Importing Dataset

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

In [ ]:
```
data = pd.read_csv("train.csv")
```

In [ ]:
```
data.head()
```

Out[ ]:

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

Data Description

In [ ]:
```
data.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
```

Data Cleaning and Preprocessing

In [ ]:
```
columns_to_drop = ["PassengerId", "Name", "Ticket", "Cabin", "Embarked"]
data_clean = data.drop(columns_to_drop, axis=1)
data_clean.head()
```

Out[ ]:

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

Label Encoding

In [ ]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
data_clean["Sex"] = le.fit_transform(data_clean["Sex"])
```

In [ ]:
```
data_clean.head()
```

Out[ ]:

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

In [ ]:
```
data_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
```

```
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Survived  891 non-null    int64
 1   Pclass    891 non-null    int64
 2   Sex       891 non-null    int32
 3   Age       714 non-null    float64
 4   SibSp     891 non-null    int64
 5   Parch     891 non-null    int64
 6   Fare      891 non-null    float64
dtypes: float64(2), int32(1), int64(4)
memory usage: 45.4 KB
```

In [ ]:
```
data_clean.describe()
```

Out[ ]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	0.647587	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	0.477990	14.526497	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	0.000000	20.125000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	1.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	1.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	1.000000	80.000000	8.000000	6.000000	512.329200

In [ ]:
```
data_clean = data_clean.fillna(data_clean["Age"].mean()) #Imputer can also be used
```

In [ ]:
```
data_clean.describe()
```

Out[ ]:

	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	0.647587	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	0.477990	13.002015	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	0.000000	22.000000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	1.000000	29.699118	0.000000	0.000000	14.454200
75%	1.000000	3.000000	1.000000	35.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	1.000000	80.000000	8.000000	6.000000	512.329200

In [ ]:
```
input_cols = ["Pclass", "Sex", "Age", "SibSp", "Parch", "Fare"]
output_cols = ["Survived"]
```

In [ ]:
```
X = data_clean[input_cols]
Y = data_clean[output_cols]
```

In [ ]:
```
X.shape,Y.shape
```

Out[ ]:
```
((891, 6), (891, 1))
```

Creating Test-Train Split

In [ ]:
```
split = int(0.7*data_clean.shape[0])
train_data = data_clean[:split]
test_data = data_clean[split:]
test_data = test_data.reset_index(drop=True)
```

In [ ]:
```
X_train = train_data[input_cols]
Y_train = np.array(train_data[output_cols]).reshape((-1,))
X_test = test_data[input_cols]
Y_test = np.array(test_data[output_cols]).reshape((-1,))
```

Random Forest Classfier

- Ensemble Learning

In [ ]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [ ]:
```
rf  = RandomForestClassifier(n_estimators=10, criterion='entropy', max_depth=8)
rf.fit(X_train,Y_train)
```

Out[ ]:
```
RandomForestClassifier(criterion='entropy', max_depth=8, n_estimators=10)
```

Train Accuracy

In [ ]:
```
rf.score(X_train,Y_train)
```

Out[ ]:
```
0.8940609951845907
```

Test Accuracy

In [ ]:
```
rf.score(X_test,Y_test)
```

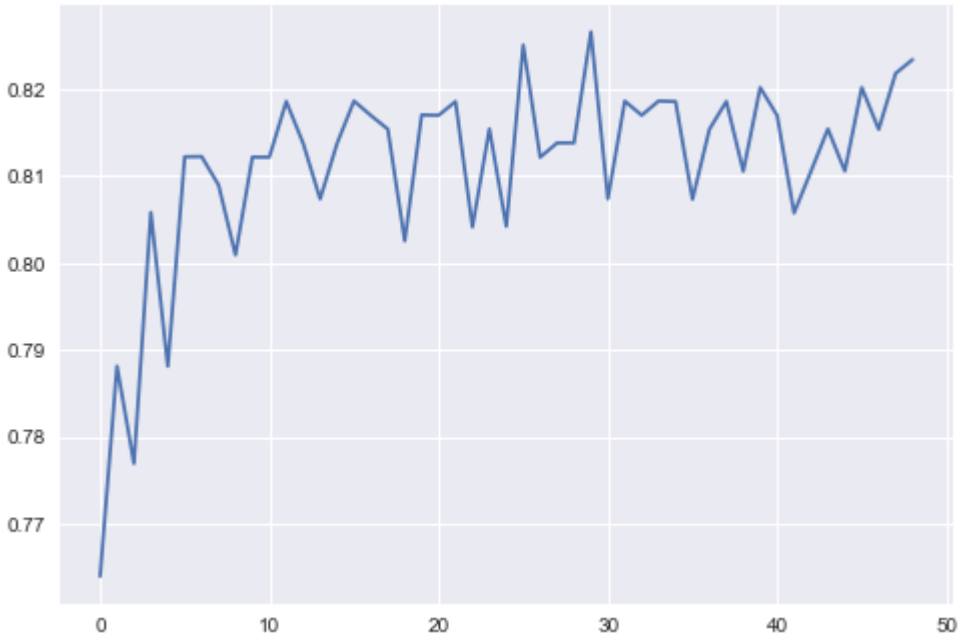
Out[ ]:
```
0.8432835820895522
```

Cross Validation

In [ ]:
```
from sklearn.model_selection import cross_val_score
```

In [ ]:
```
acc_list = []
for i in range(1,50):
    acc = cross_val_score(RandomForestClassifier(n_estimators=i,max_depth=5,criterion='
    acc_list.append(acc)
```

In [ ]:
```
import matplotlib.pyplot as plt
plt.style.use('seaborn')
plt.plot(acc_list)
plt.show()
```



In [ ]:
```
print(np.argmax(acc_list))
```

```
29
```

In [ ]:
```
rf_test = RandomForestClassifier(n_estimators=29,max_depth=5,criterion='entropy')
```

In [ ]:
```
rf_test.fit(X_train,Y_train)
rf_test.score(X_train,Y_train)
```

Out[ ]:
```
0.8619582664526485
```

In [ ]:
```
rf_test.score(X_test,Y_test)
```

Out[ ]:
```
0.8246268656716418
```

In [ ]: