A REPORT ON

SURVIVAL PREDICTION ON TITANIC

By Royal Chaudhary

(12016265)

DECLARATION STATEMENT

I, **Royal Chaudhary (12016265)** and **Prakriti Verma (12018071)** hereby declare that the research work reported in the dissertation/dissertation proposal entitled " **SURVIVAL PREDICTION ON TITANIC"**
in partial fulfilment of the requirement for the award of Degree for Bachelor of Technology in Computer Science and Engineering at Lovely Professional University, Phagwara, Punjab is an authentic work carried out under supervision. I have not submitted this work elsewhere for any degree or diploma.

I understand that the work presented herewith is in direct compliance with Lovely Professional University's Policy on plagiarism, intellectual property rights, and highest standards of moral and ethical conduct. Therefore, to the best of my knowledge, the content of this dissertation represents authentic and honest research effort conducted, in its entirety, by me. I am fully responsible for the contents of my dissertation work.

Acknowledgement

I would like to take this opportunity to express my gratitude to all those who have helped me in completing this machine learning project on survival prediction on the Titanic.

First and foremost, I would like to thank my supervisor/mentor Mr. Ved Prakash Dubey, who provided me with guidance and support throughout the project. Their feedback and suggestions were invaluable in shaping the direction of this project and ensuring its successful completion.

I would also like to thank the authors of the various Python libraries and tools that I used in this project, including Pandas, NumPy, Scikit-learn, and Matplotlib. These libraries and tools made it possible for me to efficiently analyse and visualize the data, and to develop and evaluate my machine learning model.

I am also grateful to the organizers of the Kaggle Titanic competition, which provided the dataset that I used for this project. The competition and the associated community forums were a great resource for learning and for getting inspiration for new approaches and techniques.

Index

Objective and scope of the Project

The Titanic disaster is a tragic event that has captured the attention of the public for over a century. The sinking of the Titanic was one of the deadliest maritime disasters in history, and it remains a significant event for historians, social scientists, and data scientists to study.

One of the primary objectives of a machine learning project on survival prediction on the Titanic is to use machine learning models to analyze and predict the likelihood of survival for passengers based on various factors such as age, sex, class, fare, and cabin. By doing so, the project can identify patterns and trends in the data that might have contributed to survival rates.

The scope of the project can involve collecting data from various sources such as passenger lists, crew lists, and other historical documents. The data can be cleaned and pre-processed to ensure that it is suitable for analysis. This can involve filling in missing values, removing duplicates, and encoding categorical variables.

Exploratory data analysis techniques can be used to identify patterns and relationships between variables in the data. Feature engineering techniques such as one-hot encoding, scaling, and transformation can be used to create new features that might improve the performance of the machine learning models.

Several machine learning models such as decision trees, random forests, logistic regression, and neural networks can be used to build predictive models. The models can be trained and tested using cross-validation techniques to evaluate their performance and select the best model.

The project can also involve creating visualizations to communicate the insights from the analysis and the predictive models. For example, scatterplots, heatmaps, and bar charts can be used to visualize the relationships between variables and the performance of the models.

The project's potential applications are significant. For example, the insights gained from the analysis can be used to improve the safety measures on ships, such as improving the number of lifeboats or better passenger education about emergency procedures. The project can also inform emergency response strategies in similar situations, such as natural disasters or pandemics.

Overall, the objective of the project is to use machine learning to learn from the past and help prevent tragedies in the future. The project's scope can involve collecting, cleaning, and preprocessing data, performing exploratory data analysis and feature engineering, building predictive models, evaluating their performance, and creating visualizations to communicate the insights.

Introduction

The sinking of the Titanic is an event that has captivated people's imagination for over a century. The story of the "unsinkable" ship that hit an iceberg and sank, leading to the loss of many lives, has become a symbol of human hubris and tragedy. The Titanic was a massive ship that was considered a marvel of engineering at the time, and its sinking was a shock to the world. Since then, the Titanic has become one of the most studied maritime disasters in history.

One aspect of the Titanic disaster that has received a lot of attention in recent years is the question of who survived and who did not. There were many factors that may have influenced a person's chances of survival, such as their age, sex, ticket class, and whether they were traveling alone or with family members. Researchers have attempted to use machine learning algorithms to predict which passengers on the Titanic would have survived based on these factors.

The goal of this project is to develop a machine learning model that can accurately predict survival on the Titanic. This is a challenging problem because it involves predicting a binary outcome (survival or death) based on a set of input features (such as age, sex, ticket class, and so on). To develop our model, we will make use of a dataset that contains information about 891 passengers who were onboard the Titanic when it sank.

The dataset includes a variety of information about each passenger, such as their age, sex, ticket class, and whether or not they survived the disaster. We will use this dataset to train our machine learning model, and then evaluate its performance using various metrics such as accuracy, precision, and recall.

To develop our machine learning model, we will use a variety of different techniques and algorithms. This will include data cleaning and pre-processing, where we will clean the dataset and prepare it for analysis. We will also perform feature engineering, where we will extract useful features from the data that will help our model make better predictions. We will then select an appropriate model architecture and train it on the dataset. Finally, we will tune the model's hyperparameters to optimize its performance.

Throughout the project, we will make use of various Python libraries and tools, such as Pandas, NumPy, Scikit-learn, and Matplotlib. These tools will allow us to efficiently analyze and visualize the data, and to develop and evaluate our machine learning model.

The potential applications of this project are numerous. For example, such a model could be used to help researchers better understand the factors that contributed to survival during the Titanic disaster. It could also be used to inform emergency response planning for similar disasters in the future. Additionally, this project could serve as a valuable educational tool for students who are interested in machine learning, data science, and statistics.

In conclusion, the development of a machine learning model to predict survival on the Titanic is an interesting and challenging problem that has captured the attention of researchers and enthusiasts alike. Through this project, we hope to contribute to the growing body of knowledge about the Titanic disaster and to showcase the potential of machine learning as a tool for solving complex problems in a variety of domains.

Hardware and Software used

Hardware:

The choice of hardware for a machine learning project can have a significant impact on the performance of the algorithms and the time required for training and inference. The following are the main hardware components used in a typical machine learning project:

- CPU: A Central Processing Unit (CPU) is the primary component of a computer that performs most of the processing tasks. It is capable of performing a wide range of tasks, including machine learning, but its processing power is often limited when working with large datasets or complex models. In a machine learning project on the Titanic dataset, a CPU with at least 4 cores and 8 threads, such as an Intel Core i5 or i7, would be sufficient.

- GPU: A Graphics Processing Unit (GPU) is a specialized hardware component designed to handle the parallel processing tasks required for machine learning. GPUs are particularly useful for deep learning, which involves training deep neural networks with many layers. A GPU can perform hundreds or thousands of computations in parallel, which can significantly speed up the training process. In a machine learning project on the Titanic dataset, a GPU with at least 4GB of VRAM, such as an NVIDIA GTX 1650, would be sufficient.

- RAM: Random Access Memory (RAM) is used to store data and code in a machine learning project. Sufficient RAM is required to load and manipulate the dataset in memory during the data cleaning, pre-processing, and model training stages. In a machine learning project on the Titanic dataset, a minimum of 8GB of RAM is recommended, but 16GB or higher is preferred for larger datasets or more complex models.

- Storage: Adequate storage is needed to store the dataset, the trained machine learning models, and the project code. In a machine learning project on the Titanic dataset, at least 100GB of storage space would be sufficient, but more may be required if working with larger datasets or models.

Software:

The software components used in a machine learning project are equally important as the hardware components. The following are some of the essential software tools and libraries used in a typical machine learning project:

- Python: Python is a popular programming language for machine learning due to its ease of use, large community support, and availability of libraries such as NumPy, Pandas, and Scikit-learn.

- Google Colaboratory: Google Colaboratory is a web-based interactive computing environment that allows for easy exploration and prototyping of machine learning models. It enables users to create and share documents that contain code, equations, visualizations, and narrative text.

- NumPy: NumPy is a Python library that provides support for numerical operations on large datasets, such as matrix operations, random number generation, and statistical analysis. It is the fundamental package for scientific computing with Python.

- Pandas: Pandas is a Python library that provides support for data manipulation and analysis, such as reading and writing data, cleaning and pre-processing data, and feature engineering. It provides easy-to-use data structures and data analysis tools for handling tabular data.

- Scikit-learn: Scikit-learn is a Python library that provides support for machine learning algorithms, such as linear regression, logistic regression, decision trees, and random forests. It is a widely used library for machine learning in Python and provides a consistent interface for many machine learning algorithms.

- Matplotlib: Matplotlib is a Python library that provides support for data visualization, such as creating plots, histograms, and scatterplots. It is a powerful tool for visualizing data and communicating insights to stakeholders.

Methodology/Flowchart and Algorithm implemented

The methodology for the machine learning project on survival prediction on the Titanic can be broken down into several steps, as shown in the following flowchart:

The steps involved in the methodology are as follows:

1. **Data collection**: The first step in the methodology is to collect the data. In this project, we will use a dataset that contains information of 1309 passengers who were onboard the Titanic when it sank.

Importing Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix
```

Data Collection & Processing

```
# load the data from csv file to Pandas DataFrame
titanic_data = pd.read_csv('Titanic_Final.csv')
```

```
[61] # printing the first 5 rows of the dataframe
titanic_data.head()
```

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

```
[62] # number of rows and Columns
titanic_data.shape
```

```
(1309, 12)
```

```
[63] # getting some informations about the data
     titanic_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  1309 non-null   int64
 1   Survived     1309 non-null   int64
 2   Pclass       1309 non-null   int64
 3   Name         1309 non-null   object
 4   Sex          1309 non-null   object
 5   Age          1046 non-null   float64
 6   SibSp        1309 non-null   int64
 7   Parch        1309 non-null   int64
 8   Ticket       1309 non-null   object
 9   Fare         1308 non-null   float64
 10  Cabin        295 non-null    object
 11  Embarked     1307 non-null   object
dtypes: float64(2), int64(5), object(5)
memory usage: 122.8+ KB
```

```
# check the number of missing values in each column
titanic_data.isnull().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            263
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin         1014
Embarked         2
dtype: int64
```

2. **Data cleaning and pre-processing**: The next step is to clean and pre-process the data. This involves handling missing values, dealing with outliers, and converting categorical variables into numerical ones.

Handling the Missing values

```
[65] # drop the "Cabin" column from the dataframe
     titanic_data = titanic_data.drop(columns='Cabin', axis=1)
```

```
# replacing the missing values in "Age" column with mean value
titanic_data['Age'].fillna(titanic_data['Age'].mean(), inplace=True)
```

```
[67] # finding the mode value of "Embarked" column
     print(titanic_data['Embarked'].mode())
```

```
0    S
Name: Embarked, dtype: object
```

```
[68] print(titanic_data['Embarked'].mode()[0])
```

```
S
```

```
[69] # replacing the missing values in "Embarked" column with mode value
     titanic_data['Embarked'].fillna(titanic_data['Embarked'].mode()[0], inplace=True)
```

```
[67] # finding the mode value of "Embarked" column
     print(titanic_data['Embarked'].mode())

0    S
Name: Embarked, dtype: object

[68] print(titanic_data['Embarked'].mode()[0])

S

[69] # replacing the missing values in "Embarked" column with mode value
     titanic_data['Embarked'].fillna(titanic_data['Embarked'].mode()[0], inplace=True)

     # check the number of missing values in each column
     titanic_data.isnull().sum()

PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       0
dtype: int64
```

3.**Data Exploration**: In this step, we explore the dataset to gain insights into the data and identify any patterns or relationships that could be useful in predicting survival. This may involve visualizations, statistical analysis, or machine learning algorithms.

Data Analysis

```
[71] # getting some statistical measures about the data
     titanic_data.describe()
```

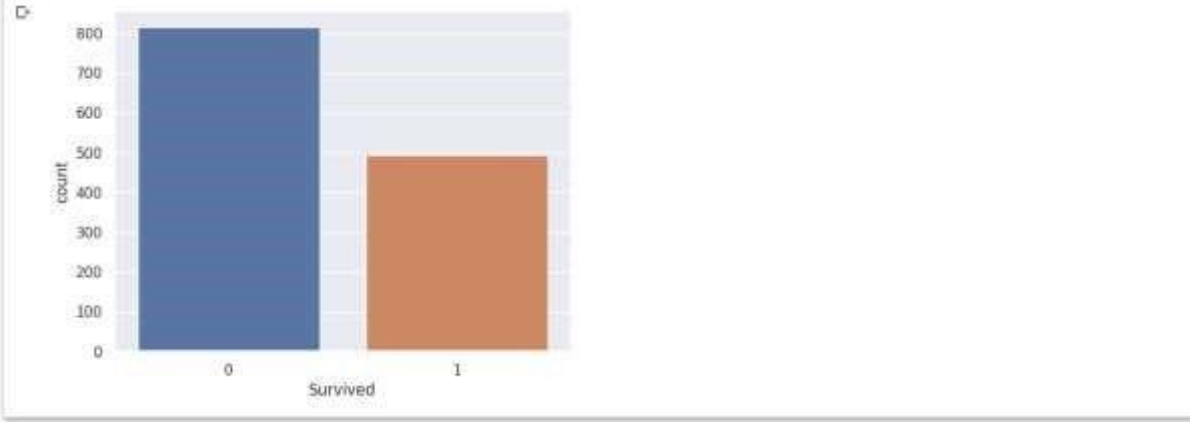
	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	1309.000000	1309.000000	1309.000000	1309.000000	1309.000000	1309.000000	1309.000000
mean	655.000000	0.377387	2.294882	29.881138	0.498854	0.385027	33.270043
std	378.020061	0.484918	0.837836	12.883193	1.041658	0.865560	51.747063
min	1.000000	0.000000	1.000000	0.170000	0.000000	0.000000	0.000000
25%	328.000000	0.000000	2.000000	22.000000	0.000000	0.000000	7.895800
50%	655.000000	0.000000	3.000000	29.881138	0.000000	0.000000	14.454200
75%	982.000000	1.000000	3.000000	35.000000	1.000000	0.000000	31.275000
max	1309.000000	1.000000	3.000000	80.000000	8.000000	9.000000	512.329200

