**Titanic: predict survival from disaster**

Rao yu - online student

Xiaochang Liu - in-class student

Ji Qi - online student

***Introduction***

RMS Titanic was a British passenger liner that sank in the North Atlantic Ocean in the early morning hours of 15 April 1912, after it collided with an iceberg during its maiden voyage from Southampton to New York City. There were an estimated 2,224 passengers and crew aboard the ship, and more than 1,500 died, making it one of the deadliest commercial peacetime maritime disasters in modern history.One of the reasons that the shipwreck led to such loss of life was that there were not enough lifeboats for the passengers and crew. Although there was some element of luck involved in surviving the sinking, some groups of people were more likely to survive than others, such as women, children, and the upper-class. Our goal is to predict if a passenger survived from the sinking of the Titanic or not and to learn from disaster.

***Data***

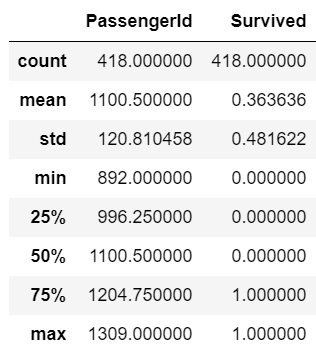
For this project, the Titanic dataset was chosen as a Kaggle competition.This dataset is used in Kaggle prediction competitions. The dataset has been split into 2 groups:the training set contains 891 records, the testing set contains 418 records.

Dataset can be found: <https://www.kaggle.com/c/titanic/data>

Overview of training data:

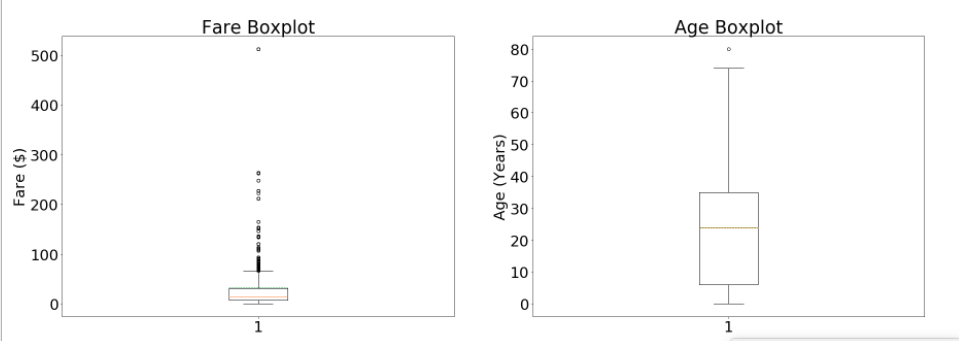


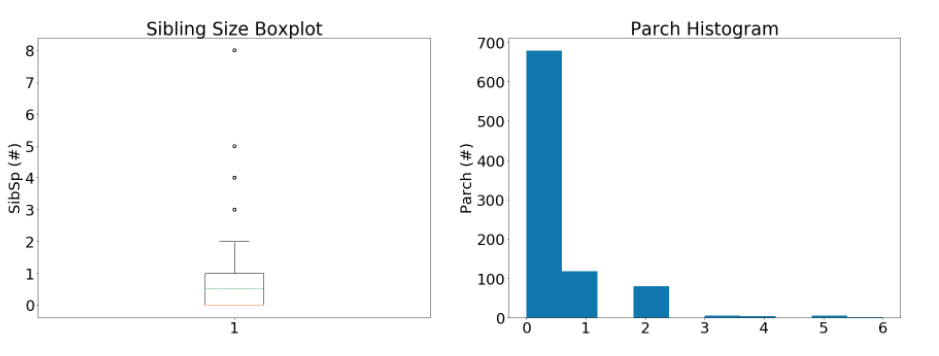
Overview of testing data:

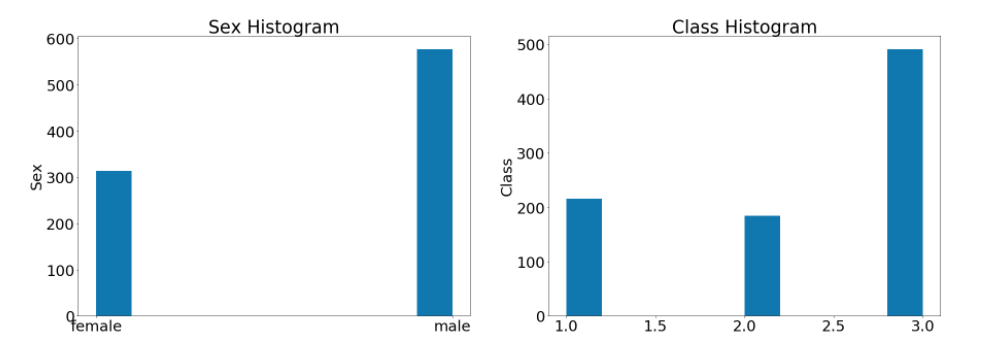


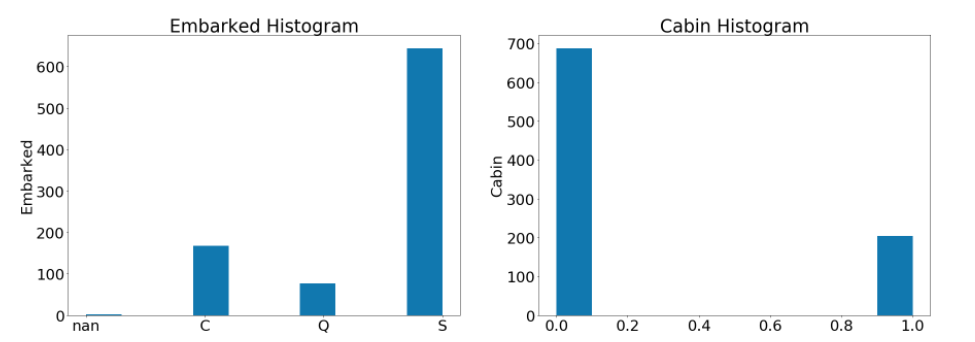
***Data exploration***

This dataset contains both metric and categorical variables, Appendix A, Table 1 lists all variables and their data types.From the histogram and boxplot of the dataset (891 records). Most of information of passengers can be exploited like time back to the accident. For the passenger on the boat, their age range is from 0 to 80. 30% of them are female, and 70% are male. More than 600 of them embarked from Southampton, rest of them embarked maybe from Cherbourg, or Queenstown. Half of passenger bought the class 3 ticket, 200 of rest people got class 2 ticket, others from class 1. The shape of Fare variable is skewed right and range from 0 to 500, but some passengers who spent $50 got a class 1 ticket, some passengers have cost $67 just for getting a class 3 ticket. The distribution of the Fare variable for class 1 is from $0 to $500, for class 2 and class 3 are both range from $0 to $70. After the sinking, less than ⅓ of people survived from the disaster.



