**Practical 10**

**Implementation of KNN Clustering on Jupyter Notebook using Python.**

import numpy as np

import matplotlib.pyplot as plt

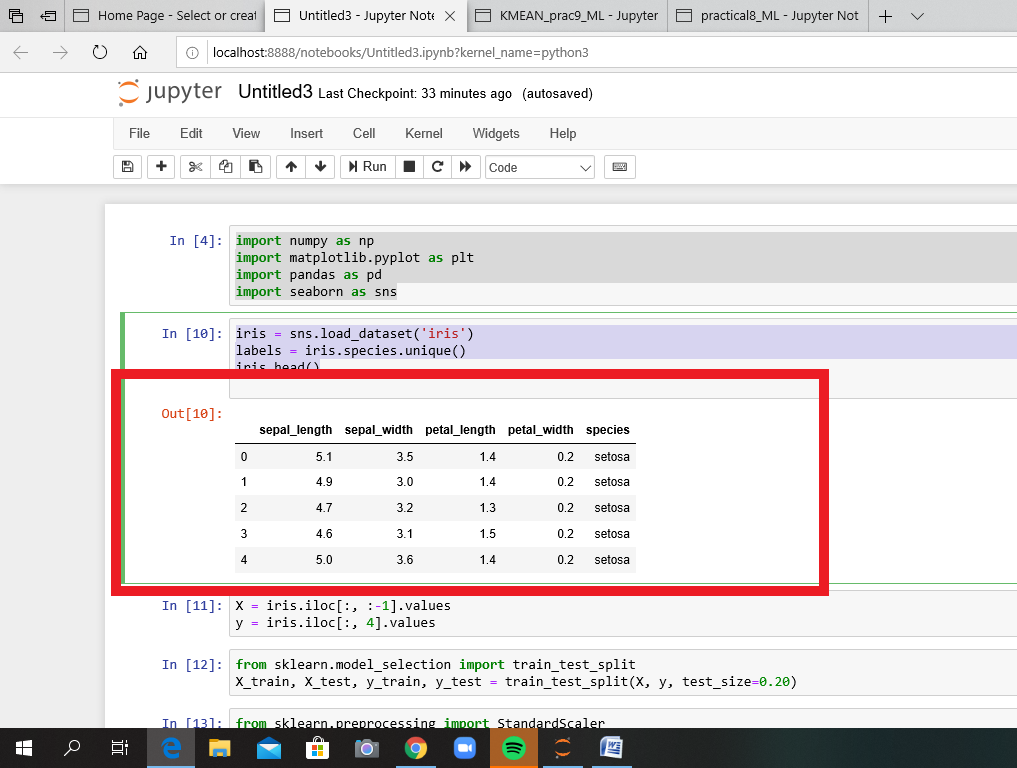
import pandas as pd

import seaborn as sns

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

labels = iris.species.unique()

iris.head() **#Output : Fig.10.1**



*Fig.10.1 iris.head()*

X = iris.iloc[:, :-1].values

y = iris.iloc[:, 4].values

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20)

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaler.fit(X\_train)

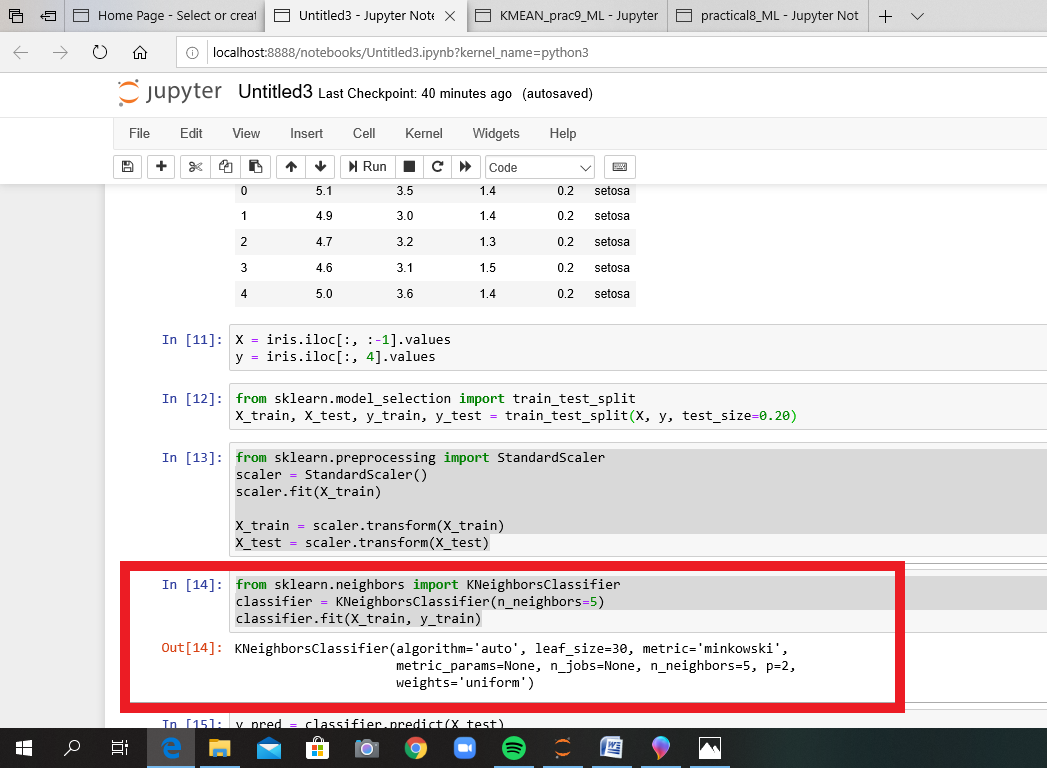
X\_train = scaler.transform(X\_train)

