University of Pittsburgh
School of Computing and Information

Final Report

INFSCI 2725: Data Analytics

December 4, 2018

Yunshu Liang (yul219)
Qi Lu (qil66)
Erin Price (eep27)
Ziyue Qi (ziq2)
Xiaoqian Xu (xix64)

Table of Contents

Introduction & Problem Definition

For the Data Analytics term project, our team named INFSCI 2725 - Fall 2018 includes Yunshu Liang, Qi Lu, Erin Price, Ziyue Qi, and Xiaoqian Xu. The Kaggle project the team will be working on is *Titanic: Machine Learning from Disaster*, located at https://www.kaggle.com/c/titanic.

The goal of the project is to predict who did and did not survive on the Titanic, based on data analytics techniques. In this project, we complete the analysis of dataset to predict what sorts of people are likely to survive. In particular, we use Python and implement machine learning algorithms to make a prediction on this binary classification problem.

We solve this problem taking the following steps:

- Define the problem
- Prepare the data & Preview the data
- Feature analysis
- Data cleaning
- Build the model
- Evaluation and Cross-Validation

Data Preparation & Preview

0. Import Libraries

import time

import seaborn as sns

import pandas as pd

import matplotlib

import numpy as np

import scipy as sp

import IPython

import sklearn

import matplotlib.pyplot as plt

import itertools

import graphviz

from sklearn.model_selection import RandomizedSearchCV, cross_val_score

from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier

from sklearn import preprocessing, model_selection, ensemble, linear_model, naive_bayes, tree

from math import isnan

from sklearn import datasets

from sklearn.preprocessing import LabelEncoder

1. Data Preview

Training and test data are stored in files train.csv and test.csv

1.1. Load train data and test data into pandas data structure—DataFrame and create a dictionary which combined dataTrain and dataTest for the purpose of processing in the same way.

Import pandas as pd

csvTrain = "train.csv"
csvTest = "test.csv"

dataTrain = pd.read_csv(csvTrain)
dataTest = pd.read_csv(csvTest)

dataCleaner = [dataTrain,dataTest]

1.2. Preview the data by seeing the first 5 record.

print (dataTrain.head())

print (dataTest.head())

```
/Users/shine/anaconda/bin/python /Users/shine/Documents/Python/analyze/analyze.py
    First 5 records of training data:
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex  Age  SibSp  \
0                            Braund, Mr. Owen Harris    male   22      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female   38      1
2                             Heikkinen, Miss. Laina  female   26      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female   35      1
4                           Allen, Mr. William Henry    male   35      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
    First 5 records of test data::
   PassengerId  Pclass                                          Name     Sex  \
0          892       3                              Kelly, Mr. James    male
1          893       3              Wilkes, Mrs. James (Ellen Needs)  female
2          894       2                     Myles, Mr. Thomas Francis    male
3          895       3                              Wirz, Mr. Albert    male
4          896       3  Hirvonen, Mrs. Alexander (Helga E Lindqvist)  female

    Age  SibSp  Parch   Ticket     Fare Cabin Embarked
0  34.5      0      0   330911   7.8292   NaN        Q
1  47.0      1      0   363272   7.0000   NaN        S
2  62.0      0      0   240276   9.6875   NaN        Q
3  27.0      0      0   315154   8.6625   NaN        S
4  22.0      1      1  3101298  12.2875   NaN        S
```

Figure 1 Fist 5 records

2. Overall Description

2.1. Attributes

dataTrain.columns.values

```
/Users/shine/anaconda/bin/python /Users/shine/Documents/Python/analyze/analyze.py
    Columns:
['PassengerId' 'Survived' 'Pclass' 'Name' 'Sex' 'Age' 'SibSp' 'Parch'
 'Ticket' 'Fare' 'Cabin' 'Embarked']
```

Figure 2 Columns

dataTrain.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
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
dtypes: float64(2), int64(5), object(5)
memory usage: 83.6+ KB
```

Figure 3 Summary of data

From Titanic competition instruction, we know what each column/attribute represents in real world:

PassengerID(int): index of passengers, which might be helpless for our prediction

Survived(int): 0 and 1 representing not survived and survived;

Pclass(int): 1 and 2 and 3. 1 represents highest class;

Name(string): title and name;

Sex(string): female and male;

Age(float): age in years, range from 0.42 to 80;

SibSp(int): of siblings / spouses aboard the Titanic;

Parch(int): of parents / children aboard the Titanic;

Ticket(string): ticket number;

Fare(float): passenger fare;

Cabin(string): cabin number;

Embarked(string): C = Cherbourg, Q = Queenstown, S = Southampton.

2.2. Descriptive statistics of data

dataTrain.describe()

