

**AN**  
**INTERNSHIP PROJECT REPORT ON**  
**Titanic Passengers' Survival Analysis**  
**A Project Report for Internship Programme**

*Submitted by*  
**ANKITA BASAK**  
**ANINDYA GIRI**  
**ARITRA DATTA**  
**SHEETAL GUPTA**

*in partial fulfillment for the award of the degree of*  
**Business Analytics**  
**in**  
**MBA**



**At**

**Euphoria GenX**



## DECLARATION

I undersigned, hereby declare that the project titled “**Titanic Passengers’ Survival Analysis**” submitted in partial fulfillment for the award of Degree of *Master of Business Administration* of *MAKAUT* is a bonafide record of work done by me under the guidance of *Animesh Ojha*. This report has not previously formed the basis for the award of any degree, diploma, or similar title of any University.

14/10/2021

Anindya Giri

## BONAFIDE CERTIFICATE

Certified that this project work was carried out under my supervision  
Development of a feature-rich, **Titanic Passengers' Survival Analysis**  
is the bonafide work of

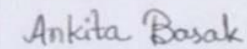
**Student Name**

**Signature**

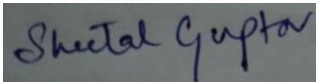
ANINDYA GIRI



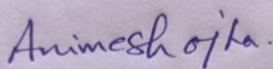
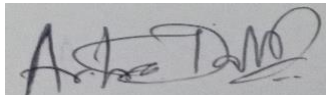
ANKITA BASAK



SHEETAL GUPTA



ARITRA DATTA



-----  
SIGNATURE

Name :

**ANIMESH OJHA**

**MCKVIE**  
**MBA in Business Analytics**



**CERTIFICATE**

*This is to certify that the report titled ‘Titanic Passengers’ Survival Analysis’ being submitted by **Anindya Giri, (11610120001)** in partial fulfilment of the requirements for the award of the Degree of Master of Business Administration, is a bonafide record of the project work done by **Anindya Giri ,Ankita Basak, Aritra Datta and Sheetal Gupta** of department of MBA, MCKVIE.*

**Name of guide – ANIMESH OJHA**  
**Designation -**  
**Director -**

## ACKNOWLEDGEMENT

Through this acknowledgement we express our sincere gratitude towards all those people who helped us in this project, which has been a learning experience.

This space wouldn't be enough to extend our warm gratitude towards our project guide **Mr. Animesh Ojha** for his efforts in coordinating with our work and guiding in right direction.

It would be injustice to proceed without acknowledging those vital supports We received from our beloved classmates and friends, without whom would have been half done.

We would like to thank **Mr. Diptayan Bhattacheryya**, for helping us to get such an excellent opportunity.

We also use this space to offer our sincere love to our parents and all others who had been there, helping us walk through this work.

**Anindya Giri**  
**Ankita Basak**  
**Aritra Datta**  
**Sheetal Gupta**

# List of Tables

Table No.	Title of the Table	Page No.
1	Abstract	10
2	Introduction	11
3	Research Methodology	15
4	Dataset (Titanic Passengers' Survival Analysis)	18
5	Data Analysis	23
6	Findings & Discussion	56
7	Implementation	58
8	Conclusions	59
9	References	60

## List of Figures

<b>Fig. No.</b>	<b>Title of the Figure</b>	<b>Page No.</b>
1	RESEARCH DESIGN FLOWCHART	16
2	COUNTPLOT FOR SURVIVED AND NOT SURVIVED	25
3	COUNTPLOT FOR SURVIVED, SEX, PCLASS, EMBARKED	26
4	PLOTTING AVERAGE SURVIVAL RATE BASED ON SEX	27
5	PLOTTING DISTPLOT FOR AGE AND FARE	28
6	PLOTTING DISTPLOT FOR FARE AFTER NORMALIZATION AND STANDARDIZATION	29
7	PLOTTING COUNTPLOT OF DIFFERENT PORT OF EMBARKATION WITH RESPECT TO 'PCLASS'	39
8	PLOTTING NORMALIZED CONFUSION MATRIX	49
9	LEARNING CURVE	53

## List of Symbols/ Abbreviations

Symbol/ Abbreviation	Explanation	Page No.
pd	import pandas as pd (for data loading, cleaning, analysis and manipulation)	24
np	import numpy as np (for mathematical Calculations)	24
plt	import matplotlib.pyplot as plt (data visualization)	24
sns	import seaborn as sns (data visualization)	24
ax	Axis	28
ss	StandardScaler	29
rb	RobustScaler	29
mm	MinMaxScaler	29
qcut	Quantile-cut	41
gs	GridSearchCV	44



# TABLE OF CONTENTS

Sl No.	Topics	Page No
<b>1</b>	<b>ABSTRACT</b>	<b>10</b>
<b>2</b>	<b>INTRODUCTION</b>	<b>11</b>
2.1	BACKGROUND OF THE STUDY	12
2.2	NEED AND SIGNIFICANCE OF THE STUDY	13
2.3	STATEMENT OF PROBLEM	13
2.4	OBJECTIVES OF THE STUDY	13
2.5	SCOPE OF THE STUDY	13
2.6	LIMITATIONS OF THE STUDY	14
2.7	ORGANIZATION OF THE REPORT	14
<b>3</b>	<b>RESEARCH METHODOLOGY</b>	<b>15</b>
3.1	OBJECTIVES	15
3.2	HYPOTHESIS	15
3.3	RESEARCH DESIGN	16
3.4	SOFTWARE REQUIREMENT	17
3.5	TOOLS DESCRIPTION	17
<b>4</b>	<b>DATASET (Titanic Passengers' Survival Analysis)</b>	<b>18</b>
4.1	SOURCES OF DATA	18
4.2	PRIMARY AND SECONDARY DATA	19
4.3	POPULATION	20
4.4	SAMPLE DESIGN	20
4.5	SAMPLING METHOD	20
4.6	METHOD OF DATA COLLECTION	21
4.7	DRAFTING A QUESTIONNAIRE	21
4.8	DATA ANALYSIS TECHNIQUES	22
<b>5</b>	<b>DATA ANALYSIS</b>	<b>23</b>
5.1	MACHINE LEARNING PROJECT: TITANIC PASSENGERS' SURVIVAL ANALYSIS	24
5.2	VISULIZATION & ANALYSIS	25
5.3	HYPERPARAMETER TUNING	44
5.4	MODEL ESTIMATION AND EVALUATION	45
5.5	CONFUSION MATRIX	47
5.6	CLASSIFICATION REPORT	50
5.7	LEARNING CURVE	52
5.8	PREDICTIONS FOR TEST DATASET	54
<b>6</b>	<b>FINDINGS (DISCUSSION)</b>	<b>56</b>
<b>7</b>	<b>IMPLEMENTATION</b>	<b>58</b>
<b>8</b>	<b>CONCLUSIONS</b>	<b>59</b>
<b>9</b>	<b>REFERENCES</b>	<b>60</b>

# **CHAPTER-1**

## **ABSTRACT**

With the greater development of technology and automation human history is predominantly updated. The technology movement shifted from large mainframes to PCs to cloud when computing the available data for a larger period. This has happened only due to the advent of many tools and practices, that elevated the next generation in computing. A large number of techniques has been developed so far to automate such computing. Research dragged towards training the computers to behave similar to human intelligence. Here the diversity of machine learning came into play for knowledge discovery. Machine Learning (ML) is applied in many areas such as medical, marketing, telecommunications, and stock, health care and so on. This paper presents reviews about machine learning algorithm foundations, its types and flavours together with Python scripts possibly for each machine learning techniques.

## CHAPTER-2

### INTRODUCTION

ML denotes to the methods tangled in distributing through massive facts in the greatest intellectual way to arise better understandings. ML algorithms are defined to be culturing an objective function (f) which better draws input identifier (g) to an output identifier (h) as in equation,  $h = f(g)$

This future output prediction is not that much easier to do manually. Hence an automated system is expected to do the process ]. Thus use of machine learning algorithms come into the scene. For every new input (g) the output (h) is predicted genuinely using machine learning algorithms. This state is said to be predictive molding/predictive analytics. The major operation is to assess the most possible predictions with the present data. Each data is segregated as training set and testing set as in figure

#### **Five basic steps for a ML task:**

1. Data accumulation: Data gathered from various sources are used for analysis.
2. Data preprocessing: Before getting into the actual processing of data, preprocessing is mandatory. This step is used to noise or other unwanted data from the gathered data.
3. Prototype training: This step contains selecting the suitable algorithm and depiction of data in a pattern (model) format. The preprocessed data is often divided into two parts namely training and testing data.
4. Pattern evaluation: In this step, the resultant pattern is validated for its correctness. However substantial time is required in data gathering and training.
5. Performance enrichment: This step involves picking another different pattern with better efficiency.

Despite any model/pattern, the above said five steps are mandatory to configure the ML technique.

Types of machine learning algorithms—scenario based

#### **Supervised learning (SL) or prognostic models**

This is used to assess the upcoming result with the help of chronological data . These models are instructive as much concentration is emphasized in training phase . For instance, SL is applied if a selling firm wishes to find its customers list. It could also be used in prediction of earthquakes, cyclones etc. to determine the Insurance credit. Few examples of these prediction algorithms are: multiple linear regression logistic regression NaiveBayes, DecisionTrees

#### **Unsupervised learning (UL) or evocative models UL**

It is suitable to train vivid models with no target and no sole feature is significant compared to one another . For instance, UL is applied in case if a vender desires to find which product does the customer buys frequently. Moreover, in medicinal business, UL may be applied to envisage

the diseases that may prone to occur laterally with diabetes. Few examples of UL based algorithms are, Simple K- means clustering

### **Reinforcement learning (RL)**

RL is applied when the system is trained to yield decisions automatically with the business requirements only with a sole motto to exploit better effectiveness (performance) . The underlying idea is a software agent is trained in an environment for problem solving. This repeated learning procedure promises lower human proficiency thus saving human effort [6]. One best example of RL algorithm is Markov Decision Process

The goal of the project was to predict the survival of passengers based off a set of data. We used Kaggle competition "Titanic: Machine Learning from Disaster" (see <https://www.kaggle.com/c/titanic/data>) to retrieve necessary data and evaluate accuracy of our predictions. The historical data has been split into two groups, a 'training set' and a 'test set'. For the training set, we are provided with the outcome (whether or not a passenger survived). We used this set to build our model to generate predictions for the test set. For each passenger in the test set, we had to predict whether or not they survived the sinking. Our score was the percentage of correct predictions.

