EN3150 -Pattern Recognition

Assignment 01

Learning from data and related challenges and linear models for regression

W.C.A Wickramaarachchi

210699K

10/09/2024

1) Data pre-processing

Max-Abs Scaling is the most suitable scaling method for Feature 1, as it is less sensitive to outliers and preserves the original data structure. It is suitable for sparse datasets, high-dimensional data, and linear models. It ensures zeros remain zeros and proportionate scaling, leading to more stable and interpretable models. Max-Abs Scaling is computationally efficient, requiring only one pass through the data and minimal resources, making it a practical choice for large datasets.

Standard Scaling was chosen as the optimal scaling method for Feature 2, preserving the central distribution characteristics and performance requirements of machine learning algorithms. It preserves the original distribution shape, neutralizes outliers, and harmonizes scale across multiple features. This uniformity enhances the effectiveness of gradient-based optimization algorithms and statistical models relying on data normalization, such as PCA and linear regression. This strategic choice aligns with the goals of achieving high predictive performance and reliable model interpretations in various machine learning applications.

2) Learning from data

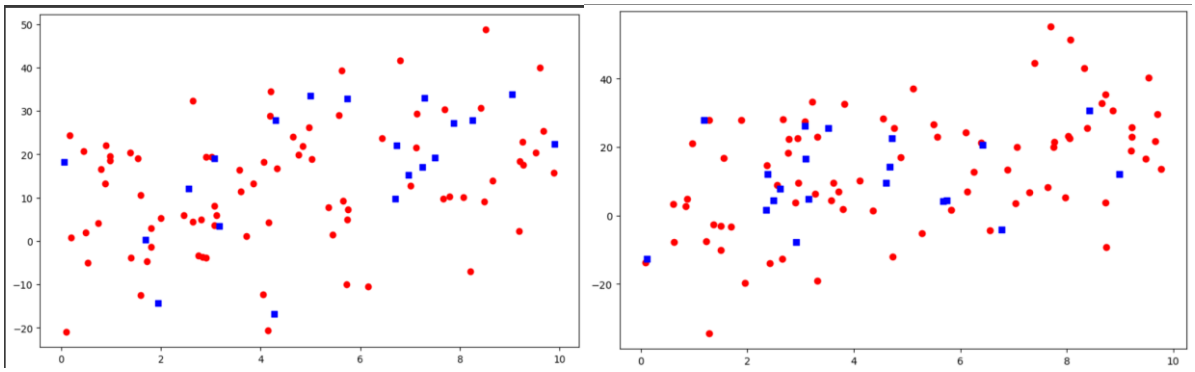
1. Use the code given in listing 1 to generate data

2. Run the code given in listing 2 multiple times and write down your observation. Why training and testing data is different in each run?

I observe that the data points in the training and testing sets are different each time. This variation occurs because the train_test_split function from scikit-learn is used to randomly divide the data into training and test sets. The code generates a random integer r using np.random.randint(104). This integer is then passed as the random_state parameter in train_test_split.

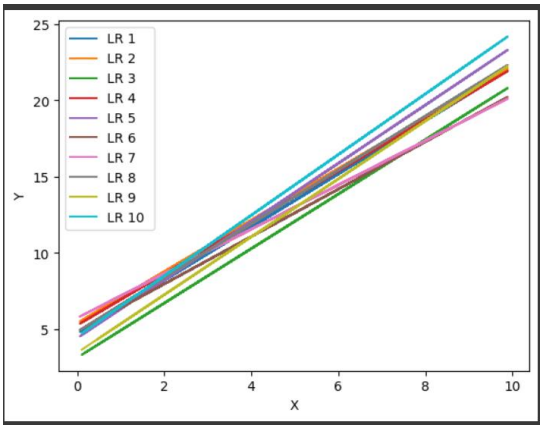
The random_state parameter controls the shuffling of the data before splitting it into training and test sets. Since r changes each time the code is run, the way the data is shuffled also changes, leading to different training and testing sets in each run.

If I use a different random seed (or random state) each time, the shuffling will result in different data points being assigned to the training and test sets. This is I observe different sets each time you run the code.



3. Use the code given in listing 3 to fit a linear regression model. Why linear regression model is different from one instance to other instance ?