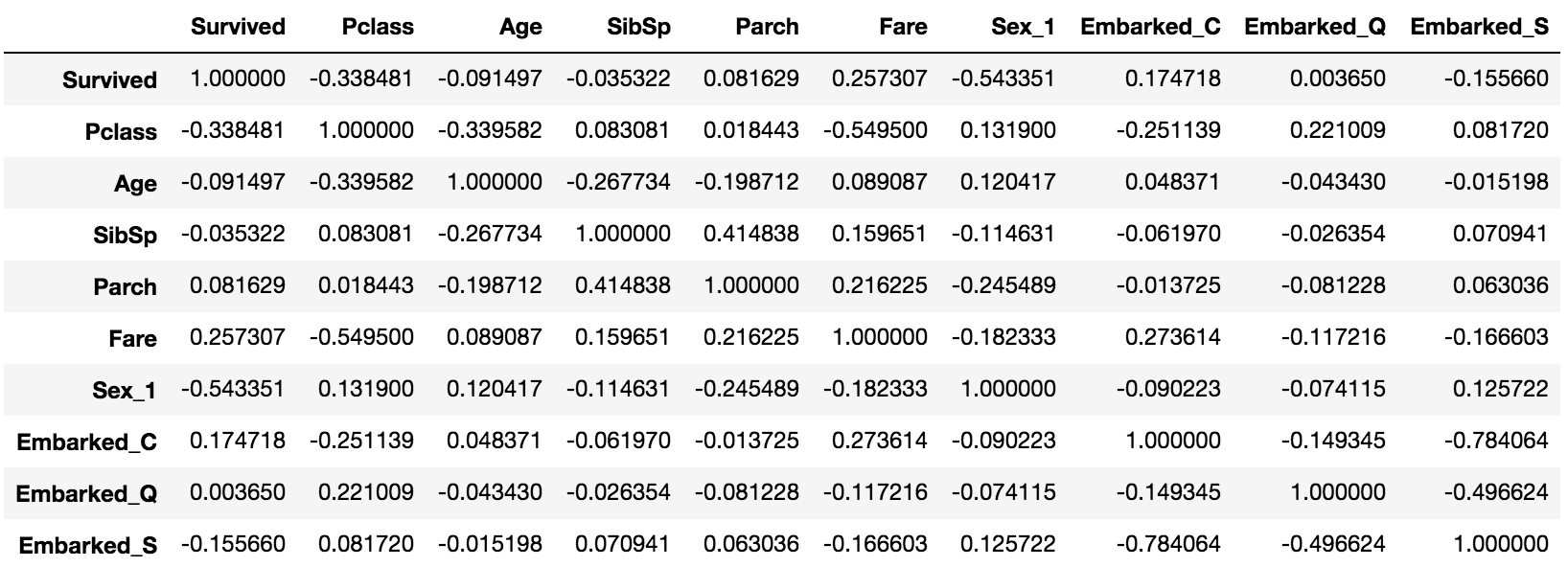


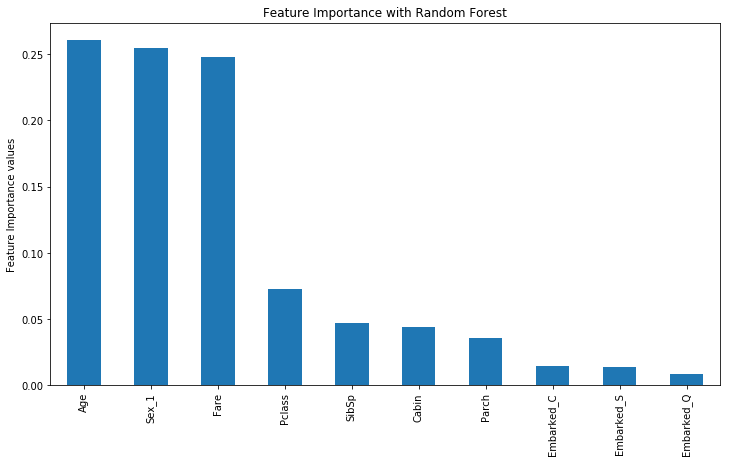




According to correlation matrix, the Fare and Pclass has the highest correlation -0.55 and Parch and SibSp has the second highest correlation: 0.41. The conclusion can be drawn that the survived rate are seem to relate to the sex, Pclass and Fare and with the increase of age, the possibility of survival are reduced if holding other variables constant. With the help of the technology,random forest complete the feature selection work to form the rank of the variables. From the plot, the importance of variable age, sex and fare can also be notice.

The correlation of training set:

Feature selection:

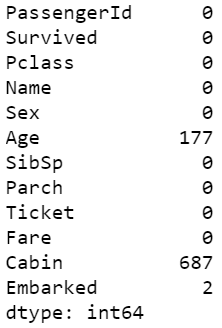
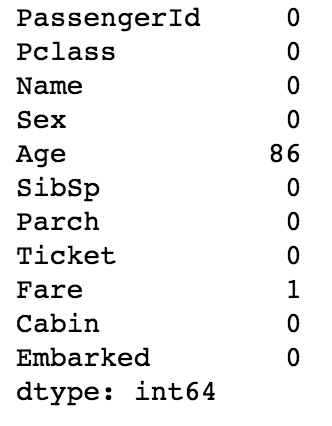


***Data preprocessing***

Some variables were not useful, such as the passenger ID variables, which cannot use in model. Other variables contained less information, such as Name,Ticket. Cabin was interesting, but, more useful when formatted as a category variable than as a other form of ticket number and was therefore converted the cabin number to 1 if there are number in column cabin.

