Titanic Survival Prediction Using Machine Learning

Challenge

Predict whether a passenger on the titanic would have been survived or not

Data

Column Name - customers.csv	Description
Survival	Survival (0 = No; 1 = Yes). Not included in test.csv file
Pclass	Ticket Class/ A Proxy for socio-economic status(SES) (1 = 1st/Upper ; 2 = 2nd/Middle; 3 = 3rd/Lower)
Name	Name
Sex	Sex
Age	Age is fractional if less than 1. If the age is estimated, is it in the form of xx.5
Sibsp	Number of Siblings (brother, sister, stepbrother, stepsister) /Spouses (husband, wife (mistresses and fiancés were ignored)) Aboard
Parch	Number of Parents (mother, father)/Children (daughter, son, stepdaughter, stepson) Aboard; Some children travelled only with a nanny, therefore parch=0 for them.
Ticket	Ticket Number
Fare	Passenger Fare
Cabin	Cabin
Embarked	Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)

Workflow stages

The competition solution workflow goes through seven stages described in the Data Science Solutions book.

1. Question or problem definition
2. Wrangle, prepare, cleanse the data
3. Exploratory Data Analysis
4. Acquire training and testing data
5. Model, predict and solve the problem

1. Question or problem definition

The competition is simple: Use the Titanic passenger data (name, age, price of ticket, etc.) to try to predict who will survive and who will die. This is a **binary classification.**

Binary classification is the task of [classifying](#) the elements of a [set](#) into two groups on the basis of a [classification rule](#).

Printing first 5 rows of dataset.

```
train_df.head(5)
```

	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 values in the second column (**"Survived"**) can be used to determine whether each passenger survived or not:

- if it's a "**1**", the passenger survived.
- if it's a "**0**", the passenger died.

2. Wrangle, prepare, cleanse the data

Here is my workflow for cleaning the data. I am not going to show all the step here, you can check my Kaggle for full code. Check my Kaggle and GitHub to took a look at all my projects❀

Duplicate records

```
#Find the number duplicate record
print('train_df - Number of duplicate Record:', train_df.duplicated().sum())

print('test_df - Number of duplicate Record:', test_df.duplicated().sum())
```

train_df — Number of duplicate Record: 0

test_df — Number of duplicate Record: 0

Missing Values

```
#Find the number of null per each columns
print('Columns in train_df with null values:\n')
print(train_df.isnull().sum())
print("-"*30)

print('Columns in test_df with null values:\n')
print(test_df.isnull().sum())
print("-"*30)
```

```
Columns in train_df with null values:

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
------------------------------
Columns in test_df with null values:

PassengerId      0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
------------------------------
```

Age : Contains 177 null values out of 891 entries. Imputed with median values for Age across sets of Pclass and Gender feature combinations

```
for dataset in combine:
    for i in range(0, 2):
        for j in range(0, 3):
            guess_df = dataset[(dataset['Sex'] == i) & (dataset['Pclass'] == 
j+1)]['Age'].dropna()

            # age_mean = guess_df.mean()
            # age_std = guess_df.std()
            # age_guess = rnd.uniform(age_mean - age_std, age_mean + age_std)

            age_guess = guess_df.median()

            # Convert random age float to nearest .5 age
            guess_ages[i,j] = int( age_guess/0.5 + 0.5 ) * 0.5

    for i in range(0, 2):
        for j in range(0, 3):
            dataset.loc[ (dataset.Age.isnull()) & (dataset.Sex == i) & 
(dataset.Pclass == j+1),'Age'] = guess_ages[i,j]

    dataset['Age'] = dataset['Age'].astype(int)
```

Embarked : Contains 2 null values. Imputed with mode of training data.

```
#Fill the null value of Embarked with the most common occurance

for dataset in combine:
    dataset['Embarked'] = dataset['Embarked'].fillna(freq_port)
```

