**CSE 3045: Mathematical Modelling for Data Science**

**Digital Assignment 3**

**Name: Tejas Rahul Rokade**

**Reg. No.: 20BDS0033**

**Faculty: Dr. Ilanthenral K P S K**

**Lab Slot: L49 & L50**

**Linear Classification**

1. Implement least square method of linear classification in python for any data of your choice.
2. Using R, perform least square method of linear classification.

**Least Square Method of linear classification in Python:**

**Code:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

#Reading the data

data = pd.read\_csv("C:/Users/bente/OneDrive/Desktop/icecreamcone.csv")

print(data.shape)

print(data.head())

X = data['viscosity'].values

Y = data['protein'].values

# Mean X and Y

mean\_x = np.mean(X)

mean\_y = np.mean(Y)

# Total number of values

n = len(X)

# Using the formula to calculate 'm' and 'c'

numer = 0

denom = 0

for i in range(n):

numer += (X[i] - mean\_x) \* (Y[i] - mean\_y)

denom += (X[i] - mean\_x) \*\* 2

m = numer / denom

c = mean\_y - (m \* mean\_x)

# Printing coefficients

print("Coefficients")

print(m, c)

# Plotting Values and Regression Line

max\_x = np.max(X) + 100

min\_x = np.min(X) - 100

# Calculating line values x and y

x = np.linspace(min\_x, max\_x, 1000)

y = c + m \* x

# Ploting Line

plt.plot(x, y, color='#58b970', label='Regression Line')

# Ploting Scatter Points

plt.scatter(X, Y, c='#ef5423', label='Scatter Plot')

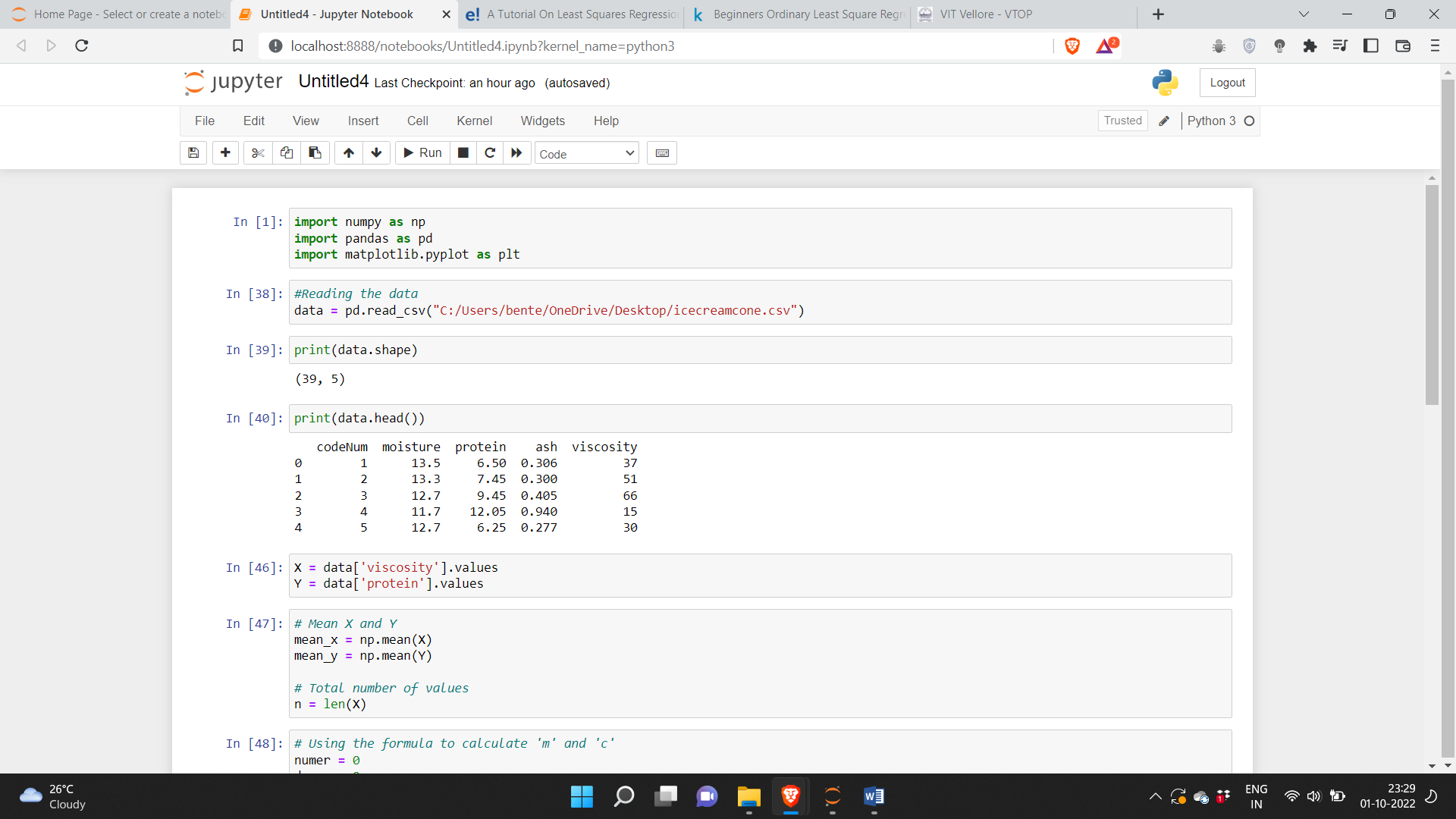
plt.xlabel('Viscosity')

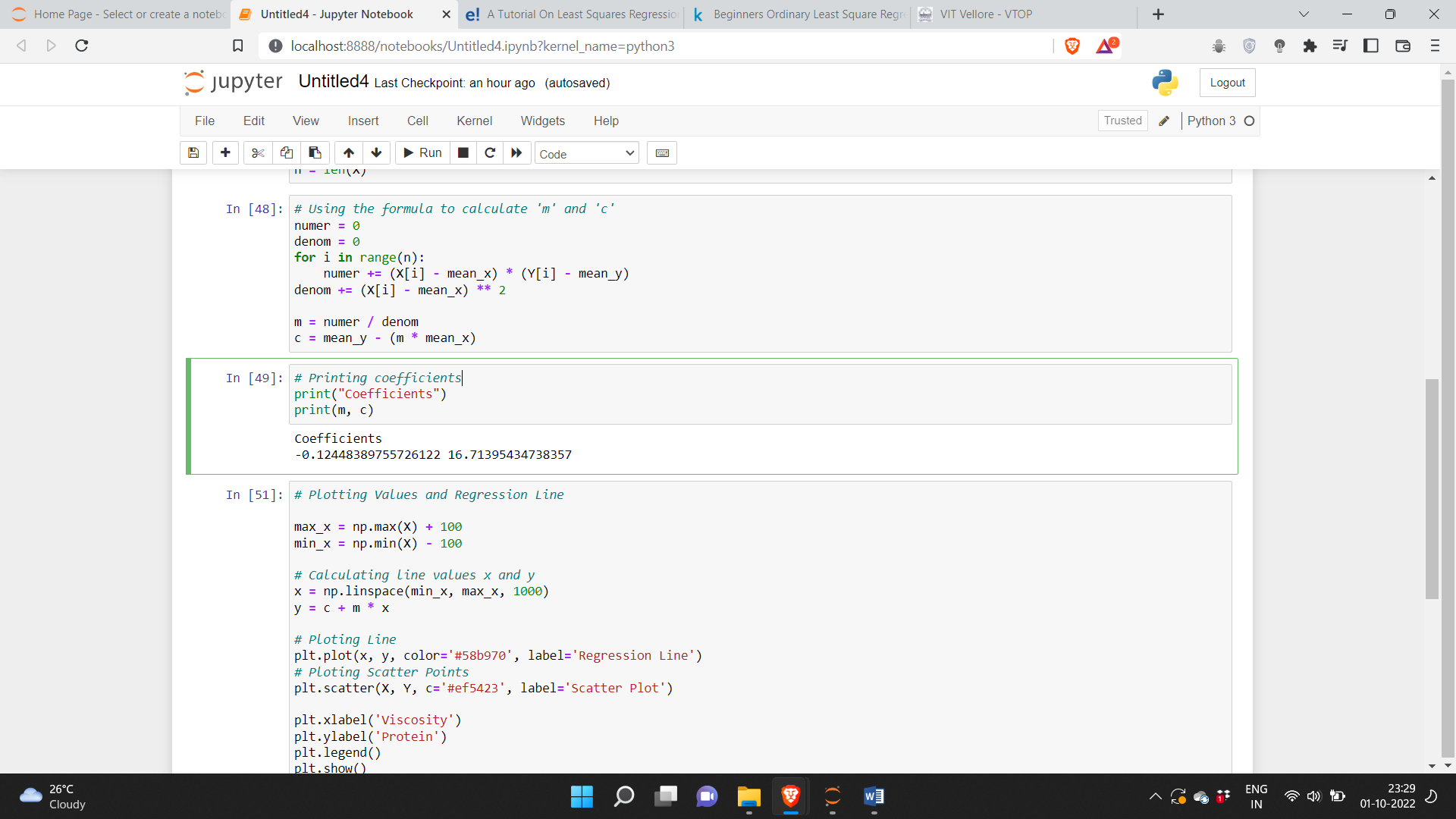
plt.ylabel('Protein')

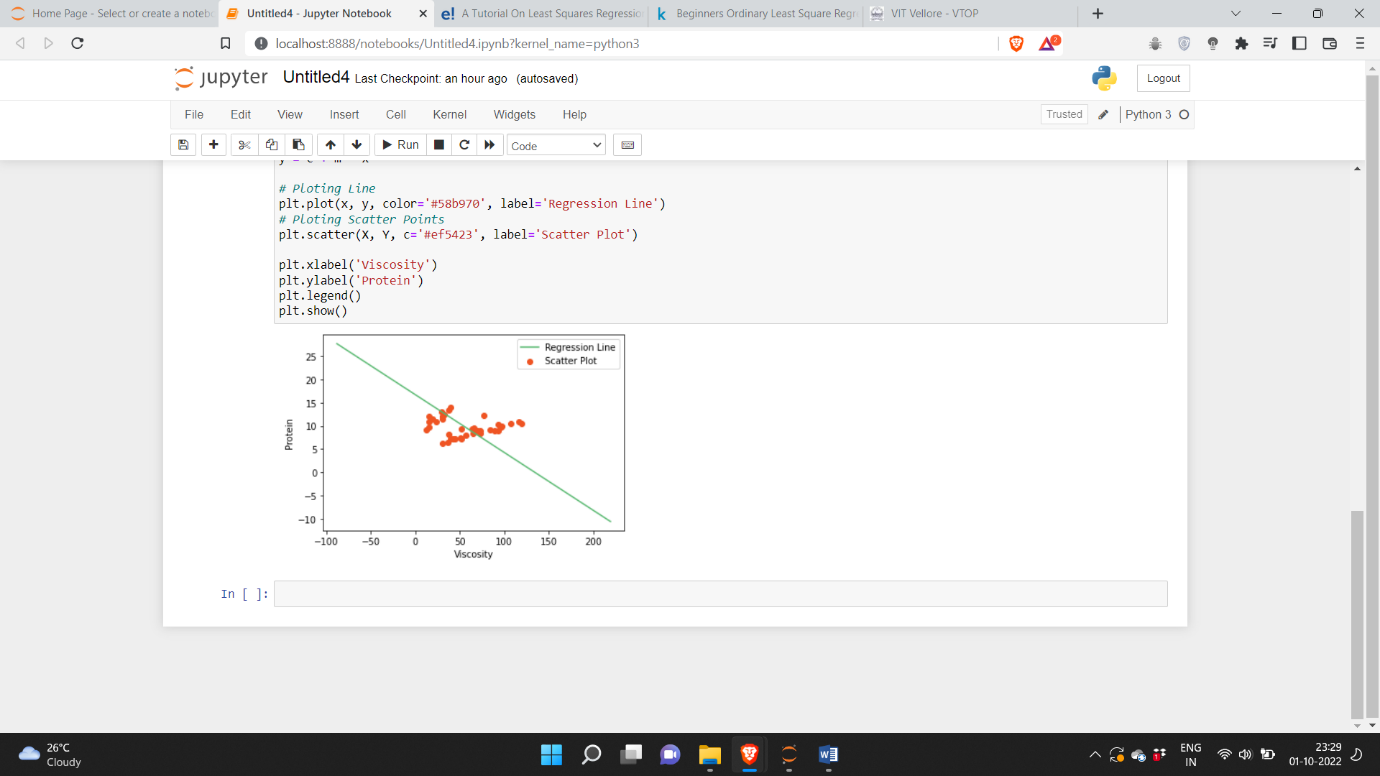
plt.legend()

plt.show()

**Output:**







**Least Square Method of linear classification in R:**

**Code:**

train <- read.csv("C:/Users/bente/OneDrive/Desktop/icecreamcone.csv")

dim(train)

train

x = train$viscosity

y = train$protein

plot(x,y, xlab = "Viscosity", ylab = "protein")

lm(x~y)

model1 = lm(x~y)

model1$fitted.values

yfit<-model1$fitted.values

plot(x,y)

lines(x,y)

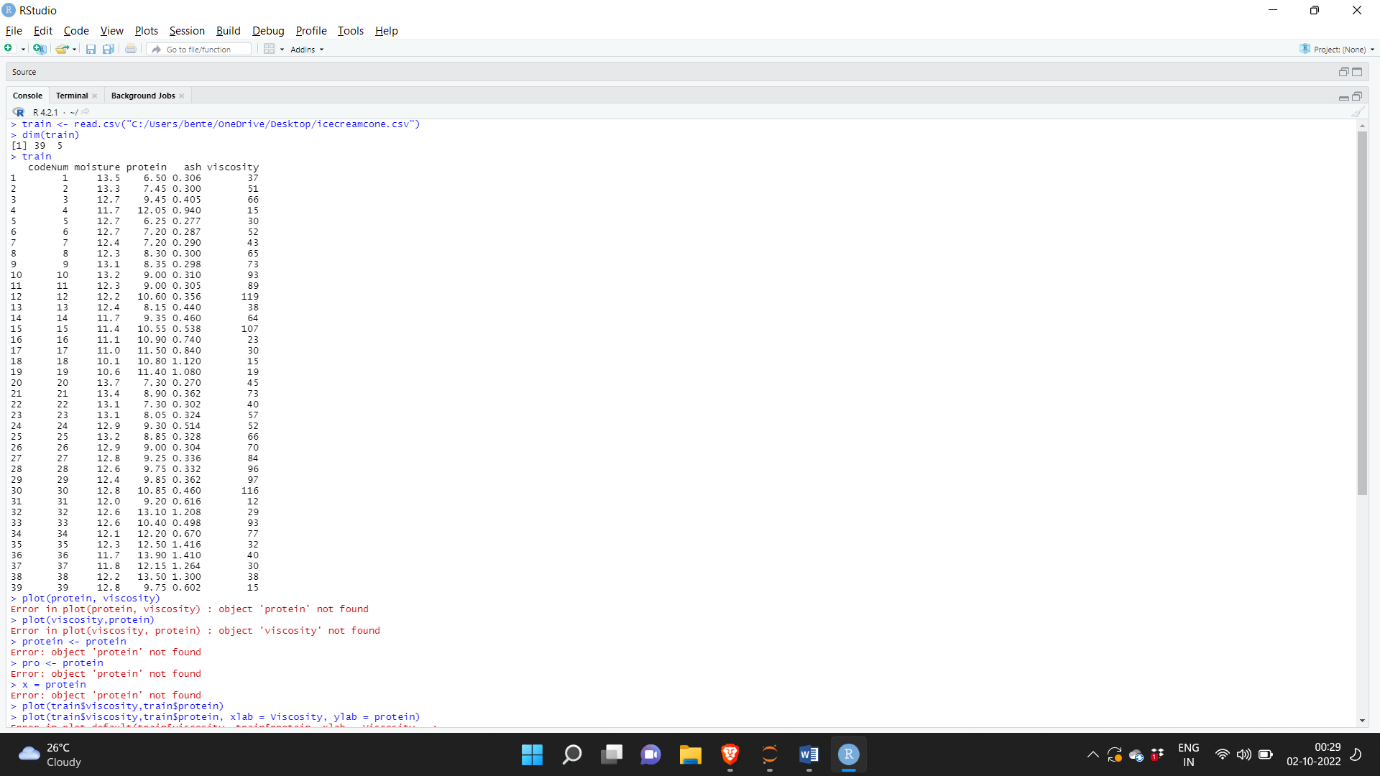
Residual = A−A.

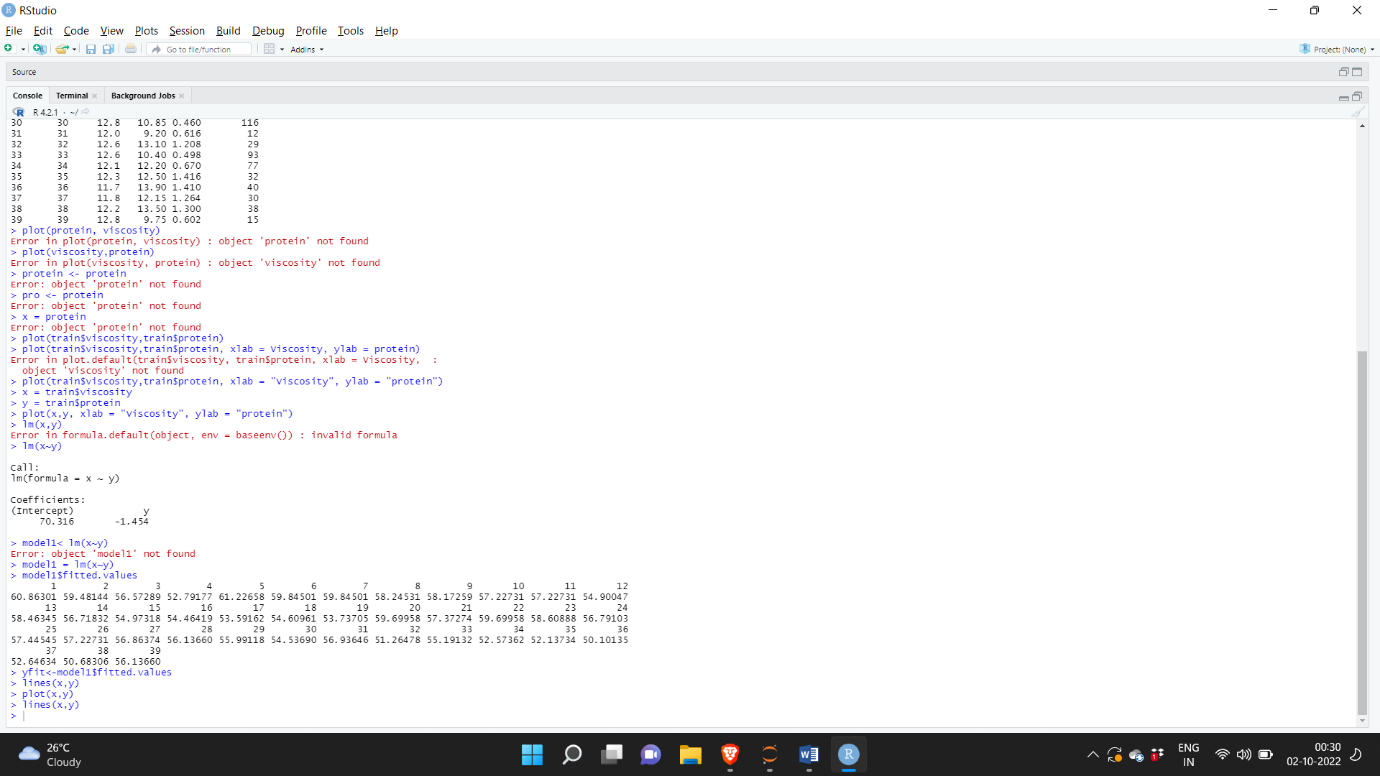
resid←risk – yfit

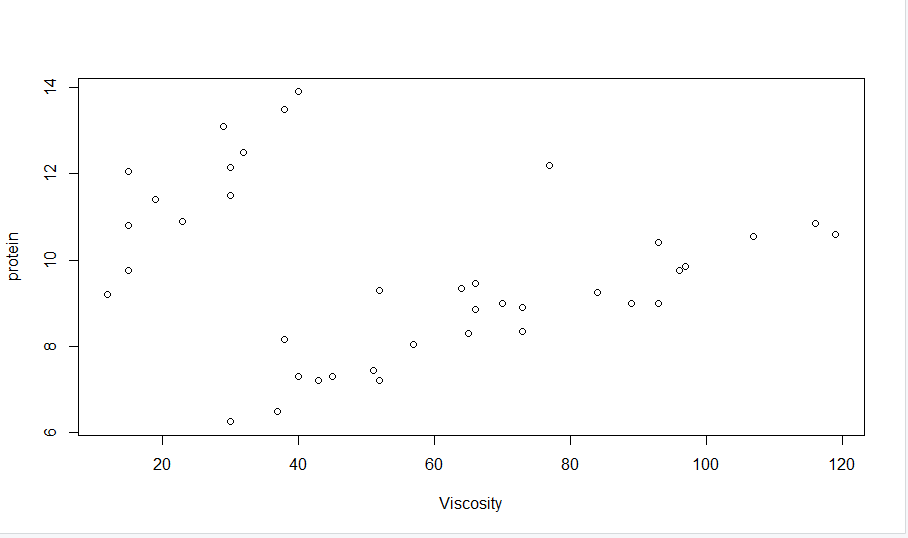
plot(x, resid)

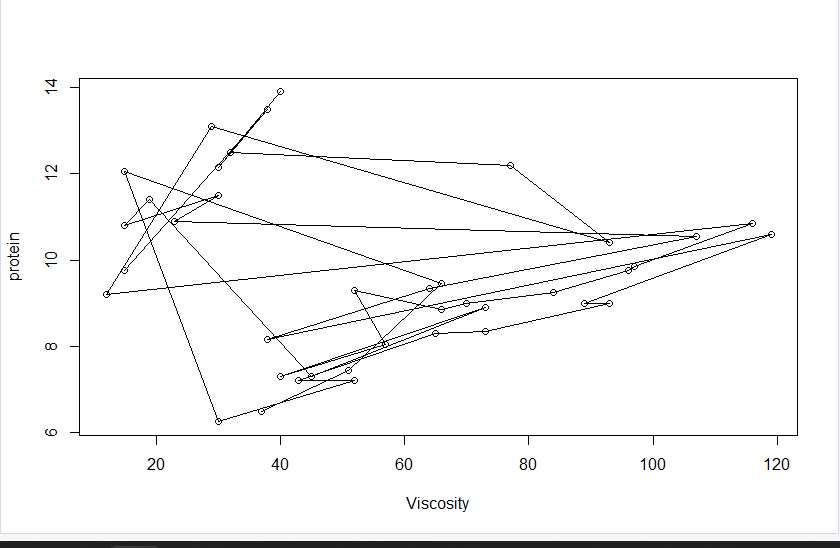
cor(x, y)

**Output:**









**Principal Component Analysis**

**PCA in python:**

**Code:**

import pandas as pd

import numpy as np

from sklearn.decomposition import PCA

from sklearn import preprocessing

import matplotlib.pyplot as plt

df = pd.read\_csv("C:/Users/bente/OneDrive/Desktop/icecreamcone.csv")

df

from sklearn.preprocessing import StandardScaler

features = ['moisture','ash','viscosity']

x = df.loc[:, features].values

y = df.loc[:, ['protein']].values

x = StandardScaler().fit\_transform(x)

pca = PCA(n\_components=2)

principalComponents = pca.fit\_transform(x)

principalDataframe = pd.DataFrame(data = principalComponents, columns = ['PC1', 'PC2'])

targetDataframe = df[['protein']]

newDataframe = pd.concat([principalDataframe, targetDataframe],axis = 1)

newDataframe

percent\_variance = np.round(pca.explained\_variance\_ratio\_\* 100, decimals =2)

columns = ['PC1', 'PC2']

plt.bar(x= range(1,5), height=percent\_variance, tick\_label=columns)

plt.ylabel('Percentate of Variance Explained')

plt.xlabel('Principal Component')

plt.title('PCA Scree Plot')

plt.show()

plt.scatter(principalDataframe.PC1, principalDataframe.PC2)

plt.title('PC1 against PC2')

plt.xlabel('PC1')

plt.ylabel('PC2')

fig = plt.figure(figsize = (8,8))

ax = fig.add\_subplot(1,1,1)

ax.set\_xlabel('PC1')

ax.set\_ylabel('PC2')

ax.set\_title('Plot of PC1 vs PC2', fontsize = 20)

targets = ['protein','moisture','ash','viscosity']

colors = ['r', 'g', 'b']

for target, color in zip(targets,colors):

indicesToKeep = newDataframe['protein'] == target

ax.scatter(newDataframe.loc[indicesToKeep, 'PC1']

, newDataframe.loc[indicesToKeep, 'PC2']

, c = color

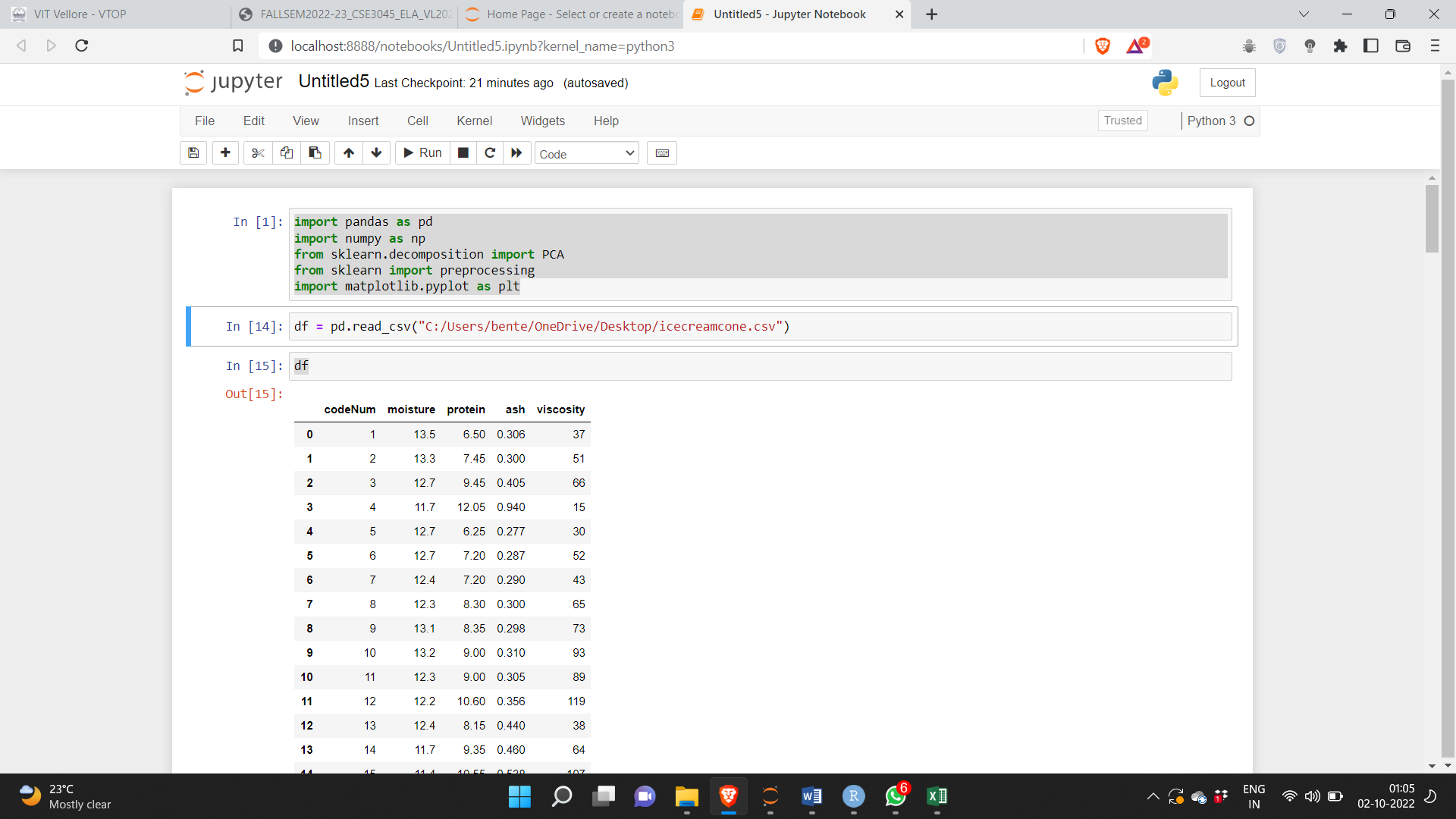
, s = 50)

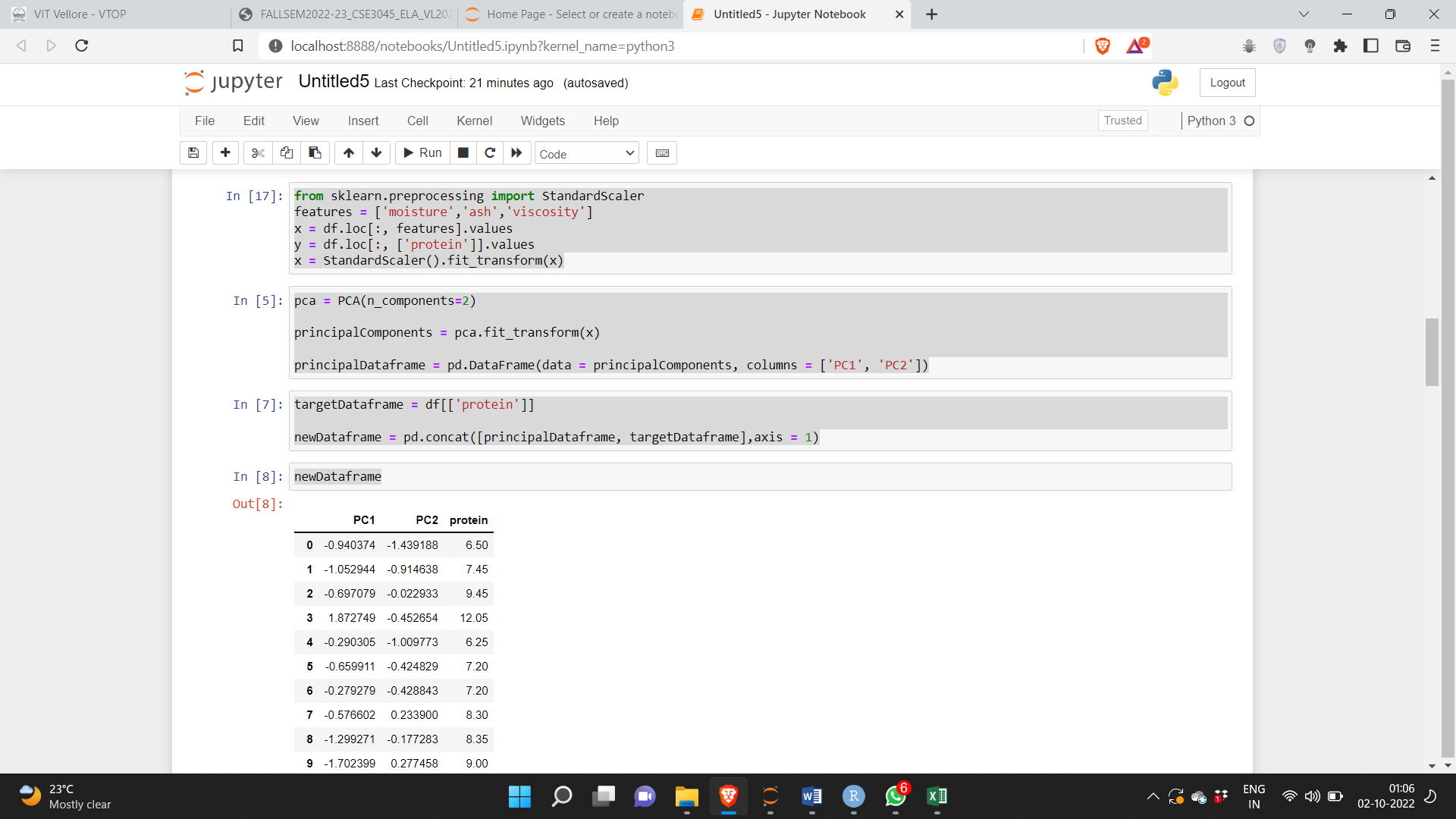
ax.legend(targets)

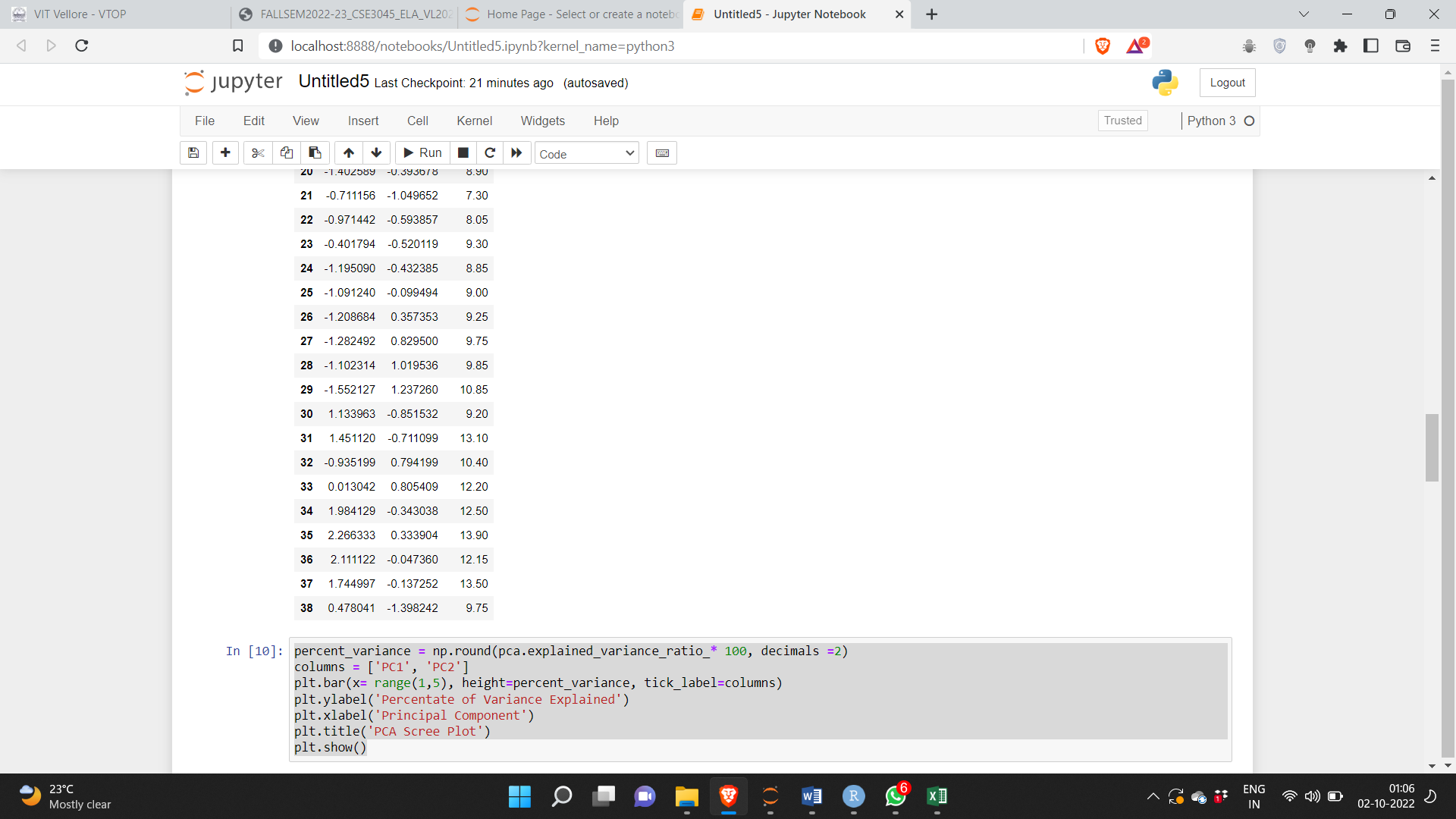
ax.grid()

