import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

#for encoding

from sklearn.preprocessing import LabelEncoder

#for train test splitting

from sklearn.model\_selection import train\_test\_split

#for decision tree object

from sklearn.tree import DecisionTreeClassifier

#for checking testing results

from sklearn.metrics import classification\_report, confusion\_matrix

#for visualizing tree

from sklearn.tree import plot\_tree

#reading the data

df = sns.load\_dataset('iris')

df.head()

df.info()

df.isnull().any()

df.shape

# let's plot pair plot to visualise the attributes all at once

sns.pairplot(data=df, hue = 'species')

plt.savefig("pne.png")

# correlation matrix

sns.heatmap(df.corr())

plt.savefig("one.png")

target = df['species']

df1 = df.copy()

df1 = df1.drop('species', axis =1)

df1.shape

df1.head()

X = df1

Target

#label encoding

le = LabelEncoder()

target = le.fit\_transform(target)

target

y = target

# Splitting the data - 80:20 ratio

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X , y, test\_size = 0.2, random\_state = 42)

print("Training split input- ", X\_train.shape)

print("Testing split input- ", X\_test.shape)

# Defining the decision tree algorithm

dtree=DecisionTreeClassifier()

dtree.fit(X\_train,y\_train)

print('Decision Tree Classifer Created')

# Predicting the values of test data

y\_pred = dtree.predict(X\_test)

print("Classification report - \n", classification\_report(y\_test,y\_pred))

cm = confusion\_matrix(y\_test, y\_pred)

plt.figure(figsize=(5,5))

sns.heatmap(data=cm,linewidths=.5, annot=True,square = True, cmap = 'Blues')

plt.ylabel('Actual label')

plt.xlabel('Predicted label')

all\_sample\_title = 'Accuracy Score: {0}'.format(dtree.score(X\_test, y\_test))

plt.title(all\_sample\_title, size = 15)

plt.savefig("one.png")

# Visualising the graph without the use of graphviz

plt.figure(figsize = (20,20))

dec\_tree = plot\_tree(decision\_tree=dtree, feature\_names = df1.columns,

class\_names =["setosa", "vercicolor", "verginica"] , filled = True , precision = 4, rounded = True)

plt.savefig("one.png")