Department of Electronic and Telecommunication Engineering

University of Moratuwa

EN3150 – Pattern Recognition

Assignment 01

Anjula M.K

210368V

Date- 02/09/2024

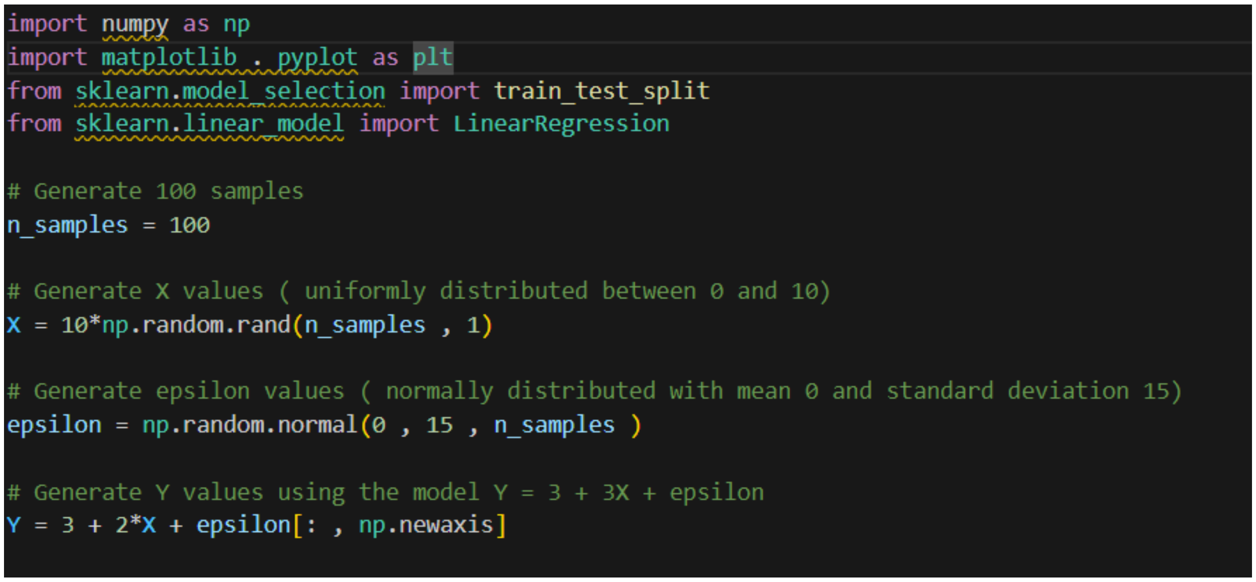
Table of Contents

1.Data Pre-processing

1. According to figure 1, Feature 1, most of its data points are in the range of -1 to 1. So, max-abs scaling will most likely to be the choice. This scaling technique keeps the feature's original structure intact by preserving its spread and center, while normalizing it to a comparable scale as other features without altering the distribution.
2. Feature 2, since there is a wide range with high variability, standard scaling is more appropriate. It will center the feature around 0 and normalize the variance, preserving the relative distances and distribution of the data, which is beneficial for models that assume a normal distribution.

2.Learning From Data

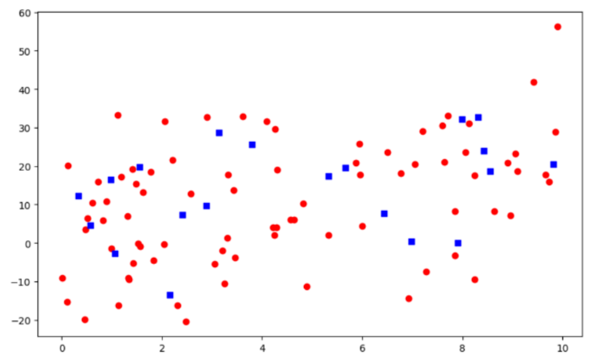
1.