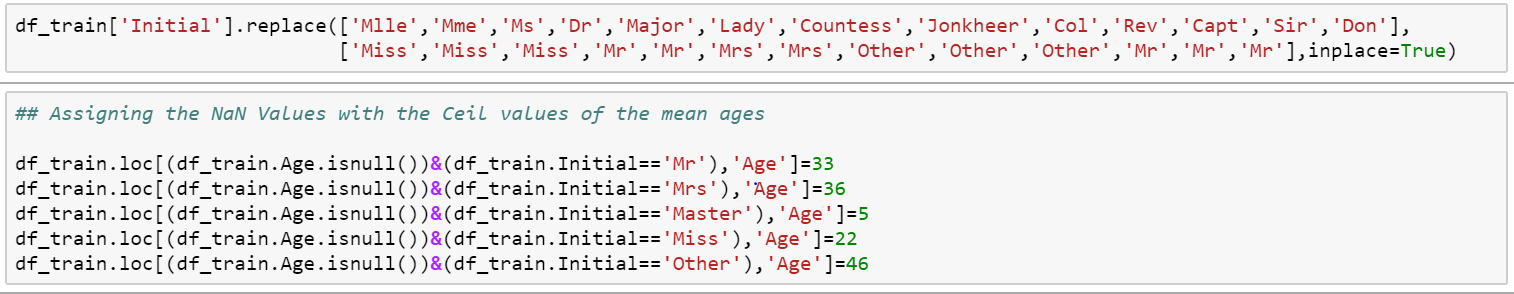
* ***Filling nan variables***

Rather than just imputing the median, we can make a more detailed analysis on the missing values to establish a better value. Age, Cabin and embarked these three variables have missing values in the training set. Age and fare have missing value in testing set.177 passengers are missing the age in train data, 88 are missing in testing set. Almost 70% passengers without cabin, but bother are very important variable in some model. Thus, that variables were handled by several ways:

* ***Filling Age based on the title***

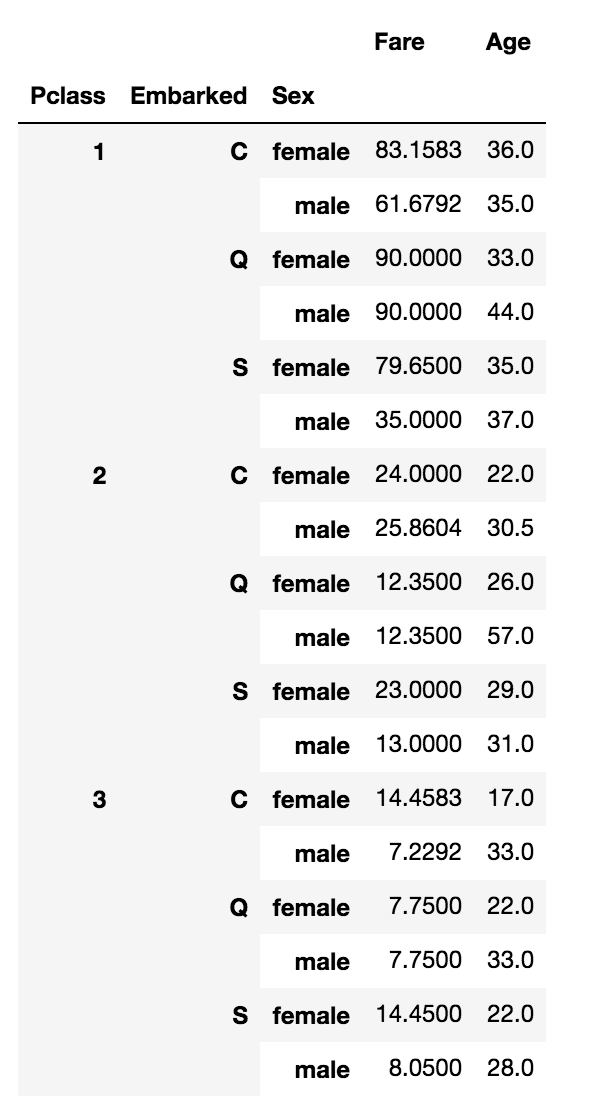
For age variable, it can be provided a probably number based on their title which is included in the passengers’ names. The passengers’ has been assign age as 33 if their title are Mr, as 36 if their title are Mrs, as 5 if they are master, as 22 if they are Miss and as 46 if they are Other.

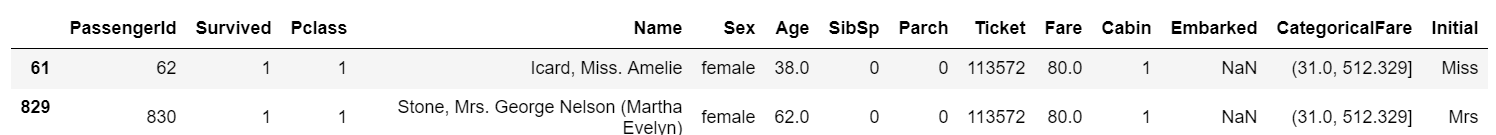


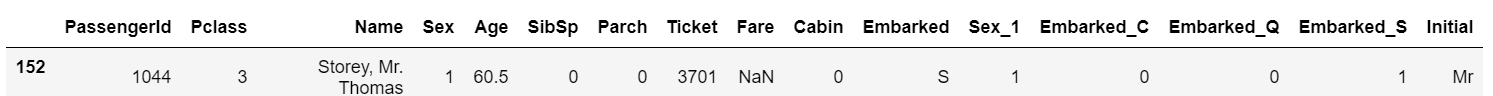
* ***Embarked and Fare in testing set according to the median of different groups***

For Embarked:Embarked has two missing value, those null values are assigned “C” because the medians of embared port has been organized by different class and sex, the closest match to our passengers is C. In 1st class, the median Age is the same age of Miss Amelie and the Fare is similar to the one paid by both.

Then apply the same step to the testing data, there is a null value in the Fare variable, and the median Fare variable in the testing data will be the good choice to assigned.The median has been organized by different class and embarked and sex, and it presents the 7.98750 is the best number to be assigned.

****





* ***Cabin***

It is important to mention that analyzation parts will be separated by “including Cabin variable” and “exclude Cabin variable” since there are 687 missing value and there is no other resource to investigate the more detail of the variable and how to fill in the missing value.

Currently, it will be assigned 1 if it assumed that passenger has a Cabin number, otherwise it will be assigned 0.

* ***Create dummy variables and splitting data***

In order to use categorical variables in the model, Dummy Variables were created for Sex, Embarked and Cabin. Data has been split for training set and testing set for predict which passenger survived in the sank. 68.1% data to train the model, 34% data to test the model.

**Methodology**

Logistic regression, decision tree, KNN, SVM and random forest these five methodologies will be applied to the training data set and different algorithms will be the result, then the algorithms will be applied to the testing data to get the output which will be used later as the comparing. As mentioned before, all of the methodologies will perform two outputs, include or exclude the Cabin variable. The output will include accuracy, f1, npv, ppv, tnr, tpr. There are explanation of what these four acronyms mean.

* ***Logistic regression***

The goal of logistic regression is to find the best fitting model to describe the relationship between the dichotomous characteristic of interest and a set of independent variables. In our model survival or not survival is the dichotomous characteristic of interest, 10 independent variables are applied to model for prediction.

* ***Decision tree***

Decision Tree Classifier, repetitively divides the working area(plot) into sub part by identifying lines. (repetitively because there may be two distant regions of same class divided by other).

Dividing efficiently based on maximum information gain is key to decision tree classifier. In our training data, the depth of tree has been changed for finding a fitting model.

