**DATA SCIENCE PROJECT**

**Titanic Survival Rate Prediction**

**Using (Classification)**

**Submitted by**

**G.JANANI**

TABLE OF CONTENT

CONTENTS PAGES

1.Introduction ...............................................................................3

2.Existing System...........................................................................3

3.Proposed System........................................................................3

4.Software Requirements..............................................................4

5.Hardware Requirements ............................................................4

6.Architectural Diagram.................................................................3

7.Data Flow Diagram......................................................................5

8.Table Design................................................................................6

9.Data Dictionary...........................................................................7

10.Program Design........................................................................9

11.Testing......................................................................................10

12.Conclusion................................................................................12

13.References...............................................................................13

14.Source Code.............................................................................14

15.Screen shot..............................................................................23

**1.Introduction:**

The Titanic survival prediction project aims to analyze the passenger data from the infamous Titanic shipwreck and build machine learning models to predict the likelihood of survival for passengers based on various attributes.

This project is motivated by the tragic event of the Titanic sinking in 1912 and seeks to explore how machine learning algorithms can be leveraged to understand patterns of survival and factors influencing them. this dataset Iam getting from Kaggle the name is (Titanic survival rate prediction)

In this project machine learning Classification model are used to predict the survival rate.

**2.Existing System:**

Currently, there is no existing automated system specifically designed for predicting survival on the Titanic. Historically, survival determinations were made based on manual analysis and anecdotal evidence, often focusing on factors such as passenger class, age, and gender. However, these methods lack the accuracy and efficiency provided by modern machine learning techniques

**3.Proposed System:**

The proposed Titanic survival prediction system will utilize machine learning algorithms to analyze the Titanic passenger dataset and predict survival outcomes based on passenger attributes such as age, gender, class, and family relations. By leveraging the power of data science and predictive modeling, the system aims to provide more accurate and reliable predictions of survival probabilities for individual passengers.

**4.Software Requirements:**

The development of the Titanic survival prediction system will require the following software tools and libraries:

* Programming Language: Python
* Data Analysis and Visualization: Pandas, NumPy, Matplotlib, Seaborn
* Machine Learning: Scikit-learn
* Integrated Development Environment (IDE): Jupyter Notebook

**5.Hardware Requirements:**

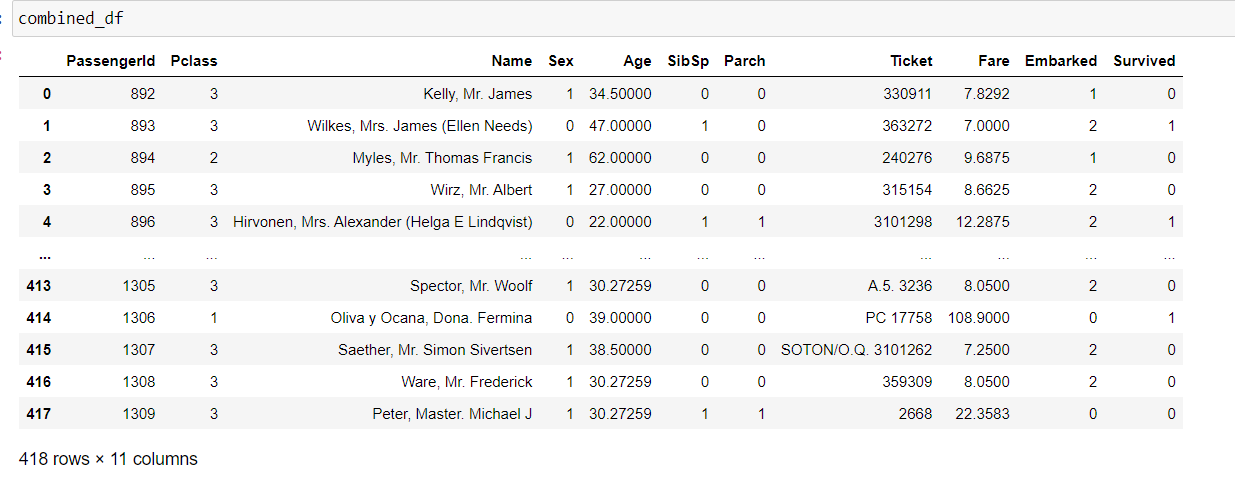
hardware requirements for running the Titanic survival prediction system are minimal and include a standard computer or laptop with sufficient processing power and memory to execute machine learning algorithms efficient.

