Titanic Survival Prediction

Xiang Liu 94457
Lan Zhang 94807

Agenda

- Project Overview
- Data Pre-processing
- Data Analysis
- Conclusion
- Reference

Project Overview

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.

In this project, we will build a predictive model that answers the question: “what sorts of people were more likely to survive?” using passenger data (ie name, age, gender, socio-economic class, etc).

Data Set

Variable	Definition	Key
survival	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

There is a total of 12 columns, 892 rows.

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.

Titanic Survival Prediction

Topic

Knowing from a training set of samples listing passengers who survived or did not survive the Titanic disaster, can our model determine based on a given test dataset not containing the survival information, if these passengers in the test dataset survived or not

Data Resources

Titanic: Machine Learning from Disaster

(Kaggle)

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

Tools and Algorithm

Python

Logistic Regression

KNN

Decision Tree

RandomForest

Raw Data Sample

	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
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C

Data Information

```
Data columns (total 12 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Name           891 non-null object
Sex            891 non-null object
Age            714 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          204 non-null object
Embarked       889 non-null object
```

Process

Data Cleaning

Working with data

Clean the data

(missing data, outliers)

80% train, 20% validation

Generate a model

Machine Learning Algorithm

Explore the traits of the data and generate visual plots

Logistic Regression

Decision Tree

KNN

Random Forest

Test the model

Testing the effectiveness and results of the models

Data Cleaning

Check missing value

Before

```
Data columns (total 12 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Name           891 non-null object
Sex            891 non-null object
Age            714 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          204 non-null object
Embarked       889 non-null object
```

Drop missing value (Choose ‘Age’ to drop)

After

```
Data columns (total 12 columns):
PassengerId    714 non-null int64
Survived       714 non-null int64
Pclass         714 non-null int64
Name           714 non-null object
Sex            714 non-null object
Age            714 non-null float64
SibSp          714 non-null int64
Parch          714 non-null int64
Ticket         714 non-null object
Fare           714 non-null float64
Cabin          185 non-null object
Embarked       712 non-null object
```

Data Cleaning

Fill out missing values for
'Cabin' and 'Embarked'

```
train_df['Cabin'] = train_df['Cabin'] .replace(np.nan, 'X')
train_df['Embarked'] = train_df['Embarked'] .replace(np.nan, 'S')
train_df.info()
```

After

```
Data columns (total 12 columns):
PassengerId    714 non-null int64
Survived       714 non-null int64
Pclass         714 non-null int64
Name           714 non-null object
Sex            714 non-null object
Age            714 non-null float64
SibSp          714 non-null int64
Parch          714 non-null int64
Ticket         714 non-null object
Fare           714 non-null float64
Cabin          714 non-null object
Embarked       714 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 72.5+ KB
```

Check Outliers

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	714.000000	714.000000	714.000000	714.000000	714.000000	714.000000	714.000000
mean	448.582633	0.406162	2.236695	29.699118	0.512605	0.431373	34.694514
std	259.119524	0.491460	0.838250	14.526497	0.929783	0.853289	52.918930
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	222.250000	0.000000	1.000000	20.125000	0.000000	0.000000	8.050000
50%	445.000000	0.000000	2.000000	28.000000	0.000000	0.000000	15.741700
75%	677.750000	1.000000	3.000000	38.000000	1.000000	1.000000	33.375000
max	891.000000	1.000000	3.000000	80.000000	5.000000	6.000000	512.329200

