In [1]:

```
import pandas as pd
```

Cleaning Data from null, convert to numeric, drop column

In [2]:

```
from sklearn.metrics import accuracy_score, recall_score, precision_score
def cross_validate(classifier, train, validation):
    X_train = train[0]
    Y_train = train[1]
    X_val = validation[0]
    Y_val = validation[1]
    train_predictions = classifier.predict(X_train)
    train_accuracy = accuracy_score(train_predictions, Y_train)
    train_recall = recall_score(train_predictions, Y_train)
    train_precision = precision_score(train_predictions, Y_train)

    val_predictions = classifier.predict(X_val)
    val_accuracy = accuracy_score(val_predictions, Y_val)
    val_recall = recall_score(val_predictions, Y_val)
    val_precision = precision_score(val_predictions, Y_val)

    print('Model metrics')
    print('Accuracy  Train: %.2f, Validation: %.2f' % (train_accuracy, val_accuracy))
    print('Recall    Train: %.2f, Validation: %.2f' % (train_recall, val_recall))
    print('Precision Train: %.2f, Validation: %.2f' % (train_precision, val_precision))
```

In [3]:

```
# open file with pd.read_csv
train = pd.read_csv("train.csv")
test = pd.read_csv("test.csv")
# print head of data set
train.info()
print('---------------------------------')
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Name           891 non-null object
Sex            891 non-null object
Age            714 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          204 non-null object
Embarked       889 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.6+ KB
---------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
PassengerId    418 non-null int64
Pclass         418 non-null int64
Name           418 non-null object
Sex            418 non-null object
Age            332 non-null float64
SibSp          418 non-null int64
Parch          418 non-null int64
Ticket         418 non-null object
Fare           417 non-null float64
Cabin          91 non-null object
Embarked       418 non-null object
dtypes: float64(2), int64(4), object(5)
memory usage: 36.0+ KB
```

Clean Train Data

In [4]:

```
train_clean = train.drop('PassengerId',axis=1).drop('Name',axis=1).drop('Cabin',axis=1)
.drop('Ticket',axis=1)
train_clean.head()
```

Out[4]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	22.0	1	0	7.2500	S
1	1	1	female	38.0	1	0	71.2833	C
2	1	3	female	26.0	0	0	7.9250	S
3	1	1	female	35.0	1	0	53.1000	S
4	0	3	male	35.0	0	0	8.0500	S

In [5]:

```
train_clean['Sex'] = train_clean['Sex'].replace(['male','female'],[0,1])
train_clean['Embarked'] = train_clean['Embarked'].replace(['S','C','Q'],[0,1,2])
train_clean.head()
```

Out[5]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	0	22.0	1	0	7.2500	0.0
1	1	1	1	38.0	1	0	71.2833	1.0
2	1	3	1	26.0	0	0	7.9250	0.0
3	1	1	1	35.0	1	0	53.1000	0.0
4	0	3	0	35.0	0	0	8.0500	0.0

In [6]:

```
train_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 8 columns):
Survived    891 non-null int64
Pclass      891 non-null int64
Sex         891 non-null int64
Age         714 non-null float64
SibSp       891 non-null int64
Parch       891 non-null int64
Fare        891 non-null float64
Embarked    889 non-null float64
dtypes: float64(3), int64(5)
memory usage: 55.8 KB
```

In [7]:

```
train_clean['Age'] = train_clean['Age'].interpolate(method = 'linear')
train_clean['Embarked'] = train_clean['Embarked'].interpolate(method = 'linear')
train_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 8 columns):
Survived    891 non-null int64
Pclass      891 non-null int64
Sex         891 non-null int64
Age         891 non-null float64
SibSp       891 non-null int64
Parch       891 non-null int64
Fare        891 non-null float64
Embarked    891 non-null float64
dtypes: float64(3), int64(5)
memory usage: 55.8 KB
```

Clean Test Data

In [8]:

```
test_clean = test.drop('PassengerId',axis=1).drop('Name',axis=1).drop('Cabin',axis=1).d
rop('Ticket',axis=1)
test_clean.head()
```

Out[8]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	male	34.5	0	0	7.8292	Q
1	3	female	47.0	1	0	7.0000	S
2	2	male	62.0	0	0	9.6875	Q
3	3	male	27.0	0	0	8.6625	S
4	3	female	22.0	1	1	12.2875	S

In [9]:

```
test_clean['Sex'] = test_clean['Sex'].replace(['male','female'],[0,1])
test_clean['Embarked'] = test_clean['Embarked'].replace(['S','C','Q'],[0,1,2])
test_clean.head()
```

Out[9]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	0	34.5	0	0	7.8292	2
1	3	1	47.0	1	0	7.0000	0
2	2	0	62.0	0	0	9.6875	2
3	3	0	27.0	0	0	8.6625	0
4	3	1	22.0	1	1	12.2875	0

