Table of Contents

Introduction

The sinking of the Titanic is one of the most infamous shipwrecks in history.

On April 15, 1912, during her maiden voyage, the widely considered “unsinkable” RMS Titanic sank after colliding with an iceberg. Unfortunately, there weren’t enough lifeboats for everyone onboard, resulting in the death of 1502 out of 2224 passengers and crew.

While there was some element of luck involved in surviving, it seems some groups of people were more likely to survive than others.

The objective of the project is to survival of the passengers based off the available data, sourced from the Kaggle competition "Titanic: Machine Learning from Disaster" (see https://www.kaggle.com/c/titanic/data). The data was already split into ‘train’ for model building and ‘test’ for validation.

Plan of Action: -

- Import the libraries.
- Import the dataset and define Data Dictionary.
- Exploratory data.
  - Data cleaning steps.
  - Hypothesis Testing.
- Model Evaluation: Goal is to find the best model with the highest accuracy.
  - Linear Regression
  - K Nearest Neighbour
  - Support Vector Machines
  - Decision Tree
  - Random Forest
- Conclusion and Next Steps.

Data Dictionary

- Libraries used sklearn, scipy.stats, pandas, numpy and seaborn.

In [3]:
```python
print(train_df.columns.tolist())
print(test_df.columns.tolist())
```

```
['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp', 'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked']
['PassengerId', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp', 'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked']
```

Variable	Definition	Key
Survived	Survival	0 = No, 1 = Yes
Pclass	Ticket class	1 = 1st, 2 = 2nd, 3 = 3rd
Sex	Sex	
Age	Age in years	
Sibsp	# of siblings / spouses aboard the Titanic	
Parch	# of parents / children aboard the Titanic	
Ticket	Ticket number	
Fare	Passenger fare	
Cabin	Cabin number	
Embarked	Port of Embarkation	C = Cherbourg, Q = Queenstown, S = Southampton

Variable Notes

Pclass: A proxy for socio-economic status (SES)
1st = Upper
2nd = Middle
3rd = Lower

Age: Age is fractional if less than 1. If the age is estimated, is it in the form of xx.5

Sibsp: The dataset defines family relations in this way...
Sibling = brother, sister, stepbrother, stepsister
Spouse = husband, wife (mistresses and fiancés were ignored)

Parch: The dataset defines family relations in this way...
Parent = mother, father

Child = daughter, son, stepdaughter, stepson
Some children travelled only with a nanny, therefore parch=0 for them.

There are 891 rows in the training dataset and 418 rows in the test dataset.

```
train_df.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
```

```
test_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 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
dtypes: float64(2), int64(4), object(5)
memory usage: 36.0+ KB
```

Exploratory Data Analysis:

Data Cleaning involves the following steps:

1. Handling missing values
2. Feature engineering
3. Removing irrelevant features

- Summary Statistics for the numerical and categorical variables:

```
train_df.describe(include=[np.number])
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

Around 38% of the passengers have survived in the training data.

	Name	Sex	Ticket	Cabin	Embarked
count	891	891	891	204	889
unique	891	2	681	147	3
top	Sedgwick, Mr. Charles Frederick Waddington	male	347082	G6	S
freq	1	577	7	4	644

- Since we have the Target variable ‘Survived’ missing in the ‘Test’ dataset, we will not be able to combine it together with ‘Test’. However when we check the distribution of the Target in the training dataset, we can see it is somewhat balanced for our analysis.

```
print(train_df.Survived.value_counts(normalize=True))
```

```
0    0.616162
1    0.383838
```

We however cannot check the same in the testing dataset.
Hence we are deciding to proceed without stratified shuffle on the train data alone.

- Printing the first few rows of the training Data :

	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

- The Correlation matrix indicate that ‘Class’ and ‘Fare’ are correlated to ‘Survived’:

```
corr = train_df.corr()
plt.figure(figsize=(10, 10))
sns.color_palette("viridis")
sns.heatmap(corr, vmax=.8, linewidths=0.01,
            square=True,annot=True,cmap='viridis',linecolor="white")
```

- Checking for null values in the data:

```
(train_df.isna().sum() *100) / train_df.shape[0]

PassengerId     0.000000
Survived        0.000000
Pclass          0.000000
Name            0.000000
Sex             0.000000
Age            19.865320
SibSp           0.000000
Parch           0.000000
Ticket          0.000000
Fare            0.000000
Cabin          77.104377
Embarked        0.224467
```

```
(test_df.isna().sum() *100) / test_df.shape[0]

PassengerId     0.000000
Pclass          0.000000
Name            0.000000
Sex             0.000000
Age            20.574163
SibSp           0.000000
Parch           0.000000
Ticket          0.000000
Fare            0.239234
Cabin          78.229665
Embarked        0.000000
```

Actionable: It is observed that we will have to treat the missing values on 'Age' and 'Embarked'

Decision taken: It is also observed that in both train and test, a high percentage of missing values in 'Cabin' and hence we are deciding to not use it in our analysis.

- Imputed 'Embarked' with the mode of the variable and binary coded the categories, such that 'S' = 0, 'C' =1 and 'Q' =2.

```
train_df['Embarked'] = train_df['Embarked'].map({"S": 0, "C": 1, "Q": 2}).astype('category')
test_df['Embarked'] = test_df['Embarked'].map({"S": 0, "C": 1, "Q": 2}).astype('category')
```

- Imputed 'Age' with a random number between *(mean – std) and (mean + std)* of the 'Age'.
  Converted 'Age' into binned categorical variables by checking distribution of survivors.

```
train_df['Band'] = pd.cut(train_df['Age'], 5)
train_df[['Band', 'Survived']].groupby(['Band'], as_index=False).mean().sort_values(by='Band', ascending=True)
```

	Band	Survived
0	(0.34, 16.336]	0.550000
1	(16.336, 32.252]	0.369942
2	(32.252, 48.168]	0.404255
3	(48.168, 64.084]	0.434783
4	(64.084, 80.0]	0.090909

Using these band ranges, changed the 'Age' column to categorical values of 0, 1, 2, 3 and 4

- Imputed 'Fare' with the median of the variable.

  Converted 'Fare' into binned categorical variables by checking distribution of survivors.

```
train_df['FareBand'] = pd.qcut(train_df['Fare'], 4)
train_df[['FareBand', 'Survived']].groupby(['FareBand'], as_index=False).mean().sort_values(by='FareBand', ascending=True)
```

	FareBand	Survived
0	(-0.001, 7.91]	0.197309
1	(7.91, 14.454]	0.303571
2	(14.454, 31.0]	0.454955
3	(31.0, 512.329]	0.581081

Using these band ranges, changed 'Fare' to categorical values of 0, 1, 2 and 3.

- Converted 'Pclass' to be read a categorical ordinal variable instead of a numerical variable.

```
train_df['Pclass']=train_df['Pclass'].astype('category')
```

- Converted 'Sex' into a categorical variable by mapping 'male' to 0 and 'female' to 1.

```
train_df['Sex'] = train_df['Sex'].map( {'female': 1, 'male': 0} ).astype(int)
test_df['Sex'] = test_df['Sex'].map( {'female': 1, 'male': 0} ).astype(int)
```

- Combined 'Parch' and 'SibSp', to form a new variable denoting family size.
  And converted into a single variable, denoting if they were travelling alone or not as 'IsAlone'.

```
train_df['FamilySize'] = train_df['SibSp'] + train_df['Parch'] + 1
test_df['FamilySize'] = test_df['SibSp'] + test_df['Parch'] + 1
```

```
train_df['IsAlone'] = 0
test_df['IsAlone'] = 0
train_df.loc[train_df['FamilySize'] == 1, 'IsAlone'] = 1
test_df.loc[test_df['FamilySize'] == 1, 'IsAlone'] = 1
```

- We decided to exclude 'Name', 'PassengerId' and 'Ticket' because of the distinct nature of the attribute and it would not add any predictive power to the model. An additional note on 'Ticket' is that the variable has more than 76 %(681/891) of the data as unique and hence we decided to exclude them as well.

```
train_df['Name'].nunique()
```

```
891
```

```
train_df['PassengerId'].nunique()
```

```
891
```

```
train_df['Ticket'].nunique()
```

```
681
```

- The final transformed version of the data looks like this:

```
train_df.head(10)
```

	Survived	Pclass	Sex	Age	Fare	Embarked	IsAlone
0	0	3	0	1	0	0	0
1	1	1	1	2	3	1	0
2	1	3	1	1	1	0	1
3	1	1	1	2	3	0	0
4	0	3	0	2	1	0	1
5	0	3	0	2	1	2	1
6	0	1	0	3	3	0	1
7	0	3	0	0	2	0	0
8	1	3	1	1	1	0	0
9	1	2	1	0	2	1	0

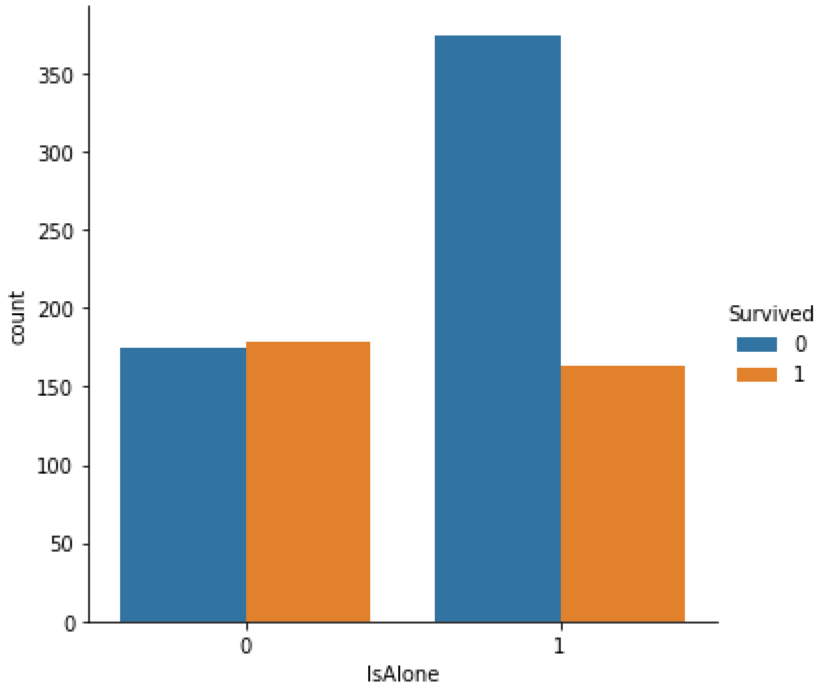
Exploration and Hypothesis testing:

1. Were 'People travelling with family' play a factor for survival?

Null Hypothesis: Passengers travelling *with* family had a higher survival rate.

Alternative Hypothesis: Passengers travelling *without* family had a higher survival rate.

Plotting the chart between 'IsAlone' and 'Survived', shows that people travelling with family had an equal chance of survival but people travelling alone had much lesser chance of survival.



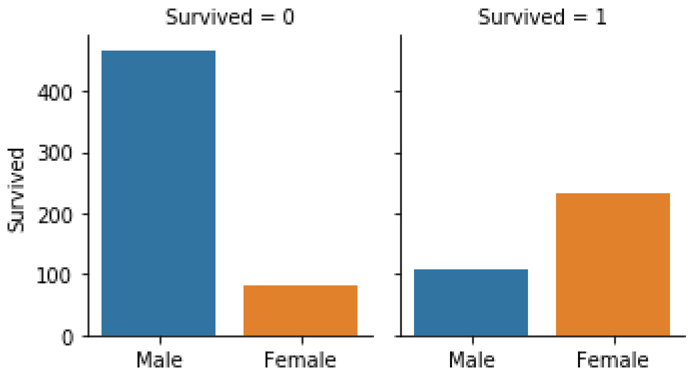
The Chi-squared statistics on a significance level of 5% indicate that there is relationship between people travelling alone and survival rate.

```
Degree of Freedom:  1
chi-square statistic:  36.85013084754587
critical_value:  3.841458820694124
p-value: 1.2756752321152476e-09
Significance level:  0.05
Degree of Freedom:  1
```

Hence we accept the Null Hypothesis.

2. Does gender play a role on Survival?
   Null Hypothesis: Females have a higher chance of survival.
   Alternate Hypothesis: Females do not have a higher chance of survival.

   Plotting the chart between 'Sex' and 'Survived', we see that the number of females survived almost twice the number of males.

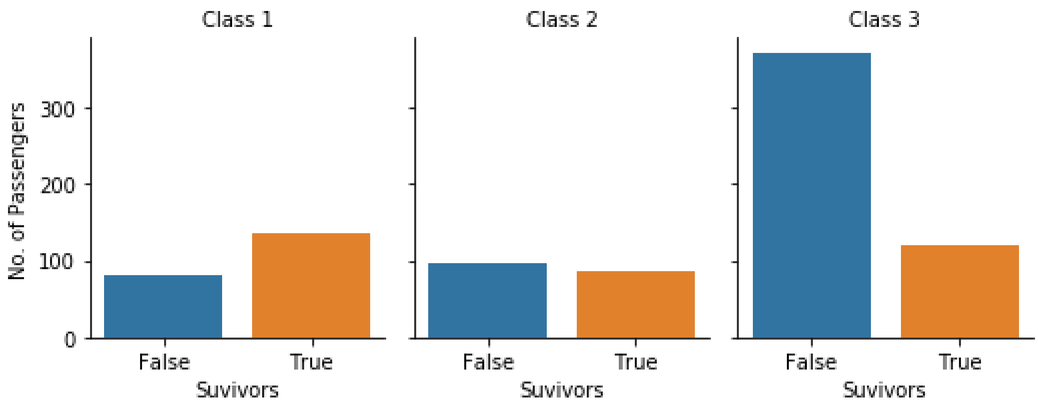


The Chi-squared statistics on a significance level of 5% indicate that there is relationship between Gender and survival rate.

