Department of Electronic and Telecommunication Engineering University of Moratuwa

EN3150 - Pattern Recognition

Assignment 01

210668P Vidmal H.V.P.

Contents

1 Data pre-processing **2**

2 Learning from data **2**
2.1 Why training and testing data is different in each run? (Q2) 2
2.2 Why the Linear Regression Model is Different Across Instances (Q3) . 3
2.3 Observation with 10,000 Samples (Q4) 4
2.4 Reason for Different Behavior Compared to 100 Samples (Q4) 5

3 Linear regression on real world data **5**
3.1 Independent variables and Dependent variables(Q2) 5
3.2 possibility to apply linear regression on this dataset (Q3) 5
3.3 NaN missing value (Q4) 5
3.4 Selected dependent features and independent features (Q5) 6
3.5 Split the data (Q6) 6
3.6 Train a linear regression model (Q7) 6
3.7 Estimate the coefficient corresponds to indepen- dent variables . . . 7
3.8 Highly Contributing Independent Feature (Q8) 7
3.9 Train a new linear regression model and estimate the coefficient corresponds to new independent variables (Q9) 7
3.10 Statistical Measures Calculation (Q10) 8
3.10.1 Python Code for Calculation 8
3.10.2 Answers for measurements 9
3.11 Discard any features based on p-value (Q11) 10

4 Performance Evaluation of Linear Regression **10**
4.1 Calculation of Residual Standard Error (RSE) (Q2) 10
4.2 Calculation of R-squared (R^2) (Q3) 11
4.3 Interpretation (Q4) 11
4.4 Comparison of Metrics: RSE vs. R^2 (Q4) 12

5 Linear regression impact on outliers **12**
5.1 What happens when a $\rightarrow$ 0 ? (Q1) 12
5.2 What value(s) of **a** and what function(s) would you choose, and why? (Q3) 13

1 Data pre-processing

- **Feature 1:** The plot shows that most of the values in Feature 1 are centered around 0, with the few significant outliers. The feature appears sparse, and standard scaling might not be the best option because of the outliers. Min-max scaling would also be affected by the outliers, leading to most values being close to one end of the scale. Max-abs scaling would be appropriate here, as it preserves the sparse nature of the feature without heavily distorting the range due to the outliers.

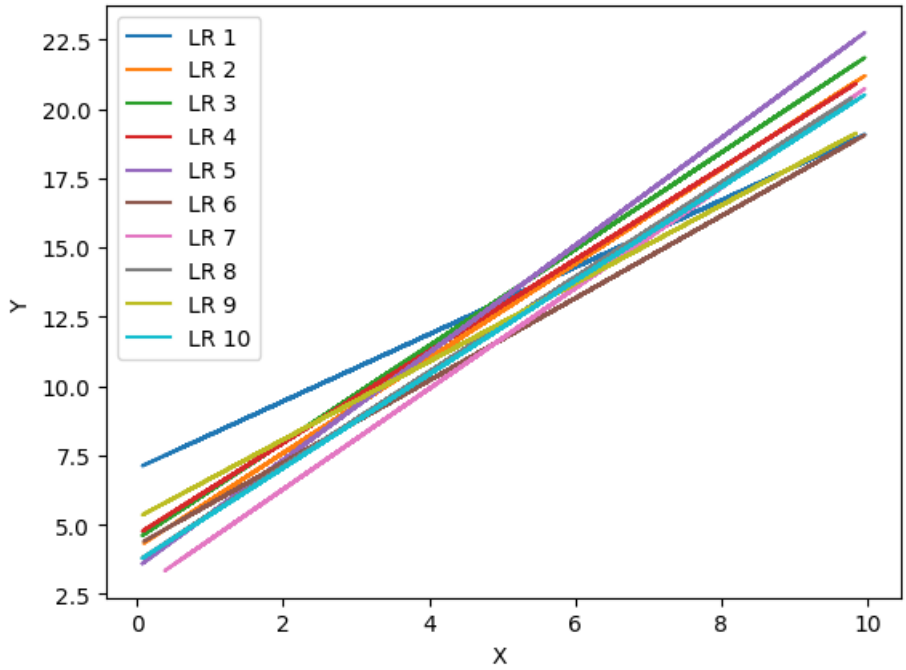
- **Feature 2:** This feature shows a wider range of values with both positive and negative variations. Since the data isn't sparse and has significant variability, standard scaling might be the most appropriate method to preserve the distribution structure and ensure that the variability around the mean is maintained.

2 Learning from data

2.1 Why training and testing data is different in each run? (Q2)

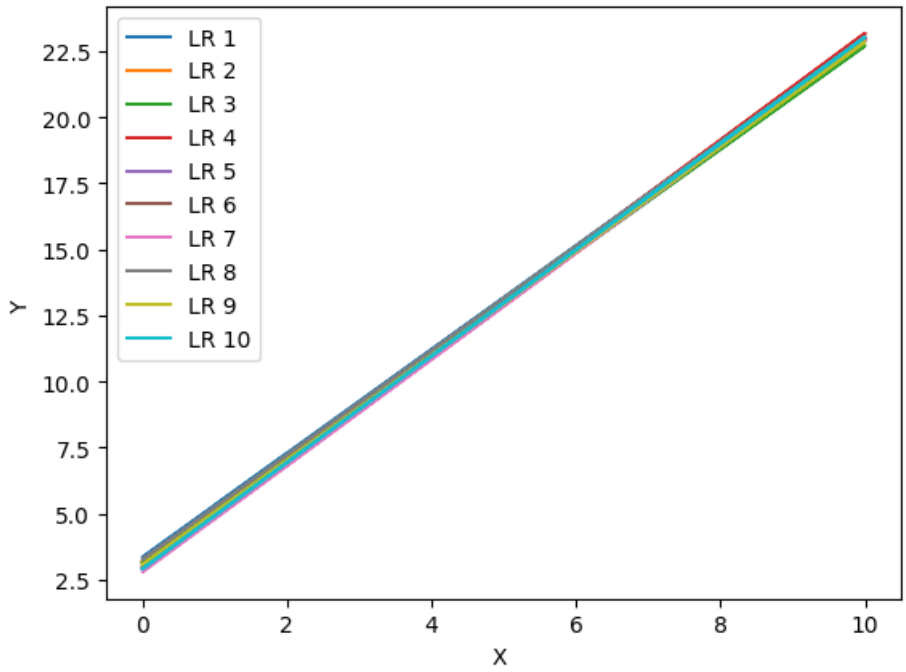
The `train_test_split` function divides the data randomly. The `random_state` parameter ensures that the random split is reproducible when given a specific seed. However, since `r` is randomly generated each time, the seed value changes with each run, leading to different splits of the dataset into training and testing sets.

2.2 Why the Linear Regression Model is Different Across Instances (Q3)



When fitting the linear regression model multiple times (10 instances), different training datasets are created each time by randomly splitting the original dataset into training and testing sets. The model differs from one instance to another because the random splitting process results in different subsets of data being used for training. This randomness in selecting the training data leads to variations in the learned model parameters (slope and intercept), causing the regression model to change with each instance.

2.3 Observation with 10,000 Samples (Q4)



When the number of data samples is increased from 100 to 10,000 and the linear regression model fitting process is repeated, the following observations can be made in comparison to using only 100 samples:

1. **Reduced Variation Across Instances:** The linear regression models across different instances (iterations) become more consistent. The fitted lines from each instance will be more similar to each other, showing less variability in predictions compared to when only 100 samples were used.

2. **More Stable Model Coefficients:** The coefficients (slope and intercept) of the linear regression model exhibit less fluctuation between different instances. The larger dataset leads to more stable and reliable estimates of these coefficients.

3. **Closer Fit to the True Relationship:** The larger sample size allows the model to capture the underlying linear relationship between X and Y more accurately. As a result, the fit of the model improves, and the predictions are generally closer to the true values.

4. **Reduced Impact of Random Data Splitting:** With 10,000 samples, the random split between training and testing sets has a smaller effect on the model's performance. The training data becomes more representative of the entire dataset, making the model less sensitive to variations caused by different random splits.

2.4 Reason for Different Behavior Compared to 100 Samples (Q4)

The different behavior when increasing the number of data samples from 100 to 10,000 can be explained by the Central Limit Theorem. This theorem states that as the sample size grows, the sampling distribution of the mean becomes closer to a normal distribution, no matter what the shape of the original data distribution is. In linear regression, this means that as we increase the number of data points, the estimates for the regression coefficients (slope and intercept) also become more normally distributed and stable.

With more data, the linear regression model has a better chance of capturing the true relationship between the variables because it has more information to learn from. This reduces the impact of noise and outliers, making the regression line fit more accurately to the overall data pattern.

3 Linear regression on real world data

3.1 Independent variables and Dependent variables(Q2)

There are 33 independent variables, including 3 categorical features and 30 continuous features.

Dependent variables:

1. aveOralF
2. aveOralM

3.2 possibility to apply linear regression on this dataset (Q3)

The dataset contains some categorical data. Since linear regression requires numerical inputs, categorical variables must be converted into numerical formats. Here are a few approaches to achieve this:

1. **One-Hot Encoding:** This method converts each category of a categorical variable into separate binary columns (with values 0 or 1)
2. **Label Encoding:** This approach assigns a unique integer to each category in the variable

3.3 NaN missing value (Q4)

No

The method described is not ideal because it may result in dropping features and dependent variables independently, leading to mismatched columns between the datasets. Instead, first concatenate the features X and the dependent variable Y. After combining them, use the `dropna()` function to handle any missing values. Then, split

the data back into X and Y as needed. This approach ensures that any rows with missing values are removed consistently across both the features and the dependent variable, maintaining alignment between them.

```
# Removing missing/NaN values
data = pd.concat([X, y], axis=1)
data = data.dropna()

X = data.drop(['aveOralF', 'aveOralM'], axis=1)
y = data[['aveOralF', 'aveOralM']]
```

3.4 Selected dependent features and independent features (Q5)

- Age
- T_atm
- Humidity
- Distance
- T_offset1

```
dependent_variable = "aveOralM"
independent_variables = ['T_atm', 'Humidity', 'Distance', 'T_offset1
    '] + [column for column in X_encoded.columns if column.startswith
    ('Age_')]

X_sel = X_encoded[independent_variables ]
y_sel = y[dependent_variable]
```

3.5 Split the data (Q6)

```
X_train, X_test, y_train, y_test = train_test_split(X_sel, y_sel,
    test_size=0.2, random_state=42)
```

3.6 Train a linear regression model (Q7)

```
model = LinearRegression()
model.fit(X_train, y_train)
```

3.7 Estimate the coefficient corresponds to indepen- dent variables

```
coefficients = model.coef_
intercept = model.intercept_

for feature, coef in zip(independent_variables, coefficients):
    print(f"{feature}: {coef}")
print(f"Intercept: {intercept}")
```

3.8 Highly Contributing Independent Feature (Q8)

- `Age_51-60`

Since the one-hot encoded feature for the `Age_51-60` category has the highest coefficient of -0.325, it significantly contributes to the `aveOralM` dependent variable among the selected features.

```
T_atm: 0.006364770313850903
Humidity: 0.0014354180191543733
Distance: 0.0025143135455547051
T_offset1: 0.18279515887175224
Age_21-25: -0.005293117954369636
Age_21-30: 0.07516273959895754
Age_26-30: -0.14829523423206117
Age_31-40: -0.026279909784432964
Age_41-50: -0.09644371345033839
Age_51-60: -0.32560687421308787
Age_>60: -0.20310069308964013
Intercept: 36.666963353262204
```

3.9 Train a new linear regression model and estimate the coefficient corresponds to new independent variables (Q9)

```
new_independent_variables = ['T_OR1', 'T_OR_Max1', 'T_FHC_Max1', '
    T_FH_Max1']

X_sel_new = X[new_independent_variables]
```

```
y_sel_new = y[dependent_variable]

X_train_new, X_test_new, y_train_new, y_test_new = train_test_split(
    X_sel_new, y_sel_new, test_size=0.2, random_state=42)

#linear regression model
model_new = LinearRegression()
model_new.fit(X_train_new, y_train_new)

#get the coeffeints relevant to the independent variables
coefficients = model_new.coef_
intercept = model_new.intercept_

for feature, coef in zip(independent_variables, coefficients):
    print(f"{feature}: {coef}")
print(f"Intercept: {intercept}")
```

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

3.10 Statistical Measures Calculation (Q10)

3.10.1 Python Code for Calculation

```
#predict new model
yhat=model_new.predict(X_train_new)

# Residual Sum of Sqaures
RSS = np.sum((yhat - y_train)**2)
print('RSS =', RSS)


N=len(y_train)
print('Total Number of Datapoints=',N)

# Residual Standard Error
RSE = np.sqrt(1/(N-4-1)*RSS)
print('RSE =', RSE)

# Total Sum of Squares (TSS)
TSS = np.sum((y_train_new- np.mean(y_train_new))**2)
print('TSS =', TSS)

```

```
# Mean Squared Error (MSE)
MSE = RSS/N
print('MSE =', MSE)

#R2 statistic
R2 = (TSS - RSS)/TSS
print('R2 =', R2)

# Add a constant term to the independent variables for statsmodels
X_train_sm = sm.add_constant(X_train_new)

# Fit the model using statsmodels
model_sm = sm.OLS(y_train_new, X_train_sm).fit()

#Get the standard errors for each feature
std_errors = model_sm.bse
print("Standard error for each feature:")
for i, error in enumerate(std_errors):
    print(f"  {X_train_sm.columns[i]}: {error}")

# Get the t-statistic for each feature
t_stats = model_sm.tvalues
print("t-statistic for each feature:")
for i, stat in enumerate(t_stats):
    print(f"  {X_train_sm.columns[i]}: {stat}")

# Get the p-value for each feature
p_values = model_sm.pvalues
print("p-value for each feature:")
for i, value in enumerate(p_values):
    print(f"  {X_train_sm.columns[i]}:  {value}")
```

3.10.2 Answers for measurements

1. Residual sum of squares= 77.97449082857881
2. Residual Standard Error = 0.31045739777400483
3. Total Sum of Squares = 223.63911855036855
4. Mean Squared Error = 0.09579175777466684
5. R2 statistic = 0.6513378726673116
    - const: 0.8005727781362718
    - T_OR1: 0.8925861814731239
    - T_OR_Max1: 0.8903729622333665
    - T_FHC_Max1: 0.04357377979615257
    - T_FH_Max1: 0.04855049451596901

6. **t-statistic for each feature**
    - const: 8.485869724004566
    - T_OR1: 0.23018254988152623
    - T_OR_Max1: 0.39106852737984066
    - T_FHC_Max1: -1.921303762154013
    - T_FH_Max1: 7.756138136584114
7. **p-value for each feature**
    - const: 1.0144610581359123e-16
    - T_OR1: 0.8180081112902403
    - T_OR_Max1: 0.6958495517085277
    - T_FHC_Max1: 0.05504450887255178
    - T_FH_Max1: 2.6339932537048642e-14

3.11 Discard any features based on p-value (Q11)

We can reject the features that have p-values grater the 5% based on p-values. Therefore, we can discard the features T_FHC_Max1 and T_OR1.

