INFSCI 2725 Data Analytics
Final Report

Zechen Wang, Weihan Chen, Zhehao Guo, Junping Wang, Zixuan Dong

zew20@pitt.edu, wec85@pitt.edu, zhg26@pitt.edu, juw86@pitt.edu, zid5@pitt.edu

Team Name: Gentlemen, Score: 0.81818, Position: 618

1. Introduction

RMS Titanic was a British passenger liner which unfortunately sank in 15 April 1912. It was a largest ship at that time but it made a great and astounding tragedy by colliding with an iceberg.

However, there were still some survivors in this tragedy, which makes us wonder to figure out what sorts of people may survive. In general, we can analyze some information about the passengers so that we may predict which passengers are likely to survive in the shipwreck.

In order to find a better approach to complete this challenge, the prediction model should be designed to maintain the accuracy. The approach and method we use need to be continually improved and facilitated to accomplish the goal.

In this challenge, we possess three data sources which are gender_submission.csv, test.csv and train.csv. In the file of gender_submission.csv, there is a set of predictions that assume all and only female passengers survive, as an example of what a submission file should look like. In the file of train.csv, there is outcome (also known as the "ground truth") for each passenger and the model is based on "features" like passengers' gender and class. And in the file of test.csv, we can see how well the model performs on unseen data without the ground truth for each passenger.

2. Analytic methodology

In the training data, there are 12 features and 891 data collections. One of the features is "Survived", which is the value we want to figure out with our prediction. In order to build our model, the first step is to visualize the data set to observe which features may play important roles in our prediction. We need to draw plots to compare different features. The second step is data cleaning. There are some missing data in the data set which will affect our prediction, so we need to fill these missing values. The third step is feature conversion. We have to transform the features with non-numeric variables into numeric ones, so that it can be used in the prediction model. For example, the "Sex" feature is transformed to Boolean value so that they can fit into numerical analysis, and for the “Embarked” feature, since it has only three categories (C, Q, and S), one hot encoding is applied for the binarization so that they have equal weigh in the analysis, which is good for prediction. The last step is building models. We need to compare different models to choose the best one for prediction.

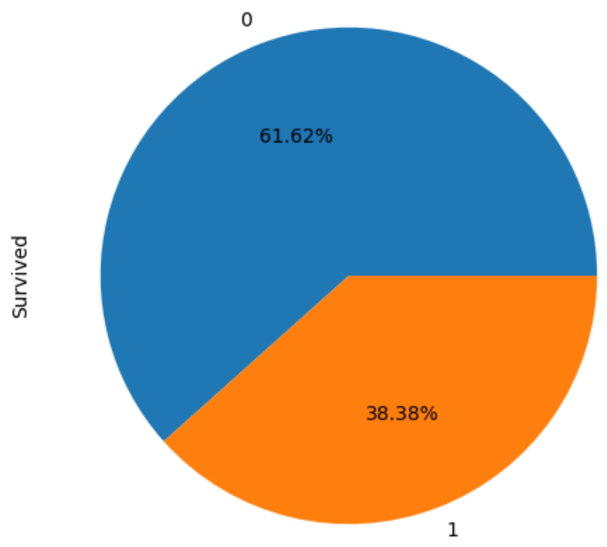
3. Process

I. Data visualization

Analyze the data in the data set first to see which features may affect the predictions.

(1) Train.csv

```
<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
```

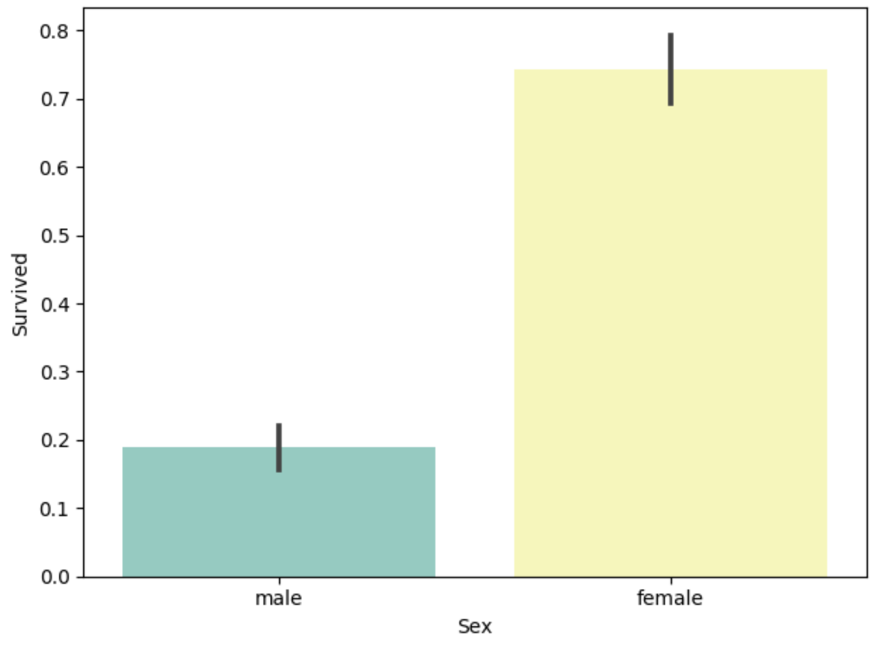


The overall survival situation of the data set is shown above.

Test.csv

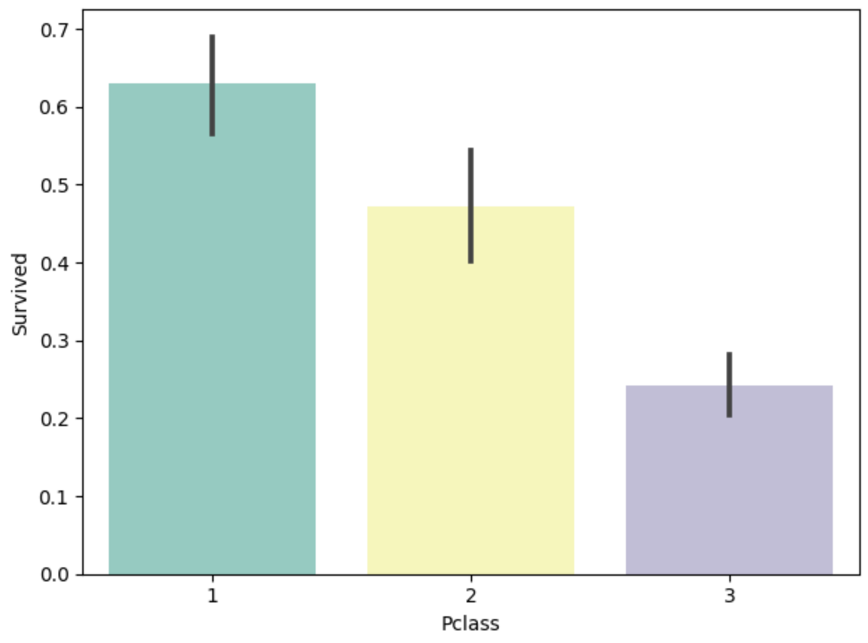
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
PassengerId    418 non-null int64
Pclass         418 non-null int64
Name           418 non-null object
Sex            418 non-null object
Age            332 non-null float64
SibSp          418 non-null int64
Parch          418 non-null int64
Ticket         418 non-null object
Fare           417 non-null float64
Cabin          91 non-null object
Embarked       418 non-null object
dtypes: float64(2), int64(4), object(5)
memory usage: 36.0+ KB
```

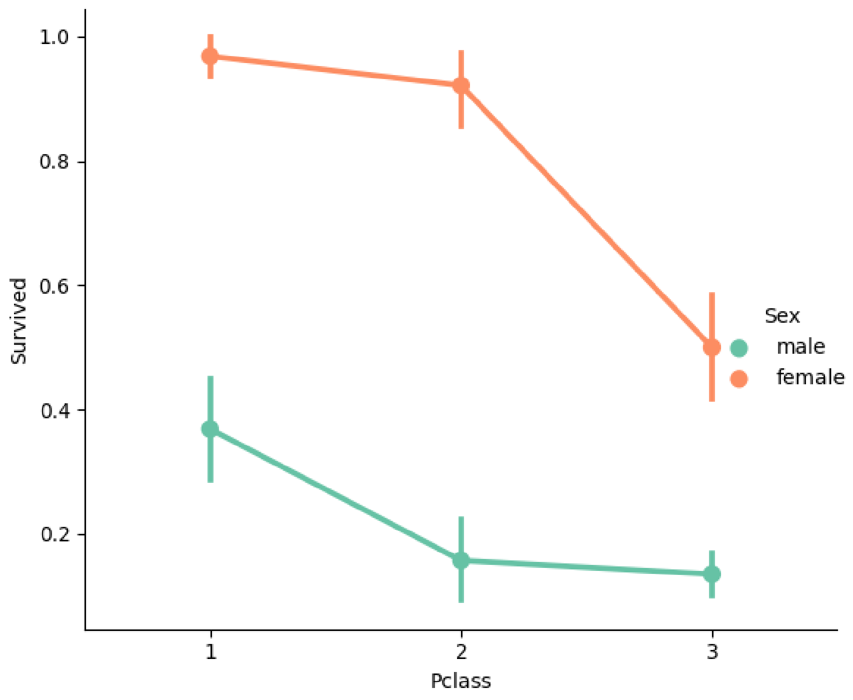
(2) Sex attribute



Female has a higher survival rate.

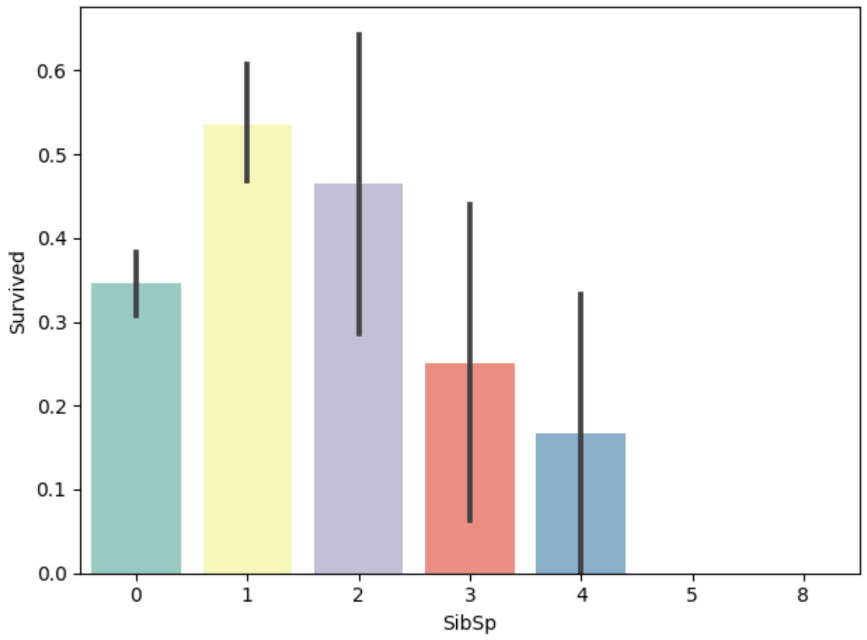
(3) Pclass attribute

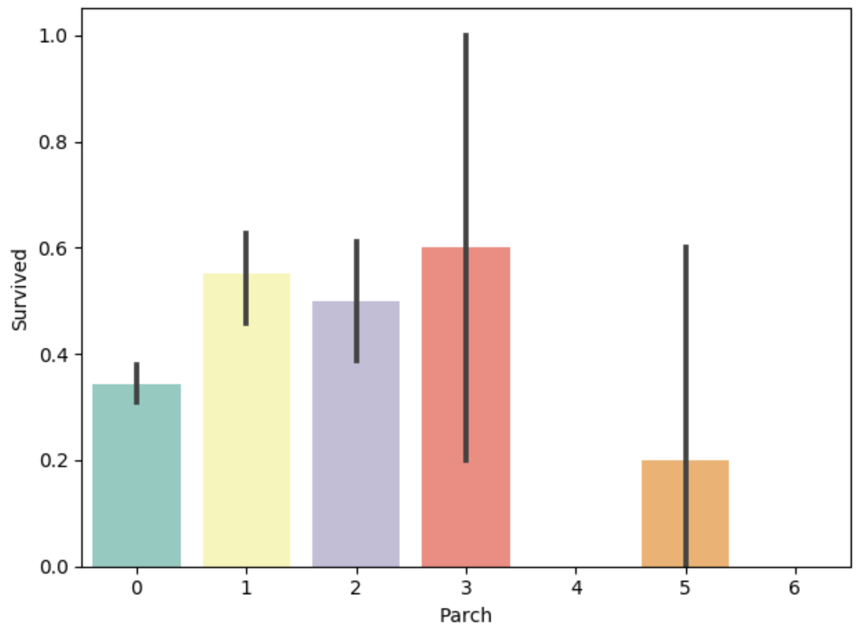




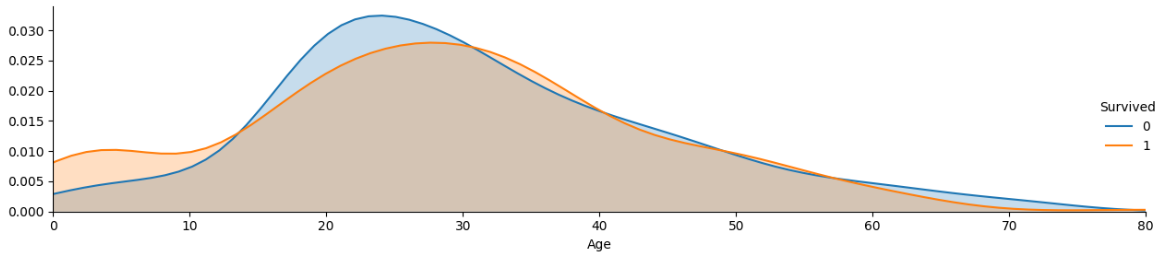
The higher the social level is, the higher survival rate will be.

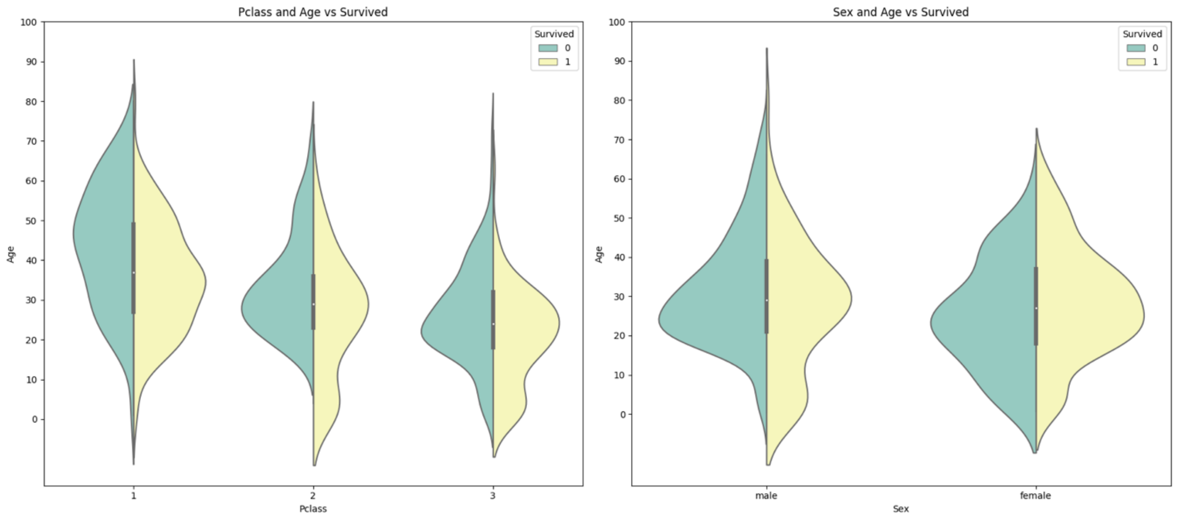
(4) SibSp attribute, Parch attribute





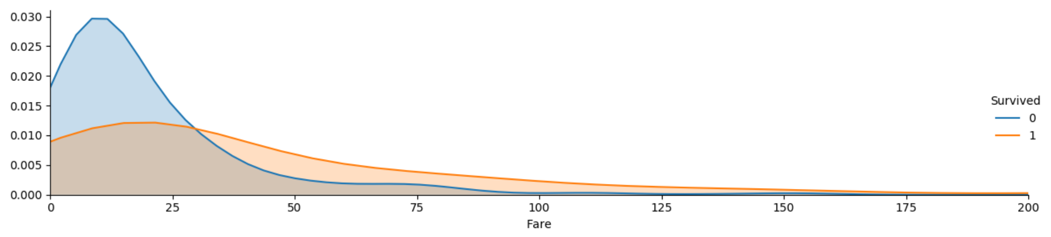
(5) Age attribute





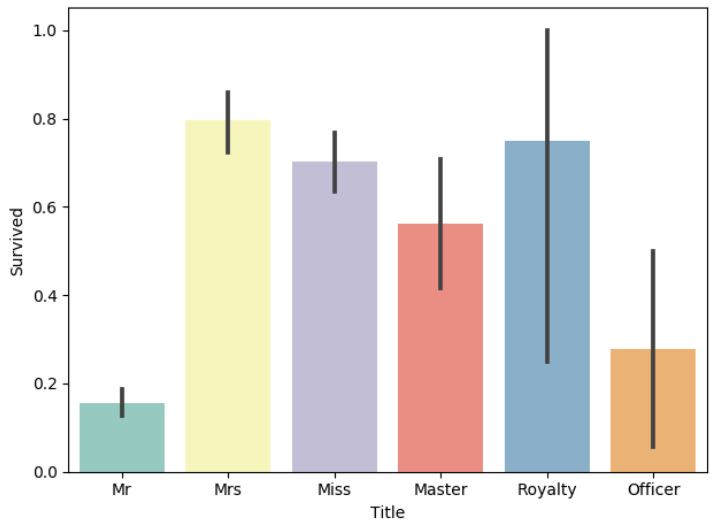
Minors have a higher survival rate than adults.

(6) Fare attribute



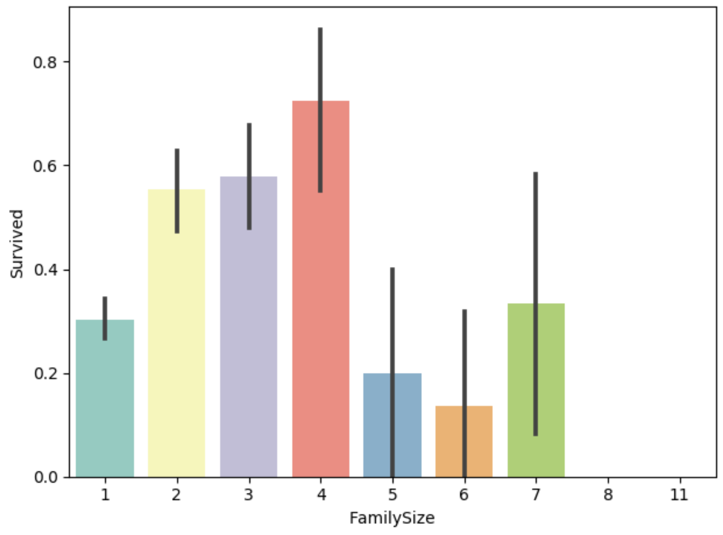
People who spend more on their tickets have a higher survival rate.

(7) Title feature



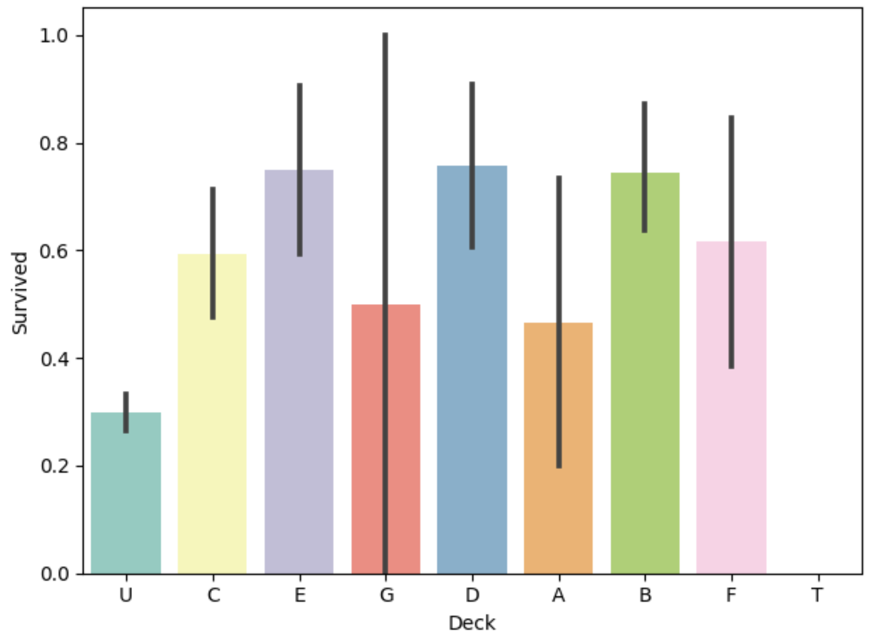
Passengers with different titles have different survival rates.

(8) Family size feature



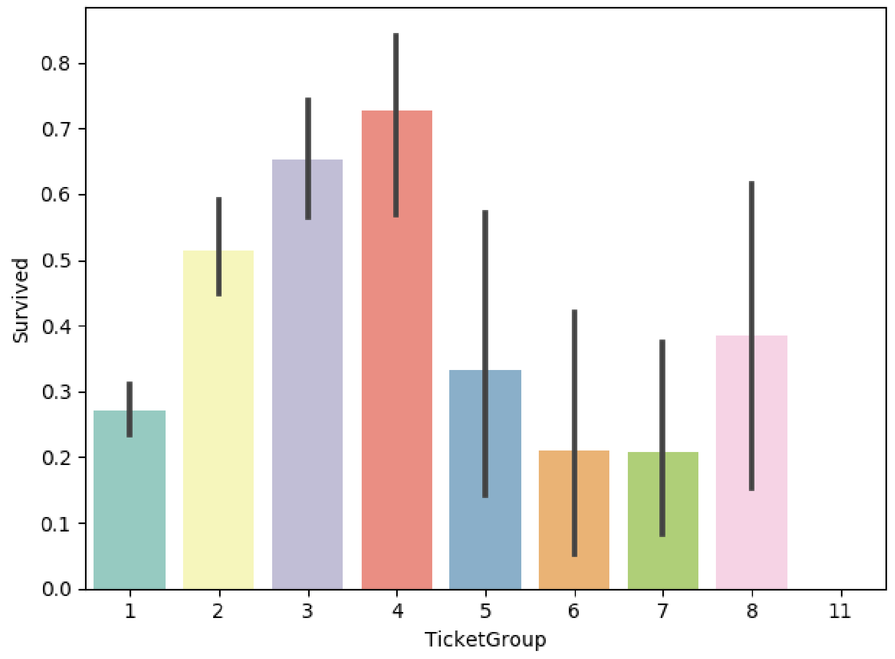
Passengers with family size of 2 to 4 have higher survival rates.

(9) Deck feature



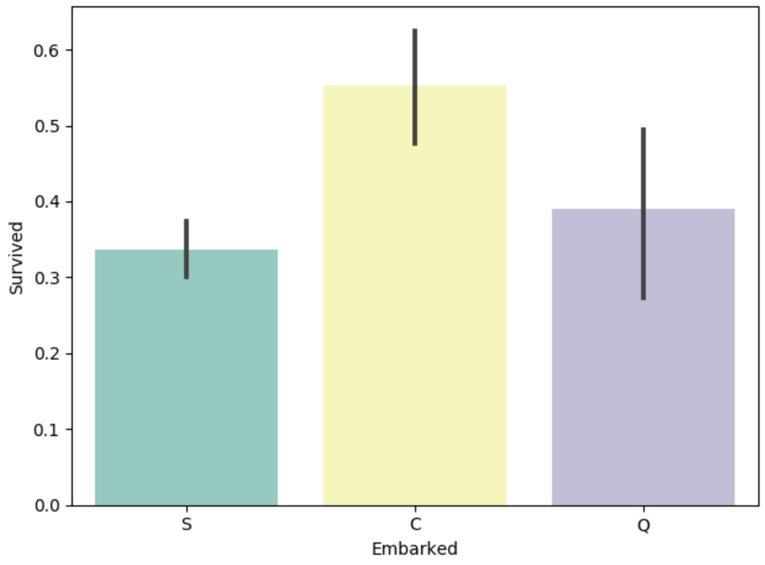
Passenger survival rates vary from deck to deck. (We set the null value as unknown)

(10) Ticket Group feature



Passengers have same ticket number with other 1 to 3 passengers have a higher survival rate.

(11) Embarked feature

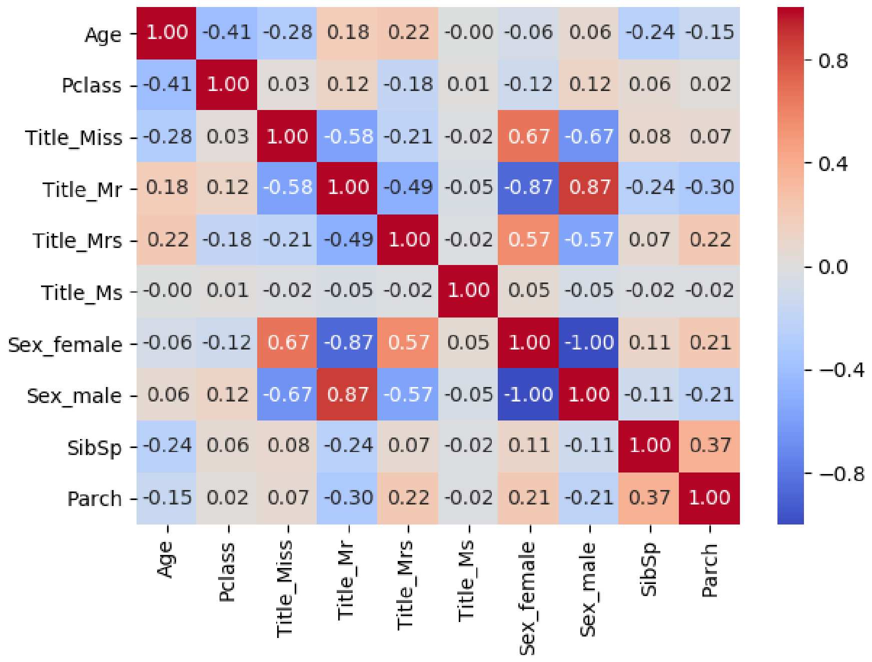


Passenger survival rates vary from port to port.

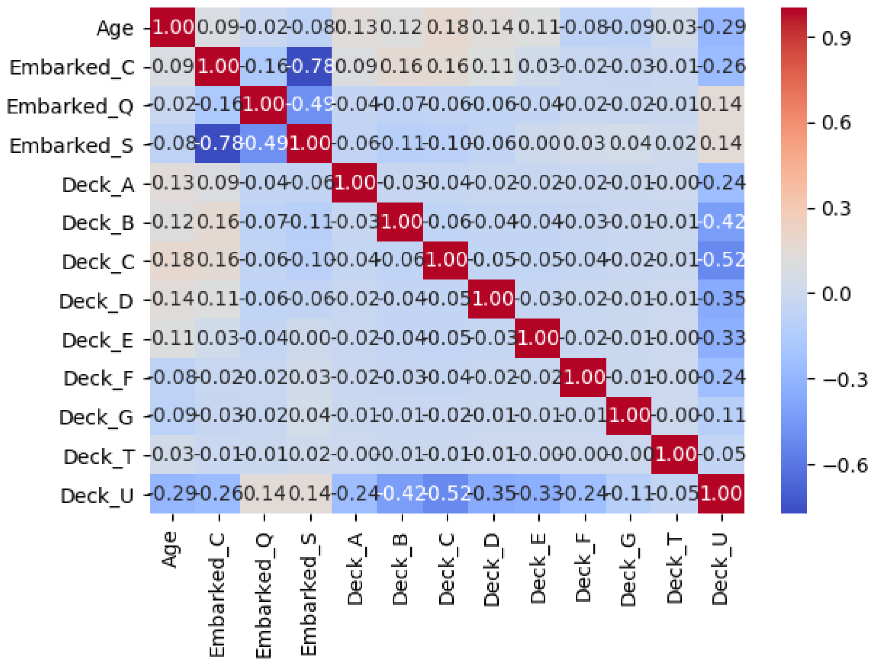
II. **Data cleaning**

From the data frame of the train data set and test data set, we can see that Age, Cabin and Embarked attributes have some missing values. Therefore, we need to fill the missing data.

'Age' attribute has many missing values, under this circumstance, we need to use other related attributes to predict the missing values.



	Age	Pclass	Title_Miss	Title_Mr	Title_Mrs	Title_Ms	Sex_female	Sex_male	SibSp	Parch
Age	1.00	-0.41	-0.28	0.18	0.22	-0.00	-0.06	0.06	-0.24	-0.15
Pclass	-0.41	1.00	0.03	0.12	-0.18	0.01	-0.12	0.12	0.06	0.02
Title_Miss	-0.28	0.03	1.00	-0.58	-0.21	-0.02	0.67	-0.67	0.08	0.07
Title_Mr	0.18	0.12	-0.58	1.00	-0.49	-0.05	-0.87	0.87	-0.24	-0.30
Title_Mrs	0.22	-0.18	-0.21	-0.49	1.00	-0.02	0.57	-0.57	0.07	0.22
Title_Ms	-0.00	0.01	-0.02	-0.05	-0.02	1.00	0.05	-0.05	-0.02	-0.02
Sex_female	-0.06	-0.12	0.67	-0.87	0.57	0.05	1.00	-1.00	0.11	0.21
Sex_male	0.06	0.12	-0.67	0.87	-0.57	-0.05	-1.00	1.00	-0.11	-0.21
SibSp	-0.24	0.06	0.08	-0.24	0.07	-0.02	0.11	-0.11	1.00	0.37
Parch	-0.15	0.02	0.07	-0.30	0.22	-0.02	0.21	-0.21	0.37	1.00



From the heat map, we can see that ‘Age’ attribute has some relationship with ‘Pclass’, ‘Title’, ‘SibSp’, ‘Parch’ and ‘Deck’ attributes. We combine these attributes and test each pairs of combinations to see which pair has a better prediction for ‘Age’ attribute. And in the end, the pair ['Pclass', 'Title', 'Deck'] performs best.

For the ‘Embarked’ attribute, it only has two missing data, and both passengers are in ‘Pclass’ 1 with ‘Fare’ 80. Because the median ‘Fare’ of the passengers with ‘Pclass’ 1 and ‘Embarked’ C is 80, we can fill the missing data with ‘C’.

For the ‘Fare’ attribute, it has only one missing data with the passenger in ‘Pclass’ 3 and ‘Embarked’ S, therefore, we can simply fill the data with the median ‘Fare’ of passengers in ‘Pclass’ 3 and ‘Embarked’ S.

III. **Data Aggregation and Grouping**

In each person’s name, there is a title which indicates their identity. We need to extract these titles and regroup them.

```
all_data['Title'] = all_data['Name'].apply(lambda x: x.split(',')[1].split('.')[0].strip())
```

We tried 3 different regroup scheme.

The 1st scheme is made according to the social meanings of the titles:

[‘Miss’, ‘Mlle’, ‘Ms’] as a group,

['Mrs', 'Mme'] as a group

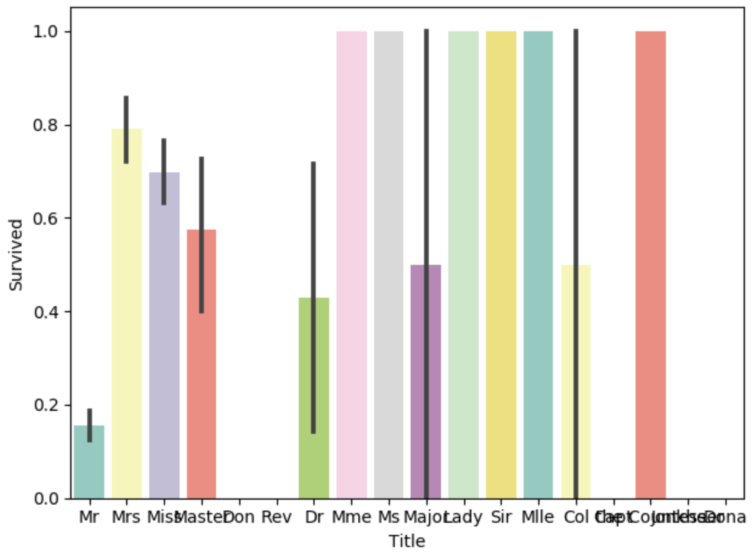
['Capt', 'Col', 'Don', 'Dr', 'Major', 'Rev', 'Jonkheer'] as a group

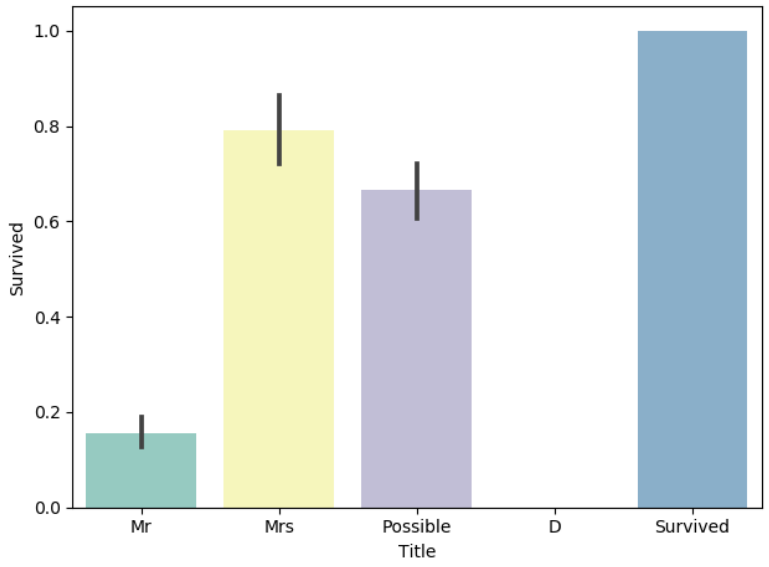
['Dona', 'Lady', 'Sir', 'the Countess'] as a group

['Master'] as a group

['Mr'] as a group

The 2nd scheme is made according to how the data is distributed:





['Mrs'] as a group

['Mr'] as a group

['Capt', 'Rev', 'Don', 'Dona', 'Jonkheer'] as a group

['Miss', 'Master', 'Dr', 'Major', 'Col'] as a group

['Mme', 'Ms', 'Lady', 'Sir', 'Mlle', 'the Countess'] as a group

The 3rd scheme is made according to the social classes of the titles:

['Capt', 'Col', 'Major', 'Dr', 'Rev'] as a group