Fare: Contains 1 null values. Imputed with mean of training data.

```
for dataset in combine:
    dataset['Fare'].fillna(dataset['Fare'].dropna().mean(), inplace=True)
    dataset['Fare'] = dataset['Fare'].astype(np.int64)
```

Cabin : 687 out of 891 Cabin entries are null , i.e. more than 50 % of the total data exists. **Deck**, is created because it is slightly more general than **Cabin**.

```
for dataset in combine:
    dataset['Deck'] = dataset['Cabin'].str.slice(0,1)
    dataset['Deck'] = dataset['Deck'].map({"A": 1, "B": 2, "C": 3, "D": 4, 
"E": 5, "F":6,"G":7, "T":8})
    dataset['Deck'] = dataset['Deck'].fillna(0)
    dataset['Deck'] = dataset['Deck'].astype(np.int64)
```

Creating new feature extracting from existing (Add Computed Column)

Title : I want to analyze if Name feature can be engineered to extract titles and test correlation between titles and survival, before dropping Name and PassengerId features

```
for dataset in combine:
    dataset['Title']  = dataset.Name.str.extract(' ([A-Za-z]+)\.', 
expand=False)pd.crosstab(train_df["Title"], train_df['Sex'])
```

Sex	0	1
Title		
Capt	1	0
Col	2	0
Countess	0	1
Don	1	0
Dr	6	1
Jonkheer	1	0
Lady	0	1
Major	2	0
Master	40	0
Miss	0	182
Mlle	0	2
Mme	0	1
Mr	517	0
Mrs	0	125
Ms	0	1
Rev	6	0
Sir	1	0

I replace many titles with a more common name or classify them as Rare.

```
for dataset in combine:
    dataset['Title'] = dataset['Title'].replace(['Lady', 'Countess','Capt', 
'Col','Don', 'Dr', 'Major', 'Rev', 'Sir', 'Jonkheer', 'Dona'], 'Rare')
    dataset['Title'] = dataset['Title'].replace('Mlle', 'Miss')
    dataset['Title'] = dataset['Title'].replace('Ms', 'Miss')
```

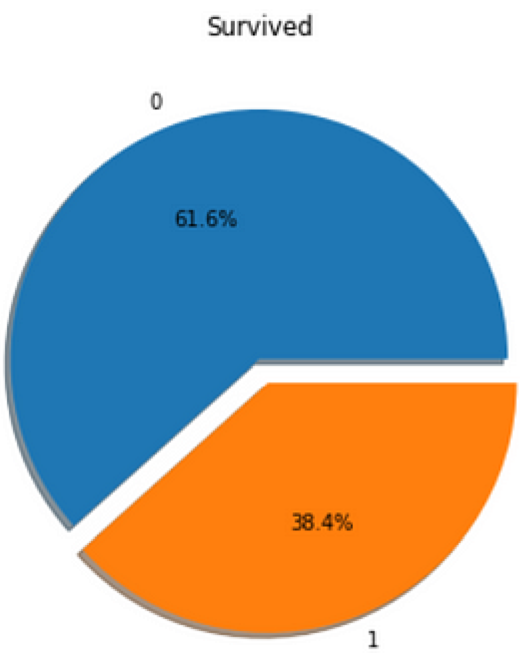
```
    dataset['Title'] = dataset['Title'].replace('Mme', 'Mrs')for dataset in combine:
    dataset['Title'] = dataset['Title'].map({"Mr": 1, "Miss": 2, "Mrs": 3, "Master": 4, "Rare": 5})
    dataset['Title'] = dataset['Title'].fillna(0)
```