```
    Descriptive statistics of training data(numeric):
       PassengerId    Survived      Pclass         Age       SibSp  \
count   891.000000  891.000000  891.000000  714.000000  891.000000
mean    446.000000    0.383838    2.308642   29.699118    0.523008
std     257.353842    0.486592    0.836071   14.526497    1.102743
min       1.000000    0.000000    1.000000    0.420000    0.000000
25%     223.500000    0.000000    2.000000   20.125000    0.000000
50%     446.000000    0.000000    3.000000   28.000000    0.000000
75%     668.500000    1.000000    3.000000   38.000000    1.000000
max     891.000000    1.000000    3.000000   80.000000    8.000000

            Parch        Fare
count  891.000000  891.000000
mean     0.381594   32.204208
std      0.806057   49.693429
min      0.000000    0.000000
25%      0.000000    7.910400
50%      0.000000   14.454200
75%      0.000000   31.000000
max      6.000000  512.329200
    For strings:
                           Name   Sex    Ticket Cabin Embarked
count                       891   891       891   204      889
unique                      891     2       681   147        3
top     Gill, Mr. John William  male  CA. 2343    G6        S
freq                          1   577         7     4      644
```

Figure 4 Descriptive statistics

From statistic summary of the training data, we know number of records, mean, max/min and distribution of each attribute, which influences how we build the model.

2.3. Missing Values

dataTrain.isnull().sum()

dataTest.isnull().sum()

```
Train columns 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
Test/Validation columns 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
```

Figure 5 Missing Values

Summary:

Age: 117/891 missing in training data, 86/418 missing in test data;

Cabin: 687/891 missing in training data, 327/418 missing in test data;

Fare: 1/418 missing in test data;

Embarked: 2/891 missing in training data.

For further analysis, we'll need a complete dataset without any null values. So missing data should be filled with proper values.

Feature Analysis

1. Sex

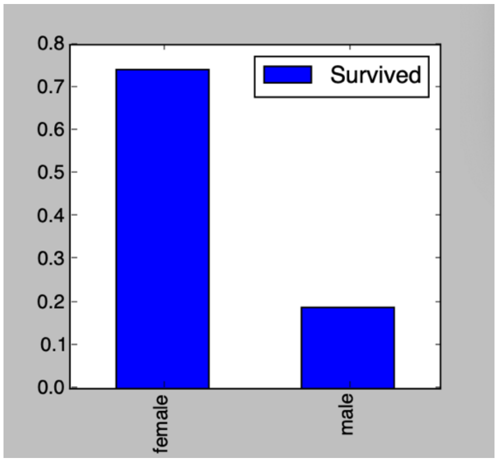

Figure 6 Sex - Survival Rate

Apparently, female has a significantly higher survival rate of roughly 75% than that of male which is below 20%.

2. Pclass

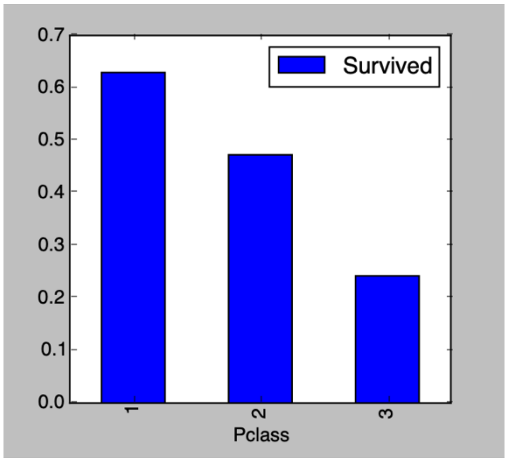

Figure 7 Pclass - Survival Rate

It is clear that first class passengers have the highest survival rate, followed by the second and third class.

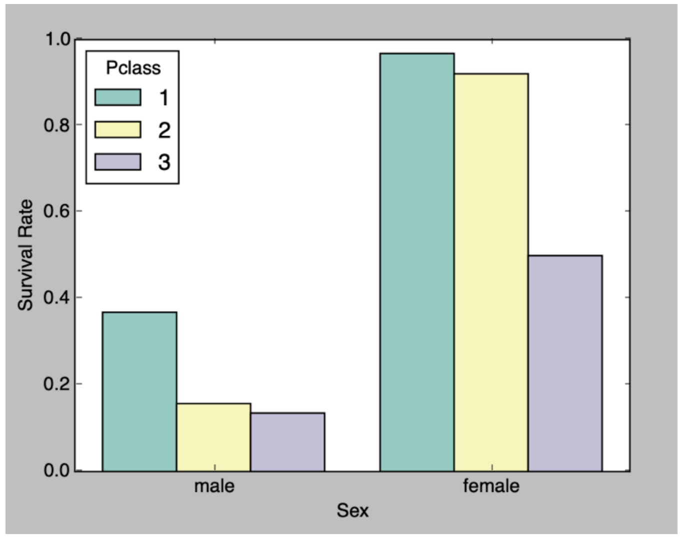

Figure 8 Sex & Pclass - Survival Rate

Figure 8 supports the two conclusions we draw above. Female are more likely to survive than male of the same class. And first class has the highest survival rate.

3. Fare

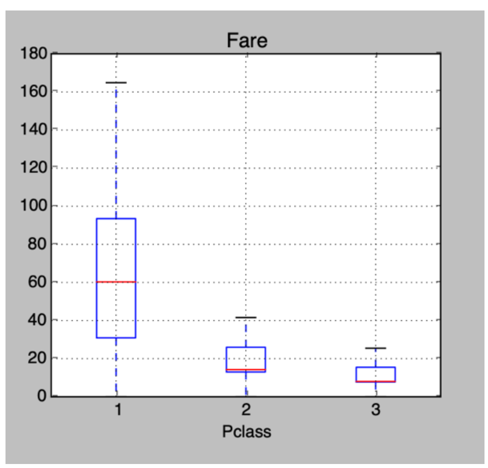

Figure 9 Pclass – Fare

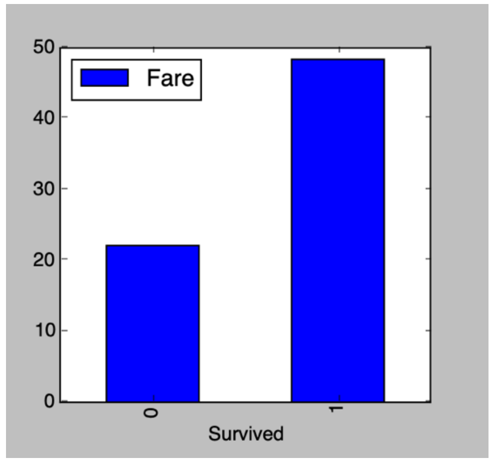

Figure 10 Fare – Survived

We can assume that higher ticket fare represents higher class, leading to the greater chance to survive. Figure 9 and 10 confirms that. The average fare of survived passengers is higher than those who didn't.

4. Age

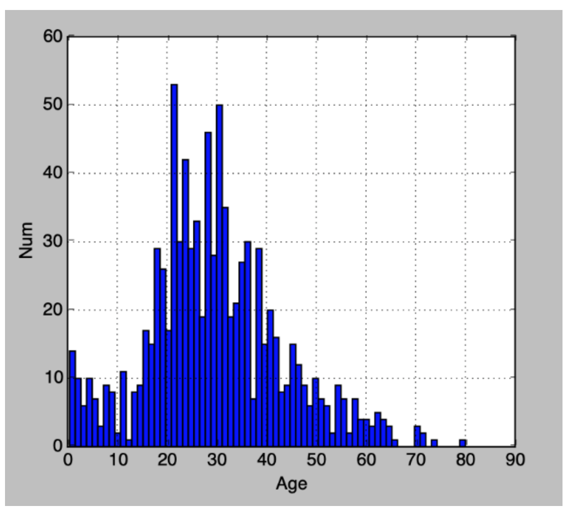

Figure 31 Age – Survived

From figure 11, it's safe to conclude that young passengers (15-40 years old) get the biggest chance to survive. Also, there's a peak corresponding to infants and young children.

5. Embarked

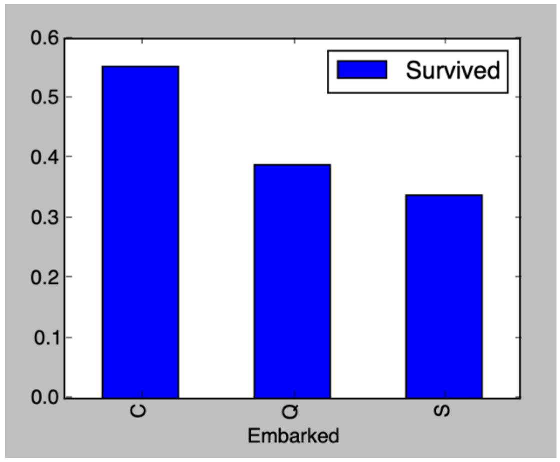

Figure 4 Embarked - Survival Rate

For unclear reasons, departure from port C gives passengers a stronger possibility to survive compared to leaving from the other two ports.

Data Cleaning

1. Completing missing data

1.1 Age

Though we can find some age categories with relatively high survival rate, no clear relation pattern is detected between how exactly age effects the survival rate. Also, the number of missing values are very large. Therefore, we fill the missing Age data in both training and test dataset with predicted values, which are generated by RandomForest algorithm.

```
# Fill missing Age data with predicted values
ageSet = dataTrain[["Age", "Survived","Fare", "Parch", "SibSp", "Pclass"]]

ageSetNull = ageSet.loc[(dataTrain["Age"].isnull())]

ageSetNotNull = ageSet.loc[(dataTrain["Age"].notnull())]

X = ageSetNotNull.values[:, 1:]

Y = ageSetNotNull.values[:, 0]

RFR = RandomForestRegressor(n_estimators=1000, n_jobs=-1)

RFR.fit(X, Y)

predictAges = RFR.predict(ageSetNull.values[:, 1:])

dataTrain.loc[dataTrain["Age"].isnull(), ["Age"]] = predictAges

dataTrain.info()
```

1.2 Embarked

Only two records have missing Embarked value, so we replace them with mode.

 # Fill missing Embarked data with mode

dataTrain.Embarked[dataTrain.Embarked.isnull()] = dataTrain.Embarked.dropna().mode().values

1.3 Cabin

There are 687/891 missing values in training data and 327/418 missing values in test data. The amount of missing values is too large to use either mode or medium to fill with. Therefore, we decide to replace missing Cabin values with “NA”.

 # Fill missing Cabin data with "NA"

dataCleaner["Cabin"] = dataCleaner.Cabin.fillna("NA")

1.4 Fare

There’s only one null value in this column, we fill it in with medium number since this’s a numeric attribute.

 # Fill missing Fare data with "median"

dataCleaner['Fare'].fillna(dataCleaner['Fare'].median(), inplace=True)

2. Coverting data formats: categorizing continuous variables

2.1 Age

Try three different grouping method and decide on the last one, which is, age 0-15 = group 0, age 15-80 =group 1. This attribute is named AgeBin.

At first, we think we should divide the people based on their escape ability.

```
# first try
# agebin=[0,6,16,35,55,80]
# dataCleaner['AgeBin']=pd.cut(dataCleaner['Age'],bin,labels=[0,1,2,3,4])
# second try
# agebin = [0, 14, 30, 50, 80]
# dataCleaner['AgeBin'] = pd.cut(dataCleaner['Age'], agebin, labels=[0, 1, 2, 3])
```

Then, we found that only kids and teenagers have higher survival rate so that we change another way to divide age bin.

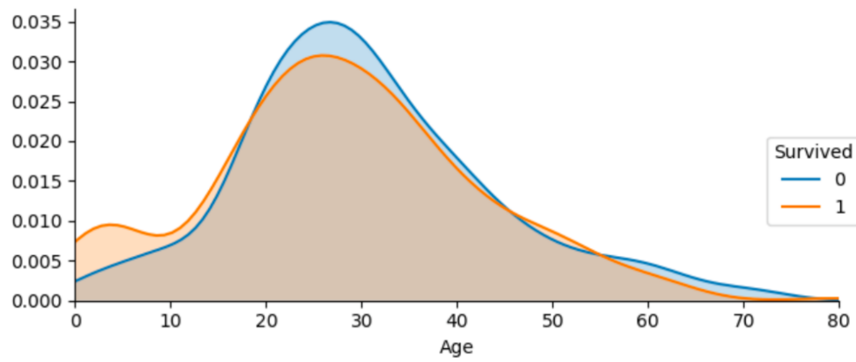

Figure 5 Age - Survival Rate