* ***KNN***

Predictions are made for a new instance (x) by searching through the entire training set for the K most similar instances (the neighbors) and summarizing the output variable for those K instances. then,Classification is computed from a simple majority vote of the nearest neighbors of K points. In our model, the K has been changed from 1 to 10 to find a fitting model with high accuracy and avoiding overfitting problem.

* ***SVM***

an SVM algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier.It attempts to find a hyperplane that can separate two class of data by the largest margin.

* ***Random Forest***

Random forest is an ensemble algorithm which builds many trees using a subset of the available input variables and their values, it inherently contains some underlying decision trees that omit the noise generating feature. In the end, when it is time to generate a prediction a vote among all the underlying trees takes place and the majority prediction value wins.

**Performance Evaluation**

* ***Confusion matrix***

F1 - it measures the accuracy of test

Npv - negative predictive value: it is the proportions of negative results in [statistics](https://en.wikipedia.org/wiki/Statistics) and [diagnostic tests](https://en.wikipedia.org/wiki/Diagnostic_test) that are [true positive](https://en.wikipedia.org/wiki/True_positive) and [true negative](https://en.wikipedia.org/wiki/True_negative) results

Ppv - positive predictive value: it is the proportions of positive results in [statistics](https://en.wikipedia.org/wiki/Statistics) and [diagnostic tests](https://en.wikipedia.org/wiki/Diagnostic_test) that are [true positive](https://en.wikipedia.org/wiki/True_positive) and [true negative](https://en.wikipedia.org/wiki/True_negative) results

Tnr - true negative rate: it measures the proportion of negatives that are correctly identified

Tpr - true positive rate: it measures the proportion of positives that are correctly identified

A confusion matrix include npv, ppv, tnr and tpr which will be used to describe the performance of the model.

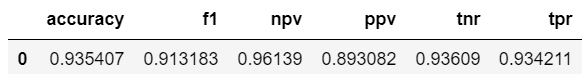
ROC curve -The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings.

**Result**

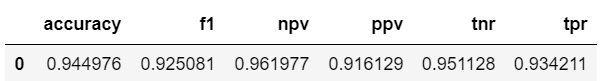
The highest accuracy is from SVM model,100%. Logistic regression and Decision tree also have relatively high accuracy, 94% and 97%. Random forest got 79% accuracy. KNN have bad performance, only 67%

* ***Logistic regression***

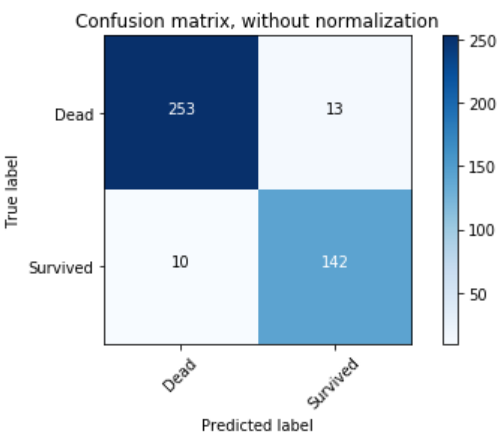
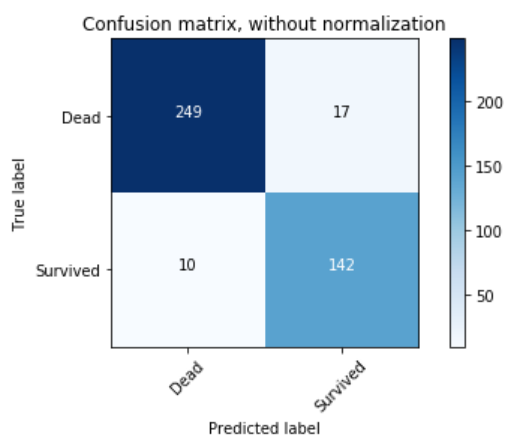
The logistic regression include the Cabin variable:



The logistic regression exclude the Cabin variable:

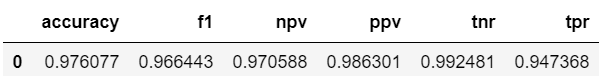


The left one presents the first model and the right one presents the second model:

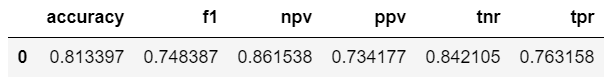


* ***Decision tree***

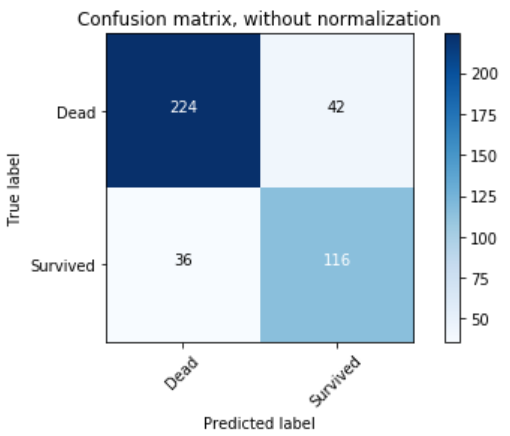
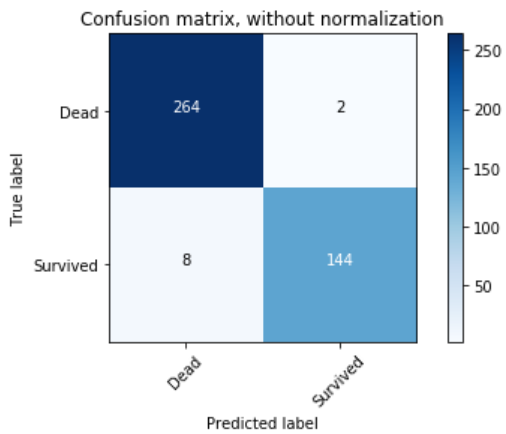
The Decision tree methodology include Cabin variable:



The Decision tree methodology exclude Cabin variable:

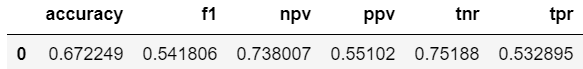


The left one presents the first model and the right one presents the second model:

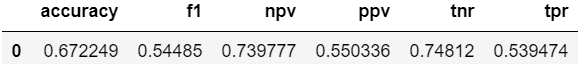


* ***KNN***

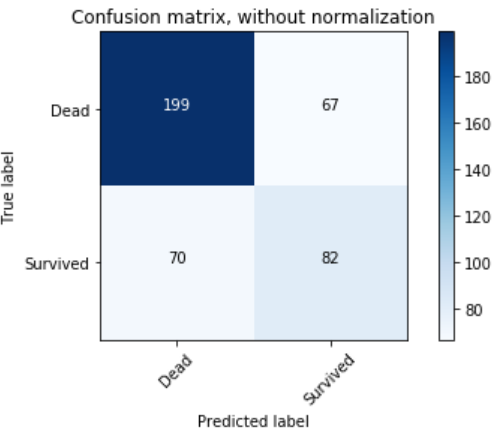
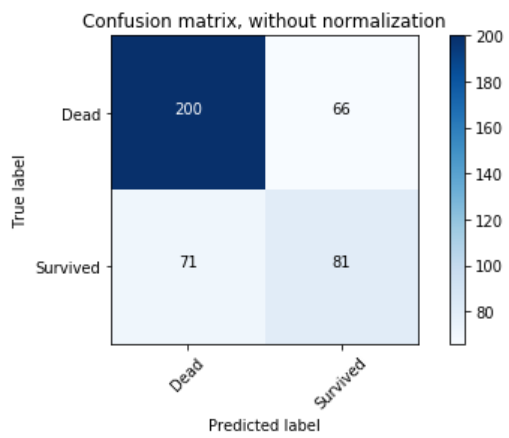
The KNN methodology include the Cabin variable:



The KNN methodology exclude the Cabin variable:

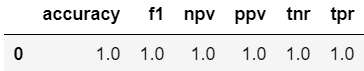
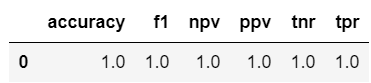


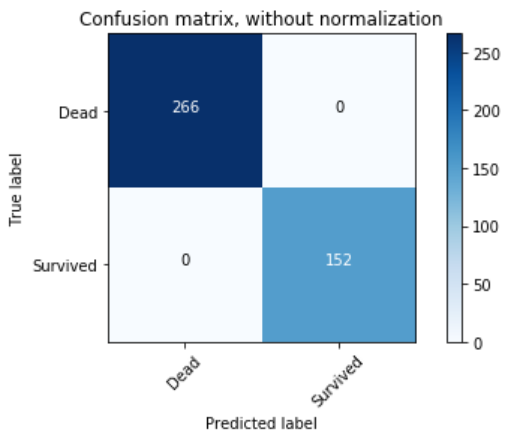
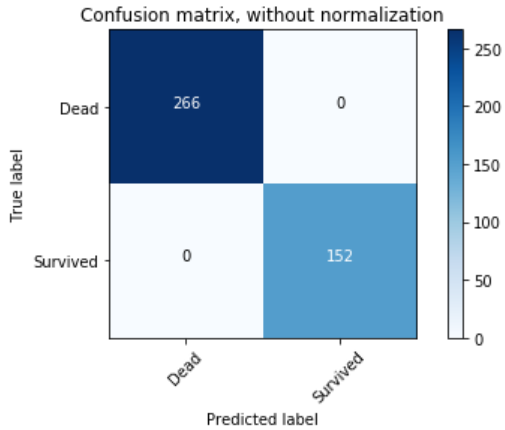
The left one presents the first model and the right one presents the second model:



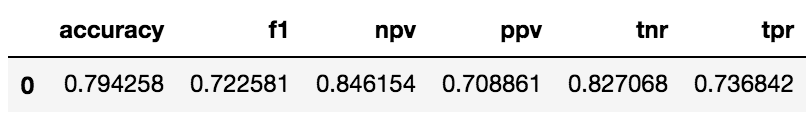
* ***SVM***

The left side is SVM methodology includes the Cabin variable, right side is the excludes one.

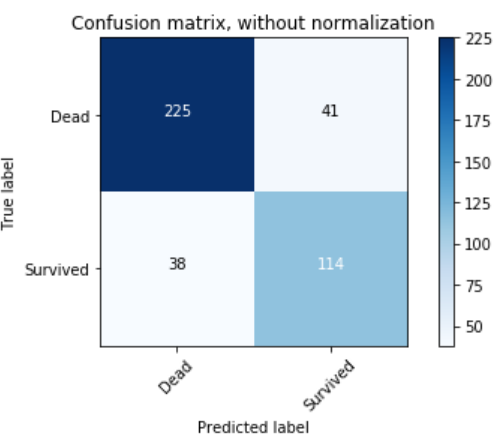
 

* ***Random forest***

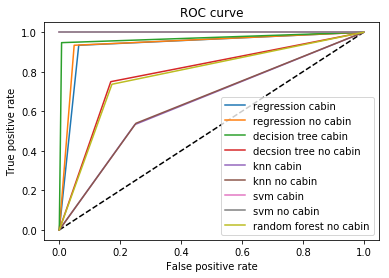
****

Here is the confusion matrix of the model:

****

***Conclusion***

The result is the SVM methodology is the best methodology for its accuracy is 100%, and according to the graph below which performs all methodologies’ ROC curves, it also presents SVM has the highest performance. The most important features are Age, Sex and Fare. From the data, the number of female, especially female in the first class and children who were survived are much more than male, which is correspond to the policy that the women and children are first putted in lifeboat. The male are the last one to be saved when sinking. Unfortunately, The lifeboat are not enough for every passenger, so more than 60% male passengers died in the disaster. However, what we could learn from the result that lifeboat are the key for survival rate. Be sure to prepare enough lifeboats before each trip of ships or boats .

****

**Appendix**

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable** | **Definition** | **Key** | **Data type** |
| survival | Survival | 0 = No, 1 = Yes | metric |
| pclass | Ticket class | 1 = 1st, 2 = 2nd, 3 = 3rd | metric |
| sex | Sex | man=1, woman=0 | String |
| Name | Passenger name |  | String |
| Age | Age in years |  | metric |
| SibSp | # of siblings / spouses aboard the Titanic |  | metric |
| Parch | # of parents / children aboard the Titanic |  | metric |
| Ticket | Ticket number |  | String |
| Fare | Passenger fare |  | metric |
| Cabin | Cabin number |  | String |
| Embarked | Port of Embarkation | C = Cherbourg,  Q = Queenstown,  S = Southampton | String |

Table 1: All variables available

**Python Code:**

import pandas as pd

import csv

import numpy as np

import sklearn

from sklearn.pipeline import Pipeline

from sklearn.preprocessing import StandardScaler, Imputer

from sklearn.neighbors import KNeighborsClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import confusion\_matrix

import pandas as pd

import numpy as np

import sklearn

from sklearn.pipeline import Pipeline

from sklearn.preprocessing import StandardScaler, Imputer

from sklearn.neighbors import KNeighborsClassifier

from sklearn.linear\_model import LogisticRegression, LinearRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn import tree

from sklearn.pipeline import Pipeline

from sklearn.preprocessing import StandardScaler, Imputer

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import mean\_squared\_error

from sklearn.metrics import accuracy\_score

import matplotlib.pyplot as plt

import matplotlib.pylab as plt

get\_ipython().magic(u'matplotlib inline')

from sklearn import datasets

import seaborn as sns

from sklearn.metrics import classification\_report

from sklearn.ensemble import AdaBoostClassifier

from sklearn.ensemble import RandomForestClassifier

import math

# # import data

df\_train=pd.read\_csv('train.csv')

