CHAPTER 1

COMPANY PROFILE

In 1998, Cranes Software decided to start a training division and Cranes Varsity was formed to provide post-professional technical training in niche domains such as digital signal processing (DSP), real time embedded Systems (RTES) and mathematical modelling in 1998 for the academic and corporate sectors.

The CEO of Cranes Varsity is Hariharan Shankar. And the Board of Directors are Asif Khader, Mueed Khader and Akthar Begum.

Cranes Varsity is a pioneer Technical Training institute turned EdTech Platform offering Technology educational services for over 25 years. A division of Cranes Software International Ltd, Cranes Varsity was established with an ambitious vision of bridging the gap between the technology academia and the industry. The team continuously strives to be an organization that brings together technology and education, empowering aspiring professionals to seek assured placements and a lucrative career path. Cranes Varsity offers high-impact hands-on technology training that catapults engineering students, graduates, and working professionals to be quickly employable in Niche high-end engineering fields. The in- house placement team further ensures that these students get placed in leading corporate firms – with whom Cranes Varsity has decades-old relationship. Cranes Varsity carries a legacy of being the Authorized-training partner for Texas Instruments, Math Works, Wind River & ARM. Cranes Varsity also has the honor of being a trusted partner of over 5000 reputed Academia, Corporate & Defense Organizations. Cranes Varsity has training leadership in EMBEDDED, MATLAB & DSP, extending training domains to emerging industry trends like Automotive, IoT, VLSI, Java full-stack, Data Science, Business Analytics and Software Programming.

Services provided by Cranes Varsity:

1. Computer Science:
    - Website Development
    - Desktop and Web Base Applications
    - Android App
    - Full Stack Java Development
    - Data Science with AIML

2. Electronics & Electrical:
    - Embedded Systems
    - Internet of Things
    - VLSI Design

CHAPTER 2

INTRODUCTION

Data science is an interdisciplinary field that combines various techniques, processes, algorithms, and systems to extract insights and knowledge from structured and unstructured data. It encompasses a wide range of activities, including data collection, cleaning, exploration, analysis, visualization, and interpretation. Data science plays a crucial role in helping organizations make data-driven decisions, solve complex problems, and gain a competitive advantage.

2.1 Key Components of Data Science:

1. **Data Collection:** Gathering relevant data from various sources, which may include databases, websites, sensors, and more.
2. **Data Cleaning**: Preprocessing data to handle missing values, outliers, and inconsistencies, ensuring data quality.
3. **Exploratory Data Analysis (EDA**): Investigating and visualizing data to understand its distribution, patterns, and relationships between variables.
4. **Feature Engineering:** Creating new features or transforming existing ones to improve the performance of machine learning models.
5. **Machine Learning:** Developing and deploying predictive and prescriptive models to make data-driven decisions and solve problems.
6. **Data Visualization:** Creating informative and visually appealing charts, graphs, and dashboards to communicate insights effectively.
7. **Statistical Analysis:** Applying statistical techniques to validate hypotheses, test assumptions, and quantify uncertainty.
8. **Big Data Technologies**: Leveraging tools and technologies to handle and analyze large volumes of data efficiently.

2.2 Introduction to Machine Learning:

Machine learning is a subset of artificial intelligence (AI) that focuses on developing algorithms and models that enable computers to learn and make predictions or decisions from data. Unlike traditional programming, where explicit rules are coded, machine learning algorithms allow systems to learn patterns and relationships from data, making them capable of handling complex tasks and adapting to new information.

2.2.1 Key Concepts in Machine Learning:

1. Supervised Learning: In supervised learning, algorithms are trained on labeled data, where the input and the corresponding desired output are provided. It's used for tasks like classification (e.g., spam email detection) and regression (e.g., predicting house prices).

2. Unsupervised Learning: Unsupervised learning involves working with unlabeled data to discover patterns, clusters, or relationships within the data. Common techniques include clustering and dimensionality reduction.
3. Reinforcement Learning: In reinforcement learning, agents interact with an environment and learn to make decisions to maximize a reward signal. It's used in applications like game playing and autonomous systems.
4. Deep Learning: Deep learning is a subset of machine learning that uses artificial neural networks with multiple layers (deep neural networks) to model complex patterns in data. It's widely used in image and speech recognition.
5. Feature Engineering: Selecting and transforming relevant features (input variables) to improve the performance of machine learning models.
6. Model Evaluation: Techniques like cross-validation and various metrics (accuracy, precision, recall, etc.) are used to assess the performance of machine learning models.

2.3 Introduction to Python :

Python is a high-level, interpreted programming language known for its simplicity and readability. It was created by Guido van Rossum and first released in 1991. Python emphasizes clean and concise code, making it a popular choice for beginners and experienced developers alike. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python has a vast standard library and a thriving ecosystem of third-party packages, which contributes to its versatility. It is commonly used for web development, data analysis, machine learning, and scientific computing. Python uses indentation for code blocks, enforcing readability. Its dynamic typing and automatic memory management make it userfriendly. Python's community is active and supportive, and it is open-source, which encourages collaboration and continuous improvement.

2.4 Numpy :

NumPy, short for "Numerical Python," is a fundamental library for scientific and numerical computing in Python. It provides support for working with large, multidimensional arrays and matrices, as well as a wide range of high-level mathematical functions to operate on this data.

2.5 Pandas :

Pandas is a Python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data. Pandas allows us to analyze big data and make conclusions based on statistical theories. Pandas can clean messy data sets, and make them readable and relevant.

2.6 Matplotlib :

Matplotlib is a popular and versatile Python library for creating high-quality, publication-ready visualizations and plots. It is widely used in various fields, including data analysis, scientific research, and data visualization. Matplotlib provides a wide range of tools and functionalities for creating a wide variety of 2D and 3D plots and charts, making it a valuable resource for data scientists, researchers, and anyone who needs to visualize data.

2.7 Linear Regression :

Linear regression is used when you have a continuous numerical target variable, and you want to predict it based on one or more continuous or numerical predictor variables. In the case of the Pokémon data set, the primary focus is on categorical and discrete data, such as Pokémon types, abilities, and characteristics.

Linear regression assumes a linear relationship between the predictor variables and the continuous target variable. Since many of the attributes in the Pokémon data set are categorical and do not have a linear relationship with continuous outcomes, linear regression is not appropriate for most analyses of this data.

When there is a single input variable (x), the method is referred to as 'Simple Linear Regression'. When there are multiple input variables, the method is referred to as 'Multiple Linear Regression'.
Simple regression equation : $Y = A + B * X$
X → Input variable (training data)
B → Coefficient of X
A → Intercept (Constant)
Y → Predicted Value (Calculated from A, B and X)

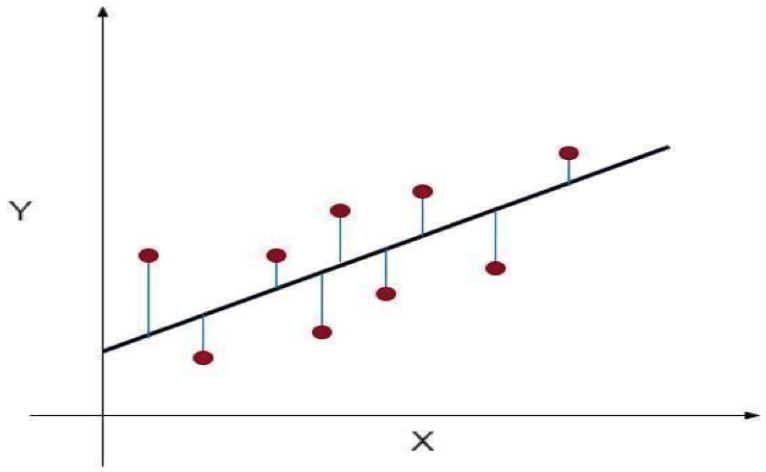

Fig 2.1 : Linear Regression Model

2.8 Logistic Regression :

Logistic regression is a statistical model used for binary classification, which means it's used to predict one of two possible outcomes. The logistic regression equation models the relationship between the input features and the probability of the binary outcome.

$$f(x) = \frac{1}{1 + e^{-x}}$$

Where ‘e’ is Base of natural logarithms.

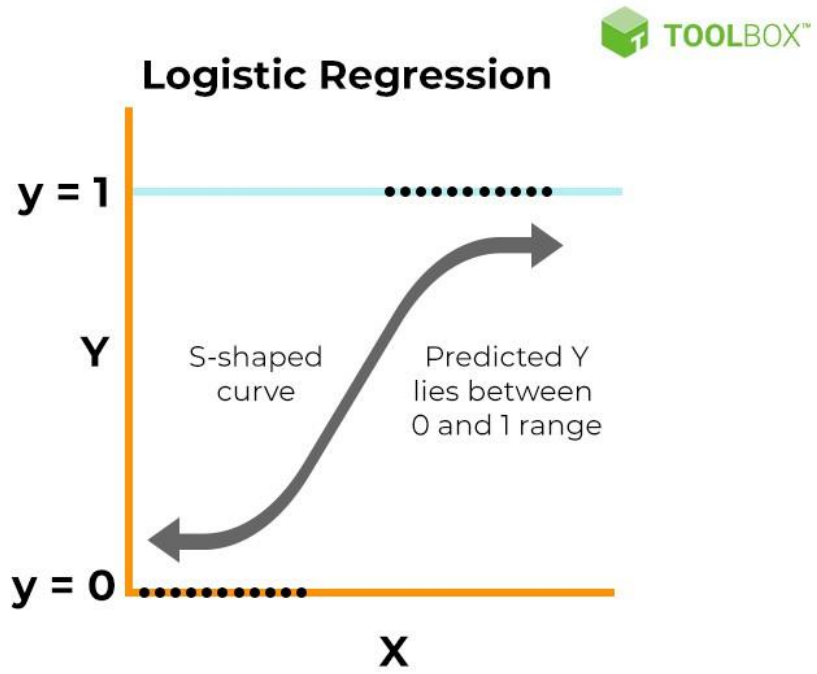

Fig 2.2 : Key Assumptions for Implementing Logistic Regression

2.9 Decision Tree Classifier :

A Decision Tree Classifier is a popular machine learning algorithm used for both classification and regression tasks. It is a supervised learning method that works by recursively partitioning the dataset into subsets based on the features, ultimately making a decision about the class label or target value of each data point. Decision trees are simple to understand, interpret, and visualize, which makes them a valuable tool in both machine learning and data analysis.

2.10 Support Vector Machine :

A Support Vector Machine (SVM) is a powerful and versatile supervised machine learning algorithm used for classification and regression tasks. SVMs are particularly effective for binary classification, but they can also be extended to handle multi-class classification and regression problems. SVMs have gained widespread popularity in various fields, including computer vision, natural language processing, and bioinformatics.

2.11 k-Nearest Neighbors (KNN) :

The k-Nearest Neighbors (KNN) classifier is a simple and intuitive machine learning algorithm used for both classification and regression tasks. It belongs to the family of instance-based or lazy learning algorithms because it doesn't explicitly build a model during the training phase. Instead, it memorizes the training data and makes predictions based on the similarity between new data points and the training instances.

2.12 Exploratory Data Analysis (EDA) :

Exploratory Data Analysis (EDA) is a crucial initial step in the data analysis process. It involves the systematic process of summarizing, visualizing, and understanding the main characteristics of a dataset.

2.12.1 Univariate exploratory data analysis (EDA) :

Univariate analysis is a fundamental technique in data analysis and statistics that focuses on examining and summarizing the characteristics of a single variable (or feature) at a time. It involves studying the distribution, central tendency, variability, and other key statistics of a single variable in isolation from other variables. The primary goal of univariate analysis is to gain insights into the individual variable's properties and behaviors.

2.12.2 Bivariate exploratory data analysis (EDA) :

Bivariate exploratory data analysis (EDA) is a statistical approach used in data analysis to examine the relationships between two variables in a dataset. Unlike univariate analysis, which focuses on a single variable at a time, bivariate EDA explores the interactions and dependencies between pairs of variables. The primary goal of bivariate EDA is to uncover patterns, associations, and correlations between two variables, which can help in understanding how they relate to each other.

2.12.3 Multivariate exploratory data analysis (EDA) :

Multivariate exploratory data analysis (EDA) is a statistical approach used in data analysis to investigate the relationships and interactions between three or more variables simultaneously. Unlike univariate EDA (which focuses on one variable) or bivariate EDA (which examines the relationships between two variables), multivariate EDA explores the complex interplay among multiple variables in a dataset. The primary goal of multivariate EDA is to uncover patterns, dependencies, and associations among variables to gain a deeper understanding of the data.

