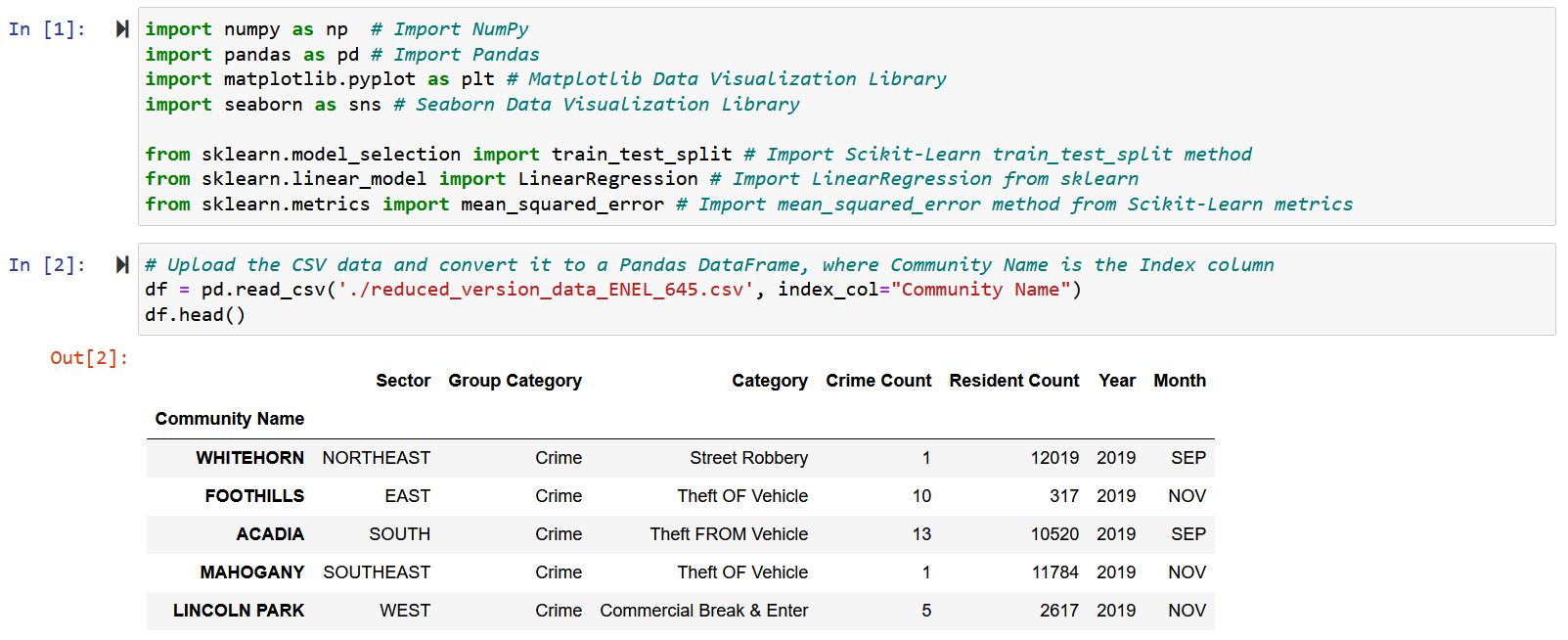
**ENEL 645 – Spring 2023 - Assignment 1 Report**

