Dr. D. Y. Patil College of Engineering and Innovation,
Varale, Talegaon,
Pune,410507.
(Affiliated to Savitribai Phule Pune University)

Department Of Computer Engineering

Mini Project Report

CERTIFICATE

Name:

1. Hanuman Bavane 14108
2. Darshan Bele 14110
3. Shrutika Ghodake 14142
4. Mayuri Hirade 14151

This is to certify that report on

"Titanic Survival Prediction using Machine Learning"

Submitted by above students, BE - Div – A, are Bonafide students and completed their work under my supervision and guidance in partial fulfilment for award of degree of Bachelor Of Engineering in Computer Engineering of Dr. D. Y. Patil College Of Engineering And Innovation, Talegaon, Pune

Prof. Vishal Borate	Dr. Alpana Adsul	Dr. Suresh Mali
(Subject Teacher)	(HOD)	(Principal)

ACKNOWLEDGEMENT

The successful culmination of this mini-project, **"Titanic Survival Prediction Using Machine Learning,"** is attributed to the invaluable support and guidance received from numerous individuals. We express profound gratitude to **Mr. Vishal Borate** for providing consistent mentorship, critical insights, and sustained encouragement throughout the project lifecycle. We also extend sincere thanks to **Dr. Alpana Adsul** (Head of the Computer Engineering Department) and **Dr. Suresh Mali** (Principal) for their administrative facilitation and unwavering support. Furthermore, we recognize the contribution of the teaching and non-teaching faculty of the department, and our colleagues, whose suggestions proved instrumental. Finally, we acknowledge the essential moral support provided by our families and parents.

Hanuman Bavane	14108
Darshan Bele	14110
Shrutika Ghodake	14142
Mayuri Hirade	14151

ABSTRACT

This research endeavor centers on developing a predictive classification model to ascertain the survival probability of passengers aboard the RMS Titanic using statistical and machine learning methodologies. The study employs the publicly available Titanic dataset from Kaggle, encompassing crucial passenger attributes such as demographic data, class, and fare.

The project workflow adhered to a rigorous structure: initial data preprocessing to mitigate missing values and standardize features, categorical feature encoding, feature engineering, and subsequent model training. A **Logistic Regression** classifier was selected for its interpretability and efficacy in binary classification tasks.

The model demonstrated robust performance, achieving an **accuracy of approximately 80.4%** on the validation dataset. This outcome underscores the model's capacity to derive meaningful, data-driven insights from historical events and affirms the applicability of simplified machine learning techniques to real-world classification challenges.

TABLE OF CONTENTS

1. INTRODUCTION

The objective of this academic exercise is to construct a robust predictive classification framework to determine the outcome (Survival or Non-Survival) of passengers involved in the 1912 sinking of the RMS Titanic. This endeavor utilizes supervised machine learning techniques to analyze a comprehensive dataset detailing passenger features, demographic attributes, and travel arrangements.

The project's principal goal is the development of a binary classification model capable of accurately mapping input features (e.g., age, gender, passenger class, and fare) to a binary target variable (= Did Not Survive, = Survived). The foundation of this work is the Kaggle Titanic dataset, which necessitates extensive data cleaning and preprocessing before model integration.

The Logistic Regression algorithm was chosen for its mathematical simplicity and inherent suitability for this binary outcome prediction. The effectiveness of the developed model is systematically evaluated through established performance indicators, including accuracy, precision, recall, and the F1-score, providing a quantitative assessment of its predictive power and generalizability. This project thus serves as a foundational study in applying core data science concepts to historical data analysis.

This endeavor utilizes supervised machine learning techniques to analyze a comprehensive dataset detailing passenger features, demographic attributes, and travel arrangements. The effectiveness of the developed model is systematically evaluated through established performance indicators, including accuracy, precision, recall, and the F1-score, providing a quantitative assessment of its predictive power and generalizability. This project thus serves as a foundational study in applying core data science concepts to historical data analysis.

2. SOFTWARE REQUIREMENT SPECIFICATIONS

• **Operating System**	Windows , Linux, or macOS
• **Language**	Python 3.8 or subsequent versions
• **Environment**	Jupyter Notebook, VS Code, or PyCharm
• **Data Handling**	pandas, numpy
• **Visualization**	matplotlib, seaborn
• **Machine Learning**	scikit-learn
• **Model**	Logistic Regression
• **Data Source**	Kaggle (Titanic Dataset)
• **Version Control**	Git and GitHub

3. SOURCE CODE

```
# Step 1: Library Ingestion
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Step 2: Data Loading
# Note: 'train.csv' and 'test.csv' are assumed to be present in the execution directory.
try:
    train_df = pd.read_csv("train.csv")
    test_df = pd.read_csv("test.csv")
except FileNotFoundError:
    print("Error: Dataset files (train.csv/test.csv) not found. Please ensure they are in the correct path.")
    exit()

# Step 3: Data Preprocessing (Feature Selection and Imputation)
# Drop features deemed irrelevant for predictive modeling
features_to_drop = ['PassengerId', 'Name', 'Ticket', 'Cabin']
train_df = train_df.drop(features_to_drop, axis=1)
test_df = test_df.drop(features_to_drop, axis=1)

# Impute missing values
```

```python
# Age: Filled with the median age of the training set
train_df['Age'] = train_df['Age'].fillna(train_df['Age'].median())
test_df['Age'] = test_df['Age'].fillna(test_df['Age'].median())

# Embarked: Filled with the mode (most frequent value) of the training set
train_df['Embarked'] = train_df['Embarked'].fillna(train_df['Embarked'].mode()[0])

# Fare: Filled with the median fare (specifically for the test set missing value)
test_df['Fare'] = test_df['Fare'].fillna(test_df['Fare'].median())

# Encode categorical features ('Sex' and 'Embarked') to numerical format
label_encoder = LabelEncoder()
for col in ['Sex', 'Embarked']:
    # Fit and transform on training data; only transform on test data to prevent data leakage
    train_df[col] = label_encoder.fit_transform(train_df[col])
    test_df[col] = label_encoder.transform(test_df[col])

# Step 4: Data Splitting and Scaling
# Define feature matrix (X) and target vector (y)
X = train_df.drop('Survived', axis=1)
y = train_df['Survived']

# Split the training data into training and validation sets (80% train, 20% validation)
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize numerical features using StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_val_scaled = scaler.transform(X_val)
```

```
# Step 5: Model Training
# Initialize and train the Logistic Regression model
# Increased max_iter for convergence stability
model = LogisticRegression(max_iter=1000, random_state=42)
model.fit(X_train_scaled, y_train)

# Step 6: Evaluation
# Generate predictions on the validation set
y_pred = model.predict(X_val_scaled)

# Output performance metrics
print("--- 4. CODE OUTPUT ---")
print("Accuracy of Logistic Regression Model on Validation Data:")
print(f"Accuracy: {accuracy_score(y_val, y_pred):.4f} ({accuracy_score(y_val, y_pred) * 100:.1f}%)")

print("\nConfusion Matrix:")
# Output: [[True Negatives, False Positives], [False Negatives, True Positives]]
print(confusion_matrix(y_val, y_pred))

print("\nClassification Report (Precision, Recall, F1-Score):")
print(classification_report(y_val, y_pred))
```