Detect_Outliners Function in Python

```python
from collections import Counter
def detect_outliers(df,n,features):
    """
    Takes a dataframe df of features and returns a list of the indices
    corresponding to the observations containing more than n outliers according
    to the Tukey method.
    """
    outlier_indices = []

    # iterate over features(columns)
    for col in features:
        # 1st quartile (25%)
        Q1 = np.percentile(df[col], 25)
        # 3rd quartile (75%)
        Q3 = np.percentile(df[col],75)
        # Interquartile range (IQR)
        IQR = Q3 - Q1

        # outlier step
        outlier_step = 1.5 * IQR

        # Determine a list of indices of outliers for feature col
        outlier_list_col = df[(df[col] < Q1 - outlier_step) | (df[col] > Q3 + outlier_step )].index

        # append the found outlier indices for col to the list of outlier indices
        outlier_indices.extend(outlier_list_col)

    # select observations containing more than 2 outliers
    outlier_indices = Counter(outlier_indices)
    multiple_outliers = list( k for k, v in outlier_indices.items() if v > n )

    return multiple_outliers

# detect outliers from Age, SibSp , Parch and Fare (train_data)
Outliers_to_drop = detect_outliers(train_df,2,["Age","SibSp","Parch","Fare"])
```

Convert Categorical variables to numerical variables

• Name	→	“Master": 0 "Miss-Mrs": 1 "Mr": 2 "Others”: 3
• Sex	→	"male": 0 "female":1
• SibSp • Parch	Fsize (Family Size) →	"Single": 0 "SmallF(2-3)" : 1 "MedF(3-4)": 2 "LargeF(5): 3
• Ticket	→	"A": 1 "C": 2 "F": 3 "L": 4 'P': 5 "S":6 "W":7 "X":8
• Cabin	→	"A": 1 "B": 2 "C": 3 "D": 4 'E': 5 "F":6 "G":7 "T":8 "X": 9
• Embarked	→	'S': 0 'C': 1 'Q': 2

Normalize the data

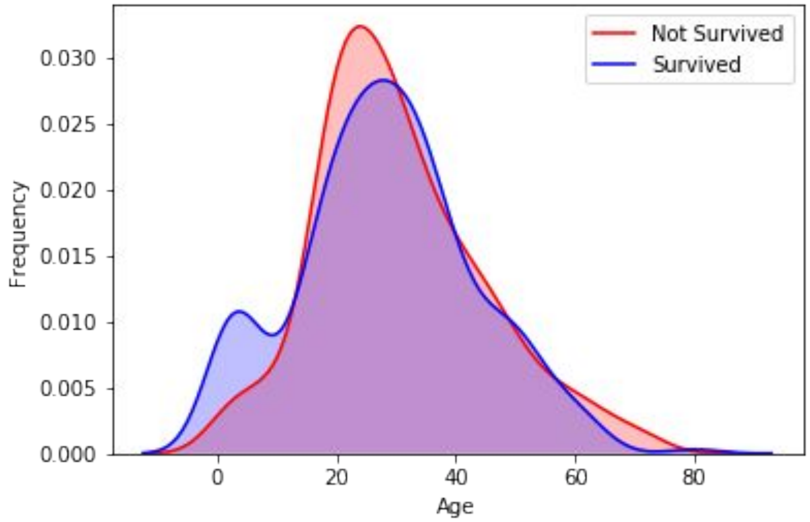
	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Title	Fsize
0	1	0	3	0	22.0	1	0	1.0	1.981001	9.0	0	2.0	1
1	2	1	1	1	38.0	1	0	5.0	4.266662	3.0	1	1.0	1
2	3	1	3	1	26.0	0	0	6.0	2.070022	9.0	0	1.0	0
3	4	1	1	1	35.0	1	0	8.0	3.972177	3.0	0	1.0	1
4	5	0	3	0	35.0	0	0	8.0	2.085672	9.0	0	2.0	0