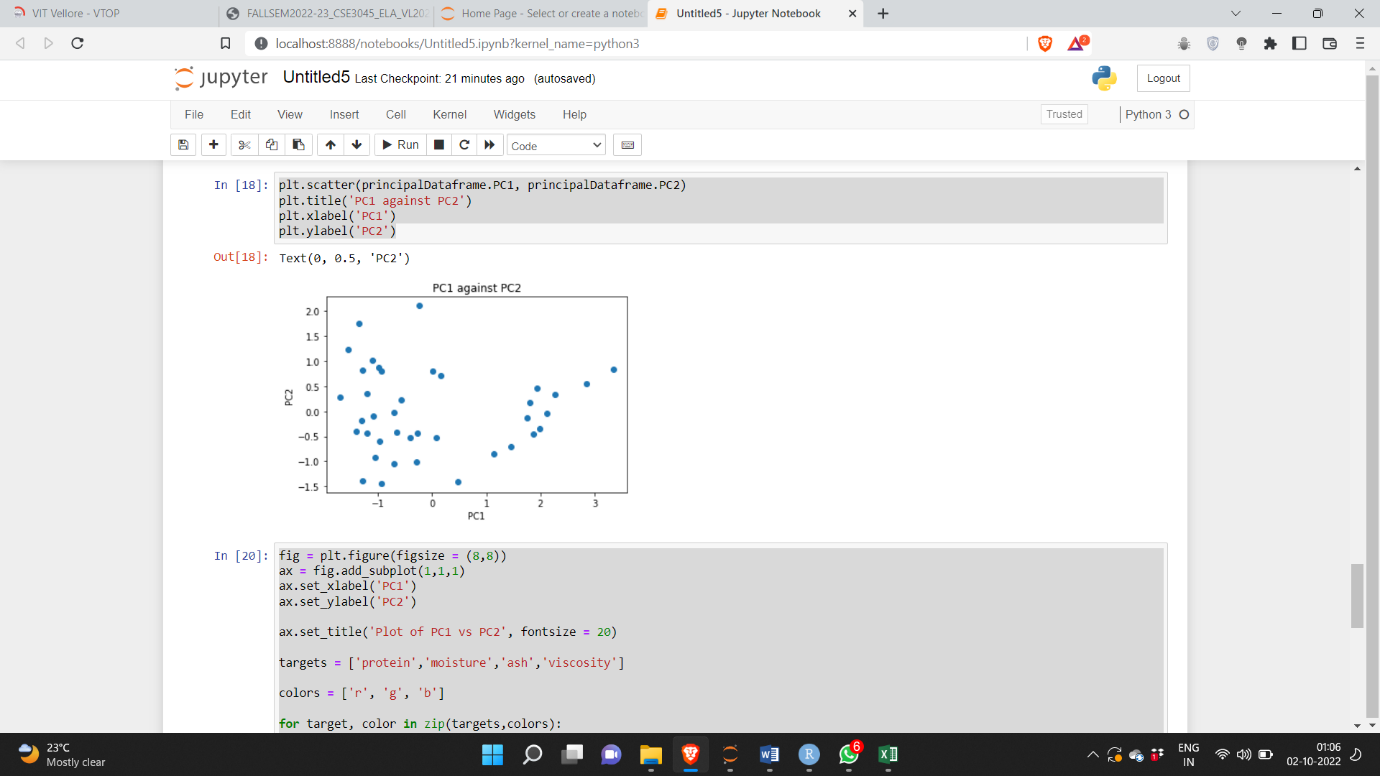
pca.explained\_variance\_ratio\_

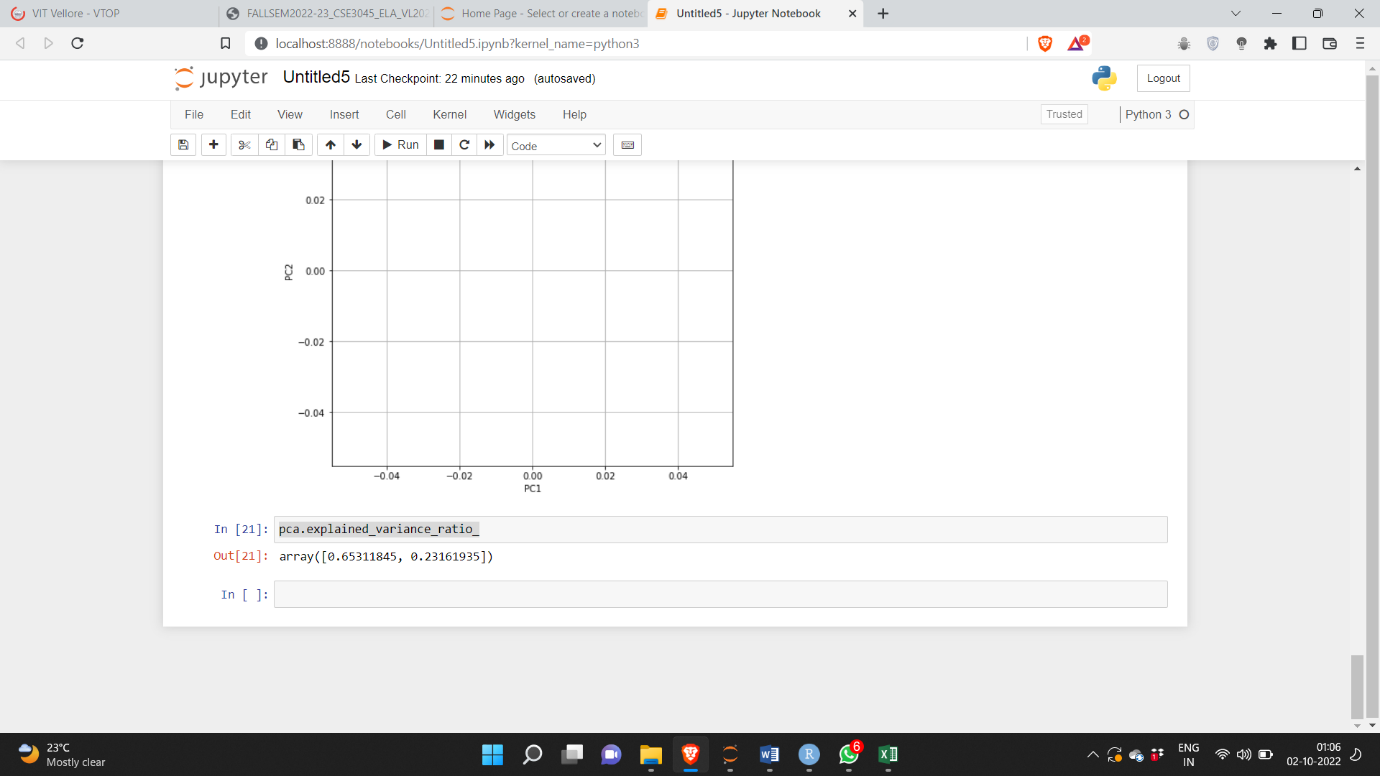
**Output:**











**PCA in R**

**Code:**

library(tidyverse)

train <- read.csv("C:/Users/bente/OneDrive/Desktop/icecreamcone.csv")

head(train)

#calculate principal components

results <- prcomp(train, scale = TRUE)

#reverse the signs

results$rotation <- -1\*results$rotation

#display principal components

results$rotation

#reverse the signs of the scores

results$x <- -1\*results$x

#display the first six scores

head(results$x)

biplot(results, scale = 0)

#display states with highest murder rates in original dataset

head(train[order(-train$protein),])

#calculate total variance explained by each principal component

results$sdev^2 / sum(results$sdev^2)

#calculate total variance explained by each principal component

var\_explained = results$sdev^2 / sum(results$sdev^2)

#create scree plot

qplot(c(1:5), var\_explained) + geom\_line() + xlab("Principal Component") + ylab("Variance Explained") + ggtitle("Scree Plot") + ylim(0, 1)

**Output:**

