

Shipwreck Survivor Prediction Analysis Competition

Zhaofeng Shang

181006463

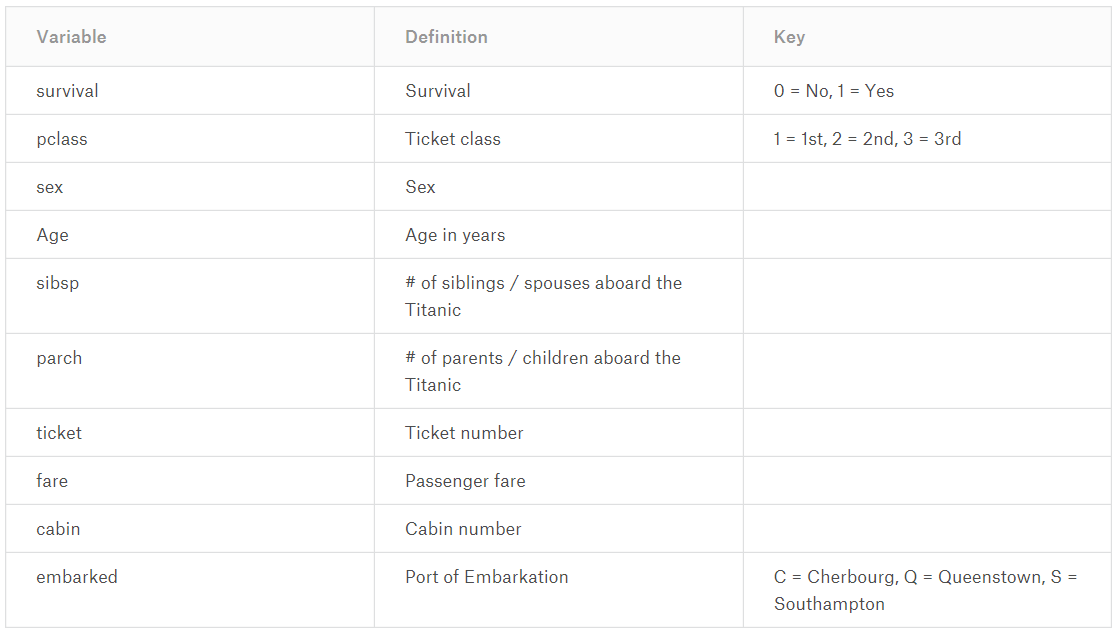
# Business Understanding

The sinking of the RMS Titanic is one of the most infamous shipwrecks in history.  On April 15, 1912, during her maiden voyage, the Titanic sank after colliding with an iceberg, killing 1502 out of 2224 passengers and crew. This sensational tragedy shocked the international community and led to better safety regulations for ships.

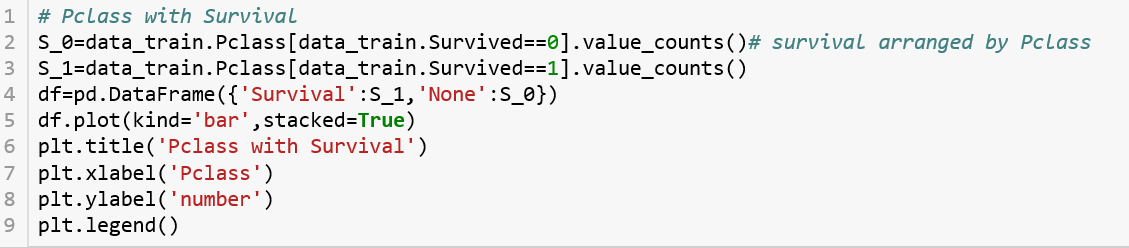
In this challenge, the job is to complete the analysis of what sorts of people were likely to survive. It is necessary to apply the tools of machine learning to predict which passengers survived the tragedy.

# Data Gathering

Here are all the attributes in this dataset, Sibling = brother, sister, stepbrother, stepsister  
Spouse = husband, wife (mistresses and fiancés were ignored).



## Visualization



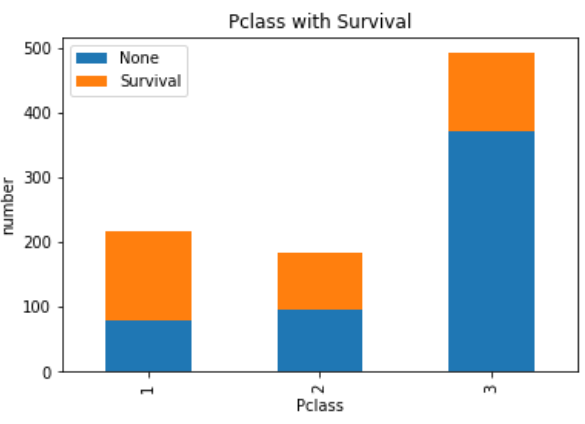
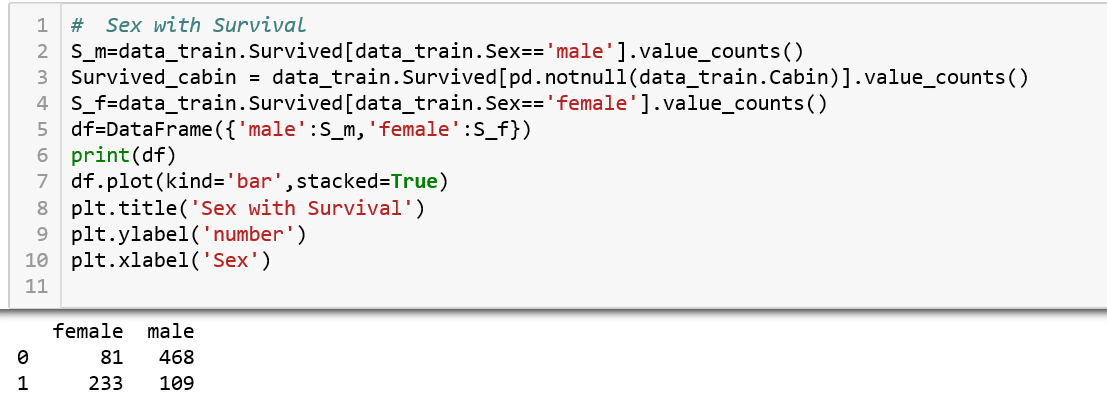
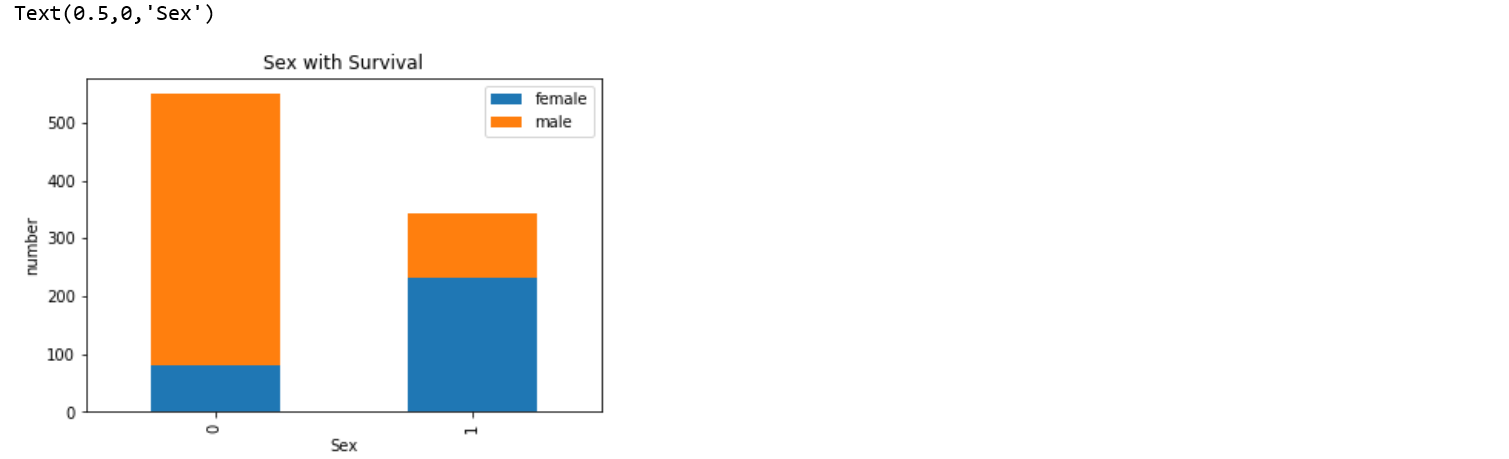
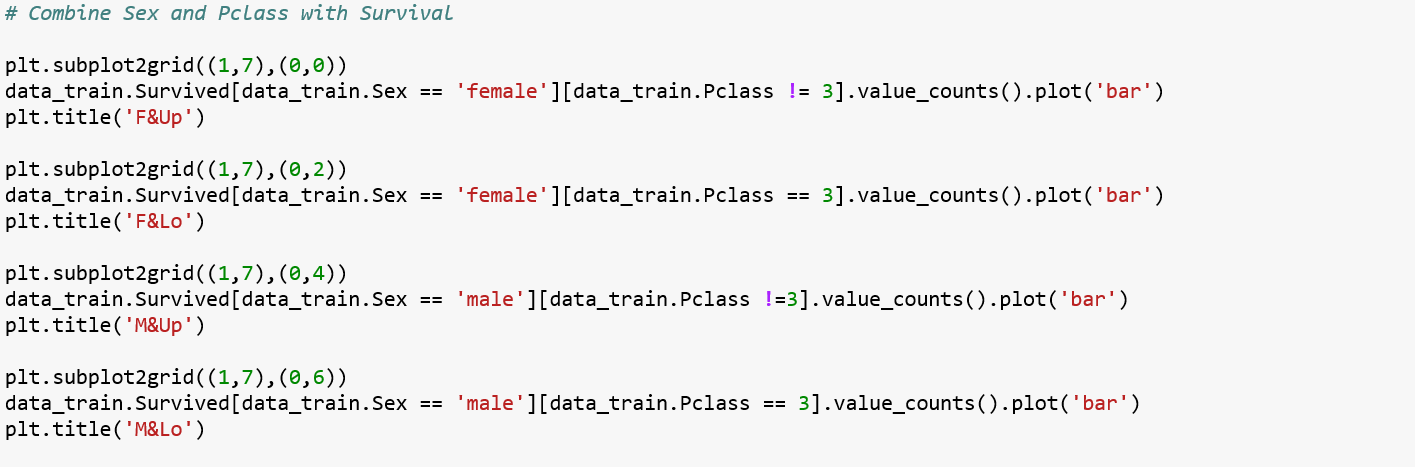


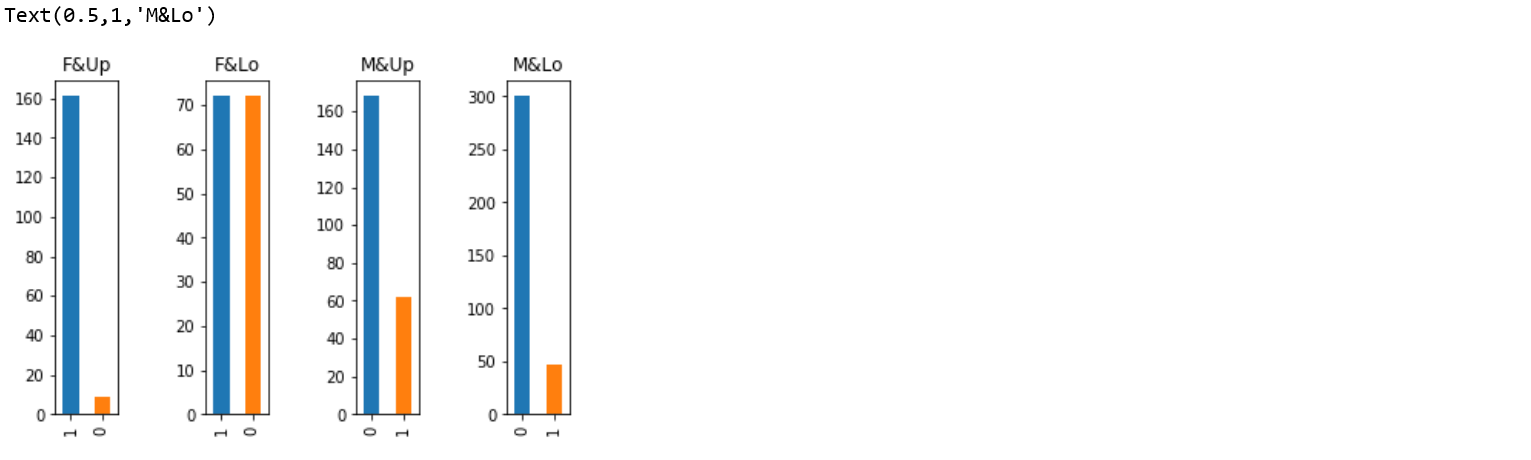
Figure above shows the relationship between “Pclass” and “Survival”, With the reduction of the class level, the number of survivors decreased.



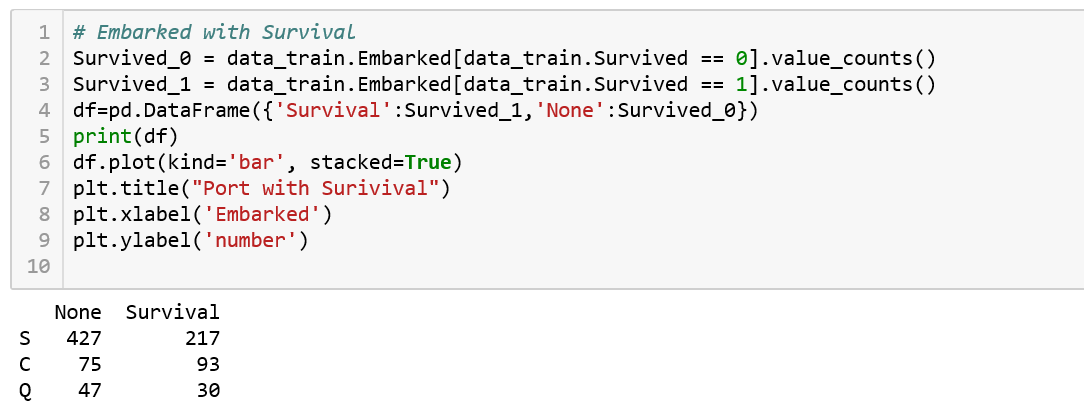


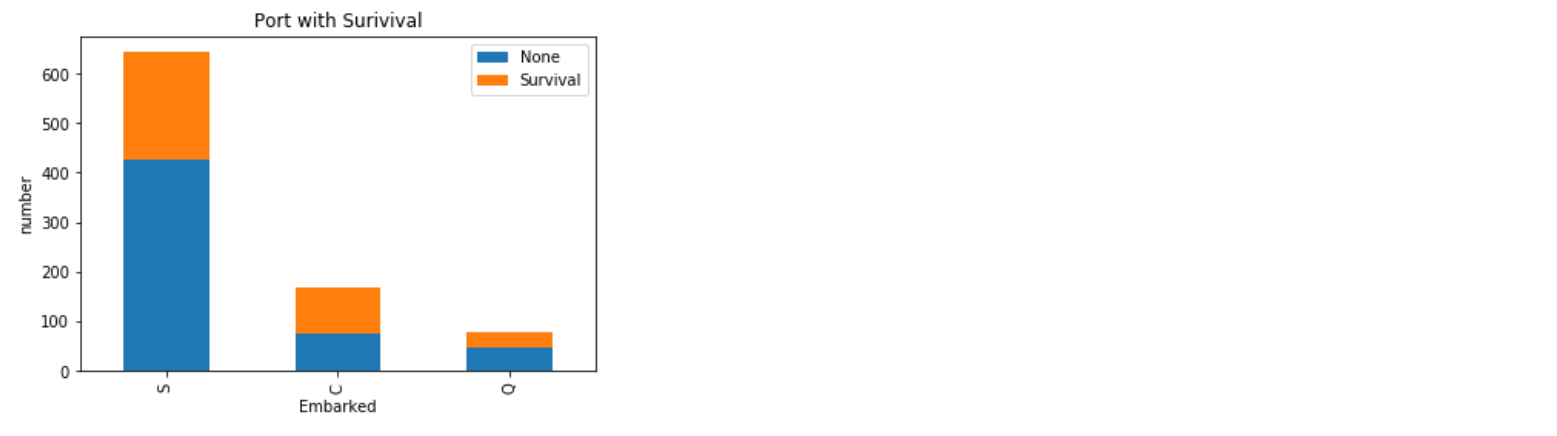
The figure above shows the relationship between “Sex” and “Survival”, “0” in the figure represent not survive and “1” is in other side. Figure shows that the survival rate of female is much higher than male.



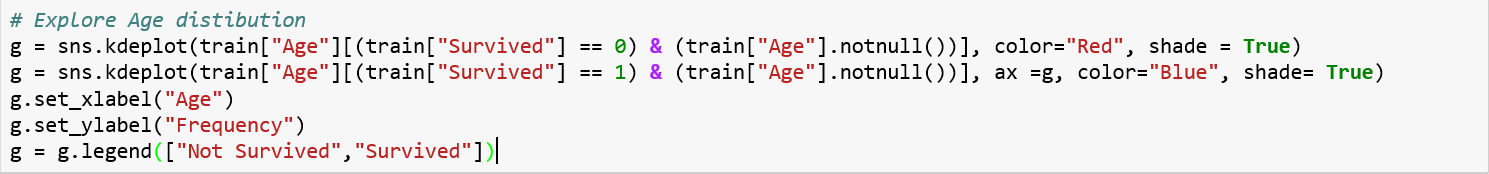
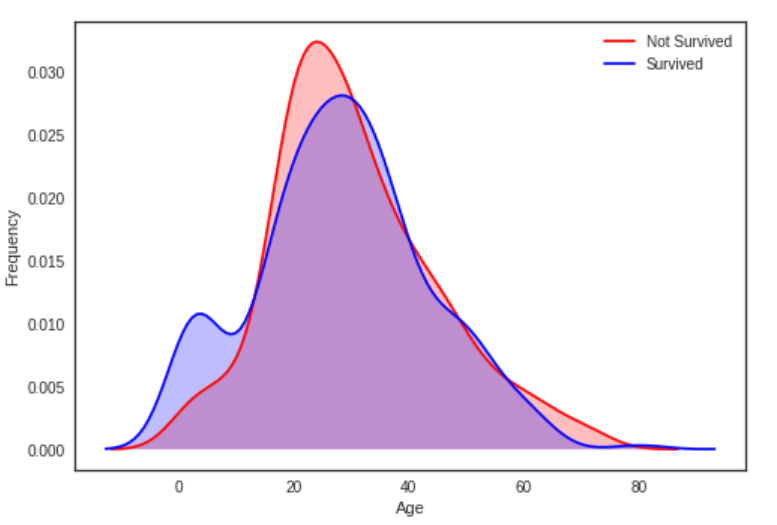


The figure above shows the relationship between “Survival” and the combination of “Sex” and “Pclass”. There are four different combination, all show that higher class level and female will get higher survival rate.

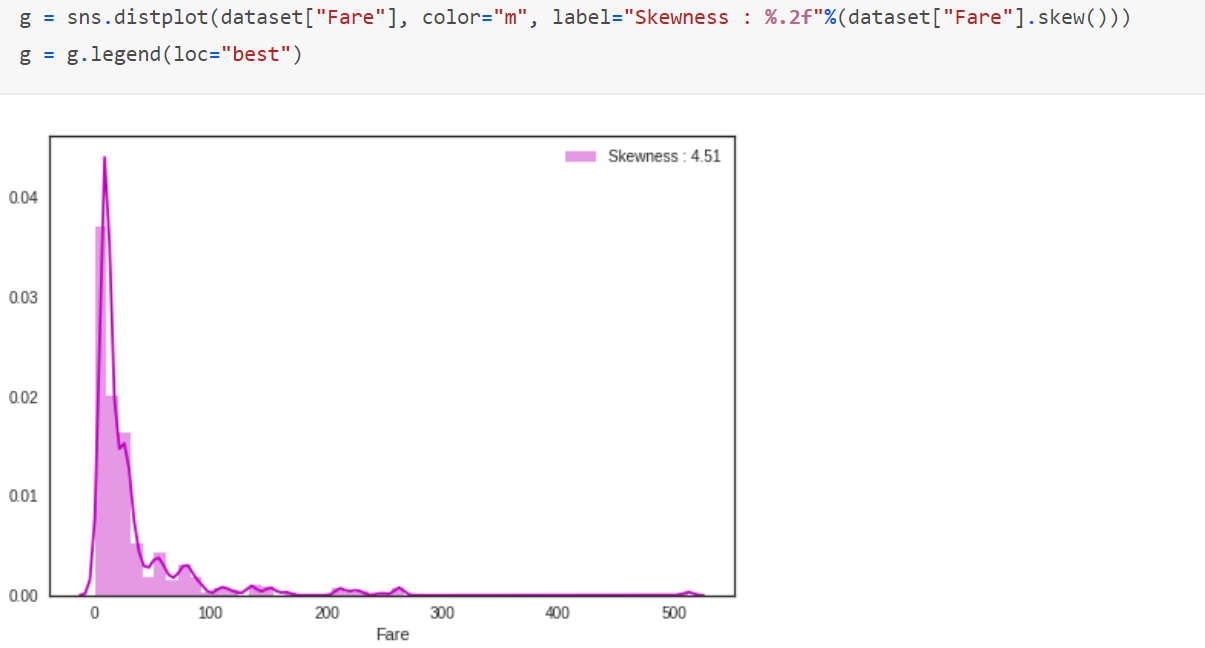




The figure above shows the relationship between “Port” and “Survival”. According to the ratio, the Embarked “C” has higher survival rate.

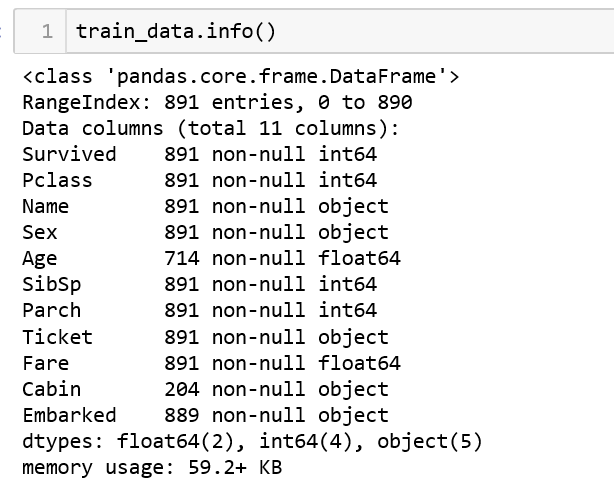


When we overlap the two graphs, it clearly shows a peak between 0-5, which represent the babies and young children. People during 20-30 has highest death rate, which represent the young people. People during 30-40 has high survival rate, which represent those matured people. And, the old person during 60-80 has low survival rate.



The figure above shows that distribution of “Fare” attribute, High skewed Fare distribution can lead to overweight values in the model, even if it is scaled. In this case, it is better to transform it with the log function to reduce this skew.

## Statistical Tests



The table above shows the basic information about the whole dataset, attribute “Age”, “Cabin” and “Embarked” has missing values, the next work is to finish data preprocessing.

# Data Preprocessing

## 3.1 Correcting

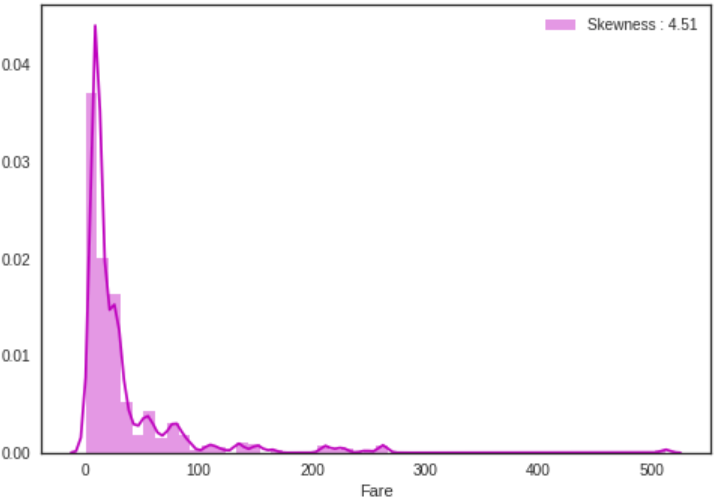
### 3.1.1 Outliers

Handling with outliers in the dataset is essentially important thing to improve the accuracy of prediction model. **Tukey–Kramer method** is a single-step multiple comparison procedure and statistical test. This method defines an interquartile range comprised between the 1st and 3rd quartile of the distribution values (IQR). An outlier is a row that have a feature value outside the (IQR +- an outlier step).

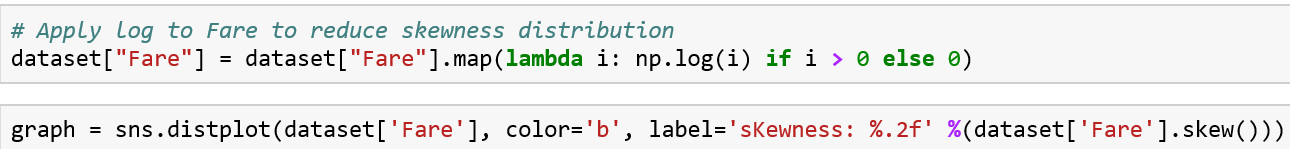
Here, using Tukey-Kramer method to handle the outliers in all numerical variables as bellow:

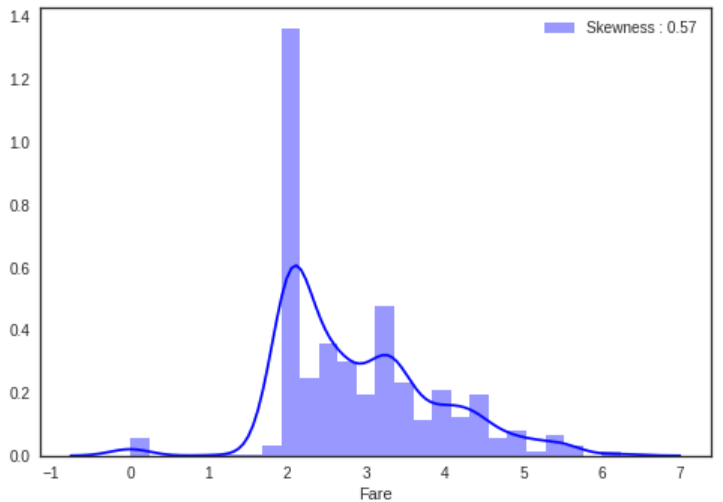


### 3.1.2 Standardization



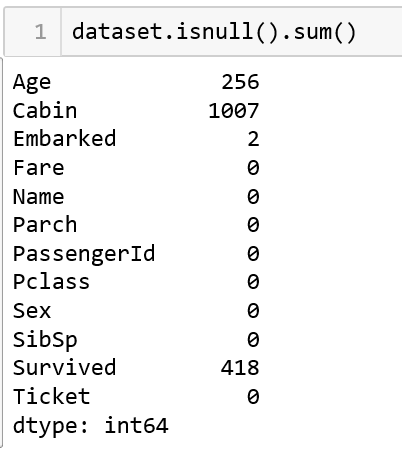
According to the “Fare” distribution above, it is very skewed, which will lead to overweight values in the model. In this case, it’s better to transform it with the log function to reduce the skew.



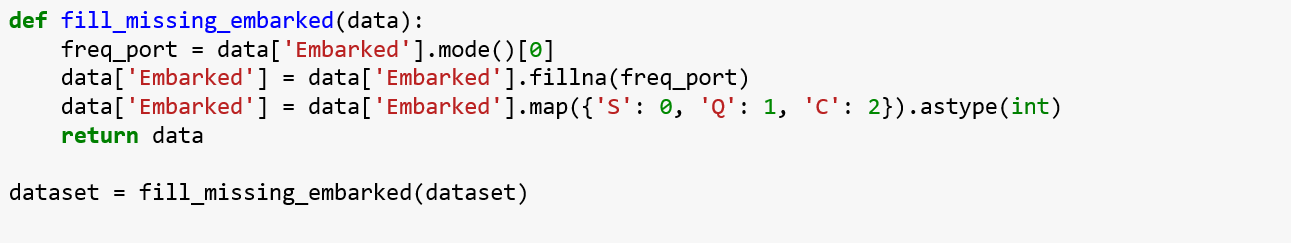
After applying the log function, the distribution of Fare as follow:

## 3.2 Completing

### 3.2.1 Missing values



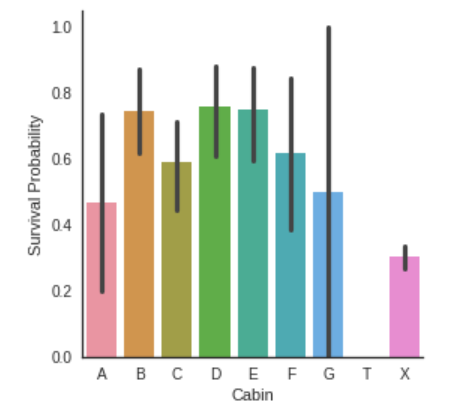
**“Embarked” attribute:**

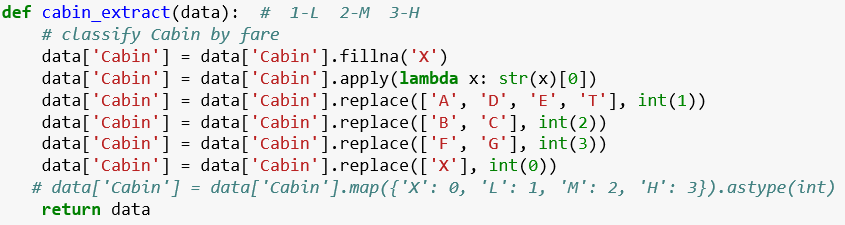


According to the information table above, the number of the “Embarked” missing value is two. Here used the mode() function to fill the missing value, and used map method to change categorical value into the numerical value.

**“Cabin” attribute:**

First method:





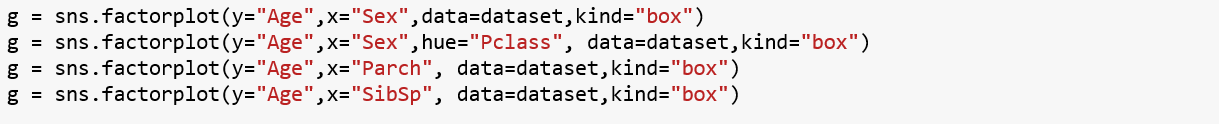
“Cabin” attribute is comprised by letter and number. Created “cabin\_extract” function to fill null data and get the letter of the “Cabin” attribute, grouped all values into four intervals.

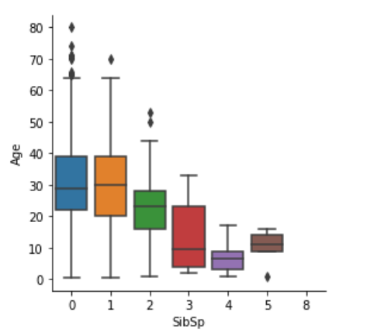
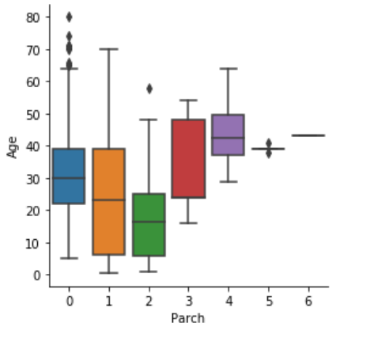
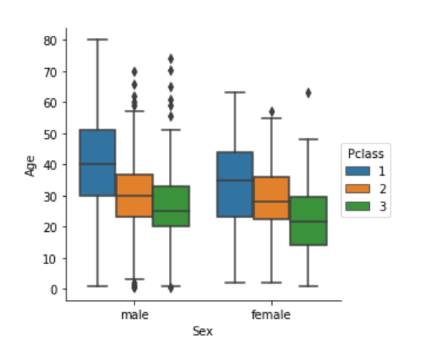
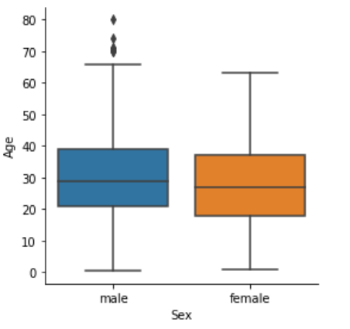
Second method:

Rather than changing the attribute “Cabin” as above, it may be better to delete the whole column because of too much missing values.

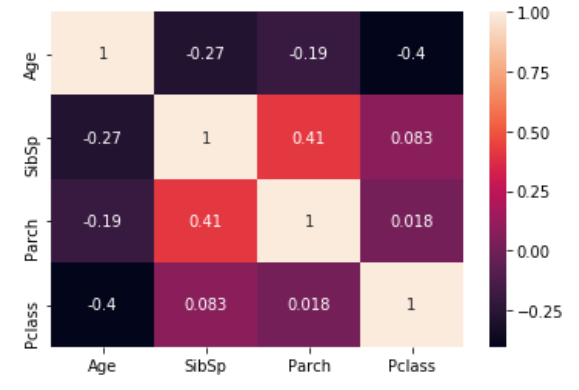


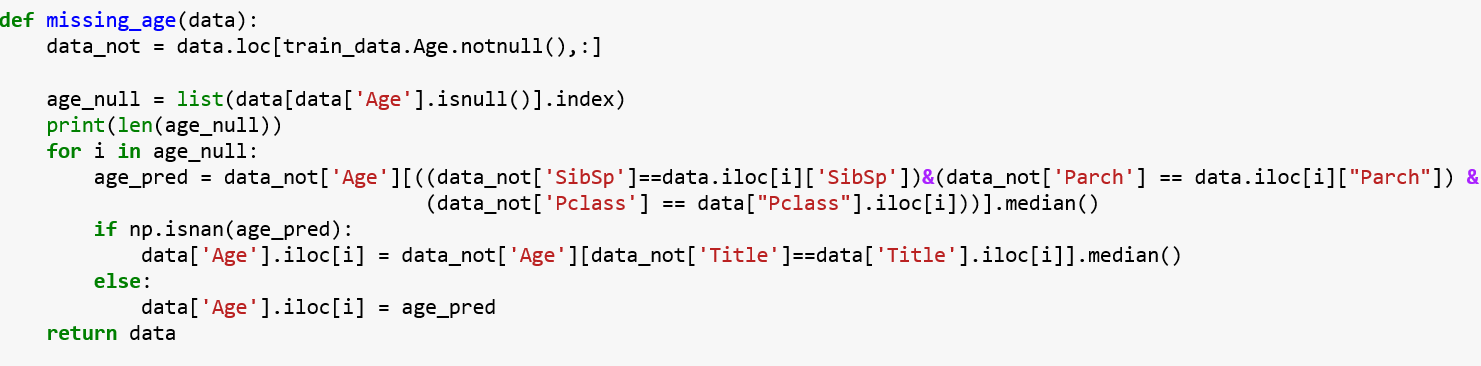
**“Age” attribute:**





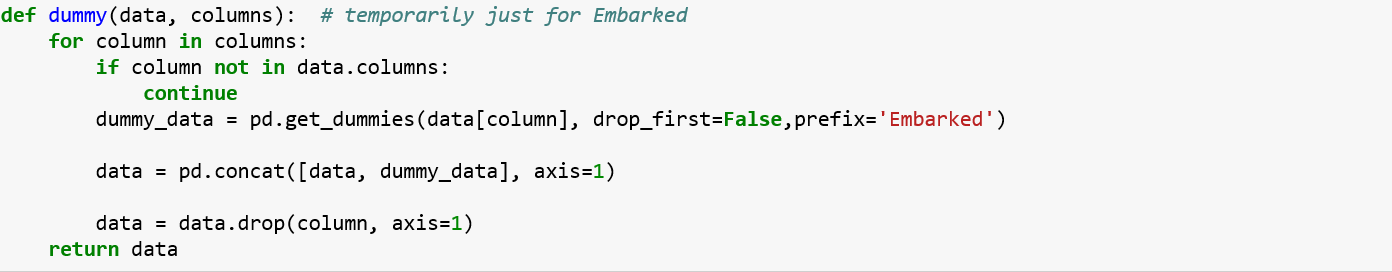
The figure above shows the relationship between “Age” and several related attributes. Attribute “Sex” has less impact on the “Age”. However, 1st class passengers are older than 2nd class passengers who are also older than 3rd class passengers. Moreover, the more a passenger has parents/children the older he is and the more a passenger has siblings/spouses the younger he is.



Based on the correlation graph of “Pclass”, “Parch”, “SibSp” and “Age”, the strategy to fill missing age is to find the median of similar rows according to those attributes which have own “Age” value, if the finding result is empty, then fill those missing age with the new feature “Title” which will be described in the following content. 