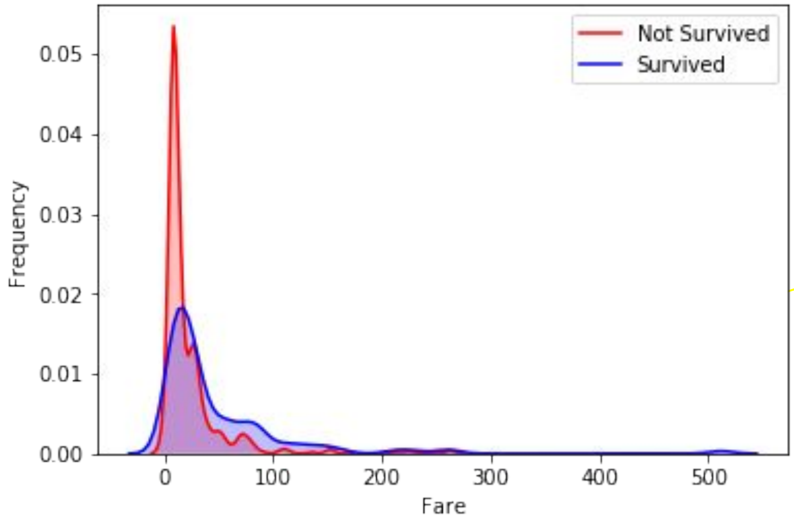
Before

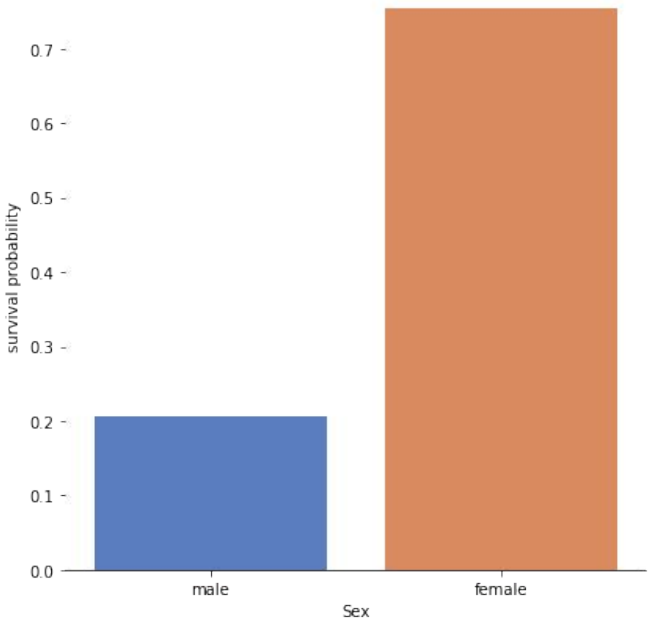
	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Title	Fsize
0	0.000000	0.0	1.0	0.0	0.271174	0.2	0.000000	0.000000	0.317521	1.00	0.0	0.666667	0.333333
1	0.001124	1.0	0.0	1.0	0.472229	0.2	0.000000	0.571429	0.683873	0.25	0.5	0.333333	0.333333
2	0.002247	1.0	1.0	1.0	0.321438	0.0	0.000000	0.714286	0.331789	1.00	0.0	0.333333	0.000000
3	0.003371	1.0	0.0	1.0	0.434531	0.2	0.000000	1.000000	0.636672	0.25	0.0	0.333333	0.333333
4	0.004494	0.0	1.0	0.0	0.434531	0.0	0.000000	1.000000	0.334298	1.00	0.0	0.666667	0.000000