### **Work Plan**

- 1) Learn programming language Python
- 2) Learn to use SciKit-Learn library in Python, including a Using Linear Regression algorithm and classification algorithm.
- 3) Performing Feature Engineering, applying machine learning algorithms, and analyzing results

## **2.1 BACKGROUND OF THE STUDY**

Titanic disaster occurred about 100 years back but still it attracts the researchers to understand and study that how some passengers survived and others perished. In this work, the characteristics of the passengers will be identified and the relationship of survival chance from the disaster is found. Feature engineering techniques will be performed, where the alphabetic values will be changed to numeric values, the family size will be calculated. Also, we will extract the title from the name, and deck label from ticket number. Classification is done using Decision tree machine learning classification algorithm using two classes which are survived and not survived. R programming has been used for its implementation. Clustering is performed using KMeans machine learning algorithm. Its implementation has been done using Python programming.

## **2.2 NEED AND SIGNIFICANCE OF THE STUDY**

The Titanic was competitively designed to be the best, but failed--the sea winning the battle. The sinking of the Titanic in the early hours of April 15, 1912, remains the quintessential disaster of the century. A total of 1,517 souls--men, women and children - lost their lives ( only 711 survived). The Titanic sunk because it was built to be the best, and thus caused people in charge to be full of pride, which caused the Marconi telegraph distress signals to be ignored. This analysis is been done so that we can learn from the Titanic's failure and analyse the causes and effects of the disaster.

## **2.3 STATEMENT OF THE PROBLEM**

The RMS Titanic, a luxury steamship, sank in the early hours of April 15, 1912, off the coast of Newfoundland in the North Atlantic after sideswiping an iceberg during its maiden voyage. Of the 2,240 passengers and crew on board, more than 1,500 lost their lives in the disaster. There was a huge loss which passengers faced during Titanic disaster. To come out from this type of problem the case study of analysis on Titanic disaster dataset is needed.

## **2.4 OBJECTIVE OF THE STUDY**

### **Feature Engineering**

Since the data can have missing fields, incomplete fields, or fields containing hidden information, a crucial step in building any prediction system is Feature Engineering. For instance, the fields Age, Fare, and Embarked in the training and test data, had missing values that had to be filled in. The field Name while being useless itself, contained passenger's Title (Mr., Mrs., etc.), we also used passenger's surname to distinguish families on board of Titanic. Below is the list of all changes that has been made to the data- Extracting Title from Name. In this project, final destiny of all people will be predicted given explanatory variables that cover approx. all aspects of travelers. The goal of this project is to create a classification model that is able to accurately estimate the survival of the people in the ship. We can use this model if any same kind of accident happen in future to predict the survival rate.

## **2.5 SCOPE OF THE STUDY**

The study aims to analyze the number of people survived, gender and survival rate. The goal of this project is to create a regression model that is able to accurately estimate the losses occurred during Titanic disaster.

## **2.6 LIMITATION OF THE STUDY**

There is no guarantee that the data will be available in time nor contains the exact requested list of features. Thus, there might be a risk that the access will be denied or delayed. If so, the study will be accomplished based only on the public dataset. Moreover, this study will not cover all regression algorithms. Also the data collected is too old.

## **2.7 ORGANIZATION OF THE REPORT**

The dataset in this section is the so-called Titanic dataset. It contains data concerning survival analysis of passengers.

# CHAPTER-3

## RESEARCH METHODOLOGY

### 3.1 OBJECTIVES:

The objective is to perform exploratory data analytics on various information in the dataset available and to know effect of each field on survival of passengers by applying analytics between every field of dataset with "Survival" field. The predictions are done for newer data sets by applying machine learning algorithm. The data analysis will be done on applied algorithms and accuracy will be checked. Different algorithms are compared on the basis of accuracy and the best performing model is suggested for predictions. The aim of this study is to get as reliable results as possible from the raw and missing data by using machine learning and feature engineering methods. In the test data, all the attributes of train data will be mentioned except the survived column. So, the objective is to get the survival status of each passenger in the test data.

The goal of this research paper is to correctly predict who would survive the Titanic given a set of demographic information. A predictive model using passenger data was built so people's genders, their ages, what class of ticket they belonged from, and their socio-economic class, all contributed to whether they would be lucky enough to survive or tragically perish on the Titanic.

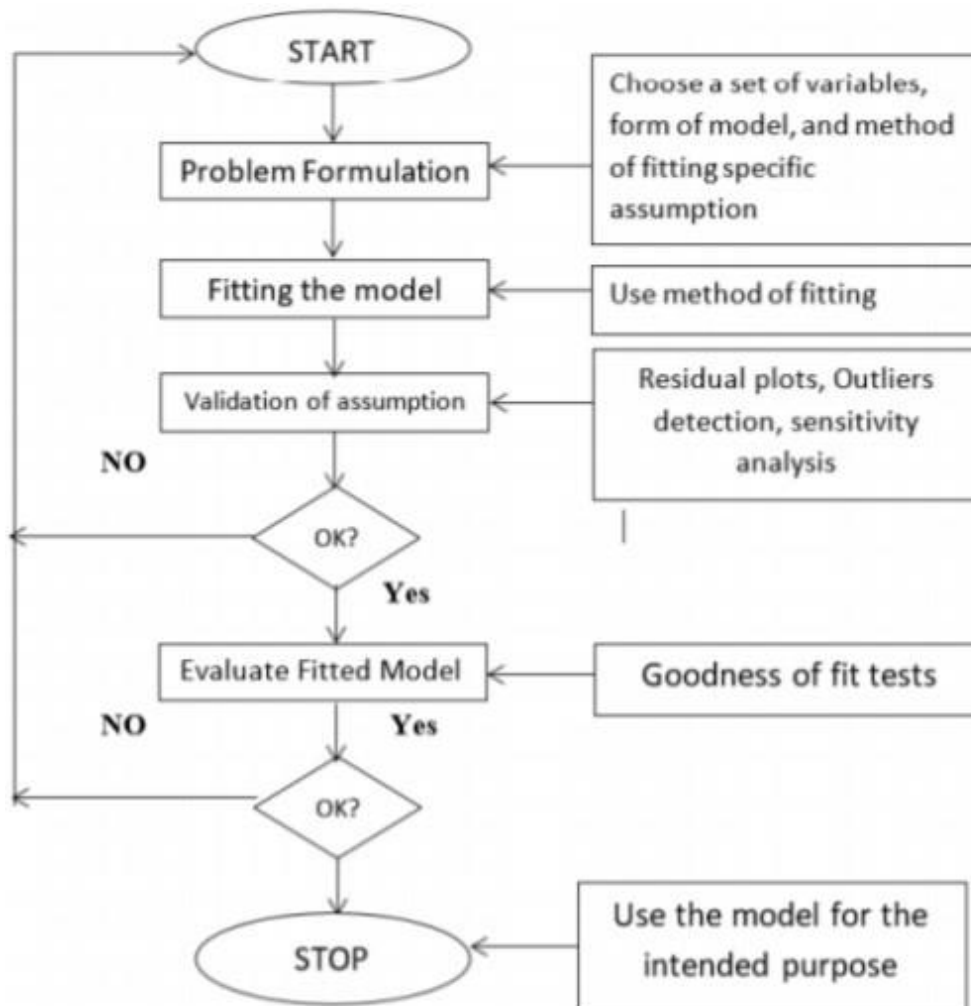
### 3.2 HYPOTHESIS:

Obtaining valuable results from the raw and missing data by using machine learning and feature engineering methods is very important for knowledge-based world. In this paper, we have proposed models for predicting whether a person survived the Titanic disaster or not. The accuracy of the model is evaluated using "confusion matrix". A confusion matrix is a table layout that allows to visualize the correctness and the performance of an algorithm. The dataset found on the Kaggle website has two perspectives website has two perspectives. One is a training data and the other is a testing data. The objective of the training data is to create a model which will help in predicting the outcomes of the test data. For the purpose of this research paper, the training data from the Kaggle website will be divided into two parts using three different ratios for training and creating the model and then predicting and the testing data from the Kaggle website will not be used. There will be application of different techniques for predicting whether a person survived the Titanic disaster or not. The data analysis will then be done and the prediction outcomes will be checked for accuracy. The accuracy will then be compared in order to suggest the better performing algorithm with respect to used dataset. With respect to the Titanic dataset study, we have a 2 different studies of research hypothesis.

- 1) Simple Hypothesis: - the study of the relationship between a single dependent variable and a single independent variable visualizes correctly and make better sense to calculate the outliers and threshold
- 2) Complex hypothesis: - In our Titanic dataset it predicts the relationship between two or more independent and dependent variables correctly by having different threshold value.

### 3.3 RESEARCH DESIGN:

To pick a specific model, there is a step-by-step technique. It is important to determine if the model to be implemented is appropriate for the given issue. In the below diagram, flow of processes can be seen acting in accordance here.



Feature engineering is the most important part of data analytics process. It deals with, selecting the features that are used in training and making predictions. In feature engineering the domain knowledge is used to find features in the dataset which are helpful in building machine learning model. It helps in understanding the dataset in terms of modelling. A bad feature selection may lead to less accurate or poor predictive model. The accuracy and the predictive power depend on the choice of correct features. It filters out all the unused or redundant features.



### 3.4 SOFTWARE REQUIREMENT

This project uses the following tools for its implementation:

- 1) Python: the programming language used to implement this project
- 2) Jupyter notebook: used to provide an interactive environment for python, for the implementation of this project
- 3) Scikit-Learn: used to implement the several machine learning models used in this project as as view their accuracies.
- 4) Matplotlib: used to plot graphs for us to understand the trend in accuracy/performance of the various machine learning algorithm implemented.
- 5) Seaborn: use for visualization of the statistical behaviour of data set.
- 6) Pandas: - used to analyse and clean our dataset,
- 7) Numpy: for mathematical Calculations.

### 3.5 TOOLS DESCRIPTION

The various tools used in this project are described below.

- 1) **Python:** Python is an interpreted, high level programming language created by Guido van Rossum in 1991 it emphasizes on code readability by using whitespace to terminate statements and blocks. Its ease of use and syntax makes it a language of preference for data analysis
- 2) **Jupyter Notebook:** It is a web application that provides an integrated development environment for python using this one can share documents that contain equations, visualizations such as graphs test as a live for this reason it is highly used tool for the purposes of data analysis
- 3) **Scikit-Learn:** It is a python package used for data modelling. It provides a number of supervised and unsupervised machine learning models. Scikit-learn makes it extremely simple to train models with simple function calls on the input data being all that is needed
- 4) **Matplotlib:** Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension
- 5) **Pandas:** Pandas is an open-source Python package that is most widely used for data science/data analysis and machine learning tasks. It is built on top of another package named NumPy, which provides support for multi-dimensional arrays .
- 6) **Numpy:** NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices

# CHAPTER 4

## DATASET (TITANIC PASSENGERS' SURVIVAL ANALYSIS)

### 4.1 SOURCES OF DATA:

The following is a representation of the test dataset provided in a comma separated value (CSV) format from Kaggle and 891 rows of data (a subset of the entire passenger manifest). The Titanic dataset consist of a training set that includes 891 passengers and a test set that includes 418 passengers which are different from the passengers in training set. A description of the features is given in Table I.

TABLE I  
NUMBER OF FEATURES IN THE DATASET

Feature	Value of Feature	Feature Characteristic
PassengerId	1-891	Integer
Survived	0,1	Integer
Pclass	1-3	Integer
Name	Name of passengers	Object
Sex	Male, female	Object
Age	0-80	Real
SibSp	0-8	Integer
Parch	0-6	Integer
Ticket	Ticket number	Object
Fare	0-512	Real
Cabin	Cabin number	Object
Embarked	S, C, Q	Object

## Data Description

### VARIABLE DESCRIPTIONS:

- pclass      Passenger Class
  - (1 = 1st; 2 = 2nd; 3 = 3rd)
- survival      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
- cabin      Cabin
- embarked      Port of Embarkation  
(C = Cherbourg; Q = Queenstown; S = Southampton)
- home.dest      Home/Destination

- SPECIAL NOTES:  
Pclass is a proxy for socio-economic status (SES)  
1st ~ Upper; 2nd ~ Middle; 3rd ~ Lower

Age is in Years; Fractional if Age less than One (1)  
If the Age is Estimated, it is in the form xx.5

With respect to the family relation variables (i.e. sibsp and parch)  
some relations were ignored. The following are the definitions used  
for sibsp and parch.

Sibling: Brother, Sister, Stepbrother, or Stepsister of Passenger Aboard Titanic  
Spouse: Husband or Wife of Passenger Aboard Titanic (Mistresses and Fiances Ignored)  
Parent: Mother or Father of Passenger Aboard Titanic  
Child: Son, Daughter, Stepson, or Stepdaughter of Passenger Aboard Titanic

Other family relatives excluded from this study include cousins,  
nephews/nieces, aunts/uncles, and in-laws. Some children travelled  
only with a nanny, therefore parch=0 for them. As well, some  
travelled with very close friends or neighbors in a village, however,  
the definitions do not support such relations.

## 4.2 PRIMARY AND SECONDARY DATA:

The data we used for our study was provided on the Kaggle website. We were given 891 passenger samples for our training set and their associated labels of whether or not the passenger survived.

For each passenger, we were given his/her passenger class, name, sex, age, number of siblings/spouses aboard, number of parents/children aboard, ticket number, fare, cabin embarked, and port of embarkation (as shown in Table 1). For the test data, we had 418 samples in the same format. The dataset is not complete, meaning that for several samples, one

or many of fields were not available and marked empty (especially in the latter fields – age, fare, cabin, and port). However, all sample points contained at least information about gender and passenger class. After importing the packages and libraries to make writing the program simpler, the data from the seaborn package is needed to be loaded and thereby few rows are to be printed. The data was then required to be analyzed by receiving counts of it. Following the receipt of a count of the dataset's rows and column, it is helpful to keep a record of every rows of the passengers who were aboard the ship. On the other hand, the columns are regarded as every commuter's assigned data. Theoretically, there were 891 passengers/rows and 15 data points/columns in the data set. Meanwhile, statistics like mean, count, standard deviation, etc. was received on the dataset.

### **4.3 POPULATION:**

Based on data set obtained from Kaggle, the purpose is about survival prediction of the titanic. We will split the historical data set that we obtained from kaggle into two groups – train set and test set. They are the two different files in csv format. The train data will be an input for the training model. This data will be used to build the model for the generations of predictions for the test data. There can be many flaws in the train set because of which training of the model on the basis of such data is not possible or can leads to an incorrect output. It is possible that in the csv file of the train set, there will be few blank cells where no values is mentioned. For an example, if the age of a person is not mentioned then this a flaw in the train data set. Hence, it has to be filled with appropriately calculated age value before training phase. This is feature engineering that we do to make the train data set perfect for giving it as an input for training the model.

### **4.4 SAMPLE DESIGN:**

Clustering the data based upon classifications and use of clustering analysis simple associations may be understood from the data. While an association might be strong through this analysis, the true cause and effect cannot be concluded. Similarly, there are modifications that needs to be performed on training data set. The prediction that whether or not the titanic passengers survived the sinking will be found for each passenger in the test set. Different machine learning techniques will be applied on the data set available. By the results obtained from those techniques we will be able to understand the statistics more clearly. Algorithms used includes feature engineering, decision tree visualization, random forest etc.

### **4.5 SAMPLING METHOD:**

Feature Engineering is required for modifying the train data set because there can be some empty spaces, title for some passengers will be missing, department for some crew members may not be mentioned. Visualization of relationship between the family size and survival is required. Hence for all such errors this technique has be applied before applying the machine learning techniques for training the model. The best way for doing this using programming for data extraction and analyzing using graphs. Classification Technique is used to determine that for which class or category a given observation will belong to. There are many algorithms related to classification but for this work, the decision tree algorithm is employed. All nodes in the tree will indicate either the passenger survived or not survived using 0 and 1 respectively. At the first level, the relation between the child nodes and the root node will indicate the gender of passenger. Further, the relation will be based on age, Passenger class and family

size. Accuracy for the resulted survival status in the test data has to be calculated by comparing with the survival status of passengers in train data. Clustering Technique is used to group a set of observations in such a way that observation of similar types belongs to a group. There can be many groups defined while clustering. This technique has also to be implemented for examining passenger's survival in titanic disaster.

## **4.6 METHOD OF DATA COLLECTION:**

Before applying any type of data analytics on the dataset, the data is first cleaned. There are some missing values in the dataset which needs to be handled. In attributes like Age, Cabin and Embarked, missing values are replaced with random sample from existing age. The data is first cleaned before implementing some form of data analytics on the dataset. Here, it needs to be analyzed. It is also seen that there are a few absent data values for the column: age as it is smaller than 891. Absent values are restored with arbitrary samples. Using the age of a person, their sex, title, cabin number, Pclass, place of embarkation, fare and size of family which includes both parch as well as sibsp, the following features on the exploratory study is done. The survival column is selected as the reaction column. Such characteristics are chosen because their ideals have an effect on the survival rate. If wrong features are chosen, bad predictions may be generated. Thus, in creating an effective predictive model, function engineering works as a backbone. NaN, NAN and na are some of the columns that contain empty values. Only four columns, namely age, deck, embarked and embarked\_town have a few absent values. In an attempt to display a few unwanted columns, the next step is to have a look every column's value name as well as count. The non-redundant rows and columns should be dropped, and their rows and missing values should be removed.

## **4.7 DRAFTING A QUESTIONNAIRE**

Over the years, a lot of thought has been put into the science of the design of survey questions. Key design principles:

1. Keep the questionnaire as short as possible.
2. Ask short, simple, and clearly worded questions.
3. Start with demographic questions to help respondents get started comfortably.
4. Use dichotomous (yes | no) and multiple-choice questions.
5. Use open-ended questions cautiously.
6. Avoid using leading-questions.
7. Pretest a questionnaire on a small number of people.
8. Think about the way you intend to use the collected data when preparing the questionnaire.

## **4.8 DATA ANALYSIS TECHNIQUES**

- I. Importing all the Required modules / libraries.
- II. Data Loading
- III. Data Cleaning
- IV. Data Mining
- V. Data Visualization.
- VI. Splitting the Data
- VII. Linear Regression Methods.

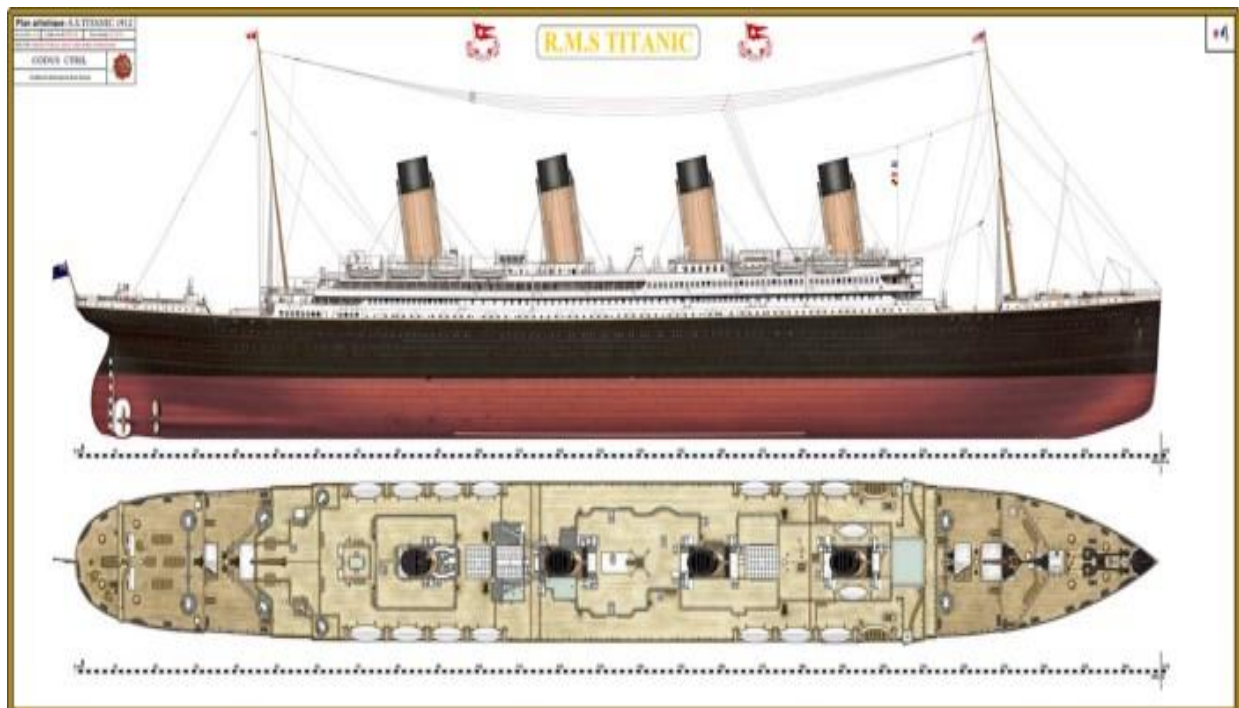
# CHAPTER-5

## DATA ANALYSIS

### 5.1 MACHINE LEARNING PROJECT: TITANIC PASSENGERS' SURVIVAL ANALYSIS

#### RMS Titanic

The RMS Titanic was a British passenger liner that sank in the North Atlantic Ocean in the early morning hours of 15 April 1912, after it collided with an iceberg during its maiden voyage from Southampton to New York City. There were an estimated 2,224 passengers and crew aboard the ship, and more than 1,500 died, making it one of the deadliest commercial peacetime maritime disasters in modern history. The RMS Titanic was the largest ship afloat at the time it entered service and was the second of three Olympic-class ocean liners operated by the White Star Line. The Titanic was built by the Harland and Wolff shipyard in Belfast. Thomas Andrews, her architect, died in the disaster.



In this project, we will develop and evaluate the performance and the predictive power of a model trained and tested on data collected from Titanic dataset.

Once we get a good fit, we will use this model to predict the chances of survival of passenger at the Titanic.

A model like this would be very valuable for analysis of past data like Titanic dataset.

## # IMPORT LIBRARIES NECESSARY FOR THIS PROJECT:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler, RobustScaler, MinMaxScaler, LabelEncoder
from sklearn.impute import KNNImputer
from sklearn.model_selection import train_test_split
```

```
#Hide all the warnings in jupyter notebook
import warnings
warnings.filterwarnings('ignore')
```

## # IMPORTING THE TITANIC DATASET:

```
train = pd.read_csv('./data/train.csv')
test = pd.read_csv('./data/test.csv')
```

## # SUMMARY OF TRAINING DATASET:

```
train.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
```



## # DESCRIPTION OF THE TRAINING DATASET:

```
train.describe()
```

	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

## # SHAPE OF DATASET:

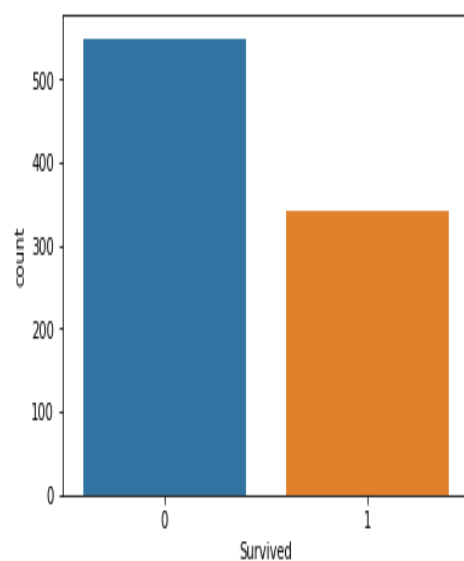
```
train.shape
```

```
(891, 12)
```

## 5.2 VISULIZATION & ANALYSIS

### # COUNTPLOT FOR SURVIVED AND NOT SURVIVED:

```
sns.countplot(train['Survived']);
```



0= NOT SURVIVED, 1= SURVIVED

In this plot we can see that number of survived is less than number of not survived.

## # IDENTIFYING THE UNIQUE NUMBER OF VALUES IN THE DATASET:

```
train.nunique().sort_values()
```

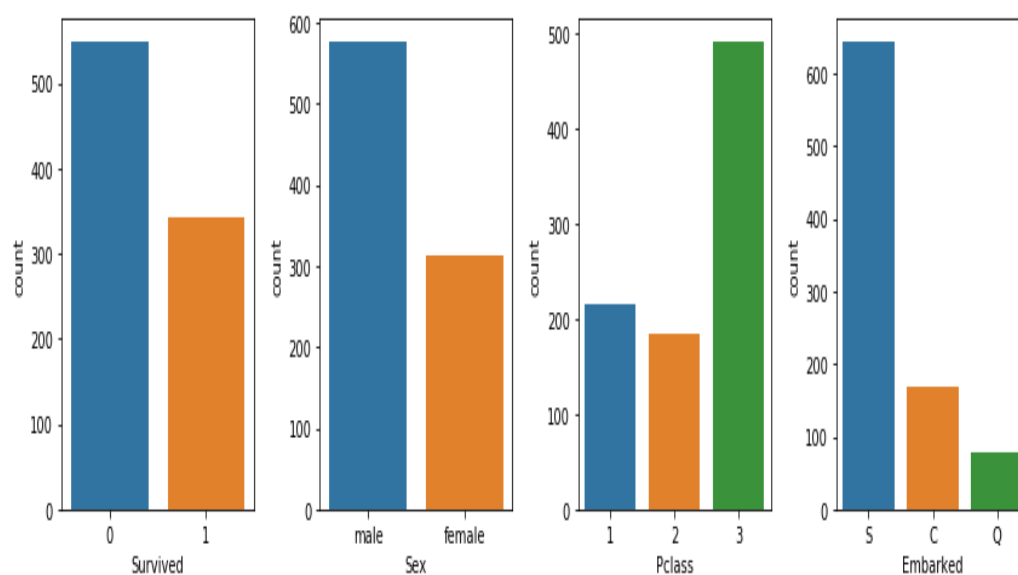
```
Survived      2
Sex            2
Pclass        3
Embarked      3
SibSp         7
Parch         7
Age           88
Cabin        147
Fare         248
Ticket       681
PassengerId  891
Name         891
dtype: int64
```

## # COUNTPLOT FOR SURVIVED, SEX, PCLASS, EMBARKED:

```
fig, ax = plt.subplots(1, 4, figsize = (12, 4)) # Making Subplots

sns.countplot(train['Survived'], ax=ax[0])
sns.countplot(train['Sex'], ax=ax[1])
sns.countplot(train['Pclass'], ax=ax[2])
sns.countplot(train['Embarked'], ax=ax[3])

plt.tight_layout() # you can use this function for clear visualization
plt.show()
```



From this plot ,  
we can see that number of male passengers is greater than that of female.  
Pclass = 3 has maximum number of passengers.  
In case of Port of Embarkation(C = Cherbourg; Q = Queenstown; S = Southampton), S has highest number of passengers.

## # FINDING AVERAGE SURVIVAL RATE BASED ON SEX:

```
sex_survived_rate = train.groupby('Sex')['Survived'].mean()  
sex_survived_rate
```

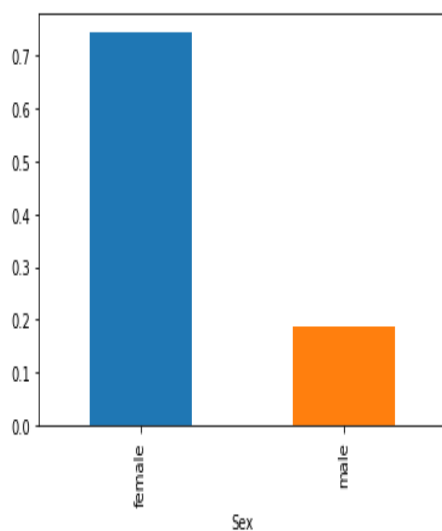
```
Sex  
female    0.742038  
male      0.188908  
Name: Survived, dtype: float64
```

Here,  
We can see that female has higher survival rate than that of male.

## # PLOTTING AVERAGE SURVIVAL RATE BASED ON SEX:

```
sex_survived_rate.plot(kind = 'bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x576c940>
```

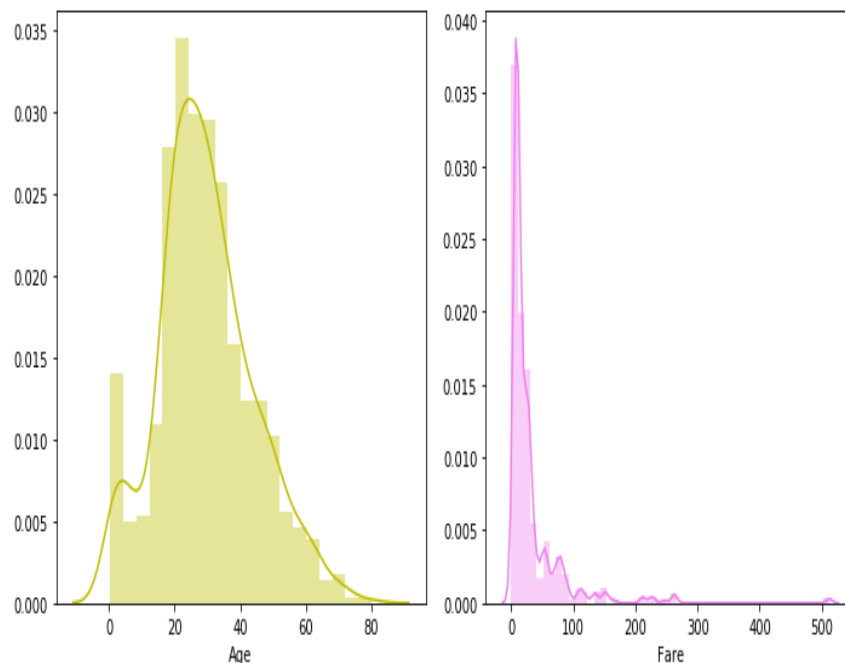


## # PLOTTING DISTPLOT FOR AGE AND FARE:

```
fig, ax = plt.subplots(1, 2, figsize = (10, 5))

sns.distplot(train['Age'].dropna(), ax=ax[0], color='y')
sns.distplot(train['Fare'].dropna(), ax=ax[1], color='violet')

plt.tight_layout()
plt.show()
```



**Normal distribution**, also known as the Gaussian distribution, is a probability distribution that is symmetric about the mean, showing that data near the mean are more frequent in occurrence than data far from the mean. In graph form, normal distribution will appear as a bell curve. If one tail is longer than another, the distribution is skewed. These distributions are sometimes called asymmetric or asymmetrical distributions as they don't show any kind of symmetry. Symmetry means that one half of the distribution is a mirror image of the other half. For example, the normal distribution is a symmetric distribution with no skew. The tails are exactly the same.

A left-skewed(negatively skewed) distribution has a long left tail. Left-skewed distributions are also called negatively-skewed distributions. That's because there is a long tail in the negative direction on the number line. The mean is also to the left of the peak.

A right-skewed(positively skewed) distribution has a long right tail. Right-skewed distributions are also called positive-skew distributions. That's because there is a long tail in the positive direction on the number line. The mean is also to the right of the peak.

Both the distplots are showing normal distribution and positively skewed distribution.

## # PLOTTING DISTPLOT FOR FARE AFTER NORMALIZATION AND STANDARDIZATION:

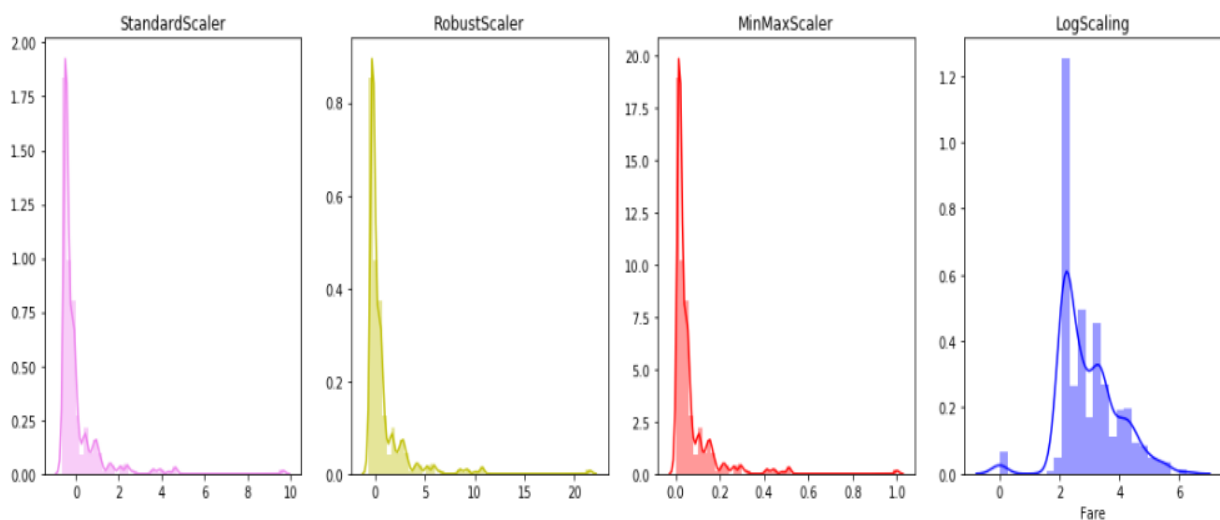
```
: ss = StandardScaler()
rb = RobustScaler()
mm = MinMaxScaler()

fare_standard = ss.fit_transform(train['Fare'].values.reshape(-1, 1))
fare_robust = rb.fit_transform(train['Fare'].values.reshape(-1, 1))
fare_minmax = mm.fit_transform(train['Fare'].values.reshape(-1, 1))

fig, ax = plt.subplots(1, 4, figsize=(15, 5))

sns.distplot(fare_standard, color='violet', ax = ax[0]).set_title('StandardScaler')
sns.distplot(fare_robust, color='y', ax = ax[1]).set_title('RobustScaler')
sns.distplot(fare_minmax, color='r', ax = ax[2]).set_title('MinMaxScaler')
sns.distplot(np.log1p(train['Fare']), color='b', ax = ax[3]).set_title('LogScaling')

plt.tight_layout()
plt.show()
```



By using RobustScaler(), we can remove the outliers and then use either StandardScaler or MinMaxScaler for preprocessing the dataset.

Standardizing a dataset involves rescaling the distribution of values so that the mean of observed values is 0 and the standard deviation is 1. Data normalization is the organization of data to appear similar across all records and fields.

It increases the cohesion of entry types leading to cleansing, lead generation, segmentation, and higher quality data.

Numpy log1p() is a mathematical function that helps the user to calculate the natural logarithmic value of  $x+1$  where  $x$  belongs to all the input array elements. log1p is reverse of  $\exp(x) - 1$ . So the distplot of 'Fare' of train dataset becomes more prominent.

## # FINDING TITLE OF ALL THE PEOPLE INSIDE TITANIC:

```
train['Name_Title'] = train['Name'].apply(lambda x: x.split(',')[1]).apply(lambda x: x.split()[0])
train['Name_Title'].value_counts()
```

```
Mr.      517
Miss.    182
Mrs.     125
Master.   40
Dr.        7
Rev.        6
Col.        2
Major.      2
Mlle.       2
the         1
Sir.         1
Don.         1
Capt.       1
Mme.         1
Ms.          1
Lady.        1
Jonkheer.    1
Name: Name_Title, dtype: int64
```

## # FINDING AVERAGE SURVIVAL RATE WITH RESPECT TO TITLE:

```
train['Survived'].groupby(train['Name_Title']).mean()
```

```
Name_Title
Capt.      0.000000
Col.        0.500000
Don.        0.000000
Dr.         0.428571
Jonkheer.   0.000000
Lady.       1.000000
Major.      0.500000
Master.     0.575000
Miss.       0.697802
Mlle.       1.000000
Mme.        1.000000
Mr.         0.156673
Mrs.        0.792000
Ms.         1.000000
Rev.        0.000000
Sir.        1.000000
the         1.000000
Name: Survived, dtype: float64
```

Here,

We can see that ‘Lady’, ‘Mlle’, ‘Mme’, ‘Ms’, ‘Sir’, ‘the’ – these titles have highest survival rate.

## # FINDING THE AVERAGE SURVIVAL RATE WITH RESPECT TO LENGTH OF NAME:

```
train['Name_Len'] = train['Name'].apply(lambda x: len(x))
train['Survived'].groupby(pd.qcut(train['Name_Len'],5)).mean()
```

```
Name_Len
(11.999, 19.0]    0.220588
(19.0, 23.0]     0.301282
(23.0, 27.0]     0.319797
(27.0, 32.0]     0.442424
(32.0, 82.0]     0.674556
Name: Survived, dtype: float64
```

The pandas documentation describes `qcut` as a “Quantile-based discretization function.” This basically means that `qcut` tries to divide up the underlying data into equal sized bins. The function defines the bins using percentiles based on the distribution of the data, not the actual numeric edges of the bins. Here the ‘Name\_Len’ of train dataset is divided into 5 bins to find the average survival rate of each bin.

## # FINDING NUMBER OF NAMES IN DIFFERENT NAME LENGTH BINS:

```
pd.qcut(train['Name_Len'],5).value_counts()
```

```
(11.999, 19.0]    204
(23.0, 27.0]     197
(32.0, 82.0]     169
(27.0, 32.0]     165
(19.0, 23.0]     156
Name: Name_Len, dtype: int64
```

## # FINDING NUMBER OF PEOPLE IN DIFFERENT SEX:

```
train['Sex'].value_counts(normalize=True)
```

```
male    0.647587
female  0.352413
Name: Sex, dtype: float64
```

## # FINDING AVERAGE SURVIVAL RATE WITH RESPECT TO SEX:

```
train['Survived'].groupby(train['Sex']).mean()
```

```
Sex
female    0.742038
male      0.188908
Name: Survived, dtype: float64
```

## # FINDING AVERAGE SURVIVAL RATE WITH RESPECT TO AGE (NULL OR NOT NULL):

```
train['Survived'].groupby(train['Age'].isnull()).mean()
```

```
Age
False    0.406162
True     0.293785
Name: Survived, dtype: float64
```

## #FINDING AVERAGE SURVIVAL RATE WITH RESPECT TO DIFFERENT AGE GROUP:

```
train['Survived'].groupby(pd.qcut(train['Age'],5)).mean()
```

```
Age
(0.419, 19.0]    0.481707
(19.0, 25.0]    0.328467
(25.0, 31.8]    0.393701
(31.8, 41.0]    0.437500
(41.0, 80.0]    0.373239
Name: Survived, dtype: float64
```

Here,

We can see that Age group 0.419 to 19.0 has highest survival rate. Age group 19.0 to 25.0 has lowest survival rate.

## # FINDING NUMBER OF PEOPLE IN DIFFERENT AGE GROUP:

```
pd.qcut(train['Age'],5).value_counts()
```

```
(0.419, 19.0]    164
(31.8, 41.0]     144
(41.0, 80.0]     142
(19.0, 25.0]     137
(25.0, 31.8]     127
Name: Age, dtype: int64
```



Here,

We can see that Age group 0.419 to 19.0 has highest number of people. Age group 25.0 to 31.0 has lowest number of people.

### **#FINDING AVERAGE SURVIVAL RATE WITH RESPECT TO NUMBER OF SIBLINGS/SPOUSES ABOARD:**

```
train['Survived'].groupby(train['SibSp']).mean()
```

```
SibSp
0    0.345395
1    0.535885
2    0.464286
3    0.250000
4    0.166667
5    0.000000
8    0.000000
Name: Survived, dtype: float64
```

Here,

We can see that number of siblings/spouses=1 has highest survival rate.

### **# FINDING NUMBER OF SIBLINGS/SPOUSES ABOARD:**

```
train['SibSp'].value_counts()
```

```
0    608
1    209
2     28
4     18
3     16
8      7
5      5
Name: SibSp, dtype: int64
```

### **# FINDING NUMBER OF PARENTS/CHILDREN ABOARD:**

```
train['Parch'].value_counts()
```

```
0    678
1    118
2     80
5      5
3      5
4      4
6      1
Name: Parch, dtype: int64
```

## # FINDING AVERAGE SURVIVAL RATE WITH RESPECT TO NUMBER OF PARENTS/CHILDREN ABOARD:

```
train['Survived'].groupby(train['Parch']).mean()
```

```
Parch
0    0.343658
1    0.550847
2    0.500000
3    0.600000
4    0.000000
5    0.200000
6    0.000000
Name: Survived, dtype: float64
```

Here,

We can see that number of parents/children=3 has highest survival rate.

## # FINDING FIRST TEN TICKET NUMBER:

```
train['Ticket'].head(n=10)
```

```
0      A/5 21171
1      PC 17599
2  STON/O2. 3101282
3      113803
4      373450
5      330877
6      17463
7      349909
8      347742
9      237736
Name: Ticket, dtype: object
```

## # FINDING NUMBER OF TICKETS FOR DIFFERENT TICKET NUMBER LENGTH:

```
train['Ticket_Len'] = train['Ticket'].apply(lambda x: len(x))
```

```
train['Ticket_Len'].value_counts()
```

```
6    419
5    131
4    101
8     76
10    41
7     27
9     26
17    14
16    11
13     10
12     10
15     9
11     8
18     6
3       2
Name: Ticket_Len, dtype: int64
```

## # FINDING AVERAGE SURVIVAL RATE WITH RESPECT TO TICKET NUMER LENGTH:

```
train.groupby(['Ticket_Len'])['Survived'].mean()
```

```
Ticket_Len
3      0.000000
4      0.366337
5      0.618321
6      0.319809
7      0.296296
8      0.539474
9      0.192308
10     0.341463
11     0.250000
12     0.400000
13     0.400000
15     0.333333
16     0.272727
17     0.428571
18     0.000000
Name: Survived, dtype: float64
```

## # FINDING NUMBER OF PEOPLE FOR DIFFERENT FARE GROUP:

```
pd.qcut(train['Fare'], 3).value_counts()
```

```
(-0.001, 8.662]    308
(26.0, 512.329]    295
(8.662, 26.0]       288
Name: Fare, dtype: int64
```

Here,

We can see that Fare Group 0 to 8.662 has highest number of people.

## # FINDING AVERAGE SURVIVAL RATE FOR DIFFERENT FARE GROUP:

```
train['Survived'].groupby(pd.qcut(train['Fare'], 3)).mean()
```

```
Fare
(-0.001, 8.662]    0.198052
(8.662, 26.0]      0.402778
(26.0, 512.329]    0.559322
Name: Survived, dtype: float64
```

Here,

We can see that Fare Group 26.0 to 512.329 has highest number of people survived.

## # FINDING CROSSTAB

```
pd.crosstab(pd.qcut(train['Fare'], 5), columns=train['Pclass'])
```

Pclass	1	2	3
Fare			
(-0.001, 7.854]	6	6	167
(39.688, 512.329]	0	24	160
(7.854, 10.5]	0	80	92
(10.5, 21.679]	64	64	52
(21.679, 39.688]	146	10	20

The crosstab method is used to compute a simple cross-tabulation of two (or more) factors. By default, computes a frequency table of the factors unless an array of values and an aggregation function are passed. Here cross-tabulation is done between 'Fare' and 'Pclass'. 'Fare' is divided into some groups and cross-tabulation is done for three types of 'Pclass'.

## # FINDING FIRST LETTER OF DIFFERENT CABINS AND NUMBER OF PEOPLE FOR DIFFERENT CABINS:

```
train['Cabin_Letter'] = train['Cabin'].apply(lambda x: str(x)[0])
```

```
train['Cabin_Letter'].value_counts()
```

```
n    687
C     59
B     47
D     33
E     32
A     15
F     13
G      4
T      1
Name: Cabin_Letter, dtype: int64
```

Here,  
We can see the first letter of cabin 'n' has highest number of people.

## # FINDING AVERAGE SURVIVAL RATE OF PASSENGERS WITH RESPECT TO FIRST LETTER OF DIFFERENT CABINS:

```
train['Survived'].groupby(train['Cabin_Letter']).mean()
```

```
Cabin_Letter
A    0.466667
B    0.744681
C    0.593220
D    0.757576
E    0.750000
F    0.615385
G    0.500000
T    0.000000
n    0.299854
Name: Survived, dtype: float64
```

## # FINDING NUMBER OF PEOPLE IN DIFFERENT CABIN NUMBERS:

```
train['Cabin_num'] = train['Cabin'].apply(lambda x: str(x).split(' ')[-1][1:])
train['Cabin_num'].replace('an', np.NaN, inplace = True)
train['Cabin_num'] = train['Cabin_num'].apply(lambda x: int(x) if not pd.isnull(x) and x != '' else np.NaN)
```

```
pd.qcut(train['Cabin_num'],3).value_counts()
```

```
(65.667, 148.0]    67
(1.999, 28.667]    67
(28.667, 65.667]   66
Name: Cabin_num, dtype: int64
```

## # FINDING AVERAGE SURVIVAL RATE FOR DIFFERENT CABIN NUMBER:

```
train['Survived'].groupby(pd.qcut(train['Cabin_num'], 3)).mean()
```

```
Cabin_num
(1.999, 28.667]    0.716418
(28.667, 65.667]   0.651515
(65.667, 148.0]    0.641791
Name: Survived, dtype: float64
```

## # FINDING CORRELATION BETWEEN CABIN NUMBER AND SURVIVED:

```
train['Survived'].corr(train['Cabin_num'])
```

```
-0.06384595922789353
```

---

It shows negative correlation, with increasing cabin number survival rate is decreasing.

## # FINDING NUMBER OF PASSENGERS FOR DIFFERENT PORT OF EMBARKATION

```
train['Embarked'].value_counts()
```

```
S    644  
C    168  
Q     77  
Name: Embarked, dtype: int64
```

```
train['Embarked'].value_counts(normalize=True)
```

```
S    0.724409  
C    0.188976  
Q    0.086614  
Name: Embarked, dtype: float64
```

Here,  
We can see that Port of Embarkation 'S' has highest number of passengers.

## # FINDING AVERAGE SURVIVAL RATE WITH RESPECT TO DIFFERENT PORT OF EMBARKATION:

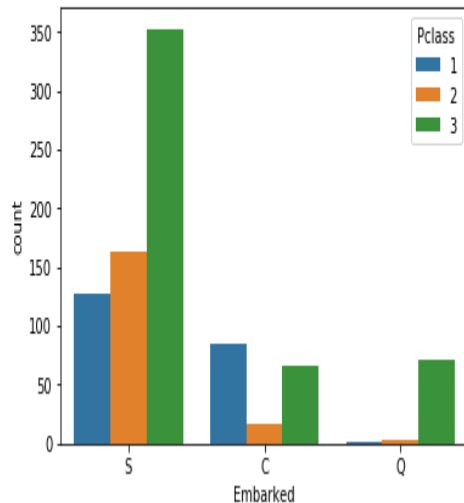
```
train['Survived'].groupby(train['Embarked']).mean()
```

```
Embarked  
C    0.553571  
Q    0.389610  
S    0.336957  
Name: Survived, dtype: float64
```

## # PLOTTING COUNTPLOT OF DIFFERENT PORT OF EMBARKATION WITH RESPECT TO 'PCLASS':

```
sns.countplot(train['Embarked'], hue=train['Pclass'])
```

<matplotlib.axes.\_subplots.AxesSubplot at 0xbcc6828>



Here,

We can see that Port of Embarkation 'S' has highest number of passengers in every Pclass. It has highest people in Pclass = 3.

## # DIFFERENT FUNCTIONS AND USAGE:

```
def names(train, test):  
    for i in [train, test]:  
        i['Name_Len'] = i['Name'].apply(lambda x: len(x))  
        i['Name_Title'] = i['Name'].apply(lambda x: x.split(',')[1]).apply(lambda x: x.split()[0])  
        del i['Name']  
    return train, test
```

This function creates two separate columns: a numeric column indicating the length of a passenger's Name field, and a categorical column that extracts the passenger's title. Looking at the relationship between the length of a name and survival rate appears to indicate that there is indeed a clear relationship.

```
def age_impute(train, test):
    for i in [train, test]:
        i['Age_Null_Flag'] = i['Age'].apply(lambda x: 1 if pd.isnull(x) else 0)
        data = train.groupby(['Name_Title', 'Pclass'])['Age']
        i['Age'] = data.transform(lambda x: x.fillna(x.mean()))
    return train, test
```

In this function the null values of the Age column are imputed by filling with the mean value of the passengers' age corresponding to title and class. This more granular approach to imputation should be more accurate than merely taking the mean age of the population.

```
def fam_size(train, test):
    for i in [train, test]:
        i['Fam_Size'] = np.where((i['SibSp'] + i['Parch']) == 0, 'Solo',
                                np.where((i['SibSp'] + i['Parch']) <= 3, 'Nuclear', 'Big'))
        del i['SibSp']
        del i['Parch']
    return train, test
```

In this function we are creating a new variable called 'Fam\_Size' for family size. The importance of this variable is high because the distribution and survival rate between the different categories does not give much hope. Same conclusions as Sibsp: passengers with zero parents or children had a lower likelihood of survival than otherwise, but that survival rate was only slightly less than the overall population survival rate. We combine the SibSp and Parch columns into a new variable that indicates family size, and group the family size variable into three categories.

```
def ticket_grouped(train, test):
    for i in [train, test]:
        i['Ticket_Lett'] = i['Ticket'].apply(lambda x: str(x)[0])
        i['Ticket_Lett'] = i['Ticket_Lett'].apply(lambda x: str(x))
        i['Ticket_Lett'] = np.where((i['Ticket_Lett'].isin(['1', '2', '3', 'S', 'P', 'C', 'A'])), i['Ticket_Lett'],
                                    np.where((i['Ticket_Lett'].isin(['W', '4', '7', '6', 'L', '5', '8'])),
                                                'Low_ticket', 'Other_ticket'))
        i['Ticket_Len'] = i['Ticket'].apply(lambda x: len(x))
        del i['Ticket']
    return train, test
```

Another piece of information is the first letter of each ticket, which, again, might be indicative of a certain attribute of the ticketholders or their rooms. The Ticket column is used to create two new columns: Ticket\_Lett, which indicates the first letter of each ticket (with the smaller-n values being grouped based on survival rate); and Ticket\_Len, which indicates the length of the Ticket field.



```
def cabin(train, test):
    for i in [train, test]:
        i['Cabin_Letter'] = i['Cabin'].apply(lambda x: str(x)[0])
        del i['Cabin']
    return train, test
```

We can see that most of the cabin letters are associated with a high survival rate, so this might very well be a useful variable. Because there aren't that many unique values, we won't do any grouping here, even if some of the values have a small count.

```
def cabin_num(train, test):
    for i in [train, test]:
        i['Cabin_num1'] = i['Cabin'].apply(lambda x: str(x).split(' ')[-1][1:])
        i['Cabin_num1'].replace('an', np.NaN, inplace = True)
        i['Cabin_num1'] = i['Cabin_num1'].apply(lambda x: int(x) if not pd.isnull(x) and x != '' else np.NaN)
        i['Cabin_num'] = pd.qcut(train['Cabin_num1'],3)
    train = pd.concat((train, pd.get_dummies(train['Cabin_num'], prefix = 'Cabin_num')), axis = 1)
    test = pd.concat((test, pd.get_dummies(test['Cabin_num'], prefix = 'Cabin_num')), axis = 1)
    del train['Cabin_num']
    del test['Cabin_num']
    del train['Cabin_num1']
    del test['Cabin_num1']
    return train, test
```

Upon first glance, this appears to be useless. Not only do we have ~700 nulls which will be difficult to impute, but the correlation with Survived is almost zero. However, the cabin numbers as a whole do seem to have a high survival rate compared to the population average, so we can use it as a feature.