2.13 Random Forest model :

A Random Forest model is a popular ensemble learning technique used in machine learning for both classification and regression tasks. It's an ensemble of decision trees, where each tree is trained on a different subset of the data, and the final prediction is obtained through a voting or averaging mechanism.

2.14 Statistical modelling :

Statistical modeling is a methodical approach used in data analysis to describe and make inferences about relationships between variables within a dataset. It involves using mathematical and statistical techniques to create models that capture patterns, trends, and dependencies in the data. The primary goal of statistical modeling is to understand, explain, predict, or simulate real-world phenomena based on available data.

Objective of the System

1. Predict Survival Outcomes: Develop machine learning models that accurately predict whether a passenger aboard the Titanic survived or did not survive the disaster.

2. Optimize Model Performance: Strive to achieve high accuracy and well-balanced precision and recall in the survival predictions, ensuring that the model performs well in correctly identifying survivors and non-survivors.

3. Feature Importance Interpretation: Interpret the trained models to understand which features played a significant role in determining survival outcomes. This analysis provides insights into the factors influencing survival on the Titanic.

4. Data Preprocessing: Ensure that the dataset is cleaned and preprocessed effectively to handle missing data, outliers, and categorical variables, allowing the model to make accurate predictions.

5. Model Selection and Hyperparameter Tuning: Select suitable machine learning algorithms for binary classification and fine-tune model hyperparameters to optimize predictive performance.

6. Cross-Validation: Implement cross-validation techniques to assess the generalization capability of the models and reduce overfitting.

7. Documentation and Reporting: Prepare a comprehensive report or documentation that details the entire process, including data preprocessing, feature engineering, model selection, and evaluation.

8. Insights into Historical Context: Provide meaningful insights into the historical context of the Titanic disaster by relating the identified influential factors to the events of that time.

9. Visualization: Create visualizations to illustrate survival rates and relevant trends based on passenger attributes (e.g., gender, class, age), making the results more accessible and understandable.

10. Future Recommendations: Suggest areas for further research or improvement, such as exploring advanced machine learning techniques or incorporating additional datasets to enhance prediction accuracy and interpretability.

CHAPTER 3

OUTCOMES OF INTERNSHIP

Internship assists the student's in development of employer-valued skills such as teamwork, communications and attention to detail.It exposes the student to the environment and expectations of performance on the part of the programmer in professional practices, private/public companies or government entities.It enhance and/or expand the student's knowledge of a particular area(s) of technical methods. It expose the student to professional role models or mentors who will provide the student with support in the early stages of the internship and provide an example of the behaviours expected in the intern's workplace.

1. Demonstrate the application of knowledge and skill sets acquired from the course and workplace in the assigned job functions.

2. Solve real life challenges in the workplace by analyzing work environmental conditions, and selecting appropriate skill sets acquired from the course;

3. Articulate career options by considering opportunities in company, sector, industry, professional and educational advancement;

4. Communicate and collaborate effectively and appropriately with different professionals in the work environment through written and oral means;

5. Exhibit critical thinking and problem solving skills by analyzing underlying issue/s to challenges;

6. Demonstrate the ability to harness resources by analyzing challenges and considering opportunities;

7. Recommend ideas to improve work effectiveness and efficiency by analyzing challenges and considering viable options;

8. Exhibit professional ethics by displaying positive disposition during internship

SKILLS ACQUIRED

- Fundamentals Python programming.
- Learned to do various examples on Jupiter Notebook.
- Usage of Python packages like Numpy, Pandas, Matplotlib etc.
- Accessing of data sets learned during internship.
- Working with csv files using python.
- Understand, evaluate, design and implement.

CHAPTER 4

SYSTEM ANALYSIS

It is a process of collecting and interpreting facts, identifying the problems, and decomposition of a system into its components.System analysis is conducted for the purpose of studying a system or its parts in order to identify its objectives. It is a problem solving technique that improves the system and ensures that all the components of the system work efficiently to accomplish their purpose.

1. Data Collection:

- Obtain the Titanic dataset, which typically includes information about passengers, such as age, gender, class, ticket fare, and whether they survived or not.

2. Data Preprocessing:

- Handle missing data by imputing missing values or removing rows with missing values.

- Deal with outliers that could affect model performance.

- Encode categorical variables using techniques like one-hot encoding.

- Create new features or engineer existing ones to extract valuable information (e.g., extracting titles from passenger names).

3. Data Splitting:

- Split the dataset into training and testing sets to evaluate model performance.

4. Feature Selection:

- Identify which features are most relevant for survival prediction. This can involve using statistical tests, feature importance from models, or domain knowledge.

5. Model Selection:

- Choose and implement machine learning algorithms suitable for binary classification, such as:

  - Decision Trees

  - Random Forests

  - Logistic Regression

  - Support Vector Machines

  - Gradient Boosting methods (e.g., XGBoost, LightGBM)

6. Model Training:

- Train the selected models on the training data.

7. Model Evaluation:

- Evaluate model performance using appropriate metrics, including:
  - Accuracy
  - Precision
  - Recall
  - F1-score
- Perform cross-validation to assess model generalization.

8. Hyperparameter Tuning:

- Fine-tune model hyperparameters to improve performance.

9. Interpretability:

- Interpret model results to understand the importance of different features in predicting survival. Techniques like feature importance, SHAP values, and partial dependence plots can be useful.

10. Visualization:

- Create visualizations to present the results, such as survival rates by gender, class, and age.

11. Model Deployment:

- Deploy the chosen model in a suitable environment if required for real-world use.

12. Documentation and Reporting:

- Prepare a report or documentation summarizing the entire process, including data preprocessing, feature engineering, model selection, and model evaluation.

13. Conclusion and Future Work:

- Suggest areas for further improvement, such as exploring more advanced machine learning techniques or incorporating additional datasets.

