

Assignment 3 Analysis Report: Linear Regression

Exercise 1: Sampling and Noise

The purpose of this exercise was to examine the impact of adding noise to a linear function. X was defined by randomly selecting 100 values from a uniform distribution with a seed value of 68. Y was defined by the function $y=12x-4$ and a scatter-plot was generated to demonstrate this relationship (Figure 1). Next, noise was randomly generated from a Gaussian distribution and the scatter-plot was regenerated with the added noise (Figure 2).

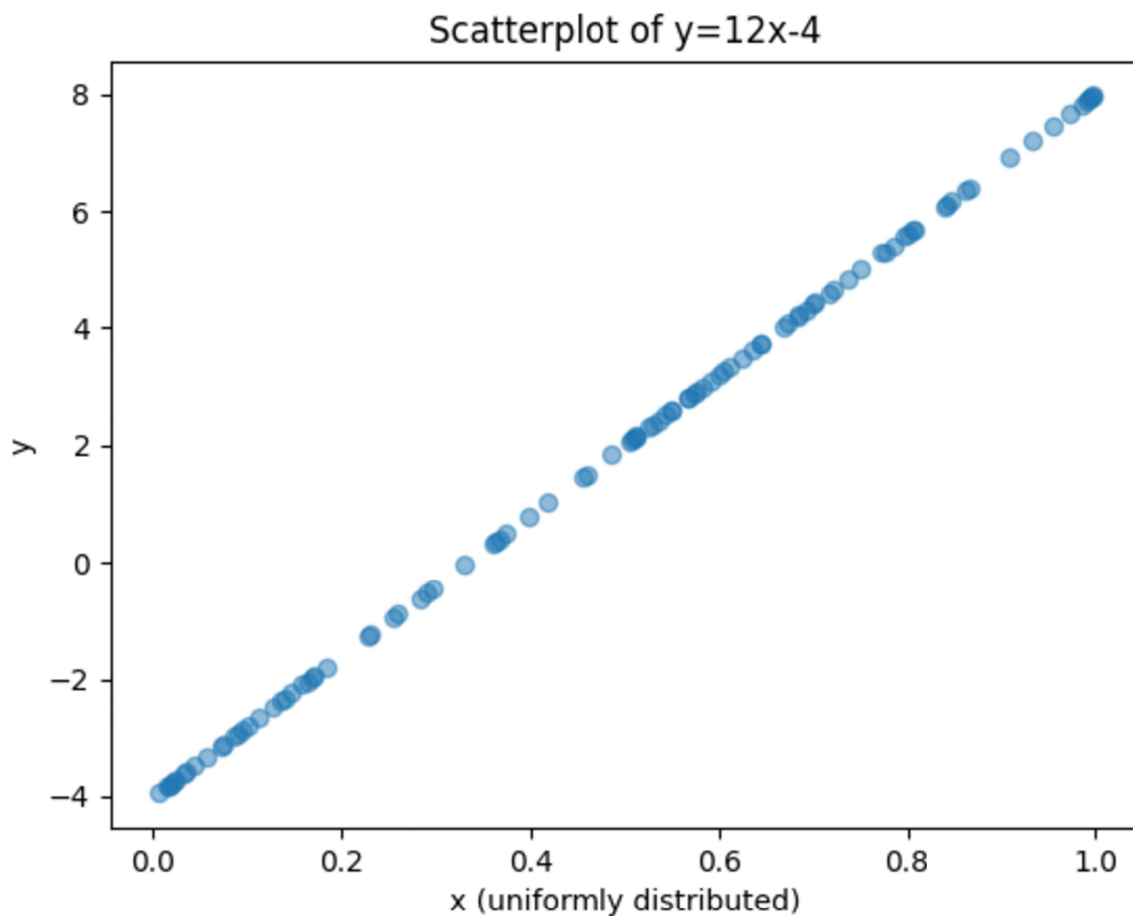


Figure 1: Scatter-plot of $y=12x-4$

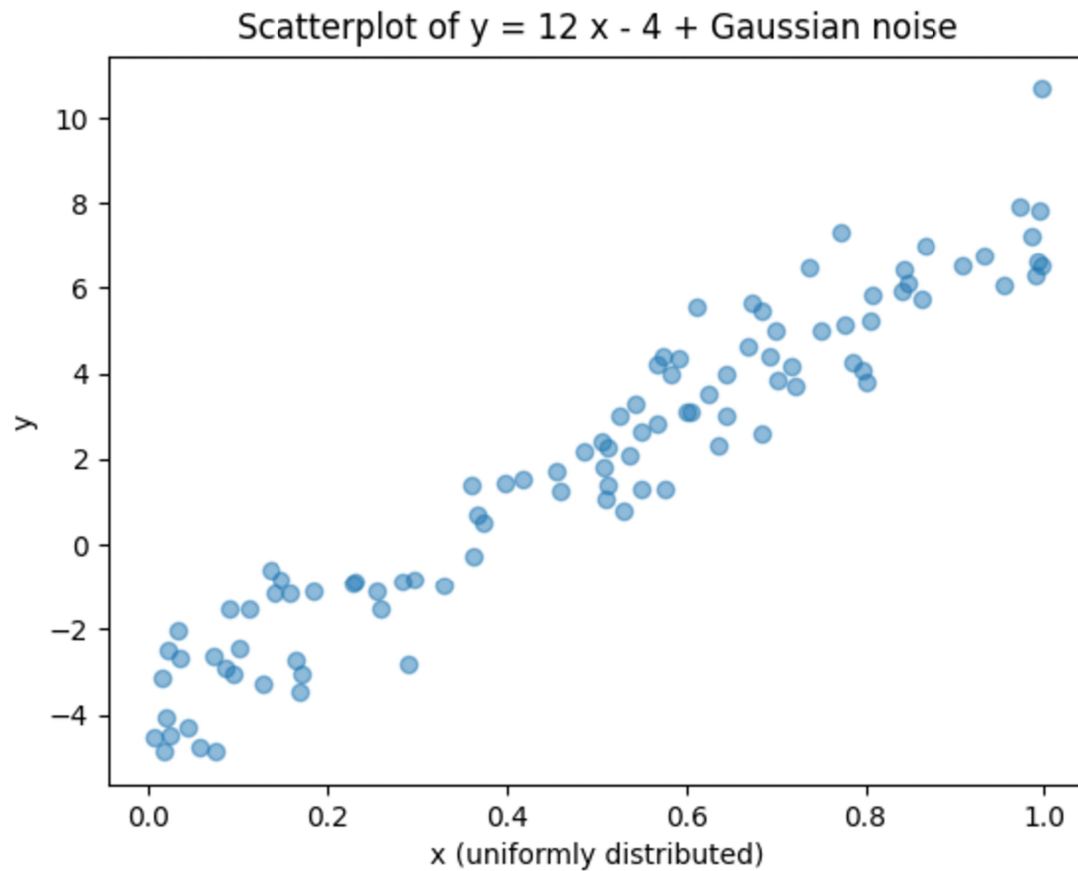


Figure 2: Scatter-plot of $y = 12x - 4 + \text{noise}$

Comparing the two plots seen above, we can see the addition of noise has scattered the points randomly outward from the line $y=12x-4$. Though the data is noticeably more random, a best fit line would still be linear. Because the noise is normally distributed, the values differ enough to create realistic noise useful for experimentation, by adding an approximated realism into data. If the noise was uniformly distributed, for instance, it would have similar impact to adding a constant and the data would more strongly linear, like in figure 1.

Exercise 2: Commerce website predictions

The purpose of this exercise was to explore the built-in functionalities of important Python libraries including Pandas, Matplotlib and Scikit-learn. Sample expense data from the provided “Ecom Expense.csv” file was loaded in to a Pandas Dataframe object `ecom_exp_hannah`.

Table 1: All variables and their data types from the original “Ecom Expenses.csv” dataset

Variable	Data type
Transaction ID	Categorical
Age	Numerical
Items	Numerical

Monthly Income	Numerical
Transaction Time	Numerical
Record	Numerical
Gender	Categorical
City Tier	Categorical
Total Spend	Numerical