## 3.3 Converting

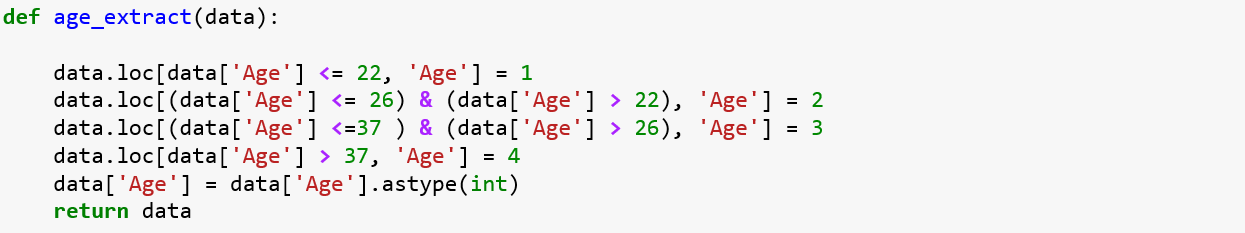
### 3.31 Dummy Variable



The attribute “Embarked” has three categorical values S, Q and C. The function above is to change the categorical values into numerical values by using Dummy.

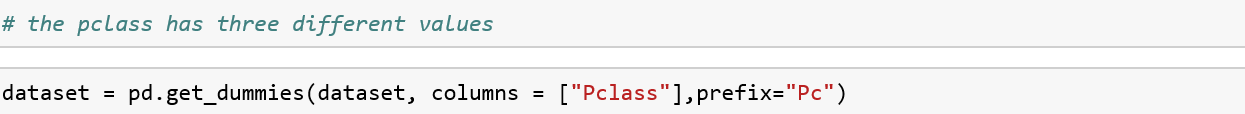
### 3.32 Factorization

**Age**:



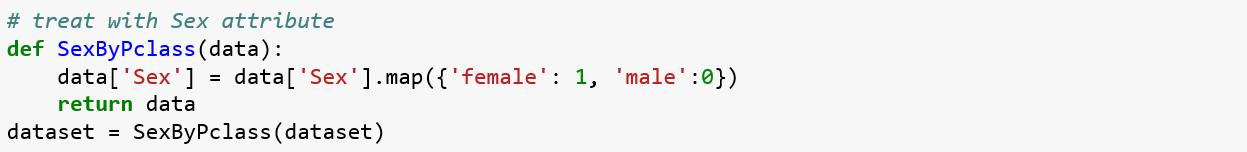
After filling all missing values, the function above changed the “Age” value into four intervals according to the quartile value. This kind of computation largely decrease the complexity.

**Pclass:**



With the same method, the pclass attribute also has three values, we apply the dummy method to deal with this attribute.

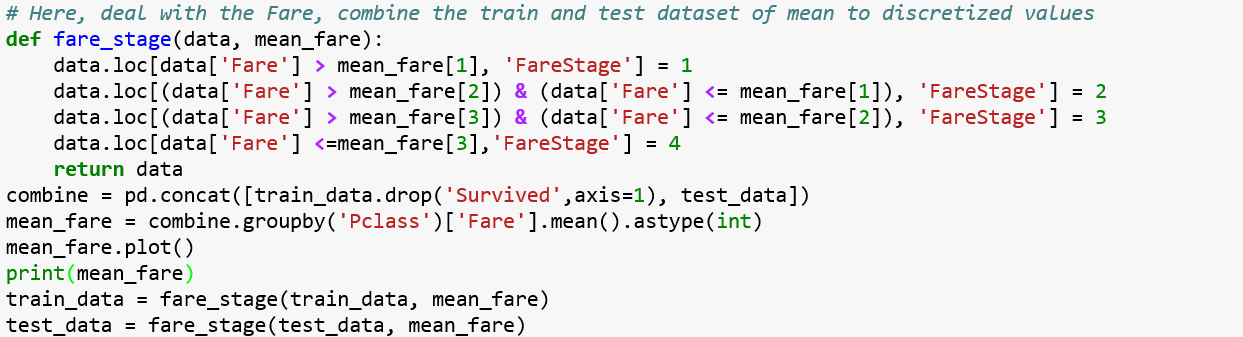
**Sex**:



According to the attribute Sex, there are two categories, male and female, here use the map function to handle this.

## 3.4 Creating

### 3.4.1 Attribute : Fare\_Stage



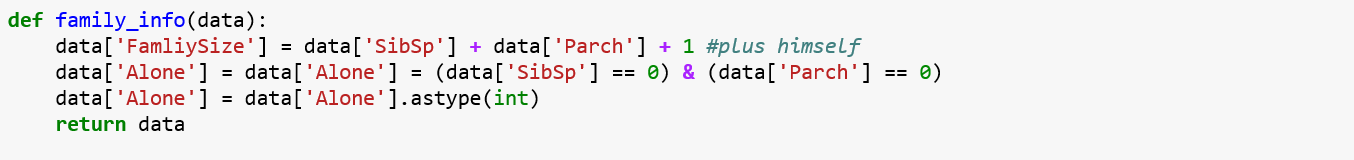
### 3.4.2 Attribute : Title

It is not enough to build model only based on existed attributes, the function above created a new attribute “FareStage” to separate “Fare” into four intervals according to the quartile value.



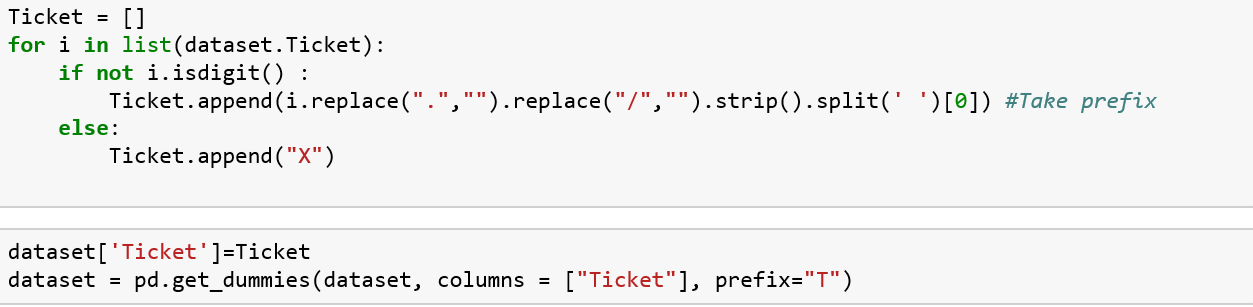
The function above extracted title from “Name” attribute by regular expression to create new attribute “Title”.

### 3.43 Attribute : FamilySize & Alone



Based on the number of “SibSP” and “Parch”, the function above created new attributes “FamilySize” and “Alone”.

### 3.44 Attribute: Ticket



The tickets show that sharing the same prefixes could be booked for cabins placed together. It could therefore lead to the actual placement of the cabins within the ship. In this way, tickets with same prefixes may have a similar class and survival. So that extracting all the tickets’ prefix and replacing the numerical values with “X” seem to be a better method.

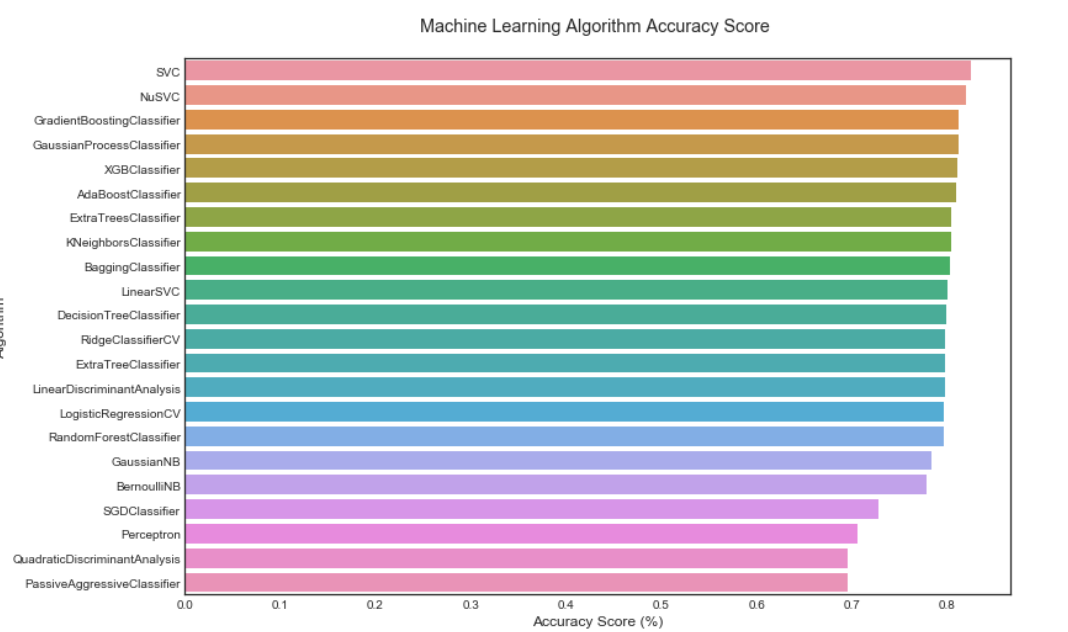
# Modeling

## 4.1 Model Selection

After finishing the data preparation part, the next job is to build the model. In the following part, I will compare twenty-two model’s performance and select some better classifiers to build the final model.

|  | **MLA Name** | **MLA Parameters** | **MLA Train Accuracy Mean** | **MLA Test Accuracy Mean** | **MLA Test Accuracy 3\*STD** | **MLA Time** |
| --- | --- | --- | --- | --- | --- | --- |
| **14** | SVC | {'C': 1.0, 'cache\_size': 200, 'class\_weight': ... | 0.835417 | 0.825283 | 0.03824 | 0.0355513 |
| **15** | NuSVC | {'cache\_size': 200, 'class\_weight': None, 'coe... | 0.828598 | 0.819623 | 0.0350029 | 0.047182 |
| **3** | GradientBoostingClassifier | {'criterion': 'friedman\_mse', 'init': None, 'l... | 0.87178 | 0.812075 | 0.0374784 | 0.0836143 |
| **5** | GaussianProcessClassifier | {'copy\_X\_train': True, 'kernel': None, 'max\_it... | 0.870455 | 0.811698 | 0.030566 | 0.225155 |
| **21** | XGBClassifier | {'base\_score': 0.5, 'booster': 'gbtree', 'cols... | 0.858523 | 0.810566 | 0.0452264 | 0.0401805 |
| **0** | AdaBoostClassifier | {'algorithm': 'SAMME.R', 'base\_estimator': Non... | 0.828788 | 0.809811 | 0.025415 | 0.0682071 |
| **2** | ExtraTreesClassifier | {'bootstrap': False, 'class\_weight': None, 'cr... | 0.897917 | 0.804528 | 0.0302923 | 0.0145634 |
| **13** | KNeighborsClassifier | {'algorithm': 'auto', 'leaf\_size': 30, 'metric... | 0.850379 | 0.803774 | 0.0501191 | 0.00250652 |
| **1** | BaggingClassifier | {'base\_estimator': None, 'bootstrap': True, 'b... | 0.893182 | 0.803019 | 0.0394768 | 0.0171609 |
| **16** | LinearSVC | {'C': 1.0, 'class\_weight': None, 'dual': True,... | 0.813258 | 0.8 | 0.0447134 | 0.0357076 |
| **17** | DecisionTreeClassifier | {'class\_weight': None, 'criterion': 'gini', 'm... | 0.897917 | 0.799623 | 0.0445555 | 0.00180109 |
| **8** | RidgeClassifierCV | {'alphas': (0.1, 1.0, 10.0), 'class\_weight': N... | 0.810038 | 0.798113 | 0.0412082 | 0.00271029 |
| **18** | ExtraTreeClassifier | {'class\_weight': None, 'criterion': 'gini', 'm... | 0.897917 | 0.798113 | 0.054821 | 0.00130076 |
| **19** | LinearDiscriminantAnalysis | {'n\_components': None, 'priors': None, 'shrink... | 0.80928 | 0.798113 | 0.0358888 | 0.00251029 |
| **6** | LogisticRegressionCV | {'Cs': 10, 'class\_weight': None, 'cv': None, '... | 0.813068 | 0.796604 | 0.0412703 | 0.0954502 |
| **4** | RandomForestClassifier | {'bootstrap': True, 'class\_weight': None, 'cri... | 0.89375 | 0.796604 | 0.0245167 | 0.0169049 |
| **12** | GaussianNB | {'priors': None} | 0.793371 | 0.783019 | 0.0386401 | 0.00171077 |
| **11** | BernoulliNB | {'alpha': 1.0, 'binarize': 0.0, 'class\_prior':... | 0.786742 | 0.778491 | 0.0320399 | 0.0019078 |
| **9** | SGDClassifier | {'alpha': 0.0001, 'average': False, 'class\_wei... | 0.727083 | 0.727925 | 0.326323 | 0.00188544 |
| **10** | Perceptron | {'alpha': 0.0001, 'class\_weight': None, 'eta0'... | 0.710227 | 0.70566 | 0.23752 | 0.00210829 |
| **20** | QuadraticDiscriminantAnalysis | {'priors': None, 'reg\_param': 0.0, 'store\_cova... | 0.716856 | 0.695849 | 0.145875 | 0.00230389 |
| **7** | PassiveAggressiveClassifier | {'C': 1.0, 'average': False, 'class\_weight': N... | 0.694508 | 0.695094 | 0.355724 | 0.00169442 |

The table above shows twenty-two classifiers’ performance, the “'MLA Test Accuracy 3\*STD” attribute is the classifier’s test score multiply triple its standard deviation value, which will show how worst the model can be.



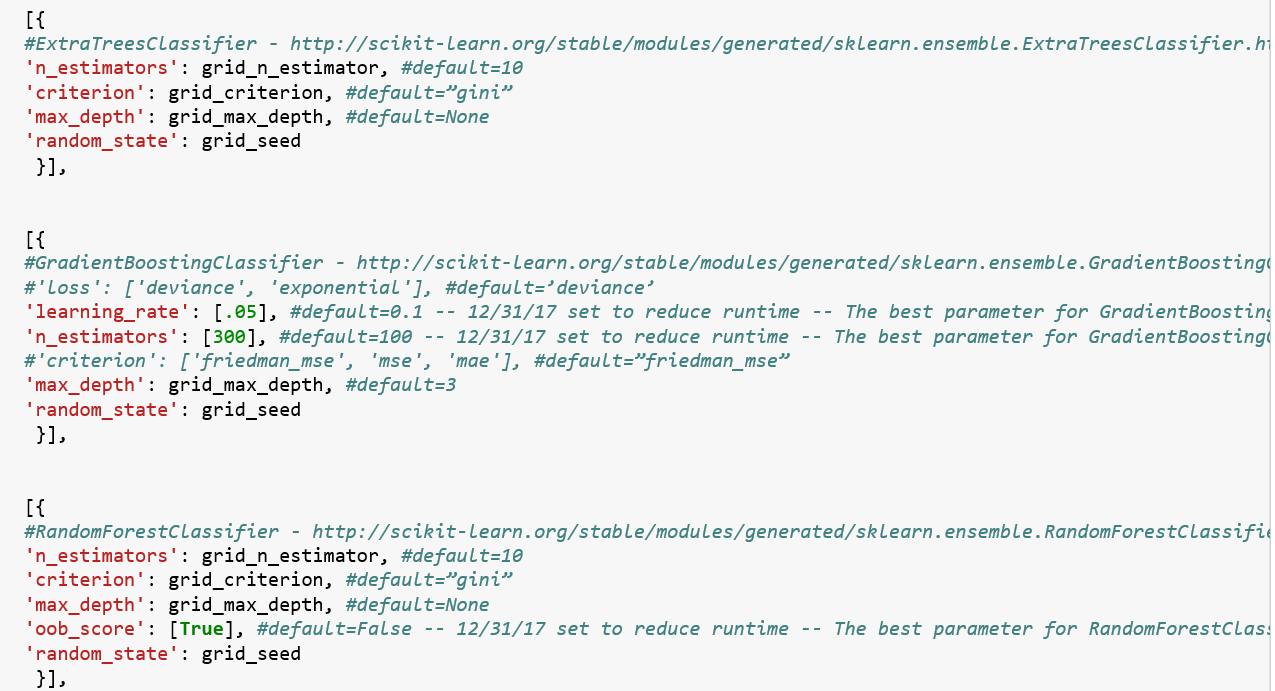
The graph above could clearly represent the performance of different kinds of models. Based on the information above, the final prediction model\_set as below:

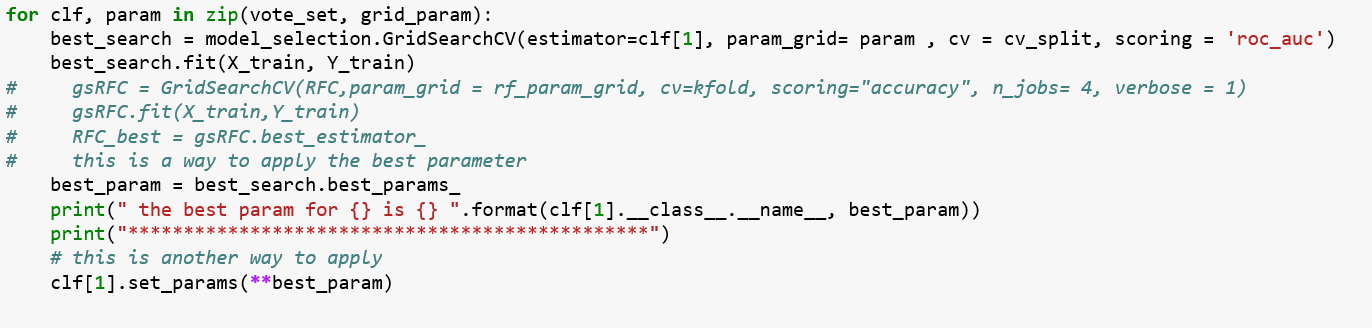


## 4.2 Hyper-Parameter Tuning

Hyper-parameters are parameters that are not directly learnt within estimators. In scikit-learn they are passed as arguments to the constructor of the estimator classes. Typical examples include C, kernel and gamma for Support Vector Classifier, alpha for Lasso, etc. [GridSearchCV](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html#sklearn.model_selection.GridSearchCV) exhaustively considers all parameter combinations, in this way, the parameters of the estimator used to apply these methods are optimized by cross-validated grid-search over a parameter grid. So that It is possible and recommended to search the hyper-parameter space for the best [cross validation](http://scikit-learn.org/stable/modules/cross_validation.html#cross-validation) score by [GridSearchCV](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html#sklearn.model_selection.GridSearchCV) method.