```
     # finding the number of people survived and not survived
     titanic_data['Survived'].value_counts()

0    815
1    494
Name: Survived, dtype: int64
```

DATA VISUALISATION

```
[73] sns.set()
```

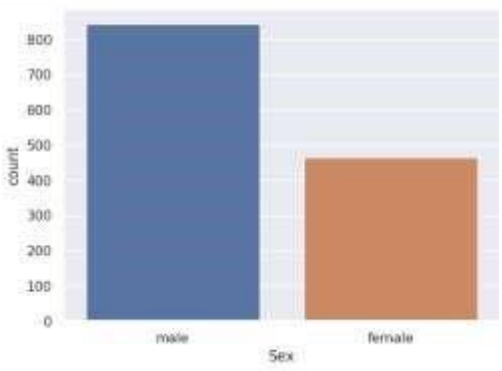
```
# making a count plot for "Survived" column
sns.countplot(x= 'Survived', data= titanic_data)
plt.show()
```




```
[75] titanic_data['Sex'].value_counts()
```

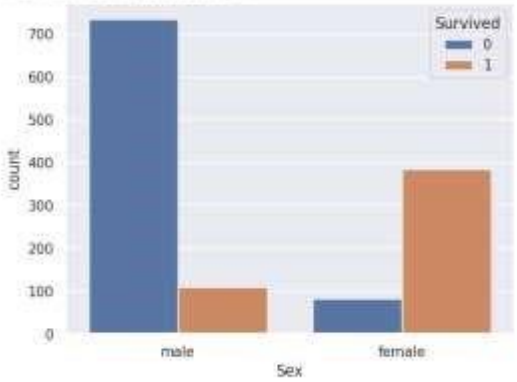
```
male      843
female    466
Name: Sex, dtype: int64
```

```
# making a count plot for "Sex" column
sns.countplot(x = 'Sex', data = titanic_data)
plt.show()
```



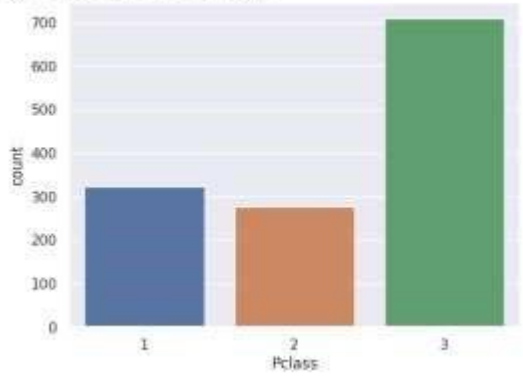

```
# number of survivors Gender wise
sns.countplot(x= 'Sex', hue='Survived', data=titanic_data)
```

