# **In Lab:**

## **Lab Task 1:**

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

model1 = tree. DecisionTreeRegressor()

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

print("Decision Tree")

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

y\_pred\_train2 = model1.predict(X\_train)

RMSE\_train2 = mean\_squared\_error (y\_train, y\_pred\_train2)

print("Decision Tree Train set: RMSE {}".format(RMSE\_train2))

y\_pred\_test2 = model1.predict(X\_test)

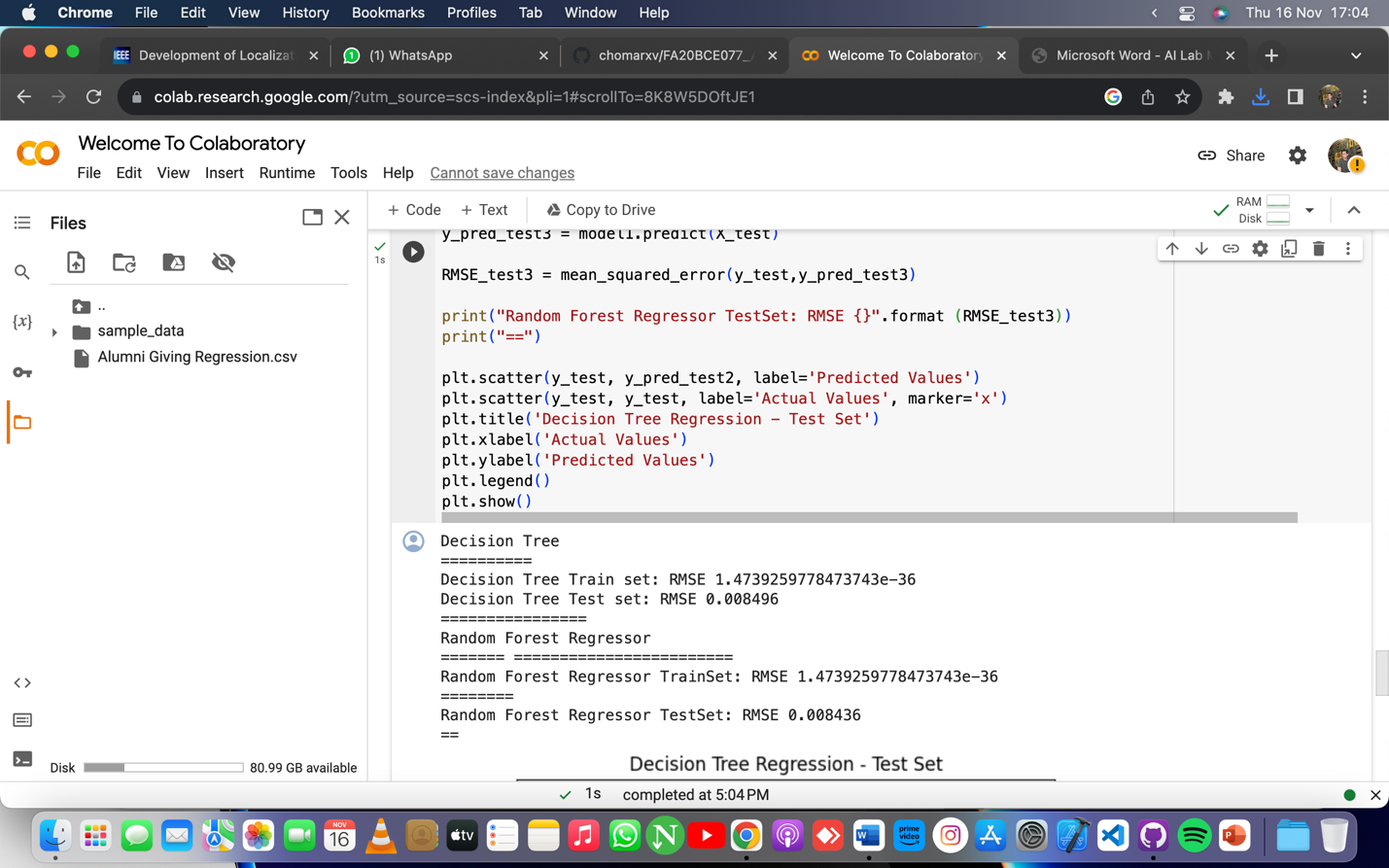
RMSE\_test2 = mean\_squared\_error(y\_test,y\_pred\_test2)

print("Decision Tree Test set: RMSE {}".format(RMSE\_test2))

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

## **Output:**

## 



## **Lab Task 2:**

# Your code for training the Decision Tree model goes here

model2 = tree.DecisionTreeRegressor()

model2.fit(X\_train, y\_train)

# Predictions on the training set

y\_pred\_train2 = model2.predict(X\_train)

# Scatter plot for actual vs. predicted values in the training set

plt.scatter(y\_train, y\_pred\_train2)

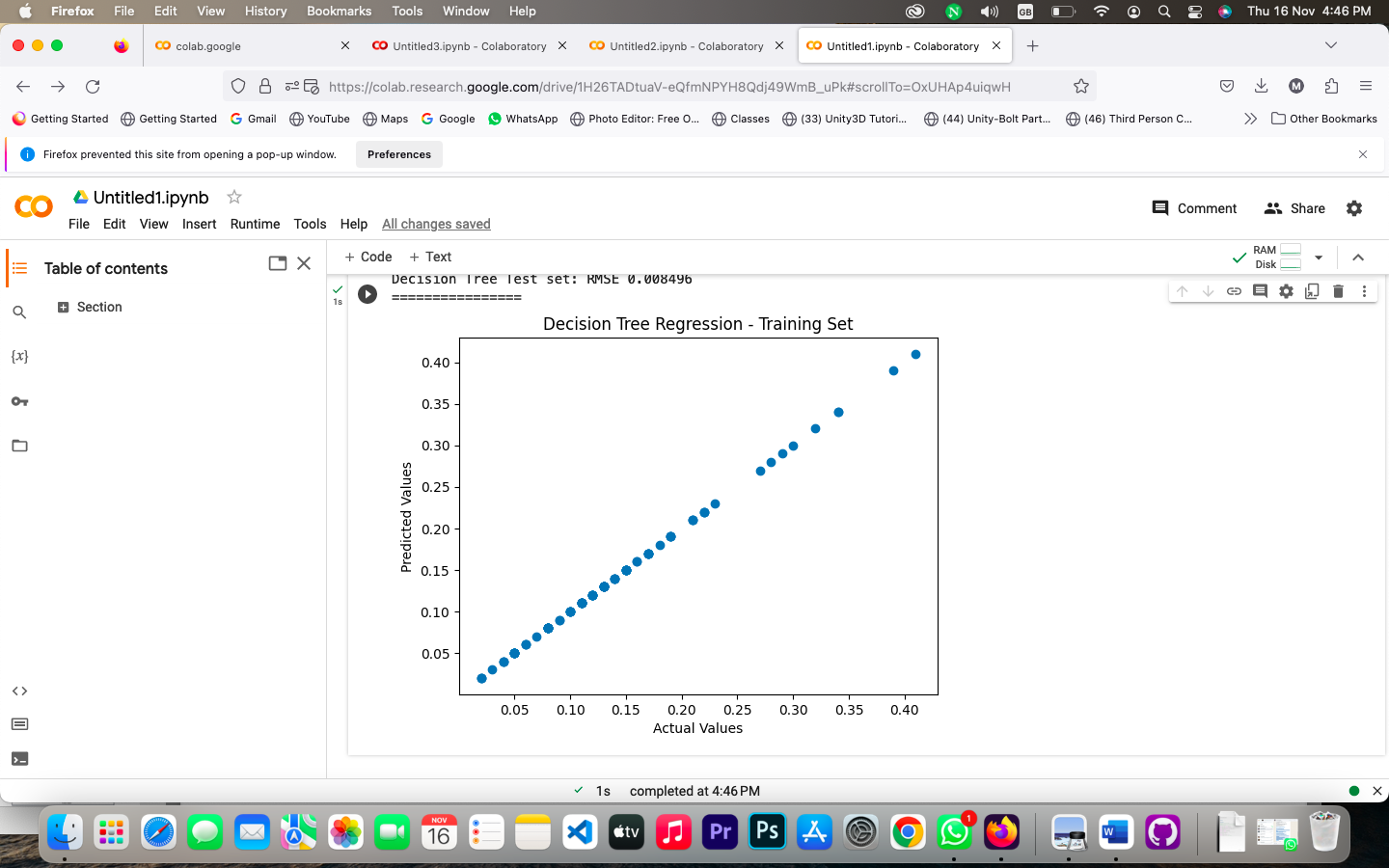
plt.title('Decision Tree Regression - Training Set')

plt.xlabel('Actual Values')

plt.ylabel('Predicted Values')

plt.show()

## **Output:**



## **Lab Task 3:**

# Your code for training the Decision Tree model goes here

model1 = tree.DecisionTreeRegressor()

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

# Predictions on the test set

y\_pred\_test2 = model1.predict(X\_test)

# Print results for the test set

print("Decision Tree")