['Don', 'Sir', 'the Countess', 'Dona', 'Lady'] as a group

['Mme', 'Ms', 'Mrs'] as a group

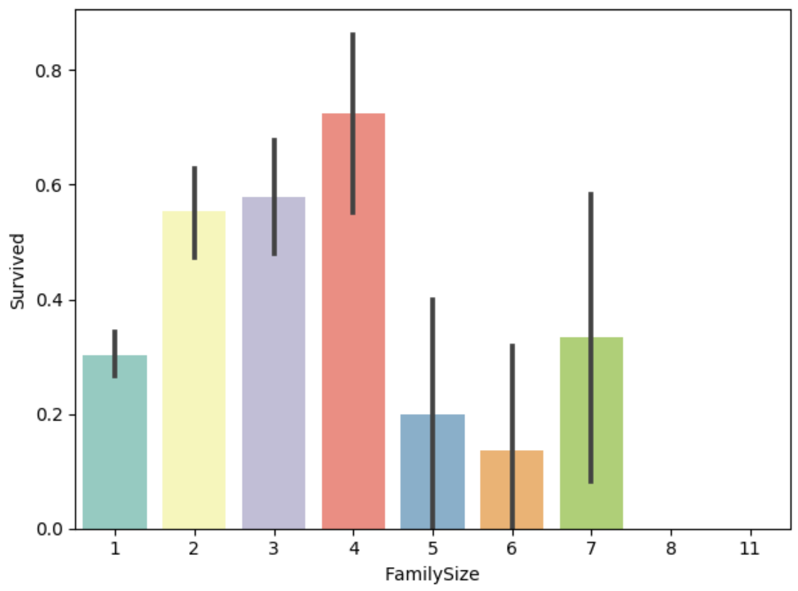
['Mlle', 'Miss'] as a group

['Mr'] as a group

['Master', 'Jonkheer'] as a group

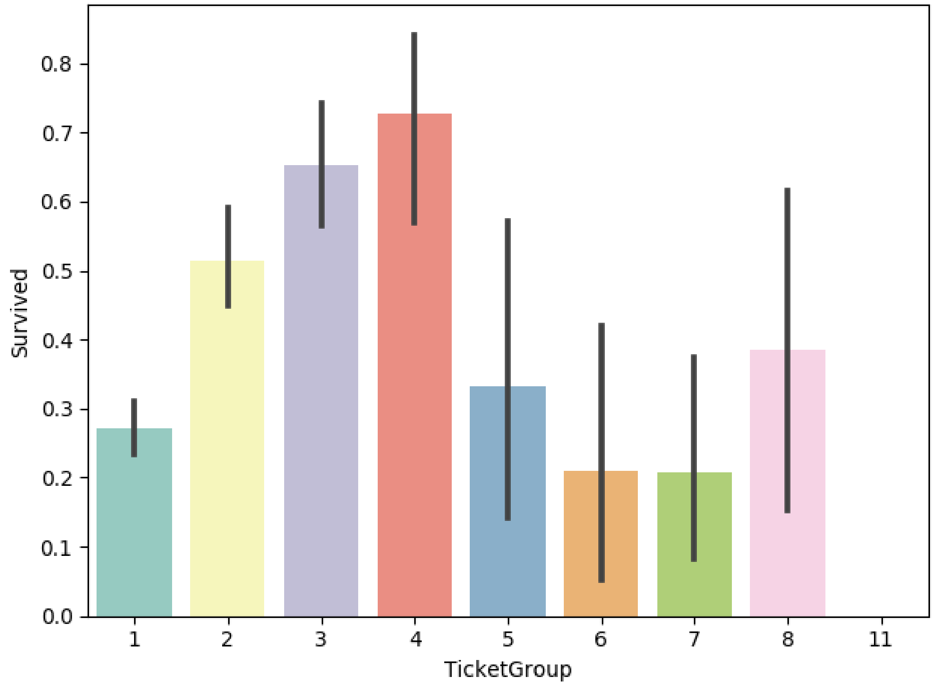
After many tests, we find that different schemes perform differently in different models, but overall, the 3rd scheme has a better accuracy.

We also combine ‘SibSp’ and ‘Parch’ attributes by adding them together to make them as one attribute (‘FamilySize’).



And then, we regroup them with [2, 3, 4], [1, 5, 6 ,7] and the rest.

We do the same thing for the ticket groups, and we regroup them with [2, 3, 4], [1, 5, 6, 7, 8] and the rest (shown as following).



IV. **Feature conversion**

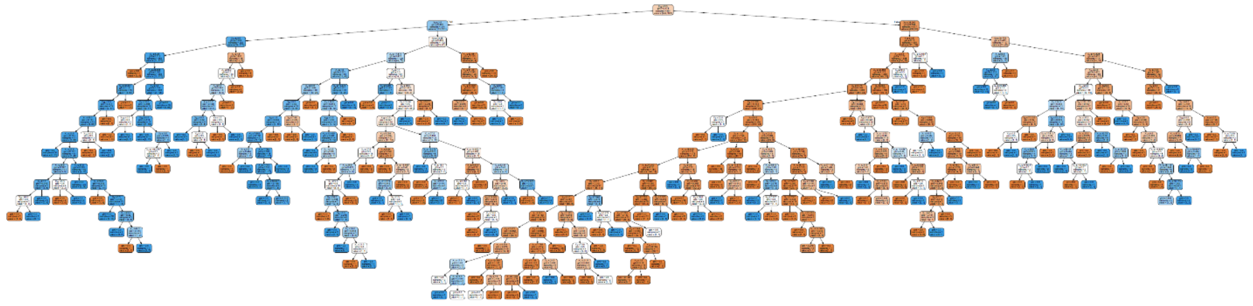
We need to transform the features with non-numeric variables into numeric ones, so that it can be used in the prediction model.

