Course - IS 665

Professor: Lin Lin

Team 4: Ullash, Saleena, Jay, Namita, Sharan

UCID: uj25, sj674, jap256, nm648, sh624

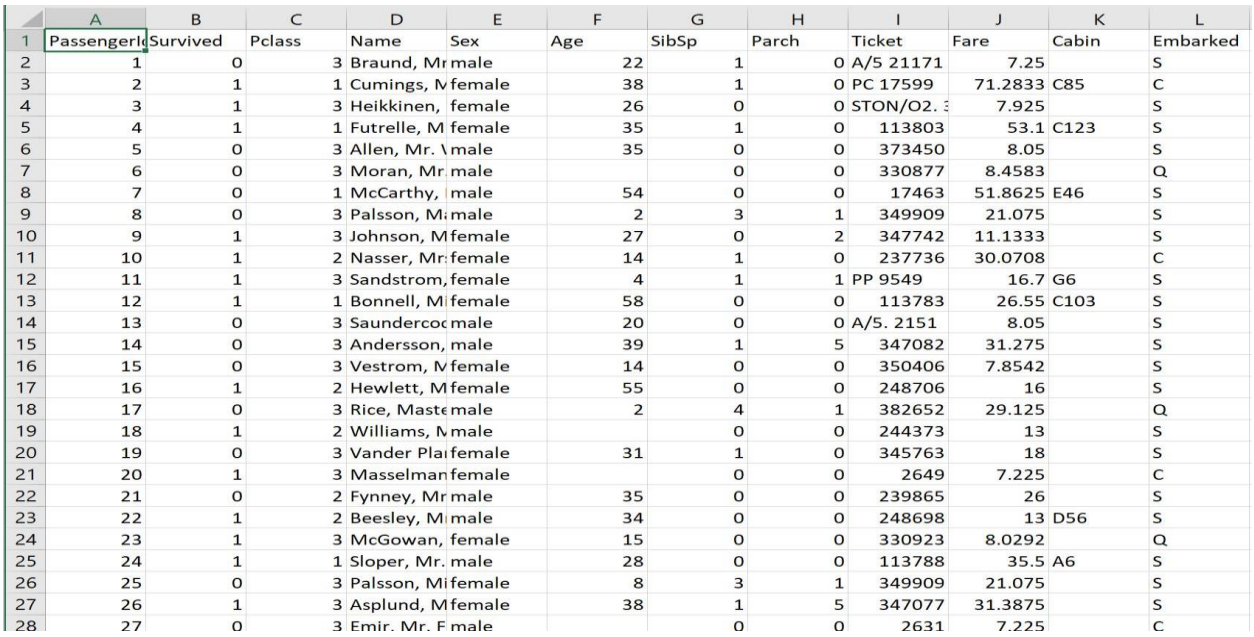
Subject: Project II Report (Data Analytics for IS)

Topic: Supervised Data Mining (Classification)