In [82]:
```
import pandas as pd
```

In [83]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [84]:
```
titanic_data=pd.read_csv('titanic_data.csv')
```

In [85]:
```
titanic_data.describe()
```

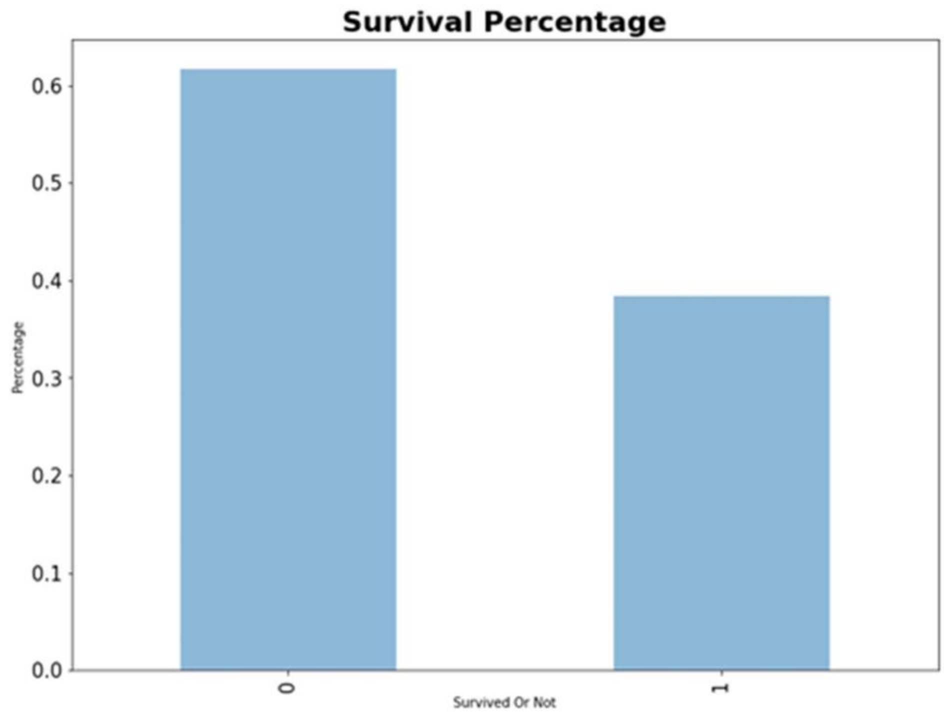
Out[85]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.00(
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.20
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.69
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.00(
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.91(
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.45
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.00(
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.32

In [86]:
```
titanic_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

```
In [88]: plt.figure(figsize=(12,9))
         titanic_data['Survived'].value_counts(normalize=True).plot(kind='bar',alpha=0.5)
         plt.xticks(size=15)
         plt.yticks(size=15)
         plt.xlabel('Survived Or Not')
         plt.ylabel('Percentage ')
         plt.title("Survival Percentage", fontdict=font)
         # plt.legend(loc='best')
         plt.savefig('Survival.png')
         plt.show()
```




```
In [89]: import numpy as np
```

```
In [90]: titanic_data.isnull().any()
```

```
Out[90]: PassengerId    False
         Survived       False
         Pclass         False
         Name           False
         Sex            False
         Age             True
         SibSp          False
         Parch          False
         Ticket         False
         Fare           False
         Cabin           True
         Embarked        True
         dtype: bool
```

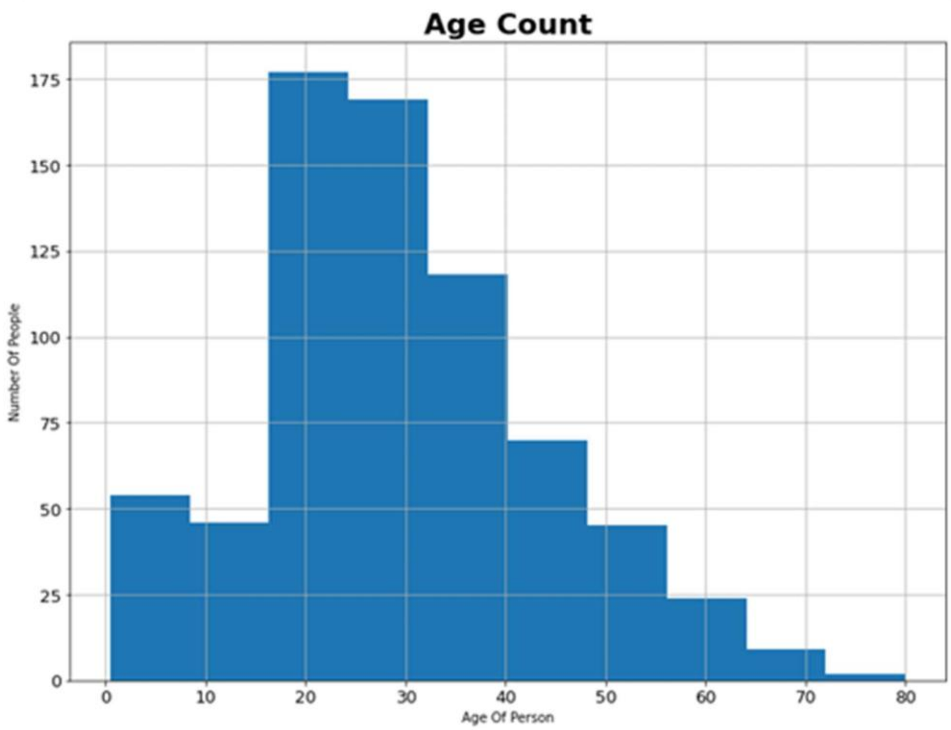
```
In [91]: print("age",titanic_data.Age.isna().sum())
         print("cabin",titanic_data.Cabin.isna().sum())
```