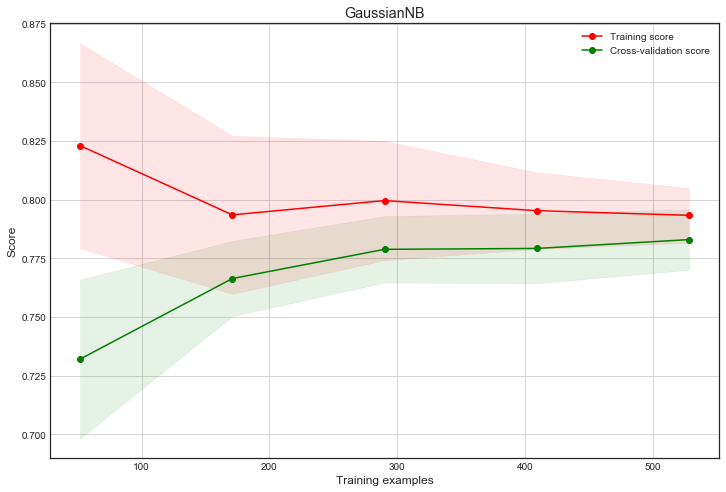
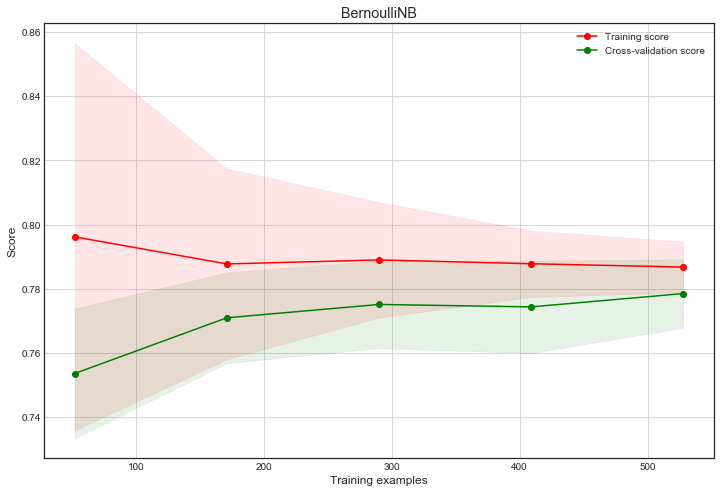
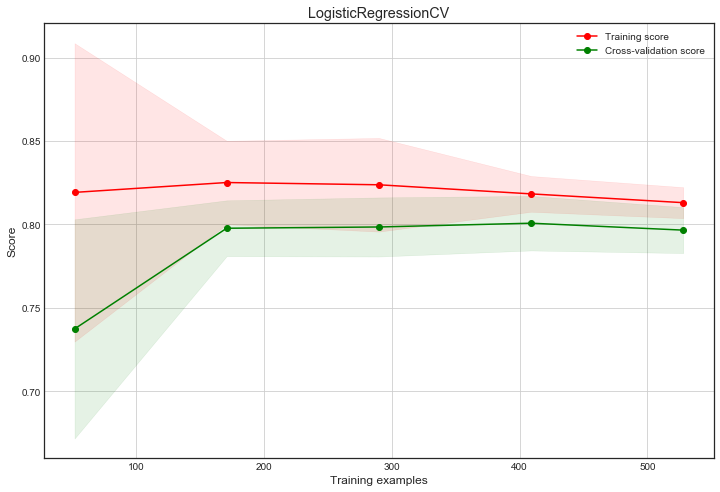
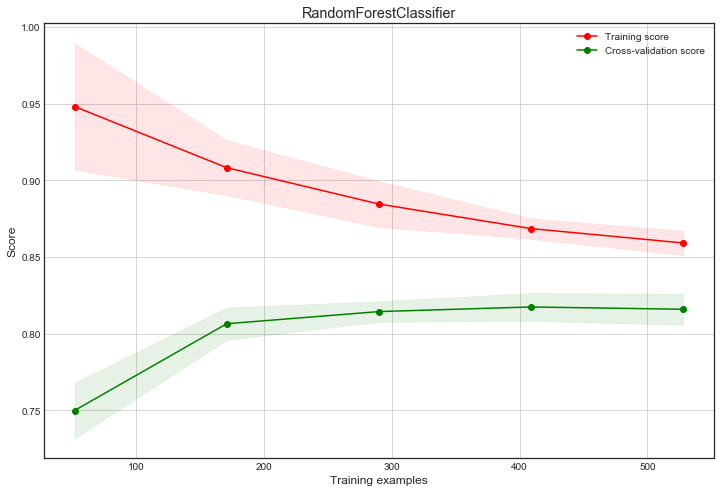
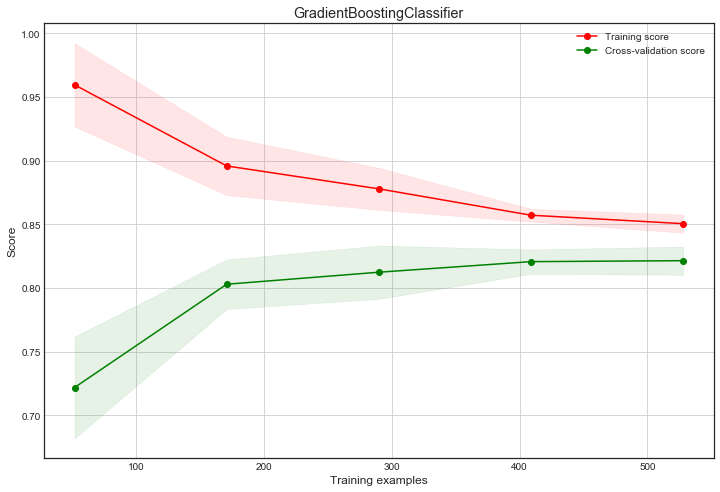
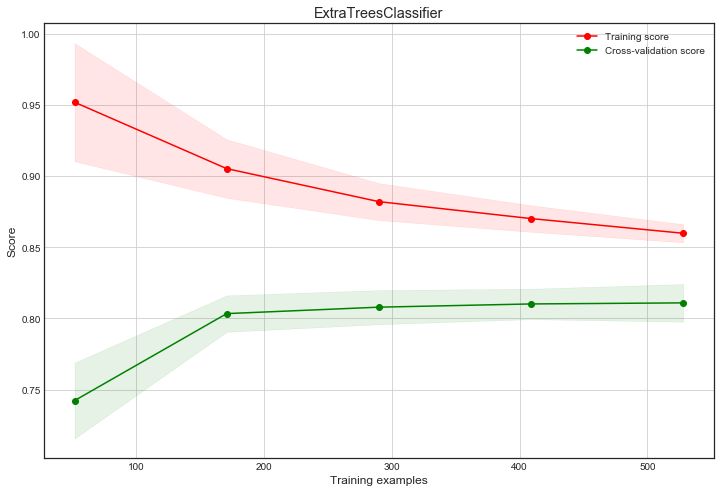
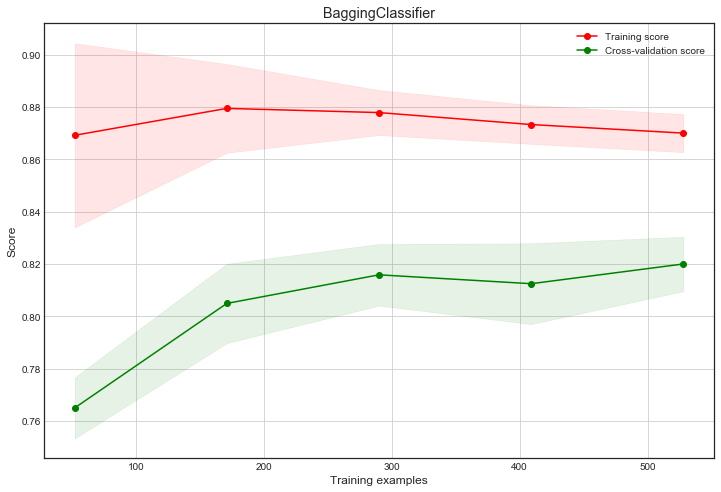
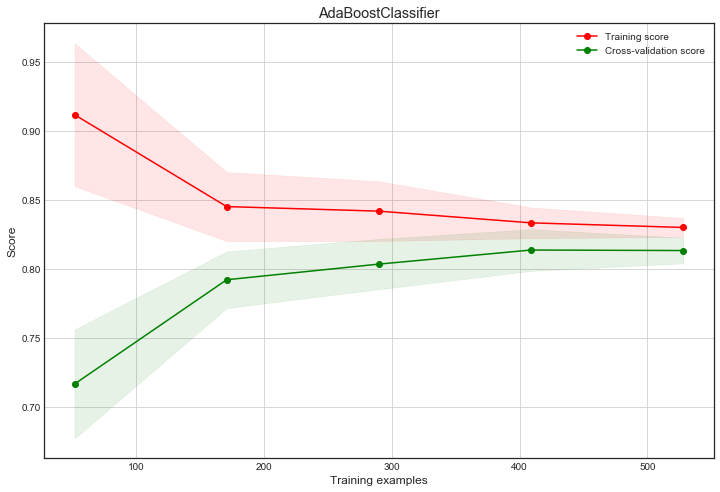
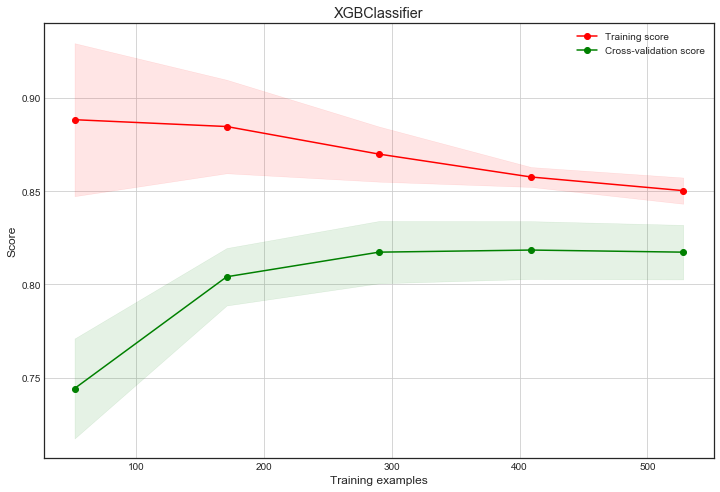
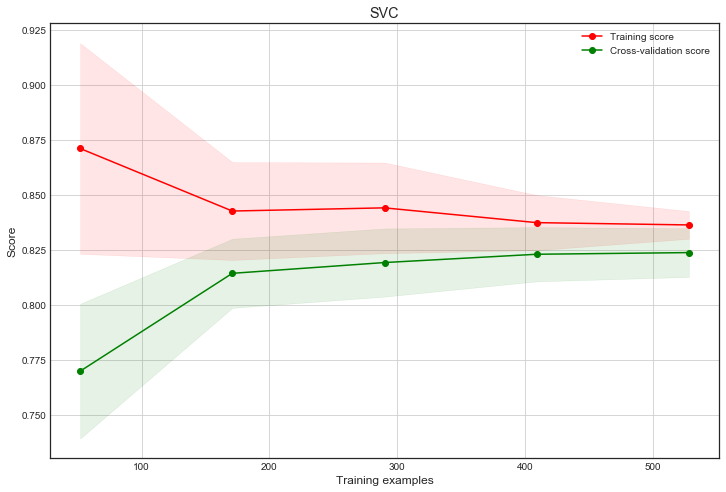
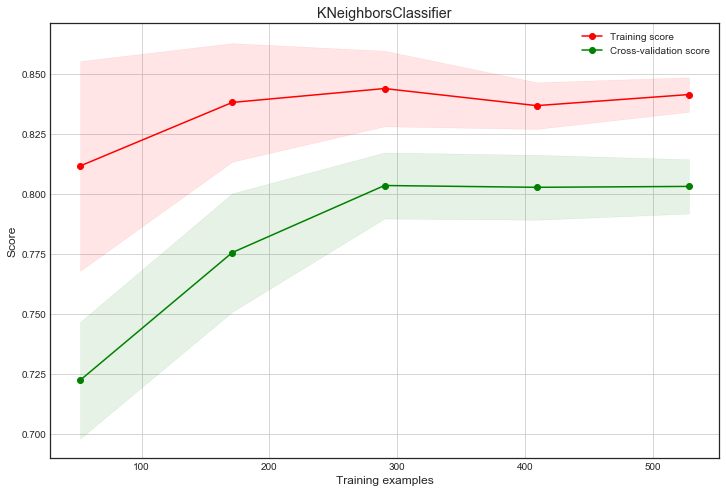