```
<Axes: xlabel='Sex', ylabel='count'>
```



```
# making a count plot for "Pclass" column
sns.countplot(x='Pclass', data=titanic_data)
```

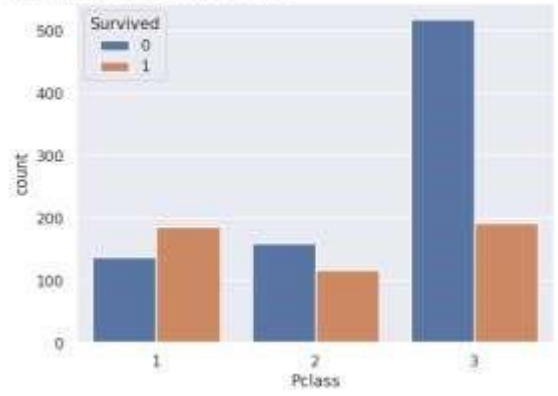
```
<Axes: xlabel='Pclass', ylabel='count'>
```



+ Code + Text

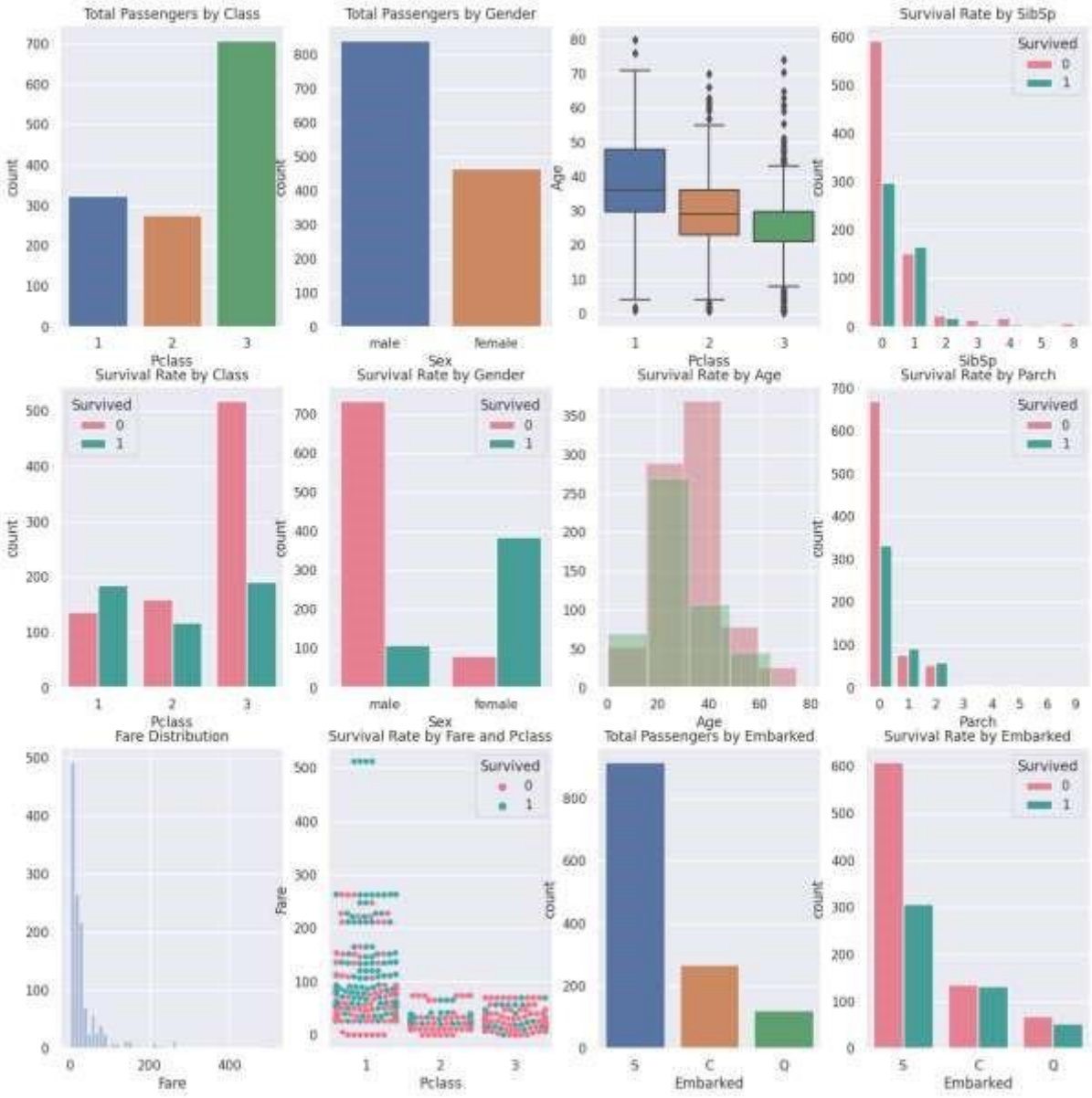
```
[78] #number of Survivors Class wise
     sns.countplot(x='Pclass', hue='Survived', data=titanic_data)
```

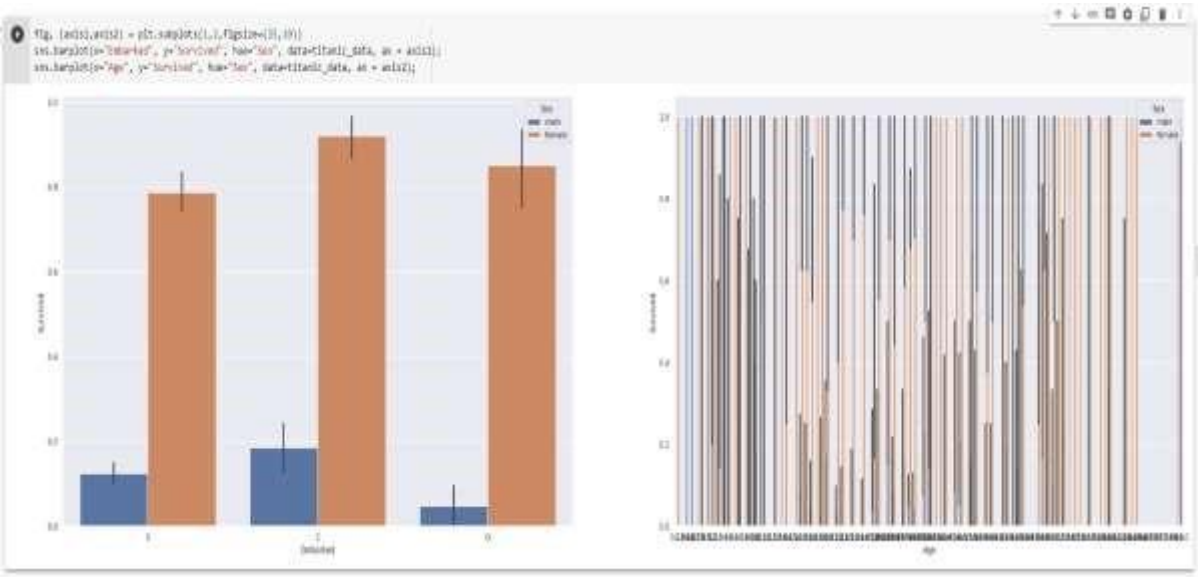
```
<Axes: xlabel='Pclass', ylabel='count'>
```




```
f,ax = plt.subplots(3, 4,figsize=(20,16))
sns.countplot(x='Pclass',data=titanic_data,ax=ax[0,0])
sns.countplot(x='Sex',data=titanic_data,ax=ax[0,1])
sns.boxplot(x='Pclass',y='Age',data=titanic_data,ax=ax[0,2])
sns.countplot(x='SibSp',hue='Survived',data=titanic_data,ax=ax[0,3],palette='husl')
sns.distplot(titanic_data['Fare'].dropna(),ax=ax[1,0],kde=False,color='b')
sns.countplot(x='Embarked',data=titanic_data,ax=ax[1,1])

sns.countplot(x='Pclass',hue='Survived',data=titanic_data,ax=ax[1,0],palette='husl')
sns.countplot(x='Sex',hue='Survived',data=titanic_data,ax=ax[1,1],palette='husl')
sns.distplot(titanic_data[titanic_data['Survived']==0]['Age'].dropna(),ax=ax[1,2],kde=False,color='r',bins=5)
sns.distplot(titanic_data[titanic_data['Survived']==1]['Age'].dropna(),ax=ax[1,2],kde=False,color='g',bins=5)
sns.countplot(x='Parch',hue='Survived',data=titanic_data,ax=ax[1,3],palette='husl')
sns.swarmplot(x='Pclass',y='Fare',hue='Survived',data=titanic_data,palette='husl',ax=ax[2,1])
sns.countplot(x='Embarked',hue='Survived',data=titanic_data,ax=ax[2,2],palette='husl')

ax[0,0].set_title('Total Passengers by Class')
ax[0,1].set_title('Total Passengers by Gender')
ax[0,3].set_title('Survival Rate by SibSp')
ax[1,0].set_title('Survival Rate by Class')
ax[1,1].set_title('Survival Rate by Gender')
ax[1,2].set_title('Survival Rate by Age')
ax[1,3].set_title('Survival Rate by Parch')
ax[2,0].set_title('Fare Distribution')
ax[2,1].set_title('Survival Rate by Fare and Pclass')
ax[2,2].set_title('Total Passengers by Embarked')
ax[2,3].set_title('Survival Rate by Embarked')
```