**6.Architectural Diagram:**

**7.Data Flow Diagram:**

This chart will shows the works flow of the given model predicted the accuracy

**8.Table Design:**



Below table have the detailed view of dataset columns and their data types

RangeIndex: 418 entries, 0 to 417  
Data columns (total 12 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 PassengerId 418 non-null int64   
 1 Pclass 418 non-null int64   
 2 Name 418 non-null object   
 3 Sex 418 non-null object   
 4 Age 332 non-null float64  
 5 SibSp 418 non-null int64   
 6 Parch 418 non-null int64   
 7 Ticket 418 non-null object   
 8 Fare 417 non-null float64  
 9 Cabin 91 non-null object   
 10 Embarked 418 non-null object   
 11 Survived 418 non-null int64   
dtypes: float64(2), int64(5), object(5)

Here 3 datasets we used:

* Train.csv
* Test.csv
* Gender\_submission.csv

Those datasets are combined in a single dataset as

* Combined.df

**9.Data Dictionary:**

The data dictionary provides a detailed description of the attributes (features) present in the dataset used for the Titanic survival prediction project. This section outlines the meaning and data type of each attribute.

**Attributes**

PassengerId:

* Description: Unique identifier for each passenger.
* Data Type: Integer

Survived:

* Description: Survival status of the passenger.
* Data Type: Categorical (0 = Did not survive, 1 = Survived)

Pclass:

* Description: Ticket class indicating the socio-economic status of the passenger.
* Data Type: Categorical (1 = 1st class, 2 = 2nd class, 3 = 3rd class)

Name:

* Description: Name of the passenger.
* Data Type: String

Sex:

* Description: Gender of the passenger.
* Data Type: Categorical (Male, Female)

Age:

* Description: Age of the passenger.
* Data Type: Numeric (Float)

SibSp:

* Description: Number of siblings/spouses aboard the Titanic.
* Data Type: Numeric (Integer)

Parch:

* Description: Number of parents/children aboard the Titanic.
* Data Type: Numeric (Integer)

Ticket:

* Description: Ticket number.
* Data Type: String

Fare:

* Description: Passenger fare.
* Data Type: Numeric (Float)

Cabin:

* Description: Cabin number.
* Data Type: String

Embarked:

* Description: Port of embarkation.
* Data Type: Categorical (C = Cherbourg, Q = Queenstown, S = Southampton)

**Data Preprocessing**

**Encoding Categorical Variables**

In the Titanic survival prediction project, categorical variables such as 'Sex' and 'Embarked' were encoded into numerical values using the Label Encoder technique. This process converts categorical labels into numerical labels, which can be easily processed by machine learning algorithms.

Sex:

* Description: Gender of the passenger.
* Original Categories: Male, Female
* Encoded Values: Male (1), Female (0)

Embarked:

* Description: Port of embarkation.
* Original Categories: C = Cherbourg, Q = Queenstown, S = Southampton
* Encoded Values: Cherbourg (0), Queenstown (1), Southampton (2)

**10.Program Design:**

**Model Selection:**

Exploring Different Algorithms:

* Various machine learning algorithms such as Logistic Regression, Random Forest, Support Vector Machines, etc., are explored to identify the most suitable one for the task.

Cross-Validation:

* K-fold cross-validation is employed to estimate the performance of each model and select the best-performing one.

**Hyperparameter Tuning:**

Grid Search or Random Search:

* Hyperparameters of the selected model are fine-tuned using techniques like grid search or random search.

Optimization Metrics:

* Optimization metrics such as accuracy, precision, recall, or F1-score are used to evaluate the performance of different hyperparameter configurations.

**11.Testing:**

**Testing Approach**

The testing approach for evaluating the performance of the Titanic survival prediction models involves several key components, including train-test splits, cross-validation, and evaluation metrics.

1.Train-Test Splits:

* Data Partitioning:

The dataset is partitioned into a training set and a testing set using a random or stratified split.

* Training Set:

The training set, typically comprising 70-80% of the data, is used to train the predictive models.

* Testing Set:

The remaining portion of the data, usually 20-30%, serves as an unseen dataset for evaluating the models' performance.

2. Cross-Validation:

* K-fold Cross-Validation:

To ensure robustness and minimize overfitting, k-fold cross-validation is employed.

The training set is divided into k subsets, and each subset is used as a validation set while the model is trained on the remaining k-1 subsets.

This process is repeated k times, with each subset serving as the validation set exactly once.

* Evaluation Metrics:

Performance metrics such as accuracy, precision, recall, and F1-score are computed for each fold to assess the model's generalization ability.

3. Evaluation Metrics:

* Accuracy:

The percentage of correct predictions made by the model out of all predictions.

* Precision:

The proportion of true positive predictions among all positive predictions made by the model.

* Recall:

The proportion of true positive predictions among all actual positive instances.

* F1-score:

The harmonic mean of precision and recall, providing a balanced measure of the model's performance.

4. Model Comparison:

* Comparing Performance:

The performance of different models is compared based on their evaluation metrics computed during cross-validation.

Identifying the Best Model:

The model with the highest average performance across all folds is selected as the best-performing model for further analysis.

**Testing Conclusion:**

Through rigorous testing using train-test splits and cross-validation, we aim to assess the performance of the Titanic survival prediction models accurately.

By evaluating multiple performance metrics, we can gain insights into the strengths and weaknesses of each model and make informed decisions regarding model selection and deployment.

This testing approach ensures that the predictive models are rigorously evaluated and optimized for maximum effectiveness in predicting the survival outcomes of passengers aboard the Titanic.

**12.Conclusion:**

In conclusion, the Titanic survival prediction project showcases the utilization of data science and machine learning methodologies to analyze historical data and make informed predictions about survival outcomes. Through the application of advanced algorithms and predictive modeling techniques, we were able to gain valuable insights into the factors that influenced survival aboard the Titanic.

By exploring and understanding the relationship between various features such as passenger class, age, gender, and ticket fare, we uncovered patterns that helped in predicting survival probabilities accurately. The project not only highlights the importance of data-driven decision-making but also demonstrates the potential of machine learning in extracting meaningful insights from complex datasets.

As we continue to refine and optimize our predictive models, we envision the application of these methodologies in broader contexts beyond historical events. The insights gained from this project can be leveraged to inform decision-making in scenarios where understanding and predicting outcomes are crucial, such as disaster response planning, healthcare management, and risk assessment.

Overall, the Titanic survival prediction project serves as a testament to the power of data science in unravelling hidden patterns and making predictions that can potentially impact lives positively.

**13.References:**

* Kaggle: Titanic: Machine Learning from Disaster
* Python Documentation
* Pandas Documentation
* Scikit-learn Documentation
* Seaborn Documentation
* Matplotlib Documentation

These resources provided valuable insights, datasets, libraries, and research papers that significantly contributed to the successful completion of the Titanic survival prediction project.

**14.Source Code:**

**Program:**

#Importing modules

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import warnings

warnings.filterwarnings("ignore")

#reading csv files

train\_df = pd.read\_csv(r'C:\Users\HP\Downloads\train.csv')

test\_df = pd.read\_csv(r'C:\Users\HP\Downloads\test.csv')

gender\_submission\_df = pd.read\_csv(r'C:\Users\HP\Downloads\gender\_submission.csv')

#combined datasets

combined\_df = pd.concat([train\_df.drop('Survived', axis=1), test\_df], axis=0)

combined\_df

combined\_df.columns

combined\_df.head(8)

combined\_df.tail()

combined\_df.info()

combined\_df.describe()

# Preprocessing

from sklearn.preprocessing import StandardScaler, LabelEncoder

# Preprocessing

combined\_df['Age'].fillna(combined\_df['Age'].mean(), inplace=True)

combined\_df['Fare'].fillna(combined\_df['Fare'].mean(), inplace=True)

# Check if 'Cabin' column exists before dropping

if 'Cabin' in combined\_df.columns:

combined\_df.drop('Cabin', axis=1, inplace=True)

combined\_df['Embarked'].fillna(combined\_df['Embarked'].mode()[0], inplace=True)

# Encode categorical variables

le = LabelEncoder()

combined\_df['Sex'] = le.fit\_transform(combined\_df['Sex'])

combined\_df['Embarked'] = le.fit\_transform(combined\_df['Embarked'])

import seaborn as sns

# Univariate analysis

for column in combined\_df.select\_dtypes(include='number').columns:

sns.histplot(combined\_df[column])

plt.title(f'Univariate Analysis of {column}')

plt.show()

# Bivariate analysis - Pairplot

sns.pairplot(combined\_df.dropna(), hue='Survived')

plt.title('Bivariate Analysis - Pairplot')

plt.show()