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

RMSE\_test2 = mean\_squared\_error(y\_test, y\_pred\_test2)

print("Decision Tree Test set: RMSE {}".format(RMSE\_test2))

# Scatter plot for actual vs. predicted values in the test set

plt.scatter(y\_test, y\_pred\_test2)

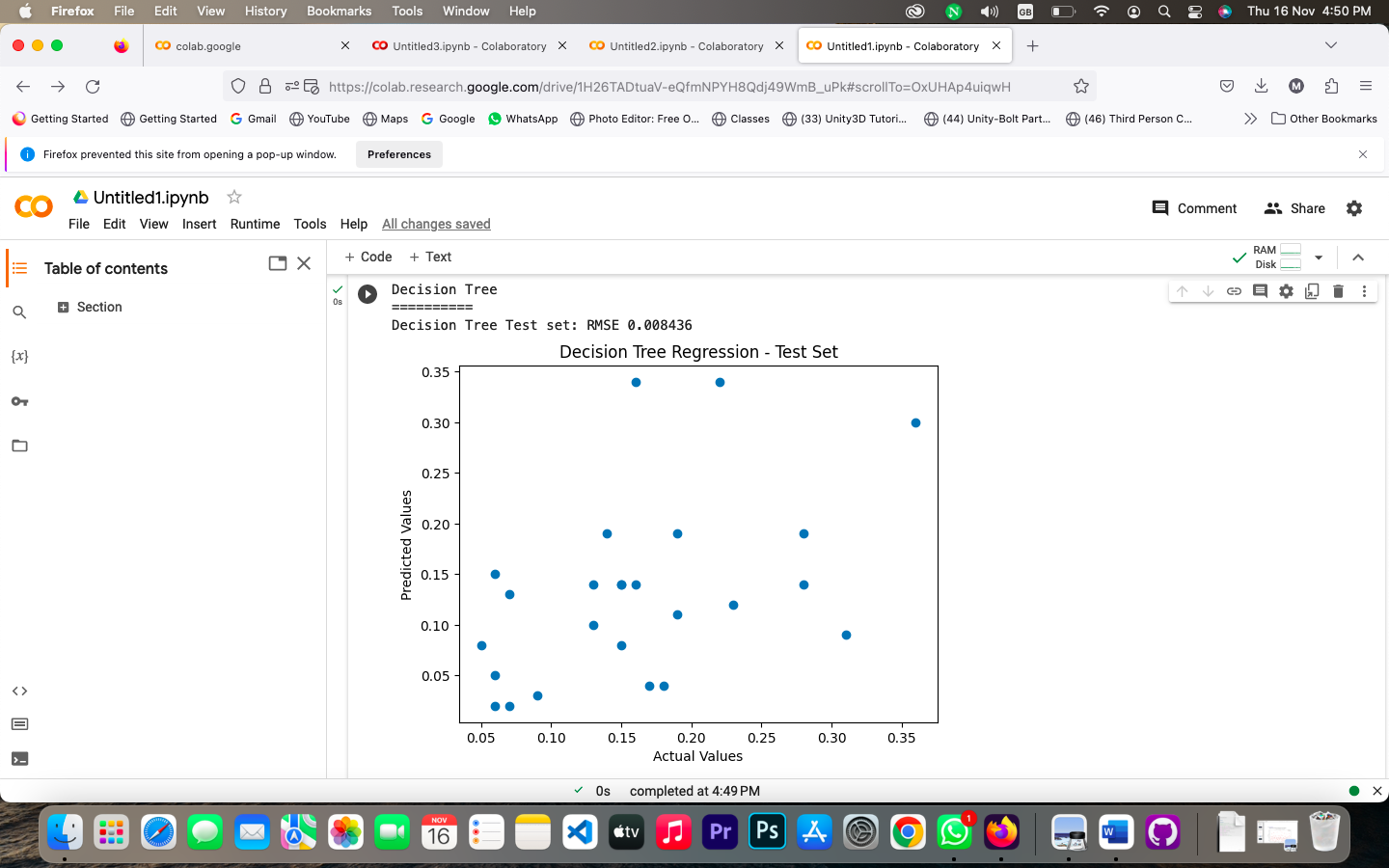
plt.title('Decision Tree Regression - Test Set')

plt.xlabel('Actual Values')

plt.ylabel('Predicted Values')

plt.show()

## **Output:**



## **Lab Task 4:**

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

print("Random Forest Regressor")

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

y\_pred\_train3 = model1.predict(X\_train)

RMSE\_train3 = mean\_squared\_error(y\_train,y\_pred\_train3)

print("Random Forest Regressor TrainSet: RMSE {}".format(RMSE\_train3))