In [10]:

```
test_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 7 columns):
Pclass      418 non-null int64
Sex         418 non-null int64
Age         332 non-null float64
SibSp       418 non-null int64
Parch       418 non-null int64
Fare        417 non-null float64
Embarked    418 non-null int64
dtypes: float64(2), int64(5)
memory usage: 22.9 KB
```

In [11]:

```
test_clean['Age'] = test_clean['Age'].interpolate(method = 'linear')
test_clean['Fare'] = train_clean['Fare'].interpolate(method = 'linear')
test_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 7 columns):
Pclass      418 non-null int64
Sex         418 non-null int64
Age         418 non-null float64
SibSp       418 non-null int64
Parch       418 non-null int64
Fare        418 non-null float64
Embarked    418 non-null int64
dtypes: float64(2), int64(5)
memory usage: 22.9 KB
```

Split data - Create a Model

Split Data

In [12]:

```
x = train_clean.drop('Survived',axis = 1)
y = train_clean['Survived']
```

In [13]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.35, random_state=
66)
```

In [14]:

```
x_train.info()
print('--------------------')
x_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 579 entries, 291 to 20
Data columns (total 7 columns):
Pclass      579 non-null int64
Sex         579 non-null int64
Age         579 non-null float64
SibSp       579 non-null int64
Parch       579 non-null int64
Fare        579 non-null float64
Embarked    579 non-null float64
dtypes: float64(3), int64(4)
memory usage: 36.2 KB
--------------------
<class 'pandas.core.frame.DataFrame'>
Int64Index: 312 entries, 28 to 628
Data columns (total 7 columns):
Pclass      312 non-null int64
Sex         312 non-null int64
Age         312 non-null float64
SibSp       312 non-null int64
Parch       312 non-null int64
Fare        312 non-null float64
Embarked    312 non-null float64
dtypes: float64(3), int64(4)
memory usage: 19.5 KB
```

Train Model

In [15]:

```
from sklearn.ensemble.forest import RandomForestClassifier

from sklearn import model_selection
```

In [16]:

```
# random forest model creation
rfc = RandomForestClassifier()
rfc.fit(x_train,y_train)
```

```
C:\Users\Alvaro Basily\Anaconda3\lib\site-packages\sklearn\ensemble\fores
t.py:246: FutureWarning: The default value of n_estimators will change fro
m 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

Out[16]:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gin
i',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

Cross Validate

In [17]:

```
cross_validate(rfc, (x_train,y_train), (x_test,y_test))
```

```
Model metrics
Accuracy  Train: 0.98, Validation: 0.81
Recall    Train: 0.98, Validation: 0.74
Precision Train: 0.96, Validation: 0.70
```

Classification Test Data with our model

In [18]:

```
predict = rfc.predict(test_clean)
```

In [19]:

```
predict_class = test_clean
```

In [20]:

```
predict_class['Survived'] = predict.astype('int')
```

In [21]:

```
predict_class.head()
```

Out[21]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Survived
0	3	0	34.5	0	0	7.2500	2	0
1	3	1	47.0	1	0	71.2833	0	0
2	2	0	62.0	0	0	7.9250	2	0
3	3	0	27.0	0	0	53.1000	0	1
4	3	1	22.0	1	1	8.0500	0	0

How about use Naive Bayes Classification?

In [22]:

```
from sklearn.naive_bayes import GaussianNB
```

In [23]:

```
nbc = GaussianNB()
```

In [24]:

```
nbc.fit(x_train, y_train)
```

Out[24]:

```
GaussianNB(priors=None, var_smoothing=1e-09)
```

In [25]:

```
cross_validate(nbc, (x_train,y_train), (x_test,y_test))
```

```
Model metrics
Accuracy  Train: 0.79, Validation: 0.78
Recall    Train: 0.76, Validation: 0.68
Precision Train: 0.71, Validation: 0.70
```

In [26]:

```
print('Probability of each class')
print('Survive = 0: %.2f' % nbc.class_prior_[0])
print('Survive = 1: %.2f' % nbc.class_prior_[1])
```

```
Probability of each class
Survive = 0: 0.60
Survive = 1: 0.40
```

With Decission Tree?

In [27]:

```
from sklearn import tree
```

In [28]:

```
dtc = tree.DecisionTreeClassifier()
dtc.fit(x_train, y_train)
```

Out[28]:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=Non
e,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=Non
e,
            splitter='best')
```

In [29]:

```
cross_validate(dtc, (x_train,y_train), (x_test,y_test))
```

```
Model metrics
Accuracy  Train: 0.99, Validation: 0.75
Recall    Train: 1.00, Validation: 0.66
Precision Train: 0.97, Validation: 0.61
```

In [ ]: