# **In Lab:**

## **Lab Task 1:**

# Download the data using wget

# !wget "https://www.dropbox.com/s/veak3ugc4wj9luz/Alumni%20Giving%20Regression%20%28Edited%2 9.csv"

from keras.models import Sequential

from keras.layers import Dense, Dropout

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

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

from sklearn.metrics import mean\_squared\_error

import numpy as np

from sklearn import linear\_model

from sklearn import preprocessing

from sklearn import tree

from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor

import pandas as pd

import csv

import matplotlib.pyplot as plt

import seaborn as sns # Import seaborn

**Output:**

## 

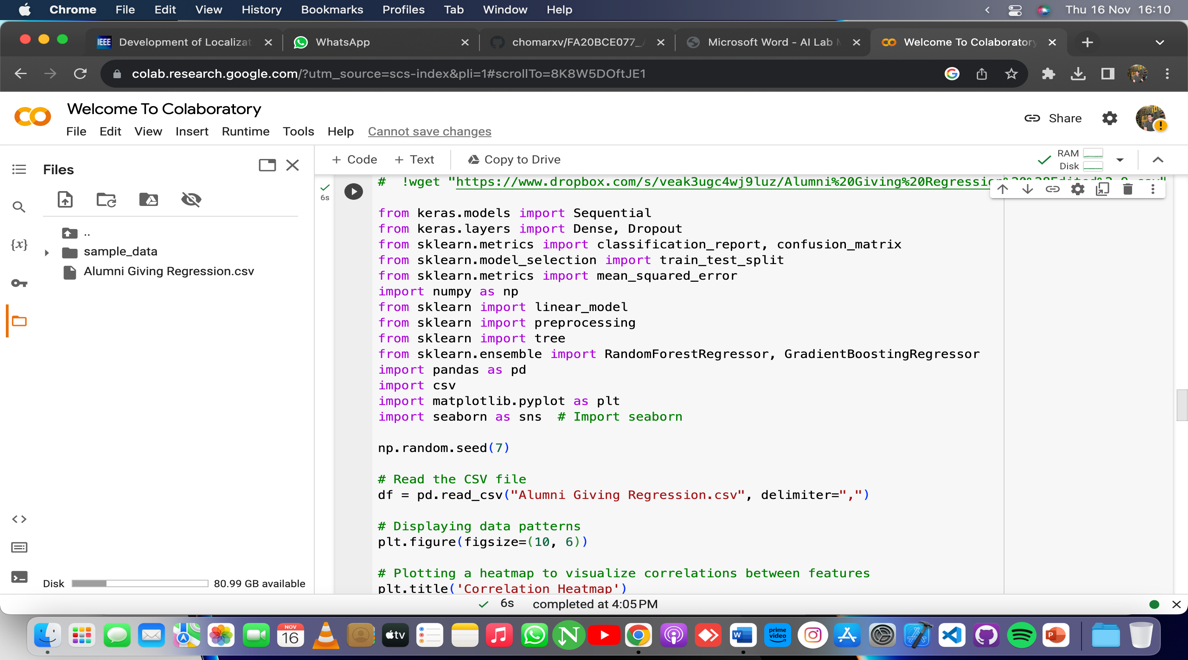
## **Lab Task 2:**

np.random.seed(7)

# Read the CSV file

df = pd.read\_csv("Alumni Giving Regression.csv", delimiter=",")

