```
import pandas as pd
import tensorflow as tf
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
from tensorflow import keras
from tensorflow.keras import layers
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_absolute_error, r2_score, mean_squared_error
from sklearn.metrics import accuracy_score
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt
```

```
df = pd.read_csv('/content/titanic.csv')
```

```
df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarke
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	male	35.0	0	0	373450	8.0500	NaN	

Next steps: Generate code with df   View recommended plots

```
df.isnull().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

```
df = df.dropna()
```

```
df.isnull().sum()
```

```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin          0
Embarked       0
dtype: int64
```

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

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

```
cols_to_drop = [
    'PassengerId',
    'Name',
    'Ticket',
    'Cabin',
    'Embarked',
]

df = df.drop(cols_to_drop, axis=1)
df.head()
```

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare
1	1	1	female	38.0	1	0	71.2833
3	1	1	female	35.0	1	0	53.1000
6	0	1	male	54.0	0	0	51.8625
10	1	3	female	4.0	1	1	16.7000
11	1	1	female	58.0	0	0	26.5500

Next steps: Generate code with `df` View recommended plots

```
sex_mapping = {
    'male' : 0,
    'female' : 1
}
df.Sex = df.Sex.map(sex_mapping)

df.head()
```

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare
1	1	1	1	38.0	1	0	71.2833
3	1	1	1	35.0	1	0	53.1000
6	0	1	0	54.0	0	0	51.8625
10	1	3	1	4.0	1	1	16.7000
11	1	1	1	58.0	0	0	26.5500

Next steps: Generate code with df View recommended plots

```
X = df.drop(columns=['Survived'])
y = df['Survived']
```

```
print(X.shape,y.shape)
```

```
(183, 6) (183,)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
dt_classifier = DecisionTreeClassifier()
decision_tree_history = dt_classifier.fit(X_train, y_train)

decision_tree_history
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
y_pred = dt_classifier.predict(X_test)
```

```
y_pred
```

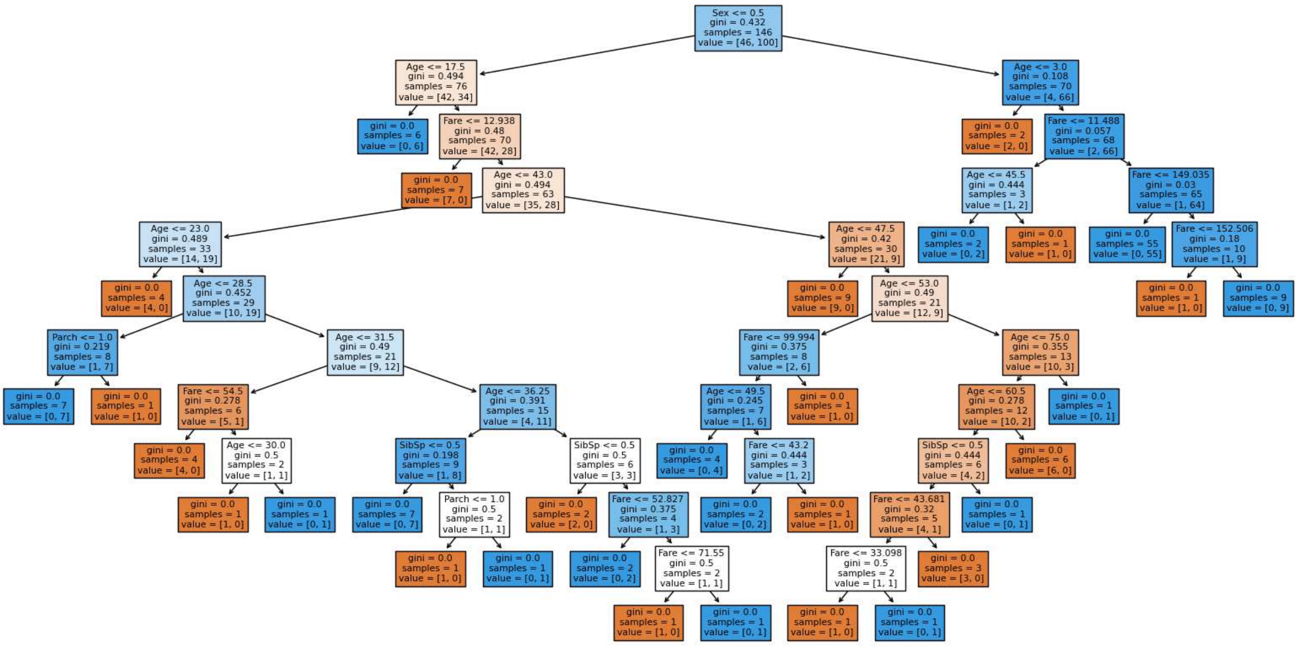
```
array([1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1,
       0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1])
```

```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy of the classifier is:", accuracy)
```

```
Accuracy of the classifier is: 0.7567567567567568
```