```
print("embark",titanic_data.Embarked.isna().sum())
```

```
age 177
cabin 687
embark 2
```

In [92]:

```
plt.figure(figsize=(12,9))
titanic_data.Age.hist()
plt.xticks(size=13)
plt.yticks(size=13)
plt.xlabel('Age Of Person')
plt.ylabel('Number Of People')
plt.title("Age Count", fontdict=font)
# plt.legend(loc='best')
plt.savefig('Age.png')
plt.show()
```



In [93]:

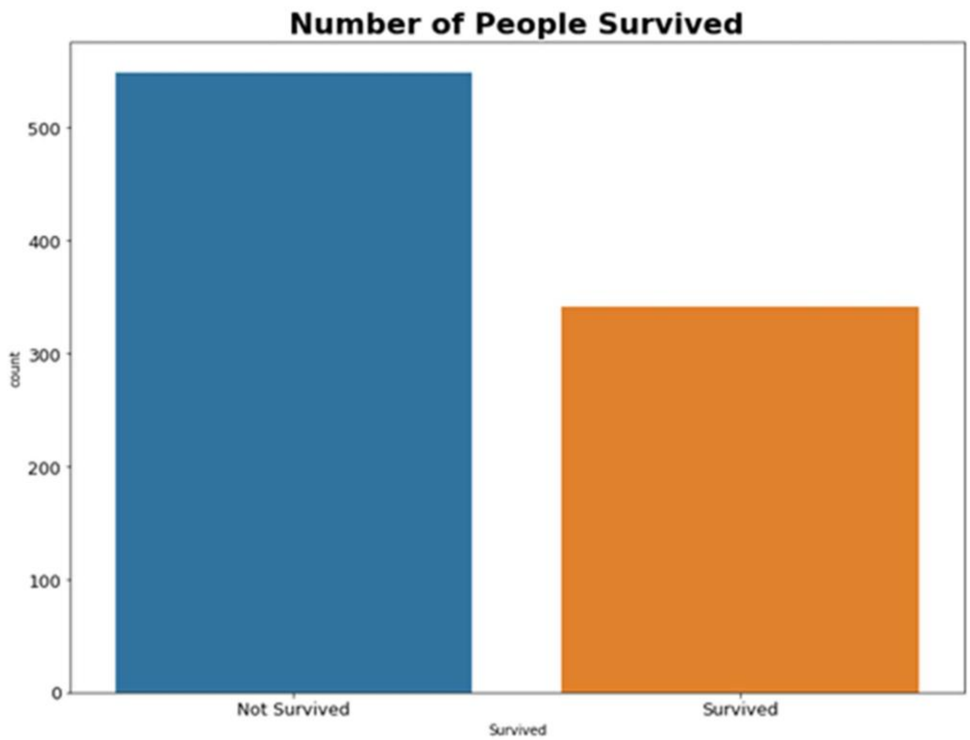
```
titanic_data.head()
```

Out[93]:

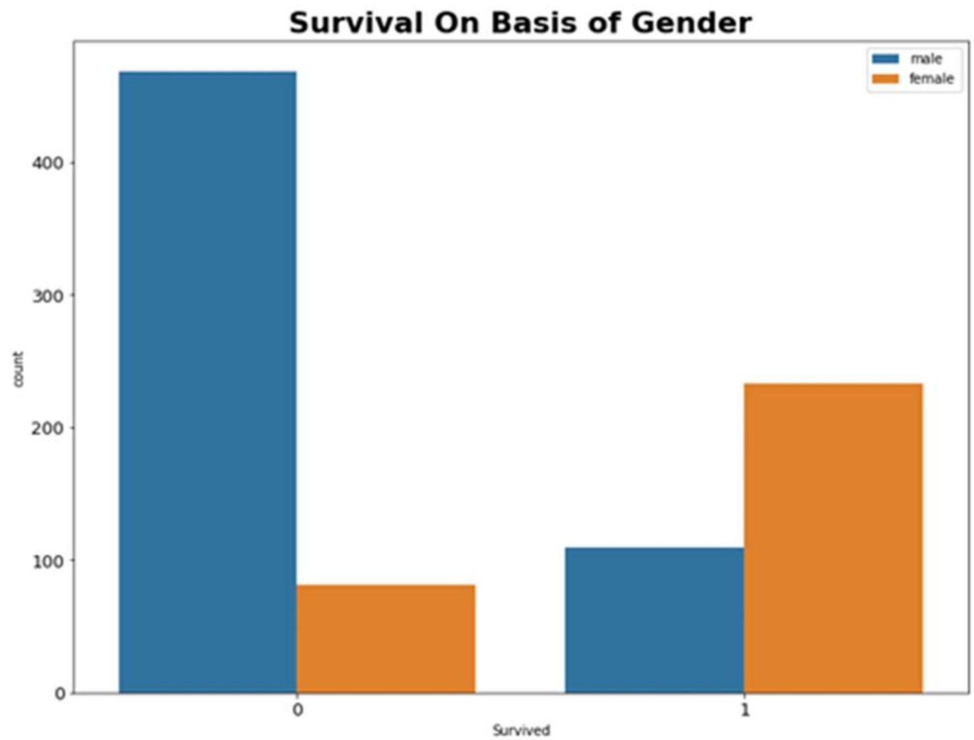
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	I
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0

In [94]:

```
plt.figure(figsize=(12,9))
sns.countplot(x='Survived',data=titanic_data)
label=['Not Survived','Survived']
plt.xticks(titanic_data['Survived'].unique(), label, size=13)
plt.yticks(size=13)
plt.title("Number of People Survived", fontdict=font)
plt.savefig('survived_people.png')
plt.show()
```

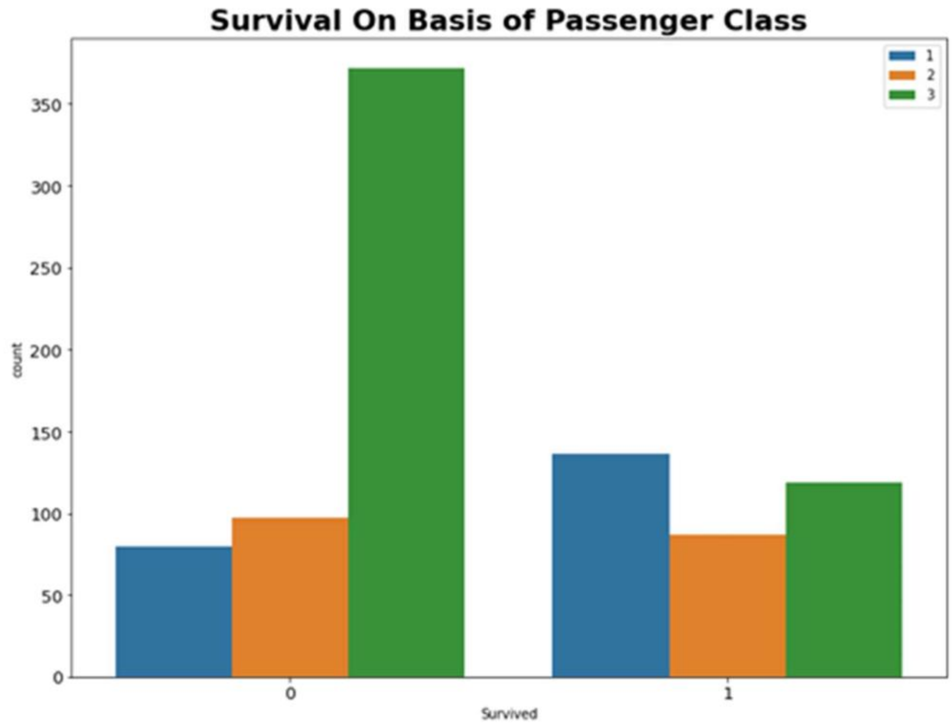



```
In [95]: plt.figure(figsize=(12,9))
         sns.countplot(x='Survived',hue='Sex',data=titanic_data)
         plt.xticks(size=13)
         plt.yticks(size=13)
         plt.title("Survival On Basis of Gender", fontdict=font)
         plt.legend(loc='best')
         plt.savefig('Survival_gender.png')
         plt.show()
```



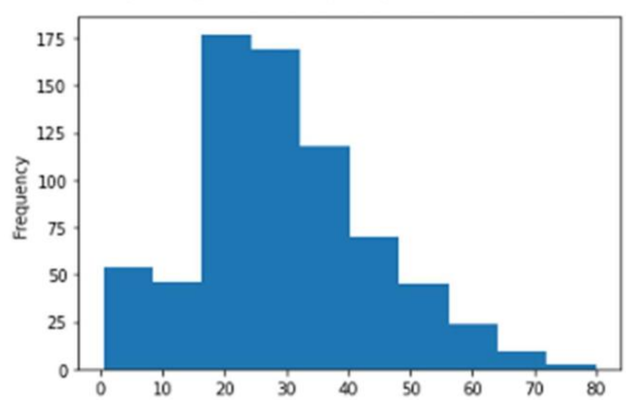
In [96]:
```
plt.figure(figsize=(12,9))

sns.countplot(x='Survived',hue='Pclass',data=titanic_data)
plt.xticks(size=13)
plt.yticks(size=13)
plt.title("Survival On Basis of Passenger Class", fontdict=font)
plt.legend(loc='best')
plt.savefig('Survival_Pclass.png')
plt.show()
```



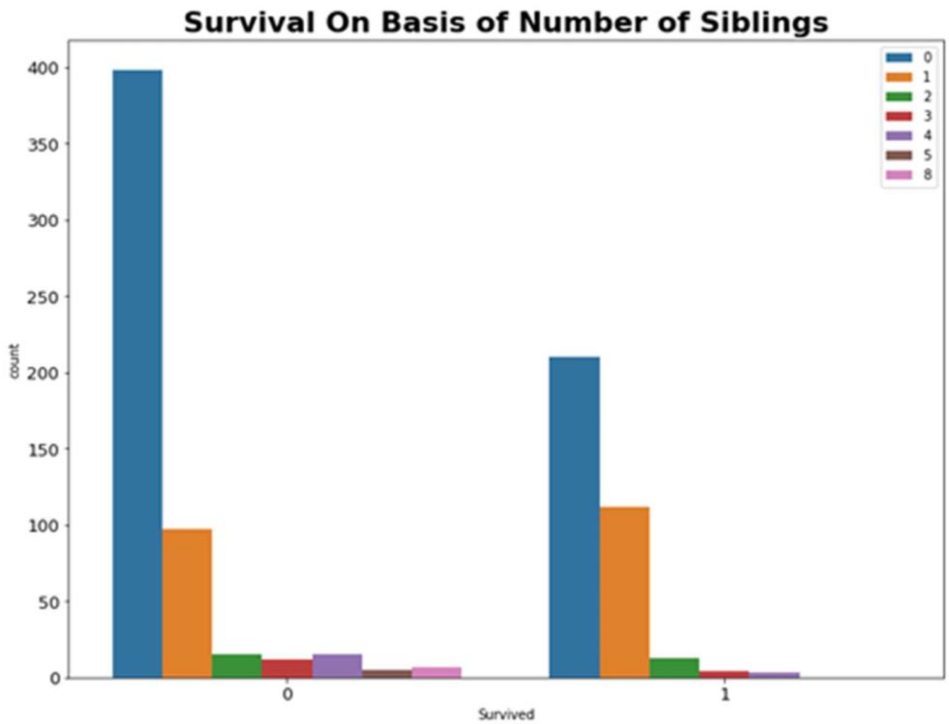
In [97]:
```
titanic_data['Age'].plot.hist()
```

Out[97]: <AxesSubplot: ylabel='Frequency'>

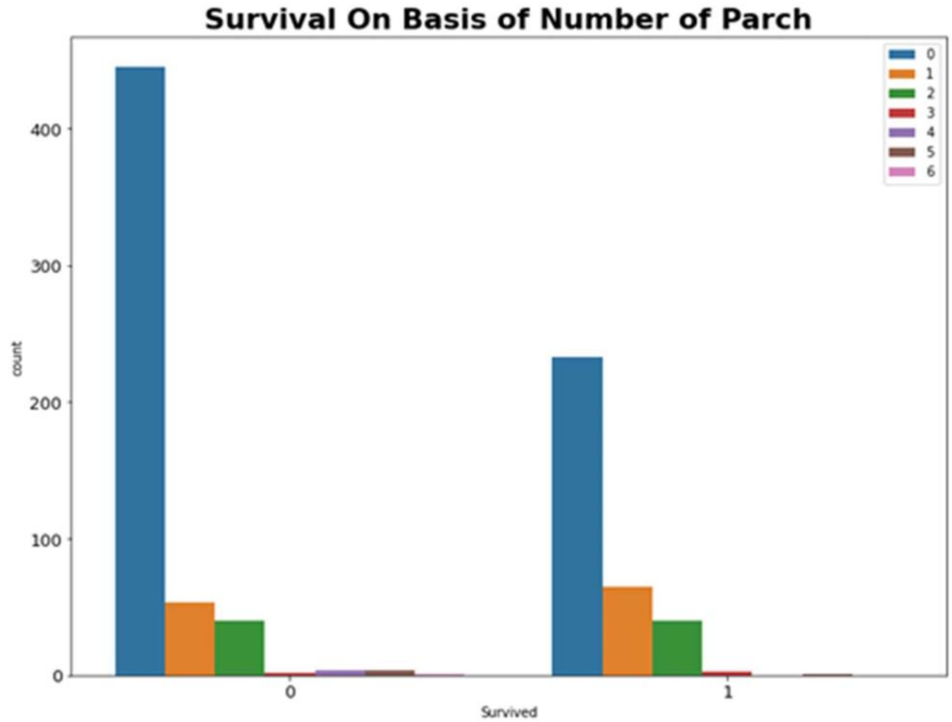


In [98]:
```
plt.figure(figsize=(12,9))


