A MINI PROJECT REPORT

ON

“Spaceship Titanic Data Prediction”

Submitted in the partial fulfillment of the requirements for

The degree of

BACHELOR OF ENGINEERING IN COMPUTER ENGINEERING

By

1) NAVNATH AUTI
2) SAHIL NAZARE
3) BINITDEV PANDEY
4) ADNAN SHAIKH

UNDER THE GUIDANCE OF

Prof. Hemalata Gosavi

Department of Computer Engineering
Saraswati College of Engineering, Kharghar, Navi-Mumbai
University of Mumbai

2022-23

Saraswati College of Engineering, Kharghar

Vision:

To be universally accepted as autonomous centre of learning in Engineering Education and Research.

Mission:

- To educate students to become responsible and quality technocrats to fulfil society and industry needs.
- To nurture student’s creativity and skills for taking up challenges in all facets of life.

Department of Computer Engineering

Vision:

To be among renowned institution in Computer Engineering Education and Research by developing globally competent graduates.

Mission:

- To produce quality Engineering graduates by imparting quality training, hands on experience and value education.
- To pursue research and new technologies in Computer Engineering and across interdisciplinary areas that extends the scope of Computer Engineering and benefit humanity.
- To provide stimulating learning ambience to enhance innovative ideas, problem solving ability, leadership qualities, team-spirit and ethical responsibilities.

SARASWATI Education Society's

SARASWATI College of Engineering

Learn Live Achieve and Contribute

Kharghar, Navi Mumbai - 410 210.

DEPARTMENT OF COMPUTER ENGINEERING

PROGRAM EDUCATIONAL OBJECTIVE'S

1. To embed a strong foundation of Computer Engineering fundamentals to identify, solve, analyze and design real time engineering problems as a professional or entrepreneur for the benefit of society.
2. To motivate and prepare students for lifelong learning & research to manifest global competitiveness.
3. To equip students with communication, teamwork and leadership skills to accept challenges in all the facts of life ethically.

SARASWATI Education Society's

SARASWATI College of Engineering

Learn Live Achieve and Contribute

Kharghar, Navi Mumbai - 410 210.

DEPARTMENT OF COMPUTER ENGINEERING

PROGRAM OUTCOMES

1. Apply the knowledge of Mathematics, Science and Engineering Fundamentals to solve complex Computer Engineering Problems.
2. Identify, formulate and analyze Computer Engineering Problems and derive conclusion using First Principle of Mathematics, Engineering Science and Computer Science.
3. Investigate Complex Computer Engineering problems to find appropriate solution leading to valid conclusion.
4. Design a software System, components, Process to meet specified needs with appropriate attention to health and Safety Standards, Environmental and Societal Considerations.
5. Create, select and apply appropriate techniques, resources and advance Engineering software to analyze tools and design for Computer Engineering Problems.
6. Understand the Impact of Computer Engineering solution on society and environment for Sustainable development.
7. Understand Societal, health, Safety, cultural, Legal issues and Responsibilities relevant to Engineering Profession.
8. Apply Professional ethics, accountability and equity in Engineering Profession.
9. Work Effectively as a member and leader in multidisciplinary team for a common goal.
10. Communicate effectively within a Profession and Society at large.
11. Appropriately incorporate principles of Management and Finance in one's own Work.

12. Identify educational needs and engage in lifelong learning in a Changing World of Technology.

SARASWATI Education Society's

SARASWATI College of Engineering

Learn Live Achieve and Contribute

Kharghar, Navi Mumbai - 410 210.

DEPARTMENT OF COMPUTER ENGINEERING

PROGRAMME SPECIFIC OUTCOME

1. Formulate and analyze complex engineering problems in computer engineering (Networking/Big data/ Intelligent Systems/Cloud Computing/Real time systems).
2. Plan and develop efficient, reliable, secure and customized application software using cost effective emerging software tools ethically.

SARASWATI Education Society's

SARASWATI College of Engineering

Learn Live Achieve and Contribute

Kharghar, Navi Mumbai - 410 210.

(Approved by AICTE, recg. By Maharashtra Govt. DTE ,Affiliated to Mumbai University)

PLOT NO. 46/46A, SECTOR NO 5, BEHIND MSEB SUBSTATION, KHARGHAR, NAVI MUMBAI-410210

Tel. : 022-27743706 to 11 * Fax : 022-27743712 * Website: www.sce.edu.in

CERTIFICATE

*This is to certify that the requirements for the project report entitled "**Spaceship Titanic Data Predictiob**" have been successfully completed by the following students:*

Roll numbers	Name
05	Navnath Auti
46	Sahil Nazare
48	Binitdev Pandey
68	Adnan Shaikh

In partial fulfillment of Sem –VII , **Bachelor of Engineering of Mumbai University in Computer Engineering** of Saraswati college of Engineering , Kharghar during the academic year 2022-23.

Internal Guide

Prof. Hemalata Gosavi

External Examiner

Project Co-Ordinator

Dr. Anjali Dadhich

Head of Department

Prof. Sujata Bhairnallykar

Principal

Dr. Manjusha Deshmukh

DECLARATION

I declare that this written submission represents my ideas in my own words and where others ideas or words have been included. I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

1. Navnath Auti
2. Sahil Nazare
3. Binitdev Pandey
4. Adnan Shaikh

Date:

ACKNOWLEDGEMENT

After the completion of this work, words are not enough to express feelings about all those who helped us to reach goal.

It's a great pleasure and moment of immense satisfaction for us to express my profound gratitude to **Project Guide**, **Prof. Hemalata Gosavi**, whose constant encouragement enabled us to work enthusiastically. His perpetual motivation, patience and excellent expertise in discussion during progress of the project work have benefited us to an extent, which is beyond expression.

We would also like to give our sincere thanks to **Prof. Sujata Bhairnallykar, Head of Department**, and **Prof. Hemalata Gosavi** Internal guide from Department of Computer Engineering, Saraswati college of Engineering, Kharghar, Navi Mumbai, for their guidance, encouragement, and support during a project.

I am thankful to **Dr. Manjusha Deshmukh, Principal,** Saraswati College of Engineering, Kharghar, Navi Mumbai for providing an outstanding academic environment, also for providing the adequate facilities.

Last but not the least we would also like to thank all the staffs of Saraswati college of Engineering (Computer Engineering Department) for their valuable guidance with their interest and valuable suggestions brightened us.

1. Navnath Auti
2. Sahil Nazare
3. Binitdev Pandey
4. Adnan Shaikh

ABSTRACT

Welcome to the year 2912, where your data science skills are needed to solve a cosmic mystery. We've received a transmission from four light years away and things aren't looking good.

The spaceship titanic was an interstellar passenger liner launched a month ago. With almost 13,000 passengers on board, the vessel set out on its maiden voyage transporting emigrants from our solar system to three newly habitable exoplanets orbiting nearby stars.

While rounding alpha centauri en route to its first destination—the torrid 55 cancri e—the unwary spaceship titanic collided with a space-time anomaly hidden within a dust cloud. Sadly, it met a similar fate as its namesake from 1000 years before. Though the ship stayed intact, almost half of the passengers were transported to an alternate dimension!

We have done Summarization, EDA and Prediction of Spaceship Titanic Data using PySpark (Python interface for Spark) pandas, numpy, seaborn and scikit-learn modules to extract insight on passengers' behaviour those are transported and those aren't and found relationship between various different features using feature engineering and prediction for data is done using Logistic Regression.

Table of Contents

List of Figures

CHAPTER 1
INTRODUCTON

1.1 GENERAL

Machine learning

Machine Learning is the science (and art) of programming computers so they can learn from data. There are so many different types of Machine Learning systems that it is useful to classify them in broad categories based on:

- Whether or not they are trained with human supervision (supervised, unsupervised, semisupervised, and Reinforcement Learning)
- Whether or not they can learn incrementally on the fly (online versus batch learning)
- Whether they work by simply comparing new data points to known data points, or instead detect patterns in the training data and build a predictive model, much like scientists do (instance-based versus model-based learning)

Data Analysis

Data analysis is about asking and answering questions about our data. The objective of the data analysis step is to increase the understanding of the problem by better understanding the problems data. This involves providing multiple different ways to describe the data as an opportunity to review and capture observations and assumptions that can be tested in later experiments.