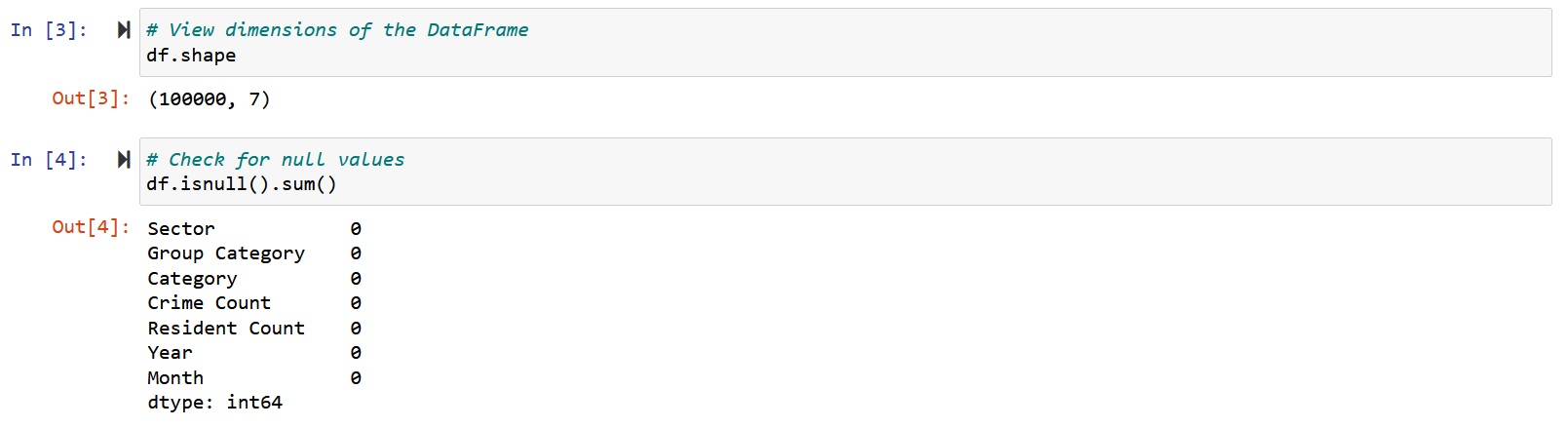
**By Michael Le (ID: 10104883)**

**Algorithm Description:**

To start, I used the pandas read\_csv method to convert the CSV containing the community crime and disorder dataset into a pandas DataFrame with the ‘Community Name’ column set as the index column of the dataset. I then used the DataFrame head() method to show the first five rows of the DataFrame, to ensure the name of the columns matches those in the CSV and that ‘Community Name’ is the index.



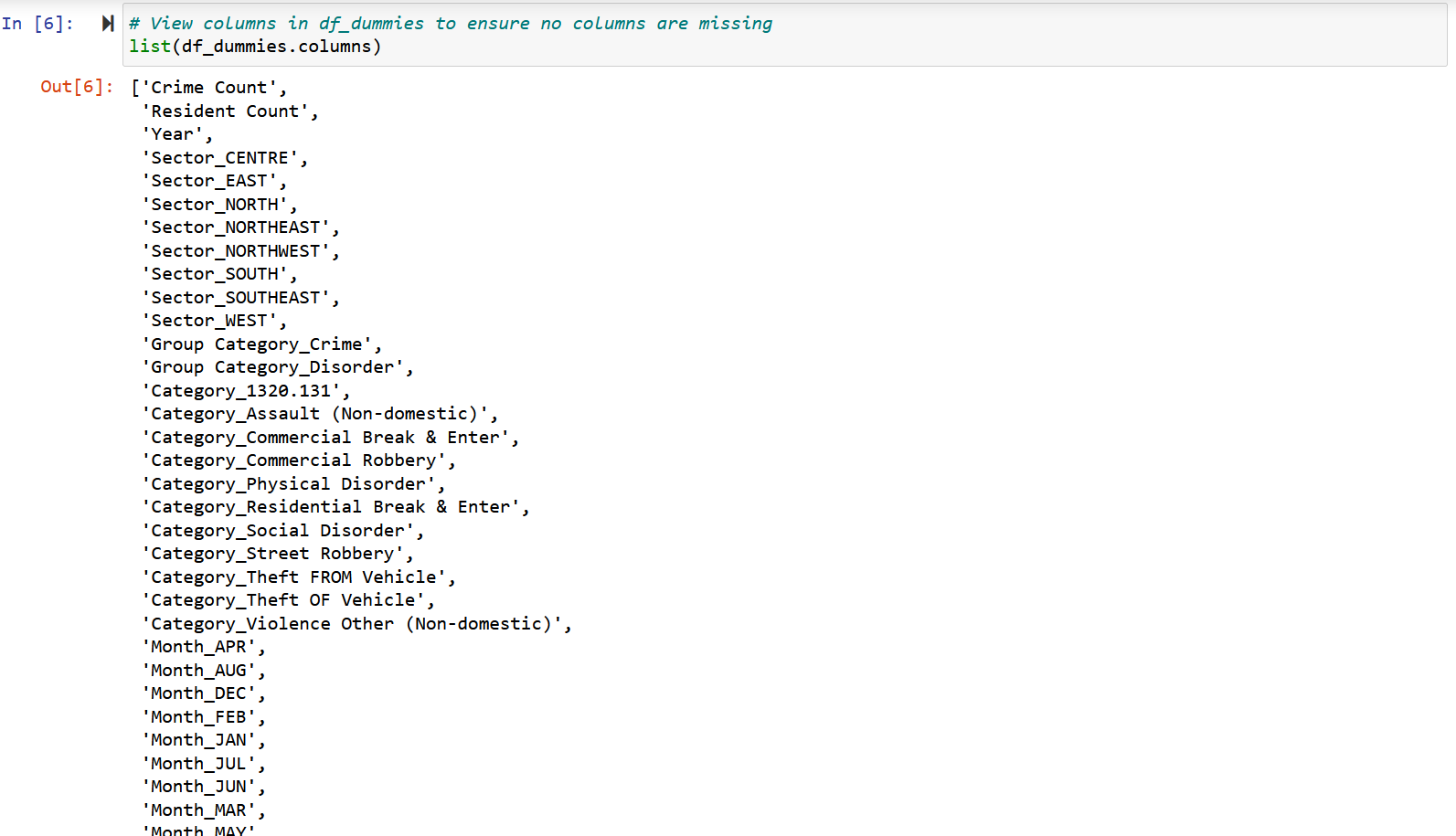
After confirming that the DataFrame displays the correct column names, I check the dimensions of the DataFrame to confirm the number of rows and columns. I also check if there are any null values in the DataFrame – fortunately, there are no null values.