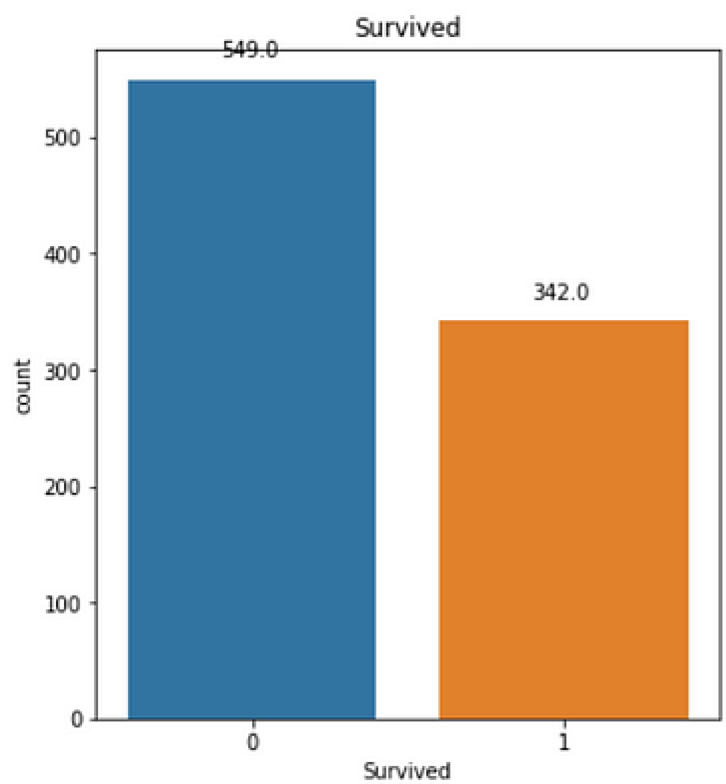
FamilySize : I create a new feature for FamilySize which combines Parch and SibSp. This will enable us to drop Parch and SibSp from our datasets.

```
for dataset in combine:
    dataset['FamilySize'] = dataset['SibSp'] + dataset['Parch'] + 1for dataset in combine:
    dataset['IsAlone'] = 0
    dataset.loc[dataset['FamilySize'] == 1, 'IsAlone'] = 1
```

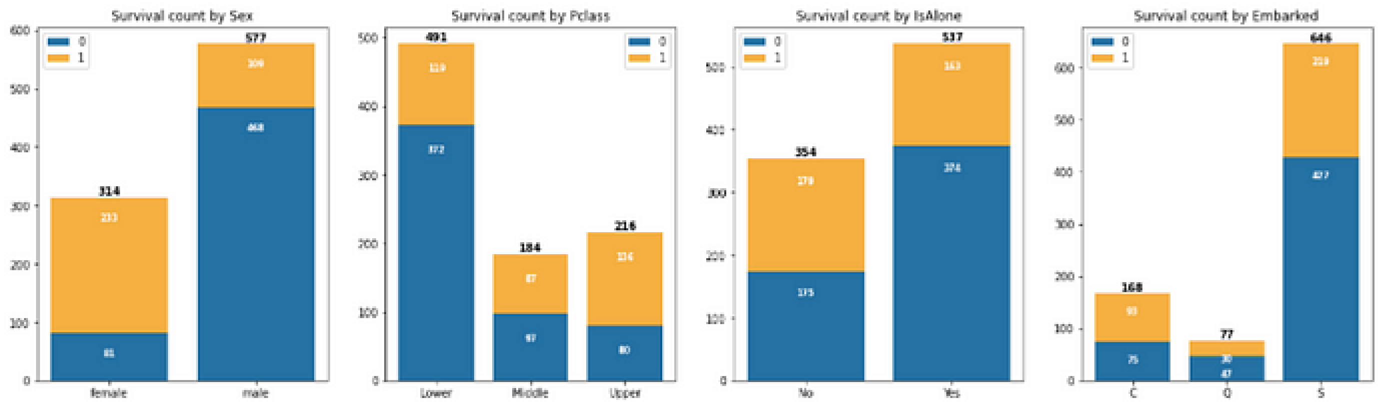
3. Exploratory Data Analysis

Analyze, identify patterns, and explore the data Analysis





Only **350** out of 891 passengers (**38.4%**) survived in the training set.



Sex

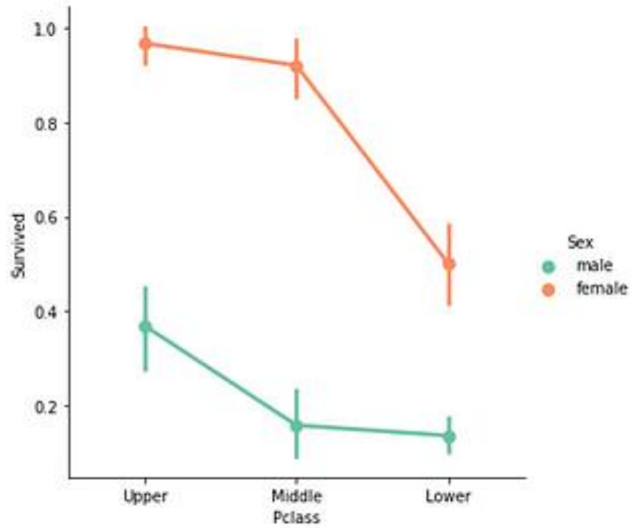
- The number of men on board the ship is much higher than the number of women, but the number of women saved is more than twice that of the number of males survived.
- The survival rates for a women on the ship is around 75% while that for men in around 19%.

Pclass

- Passenegers Of Pclass 1 has a very high priority to survive.
- The number of Passengers in Pclass 3 were a lot higher than Pclass 1 and Pclass 2, but still the number of survival from Pclass 3 is low compare to them
- Pclass 1 %survived is around 63%, for Pclass2 is around 48%, and Pclass3 survived is around 25%

Sex and class are important factors for survival. Let's determine survival rate based on sex and class together

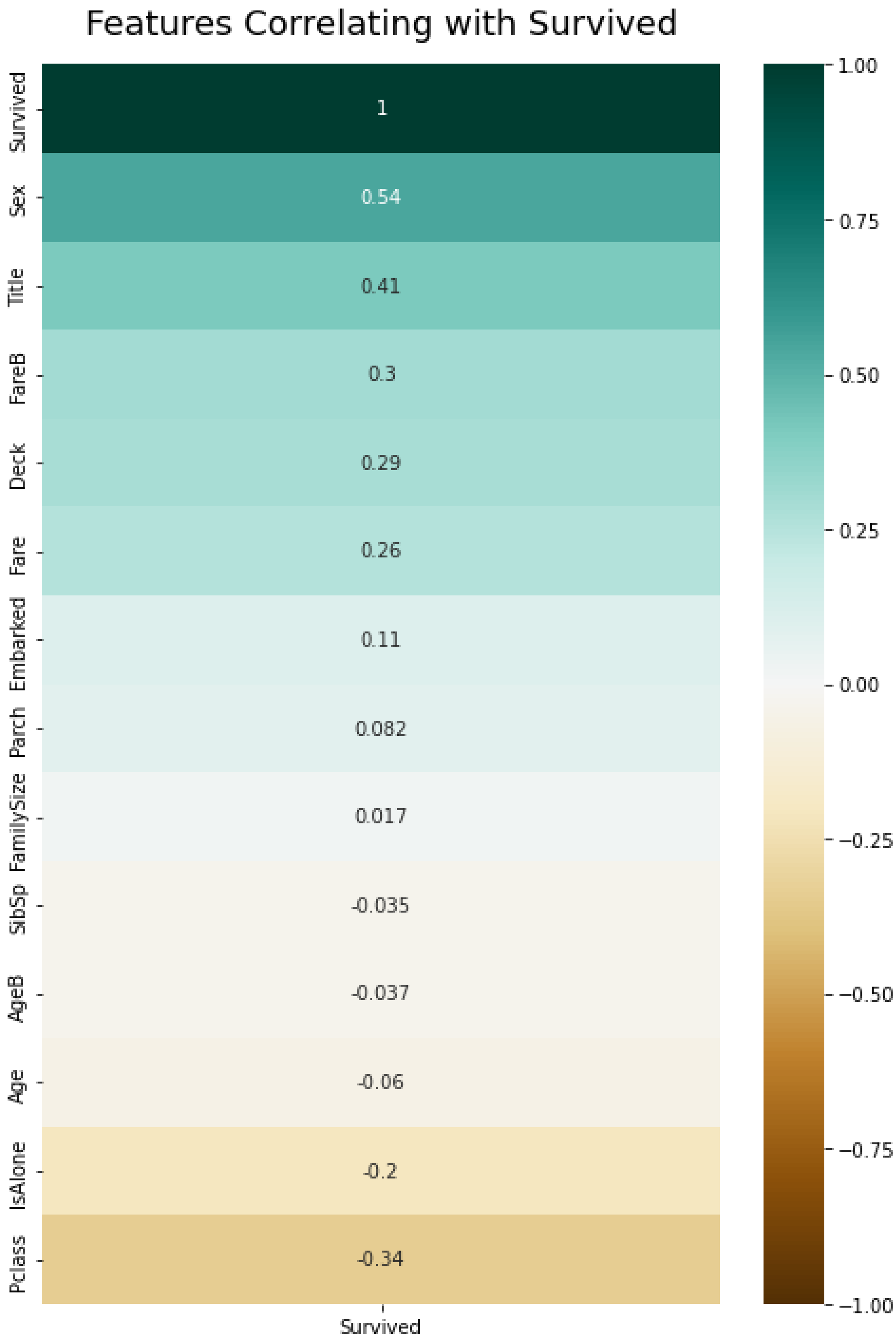
	Pclass	Lower	Middle	Upper	All
Sex	**Survived**				
female	**0**	72	6	3	81
	1	72	70	91	233
male	**0**	300	91	77	468
	1	47	17	45	109
All		491	184	216	891



