In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
```

Loading the Dataset

First we load the dataset and find out the number of columns, rows, NULL values, etc.

In [2]:
```
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')
```

In [3]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

In [4]:
```
test.info()
```

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

In [5]:
```
train.head()
```

Out[5]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [6]:
```
test.head()
```

Out[6]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

Cleaning

In [7]:
```
train.corr().style.background_gradient(cmap='BuGn')
```

Out[7]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500
Age	0.036847	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067
SibSp	-0.057527	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651
Parch	-0.001652	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225
Fare	0.012658	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000

In [8]:
```
train.drop(['PassengerId', 'Name', 'Ticket', 'Cabin'], axis=1, inplace=True)
test.drop(['PassengerId', 'Name', 'Ticket', 'Cabin'], axis=1, inplace=True)
```

In [9]:
```
train.isna().sum()
```

Out[9]:
```
Survived      0
Pclass        0
Sex           0
Age         177
SibSp         0
Parch         0
Fare          0
Embarked      2
dtype: int64
```

In [10]:
```
test.isna().sum()
```

Out[10]:
```
Pclass       0
Sex          0
Age         86
SibSp        0
Parch        0
Fare         1
Embarked     0
dtype: int64
```

In [11]:
```
train['Embarked'] = train.Embarked.fillna(train.Embarked.dropna().max())
test['Fare'] = test.Fare.fillna(test.Fare.dropna().mean())
```

In [12]:
```
# we will guess the age from Pclass and Sex:

guess_ages = np.zeros((2,3))
guess_ages
```

Out[12]:
```
array([[0., 0., 0.],
       [0., 0., 0.]])
```

Now we iterate over Sex (0 or 1) and Pclass (1, 2, 3) to calculate guessed values of Age for the six combinations.

In [13]:
```
combine = [train , test]

# Converting Sex categories (male and female) to 0 and 1:
for dataset in combine:
    dataset['Sex'] = dataset['Sex'].map( {'female': 1, 'male': 0} ).astype(int)

# Filling missed age feature:
for dataset in combine:
    for i in range(0, 2):
        for j in range(0, 3):
            guess_df = dataset[(dataset['Sex'] == i) & \
                               (dataset['Pclass'] == j+1)]['Age'].dropna()
            age_guess = guess_df.median()

            # Convert random age float to nearest .5 age
            guess_ages[i,j] = int( age_guess/0.5 + 0.5 ) * 0.5

    for i in range(0, 2):
        for j in range(0, 3):
            dataset.loc[ (dataset.Age.isnull()) & (dataset.Sex == i) & (dataset.Pclass == j+1),\
                    'Age'] = guess_ages[i,j]

    dataset['Age'] = dataset['Age'].astype(int)

train.head()
```

Out[13]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	0	22	1	0	7.2500	S
1	1	1	1	38	1	0	71.2833	C
2	1	3	1	26	0	0	7.9250	S
3	1	1	1	35	1	0	53.1000	S
4	0	3	0	35	0	0	8.0500	S

In [14]:
```
train.describe()
```

Out[14]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	0.352413	29.072952	0.523008	0.381594	32.204208
std	0.486592	0.836071	0.477990	13.326339	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	0.000000	21.000000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	0.000000	26.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	1.000000	36.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	1.000000	80.000000	8.000000	6.000000	512.329200

Splitting the Dataset

Training and Test Set

In [15]:
```
X_train = pd.get_dummies(train.drop(['Survived'], axis=1))
X_test  = pd.get_dummies(test)
y_train = train['Survived']
```

Machine Learning model

In [16]:
```
def print_scores(model, X_train, Y_train, predictions, cv_splites=10):
    print("The mean accuracy score of the train data is %.5f" % model.score(X_train, Y_train))
    CV_scores = cross_val_score(model, X_train, Y_train, cv=cv_splites)
    print("The individual cross-validation scores are: \n",CV_scores)
    print("The minimum cross-validation score is %.3f" % min(CV_scores))
    print("The maximum cross-validation score is %.3f" % max(CV_scores))
    print("The mean  cross-validation   score is %.5f ± %0.2f" % (CV_scores.mean(), CV_scores.std() * 2
```

In [17]:
```
model = RandomForestClassifier(n_estimators= 80 ,max_depth=5 , max_features=8 ,min_samples_split=3 ,ran
model.fit(X_train, y_train)
predictions = model.predict(X_test)
print_scores(model, X_train, y_train, predictions)
```

```
The mean accuracy score of the train data is 0.85859
The individual cross-validation scores are:
 [0.76666667 0.85393258 0.75280899 0.91011236 0.88764045 0.80898876
 0.80898876 0.78651685 0.87640449 0.84269663]
The minimum cross-validation score is 0.753
The maximum cross-validation score is 0.910
The mean  cross-validation   score is 0.82948 ± 0.10
```