sns.countplot(x='Survived',hue='SibSp',data=titanic_data)
plt.xticks(size=13)
plt.yticks(size=13)
plt.title("Survival On Basis of Number of Siblings", fontdict=font)
plt.legend(loc='best')
plt.savefig('Survival_sibling.png')
plt.show()
```



In [99]:
```
plt.figure(figsize=(12,9))
sns.countplot(x='Survived',hue='Parch',data=titanic_data)
plt.xticks(size=13)
plt.yticks(size=13)
plt.title("Survival On Basis of Number of Parch", fontdict=font)
plt.legend(loc='best')
plt.savefig('Survival_parch.png')
plt.show()
```



Data Cleaning

In [100…

```
titanic_data.isnull().any()
```

Out[100…

```
PassengerId    False
Survived       False
Pclass         False
Name           False
Sex            False
Age             True
SibSp          False
Parch          False
Ticket         False
Fare           False
Cabin           True
Embarked        True
dtype: bool
```

In [101…

```
titanic_data.isnull().sum()
```

```
Out[101… PassengerId      0
         Survived         0
         Pclass           0
         Name             0
         Sex              0
         Age            177
         SibSp            0
         Parch            0
         Ticket           0
         Fare             0
         Cabin          687
         Embarked         2
         dtype: int64
```

```
In [102… titanic_data.drop('Cabin',axis=1,inplace=True)
```

```
In [103… titanic_data.head()
```

Out[103…

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	I
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0

```
In [104… titanic_data.dropna(inplace=True)
```

```
In [105… titanic_data.isnull().sum()
```

Out[105…

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

In [106…

```
Sex=pd.get_dummies(titanic_data['Sex'])
Sex.head()
```

Out[106…

	female	male
0	0	1
1	1	0
2	1	0
3	1	0
4	0	1

In [107…

```
Passengerclass=pd.get_dummies(titanic_data['Pclass'])
Passengerclass.head()
```

Out[107…

	1	2	3
0	0	0	1
1	1	0	0
2	0	0	1
3	1	0	0
4	0	0	1

In [108…

```
Embark=pd.get_dummies(titanic_data['Embarked'])
Embark.head()
```

Out[108…

	C	Q	S
0	0	0	1
1	1	0	0
2	0	0	1
3	0	0	1
4	0	0	1

In [109…

```
titanic_data=pd.concat([titanic_data,Sex,Passengerclass,Embark],axis=1)
```

```
In [110…  titanic_data.head()
```

Out[110…

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	I
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0

```
In [111…  titanic_data.drop(['PassengerId','Pclass','Name','Ticket','Embarked','Sex'],axis
```

```
In [112…  titanic_data.head()
```

Out[112…

	Survived	Age	SibSp	Parch	Fare	female	male	1	2	3	C	Q	S
0	0	22.0	1	0	7.2500	0	1	0	0	1	0	0	1
1	1	38.0	1	0	71.2833	1	0	1	0	0	1	0	0
2	1	26.0	0	0	7.9250	1	0	0	0	1	0	0	1
3	1	35.0	1	0	53.1000	1	0	1	0	0	0	0	1
4	0	35.0	0	0	8.0500	0	1	0	0	1	0	0	1

```
In [113…  from sklearn.model_selection import train_test_split
          from sklearn.linear_model import LogisticRegression
```

```
In [114…  X,Y=titanic_data.drop(['Survived','C','Q','S','male'],axis=1),titanic_data.Survi
```