```
# third try
agebin = [0, 15, 80]
dataCleaner['AgeBin'] = pd.cut(dataCleaner['Age'], agebin, labels=[0, 1])
```

2.2 Fare

Fare is a continuous variable. To turn it into a discrete one, we divide Fare into 3 groups according to the statistical summary showed in figure 3. Values at 25%, 50% and 75% are set as boundaries (Fare <= 7.91, 7.91 < Fare <= 14.454, 14.454 < Fare <= 31, Fare > 31).

0,1,2 represents the three categories mentioned above.

3. Creating columns

3.1 Name

Even though we can't extract much information from the name itself, the title contained in this feature may influence the result. There are 18 titles and we categorize them as 5:

["Capt", "Col", "Major", "Dr", "Rev"] = "Officer",

["Don", "Sir", "the Countess", "Dona", "Lady"] = "Royalty",

["Mme", "Ms", "Mrs"] = "Mrs",

["Mlle", "Miss"] = "Miss",

["Mr"] = "Mr",

["Master", "Jonkheer"] = "Master".

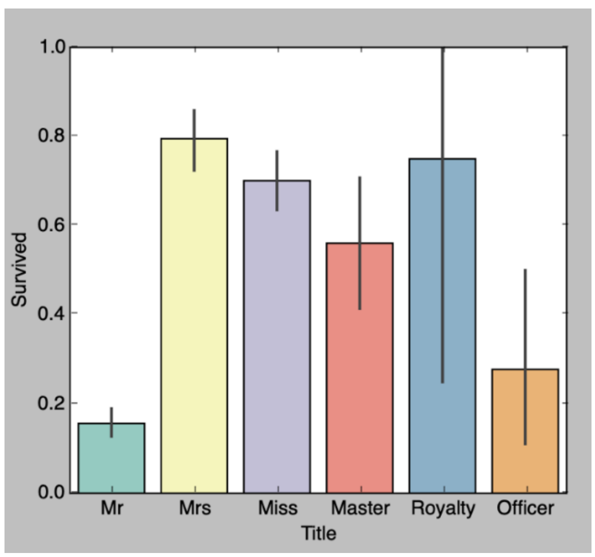

Figure 64 Title - Survival Rate

We can see from figure 14 that females (Mrs./Miss) are still the group that has the highest survival rate. Furthermore, passengers with titles which may represents higher social class (Master/Royalty/Officer) are more likely to survive than average male passengers(Mr.).

Due to the fact that Royalty and Officer are rarely used, we combine these two into one category, "Rare".

1, 2, 3, 4, 5 represents the five categories mentioned above.

3.2 FamilySize

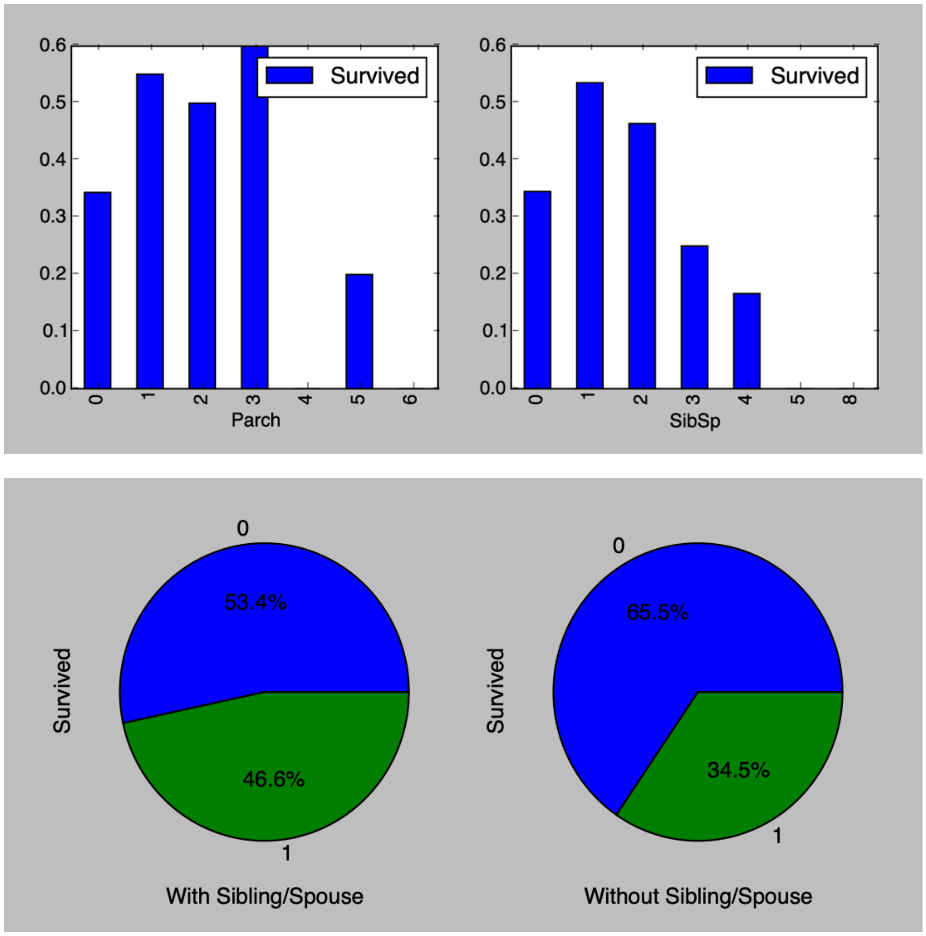

Figure 7 Parch/SibSp - Survival Rate

We can't recognize from figure 15 if Parch or SibSp has impact on survival rate. Considering the fact that these two features both represents the number of family members, we create a new attribute, FamilySize by adding them up.

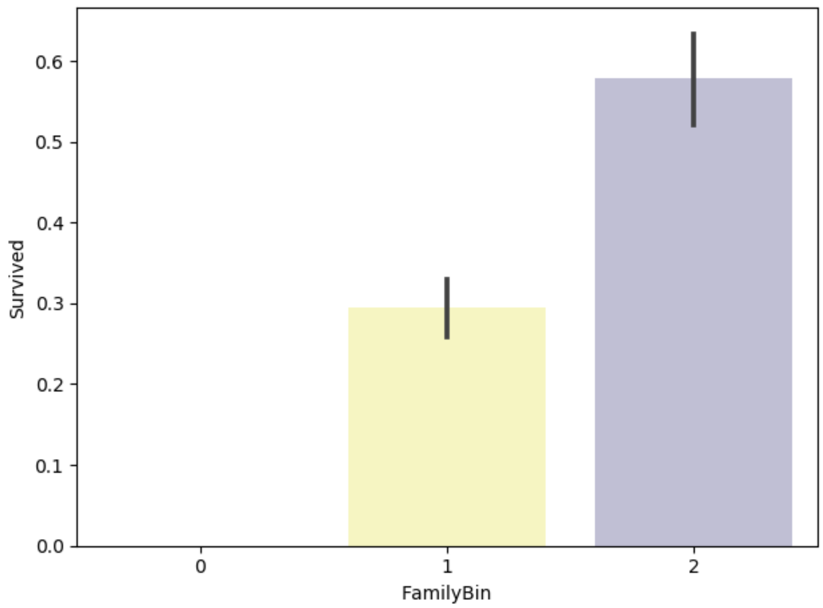

Figure 16 FamilyBin - Survival Rate

From figure 16, we conclude that large families may have difficulty escaping than the small ones. Next, a new feature called FamilyBin is created by dividing FamilySize into 3 categories: 2-4, 4-7 and above 7 persons per family.

2,1,0 represents the three categories mentioned above. Then we can drop columns Parch, SibSp, and FamilySize.

3.3 TicketSize

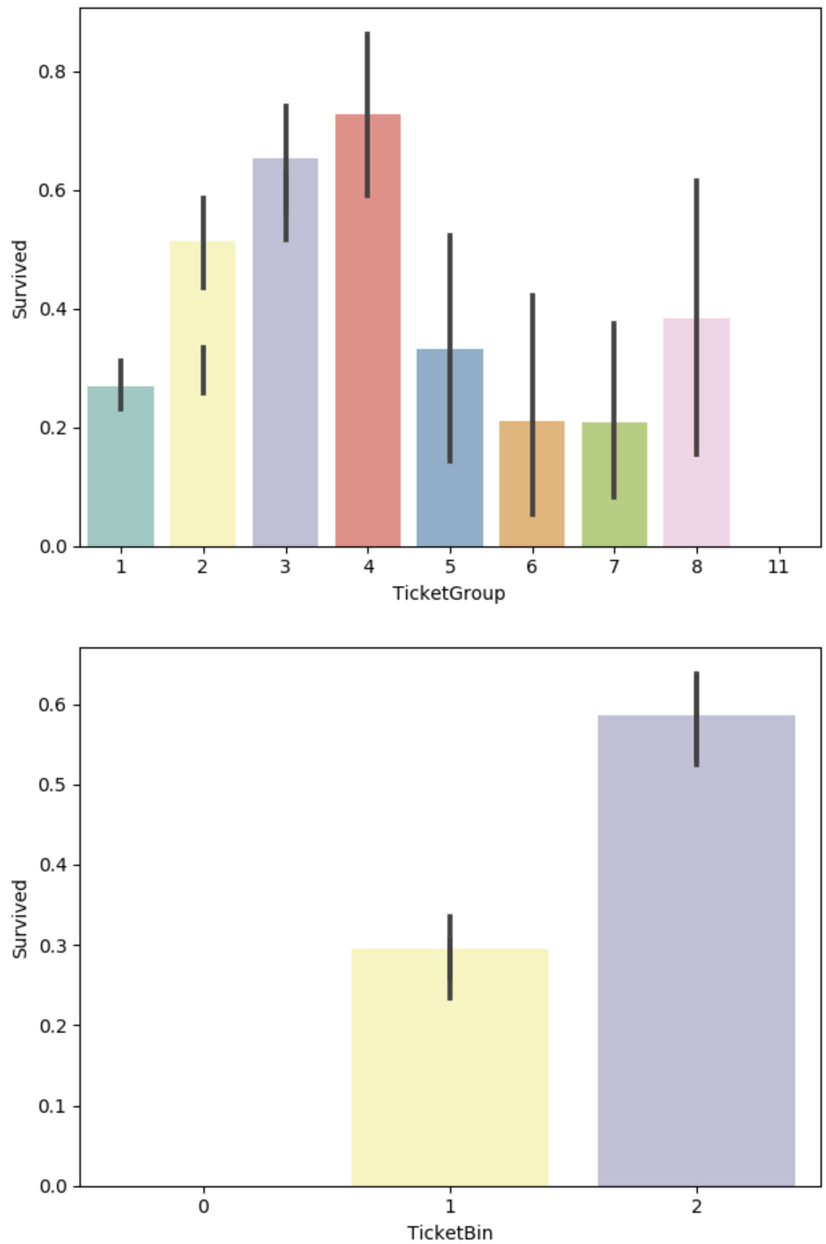

Figure 17 Ticket - Survival Rate

