Predicting Survival on Titanic

Carrie Lei
(415)810-8610
leoi06012015@gmail.com

January 22, 2020

1 Abstract

The goal for this project is to build a mathematical model to predict survival of passengers on Titanic given the information of their ages, passenger class, gender, etc. Data are retrieved through online website Kaggle.

2 Analyzing data

There are two sets of data, one for training the model and the other one for testing the performance of the model.
First step is to understand what the data means.

Next, I tried to explore the data by visualizing them.

3 Cleaning the data

As we can notice from the data above, some age data are missing. I would be nice to fill in the missing data by approximating them. Instead of fill in the average age of the whole data set, I separated the data by their passenger class and find the average age of each class and fill in the average age according to the class that the data belong to.

For the cabin data, since so many data are missing, I just dropped the whole column of data. I also dropped the name column and ticket column, since they do not matter.

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

Figure 1: Description of the 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

Figure 2: First 5 columns of the data set.

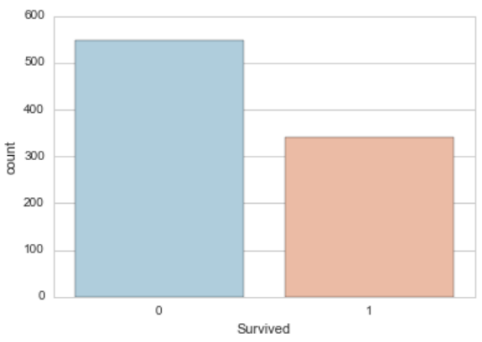

Figure 3: Count survival situation (0 represents not survived and 1 represents survived).

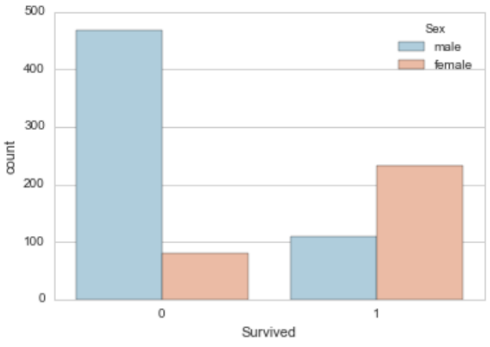

Figure 4: Count the number of different gender in each survival situation.

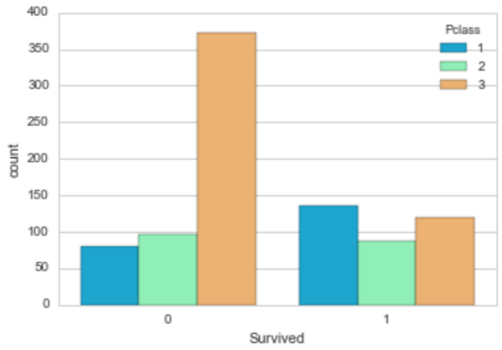

Figure 5: Count survival situation (0 represents not survived and 1 represents survived).

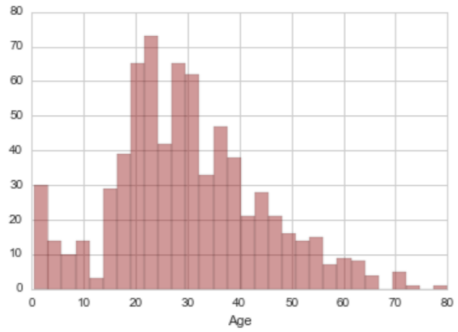

Figure 6: Count the number of different gender in each survival situation.

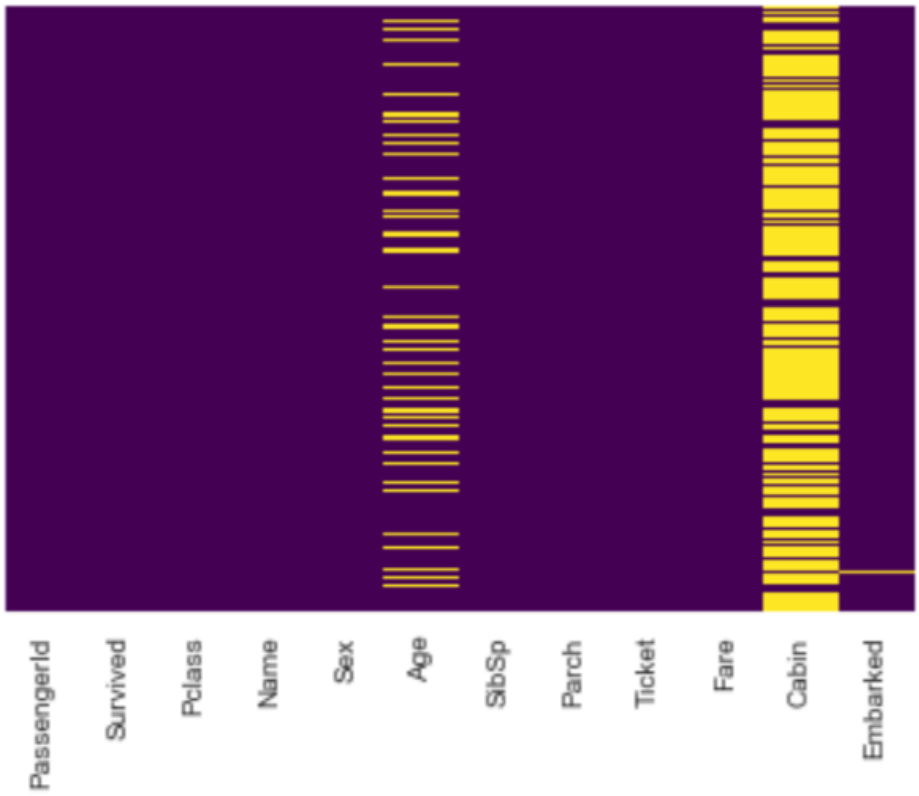

Figure 7: Heatmap to visualize what data is missing.

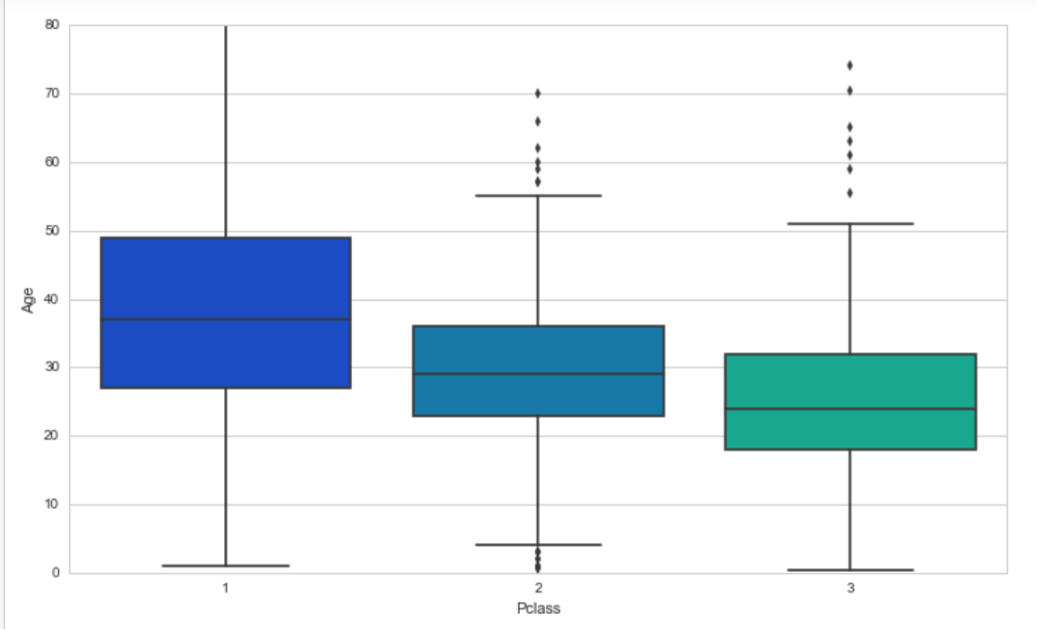

Figure 8: Box plot of age for each class.