## **Output:**

****

## **Lab Task 3:**

# Displaying data patterns

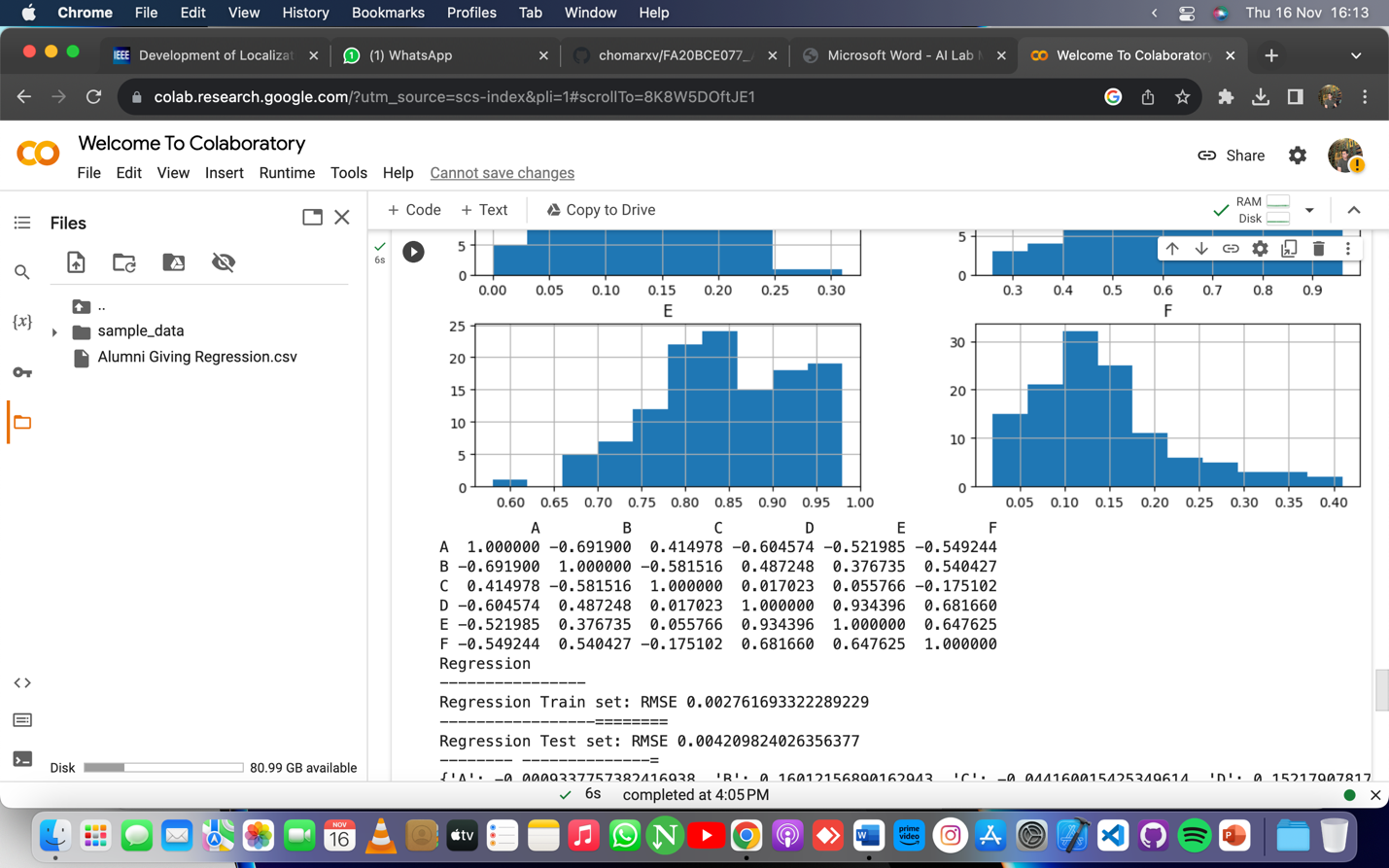
plt.figure(figsize=(10, 6))

# Describing the data

data\_description = df.describe()

print("Data Description:\n", data\_description)

## **Output:**

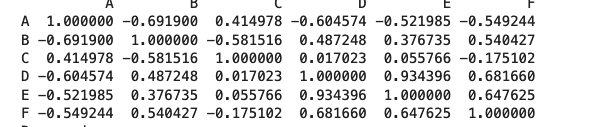


## **Lab Task 4:**

corr = df.corr(method="pearson")

print(corr)

## **Output:**



## **Lab Task 5:**

Y\_POSITION = 5

model\_1\_features = [i for i in range(0, Y\_POSITION)]