1. Summarize Data

Summarizing the data is about describing the actual structure of the data. The minimum aspects of the data to summarize are the structure and the distributions.

Data Structure

Summarizing the data structure is about describing the number and data types of attributes. For example, going through this process highlights ideas for transforms in the Data Preparation step for converting attributes from one type to another (such as real to ordinal or ordinal to binary).

Some motivating questions for this step include:

How many attributes and instances are there?

What are the data types of each attribute (e.g. nominal, ordinal, integer, real, etc.)?

Data Distributions

Summarizing the distributions of each attributes can also flush out ideas for possible data transforms in the Data Preparation step, such as the need and effects of Discretization, Normalization and Standardization.

A summary of the distribution of each real valued attribute typically includes the minimum, maximum, median, mode, mean, standard deviation and number of missing values.

Some motivating questions for this step include:

Create a five-number summary of each real-valued attribute.

What is the distribution of the values for the class attribute?

Knowing the distribution of the class attribute (or mean of a regression output variable) is useful because we can use it to define the minimum accuracy of a predictive model.

2. Visualize Data

Visualizing the data is about creating graphs that summarize the data, capturing them and studying them for interesting structure that can be described.

There are seemingly an infinite number of graphs we can create (especially in software like Python data analysis libraries such as Matplotlib and Seaborn, R, MATLAB, WEKA), we focused on Bar plots, Histograms, Box plots, Heat maps, Kernel density plots, Scatter plots and Pair plot.

Tools

PySpark

PySpark is an interface for Apache Spark in Python. It not only allows us to write Spark applications using Python APIs, but also provides the PySpark shell for interactively analyzing our data in a distributed environment. PySpark supports most of Spark’s features such as Spark SQL, DataFrame, Streaming, MLlib (Machine Learning) and Spark Core.

Pandas API on Spark

Pandas API on Spark allows us to scale our pandas workload out. With this package, we can:

- Be immediately productive with Spark, with no learning curve, if already familiar with pandas.
- Have a single codebase that works both with pandas (tests, smaller datasets) and with Spark (distributed datasets).
- Switch to pandas API and PySpark API contexts easily without any overhead

Scikit-learn

Scikit-learn (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support-vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

1.2 PROBLEM STATEMENT AND OBJECTIVE

1.2.1 PROBLEM STATEMENT

The Spaceship Titanic was an interstellar passenger liner launched a month ago. With almost 13,000 passengers on board, the vessel set out on its maiden voyage transporting emigrants from our solar system to three newly habitable exoplanets orbiting nearby stars.

While rounding Alpha Centauri en route to its first destination—the torrid 55 Cancri E—the unwary Spaceship Titanic collided with a spacetime anomaly hidden within a dust cloud. Sadly, it met a similar fate as its namesake from 1000 years before. Though the ship stayed intact, almost half of the passengers were transported to an alternate dimension!

Our task is to determine what factors are related to transportation of passengers to alternate dimension and how different features are related of given data through data analysis concepts and tools and predict which passenger is teleported to different dimension.

1.2.2 OBJECTIVE

- Load data from online/offline source
- Create Data Frame of data using PySpark pandas class
- Check and correct data types of features
- Treat NULL values using suitable statistic
- Summarization of features
- Univariate Exploratory Data Analysis (EDA)
- Multi-variate EDA
- Extracting meaningful insight from Summarization and EDA
- Training Logistic Regression model on training data
- Evaluating performance of a model using test data

CHAPTER 2
METHODOLOGY

2.1 ALGORITHMIC DETAILS

Data pre-processing

Summarization and Wrangling

1. **Info:** Data types of each feature
2. **Describe:** Statistics such as min, max, mean, median, mode, count, standard deviation and quantiles of suitable features
3. **Value counts:** Unique values in each feature with their respective count
4. **Nan with sum:** To count NULL values in each feature
5. **Cross Tabulation:** To get count of different combination of unique values in 2 features.
6. **Correlation:** To check how closely related different real features are.
7. **Normalization:** To get uniform scale for comparison
8. **Fill Nan**: To fill NULL value with suitable statistics
9. **Broadcasting:** To apply arithmetic operation and above specified operation on each value of corresponding feature

Exploratory Data Analysis (EDA)

1. **Head and Tail:** To get the values of subset of starting and ending records in frame format
2. **Bar Plot:** To visualize count of different value in features, if univariate no stacking is apply if bivariate stacking is applied and for trivariate third value is taken as a hue i.e. color. Mostly applied on categorical data type.
3. **Box Plot**: To visualize five descriptive statistics of real value features.
4. **Histogram:** To visualize count of data for real value features, automatic binning is applied.
5. **Scatter Plot:** To visualize relation between two features, if three features are used then third feature is set a hue.
6. **Kernel Density Plot:** Fit kernel suitable for data and show curve visualization accordingly
7. **Heat Map:** Visualize correlation between multiple real features.
8. **Pair Plot:** Visualize scatter (bivariate and trivariate) as well kernel density estimate between multiple features in single graph.

Train and Test splitting

Train and test splitting is used for evaluation of model accuracy, the train data is used to train the model and test is used to evaluate its accuracy on unseen data. Scikit-learn provide a method to split a data into training and test where dependent and independent vector are separated.

Algorithms:

Logistic Regression

I) Logistic regression is a classification algorithm which uses logistic function or sigmoid function which takes value in the range [0, 1].

II) Sigmoid function: f(x) = $\frac{1}{1+e-x}$ --------- (1)

III) Projecting extreme values i.e f: [$-inf$, $+inf$] $->$ [0,1].

IV) Logistic regression work much like Linear regression taking Input values (x) are combined linearly using weights or coefficient values (referred to as the Greek capital letter Beta) to predict an output value ($y = f(x) = p(x)$).

V) Equation of logistic regression:

$$y = \frac{e^{b0 + b1*x}}{1 + e^{b0 + b1*x}} = \frac{1}{1 + e^{-(b0 + b1*x)}} \text{--------- (2)}$$

VI) Where y is the predicted output, b0 is the bias or intercept term and b1 is the coefficient for the single input value (x). Each column in input data has an associated b coefficient (a constant real value).

VII) The coefficients (Beta values b) of the logistic regression algorithm must be estimated from your training data. This is done using maximum-likelihood estimation, which uses maxima-minima technique to find value of parameters such that error rates are as low as possible.

e.g.: Consider we want to create a model which predict loan to be pass or not on basis of Income.

We will find probability of Loan pass (1) and rejected (0) depending on Income.

i.e.$p(x) = p(L = 1|Income = value) = \frac{e^{b0 + b1*x}}{\left(1 + e^{b0 + b1*x}\right)}$

(Loan pass when certain value limit in Income identified).