Category - Random Forest & Decision Tree

Titanic Data Set

Titanic - Train Dataset



	A	B	C	D	E	F	G	H	I	J	K	L
1	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
2	1	0	3	Braund, Mr	male	22	1	0	A/5 21171	7.25		S
3	2	1	1	Cumings, M	female	38	1	0	PC 17599	71.2833	C85	C
4	3	1	3	Heikkinen,	female	26	0	0	STON/O2. 3	7.925		S
5	4	1	1	Futrelle, M	female	35	1	0	113803	53.1	C123	S
6	5	0	3	Allen, Mr. \	male	35	0	0	373450	8.05		S
7	6	0	3	Moran, Mr	male		0	0	330877	8.4583		Q
8	7	0	1	McCarthy,	male	54	0	0	17463	51.8625	E46	S
9	8	0	3	Palsson, M	male	2	3	1	349909	21.075		S
10	9	1	3	Johnson, M	female	27	0	2	347742	11.1333		S
11	10	1	2	Nasser, Mr	female	14	1	0	237736	30.0708		C
12	11	1	3	Sandstrom,	female	4	1	1	PP 9549	16.7	G6	S
13	12	1	1	Bonnell, Mi	female	58	0	0	113783	26.55	C103	S
14	13	0	3	Saundercoc	male	20	0	0	A/5. 2151	8.05		S
15	14	0	3	Andersson,	male	39	1	5	347082	31.275		S
16	15	0	3	Vestrom, M	female	14	0	0	350406	7.8542		S
17	16	1	2	Hewlett, M	female	55	0	0	248706	16		S
18	17	0	3	Rice, Maste	male	2	4	1	382652	29.125		Q
19	18	1	2	Williams, M	male		0	0	244373	13		S
20	19	0	3	Vander Pla	female	31	1	0	345763	18		S
21	20	1	3	Masselman	female		0	0	2649	7.225		C
22	21	0	2	Fynney, Mr	male	35	0	0	239865	26		S
23	22	1	2	Beesley, Mi	male	34	0	0	248698	13	D56	S
24	23	1	3	McGowan,	female	15	0	0	330923	8.0292		Q
25	24	1	1	Sloper, Mr.	male	28	0	0	113788	35.5	A6	S
26	25	0	3	Palsson, Mi	female	8	3	1	349909	21.075		S
27	26	1	3	Asplund, M	female	38	1	5	347077	31.3875		S
28	27	0	3	Emir, Mr. F	male		0	0	2631	7.225		C

Titanic - Test Dataset

	A	B	C	D	E	F	G	H	I	J	K
1	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
2	892	3	Kelly, Mr. J	male	34.5	0	0	330911	7.8292		Q
3	893	3	Wilkes, Mr	female	47	1	0	363272	7		S
4	894	2	Myles, Mr.	male	62	0	0	240276	9.6875		Q
5	895	3	Wirz, Mr. A	male	27	0	0	315154	8.6625		S
6	896	3	Hirvonen, M	female	22	1	1	3101298	12.2875		S
7	897	3	Svensson, M	male	14	0	0	7538	9.225		S
8	898	3	Connolly, M	female	30	0	0	330972	7.6292		Q
9	899	2	Caldwell, M	male	26	1	1	248738	29		S
10	900	3	Abrahim, M	female	18	0	0	2657	7.2292		C
11	901	3	Davies, Mr.	male	21	2	0	A/4 48871	24.15		S
12	902	3	Ilieff, Mr. Y	male		0	0	349220	7.8958		S
13	903	1	Jones, Mr. (	male	46	0	0	694	26		S
14	904	1	Snyder, Mr	female	23	1	0	21228	82.2667	B45	S
15	905	2	Howard, M	male	63	1	0	24065	26		S
16	906	1	Chaffee, M	female	47	1	0	W.E.P. 573	61.175	E31	S
17	907	2	del Carlo, M	female	24	1	0	SC/PARIS 2	27.7208		C
18	908	2	Keane, Mr.	male	35	0	0	233734	12.35		Q
19	909	3	Assaf, Mr. (	male	21	0	0	2692	7.225		C
20	910	3	Ilmakangas	female	27	1	0	STON/O2. 3	7.925		S
21	911	3	Assaf Khalil	female	45	0	0	2696	7.225		C
22	912	1	Rothschild,	male	55	1	0	PC 17603	59.4		C
23	913	3	Olsen, Mas	male	9	0	1	C 17368	3.1708		S
24	914	1	Flegenheim	female		0	0	PC 17598	31.6833		S
25	915	1	Williams, M	male	21	0	1	PC 17597	61.3792		C
26	916	1	Ryerson, M	female	48	1	3	PC 17608	262.375	B57 B59 B6	C
27	917	3	Robins, Mr	male	50	1	0	A/5. 3337	14.5		S

Data Set Explanation

PassengerId: Distinct id's for Passengers

Pclass: Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd)