X\_test = scaler.transform(X\_test)

from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n\_neighbors=5)

classifier.fit(X\_train, y\_train) **#Output: Fig10.2**



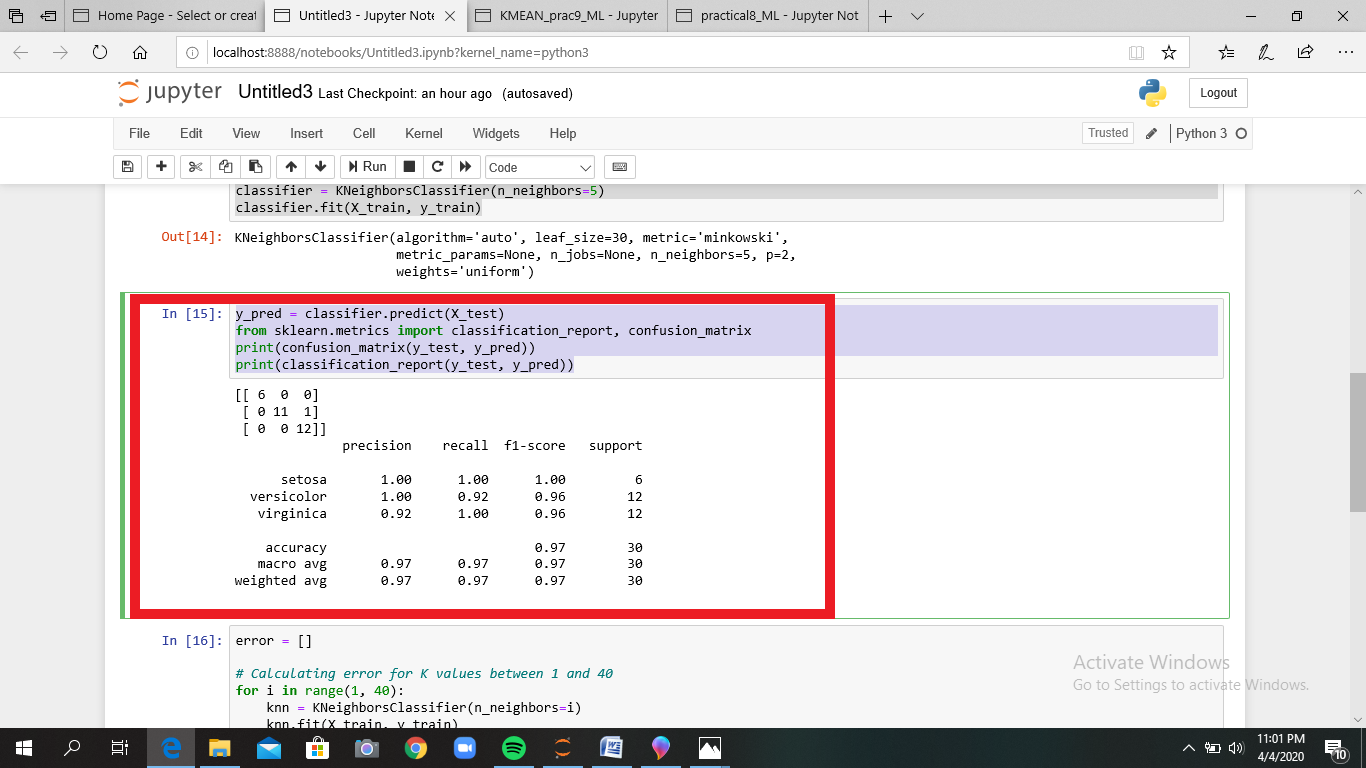
*Fig.10.2 KNeighbours Classifier*

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

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

print(confusion\_matrix(y\_test, y\_pred))

print(classification\_report(y\_test, y\_pred)) **#Output : Fig.10.3**



*Fig.10.3 Classification\_report*

error = []

# Calculating error for K values between 1 and 40

for i in range(1, 40):

knn = KNeighborsClassifier(n\_neighbors=i)

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

pred\_i = knn.predict(X\_test)

error.append(np.mean(pred\_i != y\_test))

plt.figure(figsize=(12, 6))

plt.plot(range(1, 40), error, color='red', linestyle='dashed', marker='o',

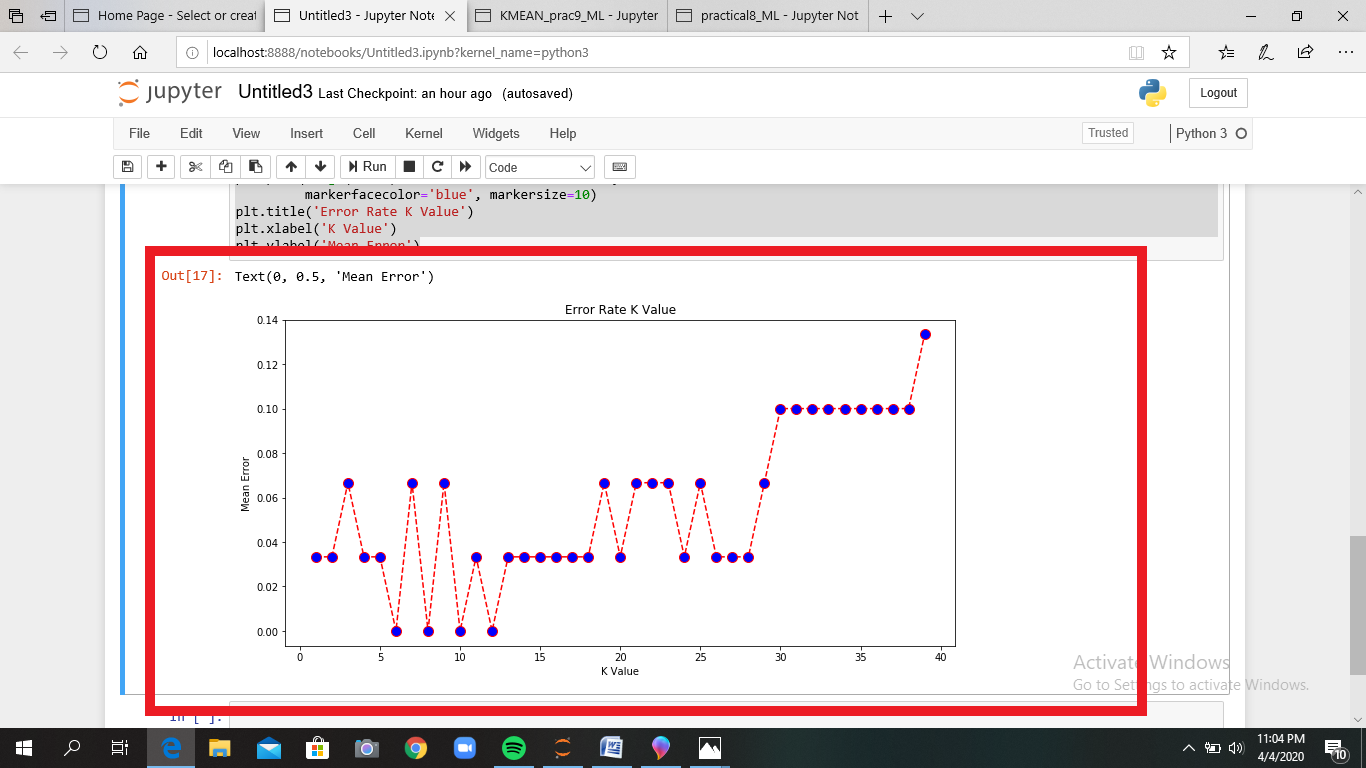
markerfacecolor='blue', markersize=10)

plt.title('Error Rate K Value')

plt.xlabel('K Value')

plt.ylabel('Mean Error')

**Result Output :**



*Fig.10.4 Result Output(Error Rate)*