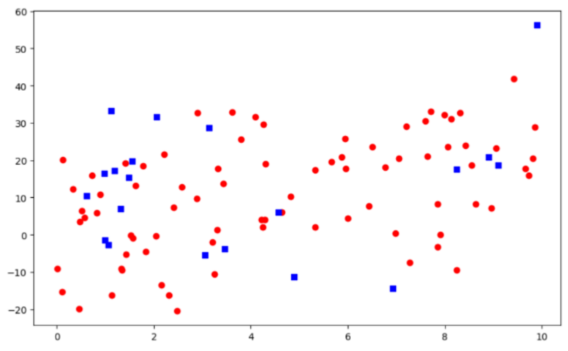


```
import numpy as np
import matplotlib . pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# Generate 100 samples
n_samples = 100

# Generate X values ( uniformly distributed between 0 and 10)
X = 10*np.random.rand(n_samples , 1)

# Generate epsilon values ( normally distributed with mean 0 and standard deviation 15)
epsilon = np.random.normal(0 , 15 , n_samples )

# Generate Y values using the model Y = 3 + 3X + epsilon
Y = 3 + 2*X + epsilon[: , np.newaxis]
```

2.

The training and testing data are different in each run because the random_state parameter in the train_test_split function is set to a random integer (r). This random integer determines how the dataset is shuffled before splitting, leading to a different selection of data points for the training and testing sets every time the code is run. This will help to validate the robustness of the model across different splits.

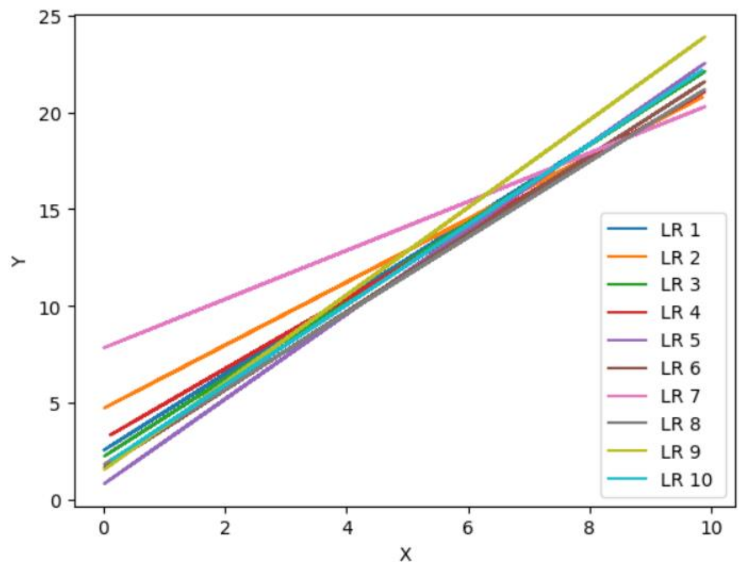




3.

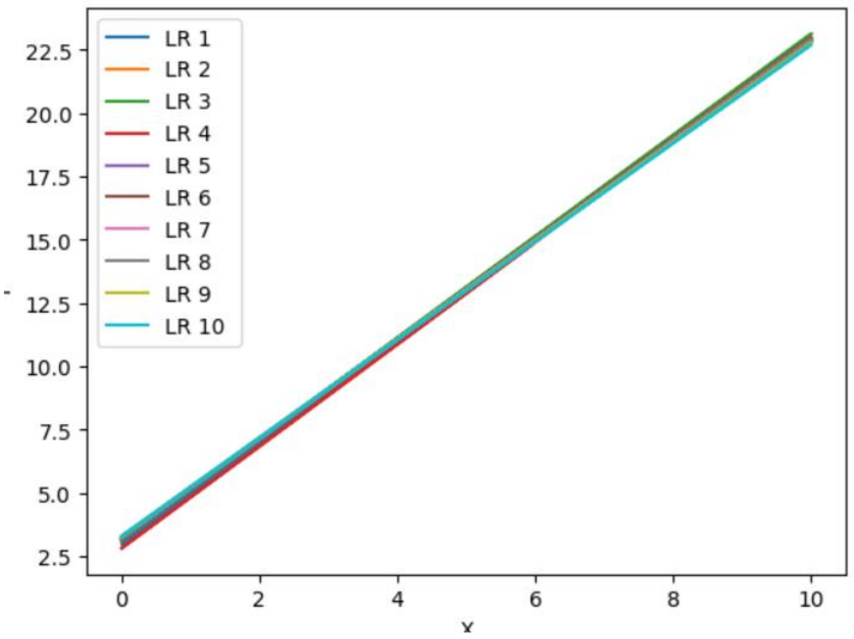
The linear regression model varies from one instance to another because the training data used to fit the model is different each time as the data is split into training and test sets using a random seed (random_state), which is generated randomly for each iteration with np.random.randint(104). As a result, the training set is different in each iteration, leading to a different fitted model each time.