```
<ipython-input-80-901e02514745>:6: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(titanic_data['Fare'].dropna(),ax=ax[2,0],kde=False,color='b')
```


Total Passengers by Class
Total Passengers by Gender
Survival Rate by SibSp
Survived
Pclass
Sex
Age
SibSp
Survival Rate by Class
Survival Rate by Gender
Survival Rate by Age
Survival Rate by Parch
Parch
Fare Distribution
Survival Rate by Fare and Pclass
Total Passengers by Embarked
Survival Rate by Embarked
Fare
Embarked
male
female
count

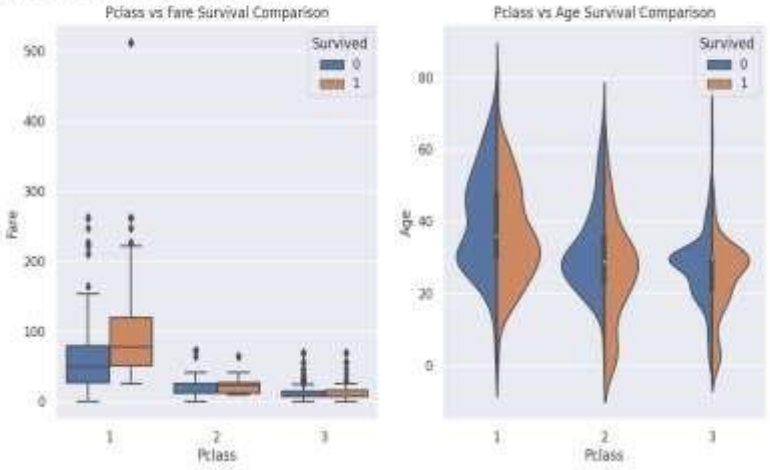


Observations for Age graph:

- 0 or blue represent women; 1 or orange represent men. Gender and age seem to have a stronger influece of the survival rate.
- We start to find where most survivors are: older women (48 to 64 year old), and younger passengers.
- What is statistically interesting is that only young boys (Age Category = 0) have high survival rates, unlike other age groups for men.

```
#graph distribution of qualitative data: Pclass
fig, (axis1,axis2) = plt.subplots(1,2,figsize=(12,6))

sns.boxplot(x = 'Pclass', y = 'Fare', hue = 'Survived', data = titanic_data, ax = axis1)
axis1.set_title('Pclass vs Fare Survival Comparison')

sns.violinplot(x = 'Pclass', y = 'Age', hue = 'Survived', data = titanic_data, split = True, ax = axis2)
axis2.set_title('Pclass vs Age Survival Comparison')
```

```
Text(0.5, 1.0, 'Pclass vs Age Survival Comparison')
```



Heatmap

```
[83] cols = ["Survived","Pclass","Sex","Age","SibSp","Parch","Fare","Embarked"]
```

```
[84]
titanic_corr = titanic_data[cols].corr()
```

```
<ipython-input-84-1a58e3fa4b11>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to sile
  titanic_corr = titanic_data[cols].corr()
```

```
[85] titanic_corr
```

	Survived	Pclass	Age	SibSp	Parch	Fare
Survived	1.000000	-0.264710	-0.049483	0.002370	0.159819	0.255914
Pclass	-0.264710	1.000000	-0.360271	0.060832	0.018322	-0.550803
Age	-0.049483	-0.360271	1.000000	-0.190747	-0.130872	0.170525
SibSp	0.002370	0.060832	-0.190747	1.000000	0.373587	0.160434
Parch	0.159819	0.018322	-0.130872	0.373587	1.000000	0.221706
Fare	0.255914	-0.550803	0.170525	0.160434	0.221706	1.000000

```
fig,(ax1,ax2) = plt.subplots(nrows=2,figsize=(10,10))
sns.heatmap(titanic_corr,annot=True,ax=ax1,cbar=False)
sns.heatmap(titanic_corr,annot=True,ax=ax2,cbar=True)
ax1.set_title("cbar=False")
ax2.set_title("cbar=True")
```

```
Text(0.5, 1.0, 'cbar=True')
```

Encoding the Categorical Columns

```
[87] titanic_data['Sex'].value_counts()
```

```
male    843
female  466
Name: Sex, dtype: int64
```

Double-click (or enter) to edit

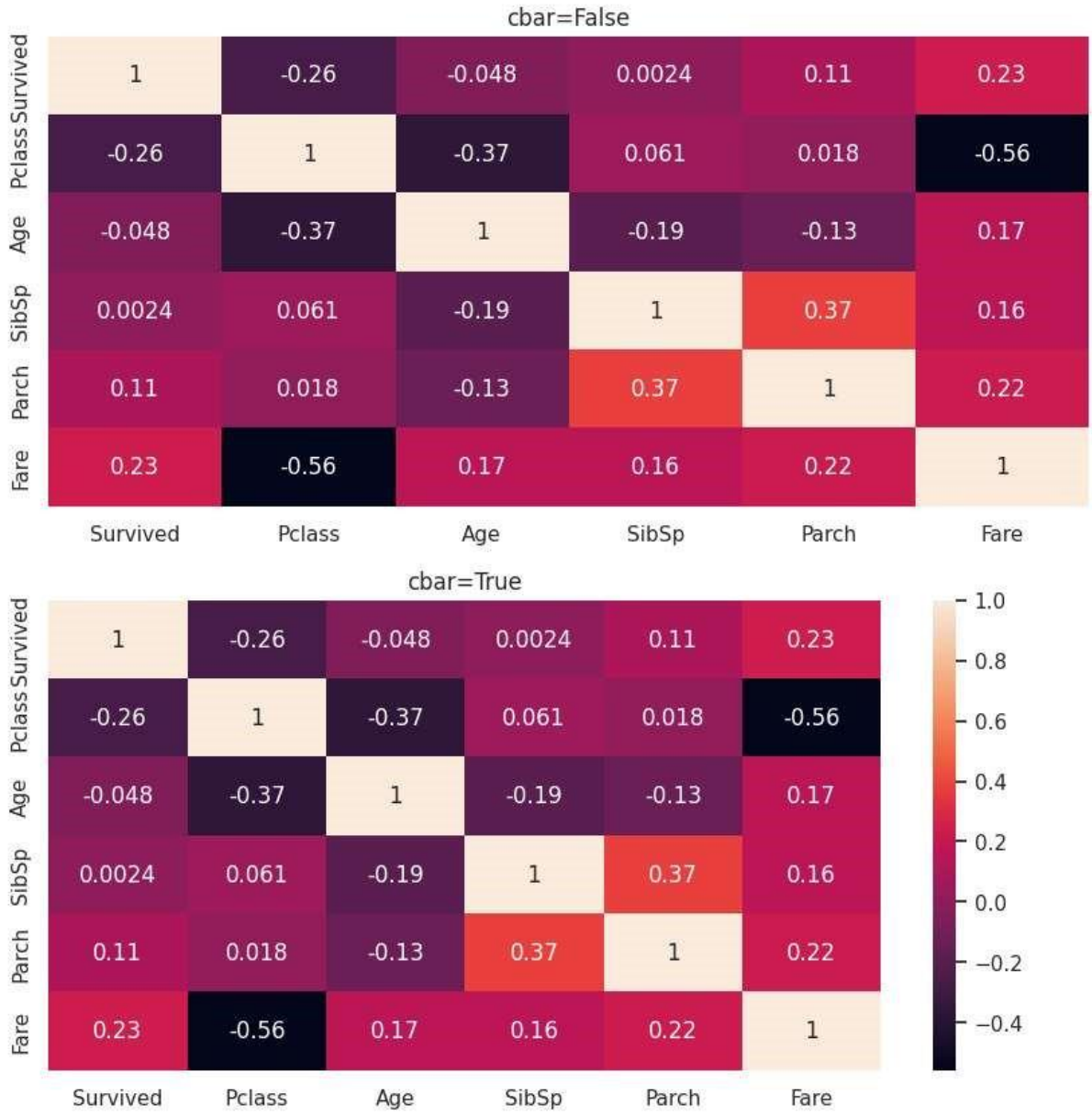
```
[88] titanic_data['Embarked'].value_counts()
```

```
S    916
C    270
Q    123
Name: Embarked, dtype: int64
```

```
# converting categorical Columns

titanic_data.replace({'Sex':{'male':0,'female':1}, 'Embarked':{'S':0,'C':1,'Q':2}}, inplace=True)
```

```
[90] titanic_data.head(10)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	1	0	3	Braund, Mr. Owen Harris	0	22.000000	1	0	A/5 21171	7.2500	0
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	38.000000	1	0	PC 17599	71.2833	1
2	3	1	3	Heikkinen, Miss. Laina	1	26.000000	0	0	STON/O2. 3101282	7.9250	0
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	1	35.000000	1	0	113803	53.1000	0
4	5	0	3	Allen, Mr. William Henry	0	35.000000	0	0	373450	8.0500	0
5	6	0	3	Moran, Mr. James	0	29.881138	0	0	330877	8.4583	2
6	7	0	1	McCarthy, Mr. Timothy J	0	54.000000	0	0	17463	51.8625	0



4. **Feature engineering**: After cleaning and pre-processing the data, we can begin feature engineering. This involves selecting the most important features that will help our machine learning model make accurate predictions. In this project, we will use features such as age, sex, ticket class, and whether or not the passenger had any family members onboard.

Encoding the Categorical Columns

```
[87] titanic_data['Sex'].value_counts()

male      843
female    466
Name: Sex, dtype: int64
```

Double-click (or enter) to edit

```
[88] titanic_data['Embarked'].value_counts()

S    916
C    270
Q    123
Name: Embarked, dtype: int64
```

```
# converting categorical Columns

titanic_data.replace({'Sex':{'male':0,'female':1}, 'Embarked':{'S':0,'C':1,'Q':2}}, inplace=True)
```

```
[90] titanic_data.head(10)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	1	0	3	Braund, Mr. Owen Harris	0	22.000000	1	0	A/5 21171	7.2500	0
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	38.000000	1	0	PC 17599	71.2833	1
2	3	1	3	Heikkinen, Miss. Laina	1	26.000000	0	0	STON/O2. 3101282	7.9250	0
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	1	35.000000	1	0	113803	53.1000	0
4	5	0	3	Allen, Mr. William Henry	0	35.000000	0	0	373450	8.0500	0
5	6	0	3	Moran, Mr. James	0	29.881138	0	0	330877	8.4583	2
6	7	0	1	McCarthy, Mr. Timothy J	0	54.000000	0	0	17463	51.8625	0

Separating features & Target

```
[91] X = titanic_data.drop(columns = ['PassengerId','Name','Ticket','Survived'],axis=1)
     Y = titanic_data['Survived']
```

```
[92] print(X)

      Pclass  Sex        Age  SibSp  Parch      Fare  Embarked
0          3    0  22.000000      1      0    7.2500         0
1          1    1  38.000000      1      0   71.2833         1
2          3    1  26.000000      0      0    7.9250         0
3          1    1  35.000000      1      0   53.1000         0
4          3    0  35.000000      0      0    8.0500         0
...      ...  ...        ...    ...    ...       ...       ...
1304       3    0  29.881138      0      0    8.0500         0
1305       1    1  39.000000      0      0  108.9000         1
1306       3    0  38.500000      0      0    7.2500         0
1307       3    0  29.881138      0      0    8.0500         0
1308       3    0  29.881138      1      1   22.3583         1

[1309 rows x 7 columns]
```

```
print(Y)

0       0
1       1
2       1
3       1
4       0
       ..
1304    0
1305    1
1306    0
1307    0
1308    0
Name: Survived, Length: 1309, dtype: int64
```

5. **Splitting the Dataset into train and test**: Once we are done with the feature engineering, we split the data preferably into 80-20 for training and testing the machine learning models.

Splitting the data into training data & Test data

```
[94] X_train, x_test, Y_train, y_test = train_test_split(X,Y, test_size=0.2, random_state=2)

[95] print(X.shape, X_train.shape, x_test.shape)

     (1309, 7) (1047, 7) (262, 7)
```

5. **Model selection**: Once we have selected the features, we can begin model selection. This involves selecting an appropriate machine learning algorithm that will be able to predict survival on the Titanic. In this project, we will consider several algorithms, including logistic regression, decision trees, random forests, naive bayes, and linear SVCs.

a) Logistic Regression
b) Random forest
c) Naïve Bayes Classifier
d) Decision Tree
e) K-Nearest Neighbor (K-NN)
f) Support Vector Machines

6. **Model evaluation**: Once we have trained and tuned our model, we can evaluate its performance. This involves using metrics such as accuracy, precision, and recall to assess how well the model is able to predict survival on the Titanic.

1. Logistic Regression

Logistic regression measures the relationship between the categorical dependent feature (in our case Survived) and the other independent features. It estimates probabilities using a cumulative logistic distribution:

- The first value shows the accuracy of this model
- The table after this shows the importance of each feature according this classifier.

Model Training

+ Code + Text

Logistic Regression