X\_test=pd.read\_csv('test.csv')

y\_test=pd.read\_csv('gender\_submission.csv')

train=pd.read\_csv('train.csv')

# # describe data

X\_test.describe()

y\_test.describe()

plt.hist(df\_train['Survived'])

plt.hist(y\_test['Survived'])

plt.hist(df\_train['Pclass'])

plt.hist(X\_test['Pclass'])

plt.hist(df\_train['Fare'])

plt.hist(df\_train[df\_train['Pclass']==1].Fare)

plt.hist(df\_train[df\_train['Pclass']==2].Fare)

plt.hist(df\_train[df\_train['Pclass']==3].Fare)

df\_train.isnull().sum()

plt.hist(df\_train[df\_train['Sex']=='female'].Survived)

plt.hist(df\_train[df\_train['Sex']=='male'].Survived)

df\_train.corr()

# # fix missing value

#df=df\_train.sort\_values(by=['Pclass','Fare'])

df\_train['CategoricalFare'] = pd.qcut(df\_train['Fare'], 4)

# # fix missing value

#cabin

df\_train.loc[(df\_train.Cabin.isnull()),'Cabin']=0

X\_test.loc[(X\_test.Cabin.isnull()),'Cabin']=0

df\_train.loc[(df\_train['Cabin'] != 0),'Cabin']=1

X\_test.loc[(X\_test['Cabin'] != 0),'Cabin']=1

#age

df\_train['Initial']=0

for i in df\_train:

df\_train['Initial']=df\_train.Name.str.extract('([A-Za-z]+)\.') #lets extract the Salutations

df\_train['Initial'].replace(['Mlle','Mme','Ms','Dr','Major','Lady','Countess','Jonkheer','Col','Rev','Capt','Sir','Don'],['Miss','Miss','Miss','Mr','Mr','Mrs','Mrs','Other','Other','Other','Mr','Mr','Mr'],inplace=True)

## Assigning the NaN Values with the Ceil values of the mean ages

df\_train.loc[(df\_train.Age.isnull())&(df\_train.Initial=='Mr'),'Age']=33

df\_train.loc[(df\_train.Age.isnull())&(df\_train.Initial=='Mrs'),'Age']=36

df\_train.loc[(df\_train.Age.isnull())&(df\_train.Initial=='Master'),'Age']=5

df\_train.loc[(df\_train.Age.isnull())&(df\_train.Initial=='Miss'),'Age']=22

df\_train.loc[(df\_train.Age.isnull())&(df\_train.Initial=='Other'),'Age']=46

df\_train.isnull().sum()

df\_train.groupby(['Pclass', 'Embarked','Sex'])['Fare', 'Age'].median()

df\_train[df\_train["Embarked"].isnull()]

df\_train.loc[(df\_train.Embarked.isnull()),'Embarked']="C"

df\_train['Sex']=(df\_train['Sex'] =='male')\*1

X\_test['Sex']=(X\_test['Sex'] =='male')\*1

X\_test.isnull().sum()

df\_train.Embarked.value\_counts()

df\_train.Sex.value\_counts()

# # creat dummy variables

Embarked\_d=pd.get\_dummies(df\_train.Embarked,prefix='Embarked')#.iloc[:,1:]

Sex\_d=pd.get\_dummies(df\_train.Sex,prefix='Sex').iloc[:,1:]

df\_train=pd.concat([df\_train,Sex\_d,Embarked\_d],axis=1)

Embarked\_dt=pd.get\_dummies(X\_test.Embarked,prefix='Embarked')#.iloc[:,1:]

Sex\_dt=pd.get\_dummies(X\_test.Sex,prefix='Sex').iloc[:,1:]

X\_test=pd.concat([X\_test,Sex\_dt,Embarked\_dt],axis=1)

X\_test.columns

df\_train=df\_train[['Survived','Pclass','Age','SibSp','Parch','Fare','Cabin','Sex\_1','Embarked\_C','Embarked\_Q','Embarked\_S']]

# # missing value for testing

#age

X\_test['Initial']=0

for i in X\_test:

X\_test['Initial']=X\_test.Name.str.extract('([A-Za-z]+)\.') #lets extract the Salutations

X\_test['Initial'].replace(['Mlle','Mme','Ms','Dr','Major','Lady','Countess','Jonkheer','Col','Rev','Capt','Sir','Don'],['Miss','Miss','Miss','Mr','Mr','Mrs','Mrs','Other','Other','Other','Mr','Mr','Mr'],inplace=True)

# Assigning the NaN Values with the Ceil values of the mean ages

X\_test.loc[(X\_test.Age.isnull())&(X\_test.Initial=='Mr'),'Age']=33

X\_test.loc[(X\_test.Age.isnull())&(X\_test.Initial=='Mrs'),'Age']=36

X\_test.loc[(X\_test.Age.isnull())&(X\_test.Initial=='Master'),'Age']=5

X\_test.loc[(X\_test.Age.isnull())&(X\_test.Initial=='Miss'),'Age']=22

X\_test.loc[(X\_test.Age.isnull())&(X\_test.Initial=='Other'),'Age']=46

X\_test.isnull().sum()

X\_test.groupby(['Pclass', 'Embarked','Sex'])['Fare', 'Age'].median()

X\_test.loc[(X\_test.Fare.isnull()),'Fare']=7.98750

X\_test=X\_test[['Pclass','Age','SibSp','Parch','Fare','Cabin','Sex\_1','Embarked\_C','Embarked\_Q','Embarked\_S']]

X\_test.to\_csv('X\_test.csv')

y\_test=y\_test['Survived']

# # separate X and y

X\_train, y\_train = df\_train[df\_train.columns.drop('Survived')], df\_train['Survived']

# # model and evaluation part

model = LogisticRegression()

model.fit(X\_train, y\_train)

yl\_pred = model.predict(X\_test)

def evaluation(y\_test, y\_pred):

tn, fp, fn, tp = sklearn.metrics.confusion\_matrix(y\_test, y\_pred).ravel()

tpr = float(tp)/(tp+fn) # recall or true positive rate

tnr = float(tn)/(tn+fp) # true negative rate

ppv= float(tp)/(tp+fp) # precision or positive predictive power

npv= float(tn)/(tn+fn) # negative predictive power

f1= 2.0/(1.0/ppv+1.0/tpr)

a=accuracy\_score(y\_test, y\_pred)

dt=pd.DataFrame({'tpr':[tpr], 'tnr':[tnr],

'ppv':[ppv],

'npv':[npv],

'f1':[f1],

'accuracy':[a],

})

return dt

#yl=pd.DataFrame(yl\_pred)

#yl=yl.to\_csv('lrpredict.csv')

evaluation(y\_test, yl\_pred)

# # ROC

from sklearn import metrics

fpr\_lr, tpr\_lr, thresholds = metrics.roc\_curve(y\_test, yl\_pred)

import itertools

def plot\_confusion\_matrix(cm, classes,

normalize=False,

title='Confusion matrix',

cmap=plt.cm.Blues):

"""

This function prints and plots the confusion matrix.