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	male	Q	S
0	1	0	3	22.0	1	0	7.2500	1.0	0.0	1.0
1	2	1	1	38.0	1	0	71.2833	0.0	0.0	0.0
2	3	1	3	26.0	0	0	7.9250	0.0	0.0	1.0
3	4	1	1	35.0	1	0	53.1000	0.0	0.0	1.0
4	5	0	3	35.0	0	0	8.0500	1.0	0.0	1.0

Figure 9: Converted data.

As we can see, some data are categorical data and it would be nice to convert them into numerical data. For gender data, I use 1 or 0 to represent whether the passenger is male or female. For Embark column, I used the same method to keep track of whether the passenger embarked at Q, S or neither (which means C).

4 Train the Model

Since I know the data that represent whether the passenger survived or not is a binary data, so I used the Logistic Regression model to train the data. Python code:

	precision	recall	f1-score	support
0	0.81	0.93	0.86	163
1	0.85	0.65	0.74	104
avg / total	0.82	0.82	0.81	267

Figure 10: Model performance evaluation.

```
from sklearn.linear_model import LogisticRegression
  logmodel = LogisticRegression()
  logmodel.fit(X_train,y_train)
```

5 Evaluate the Model

After I got the trained model, I used the test data to evaluate the performance of the model by compare the actual value from the test data and the predict values from the model. Python Code: from sklearn.metrics import classification_report

Denote TP as number of True Positive, TN as True Negative, FP as False Positive, FN as False Negative. Where $precision = \frac{TP}{TP+FN}$, $Recall = \frac{TP}{TP+FN}$, $F1_score = 2 * (Recall * Precision)/(Recall + Precision)$, support is the number of occurrences of each class in the test data.

6 Conclusion

The success rate to predict the survival of a passenger by this model is 82%.

7 Reference Python Code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
# import tensorflow as tf


train = pd.read_csv('train.csv')

# plt.subplot(2, 2, 1)
# sns.heatmap(train.isnull(), yticklabels=False, cbar=
    False, cmap='viridis')
#
# plt.subplot(2, 2, 2)
# sns.set_style('whitegrid')
# sns.countplot(x='Survived',data=train,palette='RdBu_r
    ')
#
# plt.subplot(2, 2, 3)
# sns.countplot(x='Survived',hue='Sex',data=train,
    palette='RdBu_r')
#
# plt.subplot(2,2,4)
# sns.countplot(x='Survived',hue='Pclass',data=train,
    palette='rainbow')

# sns.distplot(train['Age'].dropna(),kde=False,color='
    darkred',bins=30)
# plt.show()

# plt.figure(figsize=(10, 5))
# sns.boxplot(x='Pclass',y='Age',data=train,palette='
    summer')
# plt.show()

# calculate the average age for each passenger class
Class = train['Pclass']
# print(Class)
```

```
train_test = train.dropna(subset=['Age'])
# print(train_test.head(5))

class1_mean = train_test[train_test['Pclass'] == 1]['
    Age'].mean()
# print(class1_mean)
class2_mean = train_test[train_test['Pclass'] == 2]['
    Age'].mean()
# print(class2_mean)
class3_mean = train_test[train_test['Pclass'] == 3]['
    Age'].mean()
# print(class3_mean)

def replace_null(cols):
    Pclass = cols[0]
    Ages = cols[1]
    if pd.isnull(Ages):
        if Pclass == 1:
            return class1_mean
        if Pclass == 2:
            return class2_mean
        if Pclass == 3:
            return class3_mean

    else:
        return Ages


train['Age'] = train[['Pclass', 'Age']].apply(
    replace_null, axis=1)


sns.heatmap(train.isnull(),yticklabels=False,cbar=False
    ,cmap='viridis')


train.drop('Cabin', axis=1, inplace=True)
sex = pd.get_dummies(train['Sex'], drop_first=True)
embark = pd.get_dummies(train['Embarked'], drop_first=
    True)
train.drop(['Sex', 'Embarked', 'Name', 'Ticket'], axis
```

```
    =1, inplace=True)
train = pd.concat([train, sex, embark], axis=1)
print(train.head(3))
# plt.show()
```