CHAPTER 5

REQUIREMENT ANALYSIS

5.1 Hardware Requirements :

- CPU: Intel Core or Xeon 3GHz (or Dual Core 2GHz) or equal AMDCPU
- Cores: Single (Dual/Quad Quad Core is recommended)
- RAM: 4 GB (6 GB recommended)
- Display Resolution: 1280×1024 is recommended, 1024×768 is minimum

5.2 Software Requirements :

Front End: Python , Gradio, Joblib

Back End: Machine Learning With

Python Tools: Jupyter notebook,Google collab,VS code

The following operating systems are officially supported:

- Windows 7 (64-bit, Professional level or higher)
- Mac OS X 10.5.1+
- Ubuntu 9.10 (64bit)
- Ubuntu 8.04 (32bit/64bit)

CHAPTER 6

IMPLEMENTATION

In the Project, I will gain access to **two similar datasets** that include passenger information like name, age, gender, socio-economic class, etc. One dataset is titled **train.csv** and the other is titled **test.csv**.

Train.csv will contain the details of a subset of the passengers on board (891 to be exact) and importantly, will reveal whether they survived or not, also known as the "ground truth".

The **test.csv** dataset contains similar information but does not disclose the "ground truth" for each passenger. It is the job to predict these outcomes.

Datasets are as follows:

- Passenger ID
- Passenger Class
- Name
- Sex
- Age
- Number of passenger's siblings and spouses on board(SibSp)
- Number of passenger's parents and children on board(Parch)
- Ticket
- Fare
- Cabin
- City where passenger embarked

1. Import the relevant python libraries for the analysis

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from IPython.display import Image, display
%matplotlib inline
```

2. Load the train and test dataset and set the index if applicable

```
from google.colab import files

uploaded = files.upload()

for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))
```

 train.csv(text/csv) - 61194 bytes, last modified: 9/7/2023 - 100% done
Saving train.csv to train.csv
User uploaded file "train.csv" with length 61194 bytes

```
#load the train dataset
train = pd.read_csv('train.csv')
```

```
#inspect the first few rows of the train dataset
display(train.head())
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Name_len	Ticket_First	FamilyCount	Cabin_First	title
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S	23	A	1	NaN	Mr.
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38.0	1	0	PC 17599	71.2833	C85	C	51	P	1	C	Mrs.
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.925	NaN	S	22	S	0	NaN	Miss.
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1	C123	S	44	1	1	C	Mrs.
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05	NaN	S	24	3	0	NaN	Mr.

Fig 6.1: Table of first few rows of the train dataset

```
# set the index to passengerId
train = train.set_index('PassengerId')
```

```
from google.colab import files
```

```
uploaded = files.upload()

for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))
```

 test.csv(text/csv) - 28629 bytes, last modified: 9/7/2023 - 100% done
Saving test.csv to test.csv
User uploaded file "test.csv" with length 28629 bytes

```
#load the test dataset
test = pd.read_csv('test.csv')
```

```
#inspect the first few rows of the test dataset
display(test.head())
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

Fig 6.2: Table of first few rows of the test dataset

3. Visually inspect the head of the dataset,Examine the train dataset to understand in particular if the data is tidy, shape of the dataset,examine datatypes, examine missing values, unique counts and build a data dictionary dataframe.

Conditions to check if data is tidy

- Is every column a variable?
- Is every row an observation?
- Is every table a single observational unit?
- by calling the shape attribute of the train dataset I can observe that there are 891 observations and 11 columns in the data set

```
train.shape
```

```
(891, 11)
```

```
# Check out the data summary
# Age, Cabin and Embarked has missing data
```

```
train.head()
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Fig 6.3 :Table of data summary of age, cabin and embarked with missing data

```
# identify datatypes of the 11 columns, add the stats to the datadict
datadict = pd.DataFrame(train.dtypes)
datadict
```

	0
Survived	int64
Pclass	int64
Name	object
Sex	object
Age	float64
SibSp	int64
Parch	int64
Ticket	object
Fare	float64
Cabin	object
Embarked	object

Fig 6.4 :Table of identified datatypes of the 11 columns with the stats to the datadict

```
# identify missing values of the 11 columns,add the stats to the datadict
datadict['MissingVal'] = train.isnull().sum()
datadict
```

	0	MissingVal
Survived	int64	0
Pclass	int64	0
Name	object	0
Sex	object	0
Age	float64	177
SibSp	int64	0
Parch	int64	0
Ticket	object	0
Fare	float64	0
Cabin	object	687
Embarked	object	2

Fig 6.5:Table of identified missing values of the 11 columns with the stats to the datadict

Identify number of unique values, For object nunique will the number of levels

```
# Add the stats the data dict
datadict['NUnique']=train.nunique()
datadict
```

	0	MissingVal	NUnique
Survived	int64	0	2
Pclass	int64	0	3
Name	object	0	891
Sex	object	0	2
Age	float64	177	88
SibSp	int64	0	7
Parch	int64	0	7
Ticket	object	0	681
Fare	float64	0	248
Cabin	object	687	147
Embarked	object	2	3

Fig 6.6:Table of Identified number of unique values,For object nunique will the number of levels

```
# Identify the count for each variable, add the stats to datadict
datadict['Count']=train.count()
datadict
```

	0	MissingVal	NUnique	Count
Survived	int64	0	2	891
Pclass	int64	0	3	891
Name	object	0	891	891
Sex	object	0	2	891
Age	float64	177	88	714
SibSp	int64	0	7	891
Parch	int64	0	7	891
Ticket	object	0	681	891
Fare	float64	0	248	891
Cabin	object	687	147	204
Embarked	object	2	3	889

Fig 6.7: Table of the count for each variable, added to the stats to datadict

```
# rename the 0 column
```

```
datadict = datadict.rename(columns={0:'DataType'})
datadict
```

	DataType	MissingVal	NUnique	Count
Survived	int64	0	2	891
Pclass	int64	0	3	891
Name	object	0	891	891
Sex	object	0	2	891
Age	float64	177	88	714
SibSp	int64	0	7	891
Parch	int64	0	7	891
Ticket	object	0	681	891
Fare	float64	0	248	891
Cabin	object	687	147	204
Embarked	object	2	3	889

Fig 6.8 : Table of renamed the 0 column

4. Run discriptive statistics of object and numerical datatypes, and finally transform datatypes accoringly

```
# get discripte statistcs on "object" datatypes
train.describe(include=['object'])
```

	Name	Sex	Ticket	Cabin	Embarked
count	891	891	891	204	889
unique	891	2	681	147	3
top	Braund, Mr. Owen Harris	male	347082	B96 B98	S
freq	1	577	7	4	644

Fig 6.9:Table of discripte statistcs on "object" datatypes

```
# get discriptive statistcs on "number" datatypes
train.describe(include=['number'])
```

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

Fig 6.10 : Table of discriptive statistcs on "number" datatypes

5. Carryout univariate and multivariate analysis using graphical and non graphical(some numbers represting the data)

```
train.Survived.value_counts(normalize=True)
```

```
0    0.616162
1    0.383838
Name: Survived, dtype: float64
```

Fig 6.11: survived value counts

only 38% of the passengers were survived, where as a majority 61% the passenger did not survive the disaster

Univariate Analysis

```
fig, axes = plt.subplots(2, 4, figsize=(16, 10))
sns.countplot(data=train,x=train['Survived'],ax=axes[0,0])
sns.countplot(data=train,x= train['Pclass'],ax=axes[0,1])
sns.countplot(data=train,x= train['Sex'],ax=axes[0,2])
sns.countplot(data=train,x= train['SibSp'],ax=axes[0,3])
sns.countplot(data=train,x= train['Parch'],ax=axes[1,0])
sns.countplot(data=train,x= train['Embarked'],ax=axes[1,1])
sns.distplot(train['Fare'], kde=True,ax=axes[1,2])
sns.distplot(train['Age'].dropna(),kde=True,ax=axes[1,3])
```

<Axes: xlabel='Age', ylabel='Density'>

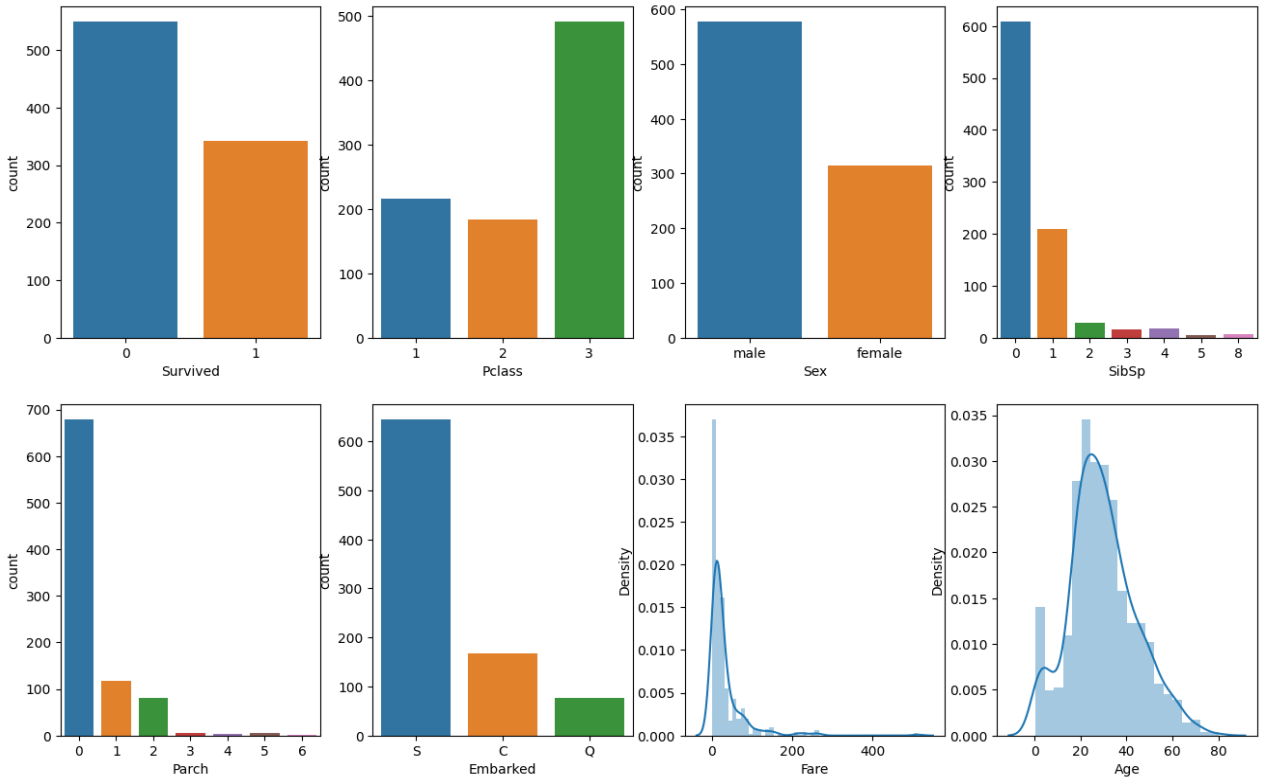

Fig 6.12 : Graph of countplot & distplot for Univariate Analysis

Bivariate EDA

- It is clearly seen that male survial rates is around 20% where as female survial rate is about 75% which suggests that gender has a strong relationship with the survival rates.

- There is also a clear relationship between Pclass and the survival by referring to first plot below. Passengers on Pclass1 had a better survial rate of approx 60% whereas passengers on pclass3 had the worst survial rate of approx 22%

- There is also a marginal relationship between the fare and survial rate.

- I have quantified the above relationships further in the last statsical modelling section

```
figbi, axesbi = plt.subplots(2, 4, figsize=(16, 10))
train.groupby('Pclass')['Survived'].mean().plot(kind='barh',ax=axesbi[0,0],xlim=[0,1])
train.groupby('SibSp')['Survived'].mean().plot(kind='barh',ax=axesbi[0,1],xlim=[0,1])
train.groupby('Parch')['Survived'].mean().plot(kind='barh',ax=axesbi[0,2],xlim=[0,1])
train.groupby('Sex')['Survived'].mean().plot(kind='barh',ax=axesbi[0,3],xlim=[0,1])
train.groupby('Embarked')['Survived'].mean().plot(kind='barh',ax=axesbi[1,0],xlim=[0,1])
sns.boxplot(x="Survived", y="Age", data=train,ax=axesbi[1,1])
sns.boxplot(x="Survived", y="Fare", data=train,ax=axesbi[1,2])
```

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

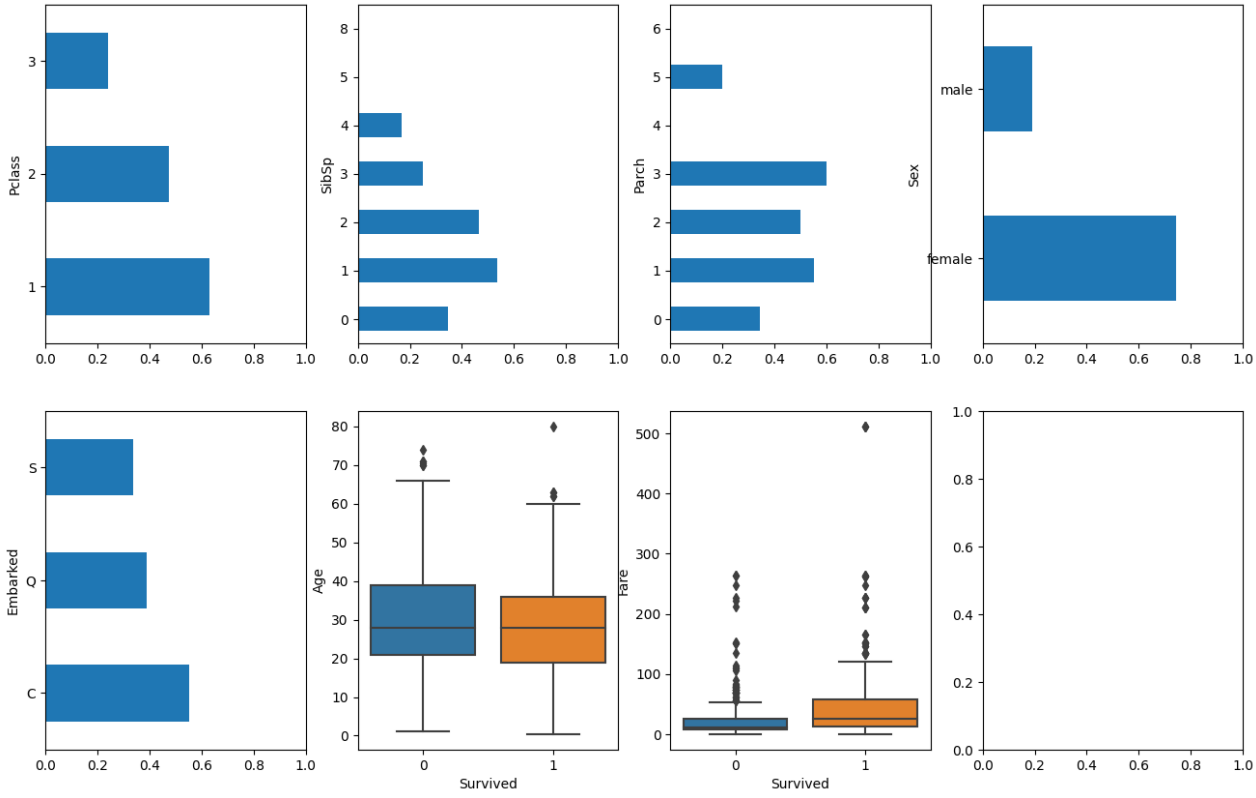

Fig 6.13 : Graph of boxplot for Bivariate analysis

```
#### Joint Plots(continous vs continous)
sns.jointplot(x="Age", y="Fare", data=train);
```

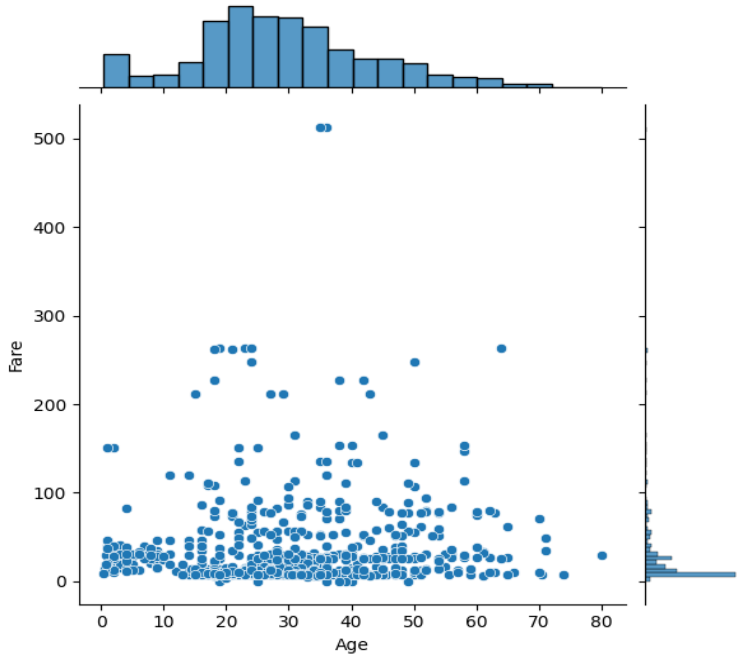

Fig 6.14 : Graph of Joint Plots(continous vs continous)

Multivariate EDA

Construct a Coorelation matrix of the int64 and float64 feature types

There is a positve coorelation between Fare and Survived and a negative coorelation between Pclass and Surived

There is a negative coorelation between Fare and Pclass, Age and Plcass

```
import seaborn as sns

f, ax = plt.subplots(figsize=(10, 8))
corr = train.corr()
sns.heatmap(corr,
            mask=np.zeros_like(corr, dtype=np.bool),
            cmap=sns.diverging_palette(220, 10, as_cmap=True),
            square=True, ax=ax)
```

```
mask=np.zeros_like(corr, dtype=np.bool),
<Axes: >
```

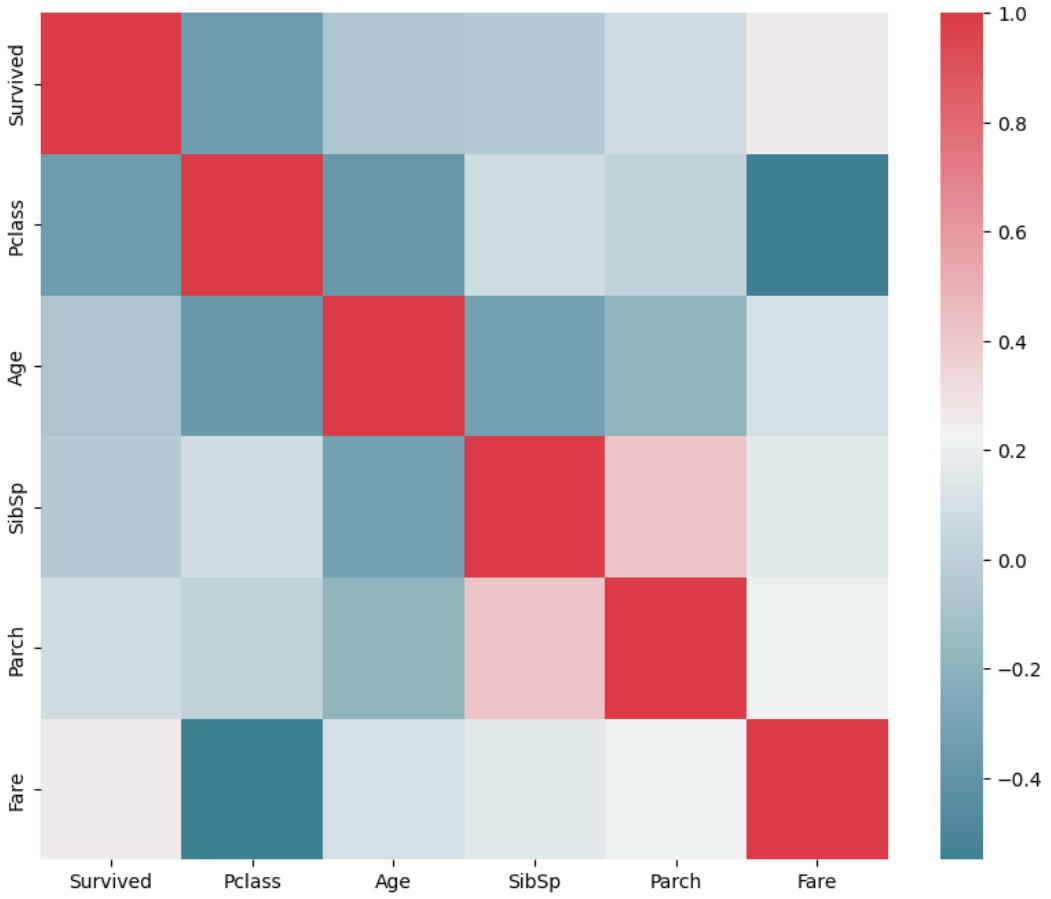

Fig 6.15 : Heatmap for Multivariate EDA

6. Feature Engineering Data- Extract title from name, Extract new features from the other features

New Features

```
train['Name_len']=train.Name.str.len()

train['Ticket_First']=train.Ticket.str[0]

train['FamilyCount']=train.SibSp+train.Parch

train['Cabin_First']=train.Cabin.str[0]

# Regular expression to get the title of the Name
train['title'] = train.Name.str.extract('\, ([A-Z][^ ]*\.)',expand=False)

train.title.value_counts().reset_index()
```

	index	title
0	Mr.	517
1	Miss.	182
2	Mrs.	125
3	Master.	40
4	Dr.	7
5	Rev.	6
6	Major.	2
7	Mlle.	2
8	Col.	2
9	Don.	1
10	Mme.	1
11	Ms.	1
12	Lady.	1
13	Sir.	1
14	Capt.	1
15	Jonkheer.	1

Fig 6.16 : Table of Data extract title from name

7. Preprocessing and Prepare data for statistical modeling

a. Input Missing or Zero values to the Fare variable

```
# It is seen that there are 15 Zero values and its reasonbale
# to flag them as missing values since every ticket
```

```
# should have a value greater than 0
print((train.Fare == 0).sum())
```

15

```
# mark zero values as missing or NaN
train.Fare = train.Fare.replace(0, np.NaN)
```

```
# validate to see if there are no more zero values
print((train.Fare == 0).sum())
```

0

```
# keep the index
train[train.Fare.isnull()].index
```

Int64Index([180, 264, 272, 278, 303, 414, 467, 482, 598, 634, 675, 733, 807, 816, 823],

dtype='int64', name='PassengerId')

```
train.Fare.mean()
```

32.75564988584475

```
#Having missing values in a dataset can cause errors with some machine learning algorithms and either the rows that has missing values should be removed or imputed

#Imputing refers to using a model to replace missing values.

#There are many options that could be considered when replacing a missing value, for example:

#>- constant value that has meaning within the domain, such as 0, distinct from all other values.
#>- value from another randomly selected record.
#>- mean, median or mode value for the column.
#>- value estimated by another predictive model.
```

```
# impute the missing Fare values with the mean Fare value
train.Fare.fillna(train.Fare.mean(),inplace=True)
```

```
# validate if any null values are present after the imputation
train[train.Fare.isnull()]
```

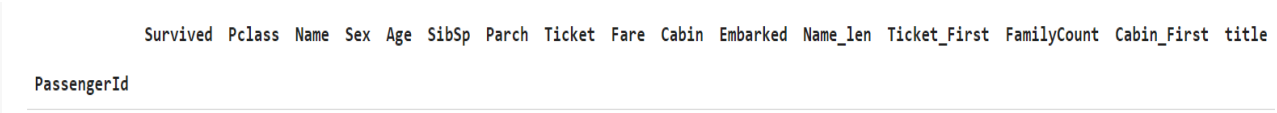

Fig 6.17 : output after validating if any null values are present after the imputation

b. Imput Missing or Zero values to the Age variable

```
# It is seen that there are 0 Zero values
print((train.Age == 0).sum())
```

0

```
# impute the missing Age values with the mean Fare value
train.Age.fillna(train.Age.mean(),inplace=True)
```

```
# validate if any null values are present after the imputation
train[train.Age.isnull()]
```

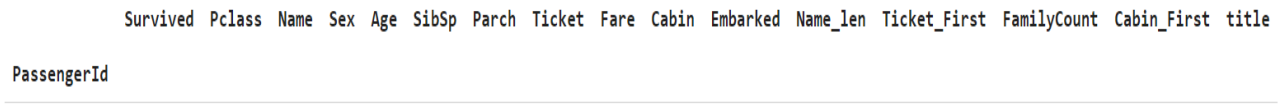

Fig 6.18 : output after validated if any null values are present after the imputation

c. Imput Missing or Zero values to the Cabin variable

```
# It is seen that a majority 77% of the Cabin variable has missing values.
# Hence will drop the column from training a machine learnign algorithem
train.Cabin.isnull().mean()
```