Normalization can be applied by setting `normalize=True`.

"""

if normalize:

cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

print("Normalized confusion matrix")

else:

print('Confusion matrix, without normalization')

print(cm)

plt.imshow(cm, interpolation='nearest', cmap=cmap)

plt.title(title)

plt.colorbar()

tick\_marks = np.arange(len(classes))

plt.xticks(tick\_marks, classes, rotation=45)

plt.yticks(tick\_marks, classes)

fmt = '.2f' if normalize else 'd'

thresh = cm.max() / 2.

for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):

plt.text(j, i, format(cm[i, j], fmt),

horizontalalignment="center",

color="white" if cm[i, j] > thresh else "black")

plt.tight\_layout()

plt.ylabel('True label')

plt.xlabel('Predicted label')

# Compute confusion matrix

cnf\_matrix = metrics.confusion\_matrix(y\_test, yl\_pred)

np.set\_printoptions(precision=2)

class\_names = ['Dead', 'Survived']

# Plot non-normalized confusion matrix

plt.figure()

plot\_confusion\_matrix(cnf\_matrix, classes=class\_names,

title='Confusion matrix, without normalization')

# Plot normalized confusion matrix

plt.figure()

plot\_confusion\_matrix(cnf\_matrix, classes=class\_names, normalize=True,

title='Normalized confusion matrix')

# # no cabin

df1\_train=df\_train[['Survived','Pclass','Age','SibSp','Parch','Fare','Sex\_1','Embarked\_C','Embarked\_Q','Embarked\_S']]

X1\_test=X\_test[['Pclass','Age','SibSp','Parch','Fare','Sex\_1','Embarked\_C','Embarked\_Q','Embarked\_S']]

X1\_train, y\_train = df1\_train[df1\_train.columns.drop('Survived')], df1\_train['Survived']

#LogisticRegression without cabin

model = LogisticRegression()

model.fit(X1\_train, y\_train)

ylnc\_pred = model.predict(X1\_test)

evaluation(y\_test, ylnc\_pred)

# Compute confusion matrix

cnf\_matrix = metrics.confusion\_matrix(y\_test, ylnc\_pred)

np.set\_printoptions(precision=2)

class\_names = ['Dead', 'Survived']

# Plot non-normalized confusion matrix

plt.figure()

plot\_confusion\_matrix(cnf\_matrix, classes=class\_names,

title='Confusion matrix, without normalization')

# Plot normalized confusion matrix

plt.figure()

plot\_confusion\_matrix(cnf\_matrix, classes=class\_names, normalize=True,

title='Normalized confusion matrix')

from sklearn import metrics

fpr\_lnc, tpr\_lnc, thresholds = metrics.roc\_curve(y\_test, ylnc\_pred)

#logistic regression ROC

plt.figure(1)

plt.plot([0, 1], [0, 1], 'k--')

plt.plot(fpr\_lr, tpr\_lr, label='cable logst')

plt.plot(fpr\_lnc, tpr\_lnc, label='no cable logst')

plt.xlabel('False positive rate')

plt.ylabel('True positive rate')

plt.title('ROC curve')

plt.legend(loc='best')

plt.show()

# # Decsion tree

#decsion tree with cabin

modeldt=DecisionTreeClassifier(max\_depth=3)

modeldt.fit(X\_train,y\_train)

yd\_pred = modeldt.predict(X\_test)

evaluation(y\_test, yd\_pred)

# Compute confusion matrix

cnf\_matrix = metrics.confusion\_matrix(y\_test, yd\_pred)

np.set\_printoptions(precision=2)

class\_names = ['Dead', 'Survived']

# Plot non-normalized confusion matrix

plt.figure()

plot\_confusion\_matrix(cnf\_matrix, classes=class\_names,

title='Confusion matrix, without normalization')

# Plot normalized confusion matrix

plt.figure()

plot\_confusion\_matrix(cnf\_matrix, classes=class\_names, normalize=True,

title='Normalized confusion matrix')

#decsion tree without cabin

model=DecisionTreeClassifier()

model.fit(X1\_train,y\_train)

ydnc\_pred = model.predict(X1\_test)

evaluation(y\_test, ydnc\_pred)

# Compute confusion matrix

cnf\_matrix = metrics.confusion\_matrix(y\_test, ydnc\_pred)

np.set\_printoptions(precision=2)

class\_names = ['Dead', 'Survived']

# Plot non-normalized confusion matrix

plt.figure()

plot\_confusion\_matrix(cnf\_matrix, classes=class\_names,

title='Confusion matrix, without normalization')

# Plot normalized confusion matrix

plt.figure()

plot\_confusion\_matrix(cnf\_matrix, classes=class\_names, normalize=True,

title='Normalized confusion matrix')

# # KNN

#KNN have cabin

model=KNeighborsClassifier(n\_neighbors=5)

model.fit(X\_train,y\_train)

yk\_pred = model.predict(X\_test)

evaluation(y\_test, yk\_pred)

# Compute confusion matrix

cnf\_matrix = metrics.confusion\_matrix(y\_test, yk\_pred)

np.set\_printoptions(precision=2)

class\_names = ['Dead', 'Survived']

# Plot non-normalized confusion matrix

plt.figure()

plot\_confusion\_matrix(cnf\_matrix, classes=class\_names,

title='Confusion matrix, without normalization')

# Plot normalized confusion matrix

plt.figure()

plot\_confusion\_matrix(cnf\_matrix, classes=class\_names, normalize=True,

title='Normalized confusion matrix')

#KNN without cabin

model=KNeighborsClassifier()

model.fit(X1\_train,y\_train)

yknc\_pred = model.predict(X1\_test)

evaluation(y\_test, yknc\_pred)

# Compute confusion matrix

cnf\_matrix = metrics.confusion\_matrix(y\_test, yknc\_pred)

np.set\_printoptions(precision=2)

class\_names = ['Dead', 'Survived']

# Plot non-normalized confusion matrix

plt.figure()

plot\_confusion\_matrix(cnf\_matrix, classes=class\_names,

title='Confusion matrix, without normalization')

