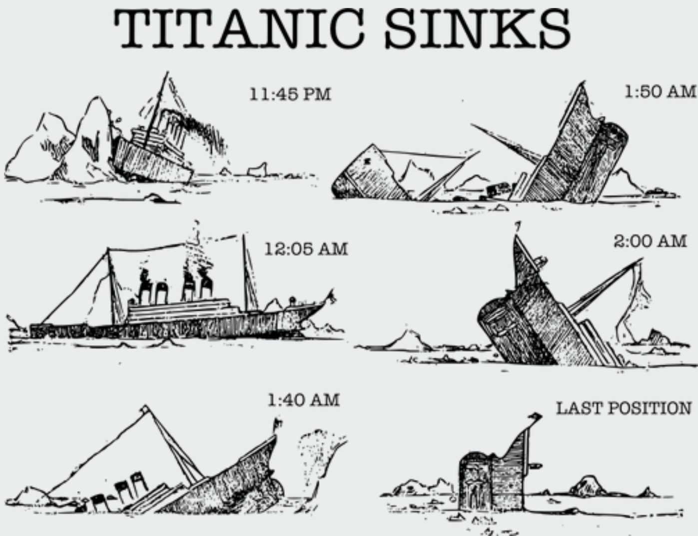
TITANIC SURVIVOR PREDICTION USING MACHINE LEARNING

BY - TANISHQ SOIN, AI-ML LIVE TUT G2 7-8PM



1.INTRODUCTION

RMS *Titanic* was a British passenger liner operated by the White Star Line that sank in the North Atlantic Ocean in the early morning hours of 15 April 1912, after striking an iceberg during her maiden voyage from Southampton to New York City. Of the estimated 2,224 passengers and crew aboard, more than 1,500 died, making the sinking one of modern history's deadliest peacetime commercial marine disasters. RMS *Titanic* was the largest ship afloat at the time she entered service and was the second of three *Olympic*-class ocean liners operated by the White Star Line. She was built by the Harland and Wolff shipyard in Belfast. Thomas Andrews, chief naval architect of the shipyard at the time, died in the disaster.

2.AIM

- The purpose of this project is to document the process I went through to create my predictions for ***Titanic Survivor Predictions.***
- The objective of this model is to build a classification model that could successfully determine whether a Titanic passenger lived or died.

3.SOFTWARE REQUIRED

1. TOOLS USED
   a. ANACONDA NAVIGATOR 1.9.6
   b. JUPYTER NOTEBOOKS 5.7.5
2. LIBRARIES USED
   a. ANALYZING: Numpy, Pandas, Sci-kit Learn
   b. VISUALIZATION: Matplotlib, Seaborn

4.MODELS USED

1. LOGISTIC REGRESSION
2. DECISION TREE CLASSIFICATION
3. RANDOM FOREST CLASSIFICATION
4. SUPPORT VECTOR CLASSIFICATION

4.1 LOGISTIC REGRESSION

Logistic regression is a statistical **model** that in its basic form uses a **logistic** function to **model** a binary dependent variable, although many more complex extensions exist. In **regression** analysis, **logistic regression** (or **logit regression**) is estimating the parameters of a **logistic model** (a form of binary **regression**).

4.2 DECISION TREE CLASSIFICATION

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with **decision nodes** and **leaf nodes**.

4.3 RANDOM FOREST CLASSIFICATION

The **random forest** is a **classification** algorithm consisting of many decisions trees. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated **forest** of trees whose prediction by committee is more accurate than that of any individual tree.

4.4 SUPPORT VECTOR CLASSIFICATION

support vector machines (SVMs) are a set of supervised learning methods used for **classification**, regression and outliers detection. The advantages of **support vector** machines are: Effective in high dimensional spaces. Still effective in cases where number of dimensions is greater than the number of samples.

5. IMPLEMENTATION

- Importing the necessary libraries.
- Importing the Dataset.
- Cleaning and analyzing the Dataset.
- Building the models.
- Cross-validating the models for best prediction.

5.1 IMPORTING THE NECESSARY LIBRARIES

```
import warnings

warnings.filterwarnings('ignore')

import pandas as pd

import matplotlib.pyplot as plt

%matplotlib inline

import seaborn as sns

from sklearn.preprocessing import StandardScaler as ss
```

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
import numpy as np
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
```

5.2 READ AND EXPLORE DATA

```
import pandas as pd
df = pd.read_csv('./TrainingData/train.csv')
df.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

```
df.describe()
```

	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

5.3 CLEANING AND ANALYZING THE DATA

```python
df1 = df[['Pclass', 'Age', 'Fare']]
```

```python
df1.head()
```

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

```python
df2 = pd.get_dummies(df['Sex'])
df2.head()
```

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

```python
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 3 columns):
Pclass     891 non-null int64
Age        891 non-null float64
Fare       891 non-null float64
dtypes: float64(2), int64(1)
memory usage: 21.0 KB
```

```python
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 2 columns):
female     891 non-null uint8
male       891 non-null uint8
dtypes: uint8(2)
memory usage: 1.8 KB
```

```python
final_data = pd.concat((df1, df2), axis=1)
final_data.head(10)
```

	Pclass	Age	Fare	female	male
0	3	22.0	7.2500	0	1

```python
X  = final_data.values
```

```python
y = df.Survived.values
```

```python
X.shape
```

```
(891, 5)
```

```python
y.shape
```

```
(891,)
```

```python
from sklearn.preprocessing import StandardScaler as ss
scale = ss()
```

```python
X = scale.fit_transform(X)
print(X)
```

```
[[ 0.82737724 -0.56573646 -0.50244517 -0.73769513  0.73769513]
 [-1.56610693  0.66386103  0.78684529  1.35557354 -1.35557354]
 [ 0.82737724 -0.25833709 -0.48885426  1.35557354 -1.35557354]
 ...
 [ 0.82737724 -0.1046374  -0.17626324  1.35557354 -1.35557354]
 [-1.56610693 -0.25833709 -0.04438104 -0.73769513  0.73769513]
 [ 0.82737724  0.20276197 -0.49237783 -0.73769513  0.73769513]]
```

```python
from sklearn.model_selection import train_test_split
```

```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 1 )
```

5.4 BUILDING THE MODELS

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC

import numpy as np
```

```
dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)
y_pred_dtc = dtc.predict(X_test)
```

```
from sklearn.metrics import accuracy_score, confusion_matrix
cm_dtc = confusion_matrix(y_test, y_pred_dtc)
acc_dtc = accuracy_score(y_test, y_pred_dtc)
print(cm_dtc)
```

```
[[91 15]
 [26 47]]
```

```
print(acc_dtc)
```

```
0.770949720670391
```

```
from sklearn.tree import export_graphviz

export_graphviz(dtc, out_file='./tree.dat')
```

```
models = []
models.append(('LR', LogisticRegression()))
models.append(('RFC', RandomForestClassifier()))
models.append(('DTC', DecisionTreeClassifier()))
models.append(('SVM', SVC(kernel='rbf')))

seed = 7
results = []
names = []
scoring = 'accuracy'
```

```
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
```

```
for name, model in models:
    kfold = KFold(n_splits=10, random_state=seed)
    cv_results = cross_val_score(model, X, y, cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

5.5 CROSS VALIDATION

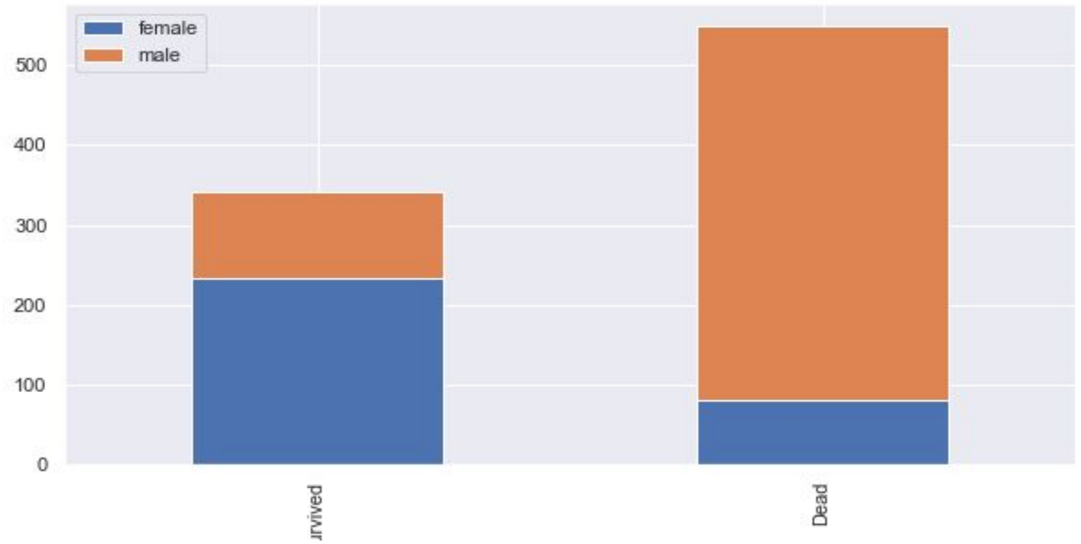
```
for name, model in models:
    kfold = KFold(n_splits=10, random_state=seed)
    cv_results = cross_val_score(model, X, y, cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

```
LR: 0.785630 (0.026821)
RFC: 0.817116 (0.037930)
DTC: 0.788976 (0.036635)
SVM: 0.802534 (0.041996)
```

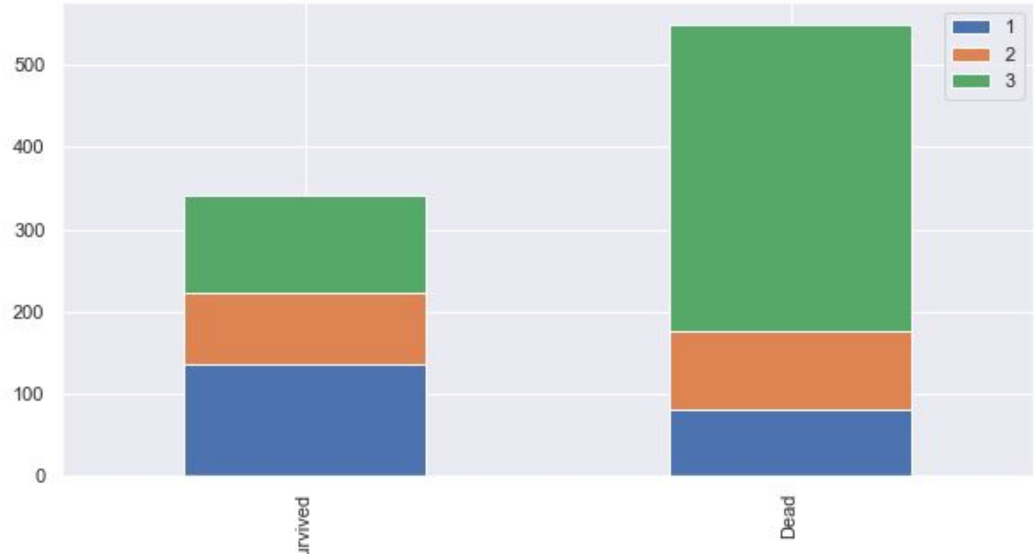
6. DATA VISUALIZATION

Seeing how individual variables are affecting 'Survival'.

6.1.Sex



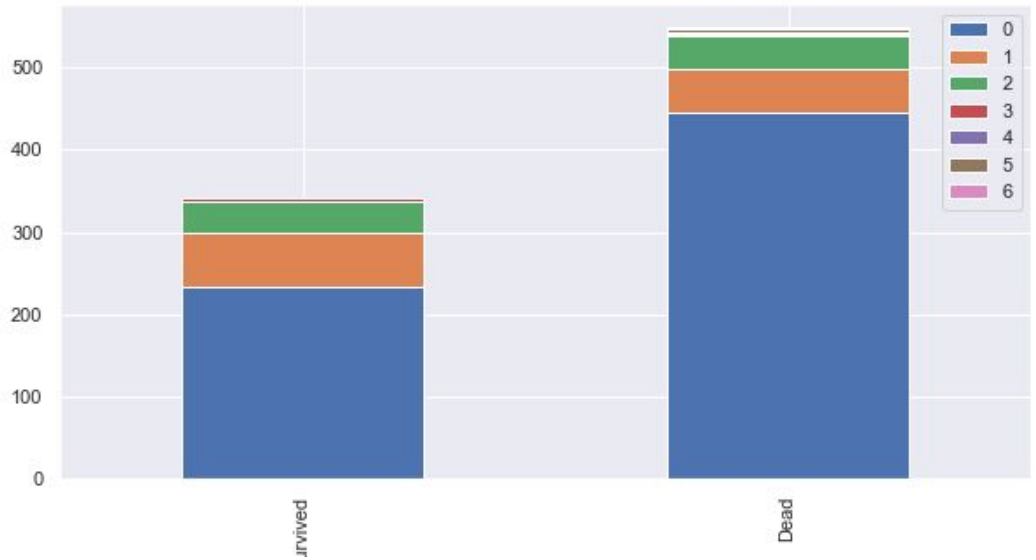
6.2. Pclass



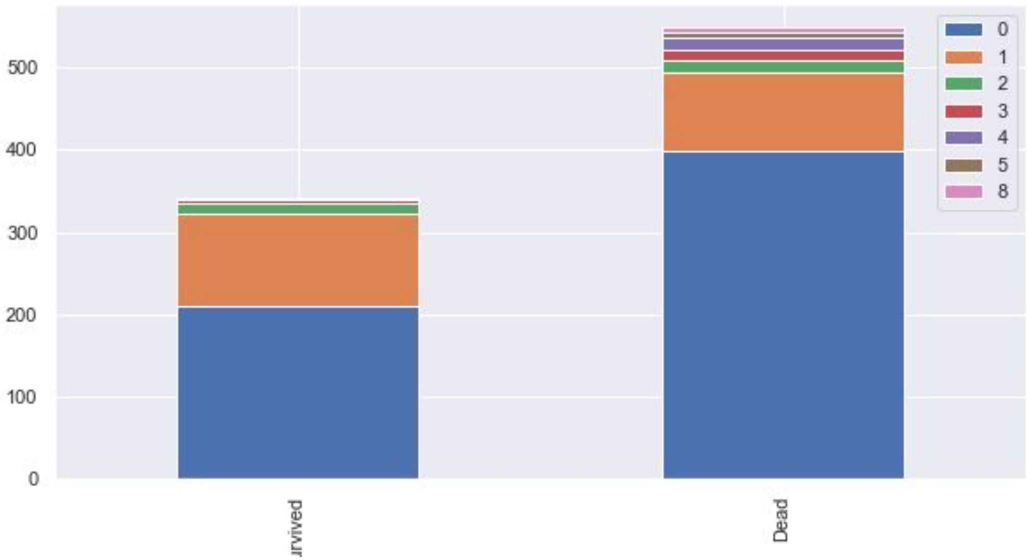

```
Survived :
 1     136
3     119
2      87
Name: Pclass, dtype: int64
Dead:
 3     372
2      97
1      80
Name: Pclass, dtype: int64
```

6.3.Parch

```
Survived :
 0    233
1     65
2     40
3      3
5      1
Name: Parch, dtype: int64
Dead:
 0    445
1     53
2     40
5      4
4      4
3      2
6      1
Name: Parch, dtype: int64
```



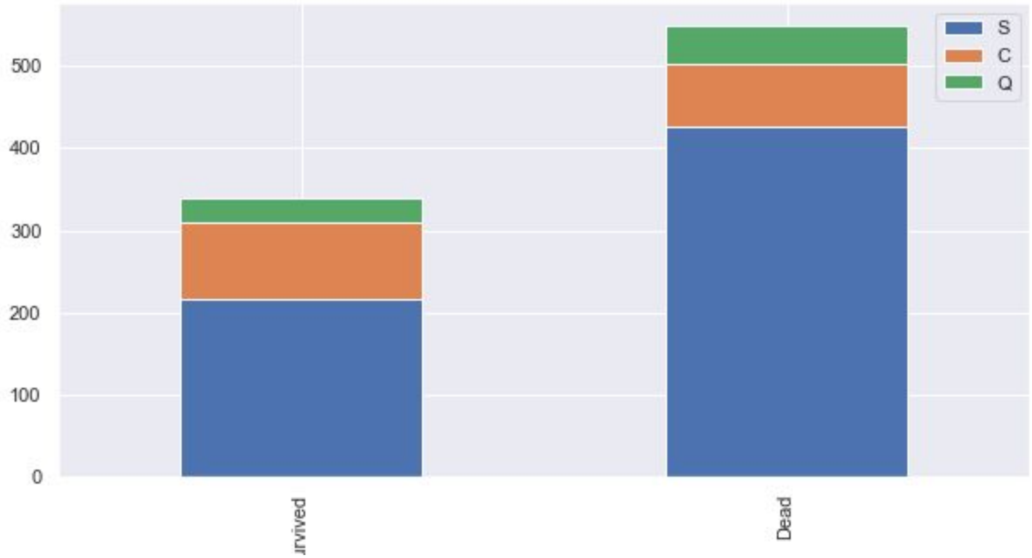
6.4.SibSp



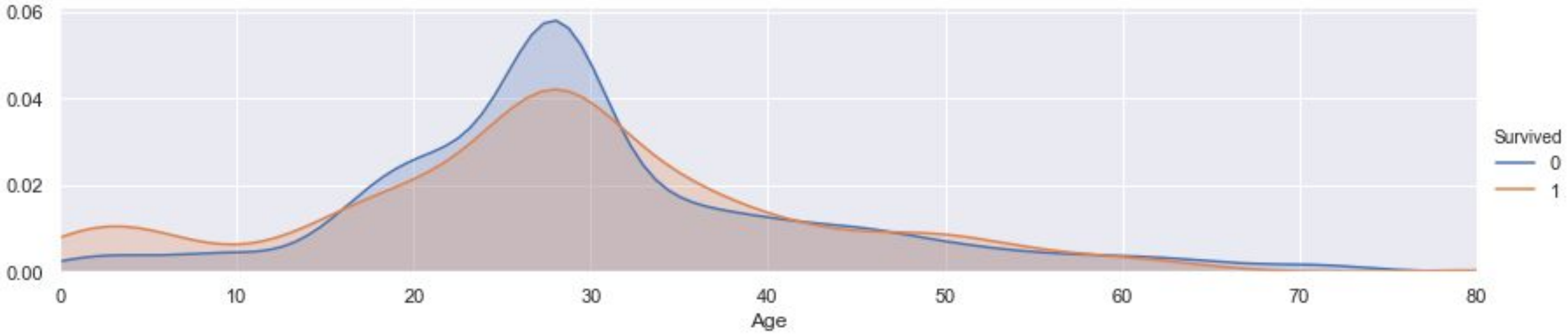

```
Survived :
 0    210
1    112
2     13
3      4
4      3
Name: SibSp, dtype: int64
Dead:
 0    398
1     97
4     15
2     15
3     12
8      7
5      5
Name: SibSp, dtype: int64
```

6.5. Embarked

```
Survived :
 S    217
C     93
Q     30
Name: Embarked, dtype: int64
Dead:
 S    427
C     75
Q     47
Name: Embarked, dtype: int64
```

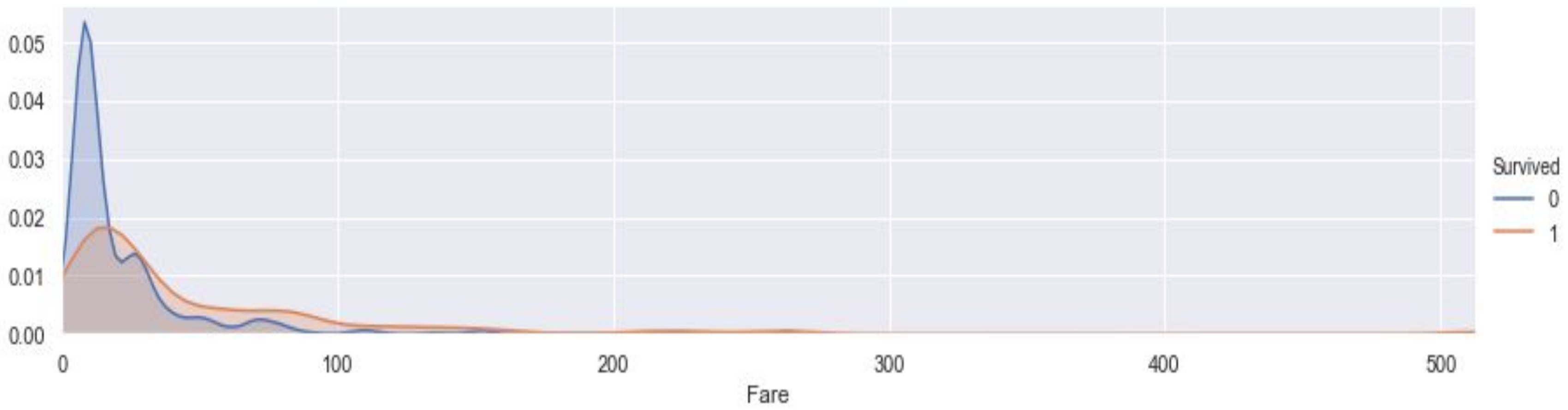


6.6.Age




```
<Figure size 432x288 with 0 Axes>
```

6.7. Fare

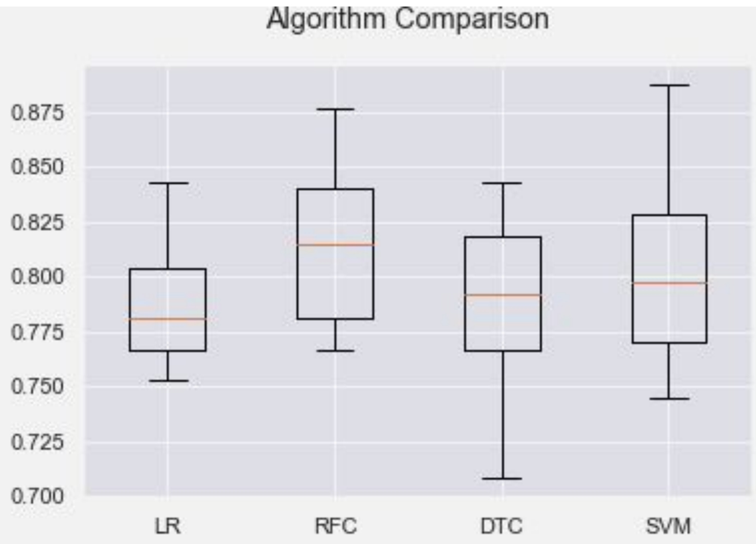


7. RESULTS- CALCULATION ACCURACY

ALGORITHM	ACCURACY(%)	STD DEV
Logistic Regression	78.5630	0.026821
Decision Tree Classification	81.7116	0.037930
Random Forest Classification	78.8976	0.036635
SVM Classification	80.2534	0.041996

```
LR: 0.785630 (0.026821)
RFC: 0.817116 (0.037930)
DTC: 0.788976 (0.036635)
SVM: 0.802534 (0.041996)
```

8.BOXPLOT OF ALGORITHMS



<Figure size 432x288 with 0 Axes>

9. CONCLUSIONS

- I have removed variables like “Passenger Id”, “Name”, “Ticket”, “cabin”, “Parch”, “Embarked” and “SibSp” as they are not affecting the target variables much.
- Women, Children and Ist class passengers had a better chance of survival.
- And I am getting an accuracy of 80.2534 % with SVC model.
- And I am getting an accuracy of 1.71168 % with DTC model.

THANK YOU :')