```
Degree of Freedom:  1
chi-square statistic:  263.05057407065567
critical_value:  3.841458820694124
p-value: 0.0
Significance level:  0.05
```

Hence we accept the Null Hypothesis.

3. Does Socio-Economics play a factor on Survival?
   Null Hypothesis: The ticket class of the passengers do not have any relation with the chance of survival.
   Alternate Hypothesis: The ticket class of the passengers has a direct relation with the chance of survival.
   Plotting the graph of ticket class versus passengers who survived, we see that passengers with a 3rd class ticket constituted the majority of fatality.



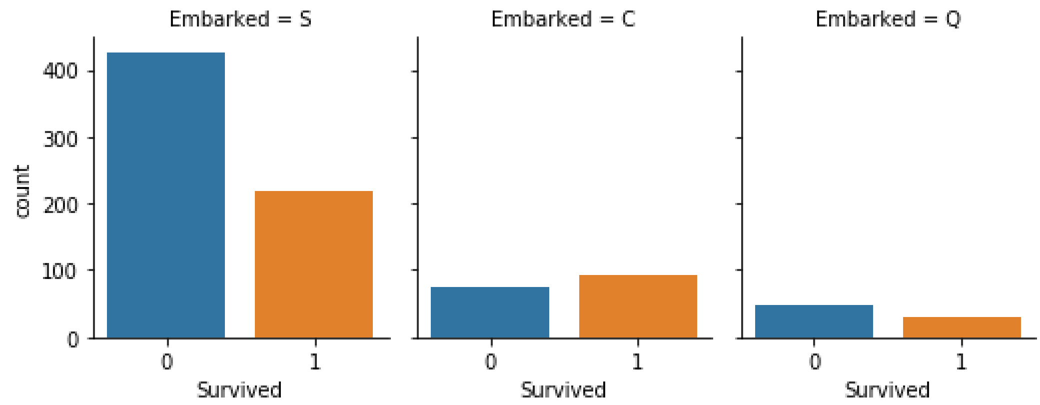
The Chi-squared statistics on a significance level of 5% indicate that there is relationship between Ticket Class and survival rate.

```
Degree of Freedom:  1
chi-square statistic:  61.335917863975695
critical_value:  3.841458820694124
p-value: 4.773959005888173e-15
Significance level:  0.05
```

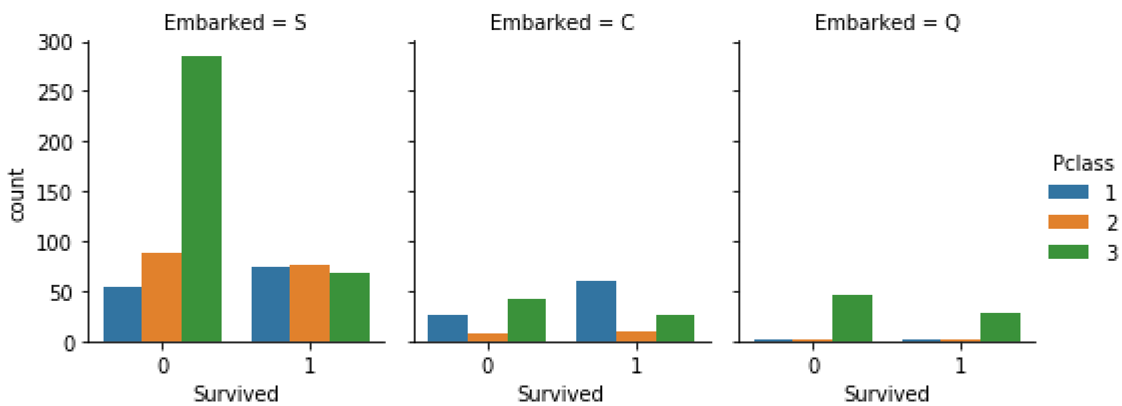
Hence we reject the ***Null hypothesis*** because there is a relation between the variables.

4. Does the place of embankment.
Null Hypothesis: Passengers' place of origin did not have a relation to survival.
Alternate Hypothesis: Passengers' place of origin has a relation to survival.

Plotting the distribution of place of embankment and survival, we see that most causalities were from Southampton.



Further looking at the Ticket class of the passengers from their place of origin, we see that most people who did not survive from Southampton were holding a 3rd class ticket. Further solidifying our assumption of Socio-Economic factor with survival.



The Chi-squared statistics on a significance level of 5% indicate that there is relationship between place of origin and survival rate.

