Predicting Survival in the Titanic Data Set We will be using a decision tree to make predictions about the Titanic data set from Kaggle. This data set provides information on the Titanic passengers and can be used to predict whether a passenger survived or not.

In [49]:
```
#Import pacakges
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [50]:
```
#import dataset
Url= 'https://raw.githubusercontent.com/BigDataGal/Python-for-Data-Science/master/

titanic = pd.read_csv(Url)
titanic.columns = ['PassengerId','Survived','Pclass','Name','Sex','Age','SibSp','
```

In [51]:
```
titanic.head(5)
```

Out[51]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabi
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	Na
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C8
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	Na
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C12
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	Na

In [52]:
```
#You use only Pclass, Sex, Age, SibSp (Siblings aboard), Parch (Parents/children a
#and Fare to predict whether a passenger survived.

df = titanic.drop(['PassengerId', 'Name','Ticket','Cabin','Embarked'], axis=1)
```

In [53]:
```
df.head(5)
```

Out[53]:

	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

In [54]:
```
df.describe()
```

Out[54]:

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

In [55]:
```
df = pd.get_dummies(df, columns=['Sex'], drop_first=True)
df.head()
```

Out[55]:

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

In [56]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 7 columns):
Survived    891 non-null int64
Pclass      891 non-null int64
Age         714 non-null float64
SibSp       891 non-null int64
Parch       891 non-null int64
Fare        891 non-null float64
Sex_male    891 non-null uint8
dtypes: float64(2), int64(4), uint8(1)
memory usage: 42.7 KB
```

In [57]:
```
df['relatives'] = df['SibSp'] + df['Parch']
```

In [58]:
```
df.loc[df['relatives'] > 0, 'not_alone'] = 0
df.loc[df['relatives'] == 0, 'not_alone'] = 1
df['not_alone'] = df['not_alone'].astype(int)
```

In [59]:
```
df['Age'].fillna(df['Age'].median(),inplace=True)
```

In [60]:
```
df['Fare'] = df['Fare'].astype(int)
df['Age'] = df['Age'].astype(int)
```

In [61]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 9 columns):
Survived     891 non-null int64
Pclass       891 non-null int64
Age          891 non-null int32
SibSp        891 non-null int64
Parch        891 non-null int64
Fare         891 non-null int32
Sex_male     891 non-null uint8
relatives    891 non-null int64
not_alone    891 non-null int32
dtypes: int32(3), int64(5), uint8(1)
memory usage: 46.2 KB
```

In [62]:
```
#divide dependent & independent variables
X = df.drop(['Survived'], axis=1).values
y = df['Survived'].values
```

In [63]:
```
# train_test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_s
```

In [64]:
```
#decision tree model building
from sklearn import tree
dtree = tree.DecisionTreeClassifier()
```

In [65]:
```
dtree.fit(X_train, y_train)
y_pred = dtree.predict(X_test)
acc_decision_tree = round(dtree.score(X_train, y_train) * 100, 2)
acc_decision_tree
```

Out[65]:
```
96.15
```

In [66]:
```
from sklearn.metrics import classification_report,confusion_matrix
print(classification_report(y_test,y_pred))
```

```
             precision    recall  f1-score   support

          0       0.79      0.85      0.82       154
          1       0.78      0.70      0.74       114

avg / total       0.79      0.79      0.79       268
```

In [67]:
```
print(confusion_matrix(y_test,y_pred))
```

```
[[131  23]
 [ 34  80]]
```

In [68]:
```
# with the precision & recall score of 0.80 the model predict the Survival accura
```