# Exploratory Data Analysis For Machine Learning

The dataset I am using for this assignment is the famous titanic dataset, which can be obtained online and from Kaggle. This data consists of variables as per below:

- passenger_id : id of passenger
- Pclass : Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd)
- Survived : Survival (0 = No; 1 = Yes)
- Name : Name
- Sex : Sex
- Age : Age
- SibSp : Number of Siblings/Spouses Aboard
- Parch : Number of Parents/Children Aboard
- Ticket : Ticket Number
- Fare : Passenger Fare (British pound)
- Cabin : Cabin
- Embarked : Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)
- boat :
- body : body type
- home.dest : destination of travellers

The sinking of the Titanic is one of the most infamous shipwrecks in history.

On April 15, 1912, during her maiden voyage, the widely considered “unsinkable” RMS Titanic sank after colliding with an iceberg. Unfortunately, there weren’t enough lifeboats for everyone onboard, resulting in the death of 1502 out of 2224 passengers and crew.

While there was some element of luck involved in surviving, it seems some groups of people were more likely to survive than others.

This dataset contains information on the passengers, their characteristics/attributes, and whether they survived the tragedy. We would explore this data, take a look at the data quality, do some pre□processing, and analyse the attributes a.k.a features and see whether they have a relation with our target variable, that is whether they survived, or not. The dataset has already been divided into train and test sets for model building purposes, but for our EDA analysis, let's just use the train data since it contains the target variable ('Survived' column) which is needed for our hypothesis testing later. Let's load in the data and take a look at the columns.

```
# imports
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

```
# loading data
data = pd.read_csv('/content/sample_data/titanic_train.csv')
```

```
data.head()
```

| | passenger_id | pclass | name | sex | age | sibsp | parch | ticket | fare | cabin |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1216 | 3 | Smyth, Miss. Julia | female | NaN | 0 | 0 | 335432 | 7.7333 | NaN |
| **1** | 699 | 3 | Cacic, Mr. Luka | male | 38.0 | 0 | 0 | 315089 | 8.6625 | NaN |
| **2** | 1267 | 3 | Van Impe, Mrs. Jean Baptiste (Rosalie Paula Go... | female | 30.0 | 1 | 1 | 345773 | 24.1500 | NaN |
| **3** | 449 | 2 | Hocking, Mrs. Elizabeth (Eliza Needs) | female | 54.0 | 1 | 3 | 29105 | 23.0000 | NaN |
| **4** | 576 | 2 | Veal, Mr. James | male | 40.0 | 0 | 0 | 28221 | 13.0000 | NaN |

```
data.shape
```

```
(850, 15)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 850 entries, 0 to 849
Data columns (total 15 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   passenger_id  850 non-null    int64  
 1   pclass        850 non-null    int64  
 2   name          850 non-null    object 
 3   sex           850 non-null    object 
 4   age           676 non-null    float64
 5   sibsp         850 non-null    int64  
 6   parch         850 non-null    int64  
 7   ticket        850 non-null    object 
 8   fare          849 non-null    float64
 9   cabin         191 non-null    object 
 10  embarked      849 non-null    object 
```

```
 11  boat          308 non-null    object
 12  body          73 non-null     float64
 13  home.dest     464 non-null    object
 14  survived      850 non-null    int64
dtypes: float64(3), int64(5), object(7)
memory usage: 99.7+ KB
```

By looking at the dataset info and the first five rows, we can see that this data contains features of different types. Here's my initial impressions on this dataset:

Preliminary Analysis :

- The data contains 850 rows and 15 columns (14 attributes and 1 target variable)
- Some of the columsn like Name, PassengerId and Ticket are simply identification tags, and we will not use it in our analysis.
- There are both numerical and categorical variables here, so later on we will do some feature engineering where we can encode our categorical variables.

## Data Cleaning & Feature Engineering

Before we continue further, let's drop the columns that we do not need for our EDA analysis

```
# dropping the columns
cols = ['passenger_id', 'name', 'ticket']
data = data.drop(columns=cols)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 850 entries, 0 to 849
Data columns (total 12 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   pclass     850 non-null    int64
 1   sex        850 non-null    object
 2   age        676 non-null    float64
 3   sibsp      850 non-null    int64
 4   parch      850 non-null    int64
 5   fare       849 non-null    float64
 6   cabin      191 non-null    object
 7   embarked   849 non-null    object
 8   boat       308 non-null    object
 9   body       73 non-null     float64
 10  home.dest  464 non-null    object
 11  survived   850 non-null    int64
dtypes: float64(3), int64(4), object(5)
memory usage: 79.8+ KB
```

```
data.head()
```

| | pclass | sex | age | sibsp | parch | fare | cabin | embarked | boat | body | home.des |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 3 | female | NaN | 0 | 0 | 7.7333 | NaN | Q | 13 | NaN | Nal |
| **1** | 3 | male | 38.0 | 0 | 0 | 8.6625 | NaN | S | NaN | NaN | Croati |
| **2** | 3 | female | 30.0 | 1 | 1 | 24.1500 | NaN | S | NaN | NaN | Nal |

## Data cleaning

Now let's look at the data quality. Even from the first five rows, we can see that there are several 'NaN' values for the Cabin. Let's dig deeper and see how many missings there are for this dataset for every column.

```
total = data.isnull().sum().sort_values(ascending=False)
percent = (data.isnull().sum() / data.isnull().count()).sort_values(ascending=False)
missing_data = pd.concat([total, percent], axis = 1, keys = ['Total', 'Percent'])

f, ax = plt.subplots(figsize = (15,6))
plt.xticks(rotation = '90')
sns.barplot(x=missing_data.index, y=missing_data['Percent'])
plt.xlabel('Features', fontsize=12)
plt.ylabel('Percent of missing values', fontsize=12)
plt.title('Percent of missing data by feature', fontsize=12)
missing_data
```

| | Total | Percent |
|---|---|---|
| **body** | 777 | 0.914118 |
| **cabin** | 659 | 0.775294 |
| **boat** | 542 | 0.637647 |
| **home.dest** | 386 | 0.454118 |
| **age** | 174 | 0.204706 |
| **embarked** | 1 | 0.001176 |
| **fare** | 1 | 0.001176 |
| **survived** | 0 | 0.000000 |

Some interesting information here. We can see that the Cabin, body, boat, home_dest column has about more than 60% missing values. It would be difficult for us to manipulate or impute these values because it would be very misleading to impute this column based on small percentage of values that is available to us. Let's drop this column from our analysis.

```
cols = ['body', 'cabin', 'boat', 'home.dest']
data = data.drop(columns=cols)
data.shape
```

```
(850, 8)
```

Now let's shift our focus to the Age column, which has about 19.86% of missing values. This is quite significant. Even if we remove the rows from our analysis, we might be removing quite a signicant information from our dataset.

There are several ways to impute this column, but I am going to try to impute it with a dependency on other variables. That is, to impute it with the mean, but not considering the global mean of the variable alone, but by mean of the variable based on other attributes.

Before explaining further, let me show the mean of the Age column.

```
data["age"].mean()
```

```
29.519847189349115
```

Now let's take a look at the mean Age value when we take into consideration their Sex.

```
data.groupby('sex')['age'].mean()
```

```
sex
female    28.858401
male      29.898256
Name: age, dtype: float64
```

The mean for Age column is 29, but if consider their sex, the average age of Male is 30, while for female it's 27.

Let's incorporate this information and impute the missing values:

```
data['age'].fillna(data.groupby('sex')['age'].transform('mean'), inplace=True)
```

```
# for fare imputing it with the mean
data['fare'] = data['fare'].fillna(data['fare'].mean())
```

The other column that has missing values is 'Embarked'. This column is categorical, and therefore,we cannot impute with mean or similar kind of impute for numerical variables.

One way to go about it is to create a new feature that indicates there is a missing value in this column.

