Preprocessing the data

nasa.columns

#creating a Pandas dataframe

column\_names = ['time\_in\_cycles', 'Altitud', 'Mach Number', 'TRA', 'T2', 'T24', 'T30',

       'T50', 'P2', 'P15', 'P30', 'Nf', 'Nc', 'epr', 'Ps30', 'phi', 'NRf',

       'NRc', 'BPR', 'farB', 'htBleed', 'Nf\_dmd', 'PCNfR\_dmd', 'W31', 'W32',

       'label\_target']

df = pd.DataFrame(nasa, columns=column\_names)

#importing libraries

from sklearn.compose import make\_column\_transformer

from sklearn.preprocessing import OrdinalEncoder, LabelEncoder

feature\_names = column\_names[:-1]

label\_name = column\_names[-1]

X\_preprocess = make\_column\_transformer((OrdinalEncoder(), feature\_names),

                                       remainder='drop')

y\_preprocess = LabelEncoder()

X = X\_preprocess.fit\_transform(df[feature\_names])

y = y\_preprocess.fit\_transform(df[label\_name])

#data in Numpy arrays

display(X, y)

#Import Libraries

from sklearn.tree import DecisionTreeClassifier

#Recall best estimator hyperparameters

cv.best\_estimator\_

dtree = DecisionTreeClassifier(ccp\_alpha=0.0, class\_weight=None, criterion='entropy',

                       max\_depth=16, max\_features=None, max\_leaf\_nodes=None,

                       min\_impurity\_decrease=0.0, min\_impurity\_split=None,

                       min\_samples\_leaf=20, min\_samples\_split=15,

                       min\_weight\_fraction\_leaf=0.0, presort='deprecated',

                       random\_state=None, splitter='best')

#fitting the tree

dtree.fit(X, y)

yhat\_train = dtree.predict(X\_train)

display(yhat\_train)

#display labels as they were before processing

label\_names = y\_preprocess.inverse\_transform(np.arange(y.max() + 1))

display(label\_names[yhat\_train])

# Visualizing the Decision Tree

#Import Libraries

from sklearn.tree import export\_graphviz

#Plot Tree

import graphviz

from sklearn.tree import export\_graphviz

dot\_data = export\_graphviz(dtree,

                           out\_file=None,

                           class\_names=class\_names.tolist(),

                           feature\_names=feature\_names,

                           filled=True,

                           rounded=True,

                           special\_characters=True,

                           rotate= False)

display(graphviz.Source(dot\_data))

Comparison

Normalizing dataset

MinMax\_norm = MinMaxScaler().fit(val\_features)

val\_features\_norm = MinMax\_norm.transform(val\_features)

MinMax\_norm = MinMaxScaler().fit(val\_labels)

val\_labels\_norm = MinMax\_norm.transform(val\_labels)

Old code for Defining Target – not needed for processed dataset

# #defining a new target variable based on a minimum threshold of 25(Data already processed)

# target = 25

# label\_positive =nasa['target'] <= target

# nasa['label\_target']=0

# nasa.loc[label\_positive,'label\_target'] = 1

# #Unit number not likely to be relevant to the process, also condition is just the data set #

# nasa.drop(columns=['max\_cycles','target','unit\_number','condition'],inplace = True)