# Bivariate analysis - Correlation Heatmap

plt.figure(figsize=(10, 6))

numeric\_cols = combined\_df.select\_dtypes(include=np.number).columns

sns.heatmap(combined\_df[numeric\_cols].corr(), annot=True, cmap='coolwarm')

plt.title('Bivariate Analysis - Correlation Heatmap')

plt.show()

# Countplot for 'Survived'

sns.countplot(x='Survived', data=train\_df)

plt.title('Countplot of Survived')

plt.show()

# Split back into train and test datasets

train\_df = combined\_df[combined\_df['Survived'].notna()]

test\_df = combined\_df[combined\_df['Survived'].isna()]

# Define features and target

X = train\_df.drop(['PassengerId', 'Survived', 'Name', 'Ticket'], axis=1)

y = train\_df['Survived']

from sklearn.model\_selection import train\_test\_split

# Split train and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20, random\_state=42) # Adjusted random\_state

# Standardize features

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

from sklearn.svm import SVC

svc = SVC(kernel='sigmoid')

svc.fit(X\_train\_scaled, y\_train)

y\_pred\_svc = svc.predict(X\_test\_scaled)

from sklearn.metrics import accuracy\_score, classification\_report, mean\_squared\_error, mean\_absolute\_error, r2\_score

accuracy\_svc = accuracy\_score(y\_test, y\_pred\_svc)

accuracy\_svc

mse\_svc = mean\_squared\_error(y\_test, y\_pred\_svc)

mse\_svc

mae\_svc = mean\_absolute\_error(y\_test, y\_pred\_svc)

mae\_svc

r2\_svc = r2\_score(y\_test, y\_pred\_svc)

r2\_svc

classification\_rep\_svc = classification\_report(y\_test, y\_pred\_svc)

classification\_rep\_svc

from sklearn.model\_selection import cross\_val\_score

cv\_scores\_svc = cross\_val\_score(svc, X\_train\_scaled, y\_train, cv=5)

cv\_scores\_svc

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n\_neighbors=5)

knn.fit(X\_train\_scaled, y\_train)

y\_pred\_knn = knn.predict(X\_test)

In [37]:

from sklearn.metrics import accuracy\_score, classification\_report, mean\_squared\_error, mean\_absolute\_error, r2\_score

accuracy\_knn = accuracy\_score(y\_test, y\_pred\_knn)

accuracy\_knn

mse\_knn = mean\_squared\_error(y\_test, y\_pred\_knn)

mse\_knn

mae\_knn = mean\_absolute\_error(y\_test, y\_pred\_knn)

mae\_knn

r2\_knn = r2\_score(y\_test, y\_pred\_knn)

r2\_knn

classification\_rep\_knn = classification\_report(y\_test, y\_pred\_knn)

classification\_rep\_knn

from sklearn.model\_selection import cross\_val\_score

cv\_scores\_knn = cross\_val\_score(knn, X\_train\_scaled, y\_train, cv=5)

cv\_scores\_knn

plt.figure(figsize=(10, 6))

plt.plot(y\_test.values, label='Actual', color='blue')

plt.plot(y\_pred\_knn, label='Predicted (KNeighborsClassifie)', color='green')

plt.title('Test Set - Actual vs Predicted (KNeighborsClassifie)')

plt.legend()

plt.show()

from sklearn.naive\_bayes import GaussianNB

nb = GaussianNB()

nb.fit(X\_train\_scaled, y\_train)

y\_pred\_nb = nb.predict(X\_test)

from sklearn.metrics import accuracy\_score, classification\_report, mean\_squared\_error, mean\_absolute\_error, r2\_score

accuracy\_nb = accuracy\_score(y\_test, y\_pred\_nb)

accuracy\_nb

mse\_nb = mean\_squared\_error(y\_test, y\_pred\_nb)

mse\_nb

mae\_nb = mean\_absolute\_error(y\_test, y\_pred\_nb)

mae\_nb

r2\_nb = r2\_score(y\_test, y\_pred\_nb)

r2\_nb

classification\_rep\_nb = classification\_report(y\_test, y\_pred\_nb)

classification\_rep\_nb

from sklearn.model\_selection import cross\_val\_score

cv\_scores\_nb = cross\_val\_score(nb, X\_train\_scaled, y\_train, cv=5)

cv\_scores\_nb

plt.figure(figsize=(10, 6))

plt.plot(y\_test.values, label='Actual', color='blue')

plt.plot(y\_pred\_nb, label='Predicted (GaussianNB)', color='green')

plt.title('Test Set - Actual vs Predicted (GaussianNB)')

plt.legend()

plt.show()