Also we can see large variation in models as we use a small number of samples(100).



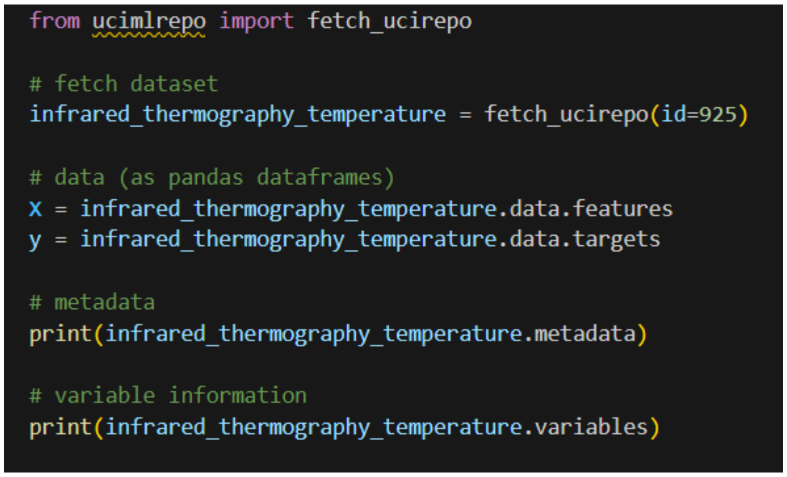
4.

With 10,000 data samples, the linear regression lines across different iterations become much more consistent compared to using only 100 samples. That means there is a higher probability of getting almost same set of training samples to create the model.This reduced variation occurs because a larger sample size provides a more representative and stable training set, leading to more reliable model parameters. The consistency improves due to the law of large numbers, which makes the model less sensitive to random variations in the data, reducing the likelihood of overfitting.



3.Linear regression on real world data

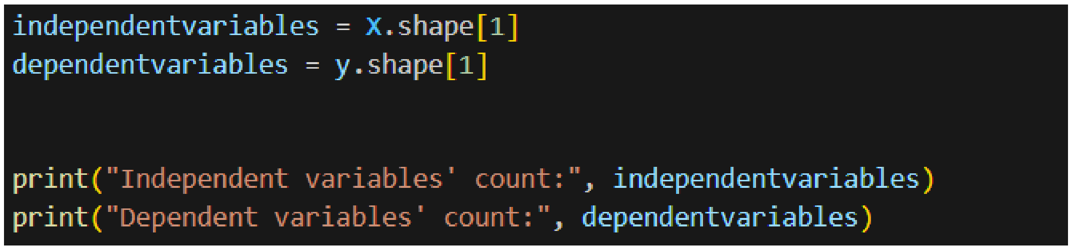
1.



```python
from ucimlrepo import fetch_ucirepo

# fetch dataset
infrared_thermography_temperature = fetch_ucirepo(id=925)

# data (as pandas dataframes)
X = infrared_thermography_temperature.data.features
y = infrared_thermography_temperature.data.targets

# metadata
print(infrared_thermography_temperature.metadata)

# variable information
print(infrared_thermography_temperature.variables)
```

2.

