**COMP 237 - Online lab assignment “Linear Regression”**

Purpose: Written Response of the lab assignment

**Exercise 1:**

The requirement of this exercise was to use uniform distribution to sample 100 numbers ranging from 0 to 1 using the seed the last two digits of my students id.

Code Section:

Before adding Noise, the code does the following:

Imports: It imports the necessary libraries, NumPy for numerical operations and Matplotlib for data visualization.

Seed Initialization: A random seed (84) is set to ensure reproducibility of random numbers generated in the code.

Generating Random Data: It generates 100 random uniform values between 0 and 1 and stores them in the variable x.

Setting Y Values: It computes the corresponding y values by applying a linear transformation: y = 12 \* x - 4.

Data Visualization: The code uses Matplotlib to create a scatter plot of the data points, with x values on the x-axis and y values on the y-axis. The plot is labeled as "Sampling 100 random uniform data (Before Noise)."

After adding the Noise, the code does the following:

Adding Noise: It introduces noise to the y distribution by generating random uniform values between 0 and 0.1 and adding them to the y values. This introduces variation to the data, simulating real-world scenarios where data is not perfectly linear.

Data Visualization (Again): A scatter plot is created again, this time representing the data with noise. The plot is labeled as "Sampling 100 random uniform data (After Noise)."

Output:

A screen shot of a computer screen

Description automatically generated

Output in graph:

A graph of a random number of numbers

Description automatically generated with medium confidenceA graph of a random number of numbers

Description automatically generated with medium confidenceA line of dots on a white background

Description automatically generated

Although, the difference between the noise added value and the normal value wasn’t that different, the graph also made slight changes. The changes can be seen in third figure. Where blue represents data before the noise and green represents data with the noise.

Conclusion:

The code successfully generates a dataset of 100 points (x and y) with a linear relationship between them. The addition of noise in the second section introduces variability.

**Exercise 2:**

This exercise involved working with a commerce website dataset. First the data need to be loaded and then perform initial exploratory analysis, transform categorical variables, normalize data, and visualize it. Then, build linear regression models to predict "Total Spend" using features like "Monthly Income," "Transaction Time," and dummy variables. This exercise needed to compare two models, one with and one without the "Record" feature and analyze their results.

Code Section:

Imports: The code begins by importing necessary libraries, including Pandas for data manipulation, NumPy for numerical operations, os for file path handling, and Matplotlib for data visualization.

Loading Data: It specifies the path to the CSV file and reads it into a Pandas DataFrame named ecom\_exp\_ujjwal.

Initial Exploration:

* It prints the first three records in the dataset using head(3).
* Displays the shape of the DataFrame using shape.
* Lists the column names using columns.
* Lists the data types of columns using dtypes.
* Counts missing values per column using isna().sum() and prints the count of missing values for each column.

Data Transformation:

* It creates dummy variables for the "Gender" and "CityTier" columns, effectively converting categorical variables into numerical ones.
* The dummy variables are combined with the original DataFrame, and the original categorical columns are dropped.
* The "TransactionID" column is dropped from the DataFrame.
* It defines a function normalize\_dataframe to normalize the data in the DataFrame using the formula (x - min) / (max - min).
* The function is called on the transformed DataFrame.
* The code prints the first two records of the normalized DataFrame.
* It generates histograms for all variables in the DataFrame using hist and sets the figure size.

Splitting the Data:

* The code imports necessary libraries for building a linear regression model, including LinearRegression and train\_test\_split from scikit-learn.
* It specifies a seed for randomization.
* It selects the feature columns and the target variable.
* It splits the data into training and testing sets using train\_test\_split.
* A linear regression model is created and fitted to the training data.
* The model coefficients (weights) and R^2 (model score) are printed.

Using "Record" Feature:

* Similar to the previous steps, the code selects additional features by including the "Record" column.
* It repeats the splitting and model fitting process.
* The model coefficients and R^2 are printed again for this updated model.

Plotting Data:

The code plots all the graphs.

1. Scatter plot for Monthly Income vs Total Spend.
2. Scatter plot for Transaction Time vs Total Spend.
3. Scatter plot for Gender vs Total Spend (Assuming 'Female' and 'Male' as binary).
4. Scatter plot for City Tier vs Total Spend (Assuming 'Tier 1', 'Tier 2', and 'Tier 3' as binary).

Output:

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

A screen shot of a computer screen

Description automatically generated

A screen shot of a computer screen

Description automatically generated

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

A screen shot of a computer screen

Description automatically generated

Graphs:

A graph with a number of dots and numbers

Description automatically generated with medium confidenceA graph with numbers and symbols

Description automatically generated with medium confidenceA green dots on a white background

Description automatically generatedA graph showing a number of blue dots

Description automatically generatedA grid of blue dots

Description automatically generated

A green graph with numbers and a few green bars

Description automatically generated with medium confidence

Comparison:

1. Model with "Record" Feature:

a. Higher R-squared (R2s) value: This indicates that the model with the "Record" feature included explains more of the variance in the "Total Spend" variable compared to the model without it. It suggests that the "Record" feature is contributing to a better fit for the model.

b. Negative coefficient for "Record" feature: A negative coefficient suggests that the "Record" feature has a negative relationship with the "Total Spend." In other words, as the "Record" feature increases, the "Total Spend" tends to decrease.

2. Model without "Record" Feature:

Lower R-squared (R2) value: This indicates that the model without the "Record" feature explains less of the variance in the "Total Spend" variable compared to the model with the "Record" feature. It suggests that the "Record" feature might be an important contributor to explaining the variance.

Positive coefficient for other features: The positive coefficients for the other features in this model imply that, without the "Record" feature, these other features have a positive relationship with "Total Spend." As these features increase, "Total Spend" tends to increase.

Conclusion:

The model with the "Record" feature included outperforms the model without the "Record" feature in terms of explaining the variance in "Total Spend" as indicated by a higher R-squared (R2) value. The positive R-squared value suggests that the model with "Record" is better at capturing and explaining the variability in the "Total Spend" variable.