4 Performance Evaluation of Linear Regression

4.1 Calculation of Residual Standard Error (RSE) (Q2)

The formula for Residual Standard Error (RSE) is:

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

where:

- SSE :Sum of Squared Errors.
- N : Number of data samples.
- d : Number of independent features.

For Model A:

- SSE = 9
- $N = 10000$
- $d = 2$ (w_1, w_2)

For Model B:

- SSE = 2
- $N = 10000$
- $d = 4$ $(w_1, w_2, w_3, w_4,)$

Calculations:

Model A:

$$\text{RSE}_A = \sqrt{\frac{9}{10000 - 2 - 1}} = \sqrt{\frac{9}{9997}} \approx 0.0300$$

Model B:

$$\text{RSE}_B = \sqrt{\frac{2}{10000 - 4 - 1}} = \sqrt{\frac{2}{9995}} \approx 0.0141$$

4.2 Calculation of R-squared (R^2) (Q3)

The formula for R^2 is:

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

where:

- SSE : Sum of Squared Errors.
- TSS : Total Sum of Squares.

Model A:

$$R_A^2 = 1 - \frac{9}{90} = 1 - 0.1 = 0.9$$

Model B:

$$R_B^2 = 1 - \frac{2}{10} = 1 - 0.2 = 0.8$$

4.3 Interpretation (Q4)

RSE:

- In terms of residual standard error, Model B performs better than Model A, as seen by the lower RSE (0.0141) of Model B over Model A (0.0300).

R^2:

- Compared to Model B, which has an R^2 of 0.8, Model A has a higher R^2 of 0.9, indicating that it explains more variation in the data.

4.4 Comparison of Metrics: RSE vs. R^2 (Q4)

Between RSE and R^2:

- **RSE** considers the number of data points and the number of predictors in the model, providing a measure of the model's accuracy relative to its complexity.
- R^2 measures the proportion of variance in the dependent variable that is predictable from the independent variables but does not penalize models for having more predictors, which can lead to overfitting.

In this case, **RSE** is a more fair metric for comparing the models because it considers model complexity by penalizing for the number of predictors. Model B, with its lower RSE, demonstrates better performance despite having more predictors. This lower RSE indicates that Model B strikes a better balance between fit and complexity, which should be given more weight than its lower R^2 compared to Model A.

5 Linear regression impact on outliers

5.1 What happens when a $\rightarrow$ 0 ? (Q1)

For $L_1(w)$:

The loss function is given by:

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

As $a \rightarrow 0$, the term a^2 in the denominator becomes negligible compared to r_i^2, unless r_i is very small. Therefore, for most residuals r_i:

$$\frac{r_i^2}{a^2 + r_i^2} \approx \frac{r_i^2}{r_i^2} = 1$$

Hence, $L_1(w)$ approximates:

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

So, as $a \rightarrow 0$, $L_1(w)$ approaches 1 for every data point, implying that all residuals contribute equally to the loss, regardless of their magnitude.

For $L_2(w)$:

The loss function is given by:

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

As $a \to 0$,

$$\exp\left(-\frac{2|r_i|}{a}\right) \approx 0$$

Thus, the loss function approximates:

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

So, as $a \to 0$, $L_2(w)$ also approaches 1 for every data point, implying that all residuals contribute equally to the loss, regardless of their magnitude.

5.2 What value(s) of a and what function(s) would you choose, and why? (Q3)

To minimize the influence of outliers where $|r_i| \geq 40$, selecting an appropriate value of a that decreases the effect of large residuals is crucial. Referring to the graph:

- For $a = 2.5$, both $L_1(w)$ and $L_2(w)$ display high function values at larger $|r_i|$, which demonstrates significant sensitivity to outliers.
- When $a = 25$, the sensitivity to large residuals decreases in both functions, but $L_2(w)$ shows a smoother, more gradual increase, suggesting that it could be better for managing larger residuals.
- As $a = 100$, both functions exhibit lower sensitivity to outliers, though $L_1(w)$ becomes more insensitive to large $|r_i|$.

.