After simplifying this equation we get $\ln\left(\frac{p(x)}{1-p(x)}\right) = b0 + b1 * x.$ ----(3) After applying maximum likelihood on given function

(Since, $p(x; b0, b1) = L(b0, b1; x)$) using our train set we will find value of b0 and b1 and we will apply these values with value of x to find corresponding value of y in test set. We will consider Loan pass if

f(x) >= 0.5 and rejected if f(x) < 0.5.

$$y = \frac{e^{b0 + b1*X}}{\left(1 + e^{b0 + b1*X}\right)}$$

$$y = \frac{e^{-100 + 0.6*150}}{\left(1 + e^{-100 + 0.6*X}\right)}$$

$$(consider, b0 = -100\ and\ b1 = 0.6)$$

$$f(x) = y = 0.0000453978687$$

$$since, f(x) < 0.5\ Loan\ is\ Rejected.$$

Flowchart:

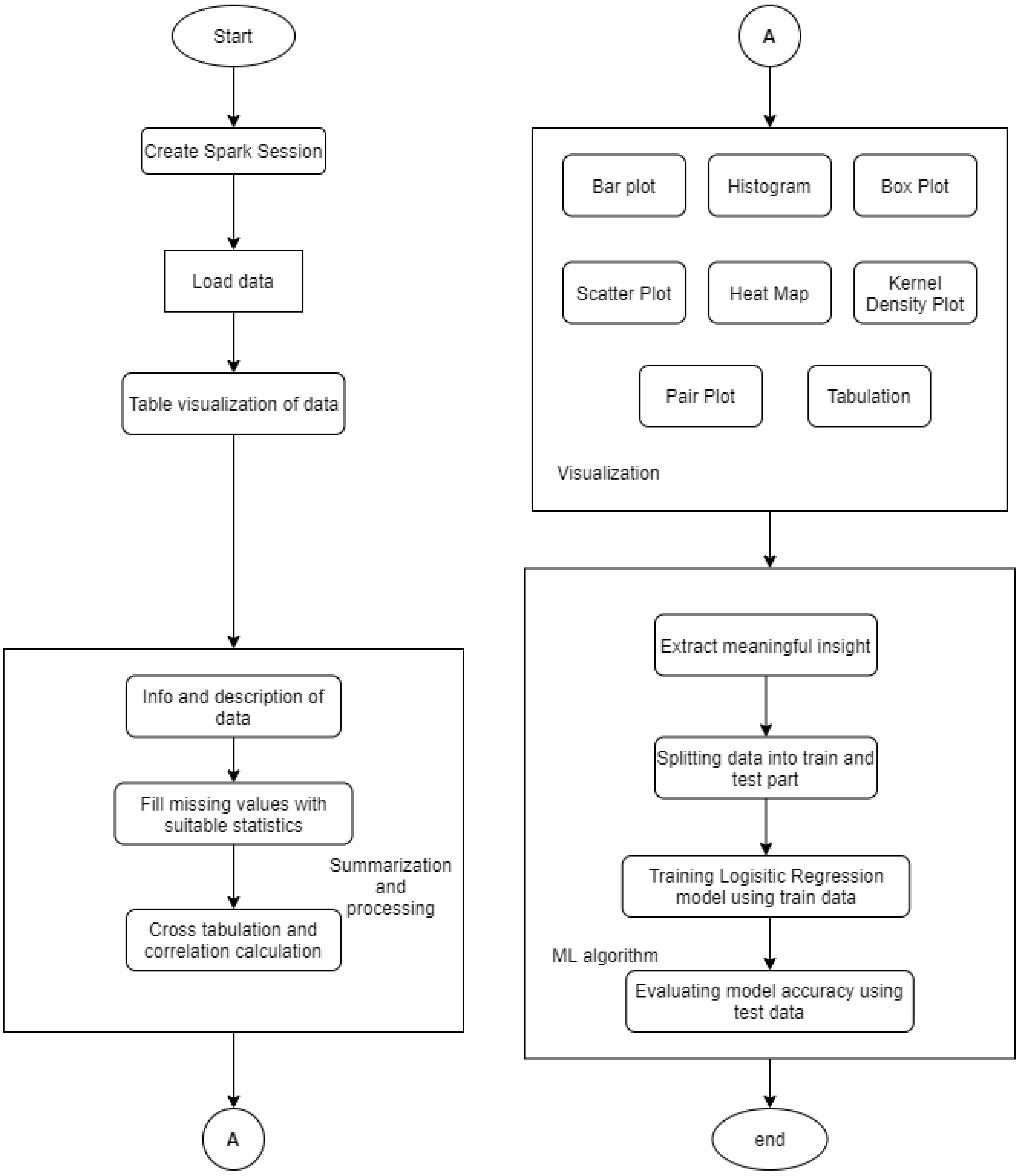

Start
Create Spark Session
Load data
Table visualization of data
Info and description of data
Fill missing values with suitable statistics
Cross tabulation and correlation calculation
Summarization and processing
A
A
Bar plot
Histogram
Box Plot
Scatter Plot
Heat Map
Kernel Density Plot
Pair Plot
Tabulation
Visualization
Extract meaningful insight
Splitting data into train and test part
Training Logisitic Regression model using train data
ML algorithm
Evaluating model accuracy using test data
end

Figure 2.1. Flow Chart

2.2 HARDWARE AND SOFTWARE REQUIREMENTS

2.2.1 HARDWARE REQUIREMENTS

1. RAM: 4GB RAM
2. Hard Drive: 40 GB Hard Drive
3. Processor: Intel Dual Core Processor

2.2.2 SOFTWARE REQUIREMENTS

1. Anaconda
2. Jupyter Notebook
3. Python
4. PySpark
5. Pandas
6. Numpy
7. Matplotlib
8. Seaborn
9. Scikit learn

2.3 DESIGN DETAILS:

Spark session is created using PySpark which return spark instance, using spark instance data is loaded from local repository (can be Hadoop cluster or local directory). Pandas is loaded using PySpark pandas module, numpy, matplotlib and seaborn loaded from their respective packages.

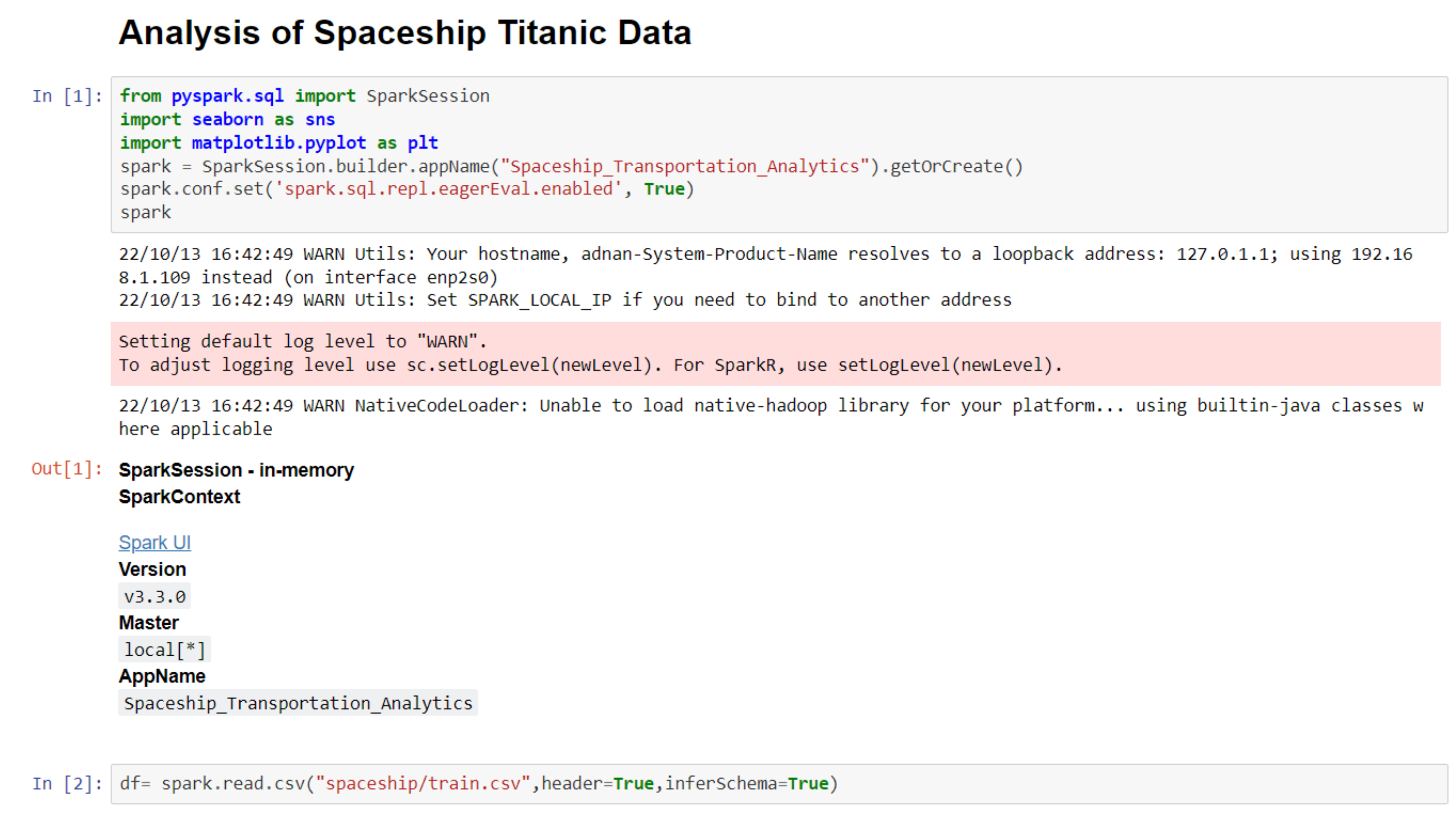
Loaded data is converted pandas data frame using inbuilt method of spark frame and it is visualize in tabular format using head method of frame.

Summarization and Pre-processing is done using built-in pandas, pandas dataframe and numpy methods and functions described in section 2.1 summarization.

Visualization is done using pandas data frame in-built methods and seaborn functions described in section 2.1 EDA. Meaningful insight is gained from these summarization and visualization of data.

Data is split into training and testing part using scikit-learn package from selection_model train_test_split and Logisitic Regression model is provided by scikit-learn which is train using train data and accuracy of this model is evaluated using test data.

CHAPTER 3
IMPLEMENTATION AND RESULTS



Analysis of Spaceship Titanic Data

In [1]:
```python
from pyspark.sql import SparkSession
import seaborn as sns
import matplotlib.pyplot as plt
spark = SparkSession.builder.appName("Spaceship_Transportation_Analytics").getOrCreate()
spark.conf.set('spark.sql.repl.eagerEval.enabled', True)
spark
```

```
22/10/13 16:42:49 WARN Utils: Your hostname, adnan-System-Product-Name resolves to a loopback address: 127.0.1.1; using 192.16
8.1.109 instead (on interface enp2s0)
22/10/13 16:42:49 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
```

```
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
```

```
22/10/13 16:42:49 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes w
here applicable
```

Out[1]:

SparkSession - in-memory

SparkContext

Spark UI

Version

`v3.3.0`

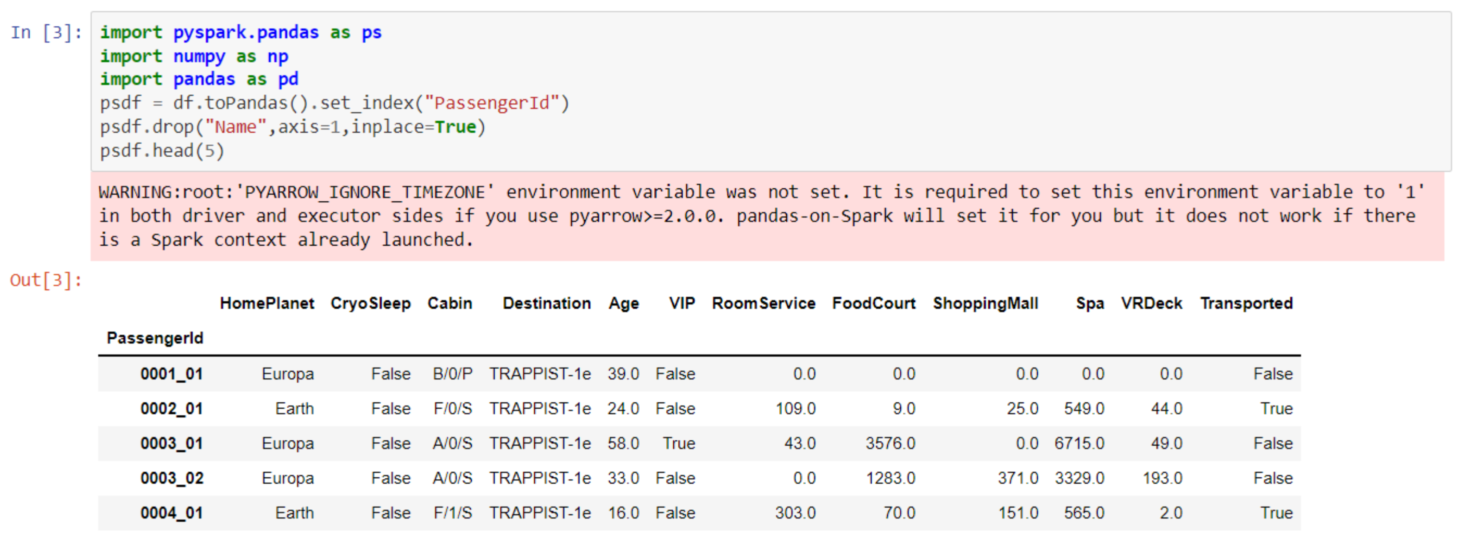
Master

`local[*]`

AppName

`Spaceship_Transportation_Analytics`

In [2]:
```python
df= spark.read.csv("spaceship/train.csv",header=True,inferSchema=True)
```



In [3]:
```python
import pyspark.pandas as ps
import numpy as np
import pandas as pd
psdf = df.toPandas().set_index("PassengerId")
psdf.drop("Name",axis=1,inplace=True)
psdf.head(5)
```

```
WARNING:root:'PYARROW_IGNORE_TIMEZONE' environment variable was not set. It is required to set this environment variable to '1'
in both driver and executor sides if you use pyarrow>=2.0.0. pandas-on-Spark will set it for you but it does not work if there
is a Spark context already launched.
```

Out[3]:

	HomePlanet	CryoSleep	Cabin	Destination	Age	VIP	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck	Transported
PassengerId												
0001_01	Europa	False	B/0/P	TRAPPIST-1e	39.0	False	0.0	0.0	0.0	0.0	0.0	False
0002_01	Earth	False	F/0/S	TRAPPIST-1e	24.0	False	109.0	9.0	25.0	549.0	44.0	True
0003_01	Europa	False	A/0/S	TRAPPIST-1e	58.0	True	43.0	3576.0	0.0	6715.0	49.0	False
0003_02	Europa	False	A/0/S	TRAPPIST-1e	33.0	False	0.0	1283.0	371.0	3329.0	193.0	False
0004_01	Earth	False	F/1/S	TRAPPIST-1e	16.0	False	303.0	70.0	151.0	565.0	2.0	True

In [4]:
```
psdf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 8693 entries, 0001_01 to 9280_02
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   HomePlanet    8492 non-null   object 
 1   CryoSleep     8476 non-null   object 
 2   Cabin         8494 non-null   object 
 3   Destination   8511 non-null   object 
 4   Age           8514 non-null   float64
 5   VIP           8490 non-null   object 
 6   RoomService   8512 non-null   float64
 7   FoodCourt     8510 non-null   float64
 8   ShoppingMall  8485 non-null   float64
 9   Spa           8510 non-null   float64
 10  VRDeck        8505 non-null   float64
 11  Transported   8693 non-null   bool   
dtypes: bool(1), float64(6), object(5)
memory usage: 823.5+ KB
```

In [5]:
```
psdf.describe()
```

Out[5]:

	Age	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck
count	8514.000000	8512.000000	8510.000000	8485.000000	8510.000000	8505.000000
mean	28.827930	224.687617	458.077203	173.729169	311.138778	304.854791
std	14.489021	666.717663	1611.489240	604.696458	1136.705535	1145.717189
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	19.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	27.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	38.000000	47.000000	76.000000	27.000000	59.000000	46.000000
max	79.000000	14327.000000	29813.000000	23492.000000	22408.000000	24133.000000