Next, I have decided to keep all features (except for Community Name which is now the index) and apply one hot encoding with the get\_dummies() method from pandas for feature engineering, in order to convert categorical features like Sector, Group Category, Category, and Month to numerical values so they can be used in my model to improve predictions. The new DataFrame with these converted categorical features is called df\_dummies, as seen below. It has a dimension of 100000 rows and 36 columns.



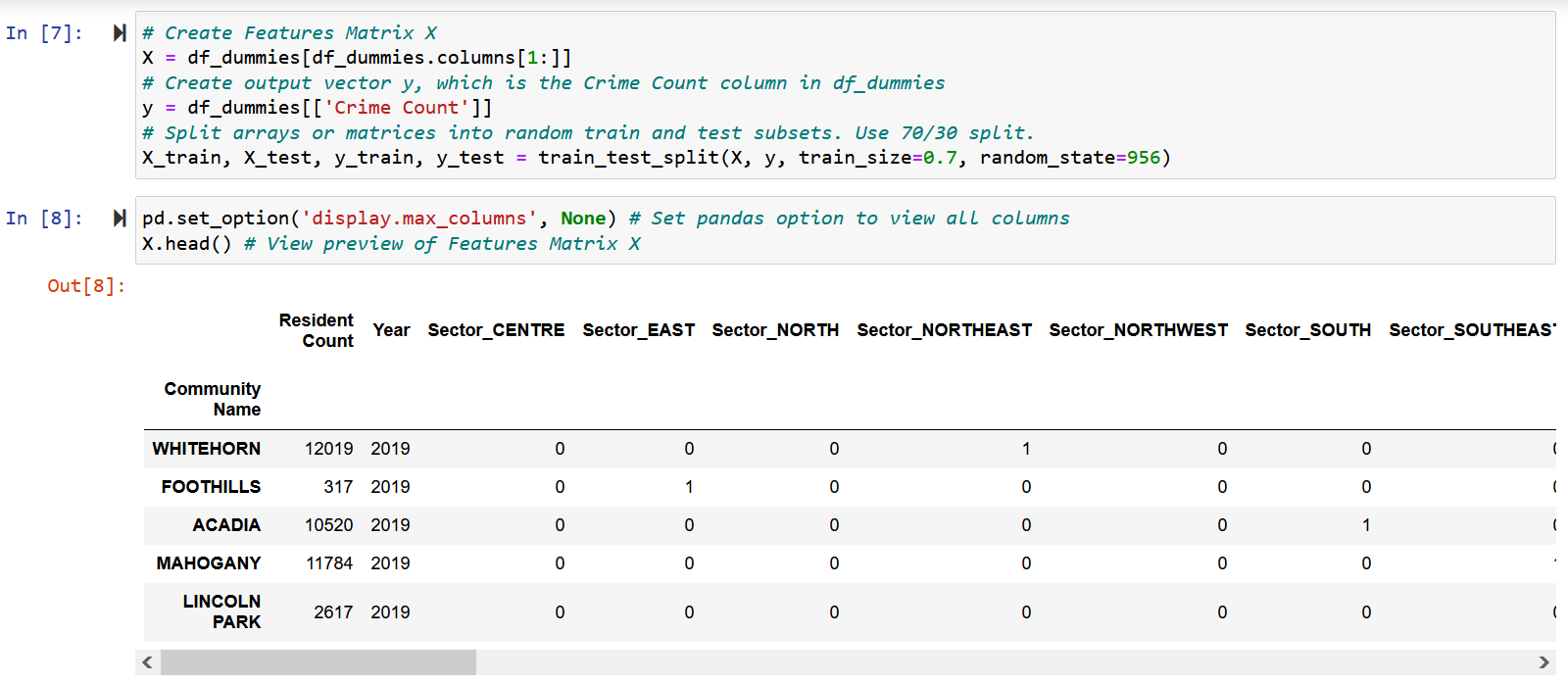
Next, I print a list of all columns inside the newly created df\_dummies to have a clearer idea of columns names in this new DataFrame.



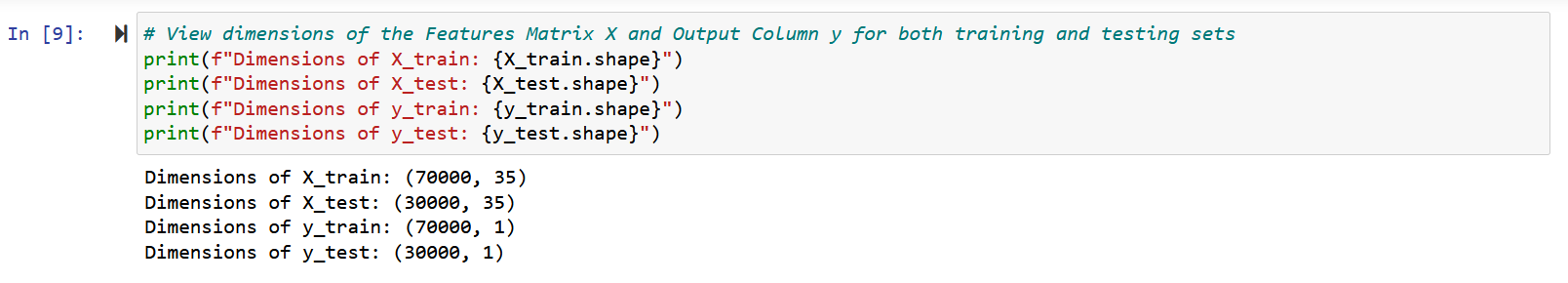
Next, I create a features matrix called X from all the columns in df\_dummies except for ‘Crime Count’, which will be the output vector y. With X and y created, I use the Scikit-Learn method train\_test\_split to randomly split the data so that the training data is comprised of 70% of the given data and the test data is comprised of the remaining 30%, as specified in the assignment requirements. A random\_state parameter of 956 is arbitrarily chosen to ensure the same random split occurs every time the code is run.

The train\_test\_split method will create a features matrix X\_train that contains features of the training data, a features X\_test that contains features of the test data, an output vector y\_train that contains the ‘Crime Count’ of the training data, and an output vector y\_test that contains the ‘Crime Count’ of the test data.