There are 681 unique Ticket numbers in 891 training data, which means some of the numbers appears more than once. Similar to what we've done to Parch and SibSp, we name the time of a ticket number's occurrences TicketGroup. A new feature called TicketBin is then created by dividing TicketGroup into 3 categories: appear 2-4 times, 4-8 times or once and above 8 times.

2,1,0 represents the three categories mentioned above. Then we can drop columns Ticket and TicketSize.

4. Correcting: Dealing with outliners

A special case is taken into consideration. We divide the passengers into different groups based on their surname. We divide the groups which have more than one passenger into two sets: female and children, male adult. We can see that female and children group mostly all died or all survived. It is the same with male adult group. We define the group in female and children group with 0.0 survival rate as died group. Then, we define the group in male adult group with 1.0 survival rate as survived group. We need to deal with variables in these two abnormal groups by changing their values and attributes. For example, we can change the sex in the survived group from ‘female’ to ‘male’.

5. Dropping: PassengerID & Cabin

Unrelated columns, drop.

6. Clean data

 # new schema of dataset

print(train.info())

print(test.info())

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 891 entries, 0 to 890
Data columns (total 9 columns):
Embarked     891 non-null int64
Fare         891 non-null int64
Pclass       891 non-null int64
Sex          891 non-null int64
Survived     891 non-null float64
Title        891 non-null int64
FamilyBin    891 non-null int64
TicketBin    891 non-null int64
AgeBin       891 non-null category
dtypes: category(1), float64(1), int64(7)
memory usage: 63.5 KB
None
<class 'pandas.core.frame.DataFrame'>
Int64Index: 418 entries, 891 to 1308
Data columns (total 8 columns):
Embarked     418 non-null int64
Fare         418 non-null int64
Pclass       418 non-null int64
Sex          418 non-null int64
Title        418 non-null int64
FamilyBin    418 non-null int64
TicketBin    418 non-null int64
AgeBin       418 non-null category
dtypes: category(1), int64(7)
memory usage: 26.5 KB
None
```

