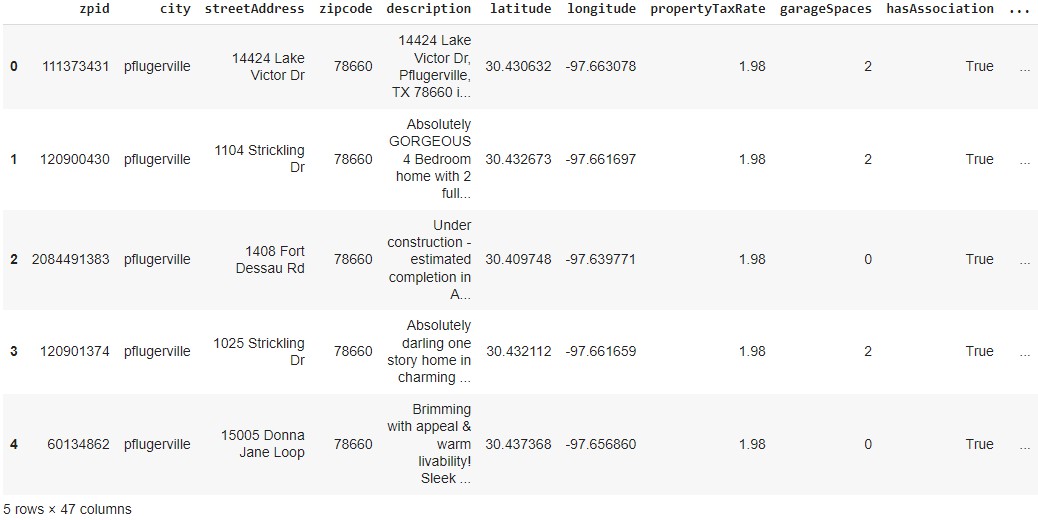
1. Write a python program to import and export data using Pandas library functions.

Code

import pandas as pd df=pd.read\_csv("/content/austinHousingData.csv")

df.head(5)

Output

1. Demonstrate various data pre-processing techniques for a given dataset.

Code

url = “https://archive.ics.uci.edu/ml/machine-learning-

databases/iris/iris.data”

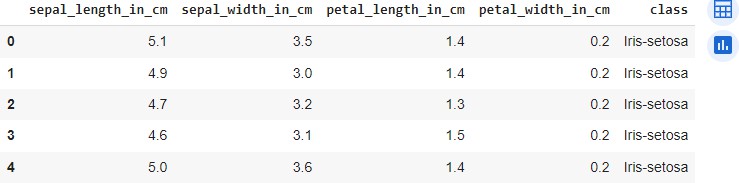
# Define the column names col\_names = [“sepal\_length\_in\_cm”,

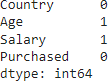
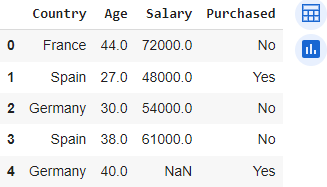
“sepal\_width\_in\_cm”, “petal\_length\_in\_cm”, “petal\_width\_in\_cm”, “class”]

# Read data from URL

iris\_data = pd.read\_csv(url, names=col\_names) iris\_data.head(5) iris\_data.to\_csv(“/content/exported\_irisData.csv”)

Output:





%matplotlib inline import numpy as np import pandas as pd

import matplotlib.pyplot as plt import seaborn as sns

import sklearn df1=pd.read\_csv(“/content/Data.csv”)

df1.head(5)

#Identifying and handling the missing values

df1.isnull().sum()

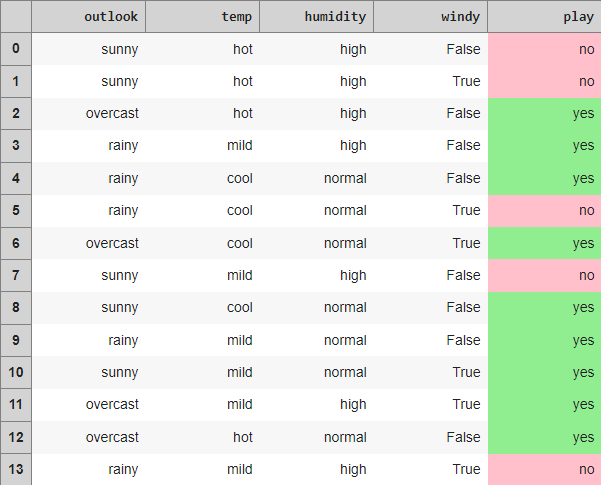
1. Use an appropriate dataset for building the decision tree(ID3) and apply this knowledge to classify a new sample.

Code

# Importing the required libraries import pandas as pd

import numpy as np import math

data = pd.read\_csv('/content/PlayTennis.csv')



def highlight(cell\_value): '''

Highlight yes / no values in the dataframe '''

color\_1 = 'background-color: pink;' color\_2 = 'background-color: lightgreen;'

if cell\_value == 'no': return color\_1

elif cell\_value == 'yes': return color\_2

data.style.applymap(highlight)\

.set\_properties(subset=data.columns, \*\*{'width': '100px'})\

.set\_table\_styles([{'selector': 'th', 'props': [('background- color', 'lightgray'), ('border', '1px solid gray'),

('font-weight',

'bold')]},

{'selector': 'tr:hover', 'props': [('background-color', 'white'), ('border', '1.5px solid black')]}])

def find\_entropy(data): """

Returns the entropy of the class or features formula: - ∑ P(X)logP(X)

"""

entropy = 0

for i in range(data.nunique()):

x = data.value\_counts()[i]/data.shape[0] entropy += (- x \* math.log(x,2))

return round(entropy,3)

def information\_gain(data, data\_): """

Returns the information gain of the features """

info = 0

for i in range(data\_.nunique()):

df = data[data\_ == data\_.unique()[i]] w\_avg = df.shape[0]/data.shape[0] entropy = find\_entropy(df.play)

x = w\_avg \* entropy info += x

ig = find\_entropy(data.play) - info return round(ig, 3)

def entropy\_and\_infogain(datax, feature): """

Grouping features with the same class and computing their entropy and information gain for splitting

"""

for i in range(data[feature].nunique()):

df = datax[datax[feature]==data[feature].unique()[i]] if df.shape[0] < 1:

continue

display(df[[feature, 'play']].style.applymap(highlight)\

.set\_properties(subset=[feature, 'play'], \*\*{'width':

'80px'})\

.set\_table\_styles([{'selector': 'th', 'props':

[('background-color', 'lightgray'), r', '1px solid gray'),

weight', 'bold')]},

[('border', '1px solid gray')]},

{'selector': 'td', 'props':

('borde ('font-

{'selector': 'tr:hover', 'props':

[('background-color', 'white'),

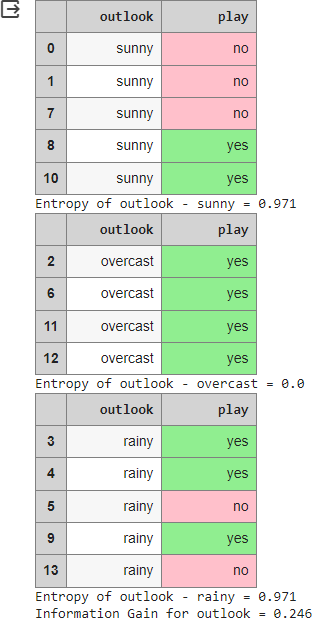
(

'border', '1.5px solid black')]}]))

print(f'Entropy of {feature} - {data[feature].unique()[i]} =

{find\_entropy(df.play)}')

print(f'Information Gain for {feature} = {information\_gain(datax, datax[feature])}')



