Name: Siyu Wu | USC ID: 2350-0175-97 | ISE 529-E2

Data Cleaning and Feature Engineering

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
# Data Cleaning - Drop specifuc columns.
def drop_cols(df,cols):
  df.drop(cols,axis=1,inplace=True)
  return df

# Data Cleaning - Fill Embark missing values with the most common category;
#                 Fill Fare missing values with the median value in that column.
def fill_val(df):
  df.fillna({'Embarked':df.Embarked.mode()[0],'Fare':df.Fare.median()},inplace=True)
  return df

# Data Cleaning - Fill Age missing values with the median value in each Pclass x Sex combination.
def fill_pivot(df):
  global df_pv
  df_pv = pd.pivot_table(df,values='Age',index='Pclass',columns='Sex',aggfunc=np.median)
  l1 = df_pv.columns.values
  l2 = df_pv.index.values
  df_cld=df.copy()
  for i in range(len(l1)):
    for j in range(len(l2)):
      df_cld.loc[(df_cld.Age.isna())&(df_cld.Sex==l1[i])&(df_cld.Pclass==l2[j]),\
                 ['Age']]=df_pv.loc[l2[j],l1[i]]
  return df_cld

# Feature Engineering - Binning specific column.
def binning(df,Range,ColName,NewColName):
  bins = pd.IntervalIndex.from_tuples(Range)
  df[NewColName] = pd.cut(df[ColName],bins)
  df[NewColName].cat.categories = list(range(0,len(Range)))
  df.drop([ColName],axis=1,inplace=True)
  df[NewColName] = df[NewColName].astype('int64')
  return df

# Feature Engineering - Create a new variable for passengers traveling with family.
def Size(df):
  df['Size'] = df.SibSp + df.Parch
  df.drop(['SibSp','Parch'],axis=1,inplace=True)
  return df

# Feature Engineering - Create new variable for passengers traveling alone.
def Alone(df):
  df['Alone'] = df['Size'].apply(lambda x:
                                 (1 if x == 0 else 0)
                                 )
  return df

# Feature Engineering - label encode Embarked and Sex.
def LabelEncode(df):
  df['Embarked'] = df['Embarked'].replace(('S','C','Q'),(0,1,2))
  df['Sex'] = df['Sex'].replace(('female','male'),(0,1))
  return df

# run pipe
def pipe_fun(file):
  df = pd.read_csv(file)
  df_cleaned = df.pipe(drop_cols,col_name).pipe(fill_val)\
  .pipe(fill_pivot).pipe(binning,AgeRange,ColName_Age,NewColName_Age)\
  .pipe(binning,FareRange,ColName_Fare,NewColName_Fare).pipe(Size).pipe(Alone).pipe(LabelEncode)
  return df_cleaned
```

In [3]:
```
# Define variables.
file_train = 'train.csv'
file_test = 'test.csv'
col_name = ['Name','Ticket','Cabin','PassengerId']
AgeRange = [(0,16),(16,32),(32,48),(48,64),(64,100)]
ColName_Age = 'Age'
NewColName_Age = 'Age_Bins'
FareRange = [(-0.1,7.9),(7.9,14.5),(14.5,31),(31,600)]
ColName_Fare = 'Fare'
NewColName_Fare = 'Fare_Bins'
```

In [4]:
```
# Do data cleaning and feature engineering for train set
# Show the pivot table for bullet point 4 in question 1
df_train = pipe_fun(file_train)
df_pv
```

Out[4]:

Sex	female	male
Pclass		
1	35.0	40.0
2	28.0	30.0
3	21.5	25.0

In [5]:
```
# Report the number of passengers in each Size category for train set.
# Report the number of passengers in each Alone category for train set.
print(df_train['Size'].value_counts(),'\n')
print(df_train['Alone'].value_counts())
```

```
0     537
1     161
2     102
3      29
5      22
4      15
6      12
10      7
7       6
Name: Size, dtype: int64

1    537
0    354
Name: Alone, dtype: int64
```

In [6]:
```
# Report the bottom five rows for train set.
df_train.tail()
```

Out[6]:

	Survived	Pclass	Sex	Embarked	Age_Bins	Fare_Bins	Size	Alone
886	0	2	1	0	1	1	0	1
887	1	1	0	0	1	2	0	1
888	0	3	0	0	1	2	3	0
889	1	1	1	1	1	2	0	1
890	0	3	1	2	1	0	0	1

In [7]:
```
# Do data cleaning and feature engineering for test set
# Show the pivot table for bullet point 4 in question 1
df_test = pipe_fun(file_test)
df_pv
```

Out[7]:

Sex	female	male
Pclass		
1	41.0	42.0
2	24.0	28.0
3	22.0	24.0

In [8]:
```
# Report the number of passengers in each Size category for test set.
# Report the number of passengers in each Alone category for test set.
print(df_test['Size'].value_counts(),'\n')
print(df_test['Alone'].value_counts())
```