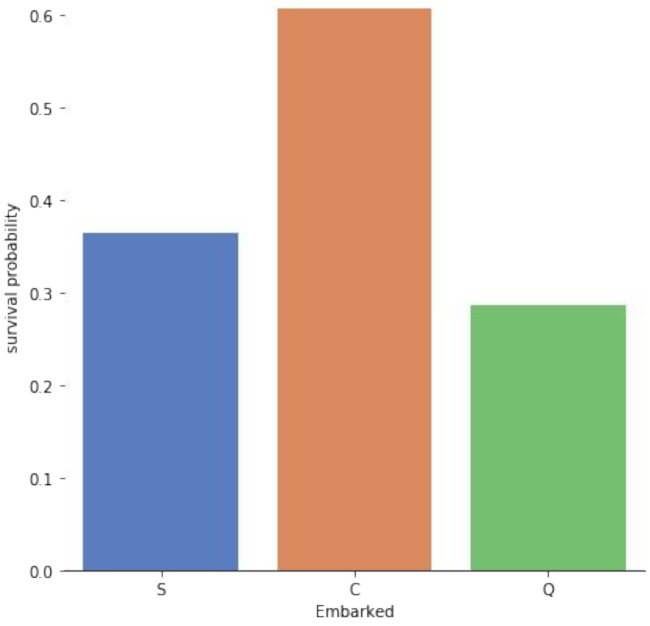
After

Data Visualization





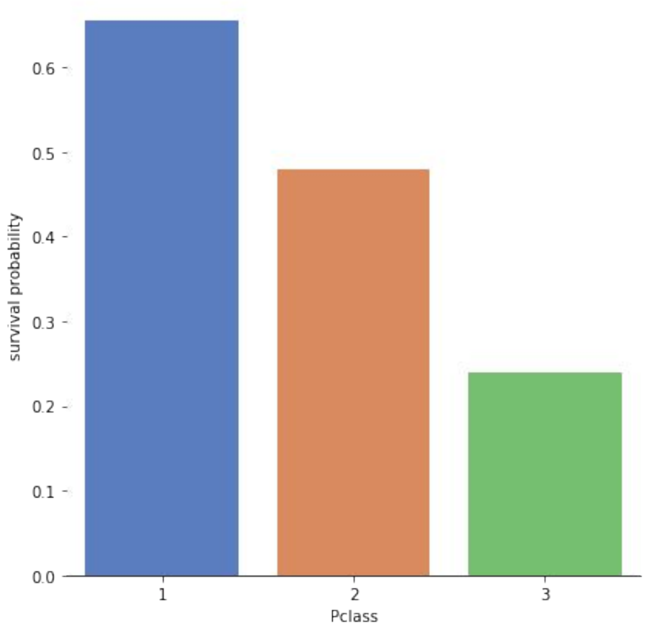


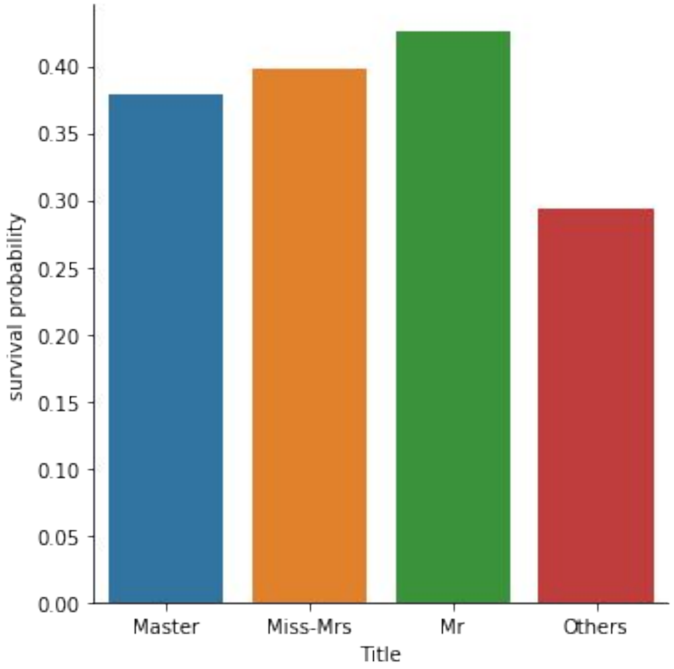


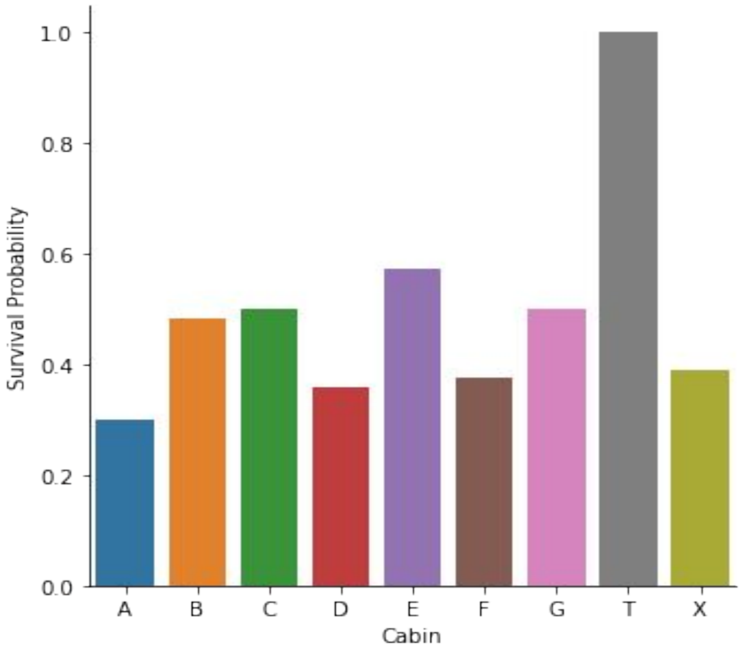
S: Southhampton

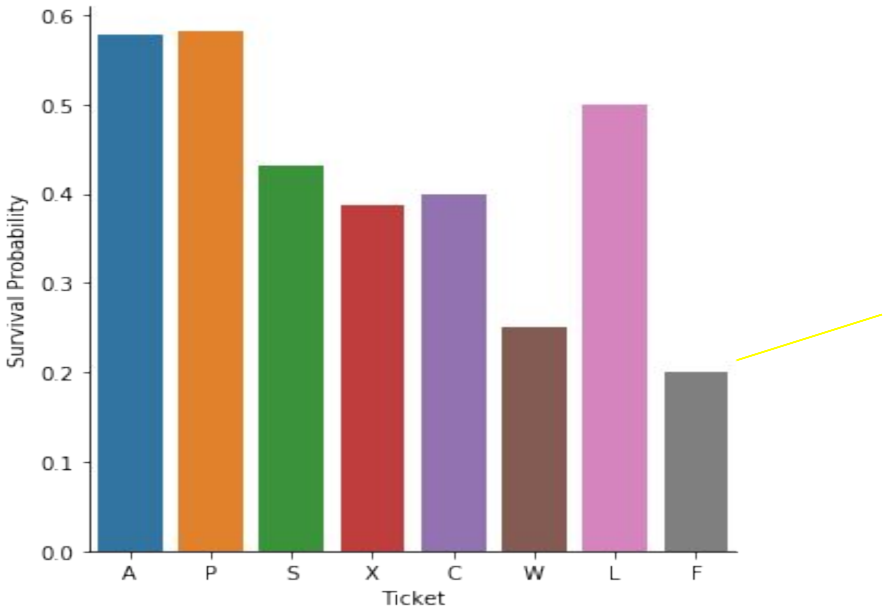
C: Cherbourg

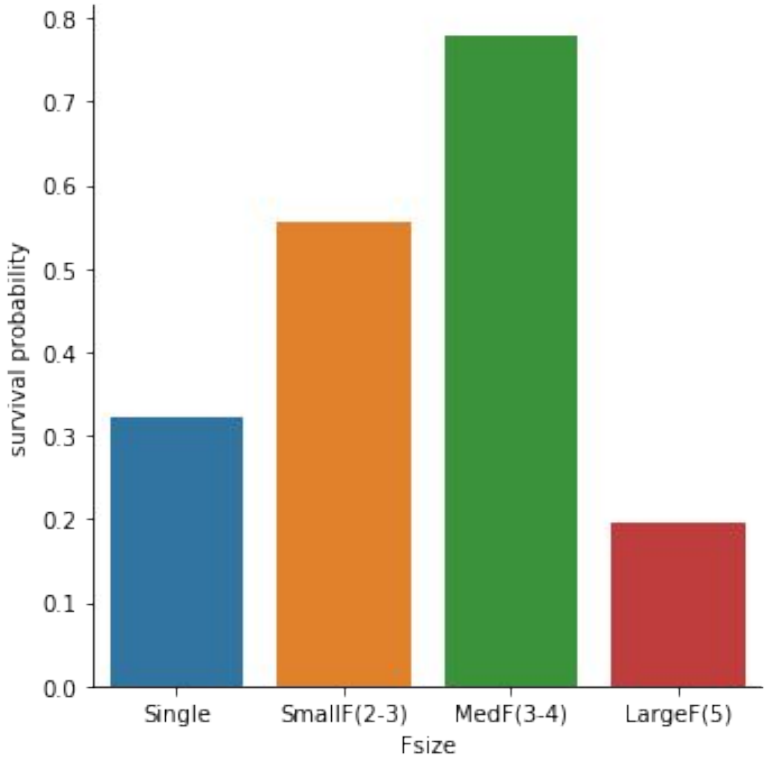
Q: Queenstown


0.6
0.5
0.4
0.3
0.2
0.1
0.0
survival probability
1
2
3
Pclass

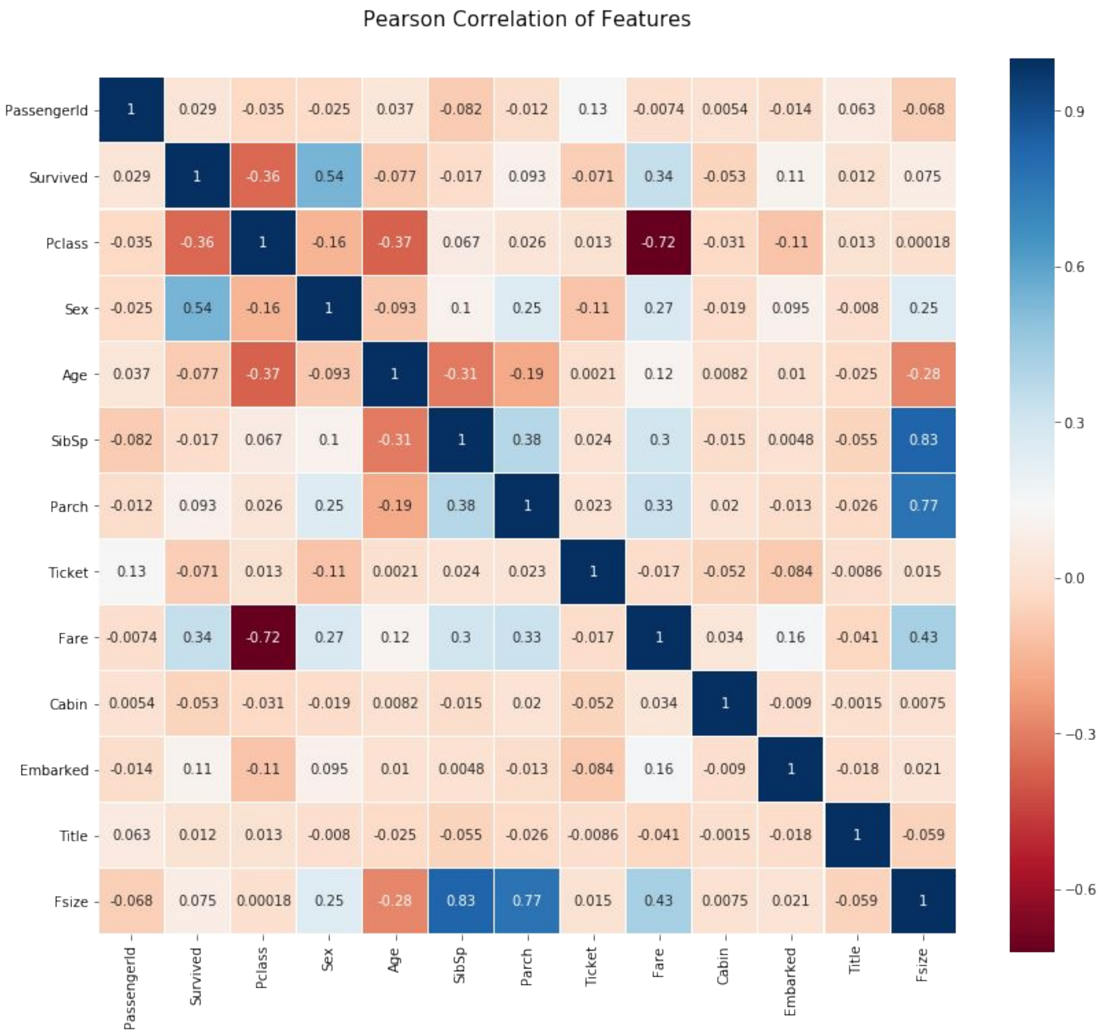

0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
survival probability
Master
Miss-Mrs
Mr
Others
Title


1.0
0.8
0.6
0.4
0.2
0.0