In each iteration of the for loop, the train_test_split function is called with a new random seed (random_state = np.random.randint(104)), which results in different random splits of the data into training and test sets.
Since the training data (used to fit the model) changes with each iteration, the input data for the linear regression model is different every time.
The LinearRegression model is then fitted (model.fit(X_train, Y_train)) using the current split of the training data.
Because the training data differs in each iteration, the model's are also different.
After fitting, the model predicts the outputs (Y_pred_train) for the training data.
Since the model's parameters differ, the predicted values (Y_pred_train) also vary between iterations.

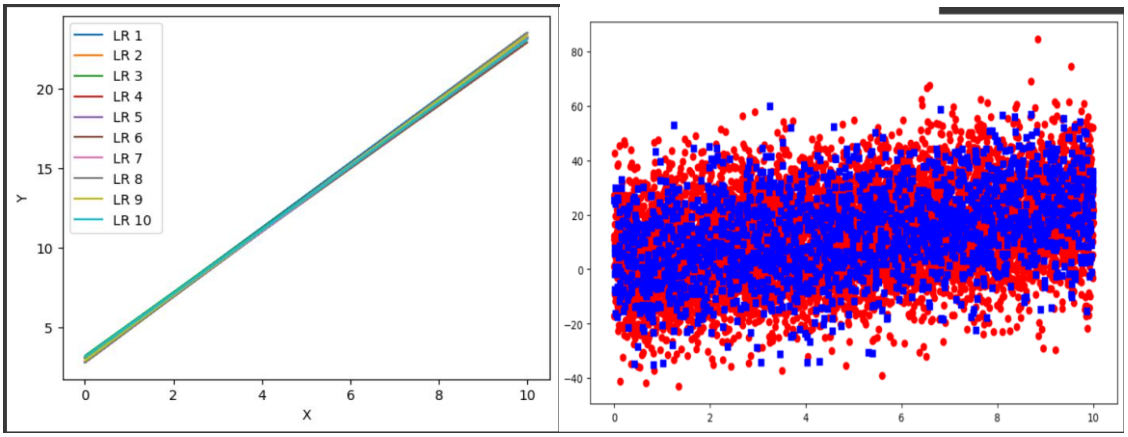


4. Increase the number of data samples to 10,000 (n_samples = 10000 in listing 1) and repeat the task 3. What is your observation in comparison to 100 data samples ? State a reason for the different behavior compared to 100 data samples.

The lines will appear more consistent and closer to each other.

Any single random split has less of an impact as the number of samples rises. The reason for this is the law of large numbers, which says that the sample mean will tend to converge to the true population mean as the sample size increases. In this instance, estimates of the regression model parameters are more stable and consistent as the datasets are larger. With a larger dataset, the training set is more likely to accurately represent the entire population, leading to a regression model that generalizes better. This reduces the variance in model parameters across different random splits.

```
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 = 10000
# 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 ]
```



3) Linear regression on real world data

1) Load the data

2) dependent variables-2

   independent variables-33

3) Yes, it is possible to apply linear regression on this dataset, but there are some steps we should follow to prepare the data before applying the regression model.

   This data set contains two target variables. Both of which are continuous . Linear regression can predict these target variables based on the features provided.

Handling categorical variables: 'Gender', 'age', 'Ethnicity' are categorical features. since linear regression requires numerical input, you'll need to convert these categorical variables into numerical format using methods like One-Hot Encoding , Label Encoding.

Handling Missing values: as mentioned in the website we don't have to consider on this matter.

Feature Scaling: Linear regression assumes that the input features are on a similar scale. Continuous features like T_atm, Humidity, etc., should be scaled using methods like Standardization or Normalization.

4) This is not correct, Specifically, the issue lies in the fact that dropping missing values (NaNs) in X and y separately can lead to misalignment between the features (X) and the targets (y). If rows are dropped from X or y independently, the corresponding features and targets might no longer match up correctly.

First We should combine X and y into a single Data Frame (concatenation), drop the rows with missing values, and then split them back into X and y.

```
import pandas as pd
data = pd.concat([X, y], axis=1)

data = pd.concat([X, y], axis=1)
data = data.dropna()
X = data.iloc[:, :-2]
y = data.iloc[:, -2:]
```

7) I get the age coefficients respective to the different ranges by using one hot encoding.

- Coefficient for Humidity: 0.0015317482200953463
- Coefficient for T_offset1: -0.10179089614957075
- Coefficient for Max1R13_1: -0.3101001403904479
- Coefficient for T_RC1: 1.0584237002627388
- Coefficient for Age_18-20: -0.1499026897232954
- Coefficient for Age_21-25: -0.09707468098772418
- Coefficient for Age_21-30: -0.007136324850578268
- Coefficient for Age_26-30: -0.09769721669579694
- Coefficient for Age_31-40: -0.13637111538609853
- Coefficient for Age_41-50: 0.04680746744059978
- Coefficient for Age_51-60: -0.06961041750574228
- Coefficient for Age_>60: 0.5109849777086363
- Intercept: 10.501230392232824

8) T_RC1 contributes highly for the dependent feature

9)

- Coefficient for T_OR1: 0.20545776323994563
- Coefficient for T_OR_Max1: 0.34819684316002775
- Coefficient for T_FHC_Max1: -0.08371846705362093
- Coefficient for T_FH_Max1: 0.376564342065323
- Intercept: 6.79355629984887

10)

- Residual Sum of Squares (RSS): 15.9233997543774
- Residual Standard Error (RSE): 0.28287291173396445
- Mean Squared Error (MSE): 0.07805588114890882
- R^2 statistic: 0.6076047374475753
- Standard errors for each feature:
  const 1.510164
  - T_OR1 1.558545
  - T_OR_Max1 1.559316
  - T_FHC_Max1 0.089823
  - T_FH_Max1 0.096804
- t-statistics for each feature:
  const 7.024985
  - T_OR1 0.359396
  - T_OR_Max1 -0.034685
  - T_FHC_Max1 0.594172
  - T_FH_Max1 1.873631
- p-values for each feature: const 3.322486e-11
  - T_OR1 7.196801e-01
  - T_OR_Max1 9.723656e-01
  - T_FHC_Max1 5.530717e-01
  - T_FH_Max1 6.244719e-02

11) A common threshold for p-values is 0.05. T_OR1, T_OR_Max1, and T_FHC_Max1 all have p-values well above 0.05, indicating they are not statistically significant.
T_FH_Max1 has a p-value slightly above 0.05

4) Perform evaluation of Linear regression

2)

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

$$RSE_A = \sqrt{\frac{9}{10000-2-1}} = 0.03$$

$$RSE_B = \sqrt{\frac{2}{10000-4-1}} = 0.014$$

Based on the RSE values we can say that B is the best model out of A and B .

3)

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

$$R_A^2 = 1 - \frac{9}{90} = 0.9$$

$$R_B^2 = 1 - \frac{2}{10} = 0.8$$

Based on the R^2 A is the better model. Why I say is that A higher R^2 values indicate that the model explains a large portion of the variance in the dependent variable, Suggesting a better fit.

4) To compare how well different models fit the same dataset, R-squared is the more appropriate metric because it provides a relative measure of fit that accounts for the overall variance in the dataset. R^2 allows us to see which model explains more of the variance in the dependent variable.

Additionally, R^2 adjusts for the number of predictors, making it a better choice for comparing models with different complexities. It offers a normalized and interpretable measure of how well each model explains the variation in the data, helping to identify the model that provides the best fit.

5) Linear regression impact on outliers

2) when $a \to 0,$ loss function L_1 and L_2 also change to following functions,

$$L_1(\omega) = \frac{1}{N}\sum_{i=1}^{N}(\frac{r_i^2}{a_i + r_i^2})$$

$$L_1(\omega) = \frac{1}{N}\sum_{i=1}^{N}(\frac{r_i^2}{r_i^2})$$

$$L_1(\omega) = \frac{1}{N}\sum_{i=1}^{N}(1) = 1$$

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

$$L_2(\omega) = \frac{1}{N}\sum_{i=1}^{N}(1) = 1$$

Both functions goes to one when a goes to zero, meaning the loss functions become non-informative . This would lead to the loss function can no longer distinguish between different residuals, making it ineffective for guiding the

optimization process in linear regression. Therefore , using a values close to 0 would not be practical in reducing the impact of outliers or in fitting the model efficiency.

3) When using the L1(w) function with a = 2.5 , many data points, not just outliers, may be ignored, as L1(w) approaches 1 when |r| is close to 10. On the other hand, L1(w) with a = 25 also reduces the influence of data points but does not reach 1 until |r| exceeds 40, minimizing the effect of points with residuals below this threshold.

Meanwhile, the L2(w) function with a = 2.5 demonstrates the steepest decrease for large residuals, significantly penalizing large errors and reducing their influence. Compared to these functions, L2(w) with a = 2.5 is the preferable choice, as it more effectively minimizes the impact of outliers, as illustrated in the graph.