- Female from Upper class is about 95–96% survived. Only 3 out of 94 Women from Upper class died.
- Female Upper class has high priority to survive
- Lower class female has more survived rate than Upper class male.

Features Correlation with Survived:

```
heatmap =
sns.heatmap(train_df.corr()[['Survived']].sort_values(by='Survived',
ascending=False), vmin=-1, vmax=1, annot=True, cmap='BrBG')
heatmap.set_title('Features Correlating with Survived',
fontdict={'fontsize':18}, pad=16);
```



- Sex is positively corrlated with Survived (with a Person's correlation coefficient of 0.54) ; Female is more likely to survive

- Pclass is negatively correlated with Survived(with a Pearson's correlation coefficient of -0.34) ; Obviously, better the ticket class (1 = 1st/Upper ; 2 = 2nd/Middle; 3 = 3rd/Lower), higher the chance of survival.

- Those important feature for prediction the Survived people

4. Acquire training and testing data

```
X_train = train_df.drop("Survived", axis=1)
Y_train = train_df["Survived"]
X_test  = test_df.drop("PassengerId", axis=1).copy()
X_train.shape, Y_train.shape, X_test.shape
```

5. Model, predict and solve the problem

Training our model with a given dataset using Supervised Learning (Classification and Regression)

```
# Logistic Regression
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# Support Vector Machines
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_svcknn = 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# Gaussian Naive Bayes
gaussian = GaussianNB()
gaussian.fit(X_train, Y_train)
Y_pred = gaussian.predict(X_test)
acc_gaussian = round(gaussian.score(X_train, Y_train) * 100, 2)
acc_gaussian# Perceptron
perceptron = Perceptron()
perceptron.fit(X_train, Y_train)
Y_pred = perceptron.predict(X_test)
acc_perceptron = round(perceptron.score(X_train, Y_train) * 100, 2)
acc_perceptron# Linear SVC
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# Stochastic Gradient Descent
sgd = SGDClassifier()
sgd.fit(X_train, Y_train)
Y_pred = sgd.predict(X_test)
acc_sgd = round(sgd.score(X_train, Y_train) * 100, 2)
acc_sgd# Decision Tree
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# Random Forest
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
```