Survived: Survived (0 = No; 1 = Yes)

Name: Name of the Passengers

Sex: Sex (M/F)

Age: Age of the Passengers

SibSp: Number of Siblings/Spouses Aboard

Parch: Number of Parents/Children Aboard

Ticket: Ticket Number

Fare: Passenger Fare (British pound)

Cabin: Cabin

Embarked: Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)

Source Code:

 # Importing important libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

 # Importing Train and Test data

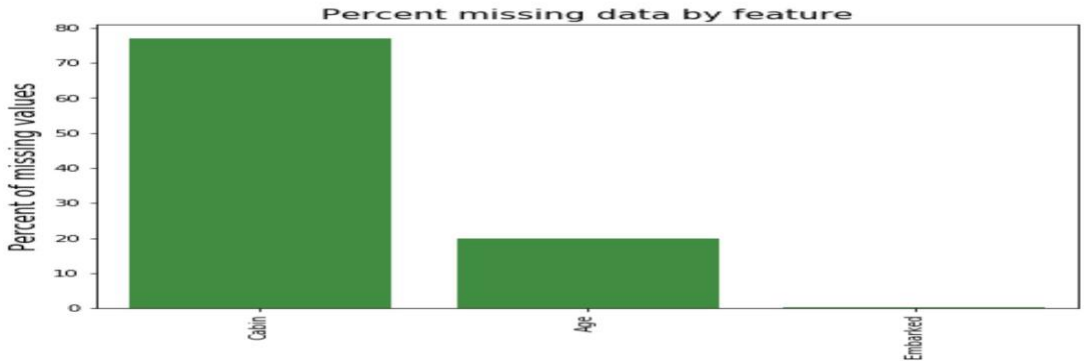
```
train_df=pd.read_csv("train.csv")
test_df=pd.read_csv("test.csv")
```

 # Looking for the missing data values

```
def missingdata(data):
    total = data.isnull().sum().sort_values(ascending = False)
    percent = (data.isnull().sum()/data.isnull().count()*100).sort_values(ascending = False)
    ms=pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
    ms= ms[ms["Percent"] > 0]
    f,ax =plt.subplots(figsize=(8,6))
    plt.xticks(rotation='90')
    fig=sns.barplot(ms.index, ms["Percent"],color="green",alpha=0.8)
    plt.xlabel('Features', fontsize=15)
    plt.ylabel('Percent of missing values', fontsize=15)
    plt.title('Percent missing data by feature', fontsize=15)
    return ms
```

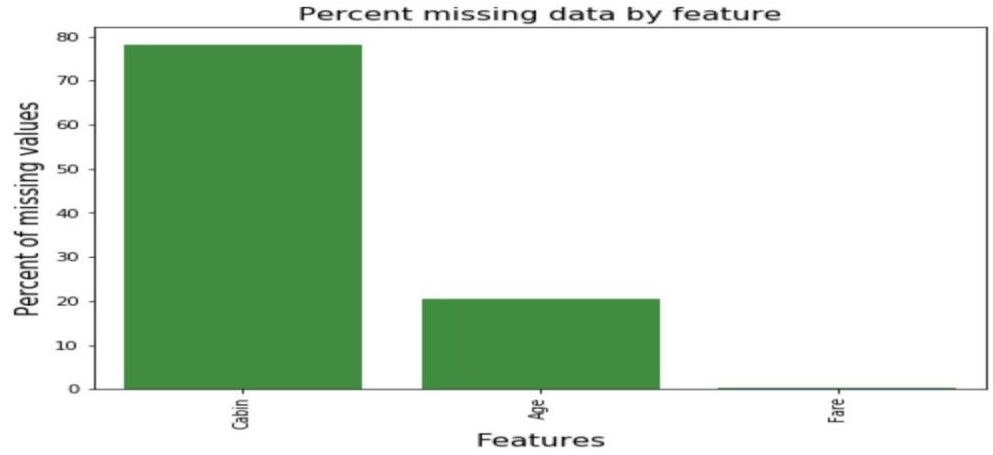
#Identifying Missing Value
missingdata(train_df)