In [6]:
```
psdf.value_counts()
```

Out[6]:
```
HomePlanet  CryoSleep  Cabin     Destination    Age   VIP    RoomService  FoodCourt  ShoppingMall  Spa   VRDeck  Transported
Europa      True       C/25/S    55 Cancri e    30.0  False  0.0          0.0        0.0           0.0   0.0     True
2
Mars        True       F/1553/P  TRAPPIST-1e    3.0   False  0.0          0.0        0.0           0.0   0.0     True
2
Europa      True       B/19/S    55 Cancri e    18.0  False  0.0          0.0        0.0           0.0   0.0     True
2
Mars        False      F/1050/P  TRAPPIST-1e    4.0   False  0.0          0.0        0.0           0.0   0.0     True
2
Earth       True       G/577/P   TRAPPIST-1e    0.0   False  0.0          0.0        0.0           0.0   0.0     True
2

..
            False      G/556/S   TRAPPIST-1e    29.0  False  0.0          4.0        0.0           0.0   904.0   False
1
                       G/556/P   PSO J318.5-22  37.0  False  308.0        35.0       415.0         10.0  0.0     True
1
                       G/553/S   PSO J318.5-22  26.0  False  6.0          0.0        400.0         2.0   373.0   False
1
                       G/551/S   TRAPPIST-1e    74.0  False  0.0          193.0      581.0         0.0   0.0     True
1
Mars        True       F/999/P   TRAPPIST-1e    15.0  False  0.0          0.0        0.0           0.0   0.0     True
1
Length: 6755, dtype: int64
```

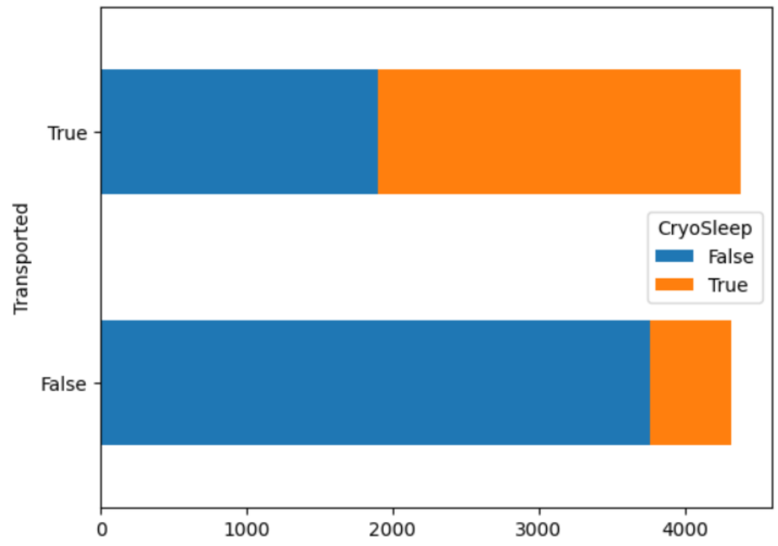
In [7]:
```
np.sum(psdf.isna())
```

Out[7]:
```
HomePlanet      201
CryoSleep       217
Cabin           199
Destination     182
Age             179
VIP             203
RoomService     181
FoodCourt       183
ShoppingMall    208
Spa             183
VRDeck          188
Transported       0
dtype: int64
```

```
In [8]: psdf.HomePlanet = psdf.HomePlanet.fillna("UnkownPlanet")
        np.sum(psdf.HomePlanet.isna())

Out[8]: 0
```

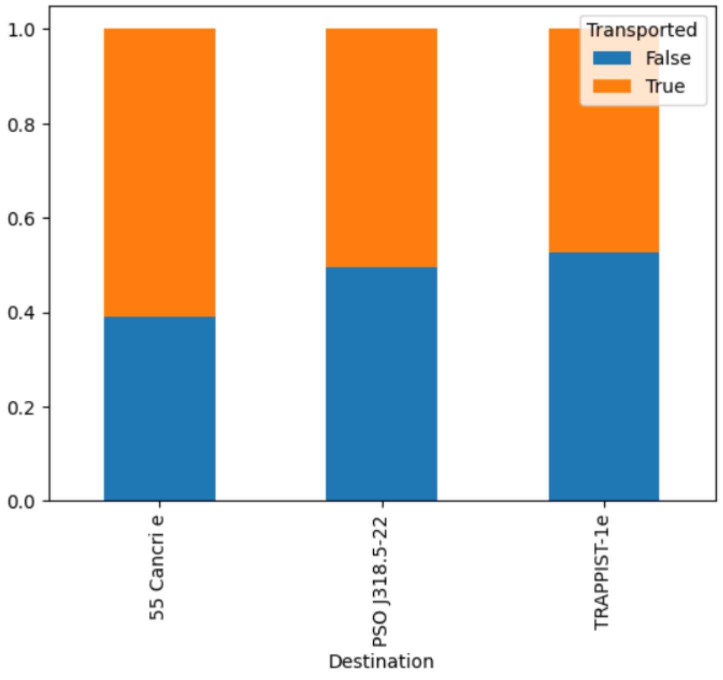
```
In [9]: psdf["CryoSleep"].fillna(False,inplace=True)
        np.sum(psdf["CryoSleep"].isna())

Out[9]: 0
```

```
In [10]: pd.crosstab(index=psdf["Transported"],columns=psdf["CryoSleep"]).plot.barh(stacked=True)
         plt.show()
```

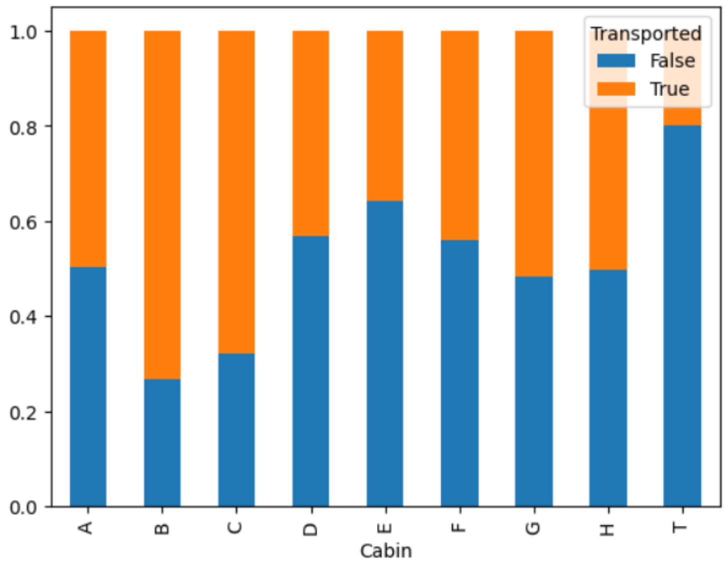


```
In [11]: psdf["Cabin"] = psdf.Cabin.map(lambda x:x[0] if x else "H")
         psdf['Destination'].fillna(psdf['Destination'].mode()[0],inplace=True)
```

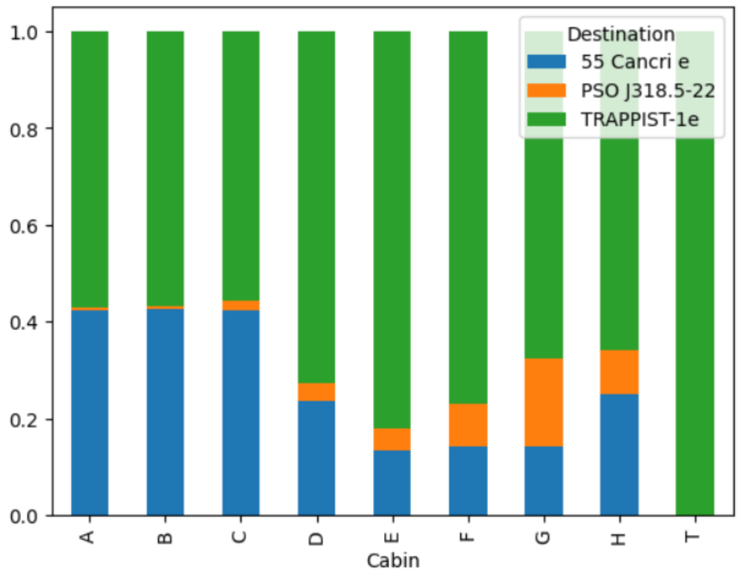
```
In [12]: temp = pd.crosstab(index=psdf["Destination"],columns=psdf["Transported"])
         temp.div(np.sum(temp,axis=1),axis=0).plot.bar(stacked=True)
         plt.show()
```