Survival Probability
A B C D E F G T X
Cabin


0.6
0.5
0.4
0.3
0.2
0.1
0.0
Survival Probability
A P S X C W L F
Ticket



0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
survival probability
Single
SmallF(2-3)
MedF(3-4)
LargeF(5)
Fsize



Survived

1. Sex
2. Pclass
3. Fare

Modeling

Logistic Regression

Accuracy = 0.8462

RMSE = 0.392

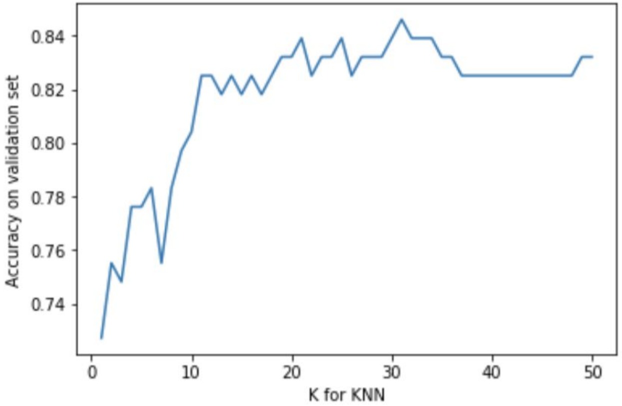
AUC : 0.89

```
              precision    recall  f1-score   support

         0.0       0.84      0.92      0.88        85
         1.0       0.86      0.74      0.80        58