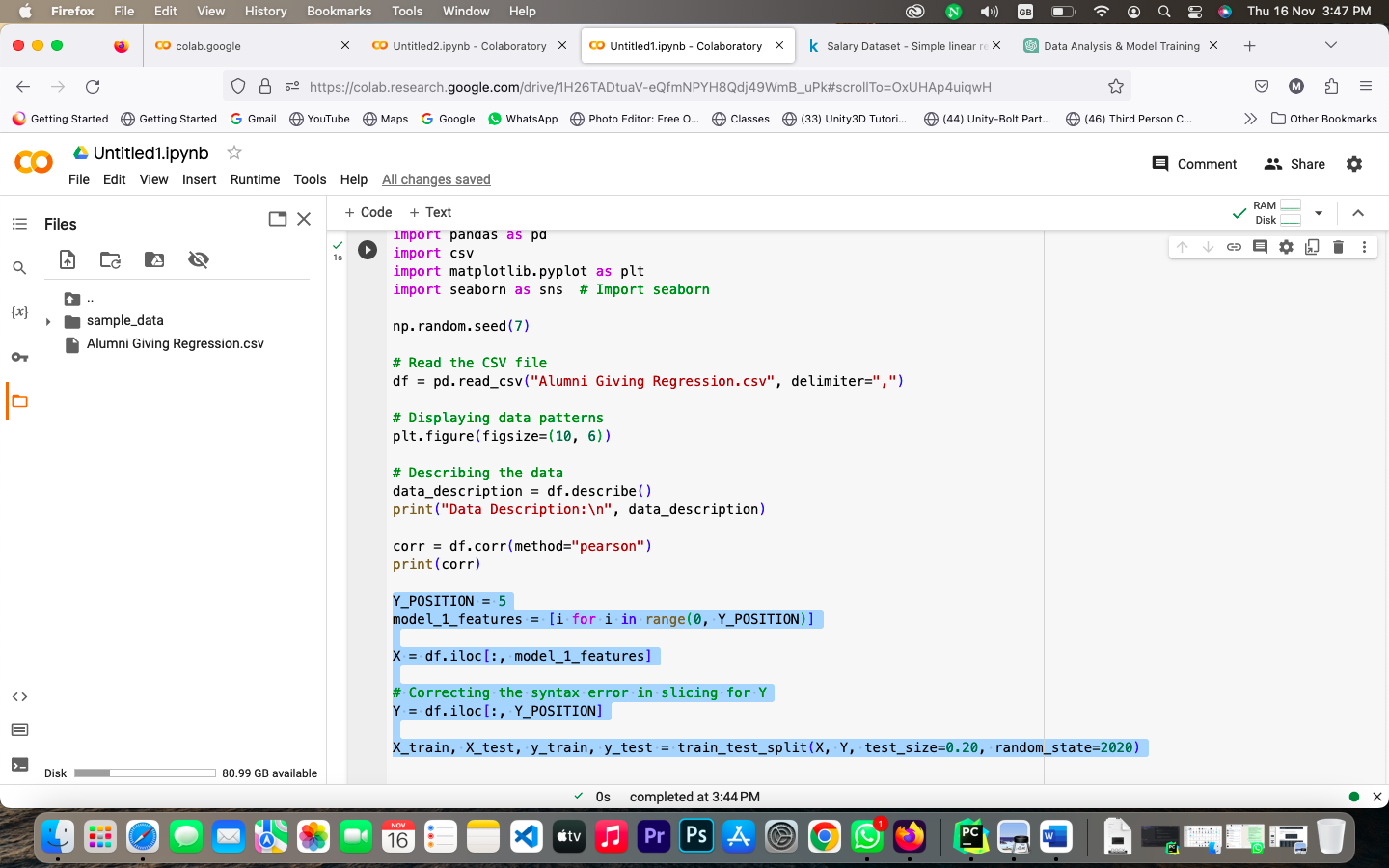
X = df.iloc[:, model\_1\_features]

# Correcting the syntax error in slicing for Y

Y = df.iloc[:, Y\_POSITION]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.20, random\_state=2020)

## **Output:**



## **Lab Task 6:**

# Assuming the data has been downloaded and loaded correctly as mentioned before

model1 = linear\_model.LinearRegression()

model1.fit(X\_train, y\_train)

y\_pred\_train1 = model1.predict(X\_train)

print("Regression")

print("----------------")

RMSE\_train1 = mean\_squared\_error(y\_train, y\_pred\_train1)

print("Regression Train set: RMSE {}".format(RMSE\_train1))

print("-----------------========")

y\_pred1 = model1.predict(X\_test) # Added missing equal sign here

RMSE\_test1 = mean\_squared\_error(y\_test, y\_pred1)

print("Regression Test set: RMSE {}".format(RMSE\_test1))

print("-------- --------------=")