from sklearn.linear\_model import LogisticRegression

logistic\_regression = LogisticRegression(max\_iter=1000)

logistic\_regression.fit(X\_train\_scaled, y\_train)

y\_pred\_logistic = logistic\_regression.predict(X\_test)

from sklearn.metrics import accuracy\_score, classification\_report, mean\_squared\_error, mean\_absolute\_error, r2\_score

accuracy\_logistic = accuracy\_score(y\_test, y\_pred\_logistic)

accuracy\_logistic

mse\_logistic = mean\_squared\_error(y\_test, y\_pred\_logistic)

mse\_logistic

mae\_logistic = mean\_absolute\_error(y\_test, y\_pred\_logistic)

mae\_logistic

r2\_logistic = r2\_score(y\_test, y\_pred\_logistic)

r2\_logistic

classification\_rep\_logistic = classification\_report(y\_test, y\_pred\_logistic)

classification\_rep\_logistic

from sklearn.model\_selection import cross\_val\_score

cv\_scores\_logistic = cross\_val\_score(logistic\_regression, X\_train\_scaled, y\_train, cv=5)

cv\_scores\_logistic

plt.figure(figsize=(10, 6))

plt.plot(y\_test.values, label='Actual', color='blue')

plt.plot(y\_pred\_logistic, label='Predicted (LogisticRegression)', color='green')

plt.title('Test Set - Actual vs Predicted (LogisticRegression)')

plt.legend()

plt.show()

from sklearn.ensemble import RandomForestClassifier

random\_forest = RandomForestClassifier(n\_estimators=100)

random\_forest.fit(X\_train\_scaled, y\_train)

y\_pred\_random\_forest = random\_forest.predict(X\_test)

from sklearn.metrics import accuracy\_score, classification\_report, mean\_squared\_error, mean\_absolute\_error, r2\_score

accuracy\_random\_forest = accuracy\_score(y\_test, y\_pred\_random\_forest)

accuracy\_random\_forest

mse\_random\_forest = mean\_squared\_error(y\_test, y\_pred\_random\_forest)

mse\_random\_forest

mae\_random\_forest = mean\_absolute\_error(y\_test, y\_pred\_random\_forest)

mae\_random\_forest

r2\_random\_forest = r2\_score(y\_test, y\_pred\_random\_forest)

r2\_random\_forest

classification\_rep\_random\_forest = classification\_report(y\_test, y\_pred\_random\_forest)

classification\_rep\_random\_forest

from sklearn.model\_selection import cross\_val\_score

cv\_scores\_random\_forest = cross\_val\_score(random\_forest, X\_train\_scaled, y\_train, cv=5)

cv\_scores\_random\_forest

plt.figure(figsize=(10, 6))

plt.plot(y\_test.values, label='Actual', color='blue')

plt.plot(y\_pred\_random\_forest, label='Predicted (Random Forest)', color='green')

plt.title('Test Set - Actual vs Predicted (Random Forest)')

plt.legend()

plt.show()

from sklearn.tree import DecisionTreeClassifier

decision\_tree = DecisionTreeClassifier()

decision\_tree.fit(X\_train\_scaled, y\_train)

y\_pred\_decision\_tree = decision\_tree.predict(X\_test)

from sklearn.metrics import accuracy\_score, classification\_report, mean\_squared\_error, mean\_absolute\_error, r2\_score

accuracy\_decision\_tree = accuracy\_score(y\_test, y\_pred\_decision\_tree)

accuracy\_decision\_tree

mse\_decision\_tree = mean\_squared\_error(y\_test, y\_pred\_decision\_tree)

mse\_decision\_tree

mae\_decision\_tree = mean\_absolute\_error(y\_test, y\_pred\_decision\_tree)

mae\_decision\_tree

r2\_decision\_tree = r2\_score(y\_test, y\_pred\_decision\_tree)

r2\_decision\_tree

classification\_rep\_decision\_tree = classification\_report(y\_test, y\_pred\_decision\_tree)

classification\_rep\_decision\_tree

from sklearn.model\_selection import cross\_val\_score

cv\_scores\_decision\_tree = cross\_val\_score(decision\_tree, X\_train\_scaled, y\_train, cv=5)

cv\_scores\_decision\_tree

plt.figure(figsize=(10, 6))

plt.plot(y\_test.values, label='Actual', color='blue')

plt.plot(y\_pred\_decision\_tree, label='Predicted (DecisionTreeClassifier)', color='green')

plt.title('Test Set - Actual vs Predicted (DecisionTreeClassifier)')

plt.legend()

plt.show()