# Plot normalized confusion matrix

plt.figure()

plot\_confusion\_matrix(cnf\_matrix, classes=class\_names, normalize=True,

title='Normalized confusion matrix')

# # SVM

#svm linear without cabin

from sklearn import svm

model = svm.SVC(kernel='linear')

model.fit(X1\_train,y\_train)

ysnc\_pred = model.predict(X1\_test)

evaluation(y\_test, ysnc\_pred)

# Compute confusion matrix

cnf\_matrix = metrics.confusion\_matrix(y\_test, ysnc\_pred)

np.set\_printoptions(precision=2)

class\_names = ['Dead', 'Survived']

# Plot non-normalized confusion matrix

plt.figure()

plot\_confusion\_matrix(cnf\_matrix, classes=class\_names,

title='Confusion matrix, without normalization')

# Plot normalized confusion matrix

plt.figure()

plot\_confusion\_matrix(cnf\_matrix, classes=class\_names, normalize=True,

title='Normalized confusion matrix')

#SVM linear with cabin

from sklearn import svm

model = svm.SVC(kernel='linear')

model.fit(X\_train,y\_train)

ys\_pred = model.predict(X\_test)

evaluation(y\_test, ys\_pred)

# Compute confusion matrix

cnf\_matrix = metrics.confusion\_matrix(y\_test, ys\_pred)

np.set\_printoptions(precision=2)

class\_names = ['Dead', 'Survived']

# Plot non-normalized confusion matrix

plt.figure()

plot\_confusion\_matrix(cnf\_matrix, classes=class\_names,

title='Confusion matrix, without normalization')

# Plot normalized confusion matrix

plt.figure()

plot\_confusion\_matrix(cnf\_matrix, classes=class\_names, normalize=True,

title='Normalized confusion matrix')

# # random forest

def rfprocess(X\_train, y\_train,X\_test,y\_test,n\_estimators = 50):

randomf= RandomForestClassifier(n\_estimators=k)

randomf.fit(X\_train,y\_train)

yrr\_pred = randomf.predict(X\_train)

a=accuracy\_score(y\_train, yrr\_pred)

trainerror=1-a

yre\_pred = randomf.predict(X\_test)

a2=accuracy\_score(y\_test, yre\_pred)

testerror=1-a2

print(classification\_report(y\_train, yrr\_pred)),(classification\_report(y\_test, yre\_pred))

print 'training accuracy is',a,' ','testing accuracy is',a2

print evaluation(y\_train, yrr\_pred)

print evaluation(y\_test, yre\_pred)

return trainerror,testerror,

fig = plt.figure()

fig.suptitle('classification error of random forest', fontsize=20)

plt.xlabel('number of trees', fontsize=18)

plt.ylabel('classification error', fontsize=16)

fig.savefig('test.jpg')

for k in range(1,200,4):

trainerror,testerror=rfprocess(X\_train, y\_train,X\_test,y\_test,n\_estimators = k)

# print k,trainerror,testerror

plt.plot(k,trainerror,'co',k, testerror,'y\*-')

plt.legend(['training error','test error'])

rfprocess(X\_train, y\_train,X\_test,y\_test,n\_estimators =45)

randomf= RandomForestClassifier(n\_estimators =45)

randomf.fit(X\_train,y\_train)

yre\_pred = randomf.predict(X\_test)

evaluation(y\_test, yre\_pred)

# Compute confusion matrix

cnf\_matrix = metrics.confusion\_matrix(y\_test, yre\_pred)

np.set\_printoptions(precision=2)

class\_names = ['Dead', 'Survived']

# Plot non-normalized confusion matrix

plt.figure()

plot\_confusion\_matrix(cnf\_matrix, classes=class\_names,

title='Confusion matrix, without normalization')

# Plot normalized confusion matrix

plt.figure()

plot\_confusion\_matrix(cnf\_matrix, classes=class\_names, normalize=True,

title='Normalized confusion matrix')

# # random forest feature selection

randomf= RandomForestClassifier(n\_estimators=45)

randomf.fit(X\_train, y\_train)

yrr\_pred = randomf.predict(X\_train)

a=accuracy\_score(y\_train, yrr\_pred)

print(classification\_report(y\_train, yrr\_pred))

print a

imp\_feat\_rf = pd.Series(randomf.feature\_importances\_, index=X\_train.columns).sort\_values(ascending=False)

imp\_feat\_rf[:64].plot(kind='bar', title='Feature Importance with Random Forest', figsize=(12,8))

plt.ylabel('Feature Importance values')

plt.subplots\_adjust(bottom=0.25)

plt.savefig('FeatImportance.png')

plt.show()

df\_train.corr()

# # ROC all

#logstic regresion

from sklearn import metrics

fpr\_lr, tpr\_lr, thresholds = metrics.roc\_curve(y\_test, yl\_pred)

#dt with cabin

fpr\_dt, tpr\_dt, thresholds = metrics.roc\_curve(y\_test, yd\_pred)

#dtwithout cabin

fpr\_dnc, tpr\_dnc, thresholds = metrics.roc\_curve(y\_test, ydnc\_pred)

#knn with cabin

fpr\_k, tpr\_k, thresholds = metrics.roc\_curve(y\_test, yk\_pred)

#knn without cabin

fpr\_knc, tpr\_knc, thresholds = metrics.roc\_curve(y\_test, yknc\_pred)

#svm with cabin

fpr\_s, tpr\_s, thresholds = metrics.roc\_curve(y\_test, ys\_pred)

#svm without cabin

fpr\_snc, tpr\_snc, thresholds = metrics.roc\_curve(y\_test, ysnc\_pred)

#random forest with cabin

fpr\_rf, tpr\_rf, thresholds = metrics.roc\_curve(y\_test, yre\_pred)

#logistic regression ROC

plt.figure(1)

plt.plot([0, 1], [0, 1], 'k--')

plt.plot(fpr\_lr, tpr\_lr, label='regression cabin ')

plt.plot(fpr\_lnc, tpr\_lnc, label='regression no cabin')

plt.plot(fpr\_dt, tpr\_dt, label='decision tree cabin')

plt.plot(fpr\_dnc, tpr\_dnc, label='decsion tree no cabin')

plt.plot(fpr\_k, tpr\_k, label='knn cabin')

plt.plot(fpr\_knc, tpr\_knc, label='knn no cabin')

plt.plot(fpr\_s, tpr\_s, label='svm cabin')

plt.plot(fpr\_snc, tpr\_snc, label='svm no cabin')

plt.plot(fpr\_rf, tpr\_rf, label='random forest no cabin')

plt.xlabel('False positive rate')

plt.ylabel('True positive rate')

plt.title('ROC curve')

plt.legend(loc='best')

plt.show()