I also preview the original features matrix X to ensure that the output column ‘Crime Count’ is not included in this DataFrame.

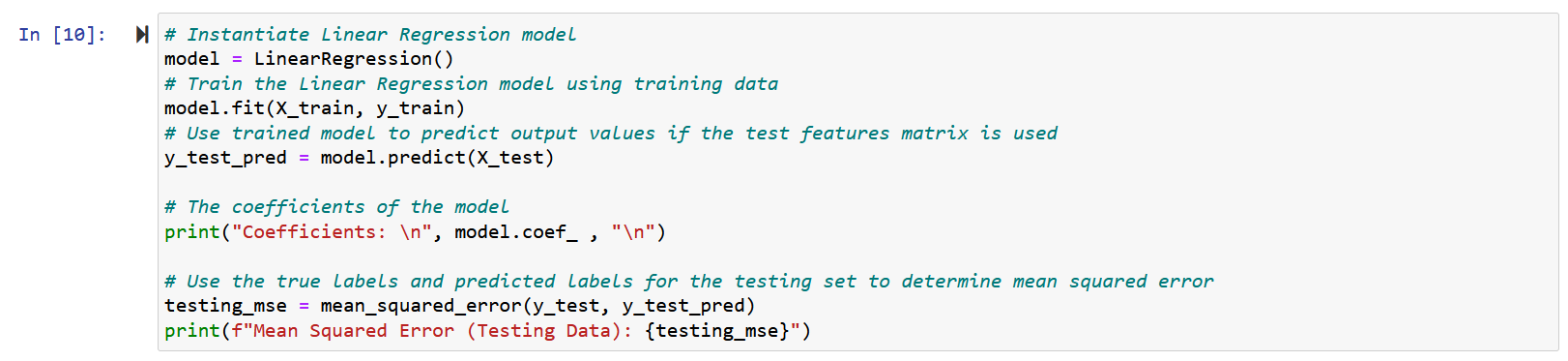


Next, I print the dimensions of the DataFrames formed by train\_test\_split to ensure that the training data contains 70% of the original 100000 rows (70000 rows) and 35/36 columns from df\_dummies, while the test data contains the remaining 30% (300000 rows) and only one column (the output column) from df\_dummies.



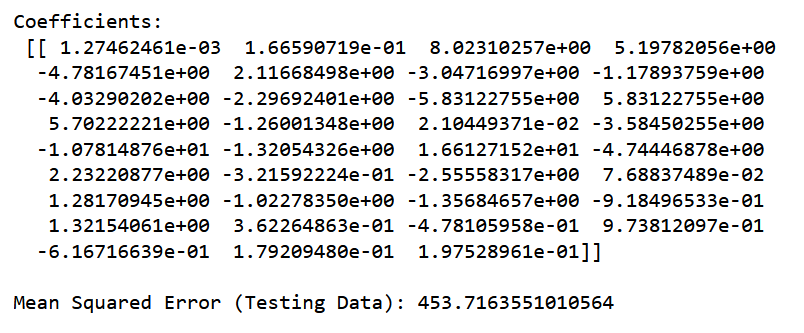
In the code block below, I am instantiating a Linear Regression model. I fit the Linear Regression model to the training data using X\_train and y\_train. Once the model is trained, I use the newly-trained model to predict Crime Count values of the remaining testing data X\_test. This predictions vector is called y\_test\_pred.

Once the prediction is calculated, I print the coefficients of the model. Finally, I also calculate and print the mean-squared error using the actual crime count values in the testing set y\_test, compared to the predicted values in y\_test\_pred.



The final results are shown in the screenshot below.

**Results:**



**Analysis:**

The performance of my model is evaluated based on mean-squared-error cost function, which is equal to 453.71635510105625. This is expected because we are using a real-world dataset from Open Calgary. The data in a Community Crime and Disorder Statistics dataset is realistic and applying a linear regression model to it will not accurately capture and predict the number of crimes in each community center.

Plotting the actual Crime Count output vector data points (y\_test) and the predicted values (y\_test\_pred) for comparison further demonstrates this point, as the output vector data points do not follow the predicted values presented by the line.