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

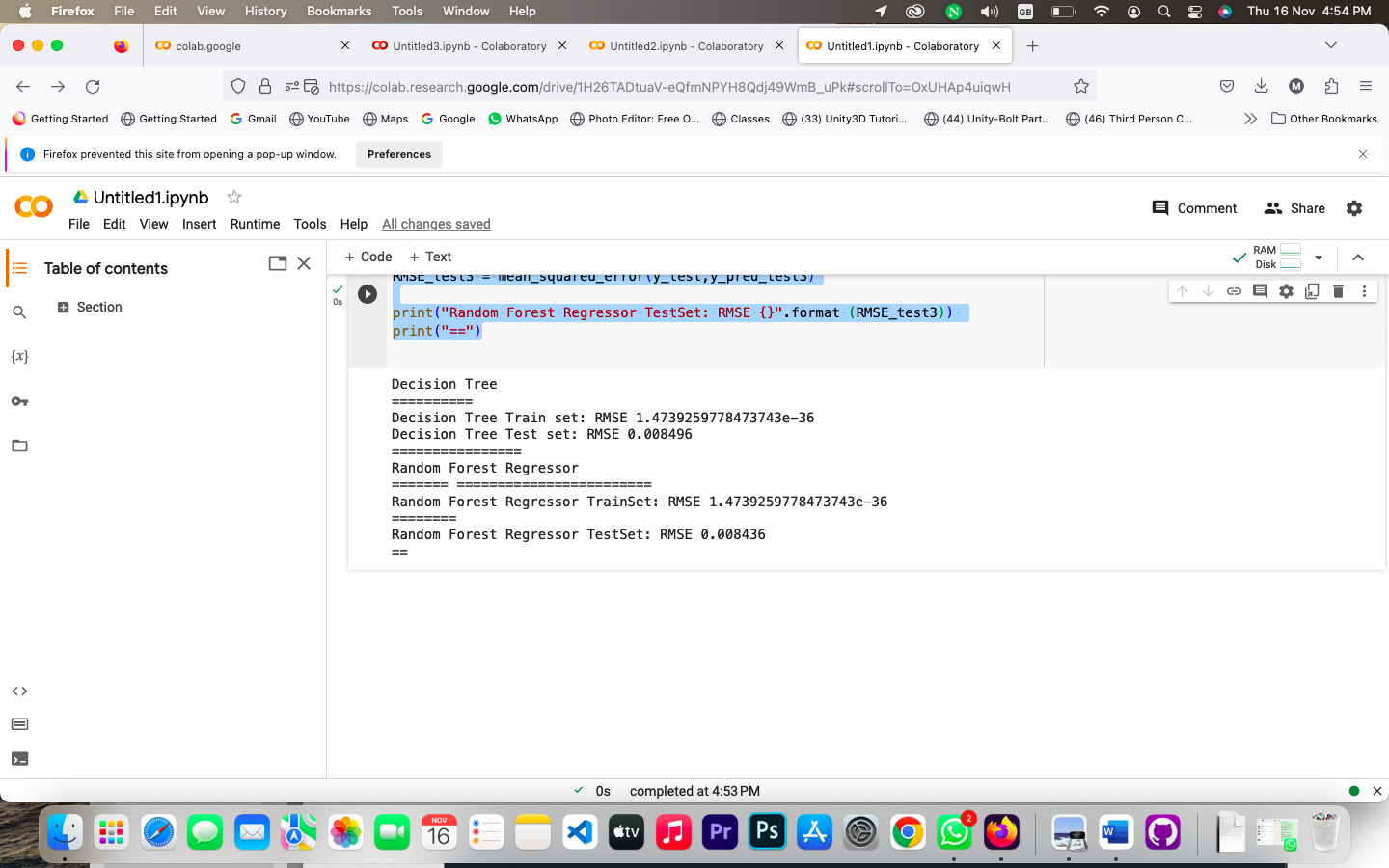
y\_pred\_test3 = model1.predict(X\_test)

RMSE\_test3 = mean\_squared\_error(y\_test,y\_pred\_test3)

print("Random Forest Regressor TestSet: RMSE {}".format (RMSE\_test3))

print("==")

## **Output:**



## **Lab Task 5:**

plt.scatter(y\_test, y\_pred\_test2, label='Predicted Values')

plt.scatter(y\_test, y\_test, label='Actual Values', marker='x')

plt.title('Decision Tree Regression - Test Set')