#Computing entropy for the entire dataset

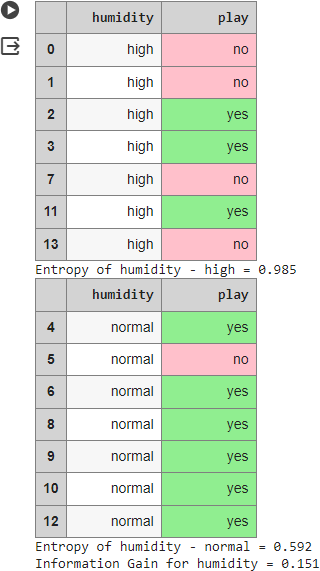
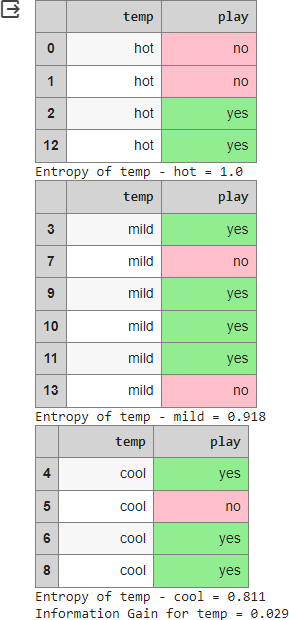
print(f'Entropy of the entire dataset: {find\_entropy(data.play)}')

#Calculate the Information Gain for each feature. #Outlook

entropy\_and\_infogain(data, 'outlook')

#Temp

entropy\_and\_infogain(data, 'temp')

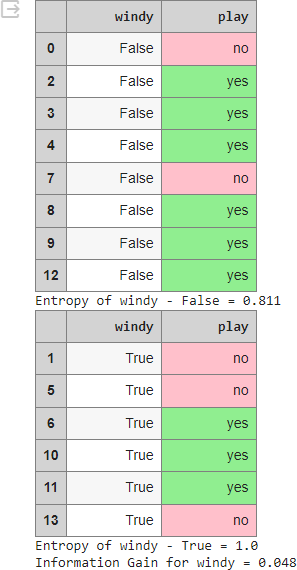


#Humidity

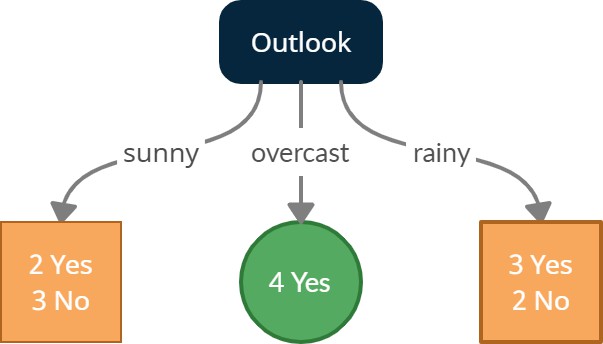
entropy\_and\_infogain(data, 'humidity')

#Windy

entropy\_and\_infogain(data, 'windy')



#Make a decision tree node using the feature with the maximum Information Gain.



sunny = data[data['outlook'] == 'sunny'] sunny.style.applymap(highlight)\

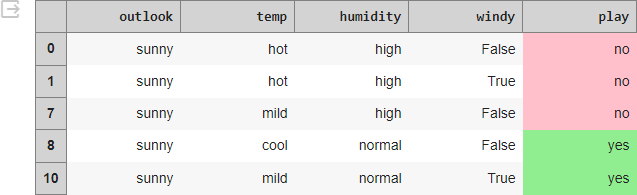
.set\_properties(subset=data.columns, \*\*{'width': '100px'})\

.set\_table\_styles([{'selector': 'th', 'props': [('background- color', 'lightgray'), ('border', '1px solid gray'),

('font-weight',

'bold')]},

{'selector': 'tr:hover', 'props': [('background-color', 'white'), ('border', '1.5px solid black')]}])

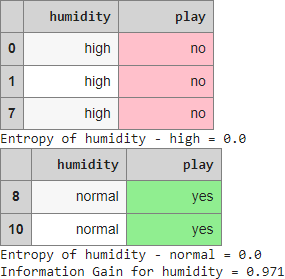


print(f'Entropy of the Sunny dataset: {find\_entropy(sunny.play)}')

#temp

entropy\_and\_infogain(sunny, 'temp')



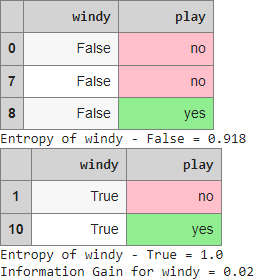


#Windy

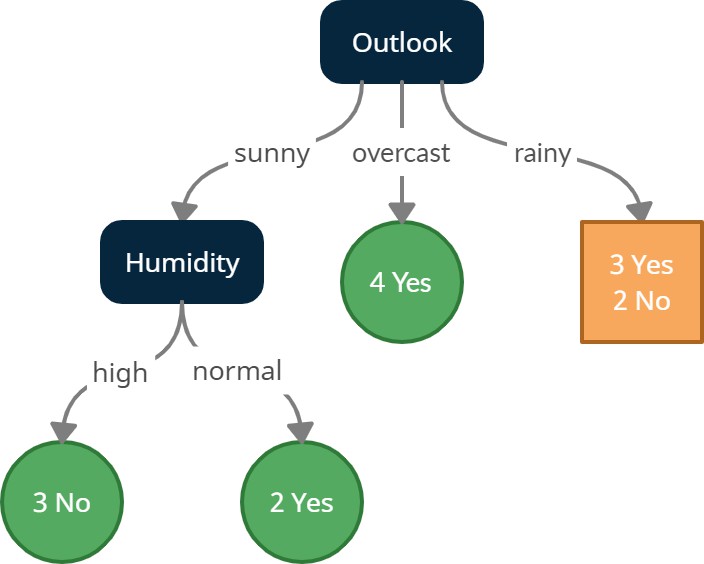
entropy\_and\_infogain(sunny, 'windy')

#Humidity

entropy\_and\_infogain(sunny, 'humidity')



#Making a decision tree node using the feature which has the maximum Information Gain



#Outlook - Rainy

rainy = data[data['outlook'] == 'rainy'] rainy.style.applymap(highlight)\

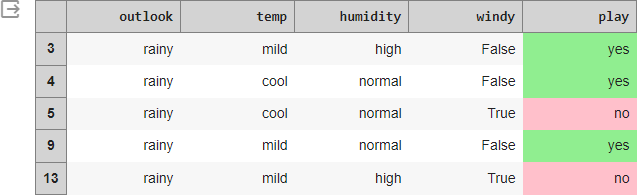
.set\_properties(subset=data.columns, \*\*{'width': '100px'})\

.set\_table\_styles([{'selector': 'th', 'props': [('background- color', 'lightgray'), ('border', '1px solid gray'),

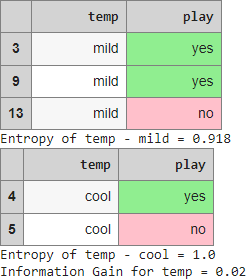
('font-weight',

'bold')]},

{'selector': 'tr:hover', 'props': [('background-color', 'white'), ('border', '1.5px solid black')]}])



print(f'Entropy of the Rainy dataset: {find\_entropy(rainy.play)}')

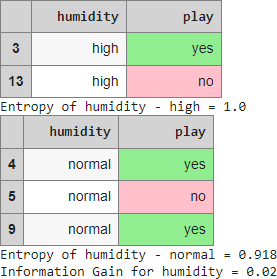


#temp

entropy\_and\_infogain(rainy, 'temp')