```
[96] logreg = LogisticRegression()
     logreg.fit(X, Y)
     Y_pred1 = logreg.predict(x_test)
     acc_log = round(logreg.score(x_test, y_test) * 100, 2)
     acc_log
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
86.64
```

```
[97] from sklearn.metrics import confusion_matrix, classification_report
     print(classification_report(y_test, Y_pred1))
     cm = pd.DataFrame(confusion_matrix(y_test, Y_pred1), ['Actual: NOT', 'Actual: SURVIVED'], ['Predicted: NO', 'Predicted: SURVIVED'])
     print(cm)
```

```
              precision    recall  f1-score   support

           0       0.88      0.92      0.90       165
           1       0.84      0.78      0.81        97

    accuracy                           0.87       262
   macro avg       0.86      0.85      0.85       262
weighted avg       0.87      0.87      0.87       262

                Predicted: NO  Predicted: SURVIVED
Actual: NOT               151                   14
Actual: SURVIVED           21                   76
```

2. Random forest

This is one of the most popular classifier. Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees (n_estimators=100) at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees

Random Forest

```
[98] from sklearn.ensemble import RandomForestClassifier
```

```
[99] random_forest = RandomForestClassifier(n_estimators=100)
     random_forest.fit(X_train, Y_train)
     random_forest_predictions = random_forest.predict(x_test)
     acc_random_forest = round(random_forest.score(x_test, y_test) * 100, 2)


     # Preparing data for Submission 2
     test_Survived = pd.Series(random_forest_predictions, name="Survived")


     acc_random_forest
```

```
79.77
```

```
     print(classification_report(y_test, random_forest_predictions))
     cm = pd.DataFrame(confusion_matrix(y_test, random_forest_predictions), ['Actual: NOT', 'Actual: SURVIVED'], ['Predicted: NO', 'Predicted: SURVIVED'])
     print(cm)
```

```
              precision    recall  f1-score   support

           0       0.83      0.86      0.84       165
           1       0.74      0.69      0.72        97

    accuracy                           0.80       262
   macro avg       0.79      0.78      0.78       262
weighted avg       0.80      0.80      0.80       262

                Predicted: NO  Predicted: SURVIVED
Actual: NOT               142                   23
Actual: SURVIVED           30                   67
```

3. Naïve Bayes Classifier

This is a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features. Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of features in a learning problem.

Naive Bayes Classifier

```
[104]
from sklearn.naive_bayes import GaussianNB
```

```
[105] gaussian = GaussianNB()
gaussian.fit(X_train, Y_train)
Y_pred3 = gaussian.predict(x_test)
acc_gaussian = round(gaussian.score(x_test, y_test) * 100, 2)
acc_gaussian
```

```
85.11
```

```
print(classification_report(y_test, Y_pred3))
cm = pd.DataFrame(confusion_matrix(y_test, Y_pred3), ['Actual: NOT', 'Actual: SURVIVED'], ['Predicted: NO', 'Predicted: SURVIVED'])
print(cm)
```

```
              precision    recall  f1-score   support

           0       0.87      0.90      0.88       165
           1       0.82      0.77      0.79        97

    accuracy                           0.85       262
   macro avg       0.84      0.84      0.84       262
weighted avg       0.85      0.85      0.85       262

                 Predicted: NO  Predicted: SURVIVED
Actual: NOT                148                   17
Actual: SURVIVED            22                   75
```

4. Decision Tree

This predictive model maps features (tree branches) to conclusions about the target value (tree leaves).

The target features take a finite set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees.

Decision Tree

```
[101] from sklearn.tree import DecisionTreeClassifier

[102] decision_tree = DecisionTreeClassifier()
      decision_tree.fit(X_train, Y_train)
      Y_pred7 = decision_tree.predict(x_test)
      acc_decision_tree = round(decision_tree.score(x_test, y_test) * 100, 2)
      acc_decision_tree

      77.86

      print(classification_report(y_test, Y_pred7))
      cm = pd.DataFrame(confusion_matrix(y_test, Y_pred7), ['Actual: NOT', 'Actual: SURVIVED'], ['Predicted: NO', 'Predicted: SURVIVED'])
      print(cm)

                    precision    recall  f1-score   support

                 0       0.82      0.84      0.83       165
                 1       0.71      0.68      0.69        97

          accuracy                           0.78       262
         macro avg       0.76      0.76      0.76       262
      weighted avg       0.78      0.78      0.78       262

                        Predicted: NO  Predicted: SURVIVED
      Actual: NOT                 138                   27
      Actual: SURVIVED             31                   66
```

5. K-Nearest Neighbours

This is a non-parametric method used for classification and regression. A sample is classified by a majority vote of its neighbors, with the sample being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.

K-Nearest Neighbors Algorithm

```
[110] from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(algorithm='auto', leaf_size=26, metric='minkowski',
                           metric_params=None, n_jobs=1, n_neighbors=10, p=2,
                           weights='uniform')


knn.fit(X_train, Y_train)
knn_predictions = knn.predict(x_test)
acc_knn = round(knn.score(x_test, y_test) * 100, 2)

# Preparing data for Submission 1
test_Survived = pd.Series(knn_predictions, name="Survived")
# Submission1 = pd.concat([PassengerId,test_Survived],axis=1)
acc_knn
```

```
63.36
```

```
[112] print(classification_report(y_test, knn_predictions))
      cm = pd.DataFrame(confusion_matrix(y_test, knn_predictions), ['Actual: NOT', 'Actual: SURVIVED'], ['Predicted: NO', 'Predicted: SURVIVED'])
      print(cm)
```

```
              precision    recall  f1-score   support

           0       0.67      0.83      0.74       165
           1       0.51      0.30      0.38        97

    accuracy                           0.63       262
   macro avg       0.59      0.56      0.56       262
weighted avg       0.61      0.63      0.61       262

                  Predicted: NO  Predicted: SURVIVED
Actual: NOT                 137                   28
Actual: SURVIVED             68                   29
```

6. Support Vector Machines (SVM)

Given a set of training samples, each sample is marked as belonging to one or the other of two categories.

The SVM training algorithm builds a model that assigns new test samples to one category or the other, making it a non-probabilistic binary linear classifier.

Support Vector Machine (SVM)

```
[114] from sklearn import svm
      from sklearn.svm import SVC

svc=SVC()
svc.fit(X_train, Y_train)
Y_pred2 = svc.predict(x_test)
acc_svc = round(svc.score(x_test, y_test) * 100, 2)
acc_svc
```

