Name: Rushikesh Jyoti

Division: A

Roll no: 27

SRN: 201901139

Question: Use the TITANIC dataset given for building a classificaion tree prediction model.

In [ ]:
```
import pandas as pd
import seaborn as sb

from matplotlib import pyplot as plot

from sklearn.preprocessing import LabelEncoder
```

In [ ]:
```
df = pd.read_csv('./titanic_data-F.csv', na_values='?')

print(df.columns)
df.head()
```

```
Index(['x', 'pclass', 'survived', 'name', 'sex', 'age', 'sibsp', 'parch',
       'ticket', 'fare', 'cabin', 'embarked', 'home.dest'],
      dtype='object')
```

Out[ ]:

	x	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	home.d
0	1	1	1	Allen, Miss. Elisabeth Walton	female	29.0000	0	0	24160	211.3375	B5	S	St Lo
1	2	1	1	Allison, Master. Hudson Trevor	male	0.9167	1	2	113781	151.5500	C22 C26	S	Montr P Chesterv
2	3	1	0	Allison, Miss. Helen Loraine	female	2.0000	1	2	113781	151.5500	C22 C26	S	Montr P Chesterv
3	4	1	0	Allison, Mr. Hudson Joshua Creighton	male	30.0000	1	2	113781	151.5500	C22 C26	S	Montr P Chesterv
4	5	1	0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0000	1	2	113781	151.5500	C22 C26	S	Montr P Chesterv

Checking for NA Values

In [ ]:
```
df.isna().sum()
```

Out[ ]:
```
x              0
pclass         0
survived       0
name           0
sex            0
age          263
sibsp          0
parch          0
ticket         0
fare           1
cabin       1014
embarked       2
home.dest    564
dtype: int64
```

We have lots of NA values :O

In [ ]:
```
df.dropna(inplace=True)
```

In [ ]:
```
df.isna().sum()
```

Out[ ]:
```
x            0
pclass       0
survived     0
name         0
sex          0
age          0
sibsp        0
parch        0
ticket       0
fare         0
cabin        0
embarked     0
home.dest    0
dtype: int64
```

~~No NAs?~~

In [ ]:
```
df.dtypes
```

Out[ ]:
```
x              int64
pclass         int64
survived       int64
name          object
sex           object
age          float64
sibsp          int64
parch          int64
ticket        object
fare         float64
cabin         object
embarked      object
home.dest     object
dtype: object
```

Drop uneccesary columns

In [ ]:
```
df.drop(["x", "name", "home.dest", "ticket"], axis=1, inplace=True)
```

Encoding categorical data

In [ ]:
```
df["sex"] = LabelEncoder().fit_transform(df["sex"])
df["cabin"] = LabelEncoder().fit_transform(df["cabin"])
df["embarked"] = LabelEncoder().fit_transform(df["embarked"])

df.head()
```

Out[ ]:

	pclass	survived	sex	age	sibsp	parch	fare	cabin	embarked
0	1	1	0	29.0000	0	0	211.3375	37	2
1	1	1	1	0.9167	1	2	151.5500	63	2
2	1	0	0	2.0000	1	2	151.5500	63	2
3	1	0	1	30.0000	1	2	151.5500	63	2
4	1	0	0	25.0000	1	2	151.5500	63	2