Model selection

```
models = pd.DataFrame({
    'Model': ['Support Vector Machines', 'KNN', 'Logistic Regression',
              'Random Forest', 'Naive Bayes', 'Perceptron',
              'Stochastic Gradient Decent', 'Linear SVC',
              'Decision Tree'],
    'Score': [acc_svc, acc_knn, acc_log,
              acc_random_forest, acc_gaussian, acc_perceptron,
              acc_sgd, acc_linear_svc, acc_decision_tree]})
models.sort_values(by='Score', ascending=False)
```

	Model	Score
3	Random Forest	96.97
8	Decision Tree	96.97
1	KNN	84.74
2	Logistic Regression	81.71
4	Naive Bayes	80.25
7	Linear SVC	77.55
5	Perceptron	73.06
0	Support Vector Machines	69.70
6	Stochastic Gradient Decent	64.09

Random Forests and **Decision Trees** are the most accurate **(96.97%)** in predicting the survival of passengers and they are select to predicted the testing data.

Visualize the decision tree

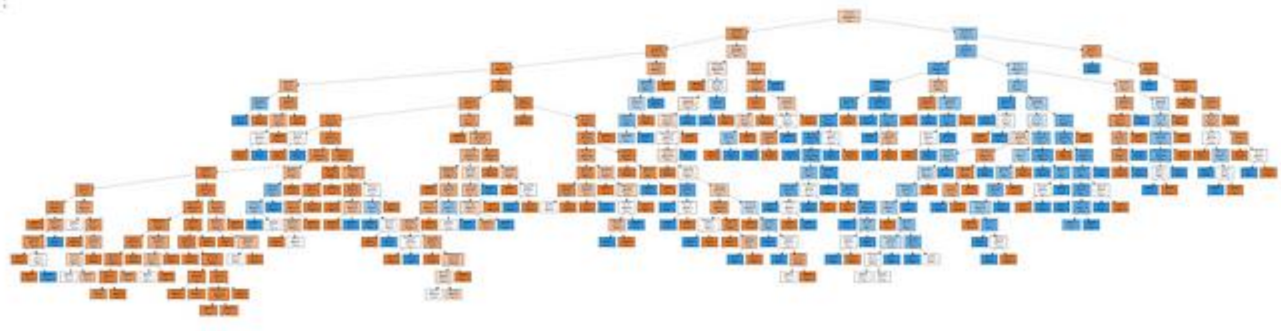
```
import graphviz

dot_data = tree.export_graphviz(clf, out_file=None,
                                feature_names=X_train.columns.values,
```

```
                                class_names='Survived',
                                filled=True)

# Draw graph
graph = graphviz.Source(dot_data, format="png")
graph
```

Out[57]:



Finding :

According to the analysis, passengers were more likely to survive if:

- Upper class ticket
- Women/Lady

On the contrary, being a Lower class old man lowered the chances of survival.

Give me a clap if you find my article is useful.👏👏👏

Reference:

I got the idea for the time series analysis below from the following.

https://medium.com/analytics-vidhya/titanic-dataset-analysis-80-accuracy-9480cf3db538

Predicting the Survival of Titanic Passengers

In this blog-post, I will go through the whole process of creating a machine learning model on the famous Titanic…

towardsdatascience.com

https://www.kaggle.com/code/startupsci/titanic-data-science-solutions

https://documentation.sas.com/doc/en/vdmmlcdc/8.1/caspg3/n0c8k27l2vkwlin196w5qptqsjqx.htm

https://www.kaggle.com/code/aaysbt/titanic-datasets-eda-fe-dc-model-predictions

https://www.kaggle.com/code/ldfreeman3/a-data-science-framework-to-achieve-99-accuracy#Step-7:-Optimize-and-Strategize

Python

Data Science

Machine Learning

Titanic

Predictions