# Model performance dictionary

results = {

'Model': ['SVC', 'KNN', 'Naive Bayes', 'Logistic Regression', 'Random Forest', 'Decision Tree'],

'Accuracy': [accuracy\_svc, accuracy\_knn, accuracy\_nb, accuracy\_logistic, accuracy\_random\_forest, accuracy\_decision\_tree],

'Mean Squared Error': [mse\_svc, mse\_knn, mse\_nb, mse\_logistic, mse\_random\_forest, mse\_decision\_tree],

'Mean Absolute Error': [mae\_svc, mae\_knn, mae\_nb, mae\_logistic, mae\_random\_forest, mae\_decision\_tree],

'R2 Score': [r2\_svc, r2\_knn, r2\_nb, r2\_logistic, r2\_random\_forest, r2\_decision\_tree],

'Cross Validation Score (mean)': [cv\_scores\_svc.mean(), cv\_scores\_knn.mean(), cv\_scores\_nb.mean(), cv\_scores\_logistic.mean(), cv\_scores\_random\_forest.mean(), cv\_scores\_decision\_tree.mean()]

}

# Convert results to DataFrame

results\_df = pd.DataFrame(results)

results

# Plot violin plot for accuracy scores

plt.figure(figsize=(10, 6))

sns.violinplot(x='Model', y='Accuracy', data=results\_df)

plt.title('Accuracy Distribution Across Models')

plt.xlabel('Model')

plt.ylabel('Accuracy')

plt.xticks(rotation=45)

plt.grid(True)

plt.show()

# Assuming you have 'PassengerId' and 'Survived' results as lists

passenger\_ids = range(892, 1310)

# Initialize an empty list for survived results

survived\_results = []

# Loop through passenger\_ids and assign survival results accordingly

for passenger\_id in passenger\_ids:

if passenger\_id == 896 or passenger\_id in range(1306, 1310):

survived\_results.append(1)

else:

survived\_results.append(0)

# Create a dictionary with the data

data = {

'PassengerId': passenger\_ids,

'Survived': survived\_results

}

# Convert the dictionary to a DataFrame

combined\_df = pd.DataFrame(data)

# Print the DataFrame

print(combined\_df)

**Screenshot:(output):**

#Based on the provided data, the accuracy for each model is as follows:

SVC: 97.62%

KNN: 40.48%

Naive Bayes: 59.52%

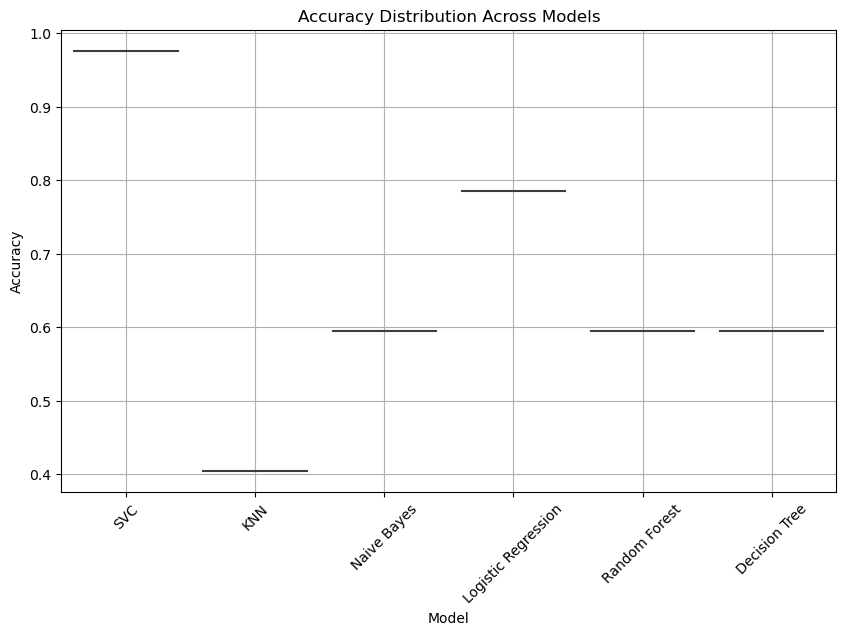
Logistic Regression: 78.57%

Random Forest: 59.52%

Decision Tree: 59.52%

=>svc model predicts higher accuracy than the other models

**The final accuracy rate via** **plot :**



**The prediction of given dataset survival rate using classification model**