Correlation Heatmap

In [ ]:
```
# sb.set({'font_scale': 1.6})
sb.set(font_scale=1.6)
```

In [ ]:
```
plot.figure(1, (15, 8))
sb.heatmap(df.corr(), annot=True)
```

Out[ ]:
```
<AxesSubplot:>
```

Splitting the dataset

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

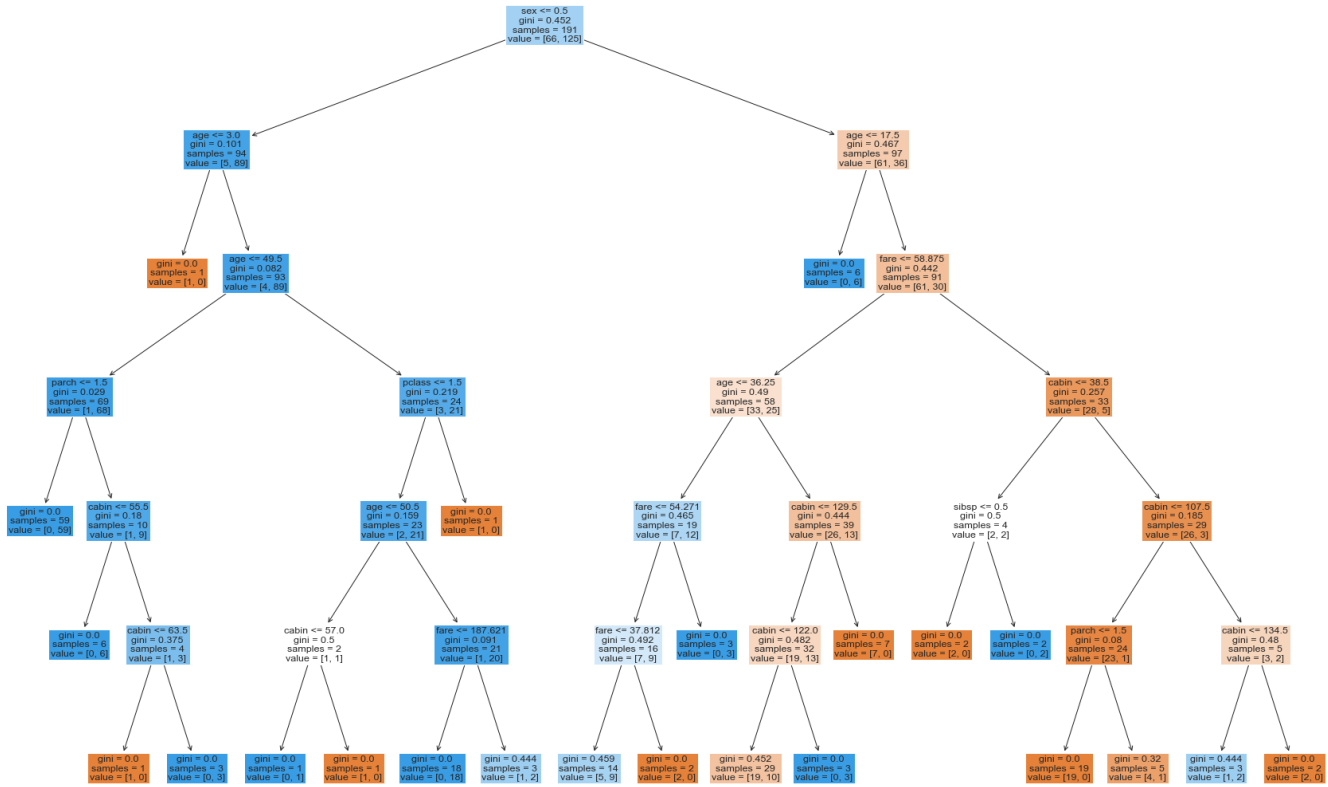
In [ ]:
```
xtrain, xtest, ytrain, ytest = train_test_split(df.drop(["survived"], axis=1), df[["survived
```

Building the Decision Tree Calssifer

In [ ]:
```
from sklearn.tree import DecisionTreeClassifier, plot_tree
```

In [ ]:
```
model = DecisionTreeClassifier(max_depth=6).fit(xtrain, ytrain)
```

In [ ]:
```
plot.figure(1, (30, 20))
_ = plot_tree(model,
    feature_names=xtest.columns,
    filled=True,
    impurity=True
)
```



In [ ]:
```
predictions = model.predict(xtest)

predictions
```

Out[ ]:
```
array([1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0,
       1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1,
       0, 1, 1, 1], dtype=int64)
```

Metrics

In [ ]:
```
from sklearn import metrics
```

Classification Report

In [ ]:
```
print(metrics.classification_report(ytest, predictions))
```

	precision	recall	f1-score	support
0	0.50	0.60	0.55	15
1	0.80	0.73	0.76	33
accuracy			0.69	48
macro avg	0.65	0.66	0.65	48
weighted avg	0.71	0.69	0.69	48

In [ ]:

```
print(f"Accuracy: {metrics.accuracy_score(ytest, predictions)}")
print(f"Mean Absolute Error: {metrics.mean_absolute_error(ytest, predictions)}")
print(f"Mean Squared Error: {metrics.mean_squared_error(ytest, predictions)}")
print(f"R2: {metrics.r2_score(ytest, predictions)}")
```

```
Accuracy: 0.6875
Mean Absolute Error: 0.3125
Mean Squared Error: 0.3125
R2: -0.4545454545454546
```