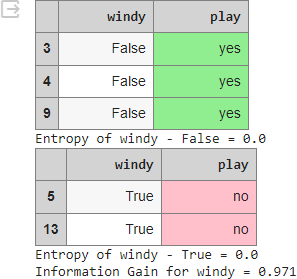
#Humidity

entropy\_and\_infogain(rainy, 'humidity')

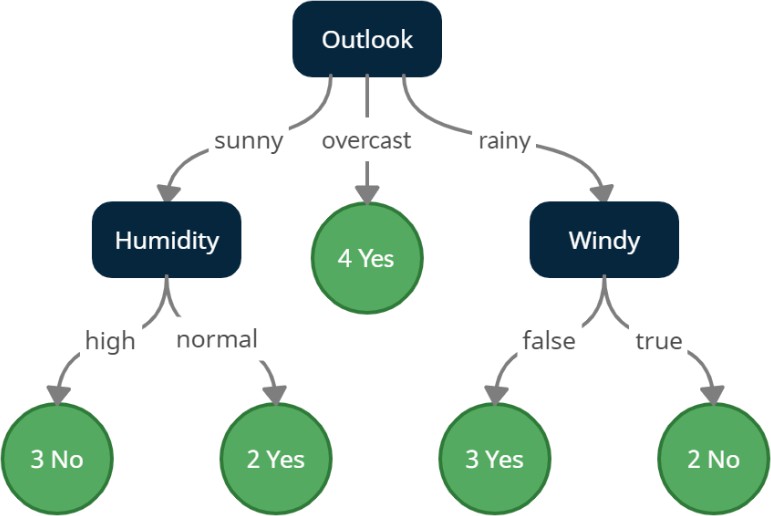


#Windy

entropy\_and\_infogain(rainy, 'windy')

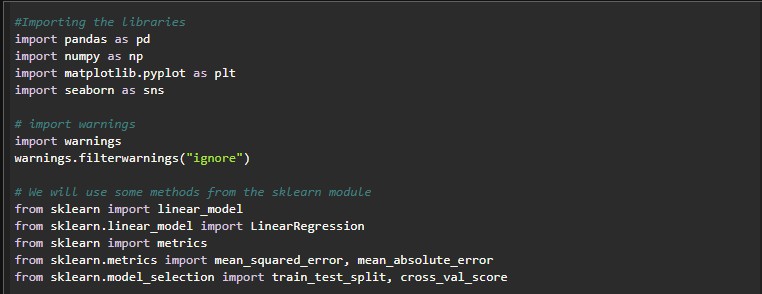


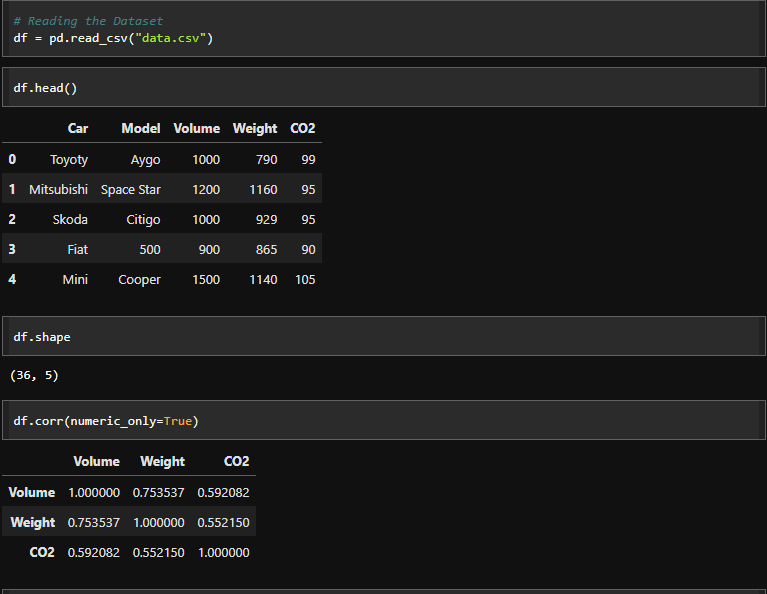
#Making a decision tree node using the feature which has the maximum Information Gain.

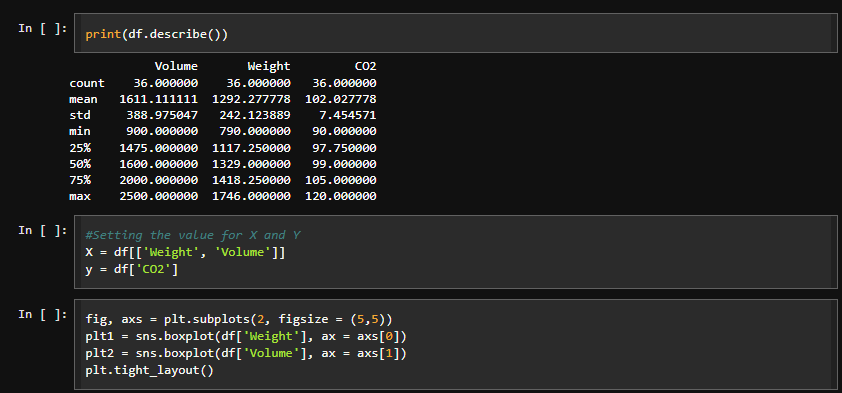


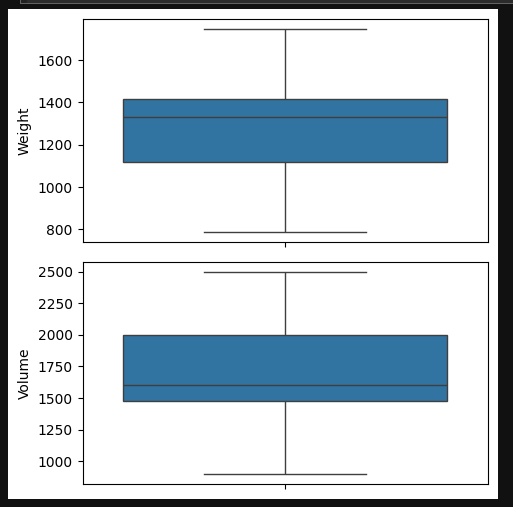
1. Implement Linear and Multi-Linear Regression algorithm using appropriate dataset