## 4.3 Model Performance

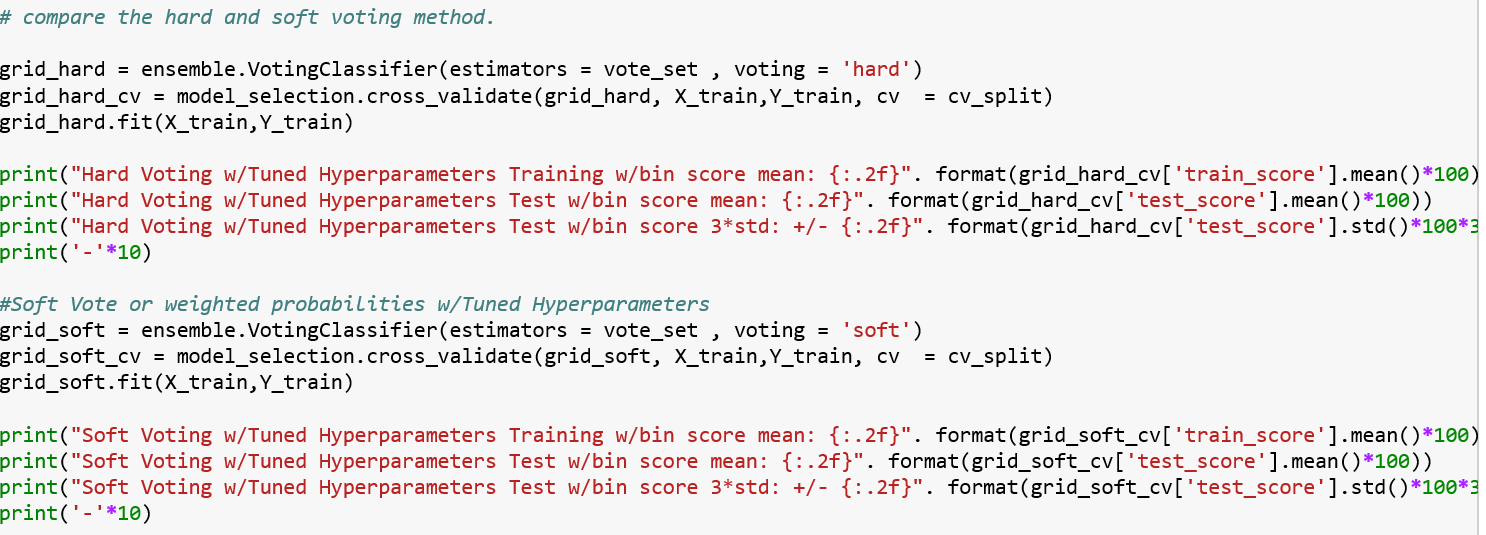
The graph below is the Learning Curve method of every classifiers. Learning curve shows the validation and training score of an estimator for varying numbers of training samples. It is a tool to find out how much we benefit from adding more training data and whether the estimator suffers more from a variance error or a bias error. If both the validation score and the training score converge to a value that is too low with increasing size of the training set, we will not benefit much from more training data.



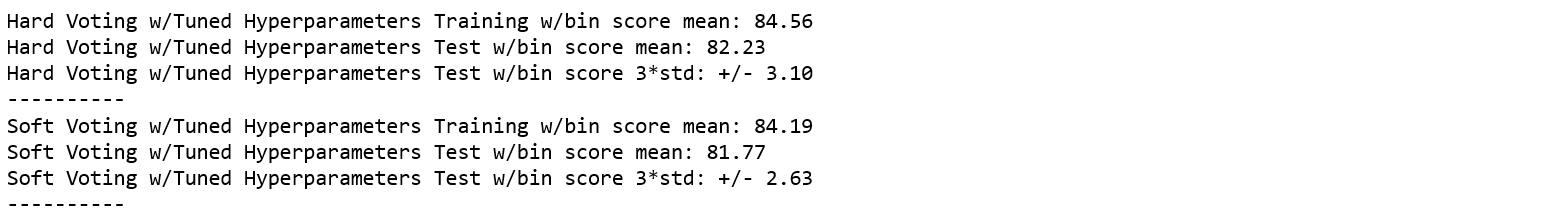
According to Random Forest Classifier, SVC, AdaBoost Classifier, LogisticRegressionCV, BernoulliNB and GaussianNB Classifier, those classifiers converged at some point, which represent the model will gain less benefit from increasing samples. The BenoulliNB and GaussianNB converged at the low score in the graph, which represent the high bias error, show that those model may suffer the underfitting problem.The ExtraTreesClassifier, BaggingClassifier and ExtraTreesClassifier doesn’t converge to a score, and the training score always better than the testing score, which represent those classifiers suffer the high variance problem, this phenomenon shows that those classifiers have overfitting problem.

## 4.4 Ensemble Classifier

Ensemble methods use multiple learning algorithms to obtain better [predictive performance](https://en.wikipedia.org/wiki/Predictive_inference) than could be obtained from any of the constituent learning algorithms alone. Here, the VotingClassifier combines all eleven classifiers together by voting to do the prediction.



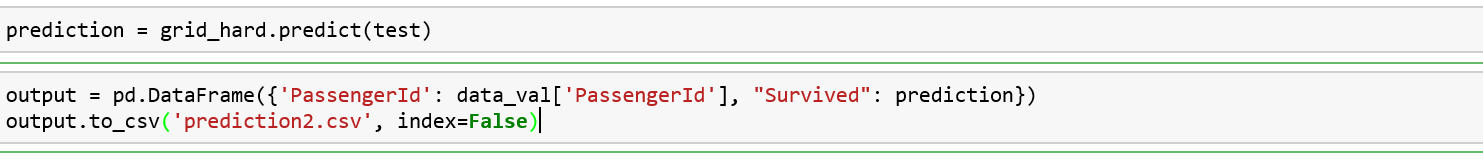
There are two different methods to do the voting, soft method and hard method. If 'hard', uses predicted class labels for majority rule voting. Else if 'soft', predicts the class label based on the argmax of the sums of the predicted probabilities, which is recommended for an ensemble of well-calibrated classifiers.



After comparing two method, the hard voting is better.

## 4.5 Prediction

The final part is to do the prediction by the ensemble classifier, the result is as follow:





# Conclusion

Based on the analysis information above, the attribute “Sex”, “Pclass” and “Title” have more impact on the final target, female, upper class level and big family member have better survived chance. The attribute “Embarked”, “Age” have few impact to the survival rate, which means those attributes are not determinant element towards final survival rate.