```
all_data = pd.get_dummies(all_data)
train = all_data[all_data['Survived'].notnull()]
test = all_data[all_data['Survived'].isnull()].drop('Survived', axis=1)
X = train.as_matrix()[:, 1:]
y = train.as_matrix()[:, 0]
```

V. **Model**

(1) Model 1: Decision Tree

```
dtc = DecisionTreeClassifier()
dtc.fit(X, y)
```



Kaggle Score: 0.72727

(2) Model 2: Gradient Boosting

```
clf = GradientBoostingClassifier(n_estimators=110, learning_rate=0.1, max_depth=6,
random_state=10, max_features='sqrt')
clf.fit(X, y)
```

Kaggle Score: 0.75598

(3) Model 3: XGBoost

```
training_features = np.array(train.drop(['Survived'], 1).columns)
clf = xgb.XGBClassifier()
cv = KFold(30, shuffle=True, random_state=1)
featSelect = RFECV(estimator=clf, cv=cv, scoring='accuracy')
featSelect.fit(X, y)
selection = np.append(['Survived'], training_features[featSelect.support_])
train2 = train[selection]
X = train2.as_matrix()[:, 1:]
y = train2.as_matrix()[:, 0]
clf.fit(X, y)
```

Kaggle Score: 0.78468

(4) Model 4: Random Forest (Our best model)

We use loops to find best parameters, and the result is K = 20, random_state = 10, n_estimators = 22, max_depth = 6 and the cv_score is 0.8473621041879469.

```
bestSol = 0
for k in range(15, 25):
    for random_state in range(10, 15):
        for estimators in range(20, 50, 2):
            for depth in range(3, 30, 3):
                select = SelectKBest(k=k)
                clf = RandomForestClassifier(random_state=random_state,
                                             warm_start=True,
                                             n_estimators=estimators,
                                             max_depth=depth,
                                             max_features='sqrt')
                pipeline = make_pipeline(select, clf)
                pipeline.fit(X, y)
                cv_score = cross_val_score(pipeline, X, y, cv=10)
                # print(k, random_state, estimators, depth)
                # print("CV Score : Mean - %.7g | Std - %.7g " % (np.mean(cv_score),
np.std(cv_score)))
                if np.mean(cv_score) > bestSol:
                    bestSol = np.mean(cv_score)
                    bestK = k
                    bestRandom = random_state
                    bestEstimator = estimators
                    bestDepth = depth
                    bestScore = np.mean(cv_score)
                    print(bestK, bestRandom, bestEstimator, bestDepth, bestScore)
```

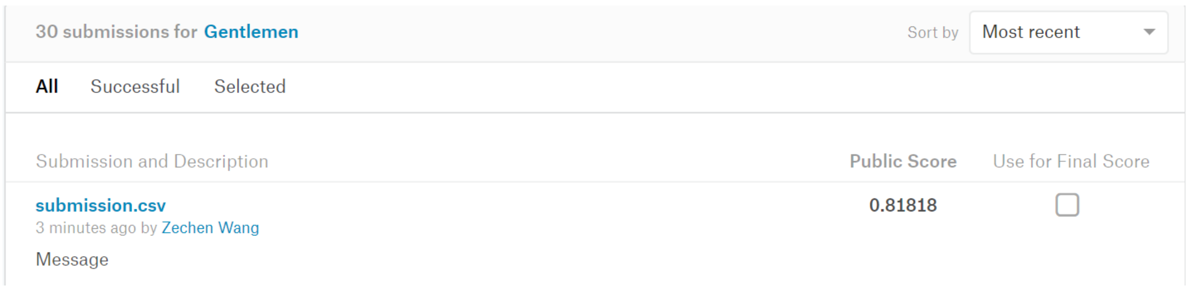
However, after several attempts, we find that if we change n_estimators from 22 to 26, we can get a higher score in Kaggle.

Therefore, the final model is:

```
select = SelectKBest(k=20)
clf = RandomForestClassifier(random_state=10, warm_start=True,
                             n_estimators=26,
                             max_depth=6,
                             max_features='sqrt')
pipeline = make_pipeline(select, clf)
pipeline.fit(X, y)
```

Kaggle Score: 0.81818

4. Performance and Score

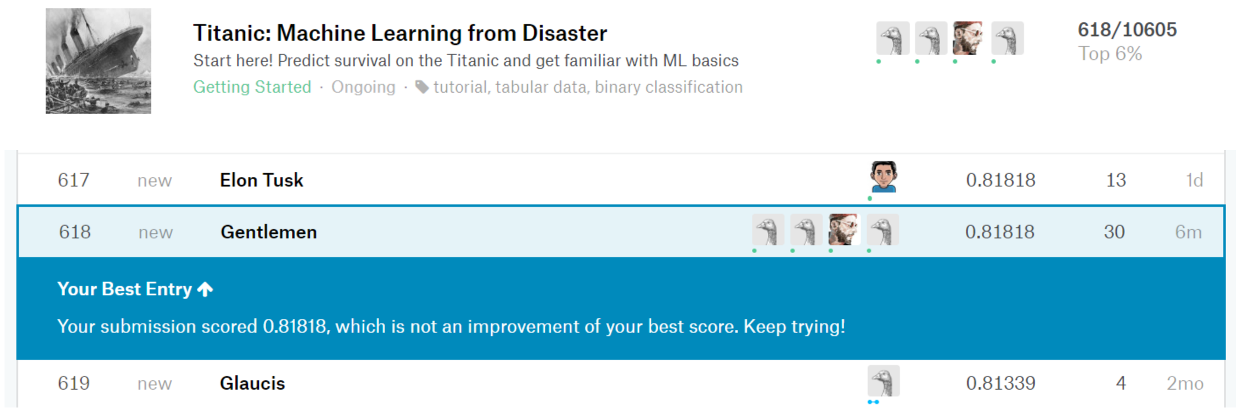


30 submissions for Gentlemen | Sort by Most recent

All | Successful | Selected

Submission and Description	Public Score	Use for Final Score
submission.csv 3 minutes ago by Zechen Wang Message	0.81818	☐

Final score of our submission is 0.81818.



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

618/10605
Top 6%

617	new	Elon Tusk	0.81818	13	1d
618	new	Gentlemen	0.81818	30	6m

Your Best Entry

Your submission scored 0.81818, which is not an improvement of your best score. Keep trying!

619	new	Glaucis	0.81339	4	2mo

Our final position is 618 of 10605 (Top 6%) at present.