EN3150 Assignment 01

210292G - Kirushanth G

Submitted as a requirement for an assignment in the course module EN3150 at the Electronic and Telecommunication Department, University of Moratuwa Colombo, Sri Lanka

3 September 2024

Data pre-processing

1. method preserves the structure/properties of the features

    - **Feature 1:** Max-Abs Scaling will scale the entire feature while preserving the relative magnitude of the values. Since most values are close to 0, they will remain small, but outliers will be capped at 1 or -1, preserving the sparsity of the data.
    - **Feature 2:** This feature has a wider spread of values. Standard scaling would effectively normalize this distribution, ensuring that the data is centered and spread in a consistent way.
      Also Max-Abs Scaling will scale the feature such that the maximum value will become 1 or -1, preserving the relative distances between values.
    - **Conclusion:** Max-abs scaling is a suitable choice for both features, as it maintains relative differences and is less affected by outliers, especially in Feature 1. While Standard scaling could also be effective, particularly for Feature 2, max-abs scaling offers a more robust approach overall.

Learning from data

1. Data was generated
2. The code uses the `train_test_split(X, Y, test_size=0.2, random_state=r)` function to split the data into training and testing sets.

    This function randomly selects 80% of the data for training and 20% for testing.

    The randomness is controlled by the `random_state` parameter, which is set to a random integer (`r=np.random.randint(104)`).
    Since the `random_state` is generated randomly every time the code runs (where `r` takes a new random integer between 0 and 103), the `train_test_split` function will produce different splits of the data for training and testing in each run.
3. Since the training data differs in each iteration, the best fit line (i.e., the regression model) will also differ.
4. The code was adjusted to generate 10,000 samples instead of 100, and the linear regression model was trained as before. Although the linear regression model still varied with each run, the differences between each model were significantly smaller than before. This is because with a larger training set, each training split is more representative of the entire dataset, leading to more consistent results when using 10,000 samples compared to just 100.

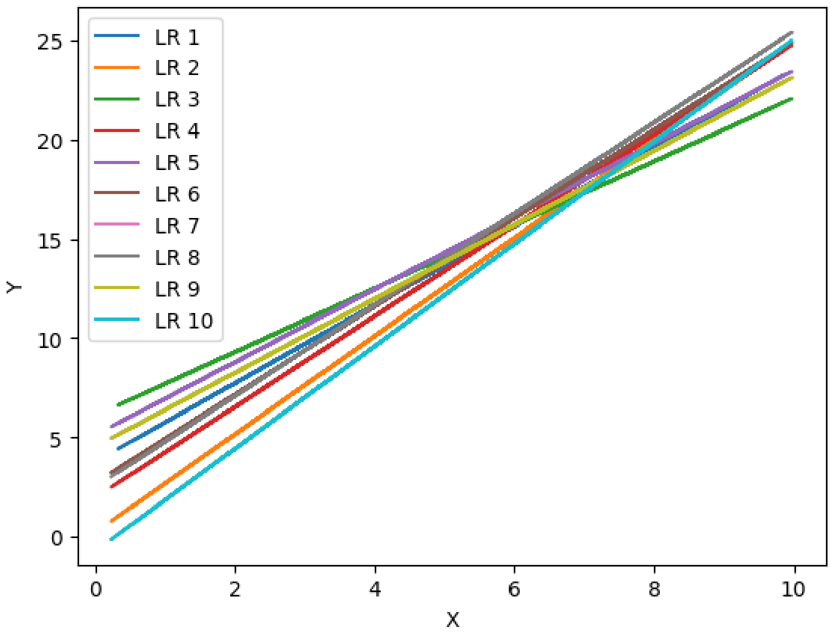

Figure 1: Fitted Lines for 100 Samples

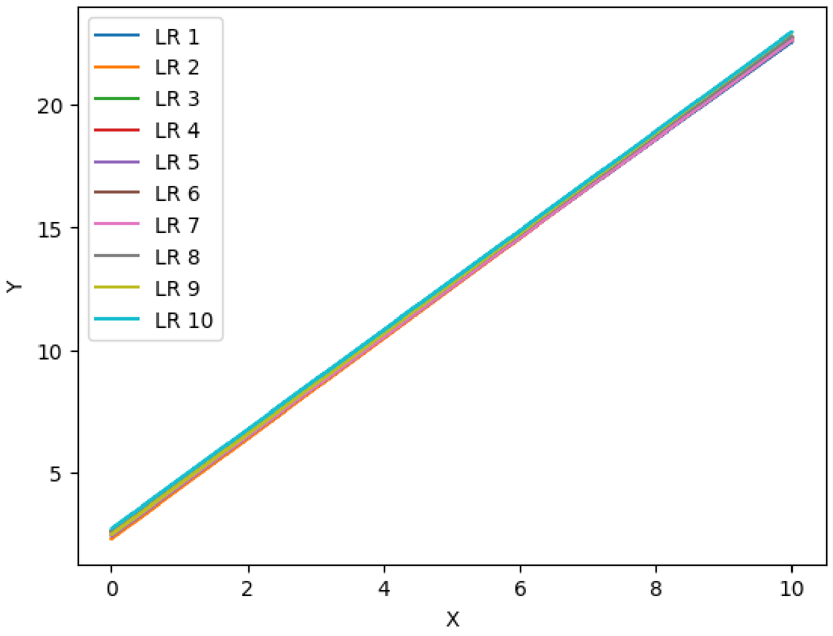

Figure 2: Fitted Lines for 10000 Samples

Linear regression on real world data

1. Data was loaded
2. Independent variable : 33
   Dependent variable : 2
3. It is not possible to apply linear regression directly to the dataset in its current form because not all data values are numerical. The dataset contains categorical variables, such as gender and age (which is provided in ranges). These categorical variables need to be

transformed before applying linear regression, either by converting them into numerical values using techniques like one-hot encoding or by averaging the age ranges. Alternatively, these categorical variables could be excluded from the regression model.

4. No, the given approach is not appropriate. Since the X and y values are dropped independently, the corresponding y values might remain even if the X values are deleted, and vice versa. Therefore, the correct approach is to concatenate the data, drop the missing values, and then separate the X and y values again.

```
import pandas as pd
# Combine X and y into a single DataFrame
data_combined = pd.concat([X, y], axis=1)

data_combined = data_combined.dropna()

# Separate X and y after dropping missing values
X = data_combined.iloc[:, :-2]  # All columns except the last two
y = data_combined.iloc[:, -2:]  # The last two columns
```

5. Features selected : 'Age', 'Distance', 'Humidity', 'T_atm', 'T_offset1'

6. Data was splitted

```
X_train, X_test, y_train, y_test = train_test_split(X_final,
    y_selected, test_size=0.2, random_state=0)
```

7. The "Age" feature is one-hot encoded to create binary columns for each category, and then concatenated with other features.

```
cat_encoder = OneHotEncoder(drop='first', sparse_output=False)
X_encoded_age = cat_encoder.fit_transform(X_selected[['Age']])
X_remaining_features = X_selected.drop(columns=['Age'])

# Combine the one-hot encoded "Age" with the other features
X_final = pd.concat([pd.DataFrame(X_encoded_age, columns=
    cat_encoder.get_feature_names_out(['Age'])),
    X_remaining_features.reset_index(drop=True)], axis=1)

# Split the data into training and testing sets (80% train, 20%
    test)
X_train, X_test, y_train, y_test = train_test_split(X_final,
    y_selected, test_size=0.2, random_state=0)

# Train a linear regression model
model = LinearRegression()
model.fit(X_train, y_train)
```

Coefficient for Age_21-25: 0.0213
Coefficient for Age_21-30: 0.1344
Coefficient for Age_26-30: -0.1200
Coefficient for Age_31-40: -0.0316
Coefficient for Age_41-50: -0.1375
Coefficient for Age_51-60: -0.3805
Coefficient for Age_>60: 0.0132
Coefficient for Distance: -0.3523
Coefficient for Humidity: 0.0013
Coefficient for T_atm: -0.0016
Coefficient for T_offset1: 0.1939
Intercept: 37.0677

8. According to these selected data, it is Age_51-60.This is because it has the largest weight in the coefficient matrix

9. Coefficient for T_OR1: 0.5034
   Coefficient for T_OR_Max1: 0.0217
   Coefficient for T_FHC_Max1: -0.0602
   Coefficient for T_FH_Max1: 0.3594
   Intercept: 7.6115

10. RSS: 75.0953
    RSE: 0.3047
    MSE: 0.0923
    R^2: 0.6429

    Standard Errors:
    const 0.793406
    T_OR1 0.859272
    T_OR_Max1 0.857762
    T_FHC_Max1 0.043671
    T_FH_Max1 0.048925

    t-statistics:
    const 9.593404
    T_OR1 0.585833
    T_OR_Max1 0.025287
    T_FHC_Max1 -1.379061
    T_FH_Max1 7.345249

    p-values:
    const 1.025961e-20
    T_OR1 5.581512e-01
    T_OR_Max1 9.798324e-01
    T_FHC_Max1 1.682570e-01
    T_FH_Max1 5.018475e-13

11. p-value less than 0.05 typically indicates that the corresponding feature has a statistically significant relationship with the dependent variable. 'T_OR1' , 'T_OR_Max1' can be discarded since they have p-values higher than 0.05

Performance evaluation of Linear regression

2.

$$\text{RSE} = \sqrt{\frac{\text{SSE}}{N-d-1}}$$

For Model A:

$$d = 2 \text{ (2 independent features)}$$
$$\text{RSE}_A = \sqrt{\frac{9}{10000-3}} \approx 0.0300$$

For Model B:

$$d = 4 \text{ (4 independent features)}$$
$$\text{RSE}_B = \sqrt{\frac{2}{10000-5}} \approx 0.0141$$

Based on RSE, Model B performs better as it has a lower RSE.

3.

$$R^2 = 1 - \frac{\text{SSE}}{\text{TSS}}$$

For Model A:

$$R_A^2 = 1 - \frac{9}{90} = 0.9000$$

For Model B:

$$R_B^2 = 1 - \frac{2}{10} = 0.8000$$

Based on R^2, Model A performs better as it has a higher R^2 value.

4. Between RSE and R-squared, R-squared is generally considered more fair for comparing models, especially when they have different numbers of predictors.R-squared is scale-independent, making it easier to compare across different datasets.

Linearregression impact on outliers

1. When a approaches 0,

   For $L_1(\mathbf{w})$:

$$L_1(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{r_i^2}{a^2+r_i^2}\right)$$

When $a \to 0$, the term a^2 in the denominator becomes negligible compared to r_i^2. So, the loss function simplifies to:

$$L_1(\mathbf{w}) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\frac{r_i^2}{r_i^2}\right) = \frac{1}{N}\sum_{i=1}^{N} 1 = 1$$

Therefore, as $a \to 0$, $L_1(\mathbf{w})$ approaches a constant value of 1, making the loss function insensitive to the actual values of the residuals r_i.

For $L_2(\mathbf{w})$:

$$L_2(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}\left(1 - \exp\left(-\frac{2|r_i|}{a}\right)\right)$$

As $a \to 0$, the exponential term $\exp\left(-\frac{2|r_i|}{a}\right)$ approaches 0 (since $\frac{2|r_i|}{a}$ becomes very large). Therefore, $L_2(\mathbf{w})$ simplifies to:

$$L_2(\mathbf{w}) \approx \frac{1}{N}\sum_{i=1}^{N}(1-0) = \frac{1}{N}\sum_{i=1}^{N}1 = 1$$

Similar to $L_1(\mathbf{w})$, $L_2(\mathbf{w})$ also approaches a constant value of 1 as $a \to 0$, making it insensitive to the residuals.

2. The loss function $L_2(\mathbf{w})$ is particularly effective at reducing the influence of outliers due to its exponential scaling. This characteristic allows $L_2(\mathbf{w})$ to aggressively suppress the impact of large residuals, making it a better choice when the goal is to minimize the contribution of such outliers.

   When selecting the hyperparameter a, the objective is to ensure that the loss function's value is clamped at or near 1 when $|r_i| \geq 40$. This clamping prevents residuals larger than 40 from disproportionately affecting the overall loss. To achieve this, a must be chosen so that the expression $1 - \exp\left(-\frac{2|r_i|}{a}\right)$ is approximately 1 when $|r_i| \geq 40$. A smaller value of $a < 18$ will suffice, as it allows the exponential term to decay rapidly, effectively clamping the loss near 1 (approximately 0.99) for these larger residuals.