```
def embarked_impute(train, test):
    for i in [train, test]:
        i['Embarked'] = i['Embarked'].fillna('S')
    return train, test
```

In this function we fill the null values in the Embarked column with the most commonly occurring value, which is 'S.'

```
def dummies(train, test, columns = ['Pclass', 'Sex', 'Embarked', 'Ticket_Lett', 'Cabin_Letter', 'Name_Title', 'Fam_Size']):
    for column in columns:
        train[column] = train[column].apply(lambda x: str(x))
        test[column] = test[column].apply(lambda x: str(x))
        good_cols = [column+'_'+i for i in train[column].unique() if i in test[column].unique()]
        train = pd.concat((train, pd.get_dummies(train[column], prefix = column)[good_cols]), axis = 1)
        test = pd.concat((test, pd.get_dummies(test[column], prefix = column)[good_cols]), axis = 1)
        del train[column]
        del test[column]
    return train, test
```

because we are using scikit-learn, we must convert our categorical columns into dummy variables. The following function does this, and then it drops the original categorical columns. It also makes sure that each category is present in both the training and test datasets.

```
def drop(train, test, bye = ['PassengerId']):
    for i in [train, test]:
        for z in bye:
            del i[z]
    return train, test
```

Our last helper function drops any columns that haven't already been dropped. In our case, we only need to drop the PassengerId column, which we have decided is not useful for our problem (by the way, I've confirmed this with a separate test). Note that dropping the PassengerId column here means that we'll have to load it later when creating our submission file.

```
test['Fare'].fillna(train['Fare'].mean(), inplace = True)
```

We also fill in the one missing value of Fare in our test set with the mean value of Fare from the training set (transformations of test set data must always be fit using training data).

## # BUILDING THE MODEL:

```
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')

train, test = names(train, test)
train, test = age_impute(train, test)
train, test = cabin_num(train, test)
train, test = cabin(train, test)
train, test = embarked_impute(train, test)
train, test = fam_size(train, test)
test['Fare'].fillna(train['Fare'].mean(), inplace = True)
train, test = ticket_grouped(train, test)
train, test = dummies(train, test, columns = ['Pclass', 'Sex', 'Embarked', 'Ticket_Lett',
                                              'Cabin_Letter', 'Name_Title', 'Fam_Size'])

train, test = drop(train, test)

X=train.drop(['Survived'], axis=1).values.astype(float)
y = train['Survived'].values

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1, stratify=y)
```

Having built our helper functions, we can now execute them in order to build our dataset that will be used in the model.

```
print(len(train.columns))
```

45

We can see that our final dataset has 45 columns, composed of our target column and 44 predictor variables. Although highly dimensional datasets can result in high variance, I think we should be fine here.

## 5.3 HYPERPARAMETER TUNING

We will use grid search to identify the optimal parameters of our random forest model. Because our training dataset is quite small, we can get away with testing a wider range of hyperparameter values.

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier

test_model = RandomForestClassifier(max_features='auto', oob_score=True, random_state=1, n_jobs=-1)
```

```
param_grid = { "criterion" : ["gini", "entropy"], "min_samples_leaf" : [1, 5, 10],
               "min_samples_split" : [2, 4, 10, 12, 16], "n_estimators": [50, 100, 400, 700, 1000]}
```

```
gs = GridSearchCV(estimator=test_model, param_grid=param_grid, scoring='accuracy', cv=3, n_jobs=-1)

gs = gs.fit(X_train, y_train)
```

```
print(gs.best_score_)
print(gs.best_params_)
# print(gs.cv_results_)
```

```
0.8293001521701072
{'criterion': 'gini', 'min_samples_leaf': 5, 'min_samples_split': 12, 'n_estimators': 700}
```

Looking at the results of the grid search:

```
0.8293001521701072
{'criterion': 'gini', 'min_samples_leaf': 5, 'min_samples_split': 12, 'n_estimators': 700}
```

...we can see that our optimal parameter settings are not at the endpoints of our provided values, meaning that we do not have to test more values. What else can we say about our optimal values? The `min_samples_split` parameter is at 12, which should help mitigate overfitting to a certain degree. This is especially good because we have a relatively large number of estimators (700), which could potentially increase our generalization error.

## 5.4 MODEL ESTIMATION AND EVALUATION:

### Random Forest Classifier:

The random forest classifier is a supervised learning algorithm which you can use for regression and classification problems. It is among the most popular machine learning algorithms due to its high flexibility and ease of implementation.

Why is the random forest classifier called the random forest?

That's because it consists of multiple decision trees just as a forest has many trees. On top of that, it uses randomness to enhance its accuracy and combat overfitting, which can be a huge issue for such a sophisticated algorithm. These algorithms make decision trees based on a random selection of data samples and get predictions from every tree. After that, they select the best viable solution through votes.

It has numerous applications in our daily lives such as feature selectors, recommender systems, and image classifiers. Some of its real-life applications include fraud detection, classification of loan applications, and disease prediction. It forms the basis for the Boruta algorithm, which picks vital features in a dataset.

```
model = RandomForestClassifier(criterion='gini',
                              n_estimators=700,
                              min_samples_split=12,
                              min_samples_leaf=5,
                              max_features='auto',
                              oob_score=True,
                              random_state=1,
                              n_jobs=-1)

model.fit(X_train, y_train)

RandomForestClassifier(min_samples_leaf=5, min_samples_split=12,
                      n_estimators=700, n_jobs=-1, oob_score=True,
                      random_state=1)

print("%.4f" % model.oob_score_)
```

0.8263

We are now ready to fit our model using the optimal hyperparameters. The out-of-bag score can give us an unbiased estimate of the model accuracy, and we can see that the score is 82.63%. Let's take a brief look at our variable importance according to our random forest model. We can see that some of the original columns we predicted would be important in fact were, including

gender, fare, and age. But we also see title, name length, and ticket length feature prominently, which are also used as useful variables.

```
predictions = model.predict(X_test)
```

```
predictions
```

```
array([0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0,  
       1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1,  
       1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0,  
       0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0,  
       0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1,  
       0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,  
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0,  
       0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0,  
       0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1,  
       1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0,  
       1, 1, 1], dtype=int64)
```

Our last step is to predict the target variable. Here we have got the array of prediction of survival. Here, 1 = Survived, 0 = Not Survived.

## 5.5 CONFUSION MATRIX:

Confusion matrix is a very popular measure used while solving classification problems. It can be applied to binary classification as well as for multiclass classification problems. An example of a confusion matrix for binary classification is shown in Table .

Table . Confusion matrix for binary classification.

Actual	Predicted	
	Negative	Positive
Negative	TN	FP
Positive	FN	TP

Confusion matrices represent counts from predicted and actual values. The output “TN” stands for True Negative which shows the number of negative examples classified accurately.

Similarly, “TP” stands for True Positive which indicates the number of positive examples classified accurately. The term “FP” shows False Positive value, i.e., the number of actual negative examples classified as positive; and “FN” means a False Negative value which is the number of actual positive examples classified as negative. One of the most commonly used metrics while performing classification is accuracy.

The accuracy of a model (through a confusion matrix) is calculated using the given formula below.

$$\text{Accuracy} = \frac{\text{TN} + \text{TP}}{\text{TN} + \text{FP} + \text{FN} + \text{TP}}$$

Accuracy can be misleading if used with imbalanced datasets, and therefore there are other metrics based on confusion matrix which can be useful for evaluating performance.

```
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
```

```
confusion_matrix(y_test, predictions)
```

```
array([[123, 14],  
       [ 21, 65]], dtype=int64)
```

Here we can see,

TN = 123,

TP = 65,

FN = 21,

FP = 14

Here ,

True Negative (TN) means passenger did not survive and it is predicted by the model correctly.

True Positive(TP) means passenger survived and it is predicted by the model correctly.

False Negative (FN) means passenger survived but it is predicted by the model incorrectly that the passenger did not survive.

False Positive(FP) means passenger did not survive but it is predicted by the model incorrectly that the passenger survived.

```
print(accuracy_score(y_test, predictions))
```

```
0.8430493273542601
```

So, we are getting accuracy score = 0.843

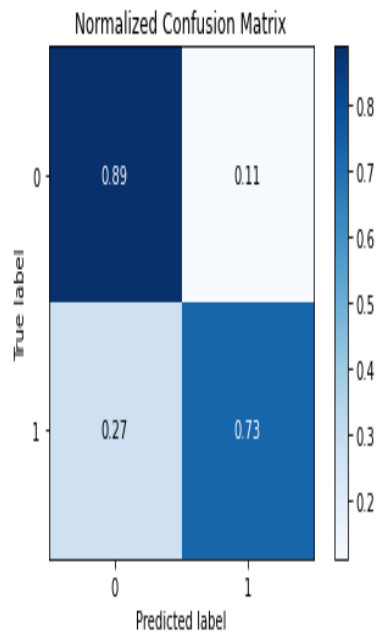


## # PLOTTING NORMALIZED CONFUSION MATRIX:

```
import scikitplot
```

```
#Confusion Matrix  
scikitplot.metrics.plot_confusion_matrix(y_test, predictions, normalize=True)
```

```
<AxesSubplot:title={'center':'Normalized Confusion Matrix'}, xlabel='Predicted label', ylabel='True label'>
```



The higher the diagonal values of the confusion matrix the better, indicating many correct predictions. A normalized confusion matrix makes it easier for the data scientist to visually interpret how the labels are being predicted.

TN = 0.89

FN = 0.27

FP = 0.11

TP = 0.73

## 5.6 CLASSIFICATION REPORT:

A Classification report is used to measure the quality of predictions from a classification algorithm. How many predictions are True and how many are False. More specifically, True Positives, False Positives, True negatives and False Negatives are used to predict the metrics of a classification report as shown below.

```
print(classification_report(y_test, predictions))
```

	precision	recall	f1-score	support
0	0.85	0.90	0.88	137
1	0.82	0.76	0.79	86
accuracy			0.84	223
macro avg	0.84	0.83	0.83	223
weighted avg	0.84	0.84	0.84	223

The report shows the main classification metrics precision, recall and f1-score on a per-class basis. The metrics are calculated by using true and false positives, true and false negatives. Positive and negative in this case are generic names for the predicted classes. There are four ways to check if the predictions are right or wrong:

TN / True Negative: when a case was negative and predicted negative

TP / True Positive: when a case was positive and predicted positive

FN / False Negative: when a case was positive but predicted negative

FP / False Positive: when a case was negative but predicted positive

**Precision** – What percent of your predictions were correct?

Precision is the ability of a classifier not to label an instance positive that is actually negative. For each class it is defined as the ratio of true positives to the sum of true and false positives.

