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.

```
In [1]:  import numpy as np  # Import NumPy
         import pandas as pd # Import Pandas
         import matplotlib.pyplot as plt # Matplotlib Data Visualization Library
         import seaborn as sns # Seaborn Data Visualization Library

         from sklearn.model_selection import train_test_split # Import Scikit-Learn train_test_split method
         from sklearn.linear_model import LinearRegression # Import LinearRegression from sklearn
         from sklearn.metrics import mean_squared_error # Import mean_squared_error method from Scikit-Learn metrics
```

```
In [2]:  # Upload the CSV data and convert it to a Pandas DataFrame, where Community Name is the Index column
         df = pd.read_csv('./reduced_version_data_ENEL_645.csv', index_col="Community Name")
         df.head()
```

Out[2]:

	Sector	Group Category	Category	Crime Count	Resident Count	Year	Month
Community Name							
WHITEHORN	NORTHEAST	Crime	Street Robbery	1	12019	2019	SEP
FOOTHILLS	EAST	Crime	Theft OF Vehicle	10	317	2019	NOV
ACADIA	SOUTH	Crime	Theft FROM Vehicle	13	10520	2019	SEP
MAHOGANY	SOUTHEAST	Crime	Theft OF Vehicle	1	11784	2019	NOV
LINCOLN PARK	WEST	Crime	Commercial Break & Enter	5	2617	2019	NOV

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.

```
In [3]:  # View dimensions of the DataFrame
         df.shape

Out[3]:  (100000, 7)
```

```
In [4]:  # Check for null values
         df.isnull().sum()

Out[4]:  Sector            0
         Group Category    0
         Category          0
         Crime Count       0
         Resident Count    0
         Year              0
         Month             0
         dtype: int64
```

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.

```
In [5]: # Use get_dummies to convert categorical variables into separate indicator columns of 0s and 1s and create a new DataFrame
        # called df_dummies

        df_dummies = pd.get_dummies(df, columns=['Sector', 'Group Category', 'Category', 'Month'])
        df_dummies
```

Out[5]:

	Crime Count	Resident Count	Year	Sector_CENTRE	Sector_EAST	Sector_NORTH	Sector_NORTHEAST	Sector_NORTHWEST	Sector_SOUTH
Community Name									
WHITEHORN	1	12019	2019	0	0	0	1	0	0
FOOTHILLS	10	317	2019	0	1	0	0	0	0
ACADIA	13	10520	2019	0	0	0	0	0	1
MAHOGANY	1	11784	2019	0	0	0	0	0	0
LINCOLN PARK	5	2617	2019	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...
WOODBINE	2	9131	2013	0	0	0	0	0	1
NORTH GLENMORE PARK	2	2333	2014	0	0	0	0	0	0
HAYSBORO	5	6943	2012	0	0	0	0	0	1
FAIRVIEW INDUSTRIAL	7	0	2013	0	0	0	0	0	1
KILLARNEY/GLENGARRY	1	6870	2013	1	0	0	0	0	0

100000 rows × 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.

```
In [6]: # View columns in df_dummies to ensure no columns are missing
        list(df_dummies.columns)

Out[6]: ['Crime Count',
         'Resident Count',
         'Year',
         'Sector_CENTRE',
         'Sector_EAST',
         'Sector_NORTH',
         'Sector_NORTHEAST',
         'Sector_NORTHWEST',
         'Sector_SOUTH',
         'Sector_SOUTHEAST',
         'Sector_WEST',
         'Group Category_Crime',
         'Group Category_Disorder',
         'Category_1320.131',
         'Category_Assault (Non-domestic)',
         'Category_Commercial Break & Enter',
         'Category_Commercial Robbery',
         'Category_Physical Disorder',
         'Category_Residential Break & Enter',
         'Category_Social Disorder',
         'Category_Street Robbery',
         'Category_Theft FROM Vehicle',
         'Category_Theft OF Vehicle',
         'Category_Violence Other (Non-domestic)',
         'Month_APR',
         'Month_AUG',
         'Month_DEC',
         'Month_FEB',
         'Month_JAN',
         'Month_JUL',
         'Month_JUN',
         'Month_MAR',
         'Month_MAY'
```

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.

```
In [7]:  # Create Features Matrix X
         X = df_dummies[df_dummies.columns[1:]]
         # Create output vector y, which is the Crime Count column in df_dummies
         y = df_dummies[['Crime Count']]
         # Split arrays or matrices into random train and test subsets. Use 70/30 split.
         X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, random_state=956)
```

```
In [8]:  pd.set_option('display.max_columns', None) # Set pandas option to view all columns
         X.head() # View preview of Features Matrix X
```

Out[8]:

	Resident Count	Year	Sector_CENTRE	Sector_EAST	Sector_NORTH	Sector_NORTHEAST	Sector_NORTHWEST	Sector_SOUTH	Sector_SOUTHEAS
Community Name									
WHITEHORN	12019	2019	0	0	0	1	0	0	(
FOOTHILLS	317	2019	0	1	0	0	0	0	(
ACADIA	10520	2019	0	0	0	0	0	1	(
MAHOGANY	11784	2019	0	0	0	0	0	0	'
LINCOLN PARK	2617	2019	0	0	0	0	0	0	(

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 [9]:  # View dimensions of the Features Matrix X and Output Column y for both training and testing sets
         print(f"Dimensions of X_train: {X_train.shape}")
         print(f"Dimensions of X_test: {X_test.shape}")
         print(f"Dimensions of y_train: {y_train.shape}")
         print(f"Dimensions of y_test: {y_test.shape}")

         Dimensions of X_train: (70000, 35)
         Dimensions of X_test: (30000, 35)
         Dimensions of y_train: (70000, 1)
         Dimensions of y_test: (30000, 1)
```

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.

```python
In [10]:
# Instantiate Linear Regression model
model = LinearRegression()
# Train the Linear Regression model using training data
model.fit(X_train, y_train)
# Use trained model to predict output values if the test features matrix is used
y_test_pred = model.predict(X_test)

# The coefficients of the model
print("Coefficients: \n", model.coef_ , "\n")

# Use the true labels and predicted labels for the testing set to determine mean squared error
testing_mse = mean_squared_error(y_test, y_test_pred)
print(f"Mean Squared Error (Testing Data): {testing_mse}")
```

The final results are shown in the screenshot below.

Results:

```
Coefficients:
 [[ 1.27462461e-03  1.66590719e-01  8.02310257e+00  5.19782056e+00
  -4.78167451e+00  2.11668498e+00 -3.04716997e+00 -1.17893759e+00
  -4.03290202e+00 -2.29692401e+00 -5.83122755e+00  5.83122755e+00
   5.70222221e+00 -1.26001348e+00  2.10449371e-02 -3.58450255e+00
  -1.07814876e+01 -1.32054326e+00  1.66127152e+01 -4.74446878e+00
   2.23220877e+00 -3.21592224e-01 -2.55558317e+00  7.68837489e-02
   1.28170945e+00 -1.02278350e+00 -1.35684657e+00 -9.18496533e-01
   1.32154061e+00  3.62264863e-01 -4.78105958e-01  9.73812097e-01
  -6.16716639e-01  1.79209480e-01  1.97528961e-01]]

Mean Squared Error (Testing Data): 453.7163551010564
```

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.

Plot of Calgary Community Crime and Disorder Statistics Dataset and Linear Regression Model Fit