```
data['null_embarked'] = data.embarked.isnull().astype(int)
data.head()
```

| | pclass | sex | age | sibsp | parch | fare | embarked | survived | null_embarked |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 3 | female | 28.858401 | 0 | 0 | 7.7333 | Q | 1 | 0 |
| **1** | 3 | male | 38.000000 | 0 | 0 | 8.6625 | S | 0 | 0 |
| **2** | 3 | female | 30.000000 | 1 | 1 | 24.1500 | S | 0 | 0 |
| **3** | 2 | female | 54.000000 | 1 | 3 | 23.0000 | S | 1 | 0 |
| **4** | 2 | male | 40.000000 | 0 | 0 | 13.0000 | S | 0 | 0 |

```
# imputing the embarked column
data['embarked'] = data['embarked'].fillna('NaN')

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

data['embarked'] = le.fit_transform(data['embarked'])
data.head()
```

| | pclass | sex | age | sibsp | parch | fare | embarked | survived | null_embarked |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 3 | female | 28.858401 | 0 | 0 | 7.7333 | 2 | 1 | 0 |
| **1** | 3 | male | 38.000000 | 0 | 0 | 8.6625 | 3 | 0 | 0 |
| **2** | 3 | female | 30.000000 | 1 | 1 | 24.1500 | 3 | 0 | 0 |
| **3** | 2 | female | 54.000000 | 1 | 3 | 23.0000 | 3 | 1 | 0 |
| **4** | 2 | male | 40.000000 | 0 | 0 | 13.0000 | 3 | 0 | 0 |

encoding same for the sex column

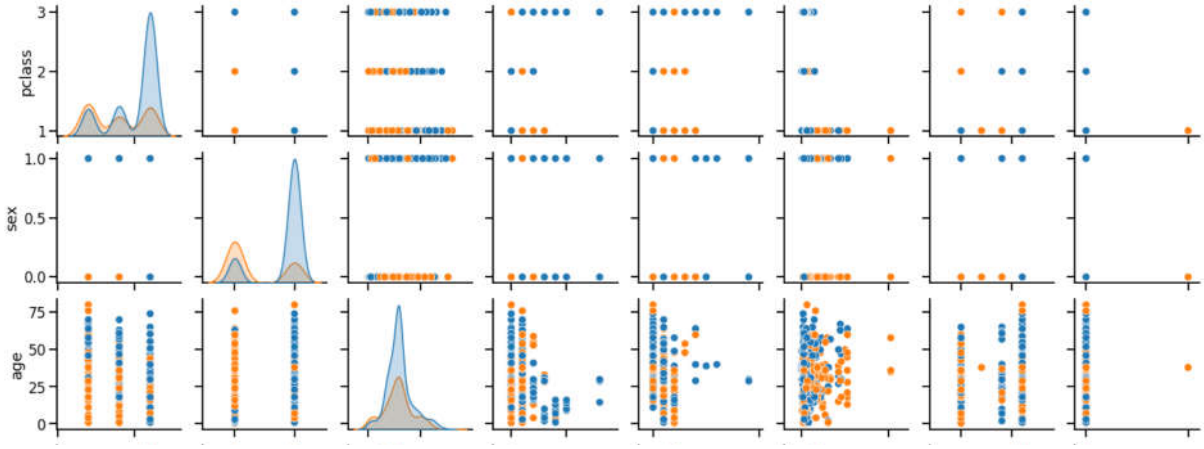
```
data['sex'] = le.fit_transform(data['sex'])
data.head()
```

| | pclass | sex | age | sibsp | parch | fare | embarked | survived | null_embarked |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 3 | 0 | 28.858401 | 0 | 0 | 7.7333 | 2 | 1 | 0 |
| **1** | 3 | 1 | 38.000000 | 0 | 0 | 8.6625 | 3 | 0 | 0 |
| **2** | 3 | 0 | 30.000000 | 1 | 1 | 24.1500 | 3 | 0 | 0 |
| **3** | 2 | 0 | 54.000000 | 1 | 3 | 23.0000 | 3 | 1 | 0 |
| **4** | 2 | 1 | 40.000000 | 0 | 0 | 13.0000 | 3 | 0 | 0 |

Comparing these independent features against the target feature

```
sns.set_context('talk')
sns.pairplot(data, hue = 'survived')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:306: UserWarning: Dat
  warnings.warn(msg, UserWarning)
<seaborn.axisgrid.PairGrid at 0x7f519831c6d0>
```




From the plots, I can guess that Fare might have a relationship whether the passenger has survived,that is, more affluent passengers will tend to be rescued compared to their poorer counterparts, but this is just an initial guess. We might need to do hypothesis test to see whether this relationship is significant.

Key Findings From the preliminary EDA analysis, here's my quick findings from the data:

- Based on the plot, the Age,SibSp, Pclass and Fare features (once cleaned) have a clear relationship with our target variable. For example, we can see that passenger class 3 have less chance of surviving compared to the other two classes.
- Male seems to have lesser chance of surviving compared to female (could be that women and children were prioritized during the rescue)
- Before we build models, we can do more feature engineering, such as dividing the Age into different categories and perform one-hot encoding on other categorical variables, though this might increase the dimensionality of our data.
- Since we are not going to do any modelling at this stage, we can stop the pre-processing over here and move on to Hypothesis Testing.

## Hypothesis Testing

Based on the data and preliminary analysis, I can formulate three hypotheses: 1) The socio economic class of the people affected their survival rate 2) The sex of the people affected their survival rate 3) The age of the people affected their survival rate

From the three hypotheses, I would like to pick the first one and test whether the effect we are seeing is significant or simply randomness/bias in the data.

Here's my Null and Alternative Hypothesis :

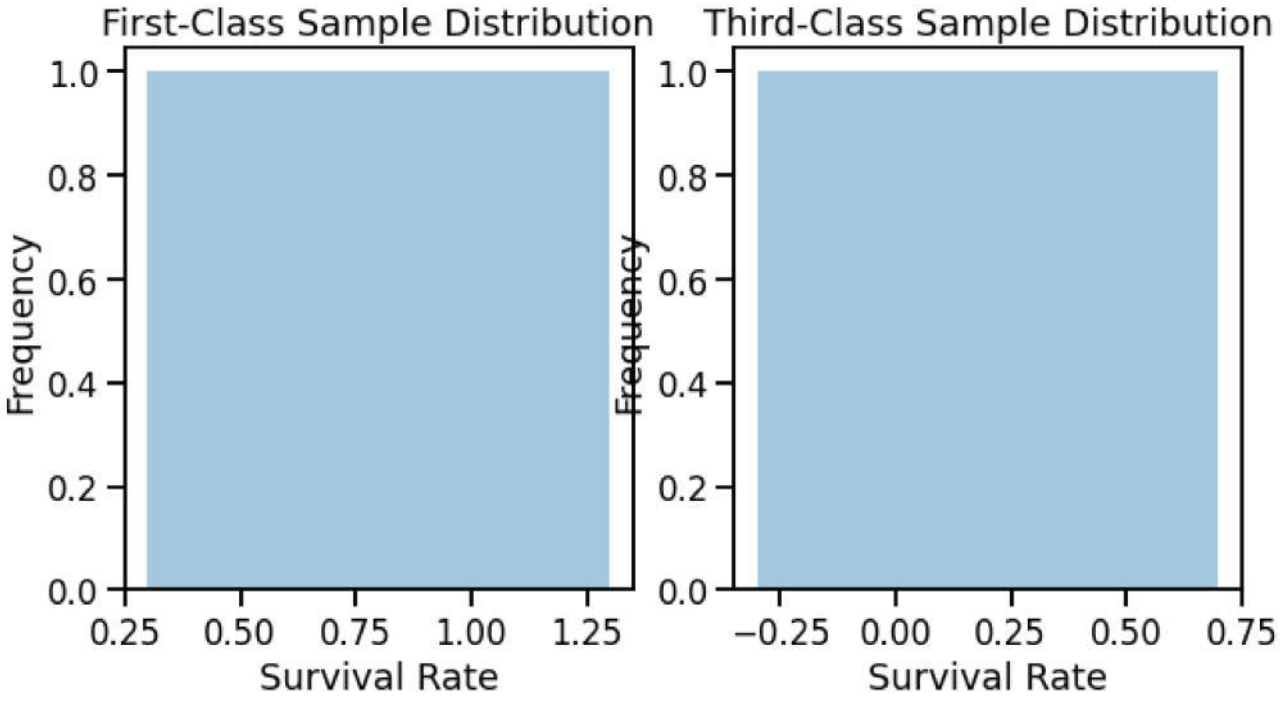
- Null Hypothesis : The socio-economic class of the people have no effect on the survival rate.
- Alternative Hypothesis : The socio-economic class of the people have an effect on their survival rate.

Now we need to compare the first class and third class passengers, to do that we need to look at the distributions of these classes and see whether it follows the Normal distribution. I would take a sample of 100 means for each population, and according to the central limit theorem, our two sample populations should be approximately normally distributed:

```
first_class_sample = np.array([np.mean(data[data['pclass'] == 1].sample(20)['survived'].va
third_class_sample = np.array([np.mean(data[data['pclass'] == 3].sample(20)['survived'].va

plt.subplots(1, 2, figsize = (10, 5))
plt.subplot(1,2, 1)
sns.distplot(first_class_sample)
plt.title("First-Class Sample Distribution")
plt.xlabel("Survival Rate")
plt.ylabel("Frequency")
plt.subplot(1, 2, 2)
sns.distplot(third_class_sample)
plt.title("Third-Class Sample Distribution")
plt.xlabel("Survival Rate")
plt.ylabel("Frequency")
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning:
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:306: UserWarning: Dat
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning:
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:306: UserWarning: Dat
  warnings.warn(msg, UserWarning)
```




Now I would calculate the Z-score and P-value. I would set the significance level at 0.05.

```
import scipy.stats as st
effect = np.mean(first_class_sample) - np.mean(third_class_sample)
sigma_first = np.std(first_class_sample)
sigma_third = np.std(third_class_sample)
sigma_difference = np.sqrt((sigma_first**2)/len(first_class_sample) + (sigma_third**2)/len
z_score = effect / sigma_difference
st.norm.sf(abs(z_score))*2
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: RuntimeWarning: divic
```

```
0.0
```

The P-value is very small at 0, much lower

1. List item
2. List item

than our 5% significance level. Therefore, we can reject the null hypothesis and say that the socio-economic status of the passengers have a significant correlation with their survival rate. Note that I do not claim that the socie-economic has a causal link with the target variable, only show that they have a relationship, and we have to dig deeper before we can claim that one caused the other or there is another factor at play.

## Conclusion and Way Forward

Titanic dataset is a good dataset for a quick start on data science analysis. From my exploratory data analysis, I can conclude that this dataset is quite a good quality, as only the missing values had to be dealed with, and after a quick imputation, we can see the relationships between the variables. At 850 rows, it is also a rather small dataset but would suffice for a good understanding of data science workflow. Though if we really want to dig deep and build a good predictive model, a much larger dataset would help, though this might come at the expense of more noise in the data (i.e. missing values, outliers, etc)

The next step in analysing this data would be to take a thorough look at all the variables, their distributions, and build a correlation plot. Once we have taken a look at the correlation, we can do feature selection and additional feature engineering and proceed to our modelling. We would need to build a predictive model and see whether the features we used is sufficient to accurately predict whether the passenger had survived.

## Final Note

Dear reviewers, Thank you for taking the time to review my assignment. All of your feedbacks and criticisms are incredibly valuable in my learning journey and I hope all of us could achieve

our desired goals in our data science journey. Take care