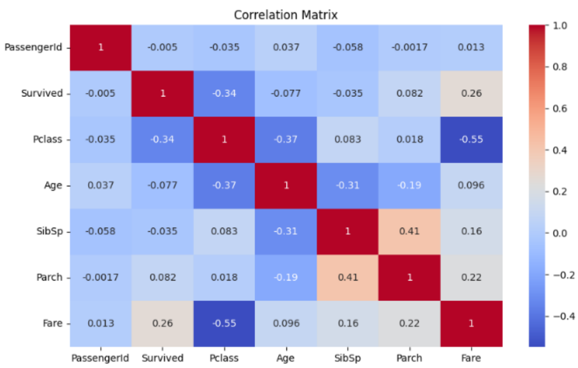
Visualizes survival based on passenger class.
Observation: First class passengers had the highest survival rate, while third class had the lowest.



Plots the age distribution of passengers.
Observation: Most passengers were between 20–40 years, and the data is right-skewed.



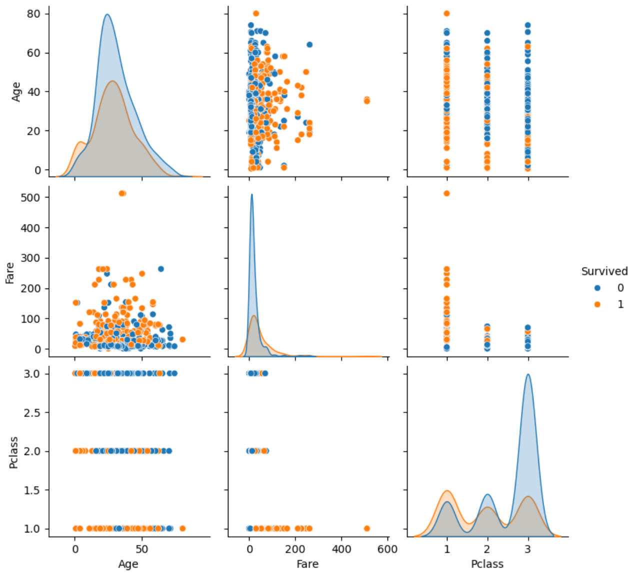
Shows how fare varies across classes.
Observation: First class passengers paid the most; outliers are present in all classes, especially in first class.



Displays correlation between numerical features.

Observation:

- Strong negative correlation between Pclass and Fare
- Survived is positively correlated with Fare and Sex (being female)



Shows pairwise scatterplots between multiple features.

Observation: Survivors cluster more in younger age + higher fare and 1st class categories.

Most survivors were **female** and in **1st class**.

Children (Age < 10) had higher survival.

Passengers who paid higher **fare** tended to survive more.

Strong correlation between **Pclass and Fare**, and **Sex and Survival**.