The accuracy with decision tree classifier is 75%.

```
plt.figure(figsize=(20,10))
plot_tree(dt_classifier, filled=True, feature_names=X.columns)
plt.show()
```



```
#fitting the model
regressor = DecisionTreeRegressor(criterion = 'absolute_error', max_depth = 15, max_features= 'log2', random_state =
decision_tree_regressor_history_2 = regressor.fit(X_train, y_train)
```

```
y_pred_test = regressor.predict(X_test)
y_pred_test
```

```
array([1., 0., 1., 0., 0., 0., 1., 1., 1., 1., 0., 1., 0., 1., 1., 0., 1.,
       0., 0., 1., 1., 1., 0., 1., 0., 1., 1., 0., 1., 1., 1., 1., 1., 1.,
       0., 1., 1.])
```

```
mae_test = mean_absolute_error(y_test, y_pred_test)
print("The MAE is:", mae_test)

mse = mean_squared_error(y_test, y_pred_test)
print("The MSE is:", mse)

r_square = r2_score(y_test, y_pred_test)
print("R_square is:",r_square)
```

```
The MAE is: 0.2972972972972973
The MSE is: 0.2972972972972973
R_square is: -0.2639751552795033
```

```
accuracy = accuracy_score(y_test, y_pred_test)
print("Accuracy of the decision tree regressor with hyperparameters is:", accuracy)
```

```
Accuracy of the decision tree regressor with hyperparameters is: 0.7027027027027027
```

```
feature_normalizer = tf.keras.layers.Normalization()

deep_mlp_ann_model = tf.keras.Sequential([
    feature_normalizer,
    layers.Dense(units=128, activation='relu', input_shape=(X_train.shape[1],)),
    layers.Dense(units=64, activation='relu'),
    layers.Dense(units=32, activation='relu'),
    layers.Dense(units=1, activation='sigmoid')
])

deep_mlp_ann_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

```
deep_mlp_model_2 = tf.keras.Sequential([
    feature_normalizer,
    layers.Dense(units=256, activation='relu', input_shape=(X_train.shape[1],)),
    layers.Dense(units=128, activation='relu'),
    layers.Dense(units=64, activation='relu'),
    layers.Dense(units=1, activation='sigmoid')
])

deep_mlp_model_2.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

deep_mlp_history_2 = deep_mlp_model_2.fit(X_train, y_train, epochs=50, batch_size=64, validation_split=0.2, verbose

test_loss, test_acc = deep_mlp_model_2.evaluate(X_test, y_test)
print('The accuracy of the DNN with hyperparameters is:', test_acc)
```

```
2/2 [==============================] - 0s 13ms/step - loss: 0.6149 - accuracy: 0.6486
The accuracy of the DNN with hyperparameters is: 0.6486486196517944
```

According to the accuracy, it seems like the model is performing better on decision tree classifier with 75% accuracy. Though we passed the hyperparameters for DNN it performed only an accuracy of 64%. The decision tree regressor performed an accuraccy of 70%.

```
Start coding or generate with AI.
```