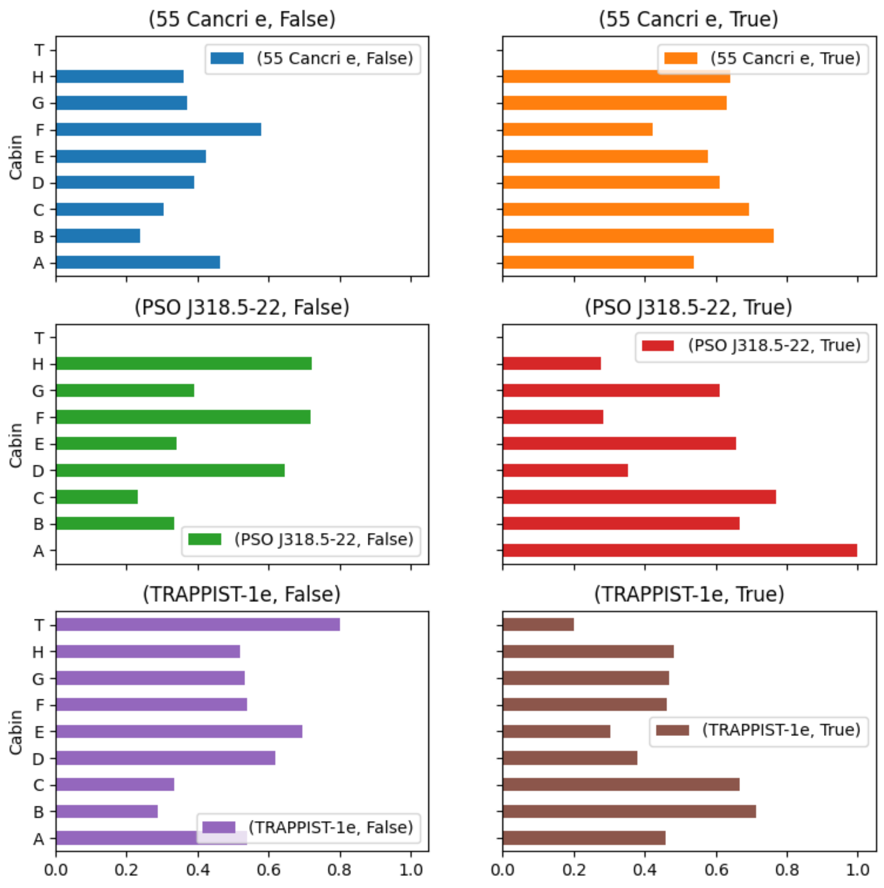
```
In [13]: temp = pd.crosstab(index=psdf["Cabin"],columns=psdf["Transported"])
         temp.div(np.sum(temp,axis=1),axis=0).plot.bar(stacked=True)
         plt.show()
```



```
In [14]: temp = pd.crosstab(index=psdf["Cabin"],columns=psdf["Destination"])
         temp.div(np.sum(temp,axis=1),axis=0).plot.bar(stacked=True)
         plt.show()
```

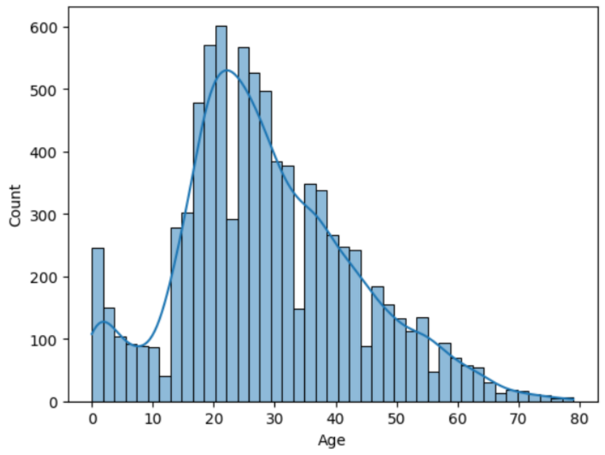


```
In [15]: temp = pd.crosstab(index=psdf["Cabin"],columns=[psdf['Transported'],psdf["Destination"]]).swaplevel(axis=1).sort_index(axis=1)
         temp = temp.divide(temp.groupby(axis=1,level=0).sum(),level=0)
         temp.plot(kind='barh',subplots=True,layout=(3,2),figsize=(9,9),sharey=True)
         plt.show()
```



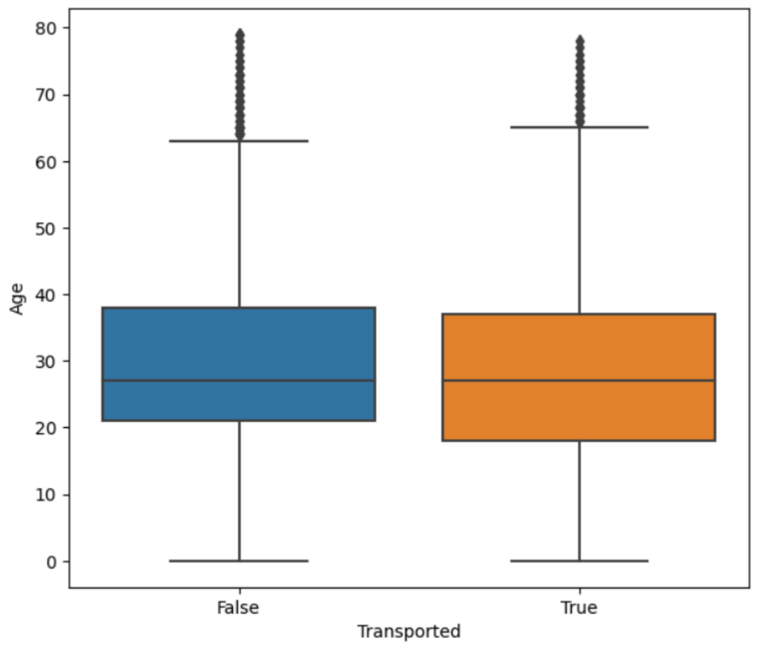
It can be Infered from above graph Cabin B and C are more dangerous regardless of Destination and people going on Trappist less likely to select A, B and C i.e Trappist people who doesn't select Cabin A, B and C have lower chance of being teleported

```
In [16]: sns.histplot(psdf["Age"],kde=True)
         plt.show()
```



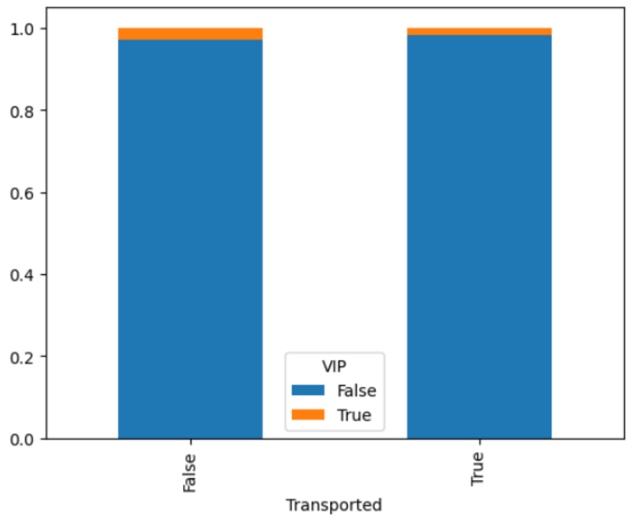
```
In [17]: psdf["Age"].fillna(psdf["Age"].median(),inplace=True)
         np.sum(psdf["Age"].isna())

Out[17]: 0
```

```
In [20]: fig,ax= plt.subplots(1,1,figsize=(7,6))
         sns.boxplot(psdf,x="Transported",y="Age",ax=ax)
         plt.show()
```



From above it can be seen that small kids have high chance of teleportation

```
In [21]: psdf["VIP"].fillna(False,inplace=True)
         temp = pd.crosstab(index=psdf["Transported"],columns=psdf["VIP"])
         temp.div(np.sum(temp,axis=1),axis=0).plot.bar(stacked=True)
         plt.show()
```



There doesn't seems to be much relation between transported and VIP feature

```
In [22]: temp = psdf.loc[psdf["VIP"]==True][["RoomService","FoodCourt","ShoppingMall","Spa","VRDeck"]]
         temp.sum()/temp.count()
```

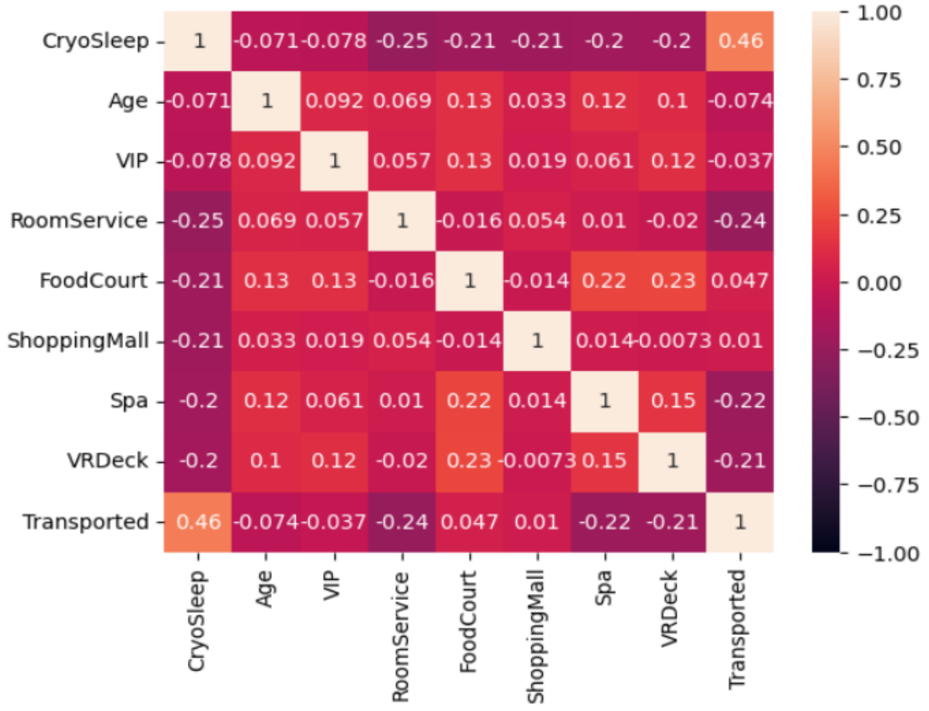
```
Out[22]: RoomService      473.615385
         FoodCourt       1811.393782
         ShoppingMall     247.726804
         Spa              760.710660
         VRDeck          1234.856410
         dtype: float64
```

```
In [23]: temp = psdf.loc[psdf["VIP"]==False][["RoomService","FoodCourt","ShoppingMall","Spa","VRDeck"]]
         temp.sum()/temp.count()
```

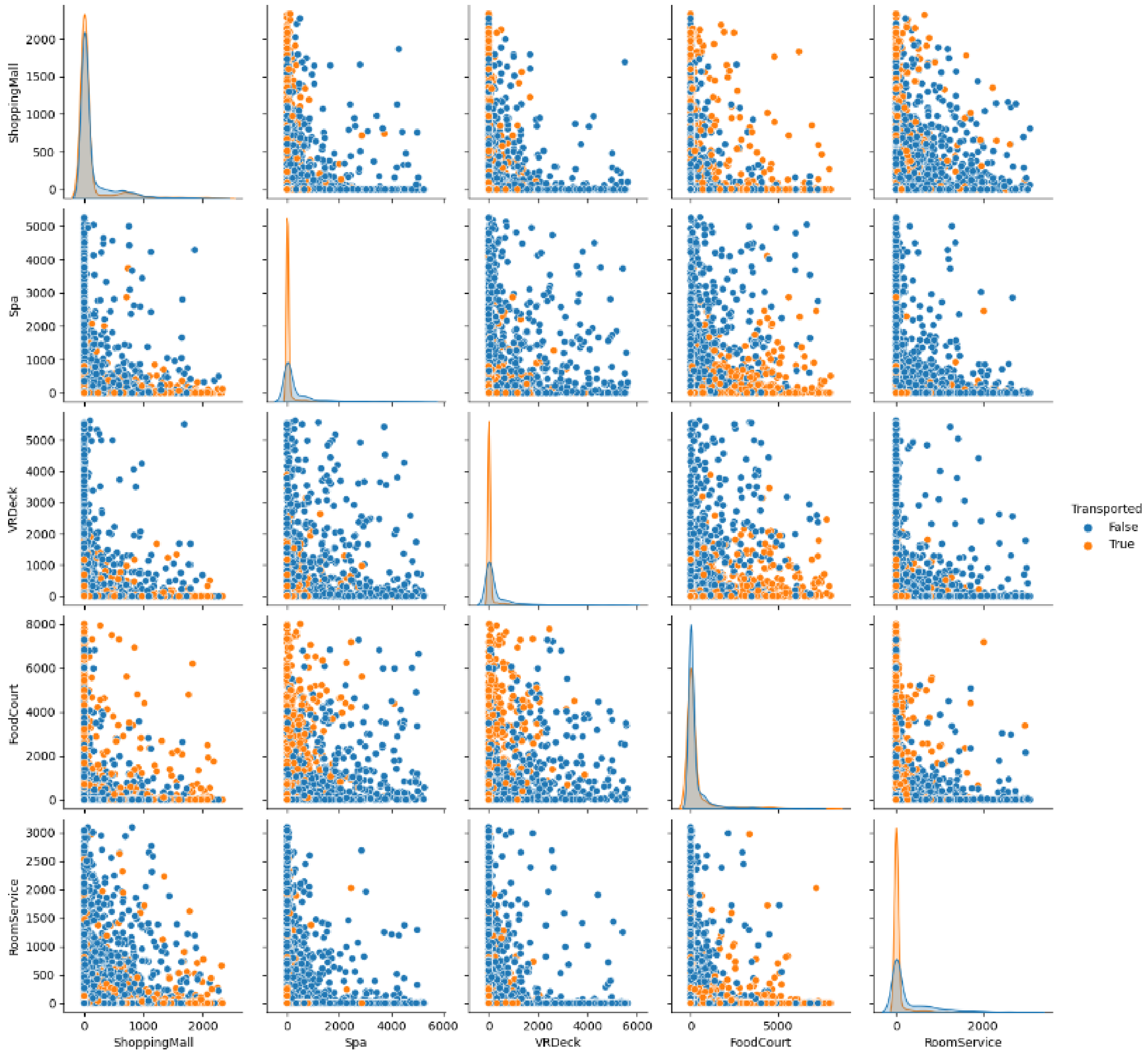
```
Out[23]: RoomService     218.851268
         FoodCourt       426.672839
         ShoppingMall    171.997708
         Spa             300.484903
         VRDeck          283.031649
         dtype: float64
```

VIP people are much likely to spend on RoomService, FoodCourt, ShoppingMall, Spa and VRDeck than normal people

```
In [24]: sns.heatmap(psdf.select_dtypes(exclude="object").corr(),annot=True,vmin=-1,vmax=1)
         plt.show()
```



```
In [25]: temp = psdf[["ShoppingMall","Spa","VRDeck","FoodCourt","Transported","RoomService"]]
         temp =  temp.loc[
             (temp["ShoppingMall"]<temp["ShoppingMall"].quantile(0.99))
             &(temp["Spa"]<temp["Spa"].quantile(0.99))
             &(temp["VRDeck"]<temp["VRDeck"].quantile(0.99))
             &(temp["FoodCourt"]<temp["FoodCourt"].quantile(0.99))
             &(temp["RoomService"]<temp["RoomService"].quantile(0.99))
         ]
         sns.pairplot(temp,diag_kind="kde",hue="Transported")
         plt.show()
```



Applying ML models

```
In [26]: train = pd.read_csv('/home/slowgamer/Desktop/College/sem7/BDA/project/spaceship/train.csv')
         test = pd.read_csv('/home/slowgamer/Desktop/College/sem7/BDA/project/spaceship/test.csv')
```

```
In [27]: STRATEGY = 'median'
         TARGET = 'Transported'
```

```
In [28]: imputer_cols = ["Age", "FoodCourt", "ShoppingMall", "Spa", "VRDeck" ,"RoomService"]
         imputer = SimpleImputer(strategy=STRATEGY )
         imputer.fit(train[imputer_cols])
         train[imputer_cols] = imputer.transform(train[imputer_cols])
         test[imputer_cols] = imputer.transform(test[imputer_cols])
         train["HomePlanet"].fillna('Z', inplace=True)
         test["HomePlanet"].fillna('Z', inplace=True)
```

```
In [29]: label_cols = ["HomePlanet", "CryoSleep","Cabin", "Destination" ,"VIP"]
         def label_encoder(train,test,columns):
             for col in columns:
                 train[col] = train[col].astype(str)
                 test[col] = test[col].astype(str)
                 train[col] = LabelEncoder().fit_transform(train[col])
                 test[col] =  LabelEncoder().fit_transform(test[col])
             return train, test

         train ,test = label_encoder(train,test ,label_cols)
```

```
In [30]: train.drop(["Name" ,"Cabin"] , axis = 1 ,inplace = True)
         test.drop(["Name" ,"Cabin"] , axis = 1 ,inplace = True)
         X = train.drop(TARGET , axis =1 )
         y = train[TARGET]
         X_train , X_test , y_train , y_test = train_test_split(X, y, random_state = 20, test_size =0.33)
```

```
In [31]: lr = LogisticRegression(solver='liblinear', random_state=0)
```

```
In [32]: lr.fit(X_train, y_train)
```

```
Out[32]: ▾                 LogisticRegression
         LogisticRegression(random_state=0, solver='liblinear')
```

```
In [33]: lr.classes_
```

```
Out[33]: array([False,  True])
```

```
In [34]: lr.intercept_
```

```
Out[34]: array([0.00066899])
```

```
In [35]: lr.coef_
```

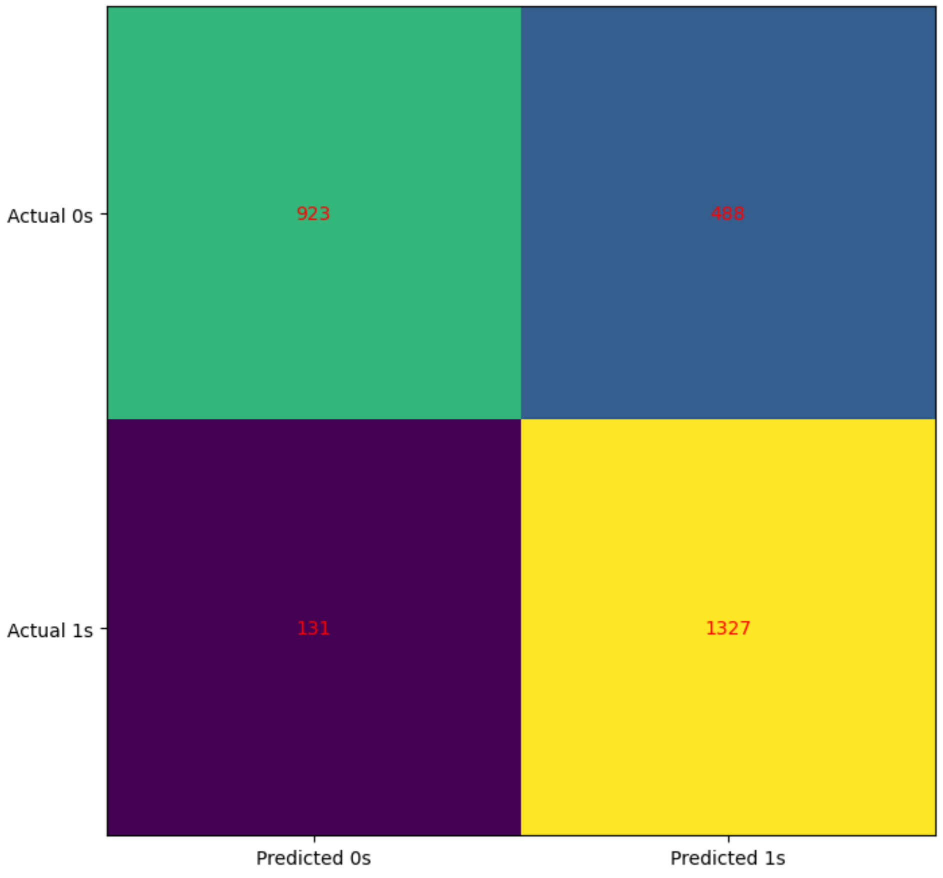
```
Out[35]: array([[ 6.74936409e-07,  2.84651910e-03,  2.05347410e-03,
                 -1.40958325e-04,  1.23454859e-02,  8.73426294e-05,
                 -2.05759484e-03,  6.93689225e-04,  2.42213681e-04,
                 -2.21878250e-03, -1.97441167e-03]])
```

```
In [36]: lr.predict_proba(X_test)
```

```
Out[36]: array([[0.30219825, 0.69780175],
                [0.08541266, 0.91458734],
                [0.21431645, 0.78568355],
                ...,
                [0.283211  , 0.716789  ],
                [0.42482896, 0.57517104],
                [0.29545708, 0.70454292]])
```

```
In [37]: lr.predict(X_test)

Out[37]: array([ True,  True,  True, ...,  True,  True,  True])

In [38]: lr.score(X_test, y_test)

Out[38]: 0.7842453816660857

In [39]: confusion_matrix(y_test, lr.predict(X_test))

Out[39]: array([[ 923,  488],
               [ 131, 1327]])

In [40]: lr.score(X_train, y_train)

Out[40]: 0.7704326923076923
```

```
In [41]: cm = confusion_matrix(y_test, lr.predict(X_test))

         fig, ax = plt.subplots(figsize=(8, 8))
         ax.imshow(cm)
         ax.grid(False)
         ax.xaxis.set(ticks=(0, 1), ticklabels=('Predicted 0s', 'Predicted 1s'))
         ax.yaxis.set(ticks=(0, 1), ticklabels=('Actual 0s', 'Actual 1s'))
         ax.set_ylim(1.5, -0.5)
         for i in range(2):
             for j in range(2):
                 ax.text(j, i, cm[i, j], ha='center', va='center', color='red')
         plt.show()
```



```
In [42]: print(classification_report(y_test, lr.predict(X_test)))

                       precision    recall  f1-score   support

                False       0.88      0.65      0.75      1411
                 True       0.73      0.91      0.81      1458

             accuracy                           0.78      2869
            macro avg       0.80      0.78      0.78      2869
         weighted avg       0.80      0.78      0.78      2869
```

Figure 3.1. Implementation and Result

CHAPTER 4
CONCLUSION AND FUTURE SCOPE

Through Data pre-processing, summarization, visualization and prediction of Spaceship Titanic data we were to gained insight of related features to transported feature, how multiple features vary each other and able to predict passengers who were transported with accuracy of 79% on test data. Pyspark, pandas, numpy, seaborn and scikit-learn came very handy for above mentioned process. From this insight we can further mixed some features to get more powerful features. With different models and more data we can achieve higher accuracy on this datasets and we can use this concept to predict which passenger have higher chances of getting into accident in real time and take precaution before any harm happen.

REFERENCES

[1] https://spark.apache.org/docs/latest/

[2] https://spark.apache.org/docs/latest/api/python/

[3] https://pandas.pydata.org/docs/

[4] https://numpy.org/doc/stable/reference/

[5] https://matplotlib.org/

[6] https://seaborn.pydata.org/api.html

[8] https://scikit-learn.org/stable/user_guide.html

[7] https://www.kaggle.com/competitions/spaceship-titanic