   micro avg       0.85      0.85      0.85       143
   macro avg       0.85      0.83      0.84       143
weighted avg       0.85      0.85      0.84       143
```

```
array([[78,  7],
       [15, 43]])
```

Modeling

KNN (k=31)
Accuracy = 0.8461
RMSE = 0.4096
AUC: 0.883



	precision	recall	f1-score	support
0.0	0.81	0.93	0.87	85
1.0	0.87	0.69	0.77	58
micro avg	0.83	0.83	0.83	143
macro avg	0.84	0.81	0.82	143
weighted avg	0.84	0.83	0.83	143

```
array([[79,  6],
       [18, 40]])
```

Modeling

Decision Tree

Accuracy = 0.8461
RMSE = 0.418
AUC: 0.83

	variable	importance
1	Sex	0.336718
2	Age	0.208421
0	Pclass	0.158157
6	Fare	0.137584
10	Fsize	0.068196
5	Ticket	0.032442
7	Cabin	0.027185
8	Embarked	0.019962
9	Title	0.007619
4	Parch	0.003717

```
              precision    recall  f1-score   support

         0.0       0.81      0.91      0.86        85
         1.0       0.83      0.69      0.75        58

   micro avg       0.82      0.82      0.82       143
   macro avg       0.82      0.80      0.81       143
weighted avg       0.82      0.82      0.81       143
```

```
array([[77,  8],
       [18, 40]])
```