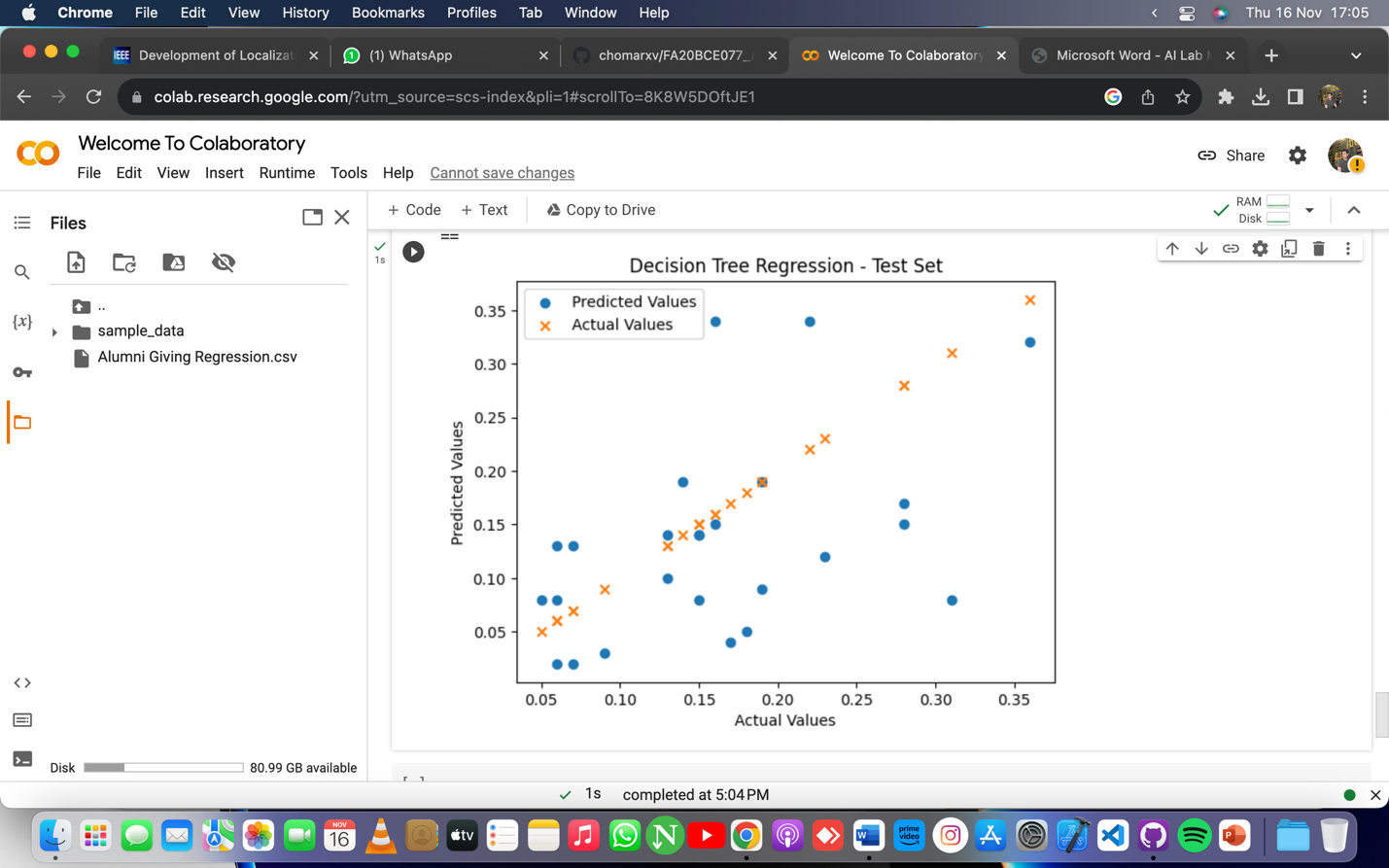
plt.xlabel('Actual Values')

plt.ylabel('Predicted Values')

plt.legend()

plt.show()

## **Output:**



# **Post Lab:**

Model 1 outperformed Model 2 and 3 on the test set, displaying the lowest RMSE and superior predictive performance. The success of Model 1 may be attributed to its capacity to capture complex data relationships. However, it's crucial to acknowledge the simplicity of Models 2 and 3, which might make them more interpretable but at the cost of predictive accuracy. Visualizations, particularly scatter plots, offered insights into the models' behavior, revealing patterns and potential outliers. These insights can inform strategies for improvement. Feature engineering and hyperparameter tuning are recommended for all models to enhance predictive capabilities.

Additionally, exploring ensemble methods such as Random Forest or Gradient Boosting could further boost overall performance. The decision on which strategy to prioritize should consider the balance between interpretability and predictive accuracy based on the specific goals and constraints of the problem.