During the exploration stage of the experiment, different metrics about the data are printed to the terminal including the first three records, the shape of the *ecom_exp_hannah* dataframe, the column names and types of columns, and the number of missing data values per column (Figure 3).

```

(.venv) PS C:\Users\hanna\Desktop\237assignment3\Exercise#2_hannah> python3 hannah_linear.py
The first three records:
Transaction ID Age Items Monthly Income Transaction Time Record Gender City Tier Total Spend
0 TXN001 42 10 7313 622.668127 5 Female Tier 1 4198.355084
1 TXN002 24 8 17747 126.904567 3 Female Tier 2 4134.976648
2 TXN003 47 11 22845 873.469701 2 Male Tier 2 5166.614455

DataFrame shape:
(2362, 9)

Column names:
Index(['Transaction ID', 'Age', 'Items', 'Monthly Income',
      'Transaction Time', 'Record', 'Gender', 'City Tier', 'Total Spend'],
      dtype='object')

Column datatypes:
Transaction ID object
Age int64
Items int64
Monthly Income int64
Transaction Time float64
Record int64
Gender object
City Tier object
Total Spend float64
dtype: object

Number of missing values by column:
Transaction ID 0
Age 0
Items 0
Monthly Income 0
Transaction Time 0
Record 0
Gender 0
City Tier 0
Total Spend 0
dtype: int64

First two records of normalized dataset:
Age Items Monthly Income Transaction Time Record Total Spend Gender_Female Gender_Male City Tier_Tier 1 City Tier_Tier 2 City Tier_Tier 3
0 0.55 0.642857 0.189663 0.627058 0.5 0.241242 1.0 0.0 1.0 0.0 0.0
1 0.10 0.500800 0.552346 0.126412 0.3 0.236305 1.0 0.0 0.0 1.0 0.0

MODEL: without 'Record'
Model coefficients:
[-5.50908086e-16 1.11022302e-16 1.00000000e+00 1.94289029e-16

```

Figure 3: Screenshot of exercise 2b (initial exploration) terminal output

Next, the data is transformed to interface with Scikit-learn's linear regression function. Pandas *get_dummies* is used to create a one hot encoding of each categorical variable (Gender and City Tier). The result is a sparse binary matrix with one column for each possible value. The two new encoded data structures *gender_encoded* and *city_encoded* are joined with the main dataframe *ecom_exp_hannah*. The unwanted Transaction ID column is dropped along with the original categorical Gender and City Tier columns. A function *normalize()* is defined to normalize numerical data in a dataframe according to the equation in figure 5. The *ecom_exp_hannah* is passed to the *normalize()* function and the normalized dataframe is returned. The first two normalized records are displayed to the terminal (Figure 3).

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Figure 5: Equation used for normalization of the dataset

A histogram was generated (Figure 6) for each variable in the dataset. A scatter-plot matrix was generated (Figure 7) to visualize the relationship between the following four variables:

1. Age
2. Monthly income
3. Transaction time
4. Total spend

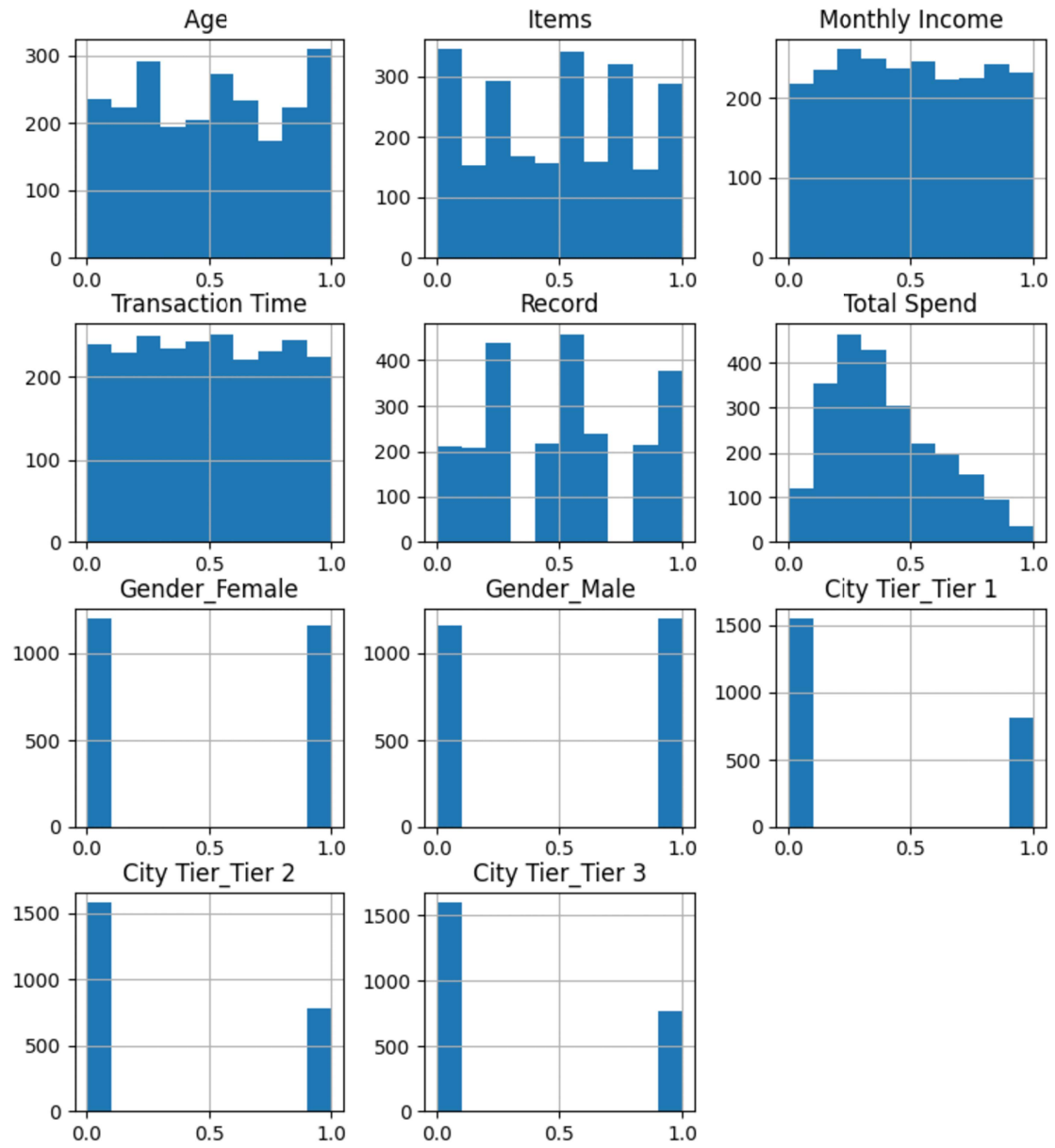


Figure 6: One histogram is generated for each variable in the dataset

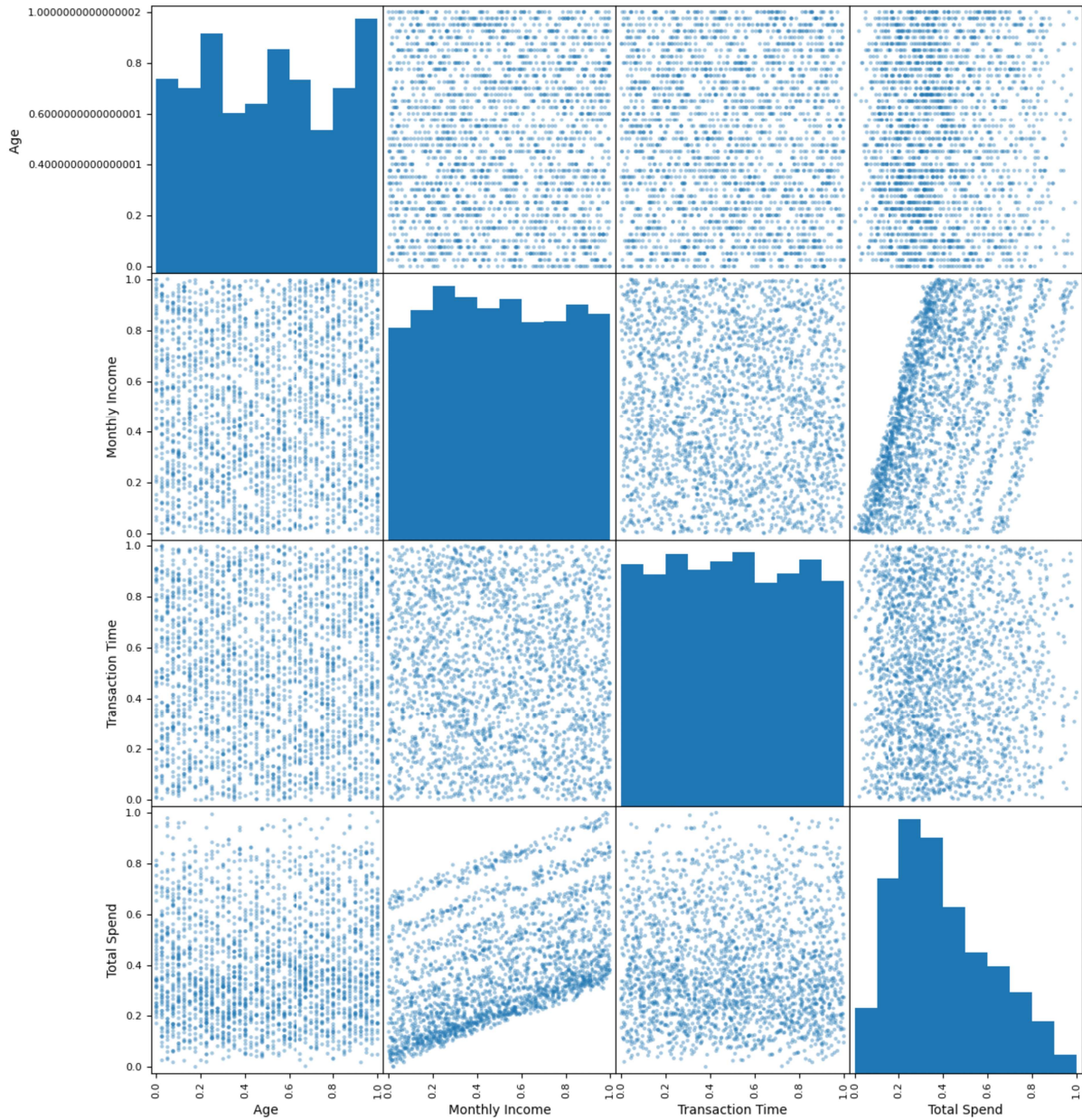


Figure 7: Scatter-plot matrix demonstrating the relationships between Age, Monthly Income, Transaction Time, and Total Spend

The relationship between the predictor variables (Monthly income, Transaction time, Gender_encoded, and CityTier_encoded) and the output variable (Total spend) is assumed to be linear. The Age, Items, and Records columns are dropped and the Scikit-learn function *train_test_split()* is

used to divide the dataset into 65% training data and 35% testing data. A linear regression model was fit to the training data and the model's resulting coefficients and score is printed to the terminal. The Records column is added back into the dataset, *train_test_split()* is run again, and the new resulting coefficients and score are printed to the terminal (Figure 8).

```

(.venv) PS C:\Users\hanna\Desktop\237assignment3\Exercise#2_hannah> python3 hannah_linear.py
Transaction ID      object
Age                int64
Items              int64
Monthly Income      int64
Transaction Time     float64
Record              int64
Gender              object
City Tier           object
Total Spend         float64
dtype: object

Number of missing values by column:
Transaction ID      0
Age                 0
Items               0
Monthly Income      0
Transaction Time     0
Record              0
Gender              0
City Tier           0
Total Spend         0
dtype: int64

First two records of normalized dataset:
   Age  Items  Monthly Income  Transaction Time  Record  Total Spend  Gender_Female  Gender_Male  City Tier_Tier 1  City Tier_Tier 2  City Tier_Tier 3