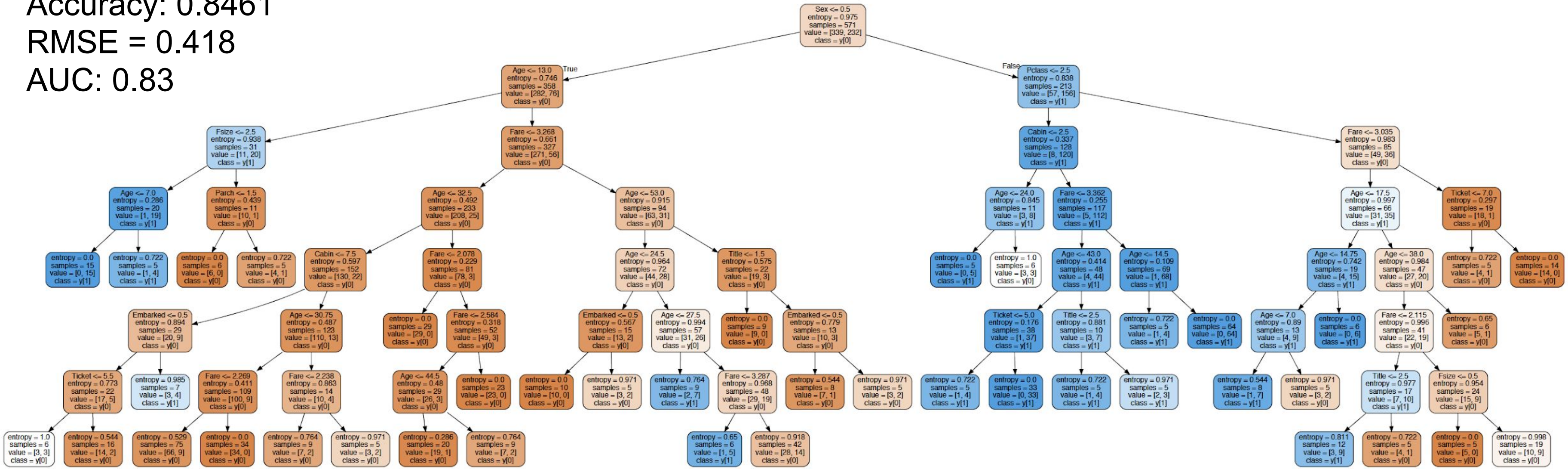
Decision Tree

Max_depth = 7

Accuracy: 0.8461

RMSE = 0.418

AUC: 0.83



Modeling

Random Forest

Accuracy = 0.7692
RMSE = 0.4803
AUC: 0.817

	variable	importance
1	Sex	0.227977
2	Age	0.208715
6	Fare	0.202071
0	Pclass	0.091753
7	Cabin	0.055946
5	Ticket	0.048107
9	Title	0.045048
10	Fsize	0.039272
3	SibSp	0.027468
4	Parch	0.027228

```
              precision    recall  f1-score   support

         0.0       0.78      0.86      0.82        85
         1.0       0.76      0.64      0.69        58

   micro avg       0.77      0.77      0.77       143
   macro avg       0.77      0.75      0.75       143
weighted avg       0.77      0.77      0.77       143
```

```
array([[73, 12],
       [21, 37]])
```

Conclusion

The correlation between

Sex, Pclass, Fare and Survival Rate is obvious.

Logistic regression model has better performance than the other three models in this case.

Reference

- A journey through Titanic
- Getting Started with Pandas: Kaggle's Titanic Competition
- Titanic Best Working Classifier