Figure 18 Clean data schema

Model Training

1. Model Selection

We have tried multiple machine learning algorithms and compared them for different scenarios. We choose:

1) Ensemble Methods: RandomForest Classifer, ExtraTrees Classifier and AdaBoost Classifier

2) Navies Bayes: naive_bayes.BernoulliNB, naive_bayes.GaussianNB()

3) DecisionTree: tree.DecisionTreeClassifier(), tree.ExtraTreeClassifier()

4) Logistic Reegression

	MLA Name	MLA Test Accuracy Mean
2	RandomForestClassifier	0.822761
1	ExtraTreesClassifier	0.822015
6	DecisionTreeClassifier	0.81791
7	ExtraTreeClassifier	0.815299
3	LogisticRegressionCV	0.808209
0	AdaBoostClassifier	0.801866
4	BernoulliNB	0.78806
5	GaussianNB	0.758209

Figure 19 Machine learning algorithms -test accuracy

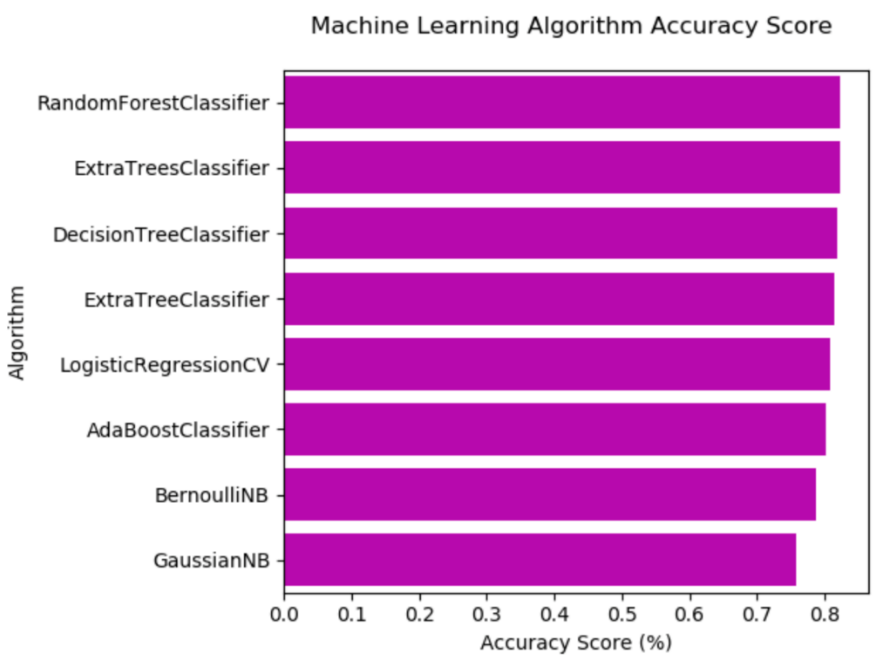

Figure 20 Machine learning algorithms - Test accuracy score

We can learn from figure 20 that Random Forest Model have highest score so that we decide to use it and tune this model.