We can use the following code snippet to find this.



```python
independentvariables = X.shape[1]
dependentvariables = y.shape[1]


print("Independent variables' count:", independentvariables)
print("Dependent variables' count:", dependentvariables)
```

Independent variables' count: 33

Dependent variables' count: 2

3.

Initially, we cannot apply linear regression on this data set.

1. Although linear regression requires numerical inputs, this data set contains categorical features like Gender, Age, Ethnicity.
   So, we can use One-Hot Encoding or Label Encoding to convert these categorical features into numerical values.

2. This data set contains continuous features like Temperature, Humidity and Distance. So, it is good to scale those features using techniques like standard scaling or min-max scaling to ensure that they contribute equally to the model.

4.

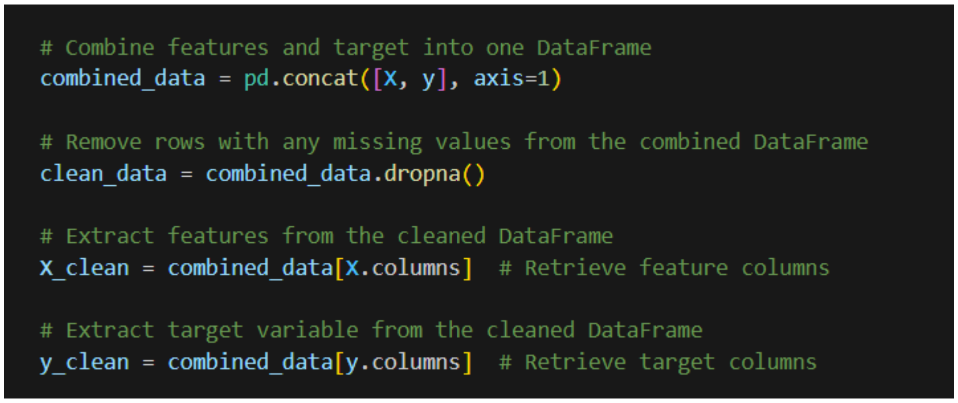
This approach of removing missing values is not entirely correct because dropping missing values separately from X and Y can result in misalignment in data between the features (X) and the target (y). This could lead to incorrect or unexpected results during model training.

According to the code below, Distance feature has 2 missing values and both the targets haven't any missing value. So, the above approach will clearly create misalignment in data between features(X) and target(y).

Correct Approach

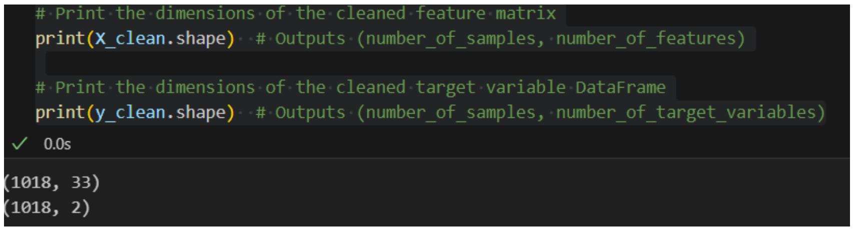
The dropna() method should be applied to the combined DataFrame, so any row with a missing value in either X or y will be remove. So, not create any misalignment in data set.

After dropping the missing values, split the combined DataFrame back into X (features) and y (target) to continue with the model training.



```
# Combine features and target into one DataFrame
combined_data = pd.concat([X, y], axis=1)

# Remove rows with any missing values from the combined DataFrame
clean_data = combined_data.dropna()

# Extract features from the cleaned DataFrame
X_clean = combined_data[X.columns]  # Retrieve feature columns

# Extract target variable from the cleaned DataFrame
y_clean = combined_data[y.columns]  # Retrieve target columns
```

We can confirm that X and y have same number of rows by below codework.