```
63.74
```

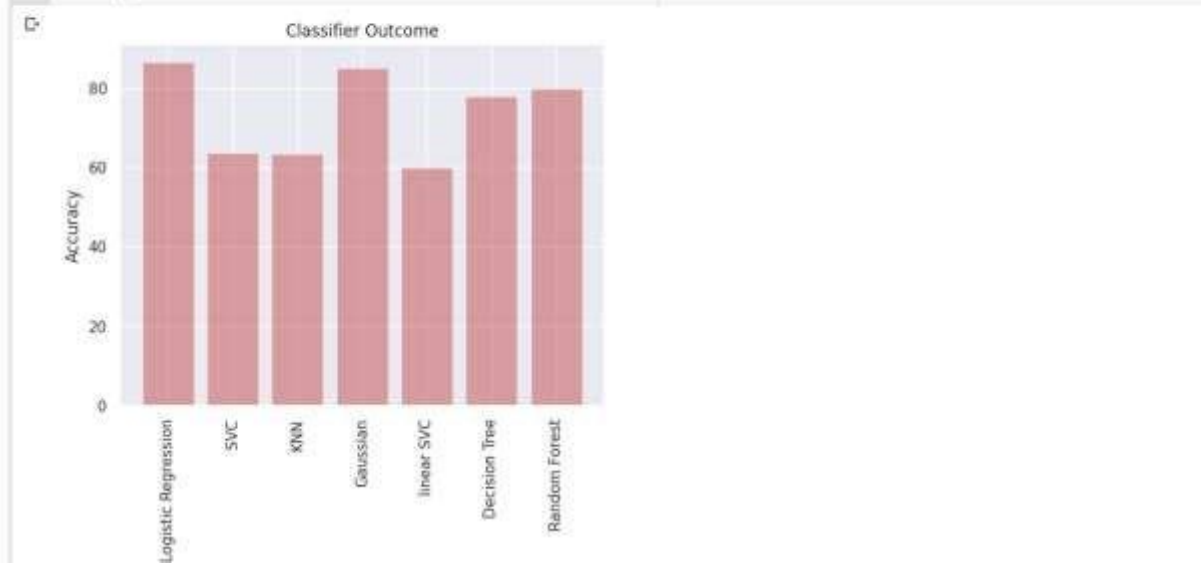
+ Code + Text

```
[116] print(classification_report(y_test, Y_pred2))
      cm = pd.DataFrame(confusion_matrix(y_test, Y_pred2), ['Actual: NOT', 'Actual: SURVIVED'], ['Predicted: NO', 'Predicted: SURVIVED'])
      print(cm)
```

```
              precision    recall  f1-score   support

           0       0.66      0.88      0.75       165
           1       0.53      0.22      0.31        97

    accuracy                           0.64       262
   macro avg       0.59      0.55      0.53       262
weighted avg       0.61      0.64      0.59       262

                  Predicted: NO  Predicted: SURVIVED
Actual: NOT                 146                   19
Actual: SURVIVED             76                   21
```

7. COMPARISION STUDY OF THE ALGORITHMS USED:

```
objects = ('Logistic Regression', 'SVC', 'KNN', 'Gaussian',  'linear SVC',  'Decision Tree', 'Random Forest')
x_pos = np.arange(len(objects))
accuracies1 = [acc_log, acc_svc, acc_knn, acc_gaussian, acc_linear_svc, acc_decision_tree, acc_random_forest]

plt.bar(x_pos, accuracies1, align='center', alpha=0.5, color='r')
plt.xticks(x_pos, objects, rotation='vertical')
plt.ylabel('Accuracy')
plt.title('Classifier Outcome')
plt.show()
```



Evaluating all the models we deployed we can clearly see that Logistic Regression gives the most accurate results that is 86.64 accuracy

FLOWCHART

The following algorithm outlines the steps involved in the machine learning project:

1. Collect the Titanic dataset
2. Clean and pre-process the data
3. Data Exploration
4. Perform feature engineering
5. Split the data into training and testing sets
6. Select an appropriate machine learning algorithm
7. Train the model on the training set
8. Evaluate the performance of the model using metrics such as accuracy, precision, and recall
9. Repeat steps 5-8 for different algorithms and compare their performance
10. Select the best-performing algorithm and use it to make predictions on new data

Overall, the methodology and algorithm for the Titanic survival prediction project involve a series of steps that involve data cleaning, feature engineering, model selection, hyperparameter tuning, and model evaluation. By following these steps, we can develop a machine learning model that is able to predict survival on the Titanic with a high degree of accuracy.

Summary

The Titanic is a tragic event that has captured the attention of people for over a century. The disaster resulted in the loss of many lives, and many people have wondered about the factors that contributed to the survival of some passengers and the death of others. The ML project on survival prediction on Titanic aims to shed some light on this question by developing a machine learning model that can accurately predict which passengers would have survived based on a set of input features.

The project will use a dataset that contains information about 891 passengers who were onboard the Titanic when it sank. This dataset includes a variety of information about each passenger, such as their age, sex, ticket class, and whether or not they survived the disaster. The dataset will be used to train a machine learning model that will predict whether a passenger would have survived or not based on their input features.

To develop the machine learning model, the project will involve several steps. The first step is data cleaning and pre-processing, where the dataset will be cleaned and prepared for analysis. The second step is feature engineering, where useful features will be extracted from the data that will help the model make better predictions. The third step is model selection, where an appropriate model architecture will be selected and trained on the dataset. The fourth step is hyperparameter tuning, where the model's hyperparameters will be tuned to optimize its performance.

The development of the machine learning model will be implemented using Python libraries and tools such as Pandas, NumPy, Scikit-learn, and Matplotlib. These tools will help to efficiently analyze and visualize the data, and to develop and evaluate the machine learning model.

The potential applications of the project are numerous. For example, the developed machine learning model could be used to help researchers better understand the factors that contributed to survival during the Titanic disaster. The model could also be used to inform emergency response planning for similar disasters in the future. Additionally, the project could serve as a valuable educational tool for students who are interested in machine learning, data science, and statistics.

In conclusion, the ML project on survival prediction on Titanic is a valuable and interesting endeavour that aims to develop a machine learning model that can accurately predict which passengers would have survived based on a set of input features. The project involves data cleaning and pre-processing, feature engineering, model selection, and hyperparameter tuning

using Python libraries and tools. The potential applications of the project are numerous, including better understanding of the factors that contributed to survival during the Titanic disaster and informing emergency response planning for similar disasters in the future.