0  0.55  0.642857  0.189663  0.627058  0.5  0.241242  1.0  0.0  1.0  0.0  0.0
1  0.10  0.500000  0.562346  0.126412  0.3  0.236305  1.0  0.0  0.0  1.0  0.0

MODEL: without 'Record'

Model coefficients:
[-5.50908065e-16  1.11022302e-16  1.00000000e+00  1.94289029e-16
 1.80411242e-16  9.02056208e-17  9.02056208e-17  9.71445147e-17]

Model score:
1.0

MODEL: with 'Record'

Model coefficients:
[-1.06166661e-16  3.33066907e-16 -4.99600361e-16  1.00000000e+00
 2.49800181e-16  5.55111512e-17  1.24900090e-16  9.71445147e-17
-2.77555756e-17]

Model score:
1.0
  
```

Figure 8: Screenshot of terminal output for exercise 2d

In both cases with and without “Record” the linear regression fit results in a score of 1.0, which suggests the fit is exactly linear. The resulting coefficients for both cases are displayed in Table 2 below. The histogram for “Record” seen in figure 6 suggests there is no distribution for this variable, therefore incorporating it or not into the fit will not impact the shape so the fit remains perfectly linear.

Table 2: Comparing the resulting coefficients and score for Linear Regression with and without the “Record” variable

	Linear Regression coefficient without “Record”	Linear Regression coefficient with “Record”
w0	-5.50908065e-16	-1.06166661e-16
Monthly income	1.11022302e-16	3.33066907e-16
Transaction Time	1.00000000e+00	-4.99600361e-16
Record	n/a	1.00000000e+00
Gender = female	1.94289029e-16	2.49800181e-16
Gender = male	1.80411242e-16	5.55111512e-17
City tier = 1	9.02056208e-17	1.24900090e-16
City tier = 2	9.02056208e-17	9.71445147e-17
City tier = 3	9.71445147e-17	-2.77555756e-17

Model score	1.0	1.0
-------------	-----	-----