coef\_dict = {} # Corrected dictionary initialization

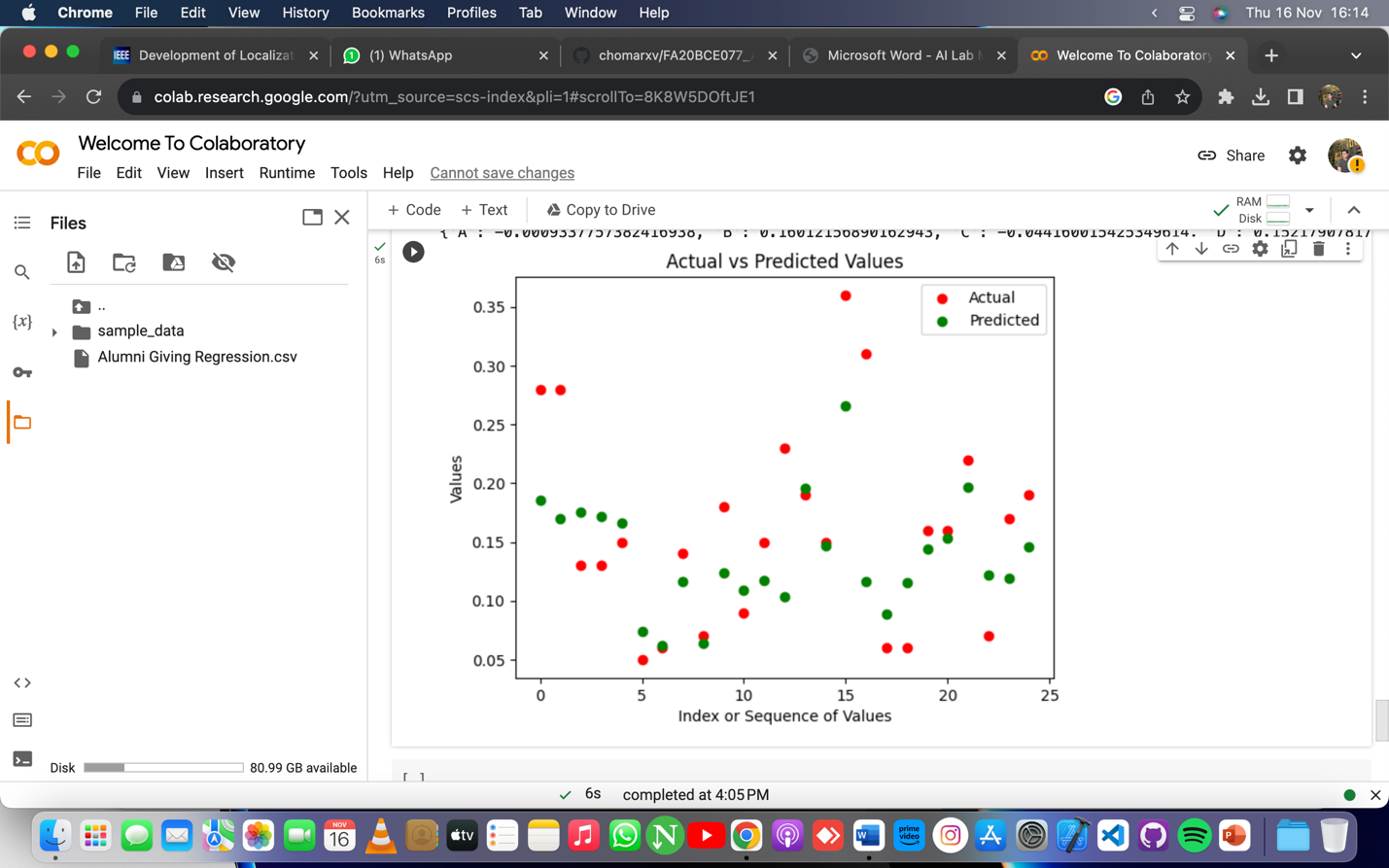
# Updated loop to access correct attributes of the model

for coef, feat in zip(model1.coef\_, model\_1\_features):

coef\_dict[df.columns[feat]] = coef

print(coef\_dict)

## **Output:**



# **Post Lab:**

import pandas as pd

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

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

from sklearn.preprocessing import PolynomialFeatures

from sklearn.model\_selection import cross\_val\_score

# Load the dataset

data = pd.read\_csv("Salary\_Data.csv")

# Extracting features and target variable

X = data.iloc[:, :-1].values # Assuming the independent variable is in the first column

y = data.iloc[:, -1].values # Assuming the dependent variable (salary) is in the last column

# Splitting the data into training and test sets

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

# Linear Regression

model = LinearRegression()

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

# Model Evaluation

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

mse = mean\_squared\_error(y\_test, y\_pred)

print(f"Linear Regression Test MSE: {mse}")

# Polynomial Regression (Degree 2)

poly = PolynomialFeatures(degree=2)

X\_poly = poly.fit\_transform(X\_train)

model\_poly = LinearRegression()

model\_poly.fit(X\_poly, y\_train)

# Model Evaluation - Polynomial Regression

X\_test\_poly = poly.transform(X\_test)

y\_pred\_poly = model\_poly.predict(X\_test\_poly)

mse\_poly = mean\_squared\_error(y\_test, y\_pred\_poly)

print(f"Polynomial Regression (Degree 2) Test MSE: {mse\_poly}")