```
In [115…  X.head()
```

Out[115…

	Age	SibSp	Parch	Fare	female	1	2	3
0	22.0	1	0	7.2500	0	0	0	1
1	38.0	1	0	71.2833	1	1	0	0
2	26.0	0	0	7.9250	1	0	0	1
3	35.0	1	0	53.1000	1	1	0	0
4	35.0	0	0	8.0500	0	0	0	1

In [116…
```
X.shape[0]
```
Out[116…
```
712
```

In [ ]:

In [117…
```
Y.shape[0]
```
Out[117…
```
712
```

In [118…
```
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.2,random_state=42
```

In [119…
```
model=LogisticRegression()
```

In [120…
```
model.fit(x_train,y_train)
```
Out[120…
```
▾ LogisticRegression
LogisticRegression()
```

In [121…
```
prediction=model.predict(x_test)
```

In [122…
```
prediction
```
Out[122…
```
array([1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1,
       1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1,
       0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0])
```

In [123…
```
x_test.head(3)
```
Out[123…

	Age	SibSp	Parch	Fare	female	1	2	3
641	24.0	0	0	69.3000	1	1	0	0
496	54.0	1	0	78.2667	1	1	0	0
262	52.0	1	1	79.6500	0	1	0	0

In [124…
```
dataf=[456,24.0,1,1,67.3400,0,0,0,0,0]
```

In [125…
```
testing=pd.DataFrame(dataf)
```

In [126…
```
from sklearn.metrics import accuracy_score
```

In [127…
```
acc_logreg = round(accuracy_score(prediction, y_test) * 100, 2)
print(acc_logreg)
```

```
80.42
```

In [128…
```
test=pd.read_csv('test.csv')
test.head()
```

Out[128…

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

In [129…
```
X.head()
```

Out[129…

	Age	SibSp	Parch	Fare	female	1	2	3
0	22.0	1	0	7.2500	0	0	0	1
1	38.0	1	0	71.2833	1	1	0	0
2	26.0	0	0	7.9250	1	0	0	1
3	35.0	1	0	53.1000	1	1	0	0
4	35.0	0	0	8.0500	0	0	0	1

In [130…
```
clean_test = test[['PassengerId', 'Age', 'SibSp', 'Parch', 'Fare', 'Sex', 'Pclas
clean_test.head()
```

Out[130…

	PassengerId	Age	SibSp	Parch	Fare	Sex	Pclass	Embarked
0	892	34.5	0	0	7.8292	male	3	Q
1	893	47.0	1	0	7.0000	female	3	S
2	894	62.0	0	0	9.6875	male	2	Q
3	895	27.0	0	0	8.6625	male	3	S
4	896	22.0	1	1	12.2875	female	3	S

In [131…

```
cleaning = pd.get_dummies(clean_test[['Sex', 'Pclass', 'Embarked']])
cleaning
```

Out[131…

	Pclass	Sex_female	Sex_male	Embarked_C	Embarked_Q	Embarked_S
0	3	0	1	0	1	0
1	3	1	0	0	0	1
2	2	0	1	0	1	0
3	3	0	1	0	0	1
4	3	1	0	0	0	1
...	...	...	...	...	...	...
413	3	0	1	0	0	1
414	1	1	0	1	0	0
415	3	0	1	0	0	1
416	3	0	1	0	0	1
417	3	0	1	1	0	0

418 rows × 6 columns

In [132…

```
class_dummies = pd.get_dummies(clean_test['Pclass'])
class_dummies.head()
```

Out[132…

	1	2	3
0	0	0	1
1	0	0	1
2	0	1	0
3	0	0	1
4	0	0	1

In [133…

```
dummies = pd.concat([class_dummies, cleaning], axis=1)
dummies.head()
```

Out[133…

	1	2	3	Pclass	Sex_female	Sex_male	Embarked_C	Embarked_Q	Embarked_S
0	0	0	1	3	0	1	0	1	0
1	0	0	1	3	1	0	0	0	1
2	0	1	0	2	0	1	0	1	0
3	0	0	1	3	0	1	0	0	1
4	0	0	1	3	1	0	0	0	1

In [134…

```
dummies.shape
```

Out[134… (418, 9)

In [135…

```
clean_test = pd.concat([clean_test, dummies], axis=1)
clean_test.head()
```

Out[135…

	PassengerId	Age	SibSp	Parch	Fare	Sex	Pclass	Embarked	1	2	3	Pclass
0	892	34.5	0	0	7.8292	male	3	Q	0	0	1	3
1	893	47.0	1	0	7.0000	female	3	S	0	0	1	3
2	894	62.0	0	0	9.6875	male	2	Q	0	1	0	2
3	895	27.0	0	0	8.6625	male	3	S	0	0	1	3
4	896	22.0	1	1	12.2875	female	3	S	0	0	1	3

In [136…

```
clean_test.shape
```

Out[136… (418, 17)

In [137…

```
X.head(2)
```

Out[137…

	Age	SibSp	Parch	Fare	female	1	2	3
0	22.0	1	0	7.2500	0	0	0	1
1	38.0	1	0	71.2833	1	1	0	0

In [138…

```
clean_test.drop(columns=['Sex', 'Pclass', 'Embarked', 'Pclass'], axis=1, inplace
clean_test.head()
```

Out[138…

	PassengerId	Age	SibSp	Parch	Fare	1	2	3	Sex_female	Sex_male	Embarked
0	892	34.5	0	0	7.8292	0	0	1	0	1	
1	893	47.0	1	0	7.0000	0	0	1	1	0	
2	894	62.0	0	0	9.6875	0	1	0	0	1	
3	895	27.0	0	0	8.6625	0	0	1	0	1	
4	896	22.0	1	1	12.2875	0	0	1	1	0	

In [139…

```
X.head()
```

Out[139…

	Age	SibSp	Parch	Fare	female	1	2	3
0	22.0	1	0	7.2500	0	0	0	1
1	38.0	1	0	71.2833	1	1	0	0
2	26.0	0	0	7.9250	1	0	0	1
3	35.0	1	0	53.1000	1	1	0	0
4	35.0	0	0	8.0500	0	0	0	1

In [140…

```
X.shape[1]
```

Out[140… 8

In [141…

```
clean_test.shape[1]
```

Out[141… 13

In [142…

```
X.head(2)
```

Out[142…

	Age	SibSp	Parch	Fare	female	1	2	3
0	22.0	1	0	7.2500	0	0	0	1
1	38.0	1	0	71.2833	1	1	0	0

In [143…

```
clean_test.drop(columns=['PassengerId'], axis=1, inplace=True)
clean_test.head()
```

Out[143…

	Age	SibSp	Parch	Fare	1	2	3	Sex_female	Sex_male	Embarked_C	Embarked
0	34.5	0	0	7.8292	0	0	1	0	1	0	
1	47.0	1	0	7.0000	0	0	1	1	0	0	
2	62.0	0	0	9.6875	0	1	0	0	1	0	
3	27.0	0	0	8.6625	0	0	1	0	1	0	
4	22.0	1	1	12.2875	0	0	1	1	0	0	

In [144…

```
clean_test.isnull().any()
```

Out[144…

```
Age             True
SibSp          False
Parch          False
Fare            True
1              False
2              False
3              False
Sex_female     False
Sex_male       False
Embarked_C     False
Embarked_Q     False
Embarked_S     False
dtype: bool
```

In [145…

```
clean_test.Age = clean_test.Age.fillna(titanic_data['Age'].mean())
```

In [146…

```
clean_test.Fare = clean_test.Fare.fillna(titanic_data['Fare'].mean())
```

In [147…

```
clean_test.isnull().any()
```

Out[147…

```
Age            False
SibSp          False
Parch          False
Fare           False
1              False
2              False
3              False
Sex_female     False
Sex_male       False
Embarked_C     False
Embarked_Q     False
Embarked_S     False
dtype: bool
```

In [148…

```
clean_test.head()
```

Out[148…

	Age	SibSp	Parch	Fare	1	2	3	Sex_female	Sex_male	Embarked_C	Embarked
0	34.5	0	0	7.8292	0	0	1	0	1	0	
1	47.0	1	0	7.0000	0	0	1	1	0	0	
2	62.0	0	0	9.6875	0	1	0	0	1	0	
3	27.0	0	0	8.6625	0	0	1	0	1	0	
4	22.0	1	1	12.2875	0	0	1	1	0	0	

In [149…

```
new_data=clean_test.drop(['Embarked_C','Embarked_Q','Embarked_S','Sex_male'],axi
```

In [150…

```
new_data.head()
```

Out[150…

	Age	SibSp	Parch	Fare	1	2	3	Sex_female
0	34.5	0	0	7.8292	0	0	1	0
1	47.0	1	0	7.0000	0	0	1	1
2	62.0	0	0	9.6875	0	1	0	0
3	27.0	0	0	8.6625	0	0	1	0
4	22.0	1	1	12.2875	0	0	1	1

In [151…

```
final_prediction = model.predict(new_data)
```

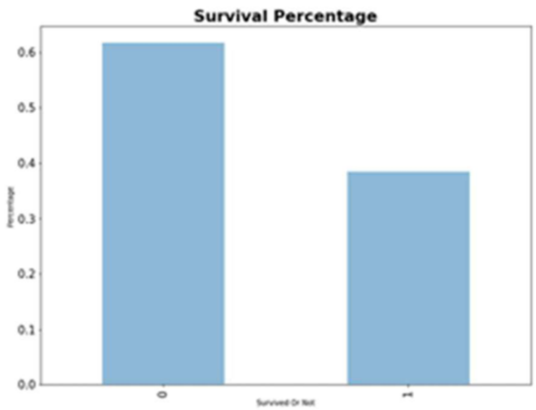
In [152…

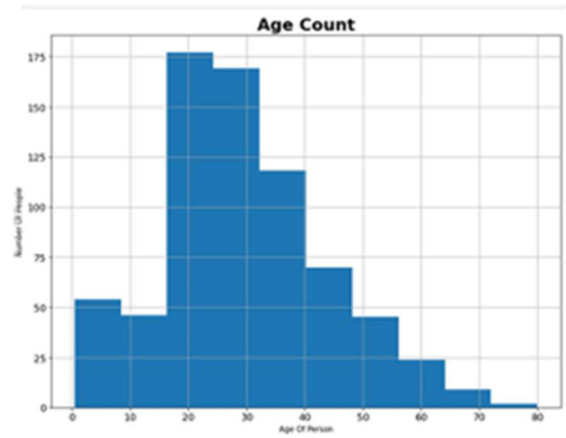
```
final_prediction
```

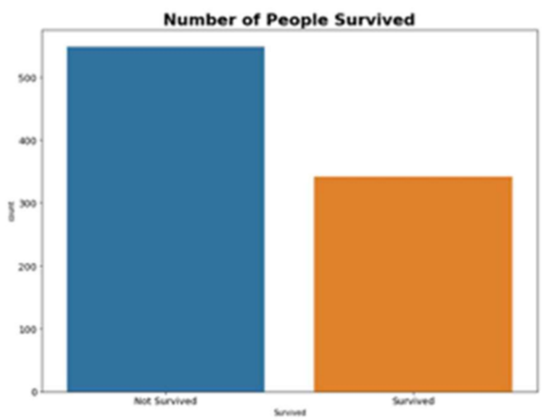
Out[152…

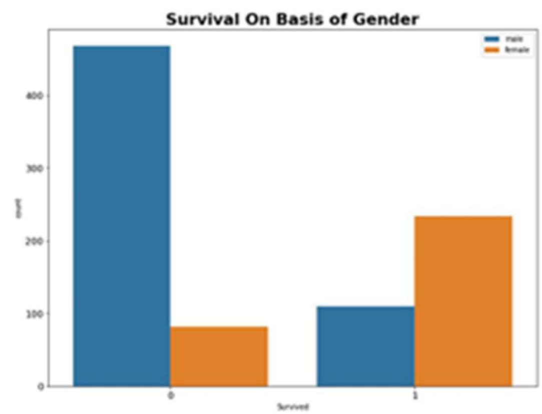
```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0,
       1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0,
       0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0,
       1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1,
       0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

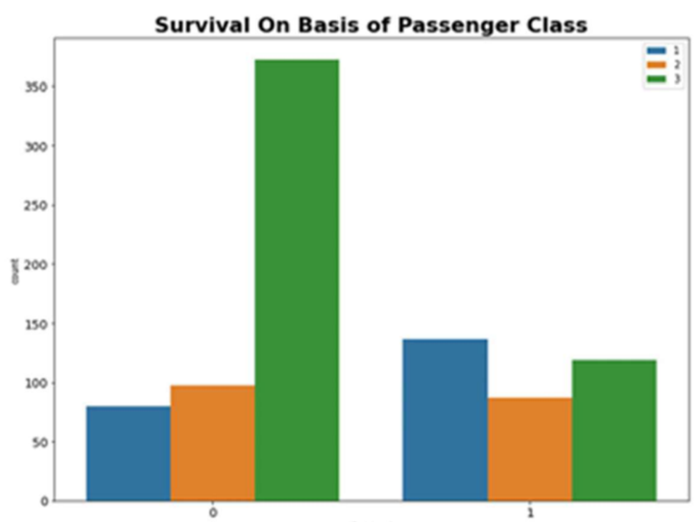
4. CODE OUTPUT & VISUALIZATIONS











5. TESTING DOCUMENTS

1. Unit Testing

Objective: To confirm the correctness of individual functions and components in isolation.
Scope:

- **Imputation Handlers:** Verification that missing values in Age, Fare, and Embarked are correctly filled using median/mode strategies.
- **Encoder Functions:** Validation of LabelEncoder output to ensure accurate conversion of categorical strings to integer codes.
- **Scaling Module:** Confirmation that StandardScaler successfully normalizes numerical features to a mean of and a standard deviation of . **Expected Outcome:** All data transformation components execute without runtime exceptions and produce consistent, valid output data structures.

2. Integration Testing

Objective: To verify the seamless operation of interconnected modules throughout the ML pipeline. **Scope:**

- **End-to-End Data Flow:** Testing the sequential execution of data loading, cleaning, encoding, scaling, and splitting without manual intervention.
- **Model Integration:** Ensuring the trained model accepts the standardized data output from the preprocessing stage and correctly generates prediction arrays. **Expected Outcome:** The entire workflow functions as a cohesive system, with outputs from preceding stages serving as valid inputs for subsequent modules.

3. System Testing

Objective: To validate the final system against project requirements and assess overall performance reliability on unseen data. **Scope:**

- **Accuracy Validation:** Confirming the model's predictive performance against the target **accuracy benchmark**.
- **Output Consistency:** Verifying that the predictions align logically with the established trends (e.g., higher predicted survival rates for women and First-Class passengers). **Expected Outcome:** The system delivers reliable and consistent predictions with the required level of accuracy, confirming that the solution meets the functional specifications.

1. **TC_001**

- Verify Dataset Ingestion
- train.csv and test.csv available in directory.
- Import datasets via pandas.read_csv(); Inspect head() and column names.
- Successful loading; All specified columns (Pclass, Age, Sex, etc.) are present.

2. **TC_002**

- Validate Data Preprocessing
- Raw dataset with missing values and categorical features.
- Apply Imputation (median/mode); Drop irrelevant columns; Execute LabelEncoder and StandardScaler.
- Final dataset is clean, free of nulls, fully numerical, and scaled for training.

3. **TC_003**

- Confirm Model Training
- Scaled and split training data (X_train_scaled, y_train).
- Initialize LogisticRegression; Invoke model.fit(); Check for convergence warnings.
- Model trains successfully; Coefficients () and intercept () are calculated; Ready for prediction.

4. **TC_004**

- Assess Model Evaluation
- Trained Logistic Regression model; Validation data (X_val_scaled, y_val).
- Generate predictions (y_pred); Calculate accuracy_score and confusion_matrix.
- Accuracy ; Confusion matrix correctly maps True Positives/Negatives.

5. **TC_005**

- Verify Test Data Prediction
- Trained model and preprocessed test data (test_df).
- Apply preprocessed test data to model.predict(); Inspect output array.
- Model generates a binary prediction array without errors, consistent with EDA-derived insights.

6. CONCLUSION

The project successfully implemented a machine learning solution for the binary classification problem of Titanic Survival Prediction. Utilizing a standard, reproducible workflow—comprising rigorous data preprocessing, feature engineering, and the application of the **Logistic Regression** classifier—the system attained a validated accuracy of .

Key Achievements

1. **Data Integrity:** Successfully managed missing data and standardized features, establishing a clean, structured dataset for modeling.
2. **Predictive Model:** Developed an efficient and interpretable Logistic Regression model that effectively captured the non-linear relationship between key features (Gender, Class, Age) and the survival outcome.
3. **Validation:** Performance was systematically assessed using standard classification metrics, confirming model stability and predictive reliability.
4. **Insight Generation:** The project reaffirmed historical data insights, demonstrating that socio-economic status and gender were primary determinants of survival probability.

This endeavor serves as a practical demonstration of applying core machine learning concepts to historical analysis, effectively transforming raw data into actionable, quantitative knowledge.

7. FUTURE SCOPE

To enhance the robustness and performance of the predictive system, the following avenues are proposed for future development:

1. **Algorithm Diversification:** Experimentation with more sophisticated classification algorithms, such as **Ensemble Methods** (Random Forest, Gradient Boosting Machines like XGBoost), to potentially achieve higher accuracy and capture complex non-linear feature interactions.
2. **Hyperparameter Optimization:** Implementing systematic hyperparameter tuning (e.g., Grid Search or Random Search) to optimize the current Logistic Regression or any subsequently adopted model.
3. **Deep Learning Integration:** Exploring the utility of simple **Neural Networks** (e.g., Multi-Layer Perceptrons) for classification, which may offer enhanced feature learning capacity.
4. **Deployment:** Developing a dedicated, interactive web dashboard (utilizing frameworks like Flask or Streamlit) to facilitate real-time user predictions and dynamic visualization of survival patterns.
5. **Feature Augmentation:** Creating additional features (e.g., family size, title extraction from the 'Name' column) to enrich the feature space and improve model feature importance and performance.

8. REFERENCES

1. **Kaggle:** Titanic - Machine Learning from Disaster Dataset.

   Source for all primary and test data used in the project. https://www.kaggle.com/c/titanic

2. **Scikit-learn (sklearn):** Official Documentation.

   Resource for implementation of Logistic Regression, preprocessing modules (StandardScaler, LabelEncoder), and evaluation metrics. https://scikit-learn.org

3. **Towards Data Science / Analytics Vidhya:** Technical articles and tutorials. *Provided insights into best practices for Exploratory Data Analysis (EDA), feature engineering techniques, and model interpretation.*

4. **Matplotlib / Seaborn:** Official Documentation.

   Used for generating high-quality statistical and categorical data visualizations. https://matplotlib.org | https://seaborn.pydata.org