```
0     253
1      74
2      57
3      14
4       7
6       4
10      4
5       3
7       2
Name: Size, dtype: int64

1    253
0    165
Name: Alone, dtype: int64
```

In [9]:
```
# Report the bottom five rows for test set.
df_test.tail()
```

Out[9]:

	Survived	Pclass	Sex	Embarked	Age_Bins	Fare_Bins	Size	Alone
413	0	3	1	0	1	1	0	1
414	1	1	0	1	2	3	0	1
415	0	3	1	0	2	0	0	1
416	0	3	1	0	1	1	0	1
417	1	3	1	1	1	2	2	0

Modeling

In [10]:
```
from sklearn.model_selection import KFold
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.neural_network import MLPClassifier
```

In [11]:
```
X_train = df_train.drop(['Survived'],axis=1)
y_train = df_train['Survived']
X_test = df_test.drop(['Survived'],axis=1)
y_test = df_test['Survived']
AR = {}
```

In [12]:
```
def run_model(model_name,model,param,cv):
  grid = GridSearchCV(model,param,cv=cv,scoring='accuracy')
  search=grid.fit(X_train,y_train)
  global AR
  AR[model_name] = list(['{:.3f}%'.format(round(search.score(X_test,y_test),5)*100),search.best_params_])
```

In [13]:
```
# KNN variables
KNN_model_name = 'KNN'
model_knn = KNeighborsClassifier()
knn_param = dict(n_neighbors=list(range(1,16)))

# Logistic variables
Logistic_model_name = 'Logistic'
model_logistic = LogisticRegression(random_state=0)
logistic_param = dict(C=list([0.01,0.1,1,5,11]))

# SVC Variables
SVC_model_name = 'SVC'
model_SVC = SVC(kernel='rbf',random_state=0)
SVC_param = dict(C=list([0.01,0.1,1,5]),gamma=list([0.01,0.1,1,5]))

# RF Variables
RF_model_name = 'RF'
model_RF = RandomForestClassifier(random_state=0)
RF_param = dict(n_estimators=list([10,50,100]),max_depth=list([5,10,15]),max_features=list([3,5,7]))

# GB Variables
GB_model_name = 'GB'
model_GB = GradientBoostingClassifier(random_state=0)
GB_param = dict(learning_rate=list(np.linspace(0.02,0.05,7)),max_features=list([3,4,5,6,7]))

# MLP Variables
MLP_model_name = 'MLP'
model_MLP = MLPClassifier(random_state=0,max_iter=5000)
MLP_param = dict(hidden_layer_sizes=list([50,100]),activation=list(['tanh','relu']),alpha=list([0.01,0.1,1.0]))

model_name = [KNN_model_name,Logistic_model_name,SVC_model_name,RF_model_name,GB_model_name,MLP_model_name]
model = [model_knn,model_logistic,model_SVC,model_RF,model_GB,model_MLP]
param = [knn_param,logistic_param,SVC_param,RF_param,GB_param,MLP_param]
cv = [5,5,5,5,10,5]
```

In [14]:
```
for i in range(len(model)):
  run_model(model_name[i],model[i],param[i],cv[i])
```

In [15]:
```
Report = pd.DataFrame(AR,index=['Accuracy Rate','Best Parameters']).T
pd.set_option('display.max_colwidth', 150)
Report.sort_values(by='Accuracy Rate', ascending=False)
```

Out[15]:

	Accuracy Rate	Best Parameters
RF	78.230%	{'max_depth': 5, 'max_features': 3, 'n_estimators': 10}
SVC	77.751%	{'C': 5, 'gamma': 0.1}
MLP	77.273%	{'activation': 'tanh', 'alpha': 0.01, 'hidden_layer_sizes': 50}
GB	77.033%	{'learning_rate': 0.05, 'max_features': 3}
Logistic	75.120%	{'C': 0.1}
KNN	72.967%	{'n_neighbors': 15}

Summary: Random Forest is the best model to predict this problem. And it's accuracy rate is 78.230%. The best parameters are {'max_depth': 5, 'max_features': 3, 'n_estimators': 10}