2. Tune Random Forest

Originally use parameters as follows:

```
{'bootstrap': True,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': 'auto',
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_impurity_split': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 'warn',
 'n_jobs': None,
 'oob_score': False,
 'random_state': 0,
 'verbose': 0,
 'warm_start': False}
```

Then tune the parameters with RandomizedSearchCV and find the best parameters:

```
{'bootstrap': False,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': 3,
 'max_features': 'auto',
```

'max_leaf_nodes': None,

'min_impurity_decrease': 0.0,

'min_impurity_split': None,

'min_samples_leaf': 2,

'min_samples_split': 2,

'min_weight_fraction_leaf': 0.0,

'n_estimators': 1000,

'n_jobs': None,

'oob_score': False,

'random_state': 0,

'verbose': 0,

'warm_start': False}

Evaluate the Model & Cross-Validation

1. Cross-Validation

We get 83.27% mean CV score using Random Forest Model.

```
#Cross Vaildation

cv_score = cross_val_score(RFC1,train[dataTrain_x_bin], train[Target], cv= 10)

print("CV Score : Mean - %.7g | Std - %.7g " % (np.mean(cv_score), np.std(cv_score)))
```

```
CV Score : Mean - 0.8327554 | Std - 0.04070334
```

Figure 21 Cross Validation

2. Evaluate the model

We also evaluate another model using decision-tree model and achieve the accuracy of 77.9%. So we believe that the first model is better than decision-tree model.

3. Submit

```
submit = dataTest[['PassengerId','Survived']]
submit.to_csv("submit.csv", index=False)

print('Validation Data Distribution: \n', dataTest['Survived'].value_counts(normalize = True))
submit.sample(10)

print("Done")
```

Results on Kaggle

1 Active Competition

Titanic: Machine Learning from Disaster
Start here! Predict survival on the Titanic and get familiar with ML basics
Getting Started · Ongoing · tutorial, tabular data, binary classification

345/10628
Top 4%

We submitted to Kaggle 22 times in order to see our result and improve our model.

Accuracy/Score: 0.83253

Your most recent submission

Name	Submitted	Wait time	Execution time	Score
submit.csv	an hour ago	0 seconds	3 seconds	0.83253

Complete

Rank: 345/10628 (Top 4%)

345	new	INFSCI 2725 - Fall 2018	0.83253	22	1h

Your Best Entry

Your submission scored 0.83253, which is not an improvement of your best score. Keep trying!

Leaderboard

Overview Data Kernels Discussion Leaderboard Rules Team My Submissions **Submit Predictions**

339	▲ 266	zaid34	0.83253	45	2d
340	new	Dhiego Souto	0.83253	18	2d
341	new	lumimevi	0.83253	4	1d
342	▲ 2295	Pioneer	0.83253	13	7h
343	new	FawnyLu	0.83253	11	2h
344	new	Joe_	0.83253	4	3h
345	new	INFSCI 2725 - Fall 2018	0.83253	22	1h

Your Best Entry

Your submission scored 0.83253, which is not an improvement of your best score. Keep trying!

346	new	yangyin	0.82775	4	2mo
347	▼ 44	universehiro	0.82775	15	1mo
348	▼ 44	Joao Megale	0.82775	10	7d
349	new	hanhua	0.82775	3	2mo

Reference

```
# ------------------------------------------
# Titanic: Machine Learning from Disaster
# Kaggle Competition
# ------------------------------------------
# INFSCI 2725: Data Analytics
# Fall 2018
# ------------------------------------------
# Yunshu Liang (yul219)
# Qi Lu (qil66)
# Erin Price (eep27)
# Ziyue Qi (ziq2)
# Xiaoqian Xu (xix64)
# ------------------------------------------
# Reference:  https://zhuanlan.zhihu.com/p/31743196
# https://www.kaggle.com/ldfreeman3/a-data-science-framework-to-achieve-99-accuracy
# https://zhuanlan.zhihu.com/p/33733586
# ------------------------------------------
```

Our code includes our references in the comments at the beginning of the file. We also list the references here:

- https://www.kaggle.com/ldfreeman3/a-data-science-framework-to-achieve-99-accuracy
- https://zhuanlan.zhihu.com/p/31743196
- https://zhuanlan.zhihu.com/p/33733586