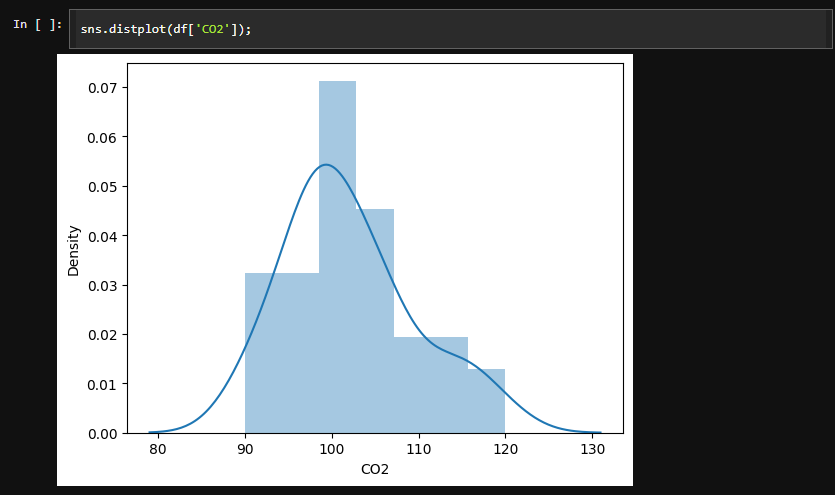
**Code:**

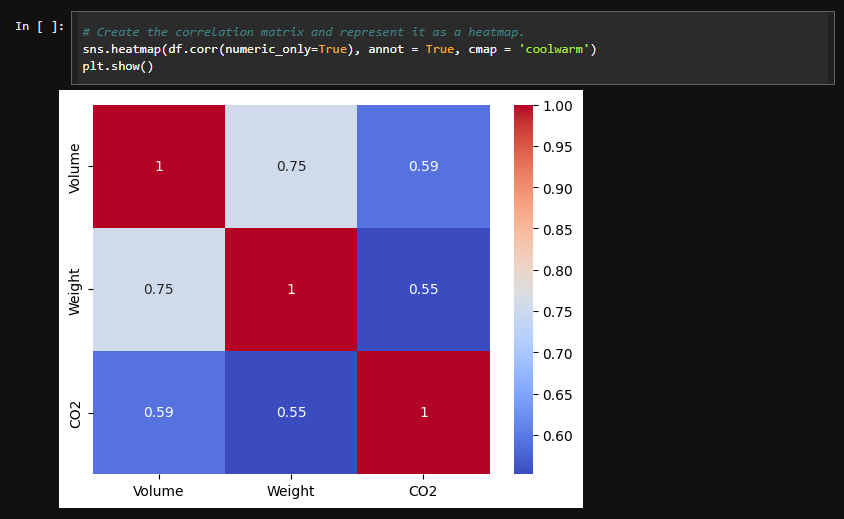
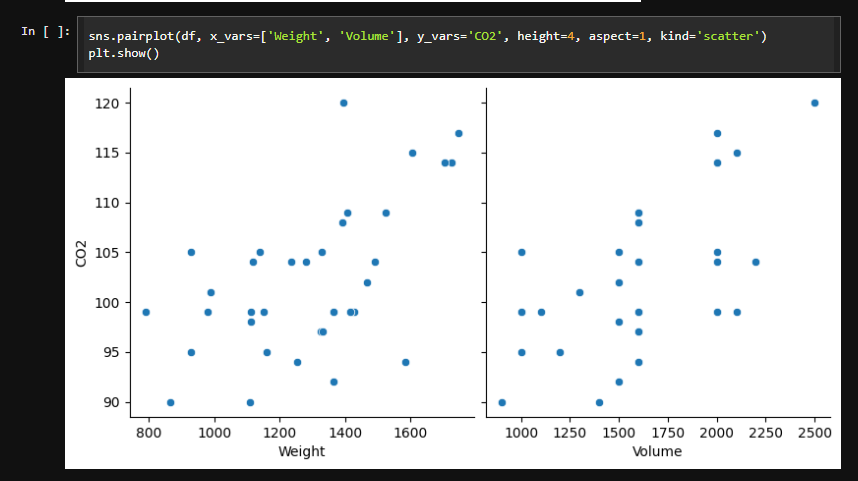