# Cross-validation

cross\_val\_scores = cross\_val\_score(model, X, y, cv=5, scoring='neg\_mean\_squared\_error')

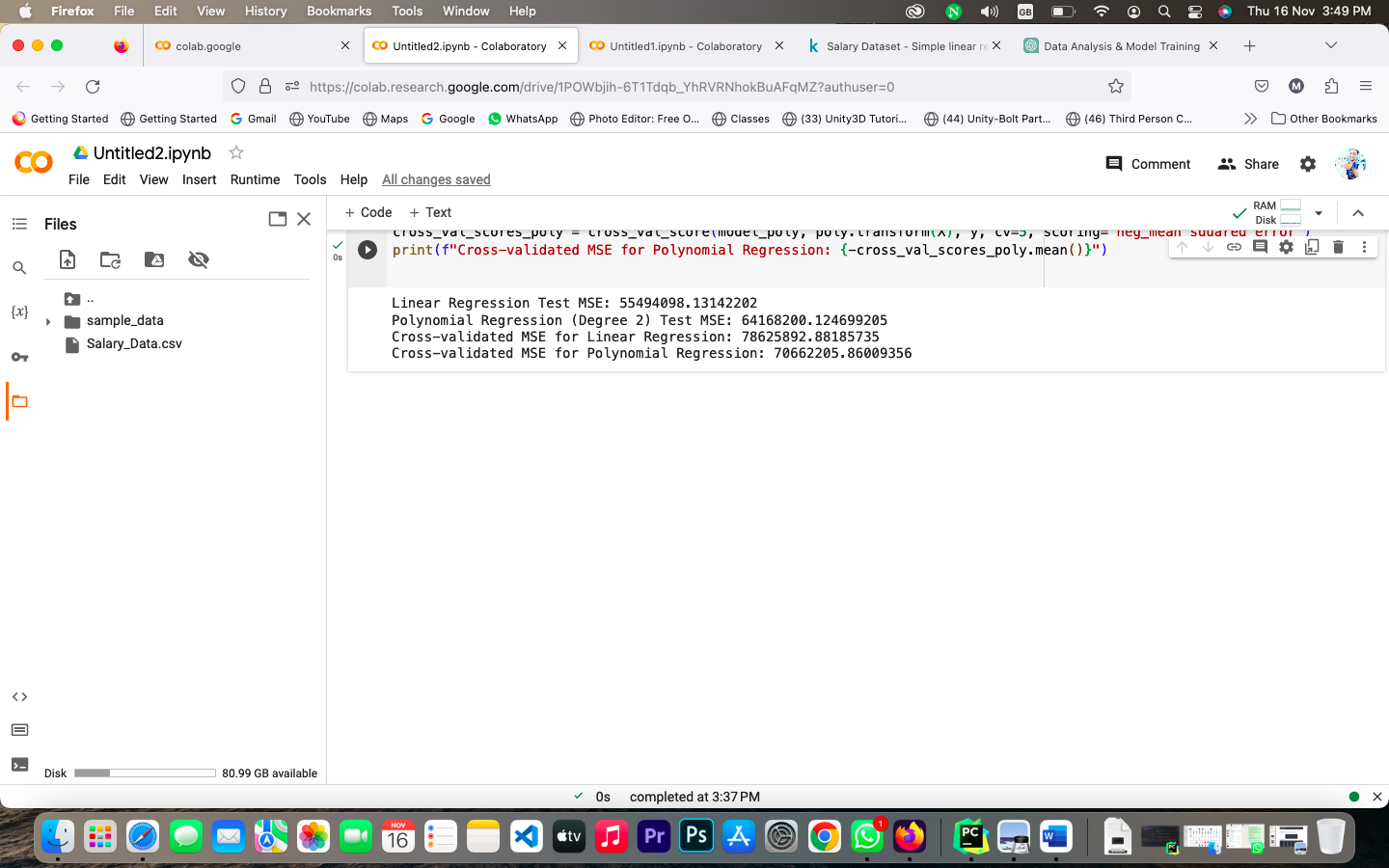
print(f"Cross-validated MSE for Linear Regression: {-cross\_val\_scores.mean()}")

# Cross-validation for Polynomial Regression

cross\_val\_scores\_poly = cross\_val\_score(model\_poly, poly.transform(X), y, cv=5, scoring='neg\_mean\_squared\_error')

print(f"Cross-validated MSE for Polynomial Regression: {-cross\_val\_scores\_poly.mean()}")

## **Output:**



**Dataset Link:**

https://www.kaggle.com/datasets/abhishek14398/salary-dataset-simple-linear-regression/