	Total	Percent
Cabin	687	77.104377
Age	177	19.865320
Embarked	2	0.224467



missingdata(test_df)

	Total	Percent
Cabin	327	78.229665
Age	86	20.574163
Fare	1	0.239234



 # Replacing the Missing embarked Column values by mode value
train_df['Embarked'].fillna(train_df['Embarked'].mode()[0], inplace = True)

 # Replacing the Missing Fare values by median value
test_df['Fare'].fillna(test_df['Fare'].median(), inplace = True)

 # Cabin Featueres has more than 75% of missing data in both Test and train data so we will remove the Cabin attribute
drop_column = ['Cabin']
train_df.drop(drop_column, axis=1, inplace = True)
test_df.drop(drop_column,axis=1,inplace=True)

```
# Both the test and train Age features contains more the 15% of missing Data so we are fill with the median
test_df['Age'].fillna(test_df['Age'].median(), inplace = True)
train_df['Age'].fillna(train_df['Age'].median(), inplace = True)

print('check the nan value in train data')
print(train_df.isnull().sum())
print('___'*30)
print('check the nan value in test data')
print(test_df.isnull().sum())

# combine test and train as single to apply some function
all_data=[train_df,test_df]

# Create new feature FamilySize as a combination of SibSp and Parch
for dataset in all_data:
   dataset['FamilySize'] = dataset['SibSp'] + dataset['Parch'] + 1

import re
# Define function to extract titles from passenger names
def get_title(name):
   title_search = re.search(' ([A-Za-z]+)\.', name)
   # If the title exists, extract and return it.
   if title_search:
      return title_search.group(1)
   return ""
# Create a new feature Title, containing the titles of passenger names
for dataset in all_data:
   dataset['Title'] = dataset['Name'].apply(get_title)
# Group all non-common titles into one single grouping "Rare"
for dataset in all_data:
   dataset['Title'] = dataset['Title'].replace(['Lady', 'Countess','Capt', 'Col','Don',
                                    'Dr', 'Major', 'Rev', 'Sir', 'Jonkheer', 'Dona'], 'Rare')

   dataset['Title'] = dataset['Title'].replace('Mlle', 'Miss')
   dataset['Title'] = dataset['Title'].replace('Ms', 'Miss')
   dataset['Title'] = dataset['Title'].replace('Mme', 'Mrs')

for dataset in all_data:
   dataset['Age_bin'] = pd.cut(dataset['Age'], bins=[0,12,20,40,120],
labels=['Children','Teenage','Adult','Elder'])



# create bin for fare features
for dataset in all_data:
```

```
   dataset['Fare_bin'] = pd.cut(dataset['Fare'], bins=[0,7.91,14.45,31,120],
labels=['Low_fare','median_fare',
                                                'Average_fare','high_fare'])

# for our reference making a copy of both DataSet start working for copy of dataset
traindf=train_df
testdf=test_df

all_dat=[traindf,testdf]

for dataset in all_dat:
   drop_column = ['Age','Fare','Name','Ticket']
   dataset.drop(drop_column, axis=1, inplace = True)

drop_column = ['PassengerId']
traindf.drop(drop_column, axis=1, inplace = True)

# Final Train data for modeling
traindf = pd.get_dummies(traindf, columns = ["Sex","Title","Age_bin","Embarked","Fare_bin"],
                         prefix=["Sex","Title","Age_type","Em_type","Fare_type"])

# Final Test data for modeling
testdf = pd.get_dummies(testdf, columns = ["Sex","Title","Age_bin","Embarked","Fare_bin"],
                         prefix=["Sex","Title","Age_type","Em_type","Fare_type"])
testdf.head()
```