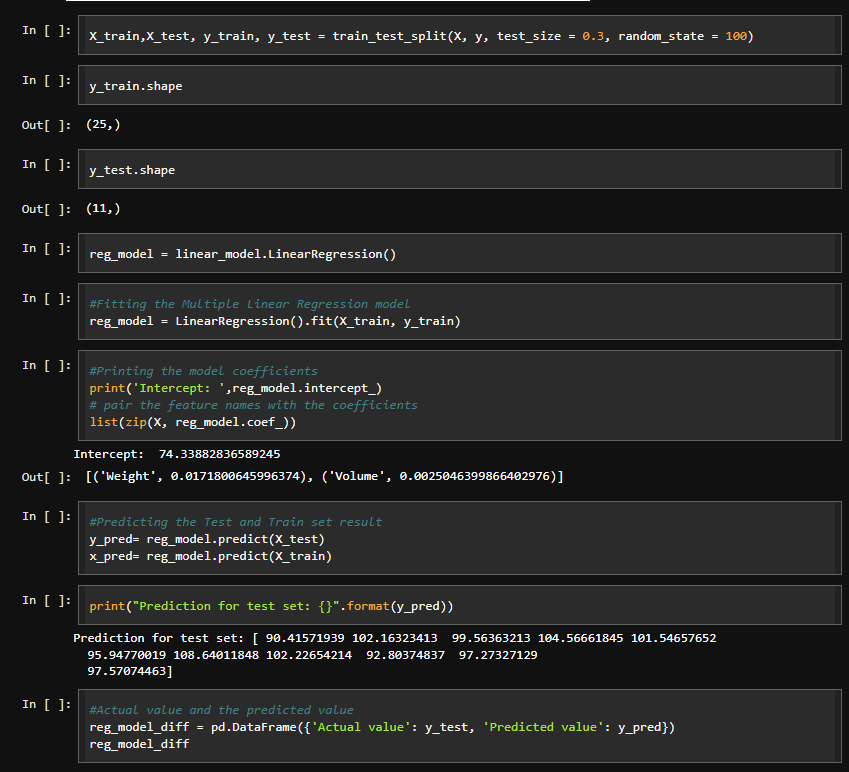


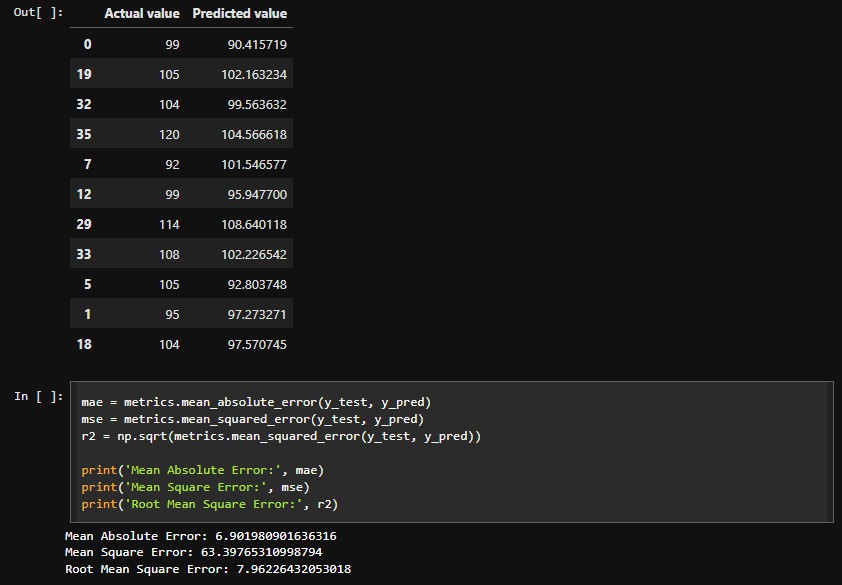










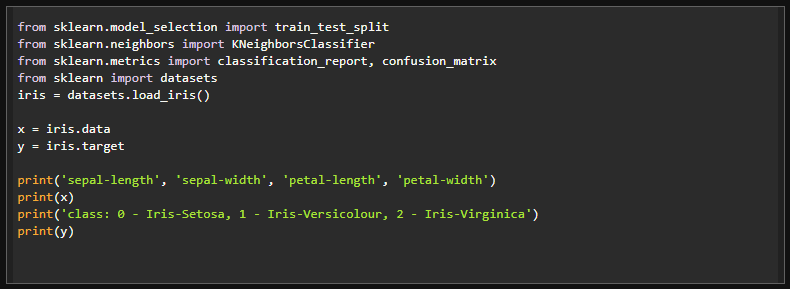


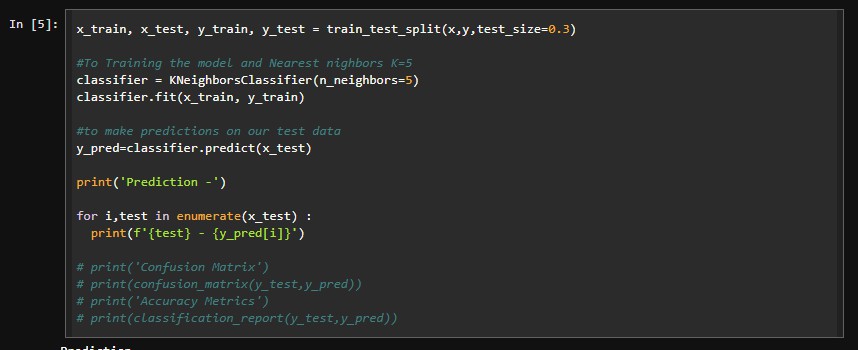
**Results**



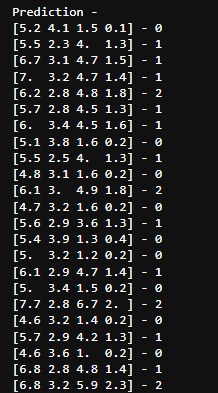
1. Build KNN Classification model for a given dataset.

Code:



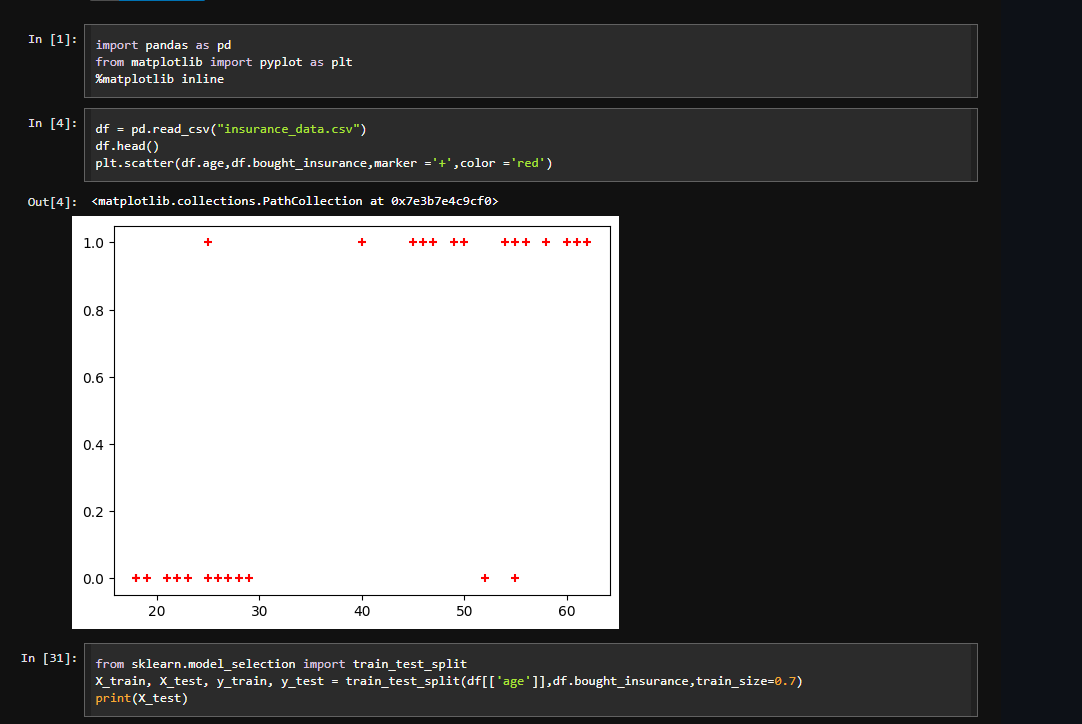


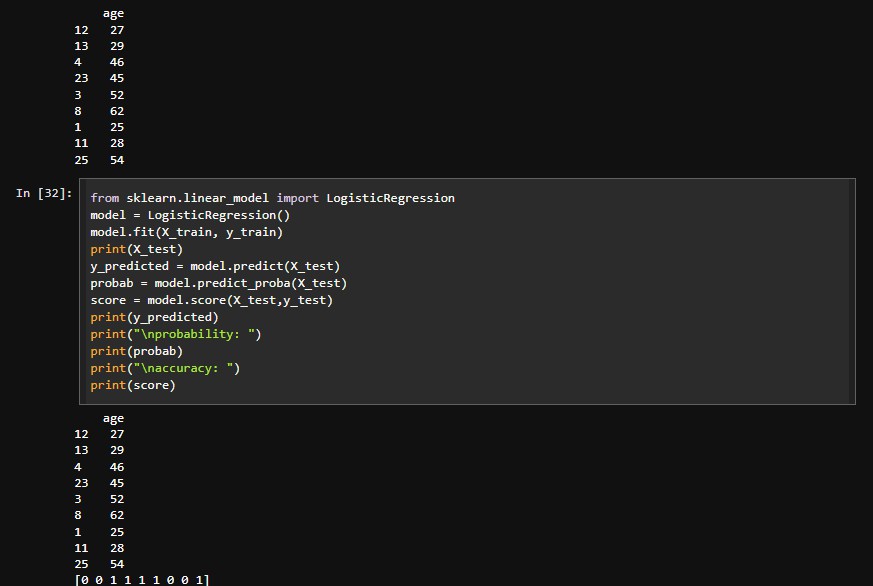
Results:

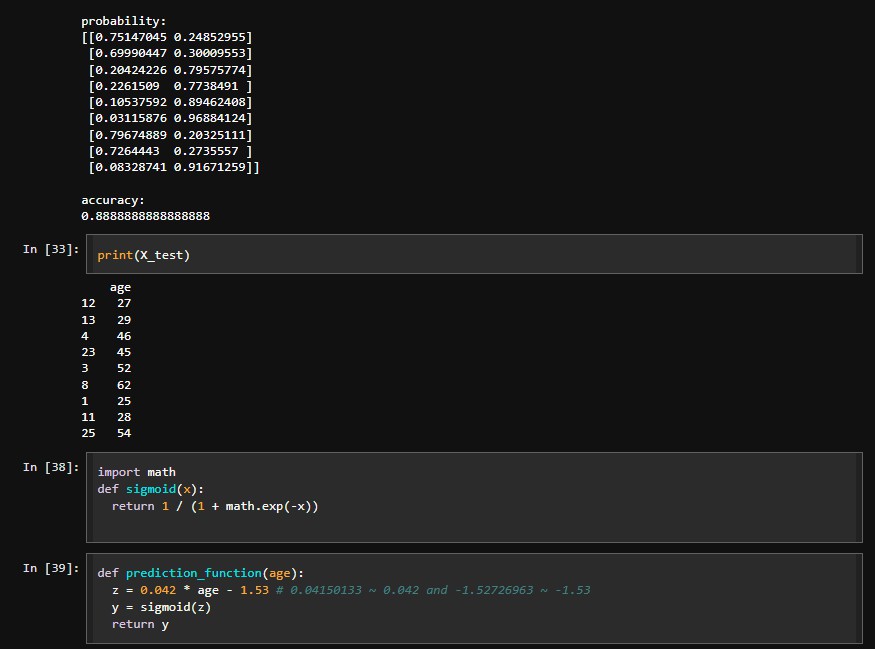


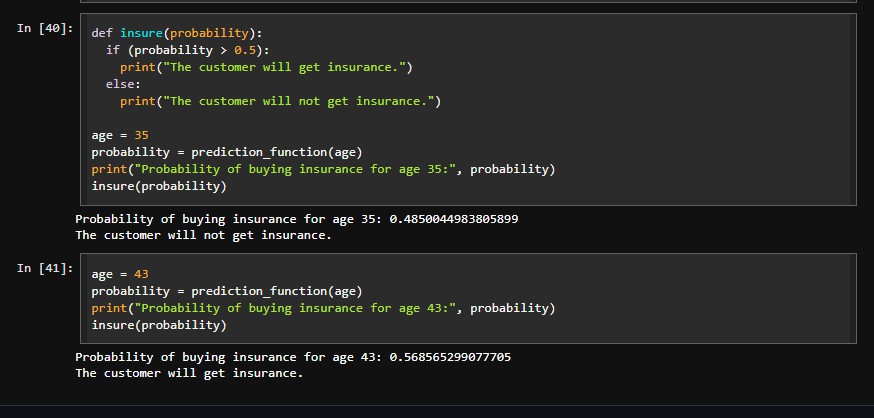
1. Build Logistic Regression Model for a given dataset

Code:

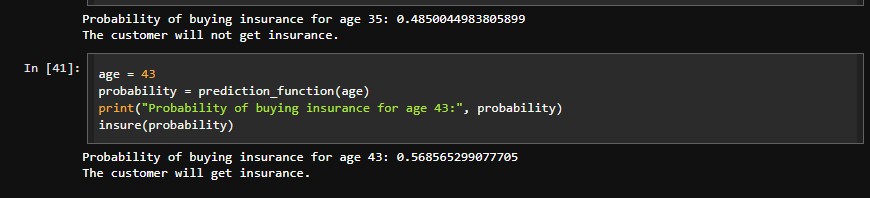




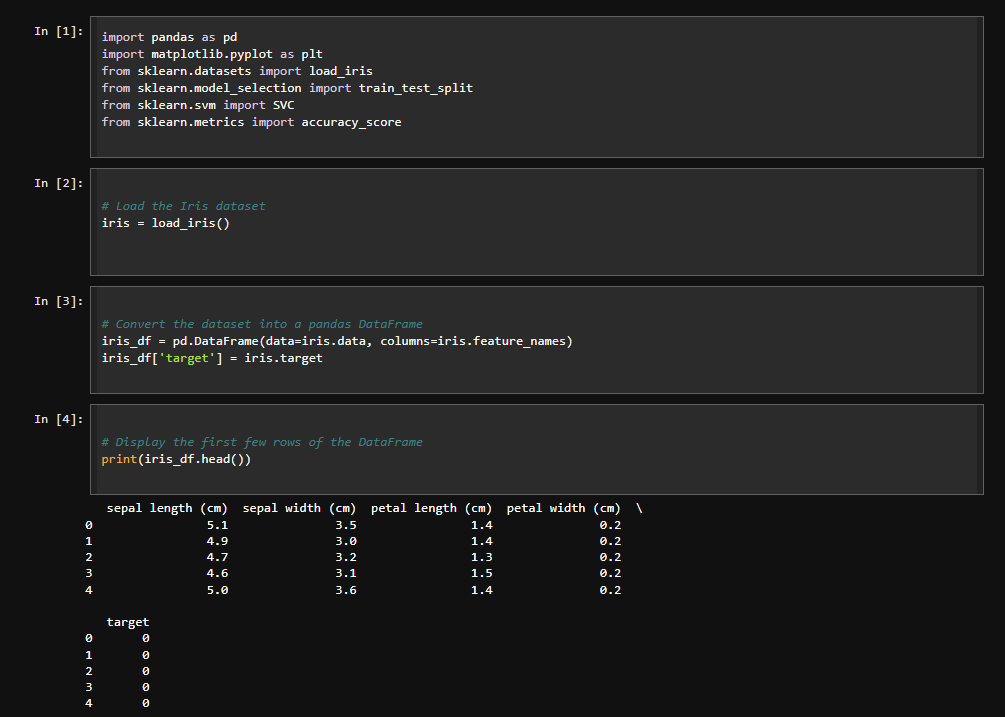




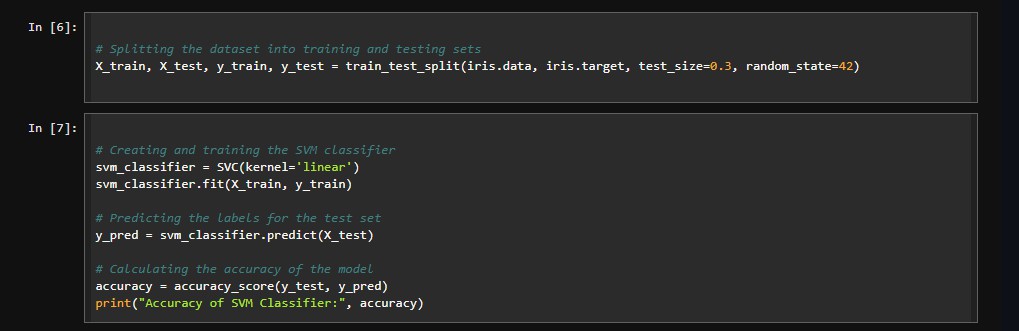
Results:



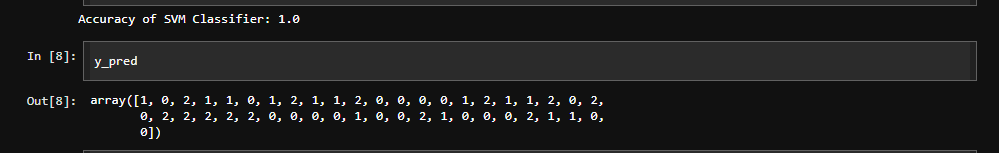
1. Build Support vector machine model for a given dataset

Code:



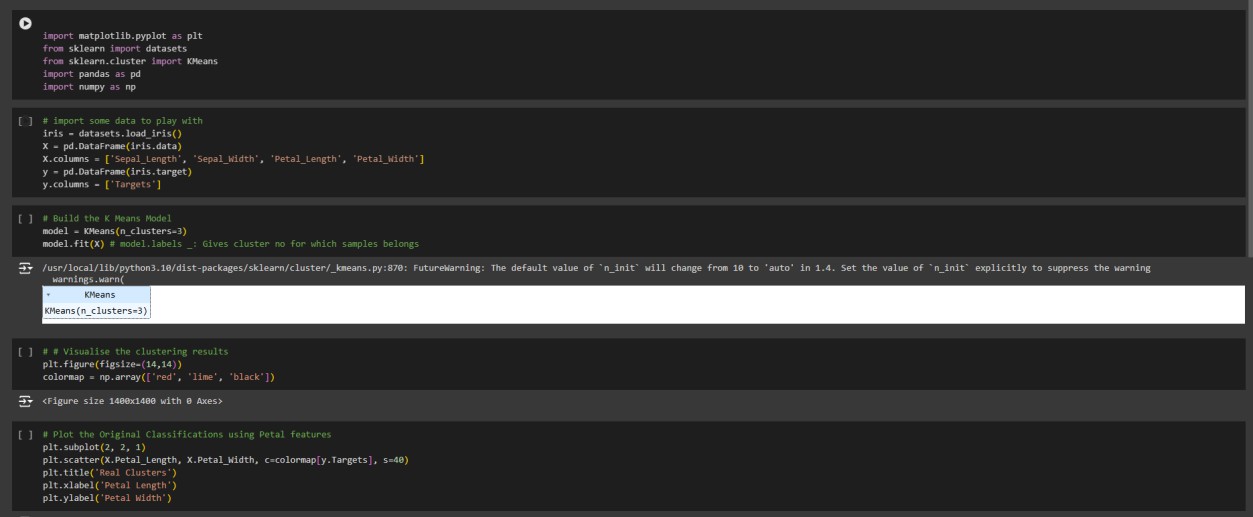


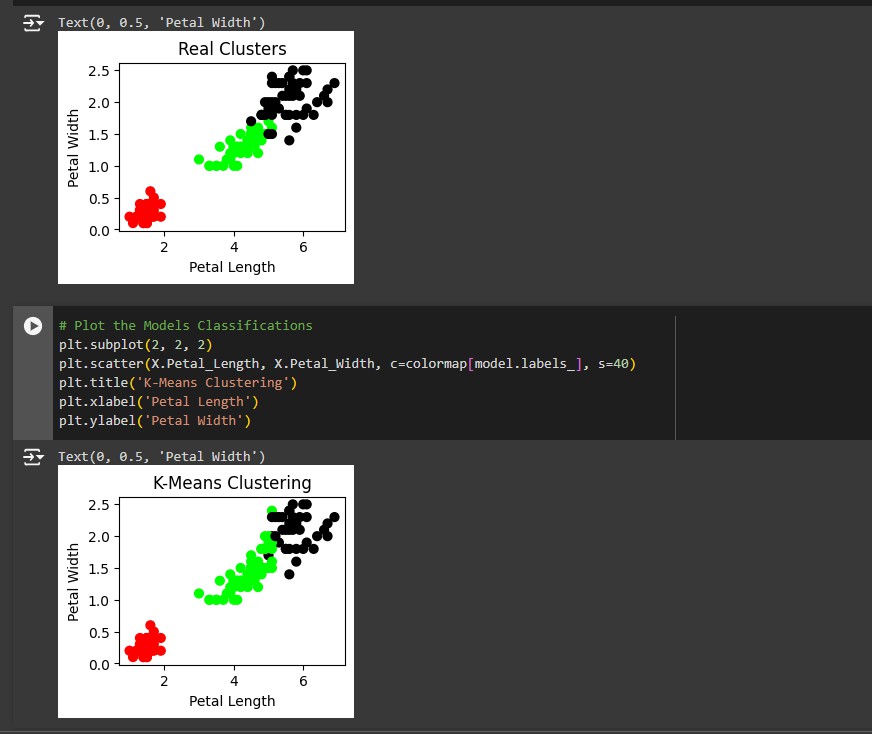
Results:



1. Build k-Means algorithm to cluster a set of data stored in a .CSV file.

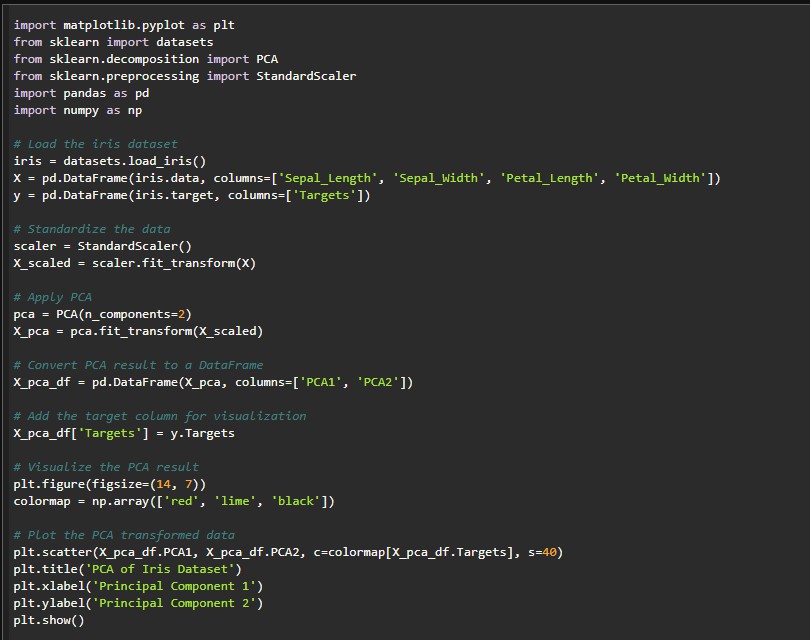
Code:

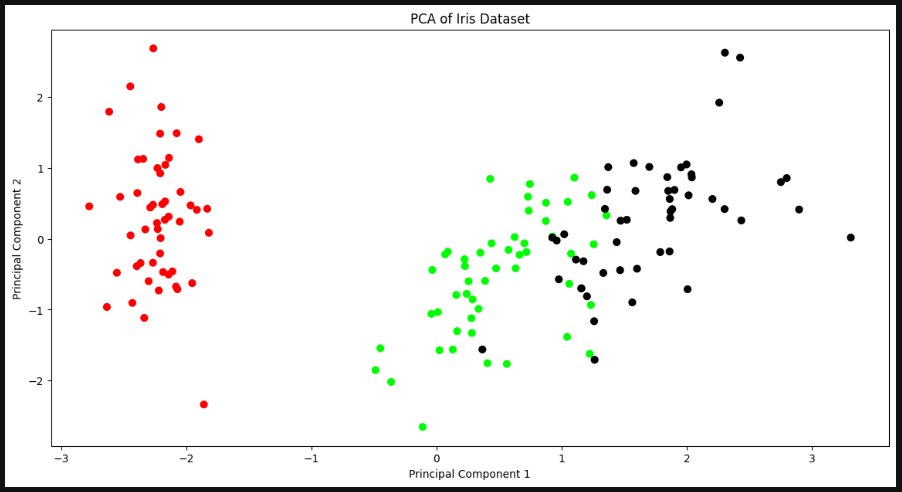




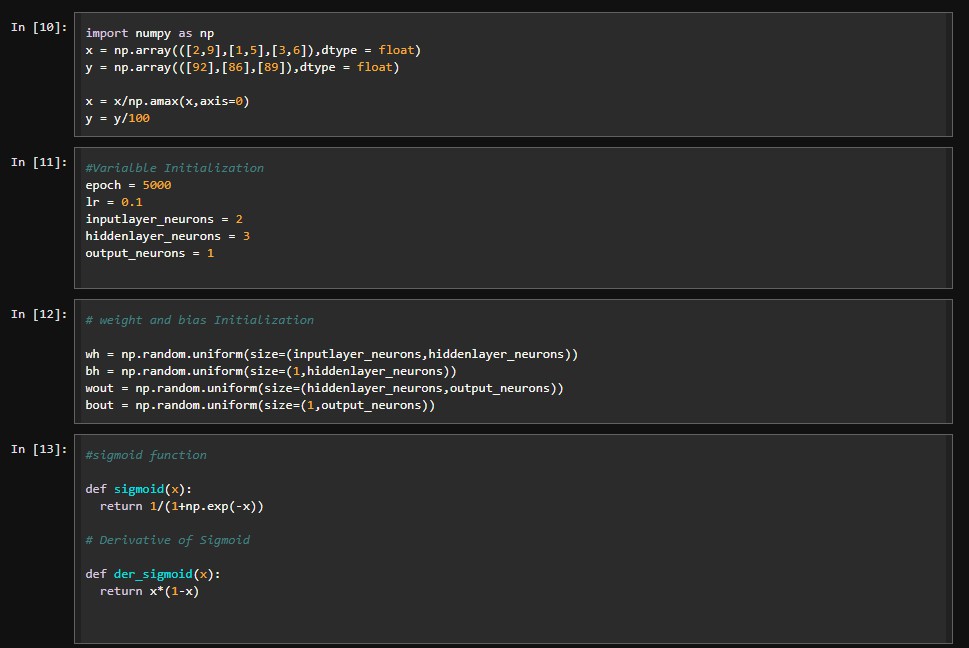
1. Implement Dimensionality reduction using Principle Component Analysis (PCA)

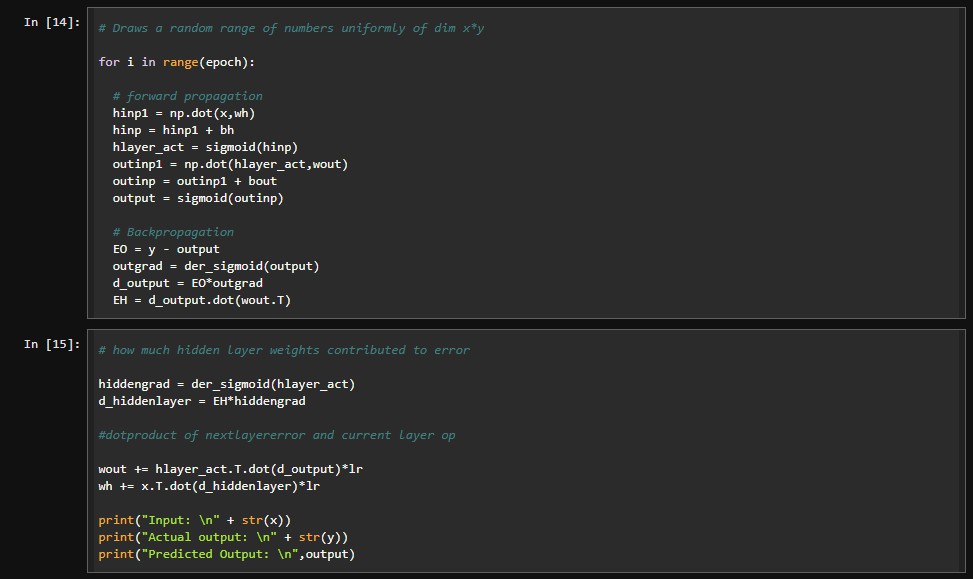
method.

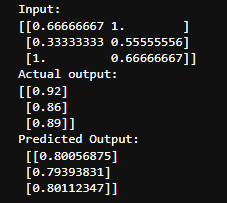
Code:

Results:

1. Build Artificial Neural Network model with back propagation on a given dataset

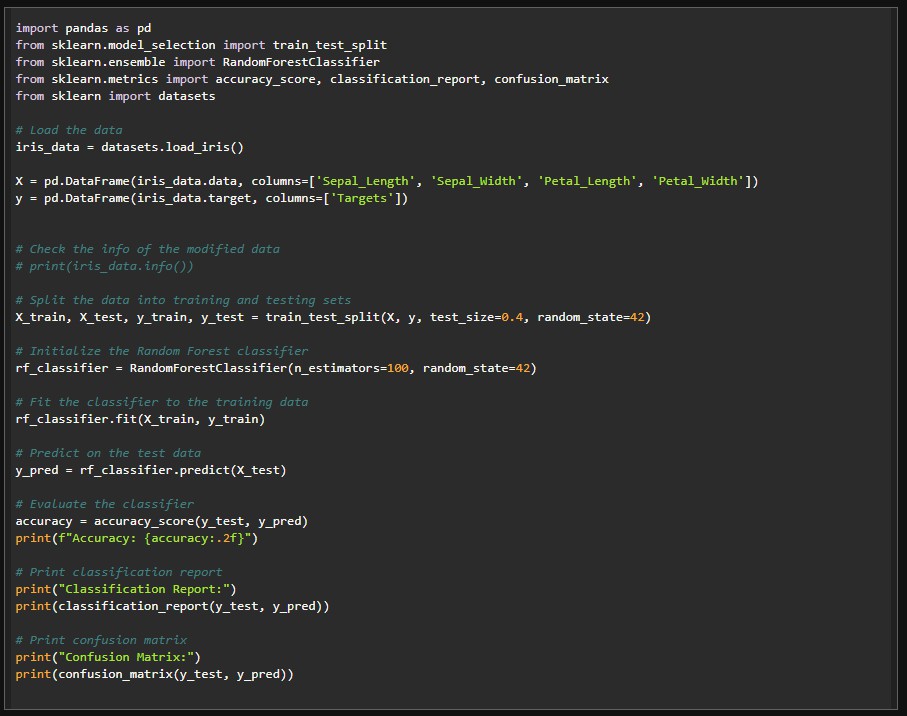
Code:



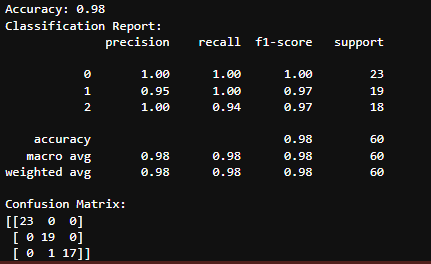
Results

1. Implement Random forest ensemble method on a given dataset.

Code:

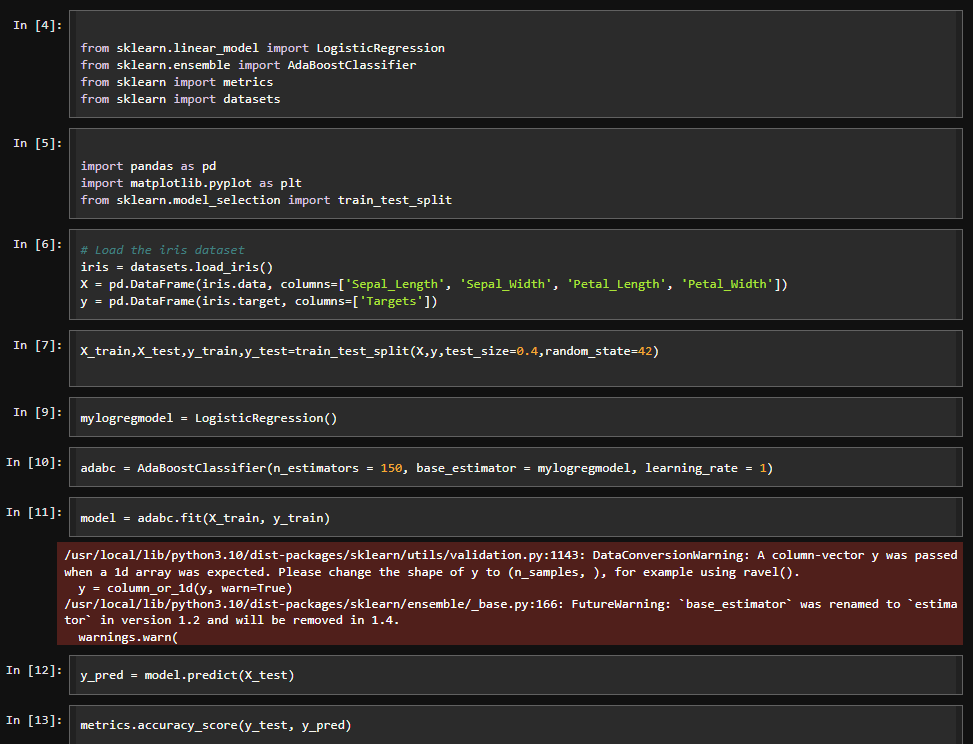


Results:



1. Implement Boosting ensemble method on a given dataset.

Code:



Results:

