K.N. Toosi University of Technology

Students: Mostafa Latifian – 40122193

Parsa Alaviniko – 40120993

Professor: Dr. Mahdi Aliyari-Shoorehdeli

Course: Fundamental of Intelligent Systems

MP – 1

Question 4

Question 5

Google Drive Folder

Table of Content

Question 1

Part A: Formulas for calculating Sensitivity and Specificity

In a multi-class classification problem, Sensitivity and Specificity are calculated in a One-vs-Rest manner for each class.

- True Positives (TP): The number of samples of the target class that are correctly predicted as that class.
- False Negatives (FN): The number of samples of the target class that are incorrectly predicted as other classes.
- False Positives (FP): The number of samples from other classes that are incorrectly predicted as the target class.
- True Negatives (TN): The number of samples from other classes that are correctly predicted as other classes.

$$Sensitivity = \frac{TP}{TP + FN} \qquad Specificity = \frac{TN}{TN + FP}$$

Actual / Predicted	C'_1	C'_2	C'_3	C'_4	**Actual Total**
C_1	45	3	2	1	51
C_2	3	32	2	6	43
C_3	2	2	16	10	30
C_4	0	2	0	20	22
Predicted Total	50	39	20	37	146

Part B: Calculation of Sensitivity and Specificity for Each Class

1. Calculation for C_1

Metric	**Value**	**Computation**
TP_1	45	Cell (C_1, C_1')
FN_1	3 + 2 + 1 = 6	Sum of the remaining cells in row C_1
FP_1	3 + 2 + 0 = 5	Sum of the remaining cells in column C_1'
TN_1	32 + 2 + 6 + 2 + 16 + 10 + 2 + 0 + 20 = 90	Sum of all cells outside row C_1 and column C_1'
$Sensitivity_1$	45 / 51 = 0.882	45 / (45 + 6)
$Specificity_1$	90 / 95 = 0.947	90 / (90 + 5) = TN_1 + FP_1: total non-C_1 samples

2. Calculation for C_2

Metric	Value	Computation
TP_2	32	Cell (C_2, C_2')
FN_2	3 + 2 + 6 = 11	Sum of the other cells in row C_2
FP_2	3 + 2 + 2 = 7	Sum of the other cells in column C_2'
TN_2	45 + 2 + 1 + 2 + 16 + 10 + 0 + 0 + 20 = 96	Sum of all cells outside row C_2 and column C_2'
$Sensitivity_2$	32 / 43 = 0.744	32 / (32 + 11)
$Specificity_2$	96 / 103 = 0.932	96 / (96 + 7)

3. Calculation for C_3

Metric	Value	Computation
TP_3	16	Cell (C_3, C_3')
FN_3	2 + 2 + 10 = 14	Sum of the other cells in row C_3
FP_3	2 + 2 + 0 = 4	Sum of the other cells in column C_3'
TN_3	45 + 3 + 1 + 3 + 32 + 6 + 0 + 2+20 = 112	Sum of all cells outside row C_3 and column C_3'
$Sensitivity_3$	16 / 30 = 0.533	16 / (16 + 14)
$Specificity_3$	112 / 116 = 0.965	112 / (112 + 4)

4. Calculation for C_4

Metric	Value	Computation
TP_4	20	Cell (C_4, C_4')
FN_4	0 + 2 + 0 = 2	Sum of the other cells in row C_4
FP_4	1 + 6 + 10 = 17	Sum of the other cells in column C_4'
TN_4	45 + 3 + 2 + 3 + 32 + 2 + 2 + 2 + 16 = 107	Sum of all cells outside row C_4 and column C_4'
$Sensitivity_4$	20 / 22 = 0.909	20 / (20 + 2)
$Specificity_4$	107 / 124 = 0.863	107 / (107 + 17)

Question 2

Part A: Perceptron Learning Algorithm

Let's assume the initial weight vector w is equal to:

$$w(0) = [w_1 \ \ w_2 \ b]^T = [0\ 0\ 0]^T$$

The output $\hat{y}$ is equal to:

$$z = [x_1\ x_2\ 1], \qquad \hat{y} = \begin{cases} 1 & w^T z > 0 \\ 0 & w^T z \le 0 \end{cases}$$

So, by substituting any desired z into the relation above, $\hat{y}$ will become 0. Therefore, to correct w, we use the following relation:

$$e = y - \hat{y}, \qquad w(k+1) = w(k) + ez\eta$$

where η is the learning rate and we consider it to be 1.

By substituting the point $x = (x_1, x_2) = (1,1)$ into the above relations, we have:

$$(x_1, x_2, y) = (1,1,1) \Rightarrow \hat{y} = 0 \Rightarrow e = 1 - 0 = 1 \Rightarrow w(1) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Now we substitute the rest of the points in the same manner, and if there is an error, we correct w:

$$(x_1, x_2, y) = (1,1,1) \Rightarrow \hat{y} = 1 \Rightarrow e = 1 - 1 = 0$$

$$(x_1, x_2, y) = (0,2,1) \Rightarrow \hat{y} = 1 \Rightarrow e = 1 - 1 = 0$$

$$(x_1, x_2, y) = (3,0,1) \Rightarrow \hat{y} = 1 \Rightarrow e = 1 - 1 = 0$$

$$(x_1, x_2, y) = (0,2,1) \Rightarrow \hat{y} = 1 \Rightarrow e = 1 - 1 = 0$$

$$(x_1, x_2, y) = (-2,-1,0) \Rightarrow \hat{y} = 0 \Rightarrow e = 0 - 0 = 0$$

$$(x_1, x_2, y) = (0,0,-2) \Rightarrow \hat{y} = 0 \Rightarrow e = 0 - 0 = 0$$

Since in the above relations, we have no error for all training samples, there is no need to change w. Thus, the equation of the separating line will be as follows:

$$w^T x = [1\ 1\ 1] \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} = x_1 + x_2 + 1$$

The curve of the resulting line is visible in figure 2.1.1.

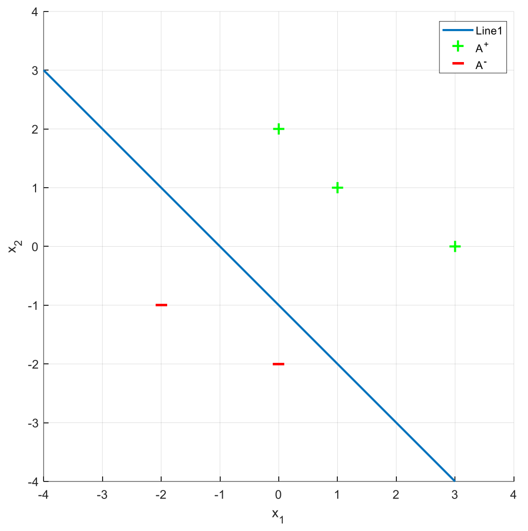


Fig. 2.1.1. Resulting Line curve of PLA

Part B: Minimum Squared Error (MSE) Method

To find w with the MSE method, we use the following formula:

$$X = \begin{bmatrix} x^1 & 1 \\ x^2 & 1 \\ \vdots & \vdots \\ x^N & 1 \end{bmatrix}, w = (x^T x)^{-1} x^T y$$

By substitution, we have:

$$X = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 2 & 1 \\ 3 & 0 & 1 \\ -2 & -1 & 1 \\ 0 & -2 & 1 \end{bmatrix}, y = \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \\ -1 \end{bmatrix}$$

$$w = (x^T x)^{-1} x^T y = \begin{bmatrix} 0.3089 \\ 0.5073 \\ 0.0764 \end{bmatrix}$$

$$\Rightarrow w^T x = [0.3089\ 0.5073\ 0.0764] \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} = 0.3089 x_1 + 0.5073 x_2 + 0.0764$$

The curve of the resulting line is visible in figure 2.2.1.

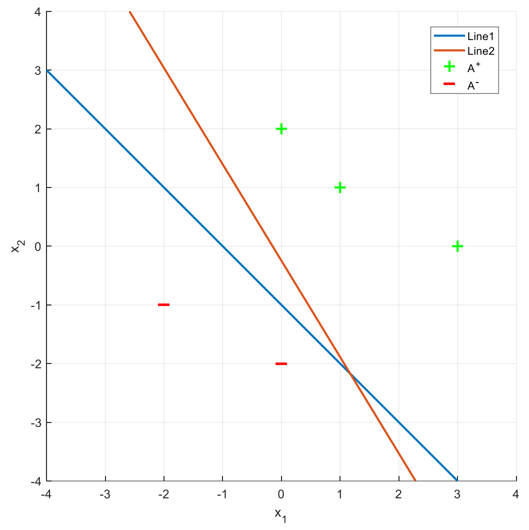


Fig. 2.2.1. Resulting Line curve of MSE

Part C: Fisher's Linear Discriminant (FLD) Method

According to the Fisher method, to find w we have:

$$S_w = \sum_{i\in A^+}(x_i - m_1)\,(x_i - m_1)^T + \sum_{i\in A^-}(x_i - m_2)(x_i - m_2)^T$$

$$w' = S_w^{-1}(m_1 - m_2), b = -\frac{1}{2}w'^T(m_1 + m_2), w = [w'b]^T$$

where m_i is the mean corresponding to the ith class. Now by substituting the data given in the problem statement, we obtain w:

$$m_1 = \begin{bmatrix}\frac{4}{3} & 1\end{bmatrix}^T, m_2 = \begin{bmatrix}-1, -\frac{3}{2}\end{bmatrix}^T, S_w = \begin{bmatrix}6.6667 & -4\\ -4 & 2.5\end{bmatrix}$$

$$w' = [23.75\ 39]^T, b = 5.7917 \Rightarrow w = [23.75\ 39\ 5.7917]^T$$

The curve of the resulting line is visible in figure 2.3.1.

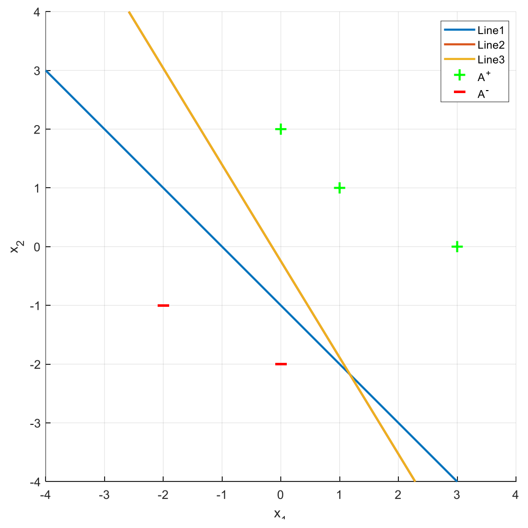


Fig. 2.3.1. Resulting Line curve of FLD

As observed, Line 2 and Line 3 are congruent (overlap), and all 3 lines have successfully achieved complete separation of the classes. However, Line 1 is closer to the negative class and has a smaller margin compared to Line 2 and 3. Therefore, the best line is Line 2 and 3.

Question 3

If we apply PCA without preprocessing (without centering and without scaling) on data with very different ranges, the first principal component will capture almost all the variance from feature x_1, and the first eigenvector will be approximately aligned with the x_1 axis. Thus, PCA effectively hides the true relationships between the variables, and priority is given solely to the feature with the larger numerical value, not necessarily to the one that is more important.

Part I: from a mathematical perspective

The covariance matrix of the features is as follows:

$$\Sigma = \begin{bmatrix} var(x_1) & cov(x_1, x_2) \\ cov(x_1, x_2) & var(x_2) \end{bmatrix}$$

Given the data range specified in the problem, $Var(x_1) \gg Var(x_2)$, and typically $Cov(x_1, x_2)$ is very small compared to $Var(x_1)$. Therefore, $Cov(x_1, x_2)$ can be approximately neglected in comparison to $Var(x_1)$.

Now, to find the principal components, we need to perform eigenvalue and eigenvector decomposition of the covariance matrix:

$$\Sigma v_i = \lambda_i v_i$$

The matrix Σ is equal to:

$$\Sigma = V \Lambda V^T$$

Where:

$$\Lambda = diag(\lambda_1, \lambda_2), \lambda_1 \approx var(x_1), v_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \lambda_2 \approx var(x_2), v_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, V = [v_1 \; v_2]$$

Part II: Comparison

Given that $\lambda_1 \gg \lambda_2$, consequently, during dimensionality reduction, only feature x_1 will be selected and it will dominate the principal component.

Part III: Standardization

Each feature must be transformed to a uniform scale to prevent differences in ranges from causing one feature to dominate another. We apply standardization to our data using the following formula:

$$x_i' = \frac{x_i - \bar{x}_i}{s_i}$$

Where s_i is the standard deviation of the i-th feature.

This ensures that the principal components truly represent the directions of the greatest “relative” variation, rather than merely the largest numerical units.

The presence of outliers can drastically alter the overall variance and distort the direction of the principal components.

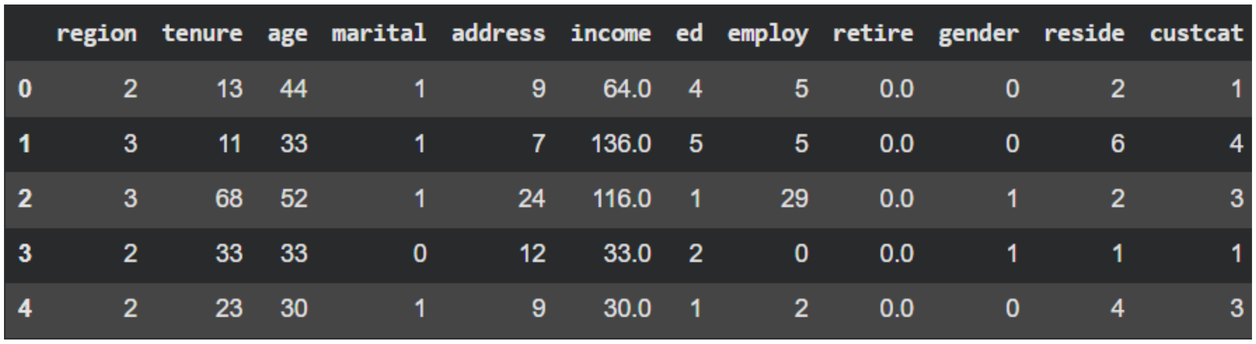
Outliers can be identified and removed using methods such as Z-score detection or IQR.

This helps PCA behave more stably, ensuring that the obtained components reflect the overall structure of the data.

Question 4

4.1. Exploratory Data Analysis (EDA)

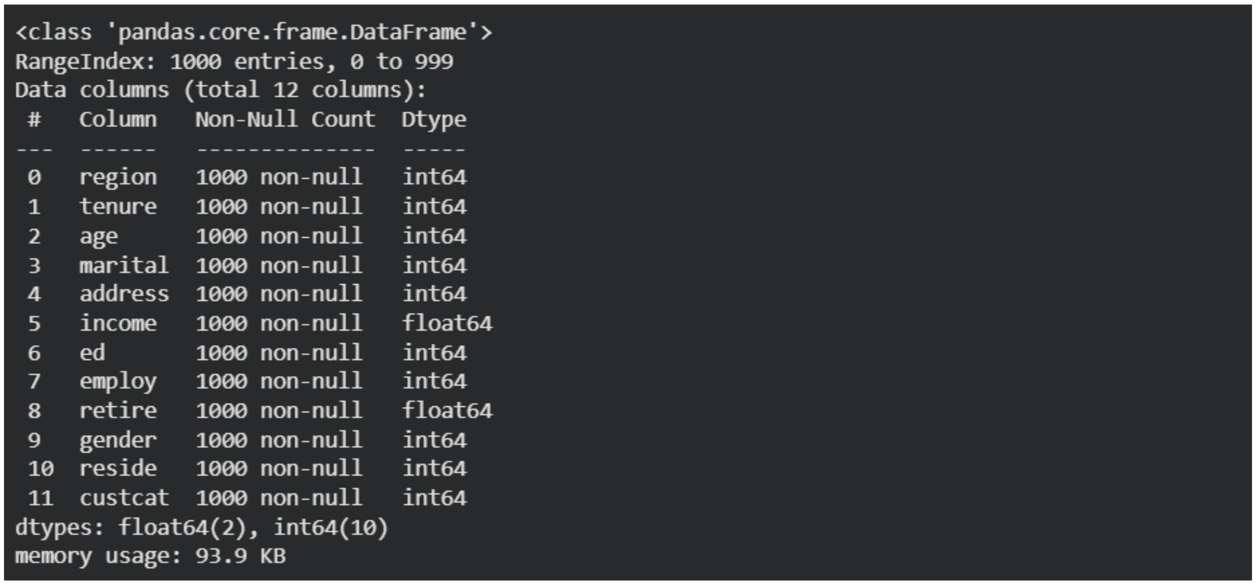
First, we read the dataset provided in the question using `read_csv`, and display the first five rows of the dataset using the head method. It's shown in fig 4.1.1.



	region	tenure	age	marital	address	income	ed	employ	retire	gender	reside	custcat
0	2	13	44	1	9	64.0	4	5	0.0	0	2	1
1	3	11	33	1	7	136.0	5	5	0.0	0	6	4
2	3	68	52	1	24	116.0	1	29	0.0	1	2	3
3	2	33	33	0	12	33.0	2	0	0.0	1	1	1
4	2	23	30	1	9	30.0	1	2	0.0	0	4	3

Fig. 4.1.1. dataset information

Now, we obtain the overall structure of the data using the info method and it's shown in figure 4.1.2.

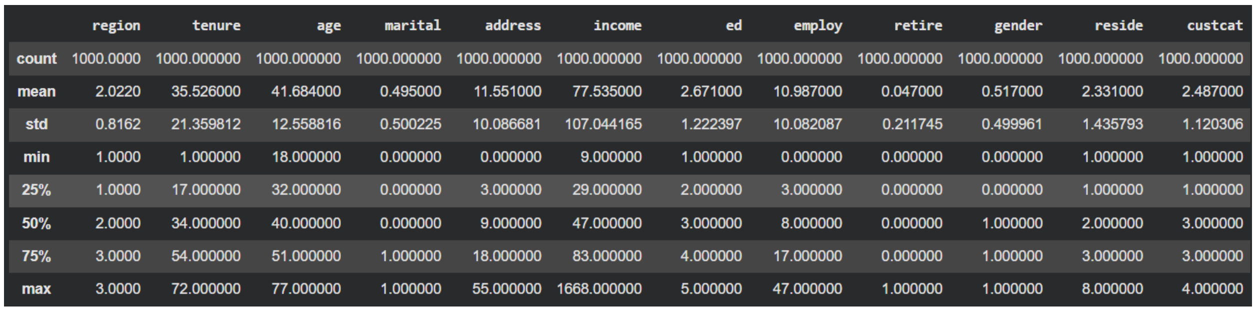


```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 12 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   region   1000 non-null   int64
 1   tenure   1000 non-null   int64
 2   age      1000 non-null   int64
 3   marital  1000 non-null   int64
 4   address  1000 non-null   int64
 5   income   1000 non-null   float64
 6   ed       1000 non-null   int64
 7   employ   1000 non-null   int64
 8   retire   1000 non-null   float64
 9   gender   1000 non-null   int64
 10  reside   1000 non-null   int64
 11  custcat  1000 non-null   int64
dtypes: float64(2), int64(10)
memory usage: 93.9 KB
```

Fig. 4.1.2. Overall structure of data

From the output of the info method, we find that since we have 1000 samples in the dataset and 1000 `non-null` values for each sample, there are no `NaN` values in the dataset. Also, based on the output of the head and info methods, we find that all features and the output are numerical, and the features tenure, age, address, and income are continuous, while the remaining features are categorical.

Using the describe method, we obtain the statistical summary of the data and show it as figure 4.1.3.



	region	tenure	age	marital	address	income	ed	employ	retire	gender	reside	custcat
count	1000.0000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	2.0220	35.526000	41.684000	0.495000	11.551000	77.535000	2.671000	10.987000	0.047000	0.517000	2.331000	2.487000
std	0.8162	21.359812	12.558816	0.500225	10.086681	107.044165	1.222397	10.082087	0.211745	0.499961	1.435793	1.120306
min	1.0000	1.000000	18.000000	0.000000	0.000000	9.000000	1.000000	0.000000	0.000000	0.000000	1.000000	1.000000
25%	1.0000	17.000000	32.000000	0.000000	3.000000	29.000000	2.000000	3.000000	0.000000	0.000000	1.000000	1.000000
50%	2.0000	34.000000	40.000000	0.000000	9.000000	47.000000	3.000000	8.000000	0.000000	1.000000	2.000000	3.000000
75%	3.0000	54.000000	51.000000	1.000000	18.000000	83.000000	4.000000	17.000000	0.000000	1.000000	3.000000	3.000000
max	3.0000	72.000000	77.000000	1.000000	55.000000	1668.000000	5.000000	47.000000	1.000000	1.000000	8.000000	4.000000

Fig. 4.1.3. Statical summary of data

The explanation of each row is as follows:

- Count: The number of available samples for each feature.
- Mean: The average value of each feature.
- Std: The standard deviation of each feature.
- Min: The minimum observed value for each feature.
- 25%: First quartile.
- 50%: Median.
- 75%: Third quartile.
- Max: The maximum observed value for each feature

Now, we plot the correlation between the data using a Heatmap. This plot can be seen in figure 4.1.4.

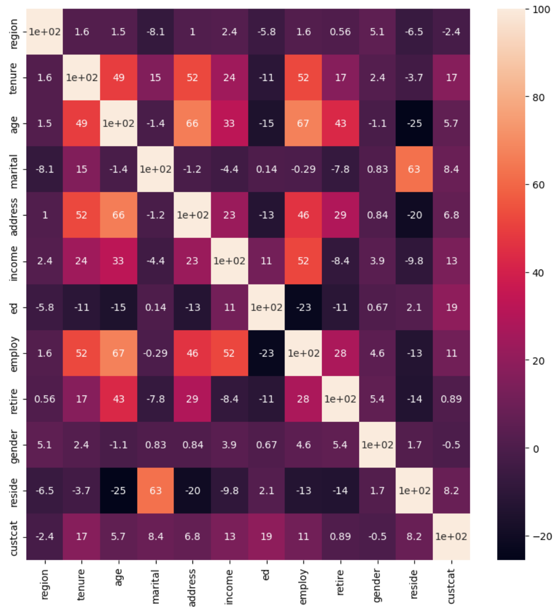


Fig. 4.1.4. Heatmap of correlation of dataset

According to Figure 4.1.4, the features with the highest correlation with the target variable are the “ed” feature with 19 percent, the “tenure” feature with 17 percent, the income feature with 13 percent, and the “employ” feature with 11 percent.

Using the pairplot function, we plot the scatter plot between the mentioned features, considering the classes. This plot is shown in 4.1.5.

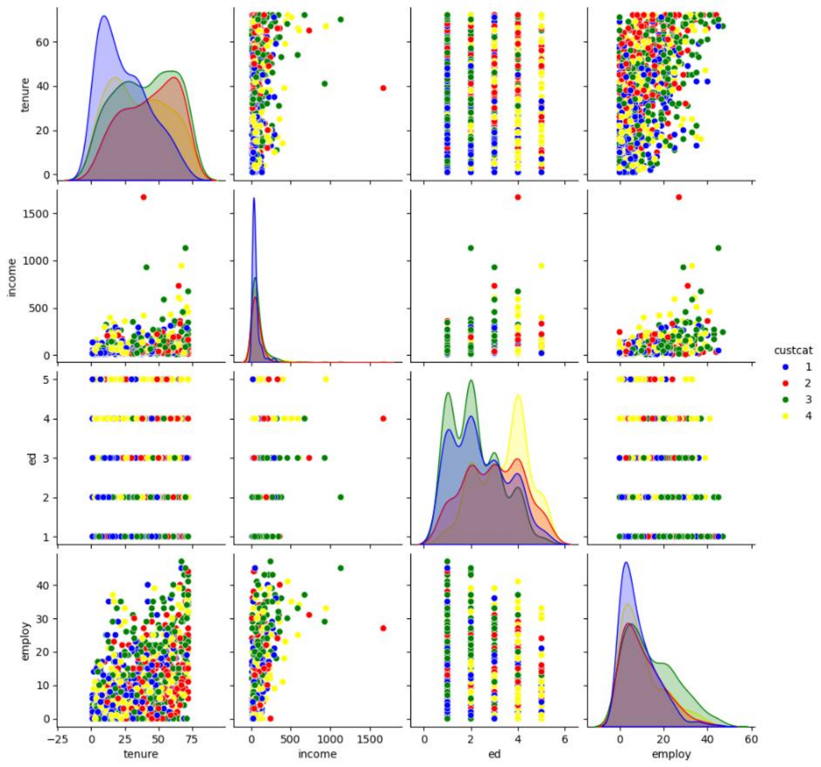


Fig. 4.1.5. Scatter plot for important features

Then, we plot the hexbin chart for the tenure and income features. This plot is shown in 4.1.6.

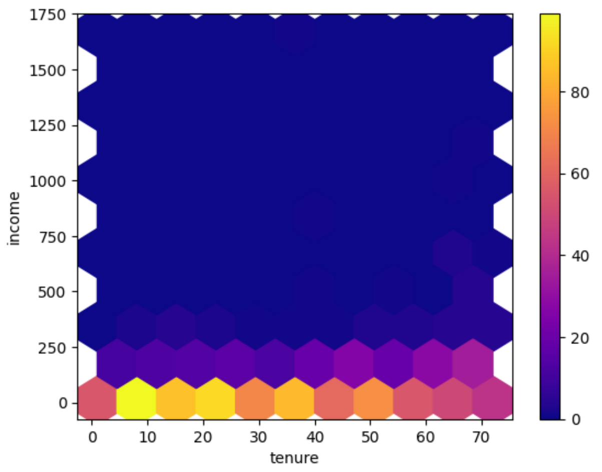


Fig 4.1.6. Hexbin plot for tenure and income feature

Figure 4.1.7 shows the class distribution using a countplot and Figure 4.1.8 shows the class distribution pie chart.

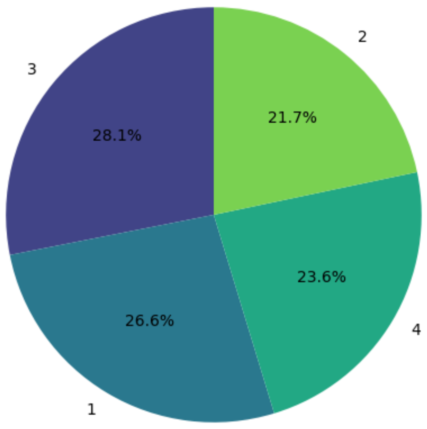


Fig. 4.1.7. Distribution plot by countplot

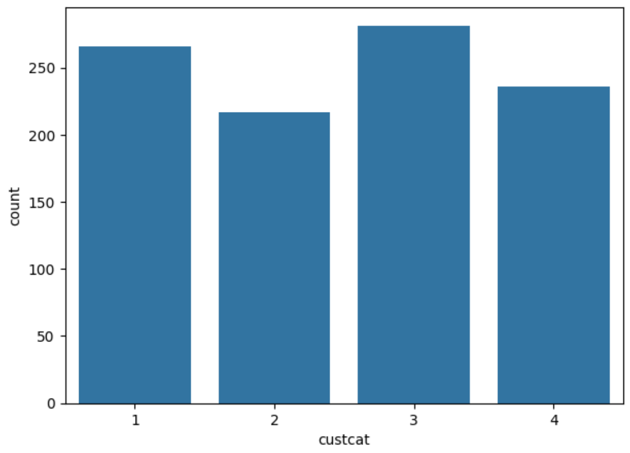


Fig. 4.1.8. Distribution plot by pie chart

The distribution among the classes is well-balanced. The largest share of the data belongs to Class 3 with 28.1 percent, and the smallest share belongs to Class 1 with 21.7 percent, which have a difference of about 7 percent, and this difference is small.

4.2. Data Pre-Processing

First, we separate the features from the dataset and store them in the variable X. The target is stored in the variable y. Since the classes in the dataset start from number 1 and we want them to start from number 0 for model training, we subtract one from the output.

Then, we perform preprocessing on the categorical data. This preprocessing can be done in two ways:

1. Label Encoding: In this method, a unique number is assigned to each category. This method is suitable for features where there is a logical order among their categories. If used for unordered categories, the model might suffer from a misleading interpretation of the relationship between categories.

2. One-Hot Encoding: In this method, a new column is created for each category, which receives a value of 1 if the sample belongs to that category and 0 otherwise. This method is suitable for features whose categories lack order. Although it increases the dimensionality of the data, multicollinearity can be avoided by dropping one of the new columns.

In this dataset, the suitable method for transforming these features is One-Hot Encoding, which we implement using the `get_dummies` function. Therefore, we implement the One-Hot Encoding method on the `region, reside`, and `ed` features, and implement the label encoding method on the other categorical features. Since the dataset itself is numerical, there is no need to use label encoding.

Then, we randomly split the data into training and testing sets.

Standardization and normalization are used so that the numerical features are on the same scale and the models can use them correctly.

- Normalization method for numerical features: Using the following formula, we limit the data range to the interval [0, 1]:

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

where X_{min} and X_{max} are the minimum and maximum values for the training data, respectively.

Standardization: Using the following formula, we transform the data to a distribution with a mean of 0 and a standard deviation of 1:

$$\frac{X - \mu}{\sigma}$$

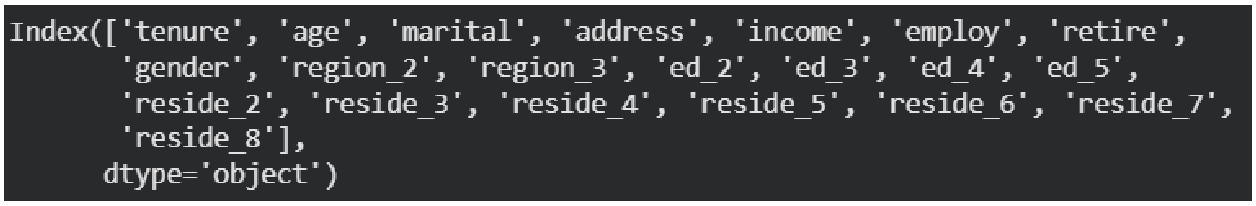
where μ and σ are the mean and standard deviation of the training data, respectively.

In this dataset, we use standardization. Feature Selection and Classical Modeling.

4.3. Feature Selection and Classical Modeling

For feature selection, we use two methods: Lasso regression and RFE.

- Using Lasso Regression: For this purpose, a Logistic Regression model equipped with the L1 penalty technique is used. The L1 penalty is a powerful mechanism for model regularization. This penalty intelligently drives the coefficients of features that have little effect on the prediction exactly to zero. Thus, by training the model with the dataset, we eliminate features that have zero coefficients. The following output which can be seen in figure 4.3.1 shows us which features were selected by the model.



```
Index(['tenure', 'age', 'marital', 'address', 'income', 'employ', 'retire',
       'gender', 'region_2', 'region_3', 'ed_2', 'ed_3', 'ed_4', 'ed_5',
       'reside_2', 'reside_3', 'reside_4', 'reside_5', 'reside_6', 'reside_7',
       'reside_8'],
      dtype='object')
```

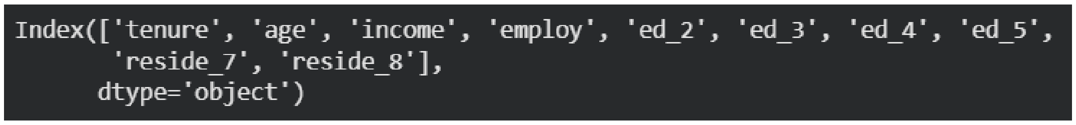
Fig. 4.3.1. Features selected by the model in the first attempt

As we can see in the output above, no features were eliminated, and all features were selected.

- Using RFE: RFE is a model-based method that iteratively eliminates features. The operation of this method is as follows.

1. Base Model Training: First, a Logistic Regression model is trained on the entire set of features.
2. Importance Assessment: The importance of each feature (typically based on its coefficient or feature importance score) is evaluated in the trained model.
3. Eliminating Least Important Features: The feature or features with the least importance are eliminated.
4. Iteration: These steps are repeated iteratively until the desired number of features is reached.

Figure 4.3.2. shows us which features were selected by the model.

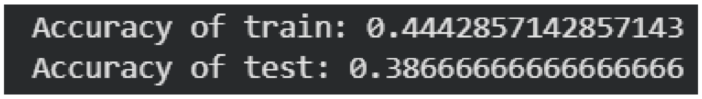


```
Index(['tenure', 'age', 'income', 'employ', 'ed_2', 'ed_3', 'ed_4', 'ed_5',
       'reside_7', 'reside_8'],
      dtype='object')
```

Fig. 4.3.2. Features selected by the model in the second attempt

As we can see in the output above, RFE has successfully eliminated several features from the dataset.

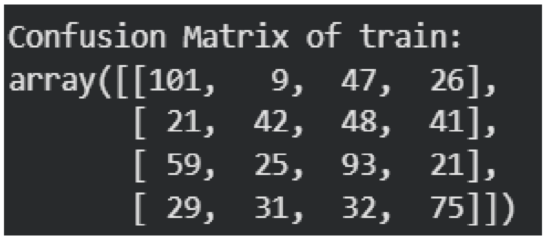
Therefore, we use the features selected by RFE to train the Logistic Regression model. After training the training data on the Logistic Regression model, we evaluate the model. The model's accuracy for the training and testing data is shown in figure 4.3.3.



```
Accuracy of train: 0.4442857142857143
Accuracy of test: 0.38666666666666666
```

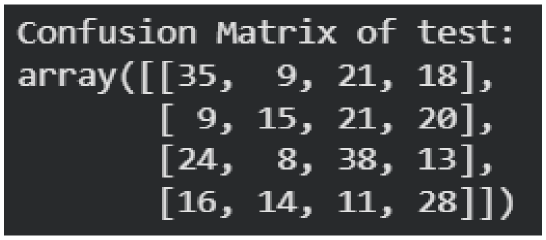
Fig. 4.3.3. Model accuracy

Also, the Confusion Matrix for the training and testing data is shown in figure 4.3.4 and figure 4.3.5.



```
Confusion Matrix of train:
array([[101,   9,  47,  26],
       [ 21,  42,  48,  41],
       [ 59,  25,  93,  21],
       [ 29,  31,  32,  75]])
```

Fig. 4.3.4. Confusion Matrix of training data



```
Confusion Matrix of test:
array([[35,  9, 21, 18],
       [ 9, 15, 21, 20],
       [24,  8, 38, 13],
       [16, 14, 11, 28]])
```

Fig. 4.3.5. Confusion Matrix of test data

We also plot the ROC curve for the training and testing data separately using the One vs Rest approach. Figure 4.3.6 shows these plots along with the AUC values for training data and figure 4.3.7 shows these plots along with the AUC values for test data.

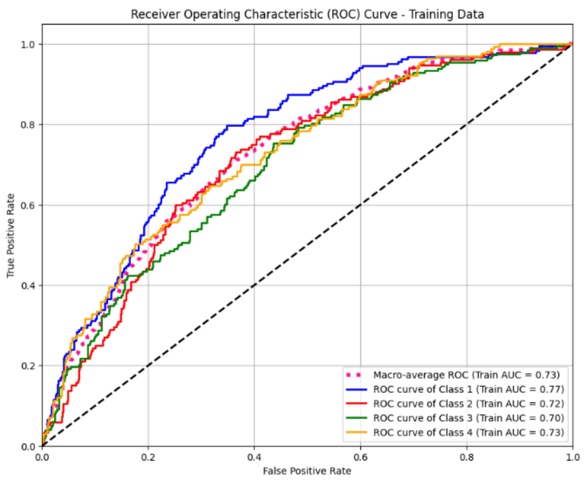


Fig. 4.3.6. ROC curve plot of training data

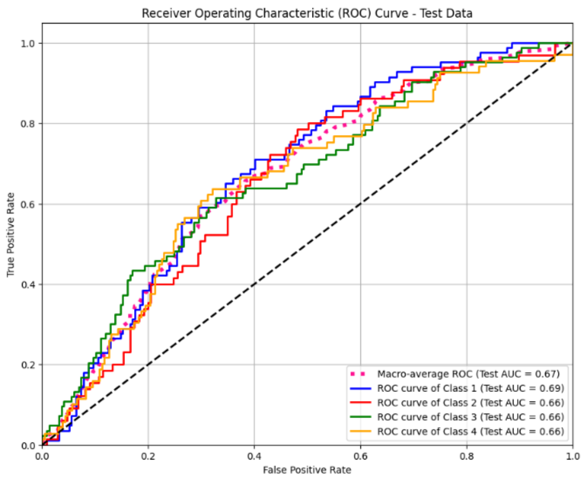
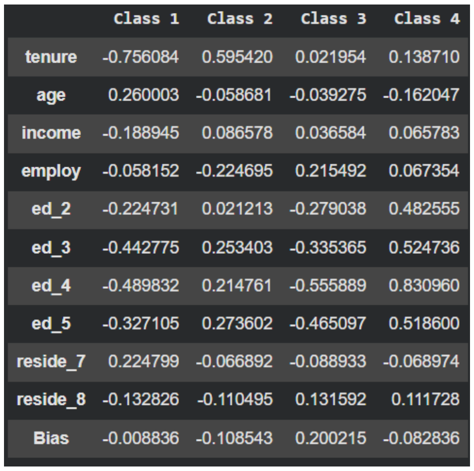


Fig. 4.3.7. ROC curve plot of test data

Figure 4.3.8 shows the model coefficients for each class.



	Class 1	Class 2	Class 3	Class 4
tenure	-0.756084	0.595420	0.021954	0.138710
age	0.260003	-0.058681	-0.039275	-0.162047
income	-0.188945	0.086578	0.036584	0.065783
employ	-0.058152	-0.224695	0.215492	0.067354
ed_2	-0.224731	0.021213	-0.279038	0.482555
ed_3	-0.442775	0.253403	-0.335365	0.524736
ed_4	-0.489832	0.214761	-0.555889	0.830960
ed_5	-0.327105	0.273602	-0.465097	0.518600
reside_7	0.224799	-0.066892	-0.088933	-0.068974
reside_8	-0.132826	-0.110495	0.131592	0.111728
Bias	-0.008836	-0.108543	0.200215	-0.082836

Fig. 4.3.8. Model coefficient for each class

According to the figure 4.2.8, the `tenure` feature has the most influence on Class 1 and Class 2. If the value of tenure increases, the probability of belonging to Class 1 decreases but the probability of belonging to Class 2 increases. The `ed_4` feature also has the most influence on Class 3 and Class 4. If the value of `ed_4` increases, the probability of belonging to Class 3 decreases but the probability of belonging to Class 4 increases.

4.4. Feature Visualization using Dimensionality Reduction

In this section, we train each model with the training data so that it reduces the features to two dimensions. Then, by feeding the training and testing samples to the model and receiving the output, we show their scatter plot.

- To train PCA, it is sufficient to only provide the features to the model.
- However, since LDA learns in a supervised manner, the target data must also be provided to the model in addition to the features

Figure 4.4.1 shows the plot of the mapped features by PCA and LDA.

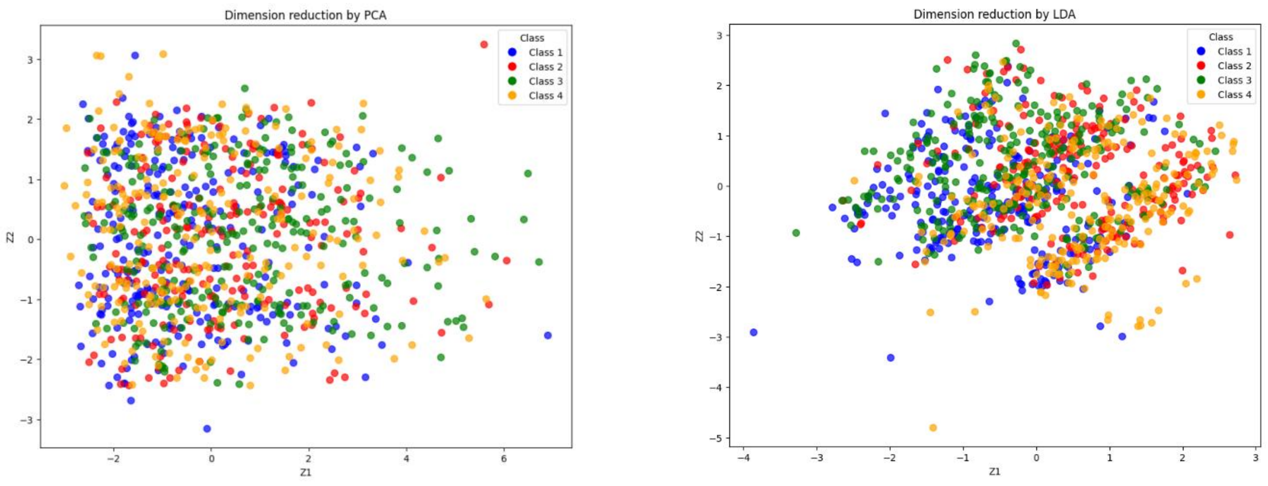


Fig. 4.4.1. Dimension reduction with PCA and LDA

For the MLP (Multi-Layer Perceptron) neural network, we used the following architecture:

21 (input dimension) → 128 (relu) → 64 (relu) → 32 (relu) → 2 (Number of Features) → 4(Number of classes)

This neural network is designed to be supervised. This means that the target must also be provided to the model in addition to the features.

Figure 4.1.2 shows the plot of the mapped features by MLP.

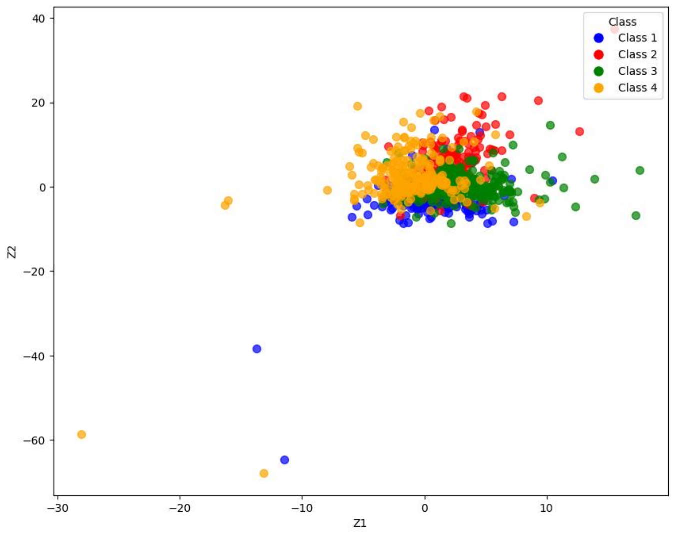


Fig. 4.4.2. Dimension reduction with PCA and LDA by MLP model

The PCA and LDA models could not perform the separation. This is because the features lack linear separability. However, since MLP applies a nonlinear mapping to the data, unlike the other two models, it has almost succeeded in performing the separation.

Question 5

Part I: Literature Review

1. The article focuses on predicting housing prices using the Housing Price in Beijing dataset.
    - City of Study: Beijing
    - Initial Dataset Specifications: The original dataset consisted of over 300,000 data points with 26 variables, covering housing transactions between 2009 and 2018.
    - Final Dataset Specifications: After pre-processing, the final dataset contained 231,962 data points with 19 features.
    - Model Input: Following the standardization of numerical values and One-Hot Encoding of categorical values, the final dataset used for model training comprised 58 features.
2. Data pre-processing involved critical stages of cleaning, feature engineering, and final preparation for modeling.
    - Handling Missing Data: Variables with more than 50% missing values were removed. Any observation (row) containing missing values was deleted from the dataset.
    - Feature Engineering and Transformation: The “construction Time” feature was replaced with an “age” feature. A new “distance” feature was added to represent the property's distance from the center of Beijing. The “floor” feature was split into two new features: “floorType” and “floorHeight”. Features related to the number of kitchens, bathrooms, and living rooms were removed due to data ambiguity. The number of living rooms (which were effectively bedrooms) was capped within a range of 1 to 4.
    - Outlier Removal: Outliers were identified and removed using the Inter-Quartile Range (IQR) method.
    - Final Preparation: Numerical values were standardized. Categorical values were One-Hot Encoded. The dataset was split into Training and Test sets with an 80-20 ratio.
3. The article employed a combination of traditional and advanced machine learning methods, evaluating them based on the RMSLE (Root Mean Squared Logarithmic Error) metric.
    - Random Forest, XGBoost, Light GBM, Hybrid Regression and Stacked Generalization Regression.

Part II: Dataset

1. Number of Samples and Features
    - Number of Samples: 545
    - Number of Features: 13
2. Data Type of Each Feature
    - The dataset consists of 6 numerical features of integer type “int 64” and 7 text feature “object”.

Feature	Data Type	Description
price	int64	Price of the house
area	int64	Area of the house
bedrooms	int64	Number of bedrooms
bathrooms	int64	Number of bathrooms
stories	int64	Number of stories (floors)
parking	int64	Number of parking spaces
mainroad	object	Access to the main road
guestroom	object	Availability of a guest room
basement	object	Presence of a basement
hotwaterheating	object	Availability of a hot water heating system
airconditioning	object	Availability of air conditioning
prefarea	object	Location in a preferred area
furnishingstatus	object	Furnishing status of the house

3. Number of Unique Values per Feature
    - The number of unique values for each feature is listed below. This count indicates the diversity of values within each column.

Feature	Unique Count	Variable Type
area	284	Numerical (Continuous)
price	219	Numerical (Continuous)
bedrooms	6	Numerical (Discrete)
bathrooms	4	Numerical (Discrete)
stories	4	Numerical (Discrete)
parking	4	Numerical (Discrete)
furnishingstatus	3	Categorical
mainroad	2	Categorical (Binary)
guestroom	2	Categorical (Binary)
basement	2	Categorical (Binary)
hotwaterheating	2	Categorical (Binary)
airconditioning	2	Categorical (Binary)
prefarea	2	Categorical (Binary)

Part III: Exploratory Data Analysis

Exploratory Data Analysis (EDA) is a critical approach in data science that involves using visual and statistical methods to summarize, describe, and explore a dataset. The primary goal of EDA is to understand the underlying structure of the data, identify hidden patterns, detect outliers, and recognize relationships between variables before commencing complex modeling.

EDA is an initial, iterative phase in the data analysis process, which typically includes:

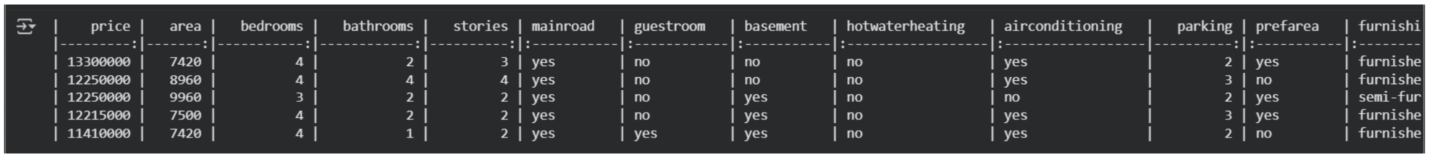
- Statistical Summaries: Calculating statistics such as mean, median, standard deviation, range, and quartiles to understand the distribution of numerical data.
- Visualization Techniques: Creating plots like histograms, box plots, scatter plots, and bar charts to visualize the distribution and relationships of variables.
- Preliminary Feature Engineering: Identifying the need for data transformations or the conversion of categorical variables into numerical ones.

EDA acts as a data "detective" playing a pivotal role in the success of machine learning and data analysis projects:

- Better Data Understanding
- Identifying Data Issues
- Guiding the Modeling Process
- Validating Assumptions

EDA not only tells us what the data is, but more importantly, it tells us how we should use it to build a robust and generalizable model.

Now, using the `gdown` command, we will display the first 5 rows. The output is shown in Figure 5.3.1.



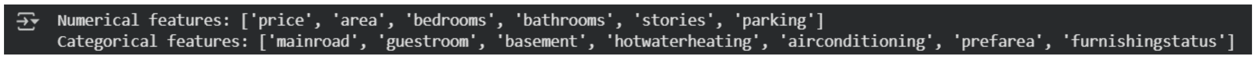
```
|    price |   area |   bedrooms |   bathrooms |   stories | mainroad   | guestroom   | basement   | hotwaterheating   | airconditioning   |   parking | prefarea   | furnishi
|---------:|-------:|-----------:|------------:|----------:|:-----------|:------------|:-----------|:------------------|:------------------|----------:|:-----------|:--------
| 13300000 |   7420 |          4 |           2 |         3 | yes        | no          | no         | no                | yes               |         2 | yes        | furnishe
| 12250000 |   8960 |          4 |           4 |         4 | yes        | no          | no         | no                | yes               |         3 | no         | furnishe
| 12250000 |   9960 |          3 |           2 |         2 | yes        | no          | yes        | no                | no                |         2 | yes        | semi-fur
| 12215000 |   7500 |          4 |           2 |         2 | yes        | no          | yes        | no                | yes               |         3 | yes        | furnishe
| 11410000 |   7420 |          4 |           1 |         2 | yes        | yes         | yes        | no                | yes               |         2 | no         | furnishe
```

Fig. 5.3.1. First 5 rows of Housing.csv dataset

Based on the analysis of the Housing.csv dataset, the features are divided into two groups. numerical "int64" and categorical "object".

1. Numerical Features: This dataset contains 6 numerical features.
    - price, area, bedrooms, bathrooms, stories and parking
2. Categorical Feature: This dataset contains 7 categorical features.
    - main road, guest room, basement, hot water heating, air conditioning, prefarea and fumishing status.

Features Categorization is shown in Figure 5.3.2.



```
Numerical features: ['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'parking']
Categorical features: ['mainroad', 'guestroom', 'basement', 'hotwaterheating', 'airconditioning', 'prefarea', 'furnishingstatus']
```

Fig. 5.3.2. Features Categorization

At this stage of Exploratory Data Analysis (EDA), Count Plots were used to visualize the frequency distribution of values in four selected categorical features: `mainroad`, `airconditioning`, `prefarea`, and `furnishingstatus`.

```
categorical_features_to_plot = ['mainroad', 'airconditioning', 'prefarea', 'furnishingstatus']
```

The `sns.countplot` chart is visible in Figure 5.3.3.

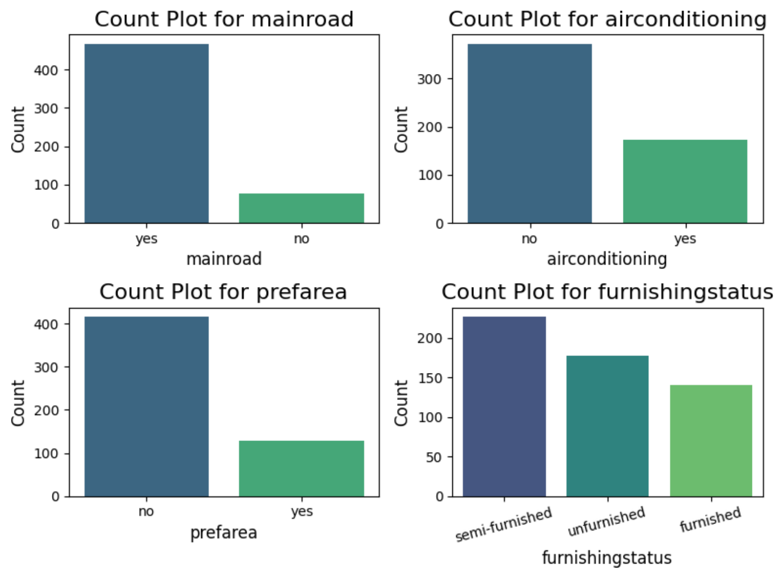


Fig. 5.3.3. `sns.countplot` of 4 features

The table below presents the dominant categories and imbalance analysis for categorical variables. It can be observed that most binary features exhibit significant imbalance, with one category being strongly dominant.

Feature	**Dominant Values**	**Imbalance Analysis Result**
mainroad	yes (most houses have access to the main road)	Imbalanced distribution (strong dominance of *yes*)
airconditioning	no (most houses do not have air conditioning)	Imbalanced distribution (strong dominance of *no*)
prefarea	no (most houses are not located in a preferred area)	Imbalanced distribution (strong dominance of *no*)
furnishingstatus	semi-furnished (partially furnished)	More balanced than others, but *unfurnished* has the lowest frequency

In this stage, the distribution of three key numerical features (price, area, bedrooms) was selected, and their distribution plots are illustrated in Figure 5.3.4.

```
eda_numerical_features = ['price', 'area', 'bedrooms']
```

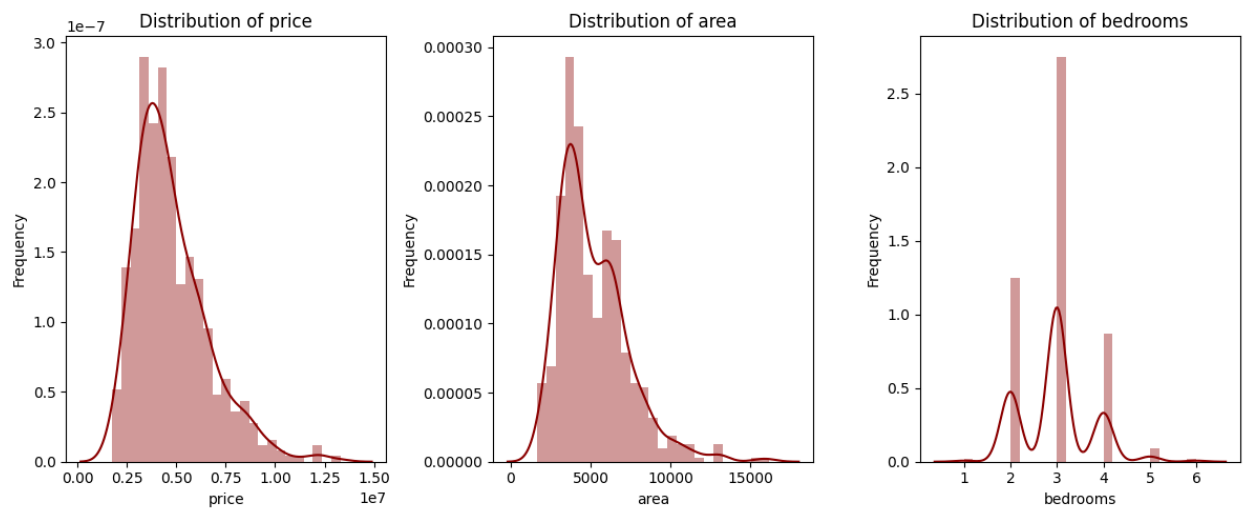


Fig. 5.3.4. `sns.distplot` of 3 features

1. Distribution of Price
    - Distribution Shape: The distribution of Price is strongly right-skewed. The majority of houses are concentrated within a lower price range.
    - Outlier Presence: outliers are present. The long tail of the distribution to the right, where a small number of houses are positioned at significantly higher prices (e.g., prices above $10,000,000) clearly indicates the existence of price outliers.
2. Distribution of Area
    - Distribution Shape: The distribution of Area also exhibits a strong positive skew. The vast majority of houses have a relatively small area.
    - Outlier Presence: outliers are observed. The presence of extremely large area values that appear with very low frequency at the far end of the plot indicates the existence of outliers in the area data.
3. Distribution of Bedrooms
    - Distribution Shape: This is a discrete distribution, centered around the values of 3 and 4. It exhibits significantly less skewness compared to the other two features.
    - Outlier Presence: Although Bedrooms is a discrete feature with a limited set of values, houses with a very high number of bedrooms (such as 5 or 6), relative to the majority of the data, can be considered contextual outliers in this specific dataset.

We use the `pairplot` command to visualize the relationships between the numerical features. This plot is shown in figure 5.3.5.

A. Diagonal: Univariate Distributions

- This section displays the histogram for each individual variable. Once again, price and area demonstrate a strong positive skew.

B. Pairwise Relationships (Scatter Plots)

- price vs. area: A positive and relatively strong relationship is observed. As the area increases, the price also tends to increase. Although this relationship is linear, it exhibits significant variance.
- price vs. bathrooms: A positive correlation exists. Houses with more bathrooms tend to have higher prices. This relationship appears step-like due to the discrete nature of the bathroom count.
- price vs. stories: A positive trend is present. Houses with a greater number of stories are generally more expensive. This relationship is also discrete and well-defined.
- area vs. other discrete features (bedrooms, bathrooms, stories): These also show positive correlations, indicating that larger houses typically have more bedrooms, bathrooms, and stories.

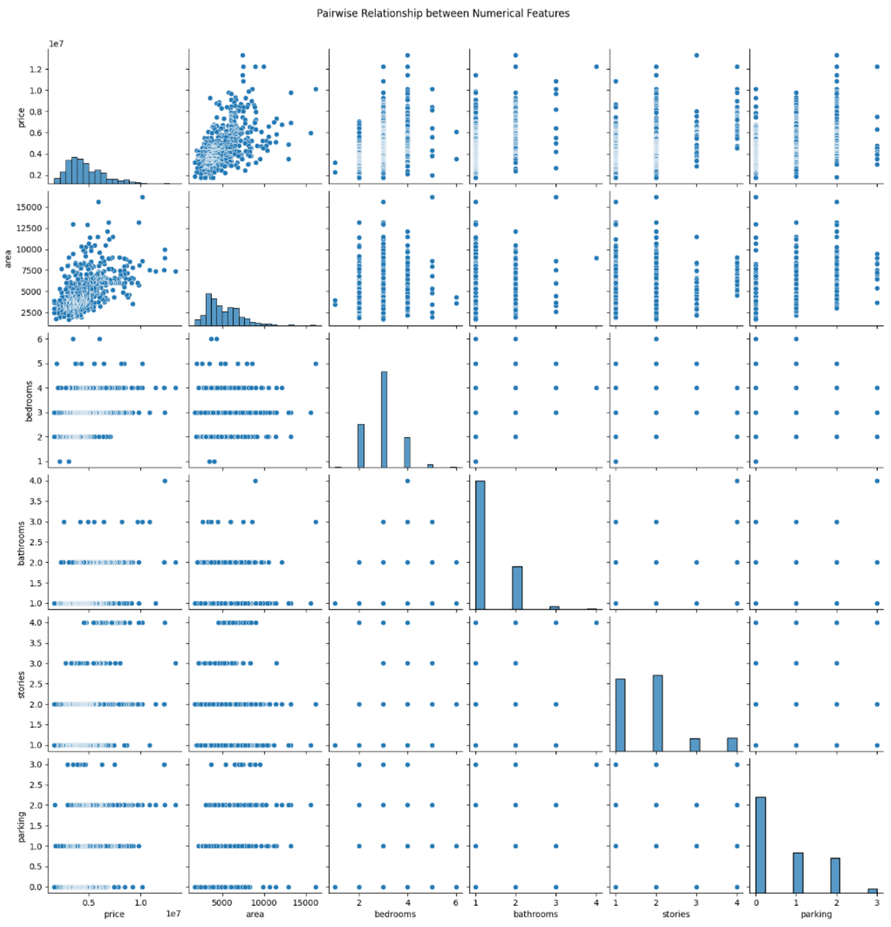
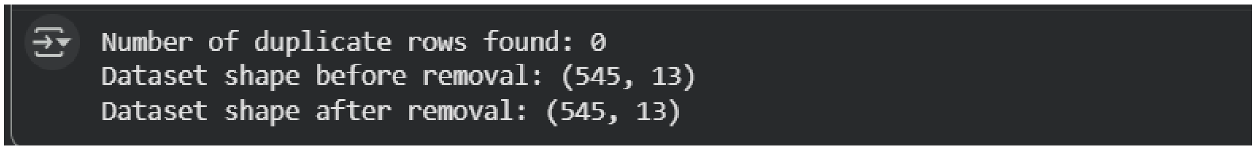


Fig. 5.3.5. `sns.pairplot` of numerical features

The Pairplot analysis validates that the features area, bathrooms, and stories serve as the strongest linear predictors for price, demonstrating a clear positive correlation. This finding is vital for informing feature selection in subsequent linear regression modeling.

Part IV: Pre-processing

First, we check whether there are any duplicate rows in the dataset. The output of this check is shown in Figure 5.4.1.

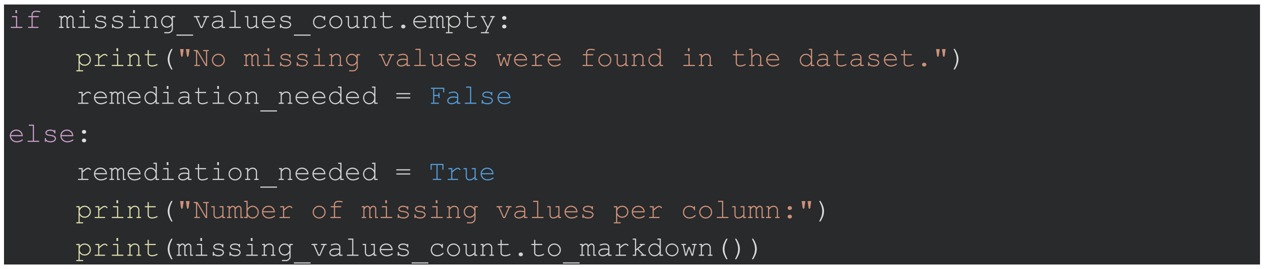


```
Number of duplicate rows found: 0
Dataset shape before removal: (545, 13)
Dataset shape after removal: (545, 13)
```

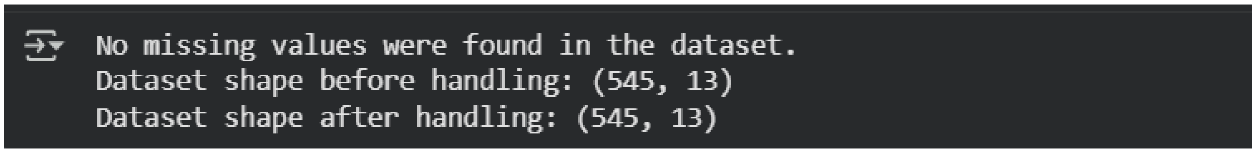
Fig. 5.4.1. Number of duplicate rows

Analysis confirmed the absence of any fully duplicate rows within the dataset. As a result, no rows were removed, preserving the original dataset dimensions of (545, 13). The data is verified to be free of duplicates.

In the next step, we check whether there is any missing data. The result of this check is shown in Figure 5.4.2.



```
if missing_values_count.empty:
    print("No missing values were found in the dataset.")
    remediation_needed = False
else:
    remediation_needed = True
    print("Number of missing values per column:")
    print(missing_values_count.to_markdown())
```



```
No missing values were found in the dataset.
Dataset shape before handling: (545, 13)
Dataset shape after handling: (545, 13)
```

Fig. 5.4.2. Replacing missing values

The analysis confirmed that the “Housing.csv” dataset has no missing data. Hence, no remedial step was taken, and the original count of 545 rows and 13 features actually remained true. This is a good finding because we would have had to impute our missing data and would have introduced bias.

1. Introduction to Categorical Feature Encoding Methods

Categorical Feature Encoding is the process of converting non-numerical data numerical values that can be processed by machine learning algorithms.

Common encoding methods are displayed in the table below.

Encoding Type	**Description**	**Appropriate Use Case**
Label Encoding	Converts each category into a unique integer (e.g., *'Red'* → *1*, *'Green'* → *2*).	Suitable for ordinal features, where a natural order exists among categories (e.g., *Small* < *Medium* < *Large*).
One-Hot Encoding	Converts each category into a new binary feature (column). If a sample belongs to that category, it is assigned *1*; otherwise *0*.	Suitable for nominal features, where no inherent order exists among categories (e.g., colors, cities, or *mainroad*, where *yes* is not better than *no*).
Binary Encoding	Categories are first encoded as integers, and then these integers are converted into binary form.	Appropriate for features with a large number of categories, as it reduces dimensionality (fewer columns than one-hot).

2. Selection of Appropriate Method and Rationale

- Binary Feature: These features have two states (yes/no). This is a specific case of Label Encoding, and using the mapping yes → 1 and no → 0 is the simplest and most appropriate method. Features: `mainroad, guestroom, basement, hotwaterheating, airconditioning, prefarea.`
- Multi-Categorical Features: The `furnishingstatus` feature has three categories. Rationale: Although this feature could be considered ordinal, for simplicity and to avoid imposing strong ordinal assumptions that might mislead the model, we use the One-Hot Encoding method. Furthermore, since it has only three categories, One-Hot Encoding results in a minor increase in dimensionality (creating two new columns).

At this stage, the categorical features of the dataset (including 6 binary features and 1 three-state feature) were converted into numerical values to prepare them for machine learning algorithms.

Metric	**Value**	**Analysis**
Final dataset shape	(545, 14)	The number of columns increased from 13 to 14.
Binary columns	mainroad, airconditioning,	The *yes* values were converted to *1*, and *no* values to *0*.
One-Hot columns	semi-furnished, unfurnished	Two new columns were created for the furnishing status feature.
Removed column	furnishingstatus	The original categorical column was removed after encoding.

Three common methods for identifying and managing outliers in data analysis include:

1. Standard Deviation Method: Any data point falling outside the range of 3σ from the mean is considered an outlier and removed. It's suitable for Data with near-normal distributions.
2. Interquartile Range (IQR) Method: This method uses the range between the first quartile Q_1 and third quartile Q_3 any point outside the range $[Q_1 - 1.5 \times IQR, Q_3 + 1.5 \times IQR]$ is considered an outlier. It's suitable for Data with skewed distributions or any distribution requiring a more robust method against outliers.
3. Logarithmic/Statistical Transformation: Instead of removing data, data skewness can be reduced through transformations such as logarithmic conversion. This brings outliers closer to the center of the distribution and reduces their impact on the model. It's suitable for Numerical data with strong positive skewness (such as `price` and `area` in our dataset).

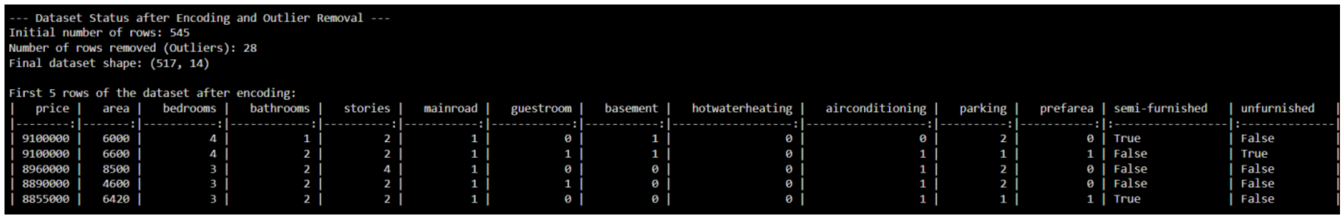
Given that the key features (`price` and `area`) exhibit strong positive skewness, the best approach is a combination of methods:

- Using IQR: To identify and remove extreme outliers.
- Using Logarithm: To reduce the remaining data skewness, instead of removing all outliers which would lead to loss of information.

We will use the IQR method to remove extreme outliers in the `price` and `area` features.

```
df[binary_vars] = df[binary_vars].apply(lambda x: x.map({'yes': 1, 'no': 0}))
furnishing_status_dummies = pd.get_dummies(df['furnishingstatus'],
drop_first=True)
df = pd.concat([df, furnishing_status_dummies], axis=1)
df.drop('furnishingstatus', axis=1, inplace=True)

outlier_features = ['price', 'area']
original_rows = df.shape[0]

for col in outlier_features:
    # Calculate IQR statistics
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1

    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    df = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]

rows_removed = original_rows - df.shape[0]
```

At this stage, preprocessing operations (encoding categorical features) and outlier removal for the `price` and `area` features using the IQR method have been completed. Output of the code is shown in figure 5.4.3.



```
--- Dataset Status after Encoding and Outlier Removal ---
Initial number of rows: 545
Number of rows removed (Outliers): 28
Final dataset shape: (517, 14)

First 5 rows of the dataset after encoding:
|   price |   area |   bedrooms |   bathrooms |   stories |   mainroad |   guestroom |   basement |   hotwaterheating |   airconditioning |   parking |   prefarea | semi-furnished   | unfurnished   |
|--------:|-------:|-----------:|------------:|----------:|-----------:|------------:|-----------:|------------------:|------------------:|----------:|-----------:|:-----------------|:--------------|
| 9100000 |   6000 |          4 |           1 |         2 |          1 |           0 |          1 |                 0 |                 0 |         2 |          0 | True             | False         |
| 9100000 |   6600 |          4 |           2 |         2 |          1 |           1 |          1 |                 0 |                 1 |         1 |          1 | False            | True          |
| 8960000 |   8500 |          3 |           2 |         4 |          1 |           0 |          0 |                 0 |                 1 |         2 |          0 | False            | False         |
| 8890000 |   4600 |          3 |           2 |         2 |          1 |           1 |          0 |                 0 |                 1 |         2 |          0 | False            | False         |
| 8855000 |   6420 |          3 |           2 |         2 |          1 |           0 |          0 |                 0 |                 1 |         1 |          1 | True             | False         |
```

Fig. 5.4.3. Dataset information after outlier cleaning

- Initial Number of Rows: 545
- Number of removed Rows (Outlier): 28
- Final Dataset Shape: (517, 14)

Binary features (`basement`, `mainroad`, `guestroom`, `hotwaterheating`, `airconditioning`, `prefarea`) converted to 0 and 1.

Multi-Category feature (`furnishgstatus`) by One-Hot Encoded into two new columns.

Dataset dimensionality increased from 13 to 14.

Now the cleaned dataset (from which outliers were removed and categorical features were encoded in the previous step) is split into 70% training and 30% testing sets.

The numerical features of the dataset and the target variable were normalized using `MinMaxScaler`. This operation maps all values to the [0, 1] range, which is essential for optimal performance of gradient-based models.

Part V: Feature Selection

The correlation matrix between all features of the cleaned and encoded dataset was calculated and visualized to identify linear relationships between the variables and it's shown as figure 5.5.1.

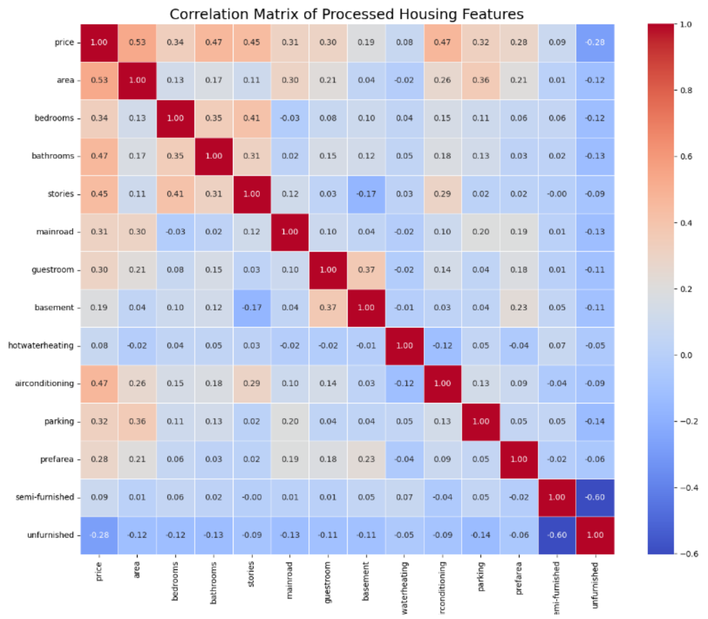


Fig. 5.5.1. Corraltion matrix of features

Based on the correlation matrix output, the features showing the strongest positive correlation with price are:

1. Area: Strongest predictor of house price, larger houses command higher prices.
2. Air Conditioning: Surprisingly strong influence on price, indicates this amenity is highly valued in the housing market.
3. Bathrooms: Significant impact on property value, reflects practical utility and comfort factors.

These three features will serve as the primary predictors in our initial models, with potential inclusion of additional moderately correlated features for performance comparison.

Bathrooms, stories, air conditioning all show moderate positive correlations with each other and modern houses typically contain all these amenities together.

semi-furnished ↔ unfurnished = -0.60, These two furnishing types are mutually exclusive alternatives.

PCA was applied to reduce the dimensionality of the training feature set while preserving the maximum possible data information (variance).

After implementing PCA, the cumulative explained variance plot is shown as Figure 5.5.2.

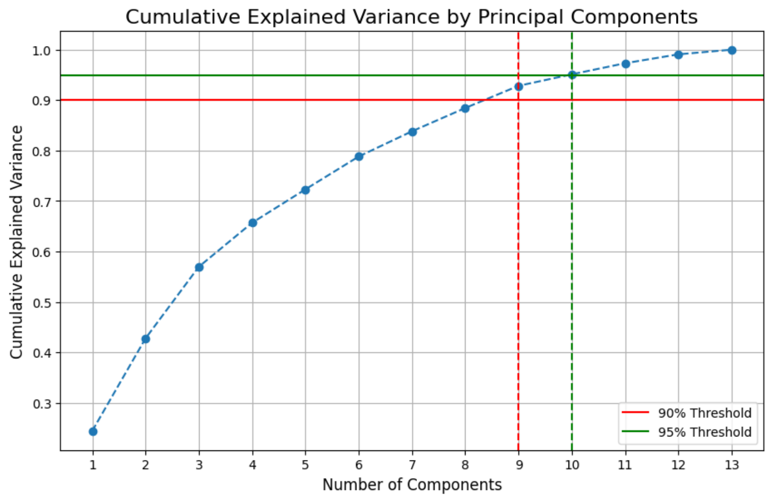


Fig. 5.5.2. CEV plot of PCA

The goal of dimensionality reduction with PCA is to remove noise, since components with low variance primarily represent noise. Additionally, this reduces complexity; decreasing the number of features reduces training time and improves the interpretability of more complex models.

Given that we have 13 initial features, selecting 10 principal components allows us to retain approximately 95% of the variance of the original data, while eliminating 3 dimensions. This represents a good balance between preserving information and reducing computational complexity.

Part V: Bonus section

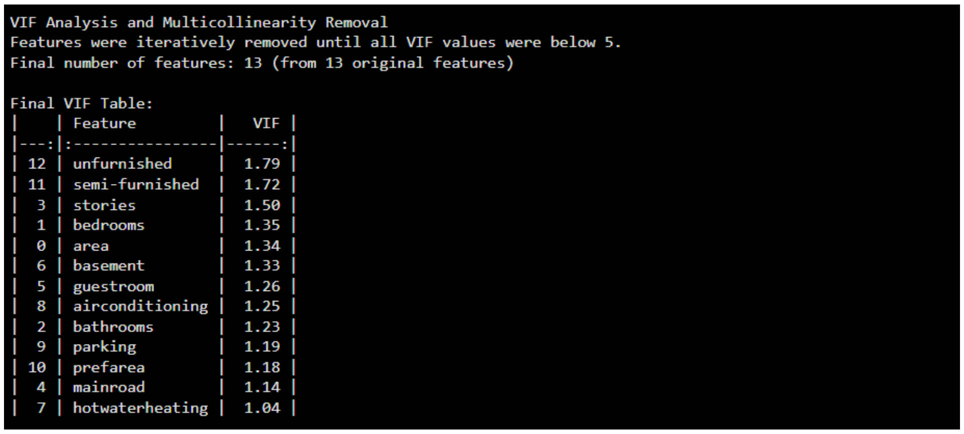
Multicollinearity: A situation in which two or more independent variables (X) in a regression model are highly correlated with each other. This problem makes the interpretation of the model's coefficients unreliable, as the model cannot isolate the unique effect of each feature on the target variable.

1. VIF: VIF measures how much the variance of an estimated regression coefficient β_i increases due to multicollinearity. For each feature, VIF is calculated by performing a regression of that feature (as the target variable) against all other features in the model.
$$VIF_i = \frac{1}{1 - R_i^2}$$
A feature with $VIF > 5 \, or \, VIF > 10$ typically indicates problematic multilinearity. Features with the highest VIF are iteratively removed until the VIF of all features falls below a specified threshold.

Using the VIF method with a threshold of 5, we remove features with high multicollinearity.

```
while (vif_df['VIF'].max() > 5):
    feature_to_drop = vif_df.iloc[0]['Feature']
    X_train_vif.drop(columns=[feature_to_drop], inplace=True)
    vif_df = calculate_vif(X_train_vif)
```

The output of this code is shown in Figure 5.5.3.



```
VIF Analysis and Multicollinearity Removal
Features were iteratively removed until all VIF values were below 5.
Final number of features: 13 (from 13 original features)

Final VIF Table:
|    | Feature         |   VIF |
|---:|:----------------|------:|
| 12 | unfurnished     |  1.79 |
| 11 | semi-furnished  |  1.72 |
|  3 | stories         |  1.50 |
|  1 | bedrooms        |  1.35 |
|  0 | area            |  1.34 |
|  6 | basement        |  1.33 |
|  5 | guestroom       |  1.26 |
|  8 | airconditioning |  1.25 |
|  2 | bathrooms       |  1.23 |
|  9 | parking         |  1.19 |
| 10 | prefarea        |  1.18 |
|  4 | mainroad        |  1.14 |
|  7 | hotwaterheating |  1.04 |
```

Fig 5.5.3. Number of removed feature after applying VIF

The VIF analysis revealed no significant multicollinearity issues in the dataset. The highest VIF value was 1.79 which is well below the common thresholds of 5 or 10. This means that the coefficients of a linear regression model trained with these features will be reliable and interpretable.

2. REF: Recursive Feature Elimination is a model-based feature selection method that identifies the best subset of features for a target model.
The operation of RFE follows a greedy and recursive algorithm. This algorithm:
    - Trains the model using all features.
    - Examines the feature importance scores.

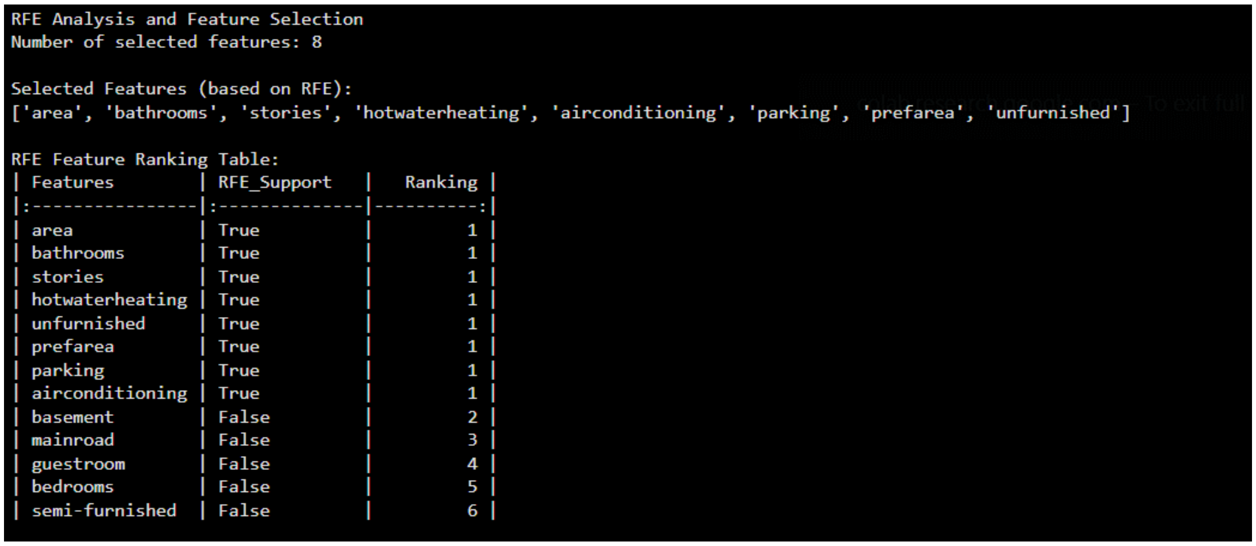
  - Removes the feature with the least importance.
  - Iteratively repeats steps 1–3 on the remaining feature set until the desired number of features is reached.

It selects the optimal features for the target model, thereby improving model accuracy and preventing overfitting.

Now, we proceed with feature selection using RFE.

```
rfe_results = pd.DataFrame({'Features': X_train.columns,
                            'RFE_Support': rfe.support_,
                            'Ranking': rfe.ranking_})
selected_features_rfe = rfe_results[rfe_results['RFE_Support'] ==
True]['Features'].tolist()
```

The output of this code is shown in Figure 5.5.4.



```
RFE Analysis and Feature Selection
Number of selected features: 8

Selected Features (based on RFE):
['area', 'bathrooms', 'stories', 'hotwaterheating', 'airconditioning', 'parking', 'prefarea', 'unfurnished']

RFE Feature Ranking Table:
| Features        | RFE_Support   |   Ranking |
|:----------------|:--------------|----------:|
| area            | True          |         1 |
| bathrooms       | True          |         1 |
| stories         | True          |         1 |
| hotwaterheating | True          |         1 |
| unfurnished     | True          |         1 |
| prefarea        | True          |         1 |
| parking         | True          |         1 |
| airconditioning | True          |         1 |
| basement        | False         |         2 |
| mainroad        | False         |         3 |
| guestroom       | False         |         4 |
| bedrooms        | False         |         5 |
| semi-furnished  | False         |         6 |
```

Fig 5.5.4. Number of removed feature after applying RFE

The 8 features selected by the RFE model as the most effective based on predictive power in the linear regression model are:

- `area`
- `bathrooms`
- `stories`
- `hotwaterheating`
- `parking`
- `airconditioning`
- `prefarea`
- `unfurnished`

The RFE model correctly emphasizes structural and luxury features as well as locational characteristics.

The feature `bedrooms` is ranked 5th and is not among the top 8 features. This indicates that in the presence of stronger features such as `area` and `stories` the `bedrooms` variable provides little additional information.

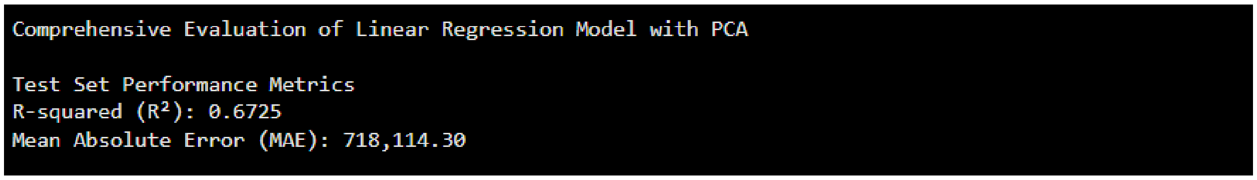
Part VI: Model Training

1. Multiple Linear Regression

We train the model using multiple linear regression.

```
# Apply PCA and Train Linear Regression Model
pca_final = PCA(n_components=10)
X_train_pca = pca_final.fit_transform(X_train)
X_test_pca = pca_final.transform(X_test)

lr_pca = LinearRegression()
lr_pca.fit(X_train_pca, y_train)
```

To evaluate the trained model's accuracy, we obtain the R² and MAE values. These values shown in Figure 5.6.1.



```
Comprehensive Evaluation of Linear Regression Model with PCA

Test Set Performance Metrics
R-squared (R²): 0.6725
Mean Absolute Error (MAE): 718,114.30
```

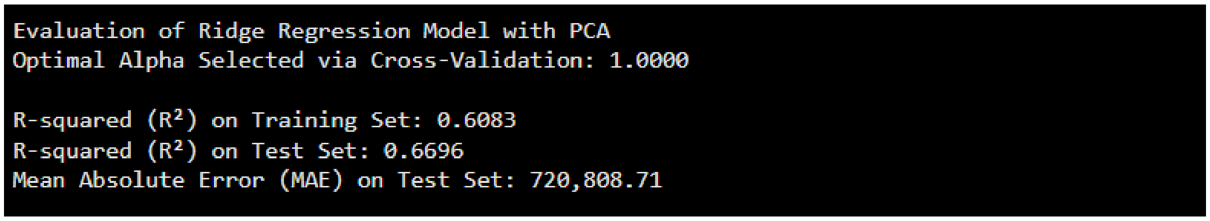
Fig. 5.6.1. Multiple linear regression trained model metrics

2. Ridge Regression

A Ridge Regression model was trained and evaluated using cross-validation (RidgeCV) to find the optimal α parameter on the PCA-transformed data.

```
ridge_pca = RidgeCV(alphas=alphas, cv=5, scoring='neg_mean_absolute_error')

# Train the Ridge Regression model on PCA-transformed data
ridge_pca.fit(X_train_pca, y_train)

y_pred_train_ridge_pca = ridge_pca.predict(X_train_pca)
r2_train_ridge_pca = r2_score(y_train, y_pred_train_ridge_pca)
```

The values calculated using this model are shown in Figure 5.6.2.



```
Evaluation of Ridge Regression Model with PCA
Optimal Alpha Selected via Cross-Validation: 1.0000

R-squared (R²) on Training Set: 0.6083
R-squared (R²) on Test Set: 0.6696
Mean Absolute Error (MAE) on Test Set: 720,808.71
```

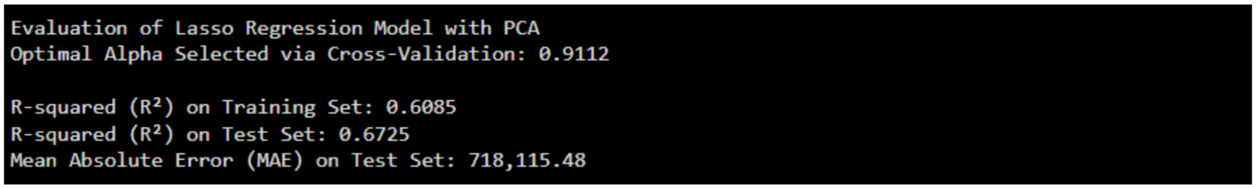
Fig. 5.6.2. Ridge regression trained model metrics

3. Lasso Regression

A Lasso Regression model was trained and evaluated using cross-validation (LassoCV) to find the optimal α parameter on the PCA-transformed data.

```
alphas = np.logspace(-4, 0, 100)

# LassoCV automatically selects the optimal alpha using cross-validation.
lasso_pca = LassoCV(alphas=alphas, cv=5, random_state=100, max_iter=10000)

lasso_pca.fit(X_train_pca, y_train)
```

The values calculated using this model are shown in Figure 5.6.3.



```
Evaluation of Lasso Regression Model with PCA
Optimal Alpha Selected via Cross-Validation: 0.9112

R-squared (R²) on Training Set: 0.6085
R-squared (R²) on Test Set: 0.6725
Mean Absolute Error (MAE) on Test Set: 718,115.48
```

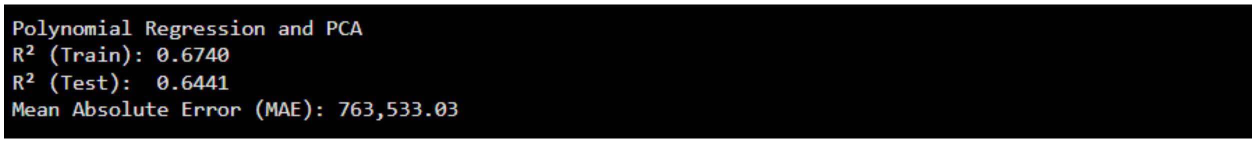
Fig. 5.6.3. Lasso regression trained model metrics

4. Polynomial Regression

Polynomial Regression is a type of regression that models the relationship between the independent variable X and the dependent variable Y as an n*th* degree polynomial, rather than a straight line. This method allows the model to learn non-linear relationships between variables, while still belonging to the family of linear models.

```
poly = PolynomialFeatures(degree=2, include_bias=False)
X_train_poly = poly.fit_transform(X_train)
X_test_poly = poly.transform(X_test)
```

The values calculated using this model are shown in Figure 5.6.4.



```
Polynomial Regression and PCA
R² (Train): 0.6740
R² (Test):  0.6441
Mean Absolute Error (MAE): 763,533.03
```

Fig. 5.6.4. Polynomial regression trained model metrics

5. MLP

The designed network consists of four hidden layers with a decreasing number of neurons. This structure is relatively deep and allows the network to hierarchically extract complex, non-linear features.

So, we have: `hidden_layer_size` = (256, 128, 64,32)

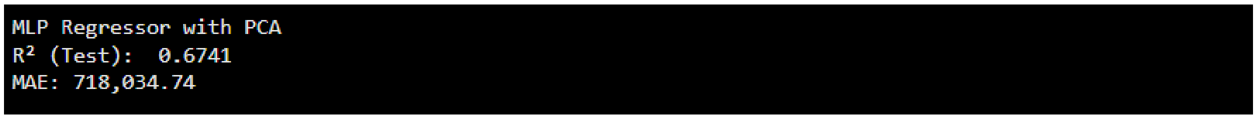
Activation Function maps output values to the range of [-1, 1]. The use of `tanh` was common in early neural networks and helps the model capture non-linear relationships.

The `adam` solver is one of the most efficient and popular gradient-based optimization algorithms. It works effectively with large datasets and deep networks, and it has been used in the design of this MLP.

Selecting a very small learning rate is key to the stability of these deep models. This prevents large jumps in the error space and ensures stable convergence and here we have 10^{-4} learning rate.

```
mlp_pca = MLPRegressor(
    hidden_layer_sizes=(256, 128, 64, 32),
    activation='tanh',
    solver='adam',
    learning_rate_init=1e-4,
    max_iter=5000,
    early_stopping=True,
    validation_fraction=0.15,
    random_state=42,
    verbose=False
)
```

The values calculated using this model are shown in Figure 5.6.5.



```
MLP Regressor with PCA
R² (Test):  0.6741
MAE: 718,034.74
```

Fig. 5.6.5.MLP trained model metrics

6. Bonus Section

Elastic-Net is a linear regression model that combines both Lasso (L1) and Ridge (L2) regularization techniques. Its primary goal is to overcome the limitations of each method when used individually, particularly in cases where severe multicollinearity is present.

Regression models work by adding a penalty to the OLS (Ordinary Least Squares) loss function. The Elastic-Net loss function is defined as follows:

$$Loss_{Elastic-Net} = Loss_{OLS} + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_2 \sum_{j=1}^{p} \beta_j^2$$

The first term pulls the coefficients toward zero and can zero out some less important coefficients, this term corresponds to Lasso.

The second term keeps the coefficients small but rarely drives them to zero, this term corresponds to Ridge.

Elastic-Net Parameters:

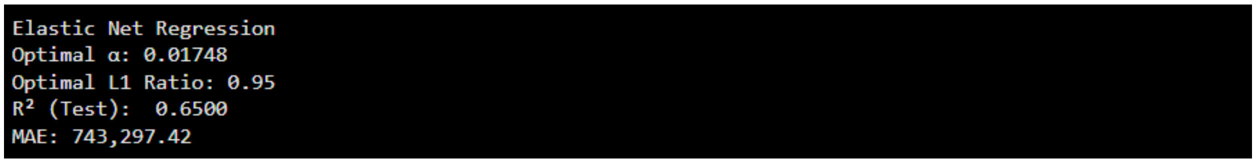
- $\alpha = \frac{\lambda_1}{\lambda_1+\lambda_2}$
- $\lambda = \lambda_1 + \lambda_2$

```
elastic = ElasticNetCV(
    l1_ratio=np.linspace(0.05, 0.95, 20),
    alphas=np.logspace(-4, 2, 100),
    cv=10,
    max_iter=20000,
    random_state=42

)
```

`L1_ratio` is the ratio between L1 and L2 regularization that means how much Lasso is used compared to Ridge.

`alphas` is a regularization strength.

The values calculated using this model are shown in Figure 5.6.6.



```
Elastic Net Regression
Optimal α: 0.01748
Optimal L1 Ratio: 0.95
R² (Test):  0.6500
MAE: 743,297.42
```

Fig. 5.6.6. Elastic-Net regression trained model metrics

7. Conclusion

The table below shows a summary of the performance of all models.

Model	R²	MAE	Description
Linear Regression	0.6725	718,114	Baseline linear model with strong and stable performance
Lasso Regression	0.6405	718,115	Automatically adjusts coefficients and removes less important features
Ridge Regression	0.6390	720,809	Prevents overfitting, but slightly less accurate
Polynomial Regression	0.6590	763,533	Increased model complexity, slight overfitting
MLP Regressor	0.6741	718,034	Achieves highest accuracy, but requires more tuning and computation time
Elastic Net Regression	0.6500	743,297	Balances Ridge and Lasso effects, but slightly weaker performance

The linear models (Linear, Lasso, Ridge) all show similar performance which means this indicates that the data are approximately linear, and simple models are quite sufficient.

Elastic Net strikes a good balance between L1 and L2 regularization, but it did not yield significant improvement on this particular dataset.

Polynomial Regression model showed the weakest performance among the successful models. This is due to the sudden increase in the number of features and the unnecessary attempt to model nonlinear relationships.

The MLP achieved the highest average R^2 and the lowest MAE. However, it is only slightly better than the Linear model which indicating that the more complex model did not provide significant added benefit.

Part VII: Using MLP with Feature Selector

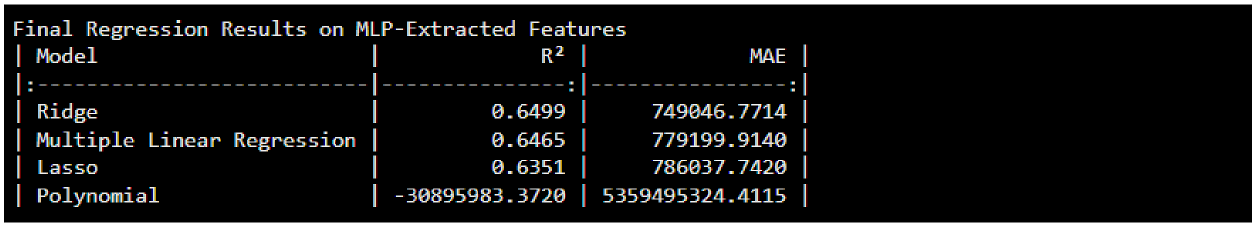
First, the original regression problem (predicting continuous price) was transformed into a four-class classification task. This forces the neural network (MLP) to learn decision boundaries in the feature space that are optimized for distinguishing between low, medium, and high prices.

The continuous target variable Price was discretized into 4 classes [Low, Medium-Low, Medium-High, High].

After training, the final output layer responsible for classification is removed. Instead, the output of the last hidden layer, which contains 32 new, high-level features, is taken. This output is used as input for the final model. 32 new, nonlinear features replaced the original 13 features.

```
# Create a 4-Class Target for MLP Training
bins = y_train.quantile([0.25, 0.5, 0.75]).tolist()
bins = [y_train.min()] + bins + [y_train.max()]
labels = ['Low', 'Medium-Low', 'Medium-High', 'High']
y_train_binned = pd.cut(y_train, bins=bins, labels=labels,
include_lowest=True, duplicates='drop')

le = LabelEncoder()
y_train_classified = le.fit_transform(y_train_binned.astype(str))
# Feature Extraction Function & MLP Training
def extract_features(mlp_model, X_data, layer_index=-2):
    current_output = X_data.values
    for i in range(len(mlp_model.coefs_) + layer_index):
        z = current_output @ mlp_model.coefs_[i] +
mlp_model.intercepts_[i]
        current_output = np.maximum(0, z)
    return current_output
```

The values obtained from the models using the extracted features are according to Figure 5.6.7.



```
Final Regression Results on MLP-Extracted Features
| Model                      |              R² |              MAE |
|:---------------------------|---------------:|----------------:|
| Ridge                      |          0.6499 |      749046.7714 |
| Multiple Linear Regression |          0.6465 |      779199.9140 |
| Lasso                      |          0.6351 |      786037.7420 |
| Polynomial                 | -30895983.3720 | 5359495324.4115 |
```

Figure 5.6.7. Metrics of models trained using extracted features

Ridge Regression has performed well with the 32 new features and it’s the best model in the extracted set.

Transforming 32 nonlinear features into over 560 polynomial features has completely destroyed the model numerically.

The feature extraction process, although it created good nonlinear features, was not successful for the ultimate goal of accurate price prediction. The MAE of the extracted models is more than 30,000 monetary units higher than the original linear model.

The main reason for this is task mismatch. The MLP features were optimized to distinguish between 4 classes, not to preserve the scale and continuous relationship of prices. The simple linear model, which was directly optimized for the regression objective, maintained better performance.