0.7710437710437711

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 891 entries, 1 to 891
Data columns (total 16 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Survived     891 non-null    int64
 1   Pclass       891 non-null    int64
 2   Name         891 non-null    object
 3   Sex          891 non-null    object
```

```
 4   Age          891 non-null    float64
 5   SibSp        891 non-null    int64
 6   Parch        891 non-null    int64
 7   Ticket       891 non-null    object
 8   Fare         891 non-null    float64
 9   Cabin        204 non-null    object
 10  Embarked     889 non-null    object
 11  Name_len     891 non-null    int64
 12  Ticket_First 891 non-null    object
 13  FamilyCount  891 non-null    int64
 14  Cabin_First  204 non-null    object
 15  title        890 non-null    object
dtypes: float64(2), int64(6), object(8)
memory usage: 118.3+ KB
```

8. Statistical Modelling

```
train.columns
```

```
Index(['Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp', 'Parch', 'Ticket',
       'Fare', 'Cabin', 'Embarked', 'Name_len', 'Ticket_First', 'FamilyCount',
       'Cabin_First', 'title'],
      dtype='object')
```

Fig 6.19 : Train dataset coloumn datatypes

```
trainML = train[['Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp', 'Parch', 'Ticket',
      'Fare', 'Embarked', 'Name_len', 'Ticket_First', 'FamilyCount',
      'title']]
```

```
# drop rows of missing values
trainML = trainML.dropna()
```

```
# check the datafram has any missing values
trainML.isnull().sum()
```

```
Survived        0
Pclass          0
Name            0
Sex             0
Age             0
SibSp           0
Parch           0
Ticket          0
Fare            0
Embarked        0
Name_len        0
Ticket_First    0
FamilyCount     0
title           0
dtype: int64
```

Fig 6.20 : dataframe with missing values

```
### A single predictor model with logistic regression
#I have used logistic regression as the response variable is a binary classification


#Regression on survival on Age


# Import Estimator AND Instantiate estimator class to create an estimator object
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()


X_Age = trainML[['Age']].values
y = trainML['Survived'].values
# Use the fit method to train
lr.fit(X_Age,y)
# Make a prediction
y_predict = lr.predict(X_Age)
y_predict[:10]
(y == y_predict).mean()
```

0.6182432432432432

```
#The prediction accuracy is marginally better than the base line accuracy of 61.5% which  got earlier


#Regression on survival on Fare


X_Fare = trainML[['Fare']].values
y = trainML['Survived'].values
# Use the fit method to train
```

```
lr.fit(X_Fare,y)
# Make a prediction
y_predict = lr.predict(X_Fare)
y_predict[:10]
(y == y_predict).mean()
```

0.6621621621621622

```
#Regression on survive on Sex(using a Categorical Variable)
```

```
X_sex = pd.get_dummies(trainML['Sex']).values
y = trainML['Survived'].values
# Use the fit method to train
lr.fit(X_sex, y)
# Make a prediction
y_predict = lr.predict(X_sex)
y_predict[:10]
(y == y_predict).mean()
```

0.786036036036036

```
#The gender of passenger is a strong predictor and purely predciting based on gender, the model accuracy increased to 78%
```

```
#Regression on survive on PClass(using a Categorical Variable)
```

```
X_pclass = pd.get_dummies(trainML['Pclass']).values
y = trainML['Survived'].values
lr = LogisticRegression()
lr.fit(X_pclass, y)
# Make a prediction
y_predict = lr.predict(X_pclass)
y_predict[:10]
(y == y_predict).mean()
```

0.6779279279279279

```
#Gender of the passenger seems a strong predictor compared to the PClass of the passenger on Survival
```

```
#Predicting Survival based on Random forest model
```

```
from sklearn.ensemble import RandomForestClassifier
X=trainML[['Age', 'SibSp', 'Parch',
```

```
      'Fare', 'Name_len', 'FamilyCount']].values # Taking all the numerical values
y = trainML['Survived'].values
RF = RandomForestClassifier()
RF.fit(X, y)
# Make a prediction
y_predict = RF.predict(X)
y_predict[:10]
(y == y_predict).mean()
```

```
0.9887387387387387
```

Random forest did a good job in predicting the survival with a 97% accuracy

CONCLUSION

In the project it is clear that machine learning is a powerful tool that has the potential to transform industries, enhance decision-making, and drive innovation. However, it is important to approach it with a clear understanding of its capabilities and limitations, as well as a commitment to ethical practices and responsible AI development.The project underscores the significance of data analysis and machine learning in extracting meaningful insights from historical datasets. The lessons learned from the Titanic dataset can be applied to various other domains, from healthcare to finance, where data-driven decision-making is paramount.

REFERENCES

- Michalski R S,et al.Machine Learning:Challenges of the eighties.Machine Learning,1986,99-102.
- Analyzing Titanic disaster using machine learning algorithms-Computing, Communication and Automation (ICCCA), 2017 International Conference on 21 December 2017, IEEE.
- Eric Lam, Chongxuan Tang, "Titanic Machine Learning From Disaster", LamTang-Titanic Machine Learning From Disaster, 2012.
- Bircan H., Logistic Regression Analysis: Practice in Medical Data, Kocaeli University Social Sciences Institute Journal, 2004 / 2: 185-208.
- J C. Bezdek Introduction of statistical model 1973.
- Vapnik V.N. The Nature of Statistical Learning Theory[M]. New YorkSpringer-Verlag.1995
- V. Vapnik, “Statistical learning theory,” Wiley, New York, 1998.
- Atakurt, Y., 1999, Logistic Regression Analysis and an Implementation in Its Use in Medicine, Ankara University Faculty of Medicine Journal, C.52, Issue 4, P.195, Ankara
- M Jamel Selim S Z The construction of decision tree vol. 61 pp. 177-188 1994.
- 10.https://machinelearningmastery.com/discover-feature-engineering-how-to-engineer-features-and-how-to-get-good-at-it/
- Unwin A, Hofmann H (1999). \GUI and Command-line{ Conict or Synergy?" In K Berk,M Pourahmadi (eds.), Computing Scienceand Statistics.
- Galit Shmueli and Otto R. Koppius MIS Quarterly, Predictive Analytics in Information System Research, , Vol. 35, No. 3(September 2011), pp. 553-572.
- Michalski R S,et al.Machine Learning:Challenges of the eighties.Machine Learning,1986,99-102