	PassengerId	Pclass	SibSp	Parch	FamilySize	Sex_female	Sex_male	Title_Master	Title_Miss	Title_Mr	...	Age_type_Teenage	Age_type_Adult	Age_t
0	892	3	0	0	1	0	1	0	0	1	...	0	1	
1	893	3	1	0	2	1	0	0	0	0	...	0	0	
2	894	2	0	0	1	0	1	0	0	1	...	0	0	
3	895	3	0	0	1	0	1	0	0	1	...	0	1	
4	896	3	1	1	3	1	0	0	0	0	...	0	1	

5 rows × 23 columns

Age_type_Adult	Age_type_Elder	Em_type_C	Em_type_Q	Em_type_S	Fare_type_Low_fare	Fare_type_median_fare	Fare_type_Average_fare	Fare_type_high_
1	0	0	1	0	1	0	0	
0	1	0	0	1	1	0	0	
0	1	0	1	0	0	1	0	
1	0	0	0	1	0	1	0	
1	0	0	0	1	0	1	0	

Importing necessary libraries for split,10 fold cross validation, accuracy and confusion matrix

```
from sklearn.model_selection import train_test_split #for split the data
from sklearn.metrics import accuracy_score  #for accuracy_score
from sklearn.model_selection import KFold #for K-fold cross validation
```

```
from sklearn.model_selection import cross_val_score #score evaluation
from sklearn.model_selection import cross_val_predict #prediction
from sklearn.metrics import confusion_matrix #for confusion matrix
all_features = traindf.drop("Survived",axis=1)
Targeted_feature = traindf["Survived"]
X_train,X_test,y_train,y_test =
train_test_split(all_features,Targeted_feature,test_size=0.3,random_state=42)
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

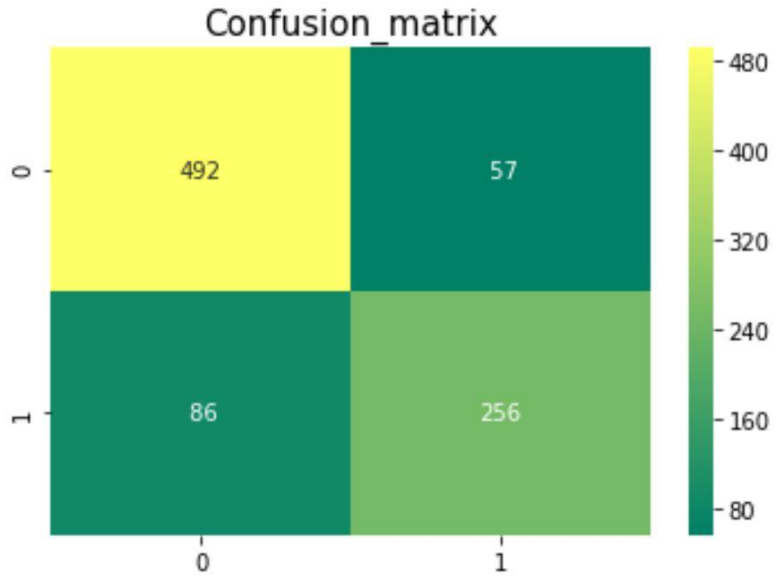
```
((623, 22), (268, 22), (623,), (268,))
```

Random Forest Accuracy & Cross Validation Score:

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(criterion='gini', n_estimators=700,
                               min_samples_split=10,min_samples_leaf=1,
                               max_features='auto',oob_score=True,
                               random_state=1,n_jobs=-1)
model.fit(X_train,y_train)
prediction_rm=model.predict(X_test)
print('--------------The Accuracy of the model----------------------------')
print('The accuracy of the Random Forest Classifier is',round(accuracy_score(prediction_rm,y_test)*10
0,2))
kfold = KFold(n_splits=10, random_state=22) # k=10, split the data into 10 equal parts
result_rm=cross_val_score(model,all_features,Targeted_feature,cv=10,scoring='accuracy')
print('The cross validated score for Random Forest Classifier is:',round(result_rm.mean()*100,2))
y_pred = cross_val_predict(model,all_features,Targeted_feature,cv=10)
sns.heatmap(confusion_matrix(Targeted_feature,y_pred),annot=True,fmt='3.0f',cmap="summer")
plt.title('Confusion_matrix', y=1.05, size=15)
```

```
--------------The Accuracy of the model----------------------------
The accuracy of the Random Forest Classifier is 82.46
The cross validated score for Random Forest Classifier is: 83.95

Text(0.5, 1.05, 'Confusion_matrix')
```