```
Degree of Freedom:  1
chi-square statistic:  20.845263081636034
critical_value:  3.841458820694124
p-value: 4.9792216417765545e-06
Significance level:  0.05
```

Hence we reject the ***Null Hypothesis*** that place of origin was not a factor of survival rate.

Model Evaluation:

[Disclaimer: The only reason I am not showing the confusion matrices of the following classifier is because of the prediction feature being missing from our testing dataset. Hence we will be gauging model performance by accuracy alone]

1. Logistic Regression:

```
# Logistic Regression
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression()
logreg.fit(X_train, Y_train)
Y_pred = logreg.predict(X_test)
acc_log = round(logreg.score(X_train, Y_train) * 100, 2)
acc_log
```

```
79.12
```

Looking at the most significant drivers, we see that Gender and Class held the most weight in determining survival.

```
coeff_df = pd.DataFrame(train_df.columns.delete(0))
coeff_df.columns = ['Feature']
coeff_df["Correlation"] = pd.Series(logreg.coef_[0])

coeff_df.sort_values(by='Correlation', ascending=False)
```

	Feature	Correlation
1	Sex	2.481312
4	Embarked	0.311042
3	Fare	-0.030062
5	IsAlone	-0.066846
2	Age	-0.495858
0	Pclass	-1.170271

2. KNN

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(X_train, Y_train)
Y_pred = knn.predict(X_test)
acc_knn = round(knn.score(X_train, Y_train) * 100, 2)
acc_knn
```

```
82.72
```

3. SVM

   We will check for both Support Vector Classifier and Linear Support Vector Classifier.

```python
# Linear SVC
from sklearn.svm import  LinearSVC

linear_svc = LinearSVC()
linear_svc.fit(X_train, Y_train)
Y_pred = linear_svc.predict(X_test)
acc_linear_svc = round(linear_svc.score(X_train, Y_train) * 100, 2)
acc_linear_svc
```

79.01

```python
# Support Vector Machines
from sklearn.svm import SVC
svc = SVC()
svc.fit(X_train, Y_train)
Y_pred = svc.predict(X_test)
acc_svc = round(svc.score(X_train, Y_train) * 100, 2)
acc_svc
```

82.04

4. Decision Tree

```python
# Decision Tree
from sklearn.tree import DecisionTreeClassifier

decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, Y_train)
Y_pred = decision_tree.predict(X_test)
acc_decision_tree = round(decision_tree.score(X_train, Y_train) * 100, 2)
acc_decision_tree
```

85.97

5. Random Forest

```python
# Random Forest
from sklearn.ensemble import RandomForestClassifier
random_forest = RandomForestClassifier(n_estimators=100)
random_forest.fit(X_train, Y_train)
Y_pred = random_forest.predict(X_test)
random_forest.score(X_train, Y_train)
acc_random_forest = round(random_forest.score(X_train, Y_train) * 100, 2)
acc_random_forest
```

85.97

Comparing the models:

```python
models = pd.DataFrame({
    'Model': ['Support Vector Machines', 'KNN', 'Logistic Regression',
              'Random Forest', 'Linear SVC',
              'Decision Tree'],
    'Score': [acc_svc, acc_knn, acc_log,
              acc_random_forest, acc_linear_svc, acc_decision_tree]})
models.sort_values(by='Score', ascending=False)
```

	Model	Score
3	Random Forest	85.97
5	Decision Tree	85.97
1	KNN	82.72
0	Support Vector Machines	82.04
2	Logistic Regression	79.12
4	Linear SVC	79.01

From the initial analysis, we observe that random forest gives the best accuracy and we can consider using it for further implementation.

Conclusions and Next steps:

From the data exploration to hypothesis testing we conclude that,

- Women had higher chances of survival.
- People travelling alone had a higher chance of not surviving.
- Class (Socio-Economic status) of the passengers had played a role in their survival.
- Passengers embarking from Southampton had a higher chance of not surviving. We also see that this may be due to the class disparity of people originating from Southampton.

There were some limitation for this dataset such as missing values for some attributes of passengers. This is not in any form an exhaustive study. More can be done on this data set.

The accuracy can further be improved for the different model by finding the optimal parameters using cross validation, I was unable to run the cross validation steps because of system limitations.