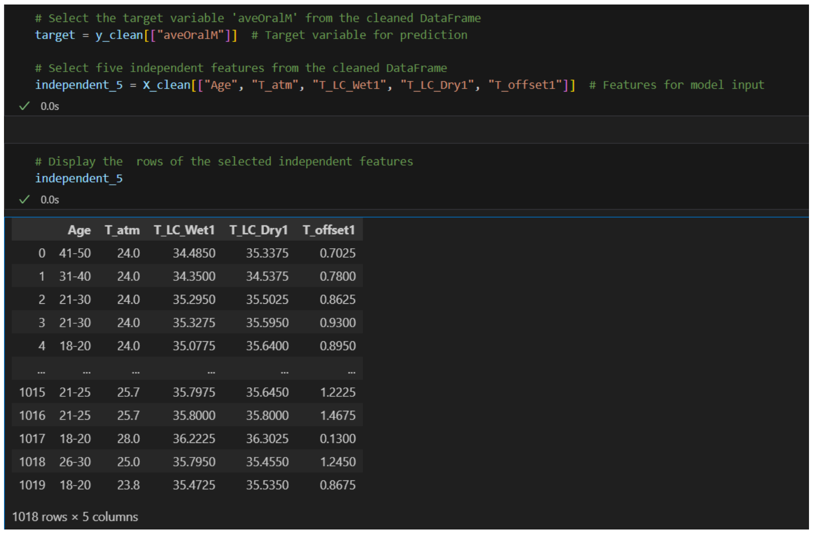


```
# Print the dimensions of the cleaned feature matrix
print(X_clean.shape)  # Outputs (number_of_samples, number_of_features)

# Print the dimensions of the cleaned target variable DataFrame
print(y_clean.shape)  # Outputs (number_of_samples, number_of_target_variables)
✓ 0.0s

(1018, 33)
(1018, 2)
```

Earlier there are 1020 rows, now there are 1018 rows for both features and targets as 2 rows are removed as 'Distance' feature has 2 missing values.

5.

I have selected “T_atm”, “T_LC_Wet1”,”T_LC_Dry1”,”T_offset1” other than “Age”



```python
# Select the target variable 'aveOralM' from the cleaned DataFrame
target = y_clean[["aveOralM"]]  # Target variable for prediction

# Select five independent features from the cleaned DataFrame
independent_5 = X_clean[["Age", "T_atm", "T_LC_Wet1", "T_LC_Dry1", "T_offset1"]]  # Features for model input
```
✓ 0.0s

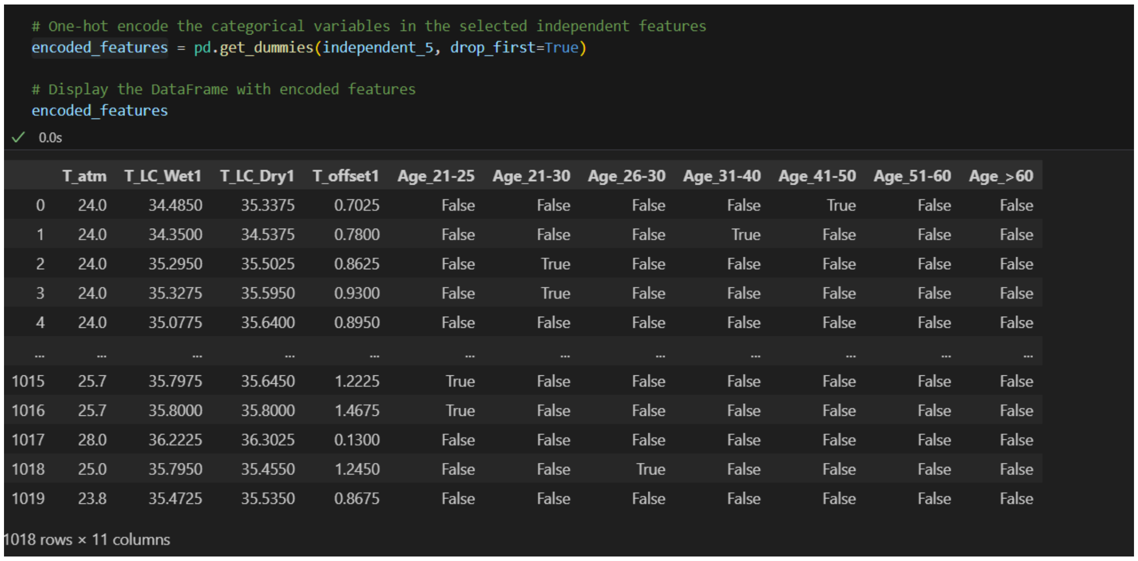
```python
# Display the  rows of the selected independent features
independent_5
```
✓ 0.0s

	Age	T_atm	T_LC_Wet1	T_LC_Dry1	T_offset1
0	41-50	24.0	34.4850	35.3375	0.7025
1	31-40	24.0	34.3500	34.5375	0.7800
2	21-30	24.0	35.2950	35.5025	0.8625
3	21-30	24.0	35.3275	35.5950	0.9300
4	18-20	24.0	35.0775	35.6400	0.8950
...	...	...	...	...	...
1015	21-25	25.7	35.7975	35.6450	1.2225
1016	21-25	25.7	35.8000	35.8000	1.4675
1017	18-20	28.0	36.2225	36.3025	0.1300
1018	26-30	25.0	35.7950	35.4550	1.2450
1019	18-20	23.8	35.4725	35.5350	0.8675

1018 rows × 5 columns

6.

Before Splitting, we have to One-Hot encode the “Age” feature as it is a categorical feature.

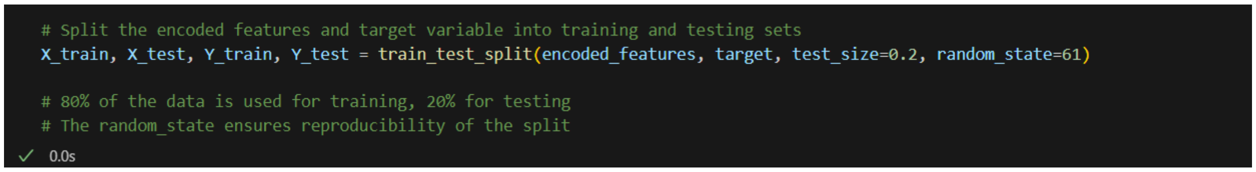


```python
# One-hot encode the categorical variables in the selected independent features
encoded_features = pd.get_dummies(independent_5, drop_first=True)

# Display the DataFrame with encoded features
encoded_features
```
✓ 0.0s

	T_atm	T_LC_Wet1	T_LC_Dry1	T_offset1	Age_21-25	Age_21-30	Age_26-30	Age_31-40	Age_41-50	Age_51-60	Age_>60
0	24.0	34.4850	35.3375	0.7025	False	False	False	False	True	False	False
1	24.0	34.3500	34.5375	0.7800	False	False	False	True	False	False	False
2	24.0	35.2950	35.5025	0.8625	False	True	False	False	False	False	False
3	24.0	35.3275	35.5950	0.9300	False	True	False	False	False	False	False
4	24.0	35.0775	35.6400	0.8950	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...
1015	25.7	35.7975	35.6450	1.2225	True	False	False	False	False	False	False
1016	25.7	35.8000	35.8000	1.4675	True	False	False	False	False	False	False
1017	28.0	36.2225	36.3025	0.1300	False	False	False	False	False	False	False
1018	25.0	35.7950	35.4550	1.2450	False	False	True	False	False	False	False
1019	23.8	35.4725	35.5350	0.8675	False	False	False	False	False	False	False

1018 rows × 11 columns

Then we can split (80% for training and 20% for testing).



```python
# Split the encoded features and target variable into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(encoded_features, target, test_size=0.2, random_state=61)

# 80% of the data is used for training, 20% for testing
# The random_state ensures reproducibility of the split
```
✓ 0.0s

By inserting fix value for random_state, we can get same training and test data points for further continuation of the assignment.

7.

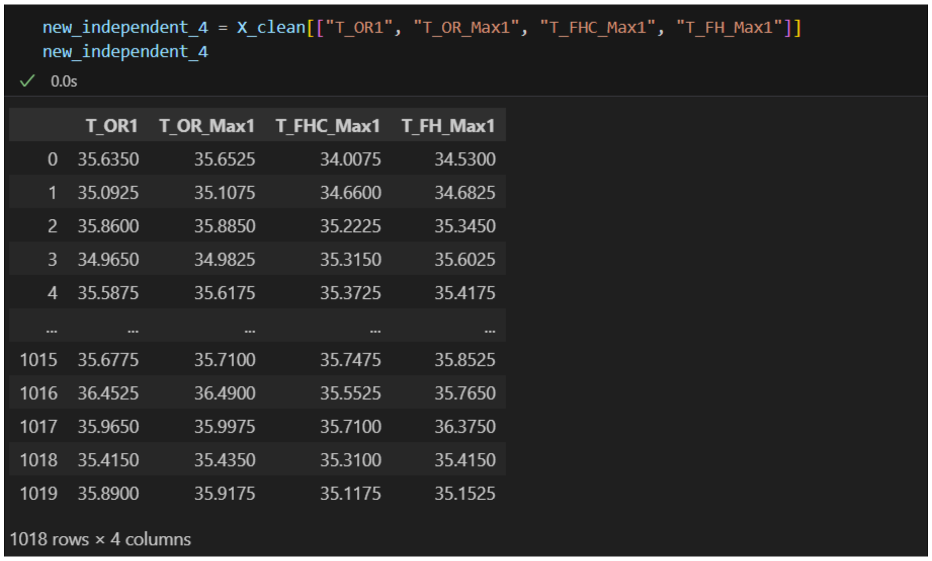
Estimated Coefficients:

Coefficient for T_atm:	-0.06584968515345042
Coefficient for T_LC_Wet1:	0.2732031569113383
Coefficient for T_LC_Dry1:	0.4936795167263089
Coefficient for T_offset1:	0.04600031294048669
Coefficient for Age_21-25:	0.037789435807562693
Coefficient for Age_21-30:	0.132256359271399
Coefficient for Age_26-30:	0.02088637616695961
Coefficient for Age_31-40:	-0.027557117880405047
Coefficient for Age_41-50:	0.17168737173379836
Coefficient for Age_51-60:	-0.03224232323633237
Coefficient for Age_>60:	0.04746740605675813

8.

T_LC_Dry1 has the highest absolute coefficient value of **0.4937**, which means it contributes the most to the dependent feature (aveOralM) in this linear regression model.

9.



```
new_independent_4 = X_clean[["T_OR1", "T_OR_Max1", "T_FHC_Max1", "T_FH_Max1"]]
new_independent_4
✓ 0.0s
```

```
      T_OR1  T_OR_Max1  T_FHC_Max1  T_FH_Max1
0     35.6350  35.6525    34.0075    34.5300
1     35.0925  35.1075    34.6600    34.6825
2     35.8600  35.8850    35.2225    35.3450
3     34.9650  34.9825    35.3150    35.6025
4     35.5875  35.6175    35.3725    35.4175
...   ...      ...        ...        ...
1015  35.6775  35.7100    35.7475    35.8525
1016  36.4525  36.4900    35.5525    35.7650
1017  35.9650  35.9975    35.7100    36.3750
1018  35.4150  35.4350    35.3100    35.4150
1019  35.8900  35.9175    35.1175    35.1525

1018 rows × 4 columns
```

Estimated Coefficients:

Coefficient for T_OR1: -0.3303900634585482

Coefficient for T_OR_Max1: 0.8643024976767969

Coefficient for T_FHC_Max1: -0.05238889519219426

Coefficient for T_FH_Max1: 0.34800950045379897

10.

Residual Sum of Squares (RSS): 19.95396543013286

Residual Standard Error (RSE): 0.1570508701338702

Mean Squared Error (MSE): 0.024513471044389265

R-Squared: 0.905067

Feature	**Coefficient**	**Standard Error**	**t-Value**	**p-Value**
Const	7.8723	1.617	4.868	0.000
T_OR1	3.0295	1.935	1.566	0.119
T_OR_Max1	-2.4463	1.934	-1.265	0.207
T_FHC_Max1	-0.0847	0.091	-0.934	0.352
T_FH_Max1	0.3201	0.104	3.088	0.002

11.

A typical threshold for significance level is 5% (0.05). If a p-value is below this threshold, the feature is considered statistically significant and likely contributes meaningfully to the model. Based on the p-values we got, we can discard T_OR1, T_OR_Max1 , T_FHC_Max1 features as they are not statistically significant and might not be contributing much to the model. T_FH_Max1 should be kept as it has a p-value less than the significance level.

4.Performance Evaluation of Linear Regression

2.

RSE for model A

d_A = 2 features

N_A= 10000

SSE_A= 9

So,

RSE= sqrt(SSE/(N_A- d_A-1))= 0.03

RSE for model B

d_B = 4 features

N_B= 10000

SSE_B= 2

So,

RSE= sqrt(SSE/(N_B- d_B-1))= 0.0141

Model with lowest RSE performs better. So, according to RSE model **B** performs better.

3.

R^2_A = 1- (SSE_A/TSS_A) = 1-(9/90) = 0.9

R^2_B = 1- (SSE_B/TSS_B) = 1-(2/10) = 0.8

Model with highest R^2 value performs better. So, model **A** performs better in this scenario.

4.

R-squared (R^2) is a fair metric for comparing models because it standardizes model performance on a consistent scale (0 to 1), reflecting the proportion of variance explained by the model. This makes it easier to compare models with different numbers of features and ensures fair evaluation across different datasets.

5.Linear Regression impact on outliers.

2.

As **a** approaches zero, the behavior of both loss functions L1(w) and L2(w) changes significantly. For L1(w), the loss value converges to 1 for all data points, regardless of the residuals r_i. This implies that the function becomes entirely insensitive to the size of the residuals, losing its capacity to distinguish between small and large errors. Similarly, L2(w) also approaches a value of 1 across all data points as **a** decrease. Consequently, like L1(w), L2(w) also loses its sensitivity to the residuals.

In essence, as **a** tends to zero, both L1(w) and L2(w) become ineffective at discriminating between different levels of residuals, making them unsuitable for regression analysis where the differentiation of residuals is crucial for model accuracy.

3.

Both functions are effective at minimizing the impact of residuals when $|r_i| \geq 40$. Comparing the two, L2(w) could be more effective due to the exponential scaling, which aggressively reduces the influence of outliers. This is evident in Figure 2, where L2(w) shows a lower loss value for larger residuals compared to L1(w). Considering the parameter a, let's assume $| r_i | = 40$ and we aim for the loss to be capped at 1. This ensures that any residuals greater than 40 won't increase the loss beyond 1. Choosing a low **a** value, ideally between 5 and 15, satisfies this condition as it makes the expression $(1 - \exp(-2 | r_i | / a)) \approx 1$.Therefore, a range of $5 \leq a \leq 15$ is appropriate.

Both functions are effective in reducing the impact of residuals when $|r_i| \geq 40$. When comparing the two, $L_2(w)$ may be more advantageous due to its exponential scaling, which more aggressively diminishes the influence of outliers. This is illustrated in Figure 2, where $L_2(w)$ demonstrates a lower loss value for larger residuals compared to $L_1(w)$. For the parameter **a**, if we set $|r_i| = 40$ and intend to limit the loss to 1, it ensures that any residuals exceeding 40 will not cause the loss to go beyond 1. Selecting a small **a** value, preferably between 5 and 15, meets this requirement since it makes the expression $(1 - \exp(-2 | r_i | / a)) \approx 1$. Therefore, an appropriate range for **a** would be $5 \leq a \leq 15$