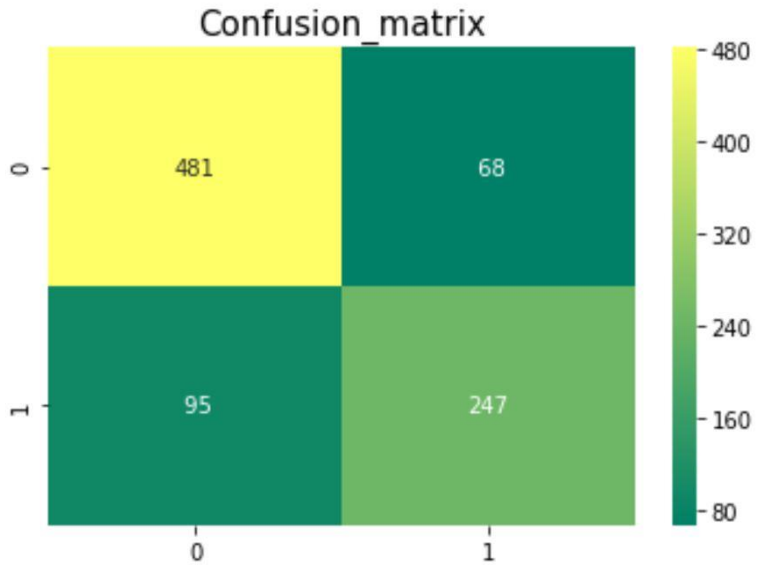


Decision Tree Accuracy & Cross Validation Score :

```
from sklearn.tree import DecisionTreeClassifier
model= DecisionTreeClassifier(criterion='gini',
                    min_samples_split=10,min_samples_leaf=1,
                    max_features='auto')
model.fit(X_train,y_train)
prediction_tree=model.predict(X_test)
print('--------------The Accuracy of the model----------------------------')
print('The accuracy of the DecisionTree Classifier is',round(accuracy_score(prediction_tree,y_test)*10
0,2))
kfold = KFold(n_splits=10, random_state=22) # k=10, split the data into 10 equal parts
result_tree=cross_val_score(model,all_features,Targeted_feature,cv=10,scoring='accuracy')
print('The cross validated score for Decision Tree classifier is:',round(result_tree.mean()*100,2))
y_pred = cross_val_predict(model,all_features,Targeted_feature,cv=10)
sns.heatmap(confusion_matrix(Targeted_feature,y_pred),annot=True,fmt='3.0f',cmap="summer")
plt.title('Confusion_matrix', y=1.05, size=15)
```

```
--------------The Accuracy of the model----------------------------
The accuracy of the DecisionTree Classifier is 76.87
The cross validated score for Decision Tree classifier is: 82.28

Text(0.5, 1.05, 'Confusion_matrix')
```



<u>Insight:</u> Hence looking at the Accuracy and the Cross-Validation Score, we can say Random Forest is more precise and accurate in predicting classes as compared to Decision Tree Algorithm. And as we know that Random Forest model involves more than one decision tree, so our accuracy value depicts the same and also cross validation score confirms that for labeling, Random Forest classifier is better compared to Decision Tree.

REFERENCES:
Dataset: https://www.kaggle.com/vinothan/titanic-model-with-90-accuracy