TP – True Positives

FP – False Positives

Precision – Accuracy of positive predictions.

Precision =  $TP / (TP + FP)$

**Recall** – What percent of the positive cases did you catch?

Recall is the ability of a classifier to find all positive instances. For each class it is defined as the ratio of true positives to the sum of true positives and false negatives.

FN – False Negatives

Recall: Fraction of positives that were correctly identified.

$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$

**F1 score** – What percent of positive predictions were correct?

The F1 score is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0. Generally speaking, F1 scores are lower than accuracy measures as they embed precision and recall into their computation. As a rule of thumb, the weighted average of F1 should be used to compare classifier models, not global accuracy.

$\text{F1 Score} = 2 * (\text{Recall} * \text{Precision}) / (\text{Recall} + \text{Precision})$

## **Support**

Support is the number of actual occurrences of the class in the specified dataset. Imbalanced support in the training data may indicate structural weaknesses in the reported scores of the classifier and could indicate the need for stratified sampling or rebalancing. Support doesn't change between models but instead diagnoses the evaluation process.

Here,

For 'Not-Survived'- Precision: 0.85 , Recall: 0.90, F1-Score: 0.88, Support: 137

And for 'Survived'- Precision: 0.82 , Recall: 0.76, F1-Score: 0.79, Support: 86

The accuracy is : 0.84

## 5.7 LEARNING CURVE:

A learning curve is a plot of model learning performance over experience or time.

Learning curves are a widely used diagnostic tool in machine learning for algorithms that learn from a training dataset incrementally. The model can be evaluated on the training dataset and on a hold out validation dataset after each update during training and plots of the measured performance can be created to show learning curves.

Reviewing learning curves of models during training can be used to diagnose problems with learning, such as an underfit or overfit model, as well as whether the training and validation datasets are suitably representative.

### Underfit Learning Curves

Underfitting refers to a model that cannot learn the training dataset. An underfit model can be identified from the learning curve of the training loss only.

It may show a flat line or noisy values of relatively high loss, indicating that the model was unable to learn the training dataset at all.

An underfit model may also be identified by a training loss that is decreasing and continues to decrease at the end of the plot.

This indicates that the model is capable of further learning and possible further improvements and that the training process was halted prematurely.

A plot of learning curves shows underfitting if:

- The training loss remains flat regardless of training.
- The training loss continues to decrease until the end of training.

### Overfit Learning Curves

Overfitting refers to a model that has learned the training dataset too well, including the statistical noise or random fluctuations in the training dataset. The problem with overfitting, is that the more specialized the model becomes to training data, the less well it is able to generalize to new data, resulting in an increase in generalization error. This increase in generalization error can be measured by the performance of the model on the validation dataset.

This often occurs if the model has more capacity than is required for the problem, and, in turn, too much flexibility. It can also occur if the model is trained for too long.

A plot of learning curves shows overfitting if:

- The plot of training loss continues to decrease with experience.
- The plot of validation loss decreases to a point and begins increasing again.

The inflection point in validation loss may be the point at which training could be halted as experience after that point shows the dynamics of overfitting.

## Good Fit Learning Curves

A good fit is the goal of the learning algorithm and exists between an overfit and underfit model.

A good fit is identified by a training and validation loss that decreases to a point of stability with a minimal gap between the two final loss values.

The loss of the model will almost always be lower on the training dataset than the validation dataset. This means that we should expect some gap between the train and validation loss learning curves. This gap is referred to as the “generalization gap.”

A plot of learning curves shows a good fit if:

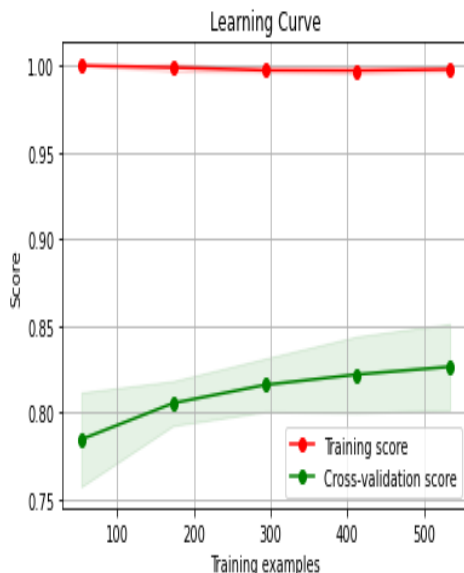
- The plot of training loss decreases to a point of stability.
- The plot of validation loss decreases to a point of stability and has a small gap with the training loss.

Continued training of a good fit will likely lead to an overfit.

```
#Learning Curve
```

```
scikitplot.estimators.plot_learning_curve(test_model, X_train,y_train)
```

```
<AxesSubplot:title={'center':'Learning Curve'}, xlabel='Training examples', ylabel='Score'>
```



From the curve, we can clearly see that as the size of the training set increases, distance between the training score curve and the cross-validation score curve decreases. The cross-validation accuracy increases as we add more training data. So adding training data is useful in this case. Since the training score is very accurate, this indicates low bias and high variance. So

this model also begins overfitting the data because the cross-validation score is relatively lower and increases very slowly as the size of the training set increases.

## 5.8 PREDICTIONS FOR TEST DATASET:

```
final_pred = model.predict(test)
```

```
final_pred
```

```
array([0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0,  
       1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,  
       1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1,  
       1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1,  
       1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,  
       0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0,  
       0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,  
       0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,  
       1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1,  
       0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0,  
       1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,  
       1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,  
       0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0,  
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,  
       1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,  
       1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0,  
       0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0,  
       1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,  
       0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0],  
      dtype=int64)
```

```
final_pred = pd.DataFrame(final_pred, columns=['Survived'])
```

Here we have got the array of prediction of survival.  
Here, 1 = Survived, 0 = Not Survived.

## # WRITING FINAL PREDICTION OF TEST DATASET TO A SEPARATE CSV FILE:

```
test = pd.read_csv('test.csv')  
final_pred = pd.concat((test.iloc[:,0], final_pred), axis=1)
```

```
final_pred.to_csv('Final_submission.csv', sep=",", index=False)
```

```
final_pred.to_csv('Final_submission.')
```

```
test = pd.read_csv('test.csv')  
final_pred = pd.concat((test.iloc[:,0], final_pred), axis=1)
```

```
final_pred.to_csv('Final_submission.')
```

## CHAPTER-6

# FINDINGS & DISCUSSION

- We can see that number of survived is less than number of not survived.
- From this plot , we can see that number of male passengers is greater than that of female.
- Pclass = 3 has maximum number of passengers.
- In case of Port of Embarkation(C = Cherbourg; Q = Queenstown; S = Southampton), S has highest number of passengers.
- Here, We can see that female has higher survival rate than that of male.
- We find that 'Lady', 'Mlle', 'Mme', ' Ms', 'Sir', 'the' – these titles have highest survival rate.
- Here, we can see that Age group 0.419 to 19.0 has highest survival rate. Age group 19.0 to 25.0 has lowest survival rate.
- Age group 0.419 to 19.0 has highest number of people. Age group 25.0 to 31.0 has lowest number of people.
- Number of siblings/spouses=1 has highest survival rate.
- Number of parents/children=3 has highest survival rate.
- Here, we can see that Fare Group 0 to 8.662 has highest number of people.
- Fare Group 26.0 to 512.329 has highest number of people survived.
- The first letter of cabin 'n' has highest number of people.
- It shows negative correlation , with increasing cabin number survival rate is decreasing.
- Port of Embarkation 'S' has highest number of passengers in every Pclass. It has highest people in Pclass = 3.
- Sibsp: passengers with zero parents or children had a lower likelihood of survival than otherwise, but that survival rate was only slightly less than the overall population survival rate.
- From Model Estimation and Evaluation we get the out-of-bag score.  
The out-of-bag score can give us an unbiased estimate of the model accuracy, and we



can see that the score is 82.63%.

- From Confusion Matrix we get,

TN = 123,

TP = 65,

FN = 21,

FP = 14

So, we are getting accuracy score = 0.843

- From Classification Report we get:

For 'Not-Survived'- Precision: 0.85 , Recall: 0.90, F1-Score: 0.88, Support: 137

And for 'Survived'- Precision: 0.82 , Recall: 0.76, F1-Score: 0.79, Support: 86

The accuracy is : 0.84

- From the Learning curve, we can clearly see that as the size of the training set increases, distance between the training score curve and the cross-validation score curve decreases. The cross-validation accuracy increases as we add more training data. So adding training data is useful in this case. Since the training score is very accurate, this indicates low bias and high variance. So this model also begins overfitting the data because the cross-validation score is relatively lower and increases very slowly as the size of the training set increases.

## **CHAPTER-7**

# **IMPLEMENTATION**

The implementation of this project involved the following steps: -

- Import the necessary packages
- Import the required dataset and clean the dataset
- Visualize the dataset
- Split the dataset into a training and test set
- Apply the algorithm
- Generate graphs comparing train and test accuracies
- Generate Confusion Matrix, Classification Report and Learning curve

# CHAPTER-8

## CONCLUSIONS

We can conclude that ,

- We are getting accuracy score = 0.843, this means our model is not 100% accurate.
- From Classification Report we get:

For 'Not-Survived' - Precision: 0.85 , Recall: 0.90, F1-Score: 0.88, Support: 137

And for 'Survived' - Precision: 0.82 , Recall: 0.76, F1-Score: 0.79, Support: 86

which shows that our algorithm is not that accurate, but it can still make good predictions.

In this story, we applied the concepts of Random Forest on Titanic Dataset.

## CHAPTER-9

## REFERENCES

- ❖ <https://towardsdatascience.com/predicting-the-survival-of-titanic-passengers-30870ccc7e8>
- ❖ <https://www.kaggle.com/zlatankr/titanic-random-forest-82-78>
- ❖ <https://www.datacourses.com/evaluation-of-regression-models-in-scikit-learn-846/>
- ❖ <https://realpython.com/best-python-books/>
- ❖ <https://machinelearningmastery.com/machine-learning-in-python-step-by-step/>
- ❖ [https://www.w3schools.com/python/python\\_ml\\_getting\\_started.asp](https://www.w3schools.com/python/python_ml_getting_started.asp)

**THANK YOU**