Decision Trees and Random Forests

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Get the Data

```
df = pd.read_csv('kyphosis.csv')
```

```
df.head()
```

	Kyphosis	Age	Number	Start
0	absent	71	3	5
1	absent	158	3	14
2	present	128	4	5
3	absent	2	5	1
4	absent	1	4	15

Train Test Split

```
from sklearn.model_selection import train_test_split
```

```
X = df.drop('Kyphosis',axis=1)
y = df['Kyphosis']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30)
```

Decision Trees

```
from sklearn.tree import DecisionTreeClassifier
```

```
dtree = DecisionTreeClassifier()
```

```
dtree.fit(X_train,y_train)
```

```
DecisionTreeClassifier()
```

Prediction and Evaluation

```
predictions = dtree.predict(X_test)
```

```
from sklearn.metrics import classification_report,confusion_matrix
```

```
print(classification_report(y_test,predictions))
```

```
                  precision    recall  f1-score   support

          absent       0.94      0.73      0.82        22
         present       0.25      0.67      0.36         3

        accuracy                           0.72        25
       macro avg       0.60      0.70      0.59        25
    weighted avg       0.86      0.72      0.77        25
```

```
print(confusion_matrix(y_test,predictions))
```

```
[[16  6]
 [ 1  2]]
```

Tree Visualization

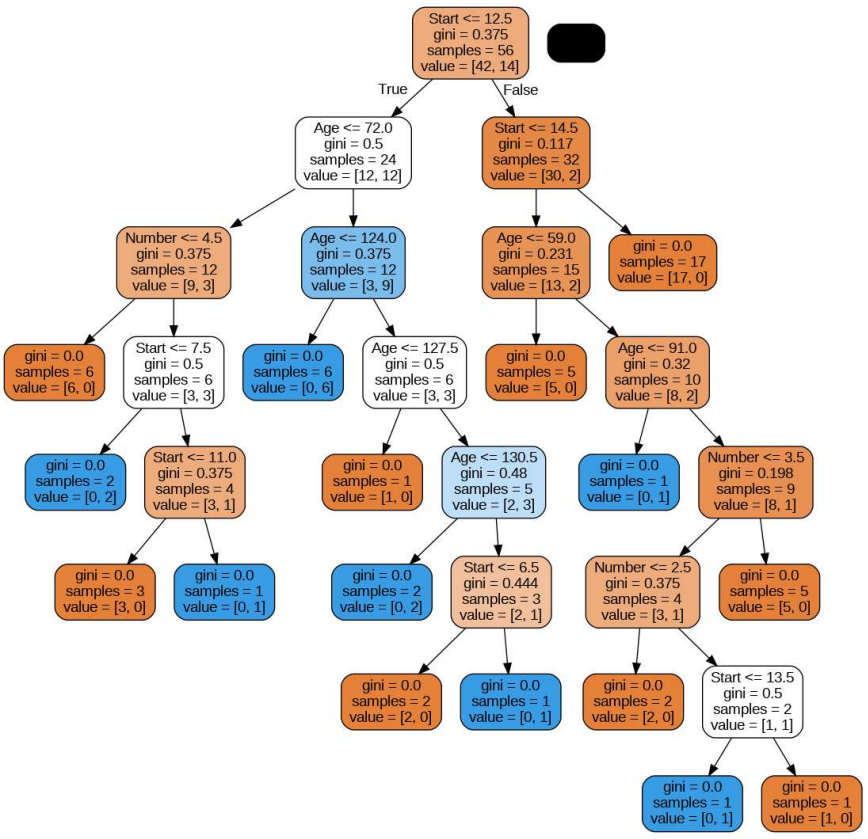
```
from IPython.display import Image
from six import StringIO
from sklearn.tree import export_graphviz
import pydot

features = list(df.columns[1:])
features
```

```
['Age', 'Number', 'Start']
```

```
dot_data = StringIO()
export_graphviz(dtree, out_file=dot_data,feature_names=features,filled=True,rounded=True)

graph = pydot.graph_from_dot_data(dot_data.getvalue())
Image(graph[0].create_png())
```



Random Forests

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=100)
rfc.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier()
```

```
rfc_pred = rfc.predict(X_test)
```

```
print(confusion_matrix(y_test,rfc_pred))
```

```
[[21  1]
 [ 2  1]]
```

```
print(classification_report(y_test,rfc_pred))
```

```
                  precision    recall  f1-score   support

          absent       0.91      0.95      0.93        22
         present       0.50      0.33      0.40         3

        accuracy                           0.88        25
       macro avg       0.71      0.64      0.67        25
    weighted avg       0.86      0.88      0.87        25
```

Pronto!