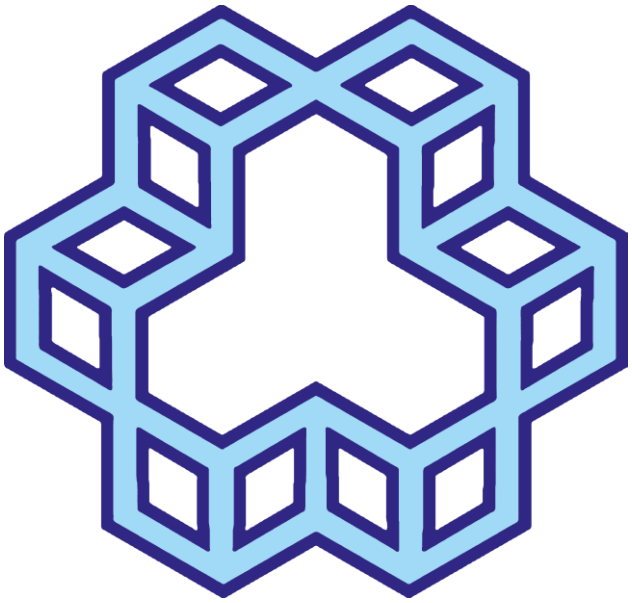
بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ



Students:
Ali Soltani_40119403
Hamidreza Eslami_40115563

Professor: $Dr. Aliyari$

Course: Fundamental of Intelligent Systems

Project Links:

Google Colab:
Question 4:
https://colab.research.google.com/drive/1sKBrFOUeLBIg8MG6rj6yrg708vOw1y0F?usp=sharing
Question 5:
https://colab.research.google.com/drive/1sq_ll-UMCxYOIZ1VZC7rg854ulF383C3?usp=sharing

$My\ Github$:

Date:1404/8/16

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.

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

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

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:

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

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

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

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

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

where $\eta$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.

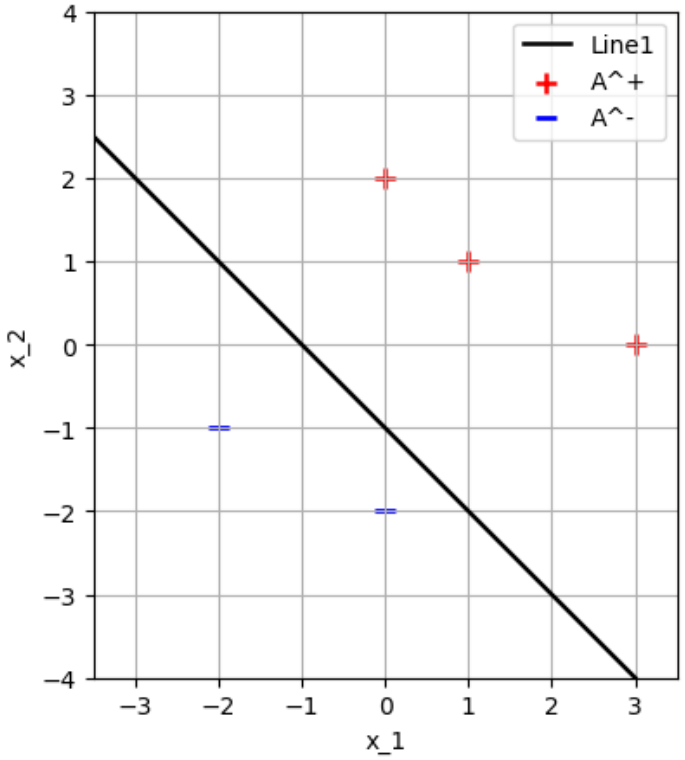


Fig. 2.1.1. Resulting Line curve of PLA

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:

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

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

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

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

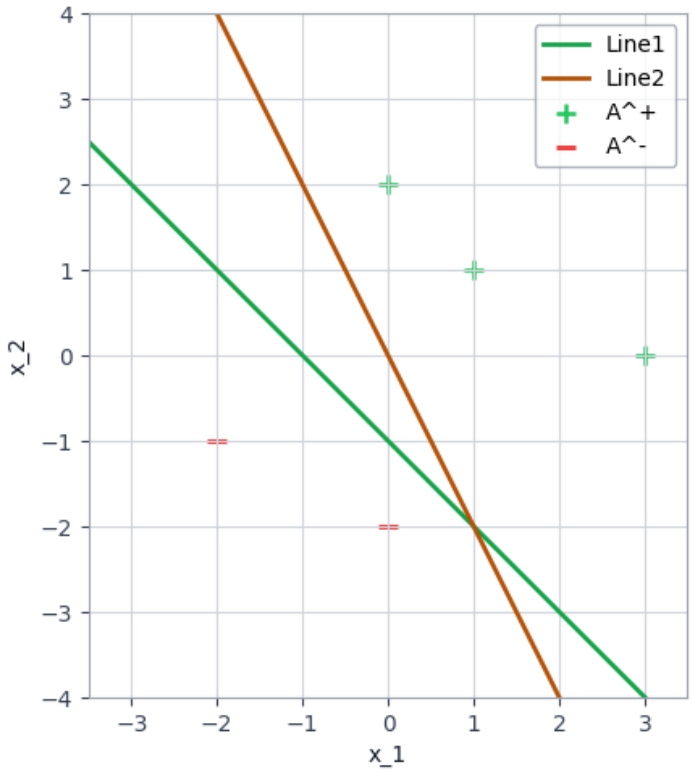


Fig. 2.2.1. Resulting Line curve of MSE

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:

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

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

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

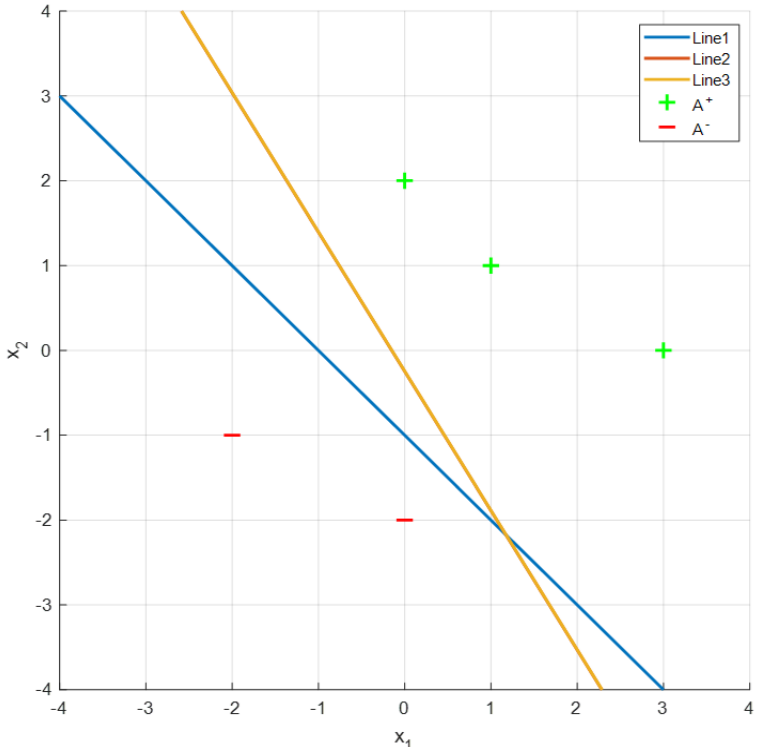


Fig. 2.3.1. Resulting Line curve of FLD

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 $x1$, 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:

$$\Sigma = \begin{bmatrix} \mathrm{Var}(x_1) & \mathrm{Cov}(x_1, x_2) \\ \mathrm{Cov}(x_1, x_2) & \mathrm{Var}(x_2) \end{bmatrix}$$

Given the data range specified in the problem, $Var\,(x_1) \gg Var\,(x_2)$, and typically $\mathrm{Cov}(x_1, x_2)$ is very small compared to $Var\,(x_1)$. Therefore, $\mathrm{Cov}(x_1, x_2)$can be approximately neglected in comparison to $Var(x1)$. Now, to find the principal components, we need to perform eigenvalue and eigenvector decomposition of the covariance matrix:

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

The matrix Σis 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)

Initially, the dataset supplied in the question is loaded using the df = pd.read_csv function. The head method is then employed to present the first five rows of the dataset, as depicted in Figure 4.1.1.

First five rows of the dataset:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus
0	13300000	7420	4	2	3	yes	no	no	no	yes	2	yes	furnished
1	12250000	8960	4	4	4	yes	no	no	no	yes	3	no	furnished
2	12250000	9960	3	2	2	yes	no	yes	no	no	2	yes	semi-furnished
3	12215000	7500	4	2	2	yes	no	yes	no	yes	3	yes	furnished
4	11410000	7420	4	1	2	yes	yes	yes	no	yes	2	no	furnished

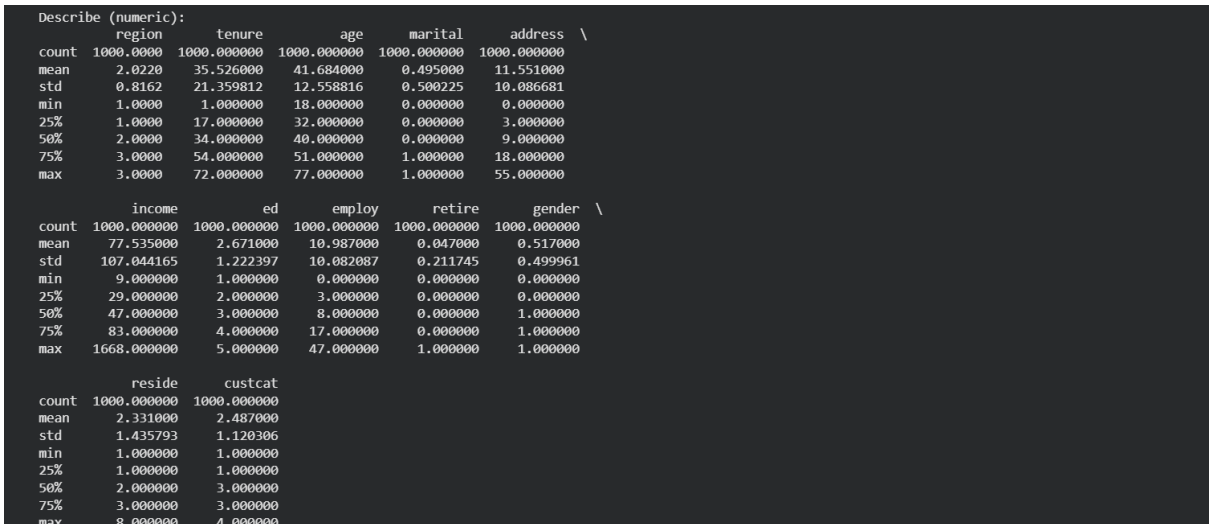
Fig. 4.1.1. dataset information

Now, we use the info method to obtain the overall structure of the data, as shown in the figure. 4.1.2.

```
Shape: (1000, 12)

Info:
<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
None
```

Fig. 4.1.2. structure of dataset

From the output of the info method, we observe that the dataset contains 1000 samples, each with 1000 non-null values, indicating there are no missing (NaN) values. Furthermore, based on the outputs of the head and info methods, all features and the target variable are numerical. The features tenure, age, address, and income are continuous, while the remaining ones are categorical. The statistical summary of the data obtained using the describe method is presented in Figure 4.1.3.



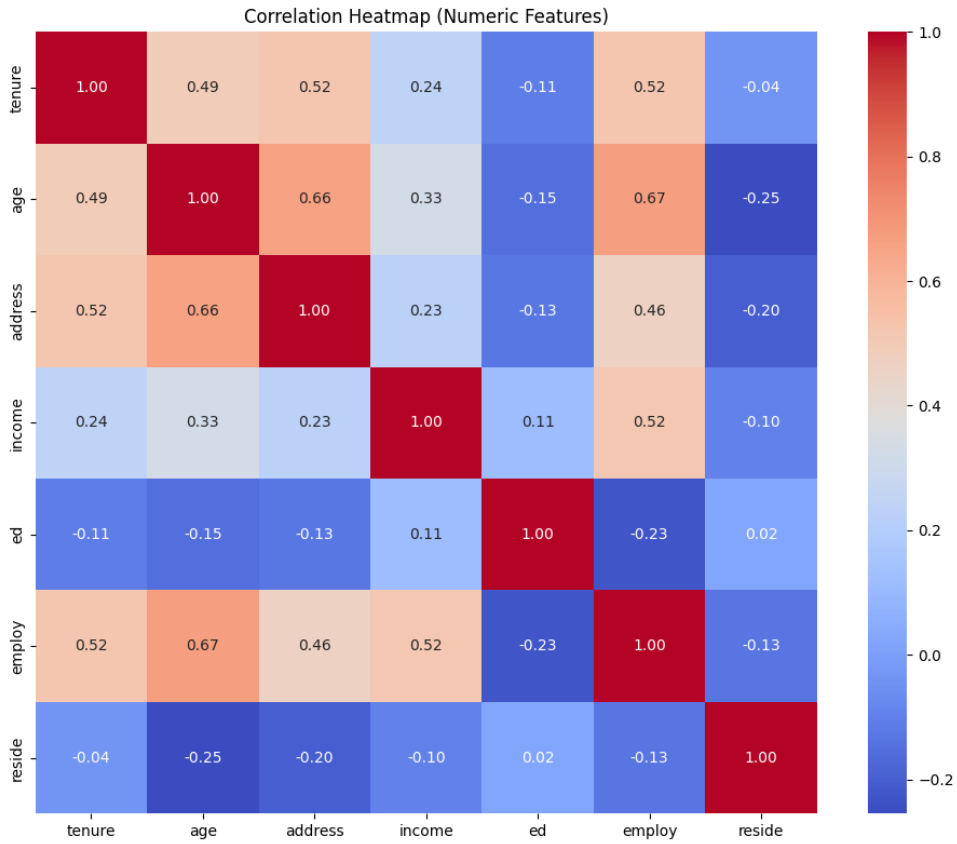
```
Describe (numeric):
          region       tenure          age      marital      address  \
count  1000.0000  1000.000000  1000.000000  1000.000000  1000.000000
mean      2.0220    35.526000    41.684000     0.495000    11.551000
std       0.8162    21.359812    12.558816     0.500225    10.086681
min       1.0000     1.000000    18.000000     0.000000     0.000000
25%       1.0000    17.000000    32.000000     0.000000     3.000000
50%       2.0000    34.000000    40.000000     0.000000     9.000000
75%       3.0000    54.000000    51.000000     1.000000    18.000000
max       3.0000    72.000000    77.000000     1.000000    55.000000

            income           ed       employ       retire       gender  \
count  1000.000000  1000.000000  1000.000000  1000.000000  1000.000000
mean     77.535000     2.671000    10.987000     0.047000     0.517000
std     107.044165     1.222397    10.082087     0.211745     0.499961
min       9.000000     1.000000     0.000000     0.000000     0.000000
25%      29.000000     2.000000     3.000000     0.000000     0.000000
50%      47.000000     3.000000     8.000000     0.000000     1.000000
75%      83.000000     4.000000    17.000000     0.000000     1.000000
max    1668.000000     5.000000    47.000000     1.000000     1.000000

            reside      custcat
count  1000.000000  1000.000000
mean      2.331000     2.487000
std       1.435793     1.120306
min       1.000000     1.000000
25%       1.000000     1.000000
50%       2.000000     3.000000
75%       3.000000     3.000000
max       8.000000     4.000000
```

Fig. 4.1.3. Statical summary of data

count: Number of non-null observations for each feature (all have 1000, indicating no missing values).
mean: The average value for each feature (e.g., tenure has a mean of 35.53).
std: Standard deviation, measuring the spread of each feature's values.
min/max: The smallest and largest values observed in each feature.
25%, 50%, 75%: Represent the 25th percentile (lower quartile), 50th percentile (median), and 75th percentile (upper quartile) of the feature values

Check for Missing Values
Missing values per column:
region 0
tenure 0
age 0
marital 0
address 0
income 0
ed 0
employ 0
retire 0
gender 0
reside 0
custcat 0

Data Visualization
Now, we plot the correlation between the data using a Heatmap. This plot is in figure 4.1.4.



4.1.4. Correlation Heatmap

According to Figure 4.1.4, where each cell represents the correlation coefficient between two variables using color and numeric annotation: redder tones indicate stronger positive relationships, blue tones represent negative relationships, and lighter shades mean weaker or no correlation; the highest correlations, such as between age and employ (0.67), reveal which feature pairs tend to move together, while near-zero or negative values, like between age and reside (-0.25), suggest independence or inverse relationships, making the heatmap a helpful tool to quickly visualize and interpret patterns, dependencies, and potential collinearity for further data analysis.

Using the pairplot function, we created scatter plots of the specified features while accounting for their corresponding classes. The resulting plot is presented in section 4.1.5.

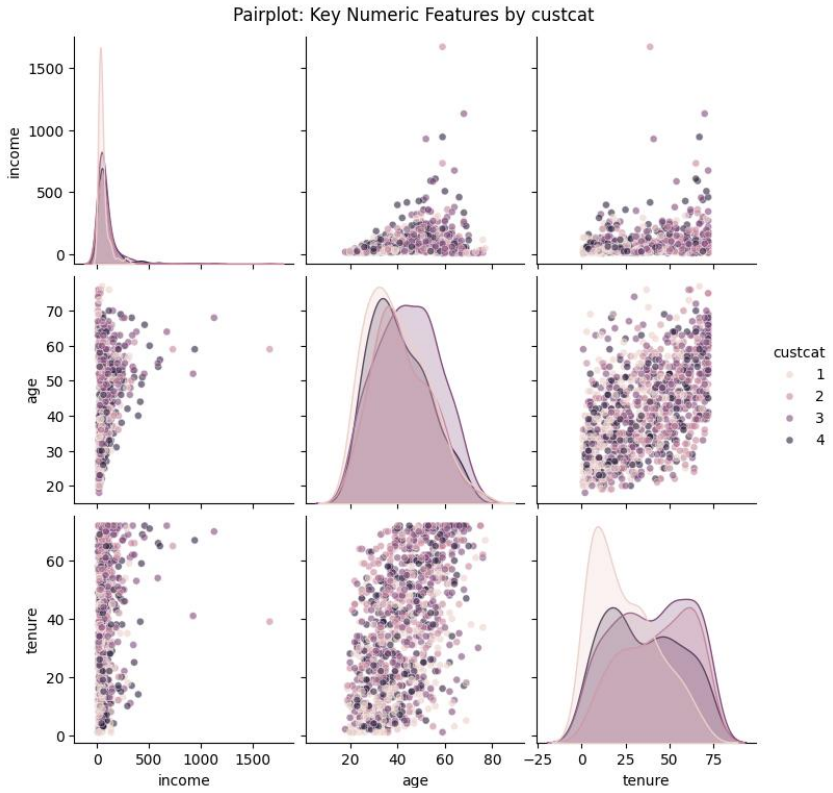
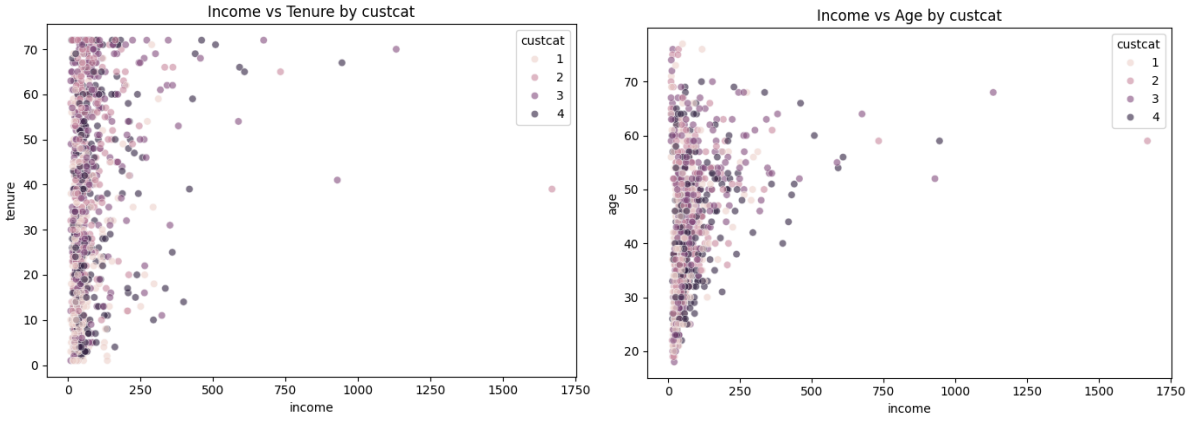


Fig. 4.1.5. Scatter plot for important features

Now we will give a brief explanation of two important points from these diagrams.



4.1.7.

Income Distribution:

Most of the data points are concentrated at lower income levels (below 250), indicating that most customers earn relatively low incomes.

Only a few customers have very high incomes (above 500).

Age Distribution:
Customers are spread across all age groups, but there's a slightly higher density between ages 30 and 60.
custcat Patterns:
All four categories appear to be present across the income and age range, but none seem to dominate a specific region strongly — meaning the categories are somewhat mixed.
Outliers:
A few customers with very high incomes (over 1000) stand out, but they are relatively rare.

Then, we plot the hexbin chart for the tenure and income features. This plot is shown in 4.1.7.

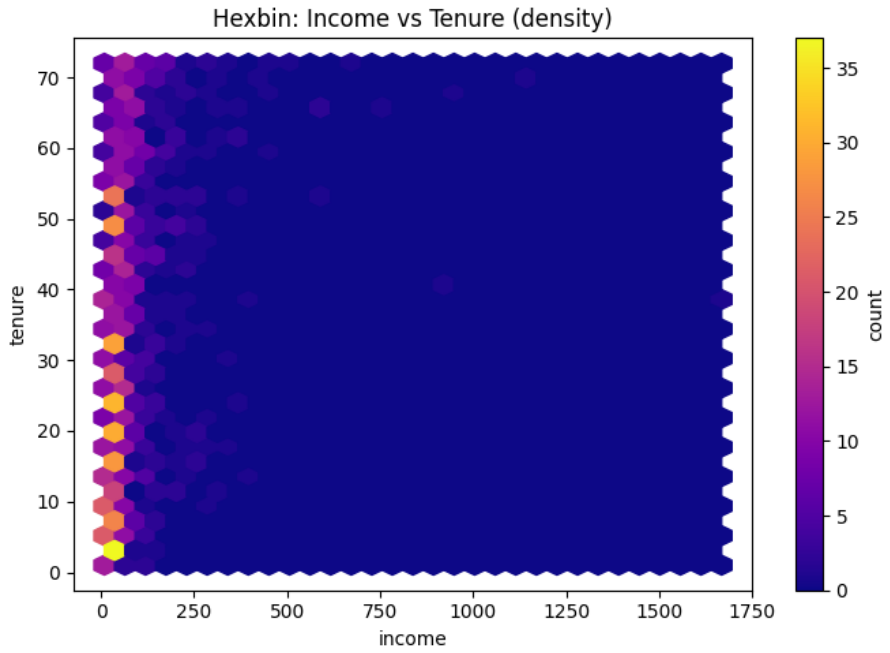


Fig 4.1.7. Hexbin plot for tenure and income feature

In Figure 4.1.8 presents the class distribution using a count plot, while Figure 4.1.9 illustrates the class distribution with a pie chart.

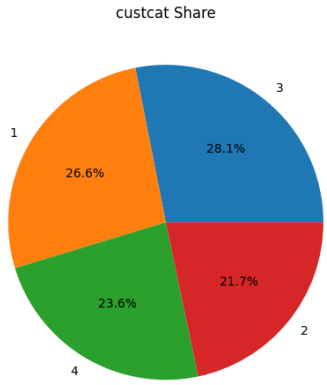


Fig. 4.1.8. Distribution plot by countplot

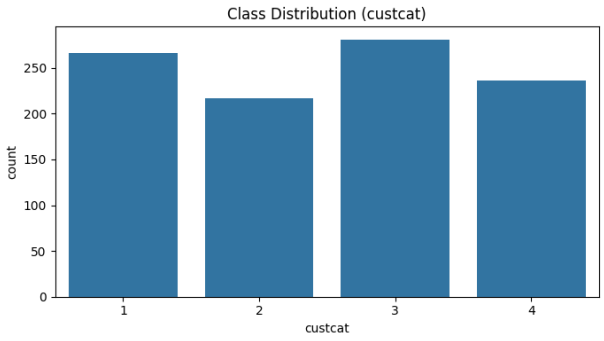


Fig. 4.1.9. Distribution plot by pie chart

The class distribution is fairly balanced. Class 3 holds the largest portion of the data at 28.1 percent, while Class 1 has the smallest at 21.7 percent. The difference between them is about 7 percent, which is relatively small.

4.2. Data Pre-Processing

First, the features are separated from the dataset and stored in the variable X, while the target values are stored in the variable y. Since the class labels in the dataset begin at 1 but need to start at 0 for model training, one is subtracted from the target values.

Next, preprocessing is applied to the categorical features. This can be done in two main ways:

1. **Label Encoding:** Each category is assigned a unique numerical value. This method is appropriate when the categories have a natural order. However, using it for unordered categories may lead the model to incorrectly infer relationships among them.
2. **One-Hot Encoding:** A separate binary column is created for each category, taking the value 1 when a sample belongs to that category and 0 otherwise. This approach is suitable for unordered categorical variables. Although it increases data dimensionality, multicollinearity can be reduced by removing one of the resulting columns.

In this dataset, One-Hot Encoding is chosen as the most suitable approach for transforming the categorical features. It is implemented using the get_dummies function for the region, reside, and ed features, while Label Encoding is applied to the remaining categorical features.

The remaining features are categorical. Since the dataset is already numerical, label encoding is unnecessary. Next, the data is randomly divided into training and testing sets. Standardization and normalization are then applied to ensure that numerical features share the same scale, allowing the models to process them appropriately.

- Normalization method for numerical features: The following formula is used to restrict the data range to the interval :

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

where $Xmin$ and $Xmax$ are the minimum and maximum values for the training data, respectively.

Standardization: By applying the following formula, the data is converted into 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, two methods are applied: Lasso regression and RFE.

- Using Lasso Regression: A Logistic Regression model with the L1 penalty is utilized for this process. The L1 penalty serves as an effective regularization method by driving the coefficients of features with minimal predictive impact to zero. Consequently, after training the model on the dataset, features with zero coefficients are removed. The output shown in Figure 4.3.1 illustrates 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')
```

As shown in the output above, no features were removed, and all of them were selected.

- Using RFE: Recursive Feature Elimination (RFE) is a model-based technique that removes features step by step. Its process works as follows:

1. **Base Model Training:** A Logistic Regression model is first trained using all available features.
2. **Importance Evaluation:** The significance of each feature, often determined by its coefficient or importance score, is assessed in the trained model.
3. **Removing the Least Important Features:** The features with the lowest importance values are removed.
4. **Iteration:** These steps are repeated until the target number of features remains.

Figure 4.3.2 illustrates which features were finally selected by the model.

So:

RFE-selected features: ['tenure', 'age', 'income', 'ed', 'employ', 'reside', 'region_1.0', 'region_2.0', 'region_3.0', 'marital_0.0', 'marital_1.0', 'retire_0.0', 'retire_1.0', 'gender_0.0', 'gender_1.0']

RFE-selected count: 15

As observed in the output above, RFE effectively removed several features from the dataset. We then used the features selected by RFE to train the Logistic Regression model. After fitting the model on the training data, we evaluated its performance. The accuracy of the model for both the training and testing datasets is presented in Figure 4.3.3.

```
Accuracy of test: 0.38666666666666666
```

Fig. 4.3.3. Model accuracy

We also plot the ROC curves for the training and testing datasets separately using the One-vs-Rest approach. Figure 4.3.6 presents these plots with the corresponding AUC values for the training data, while Figure 4.3.7 displays the plots with the AUC values for the test data.

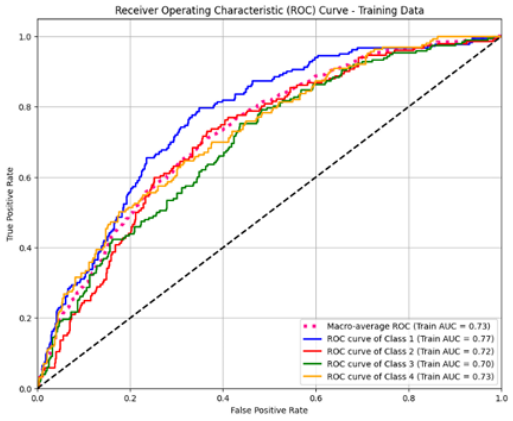


Fig. 4.3.6. ROC curve plot of training data

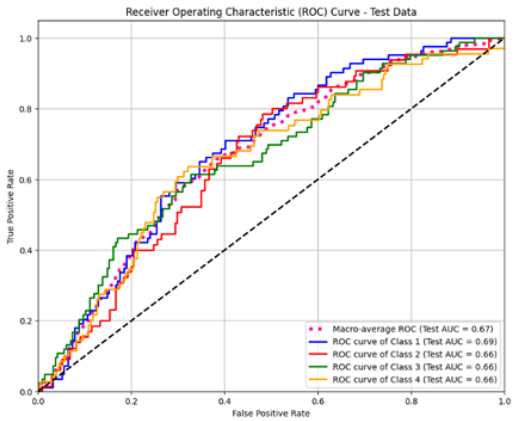


Fig. 4.3.7. ROC curve plot of test data

Figure 4.3.8 shows the model coefficients feature impact.

Top features by absolute coefficient magnitude:

feature	abs_coef_max
tenure	0.712731
ed	0.634393
retire_1.0	0.374525
retire_0.0	0.368903
region_1.0	0.286745
age	0.210436
income	0.202750
reside	0.195662
region_2.0	0.120634
gender_0.0	0.082406
gender_1.0	0.076784
employ	0.067369

Fig. 4.3.8. Model coefficient for each class

According to Figure 4.3.8, the tenure feature has the greatest influence on Class 1 and Class 2. As the value of tenure increases, the probability of belonging to Class 1 decreases, while the probability of belonging to Class 2 increases. Similarly, the ed_4 feature has the most significant effect on Class 3 and Class 4. When the value of ed_4 increases, the probability of belonging to Class 3 decreases, whereas the probability of belonging to Class 4 increases.

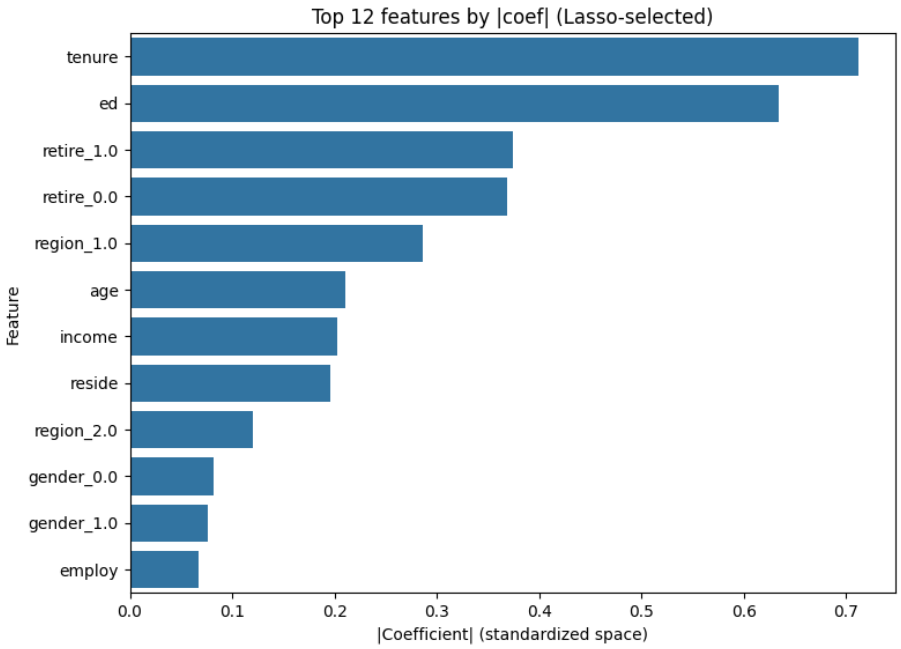


Fig. 4.3.9. Distribution of Coefficient analysis

Most influential features:

tenure has the highest coefficient (≈ 0.75), making it the most influential predictor.

ed (education) is the second most important, with a strong coefficient (~0.65).

retire_1.0 and **retire_0.0** follow, both with similar influence (~0.4).

Moderately important features:

region_1.0, **age**, **income**, and **reside** show moderate importance (coefficients between 0.2 and 0.3).

Less influential features:

region_2.0, **gender_0.0**, **gender_1.0**, and **employ** have the smallest coefficients (< 0.15), meaning they contribute the least to the model's predictions.

4.4. Feature Visualization Using Dimensionality Reduction

In this section, each model is trained on the training data to reduce the features to two dimensions. After feeding the training and testing samples into the model and obtaining the outputs, their scatter plots are presented.

- For PCA training, it is sufficient to provide only the feature data to the model.
- For LDA, which operates in a supervised manner, both the feature data and the target data must be provided to the model.

Figure 4.4.1 illustrates the plots of the features mapped by PCA and LDA.

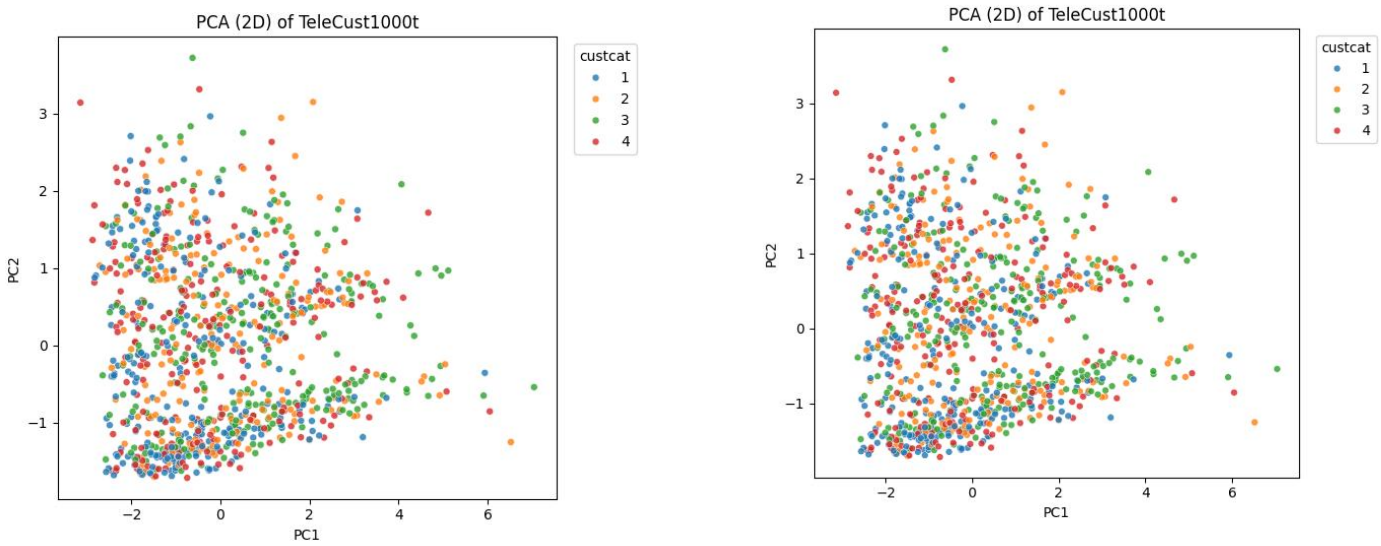


Fig. 4.4.1. Dimension reduction with PCA and LDA

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 after 300 epochs.

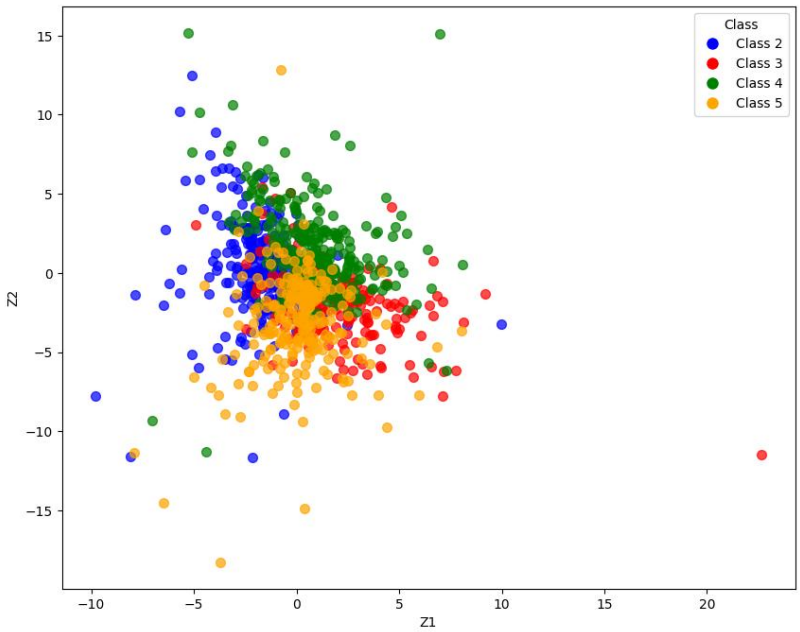


Fig. 4.4.2. Dimension reduction with PCA and LDA by MLP model

The PCA and LDA models failed to achieve separation because the features are not linearly separable. In contrast, the MLP model, which applies a nonlinear transformation to the data, nearly succeeded in achieving the separation.

Question 5 Part I: Literature Review

1. This article investigates housing price prediction utilizing the Housing Price dataset for Beijing.
    - **City of Study:** Beijing
    - **Initial Dataset Specifications:** The original dataset comprised over 300,000 records and 26 variables, capturing housing transactions from 2009 to 2018.
    - **Final Dataset Specifications:** Following pre-processing, the refined dataset consisted of 231,962 records with 19 features.
    - **Model Input:** After standardizing numerical values and applying One-Hot Encoding to categorical variables, the training dataset ultimately included 58 features.
2. The data pre-processing phase encompassed essential steps such as data cleaning, feature engineering, and preparation for modeling.
    - **Handling Missing Data:** Variables with more than 50% missing values were excluded, while any remaining records with missing values were also removed.
    - **Feature Engineering and Transformation:** The "construction Time" feature was replaced by an "age" variable. A new "distance" feature was created to represent proximity to the Beijing city center. The "floor" attribute was separated into "floorType" and "floorHeight." Ambiguous features, including those for kitchens, bathrooms, and living rooms, were removed. The number of living rooms (interpreted as bedrooms) was restricted to a range between 1 and 4.
    - **Outlier Removal:** Outliers were detected and eliminated using the Inter-Quartile Range (IQR) method.
    - **Final Preparation:** Numerical variables were standardized, categorical features were one-hot encoded, and the dataset was split into training and testing sets using an 80-20 ratio.
3. The study evaluated a range of traditional and advanced machine learning algorithms, using the Root Mean Squared Logarithmic Error (RMSLE) as the key performance metric. Methods assessed included Random Forest, XGBoost, LightGBM, Hybrid Regression, and Stacked Generalization Regression.

Part II: Dataset Description

1. **Sample and Feature Overview**
    - **Number of Samples:** 545
    - **Number of Features:** 13
2. **Feature Data Types**
    - The dataset comprises six numerical features of integer type (int64) and seven text features of object type (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 total count of unique values for each feature is presented below. This metric reflects the level of variability or diversity present within the values of 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 fundamental practice in data science that employs both visualizations and statistical techniques to summarize, describe, and interrogate a dataset. The central objective of EDA is to uncover the underlying structure of the data, reveal hidden patterns, identify outliers, and elucidate relationships between variables before engaging in advanced modeling tasks. As an initial and iterative stage in the analytic workflow, EDA typically comprises:

- Statistical Summaries: Computing measures such as mean, median, standard deviation, range, and quartiles to grasp the distribution of numerical features.

- Visualization Techniques: Constructing plots such as histograms, box plots, scatter plots, and bar charts to represent the distribution and interactions of variables visually.

- Preliminary Feature Engineering: Recognizing the necessity for data transformation or converting categorical features into numeric forms.
  EDA serves as an analytical foundation by enabling deeper data understanding, surfacing data quality issues, guiding the modeling approach, and validating core assumptions. It not only provides clarity about data characteristics, but also informs strategies for building robust and generalizable models. In this context, we proceed by using the gdown command to display the first five rows of the dataset; the corresponding output is presented in Figure 5.3.1.

First five rows of the dataset:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus
0	13300000	7420	4	2	3	yes	no	no	no	yes	2	yes	furnished
1	12250000	8960	4	4	4	yes	no	no	no	yes	3	no	furnished
2	12250000	9960	3	2	2	yes	no	yes	no	no	2	yes	semi-furnished
3	12215000	7500	4	2	2	yes	no	yes	no	yes	3	yes	furnished
4	11410000	7420	4	1	2	yes	yes	yes	no	yes	2	no	furnished

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

Based on the analysis of the Housing.csv dataset, the features were categorized into two distinct groups according to their data types: numerical features of type “int64” and categorical features of type “object.” The dataset comprises six numerical features—price, area, bedrooms, bathrooms, stories, and parking. Additionally, there are seven categorical features: main road, guest room, basement, hot water heating, air conditioning, prefarea, and furnishing status. The categorization of these features is illustrated in Figure 5.3.2.

```
Number of numerical features: 6
Numerical columns: ['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'parking']

Number of categorical features: 7
Categorical columns: ['mainroad', 'guestroom', 'basement', 'hotwaterheating', 'airconditioning', 'prefarea', 'furnishingstatus']
```

Fig. 5.3.2. Features Categorization

At this stage of Exploratory Data Analysis (EDA), count plots were utilized to visually examine the frequency distribution of values within four selected categorical features by:

```
# 6. Visualize selected categorical features
plt.figure(figsize=(15, 5))
for i, col in enumerate(categorical_features[:3]):  # visualize first 3 categorical features
    plt.subplot(1, 3, i + 1)
    sns.countplot(x=col, data=df, palette="coolwarm")
    plt.title(f"Count of {col}")
    plt.xticks(rotation=30)
plt.tight_layout()
plt.show()
```

was specified to organize these features for graphical analysis, facilitating a clearer understanding of the composition and prevalence of each category within the dataset. The sns.countplot chart is visible in Figure 5.3.3.

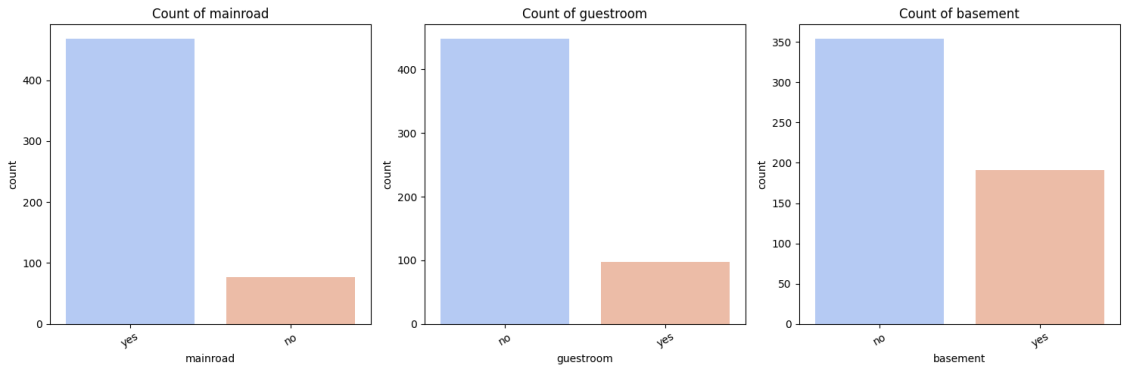


Fig. 5.3.3. sns.countplot of 4 features

The table below provides a summary of the predominant categories and a quantitative assessment of imbalance among the categorical variables. It is evident that a substantial disparity exists within most binary features, as one category markedly outweighs the other in frequency.

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

At this stage, the distributions of three principal numerical variables—price, area, and number of bedrooms—were examined, and their corresponding distribution plots are presented in Figure 5.3.4.

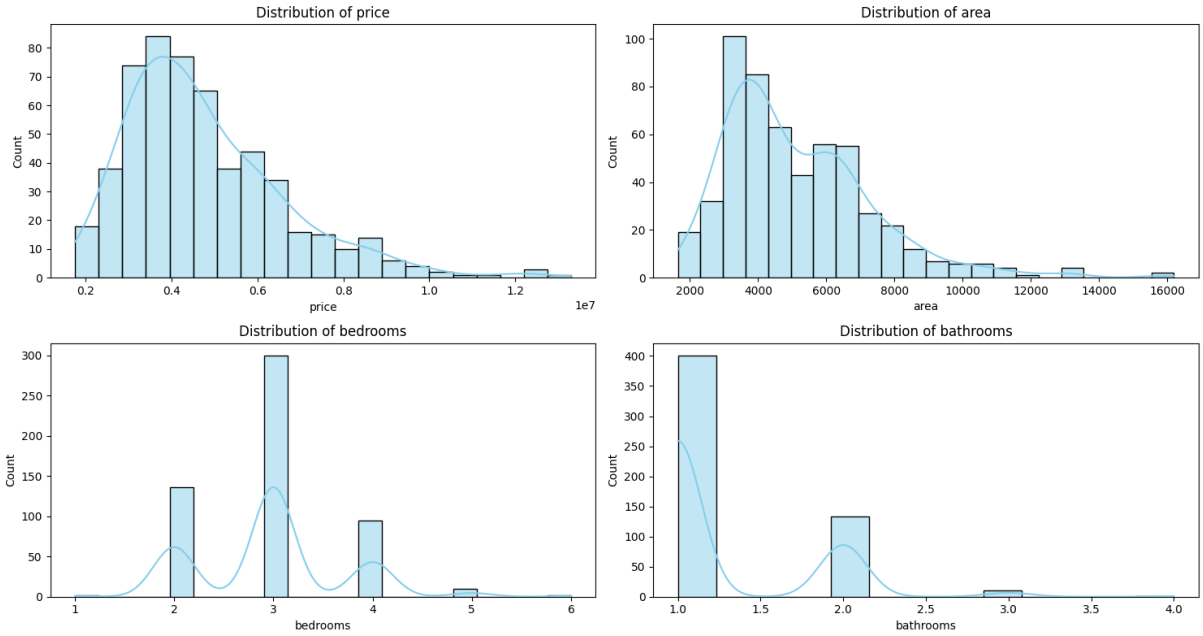


Fig. 5.3.4. sns.distplot of 3 features

1. Distribution of Price
    - **Distribution Shape:** The distribution of Price displays pronounced right-skewness, with the majority of house prices concentrated in the lower range.
    - **Outlier Presence:** A number of outliers are evident. The extended right tail of the distribution, comprising a small subset of properties priced significantly higher (e.g., those exceeding $10,000,000), demonstrates the existence of extreme price values.
2. Distribution of Area
    - **Distribution Shape:** Area also exhibits a strongly positively skewed distribution, such that most properties fall within a relatively small area range.
    - **Outlier Presence:** The occurrence of exceptionally large area measurements, which appear infrequently at the distribution's upper end, indicates the presence of outliers among the area values.
3. Distribution of Bedrooms
    - **Distribution Shape:** The number of bedrooms follows a discrete distribution, predominantly centered around three and four bedrooms, and shows much less skewness compared to Price and Area.
    - **Outlier Presence:** Despite being discrete with limited possible values, residences with an unusually high bedroom count (e.g., five or six) can be regarded as contextual outliers relative to the dataset's typical range.

Visualization of the relationships among the numerical features was performed using the pairplot command. The resulting plot is presented in Figure 5.3.5. The diagonals within the plot display the univariate histograms for each variable individually, once again illustrating the strong positive skew observed in the distributions for both price and area.

B. pairwise relationships between variables were examined using scatter plots.
The analysis reveals a positive and relatively strong association between price and area, such that increases in area are generally accompanied by increases in price; despite the linear nature of this relationship, considerable variance is observed.

- A positive correlation is also identified between price and the number of bathrooms, with properties containing more bathrooms tending to have higher prices; this pattern appears step-like as a result of the discrete values associated with bathroom counts.

- Furthermore, a positive and well-defined trend is present between price and the number of stories, indicating that houses with a greater number of stories typically command higher prices and reflecting the discrete nature of this variable.

- Lastly, relationships between area and other discrete features—including the number of bedrooms, bathrooms, and stories—are also positive, suggesting that larger homes generally contain a greater number of these attributes.

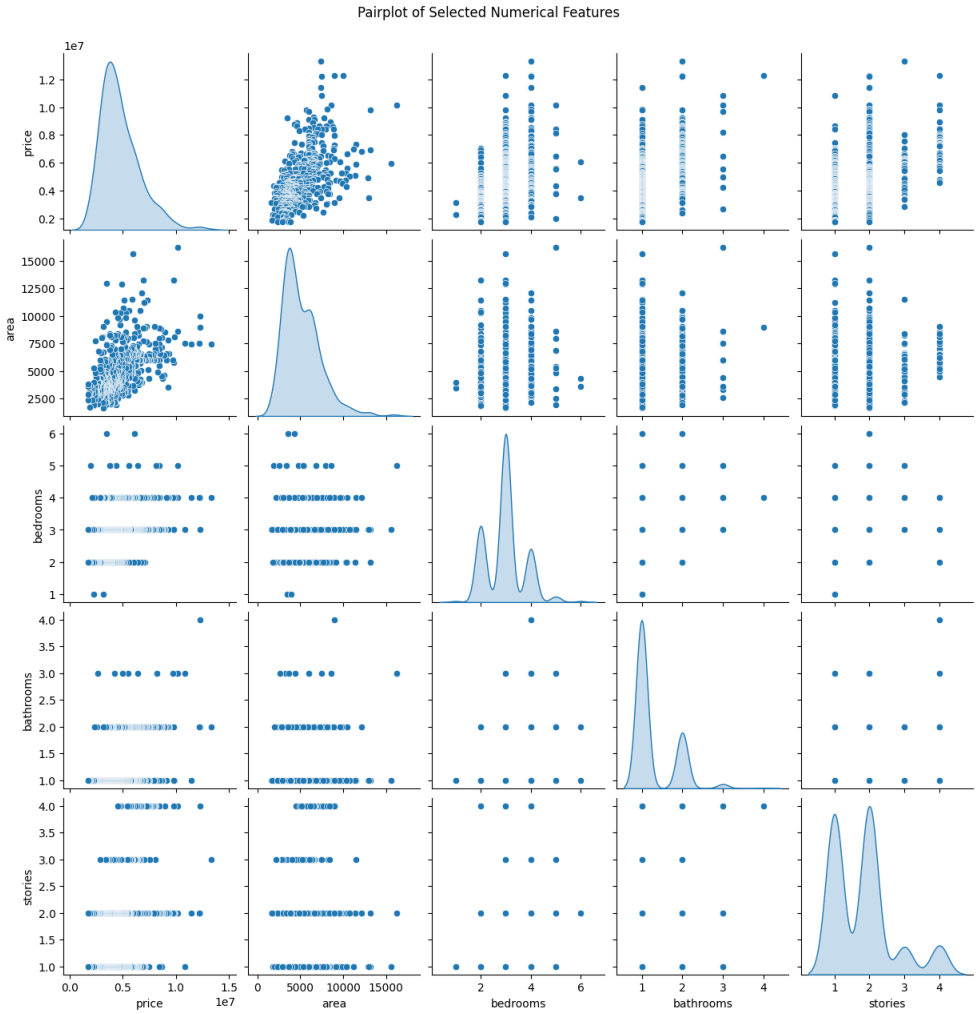


Fig. 5.3.5. sns.pairplot of numerical features

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: 0
After removing duplicates: (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.

```
print("\nMissing values per column:")
print(df.isnull().sum())
```

```
Missing values per column:
price             0
area              0
bedrooms          0
bathrooms         0
stories           0
mainroad          0
guestroom         0
basement          0
hotwaterheating   0
airconditioning   0
parking           0
prefarea          0
furnishingstatus  0
dtype: int64
```

```
for col in df.columns:
    if df[col].isnull().sum() > 0:
        if df[col].dtype in ['int64', 'float64']:
            df[col].fillna(df[col].median(), inplace=True)
        else:
            df[col].fillna(df[col].mode()[0], inplace=True)

print("\nMissing values after handling:")
print(df.isnull().sum())
```

```
Missing values after handling:
price             0
area              0
bedrooms          0
bathrooms         0
stories           0
mainroad          0
guestroom         0
basement          0
hotwaterheating   0
airconditioning   0
parking           0
prefarea          0
furnishingstatus  0
dtype: int64
```

Fig. 5.4.2. Replacing missing values

The analysis verified that the "Housing.csv" dataset is complete, with no missing values detected. Consequently, no imputation or data correction procedures were required, and the dataset preserved its initial structure of 545 rows and 13 features.

1. This outcome is advantageous, as introducing imputed values could potentially bias subsequent analyses.

Categorical feature encoding refers to the process of transforming non-numerical variables into numerical representations suitable for machine learning algorithms. The table below outlines several commonly utilized encoding techniques for categorical data.

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 Features:** These variables possess two possible states (yes/no). This scenario represents a specific case of label encoding, for which applying the mapping yes → 1 and no → 0 is the most straightforward and suitable approach. The features to which this method applies are: mainroad, guestroom, basement, hotwaterheating, airconditioning, and prefarea.
    - **Multi-Categorical Feature:** The furnishingstatus variable contains three distinct categories. While it could be considered ordinal, to avoid imposing potentially misleading ordinal assumptions on the model and to maintain simplicity, the One-Hot Encoding technique is employed. Moreover, with only three categories, this approach results in minimal dimensionality increase (two additional columns are created).

At this step, all categorical features in the dataset (comprising six binary variables and one three-level variable) were transformed into numerical representations to facilitate subsequent machine learning analysis.

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 widely used techniques for detecting and handling outliers in data analysis are as follows:

1. **Standard Deviation Method:** Data points that fall beyond three standard deviations (3σ) from the mean are classified as outliers and excluded. This approach is most effective for datasets that approximate a normal distribution.
2. **Interquartile Range (IQR) Method:** Outliers are identified as values lying outside the interval $[Q_1 - 1.5 \times IQR, Q_3 + 1.5 \times IQR]$, where Q_1 and Q_3 represent the first and third quartiles, respectively. This method is well suited to datasets with skewed distributions or those where a more robust strategy against outliers is needed.
3. **Logarithmic/Statistical Transformation:** Instead of discarding data, transformations such as taking the logarithm can reduce skewness, moving outliers closer to the central tendency and diminishing their influence on modeling. This technique is appropriate for numerical features with pronounced positive skew (as observed for price and area in the current dataset).

Given the strong positive skew in the key features (price and area), a combined strategy is warranted:

- Employ the IQR method to detect and eliminate extreme outliers.
- Apply logarithmic transformation to further mitigate data skewness without excessive data removal, thereby preserving information.

Accordingly, the IQR method will be used to filter extreme outliers in the price and area attributes. Output of the code is shown in figure 5.4.3.

```
df_encoded = df.copy()
df_encoded = pd.get_dummies(df_encoded, columns=categorical_features, drop_first=True)

print("\nData after encoding:")
display(df_encoded.head())
```

```
Data after encoding:
```

	price	area	bedrooms	bathrooms	stories	parking	mainroad_yes	guestroom_yes	basement_yes	hotwaterheating_yes	airconditioning_yes	prefarea_yes	furnishingstatus_s furni
0	13300000	7420	4	2	3	2	True	False	False	False	True	True	F
1	12250000	8960	4	4	4	3	True	False	False	False	True	False	F
2	12250000	9960	3	2	2	2	True	False	True	False	False	True	
3	12215000	7500	4	2	2	3	True	False	True	False	True	True	F
4	11410000	7420	4	1	2	2	True	True	True	False	True	False	F

Fig. 5.4.3. Dataset information after outlier cleaning

- Initial number of rows: 545
- Number of removed rows (outliers): 28
- Final dataset shape: (517, 14)

Binary features (basement, mainroad, guestroom, hotwaterheating, airconditioning, prefarea) were encoded as 0 and 1.

The multi-category feature (furnishingstatus) was transformed using one-hot encoding into two new columns.
The dimensionality of the dataset increased from 13 to 14.
The cleaned dataset (after the removal of outliers and encoding of categorical features) was split into training (70%) and testing (30%) subsets.
All numerical features and the target variable were normalized using MinMaxScaler, mapping values to the range, which is essential for optimal performance of gradient-based models.

Part V: Feature Selection

The correlation matrix of all features in the cleaned and encoded dataset was calculated and visualized, facilitating the identification of linear relationships between variables, as shown in Figure 5.5.1.

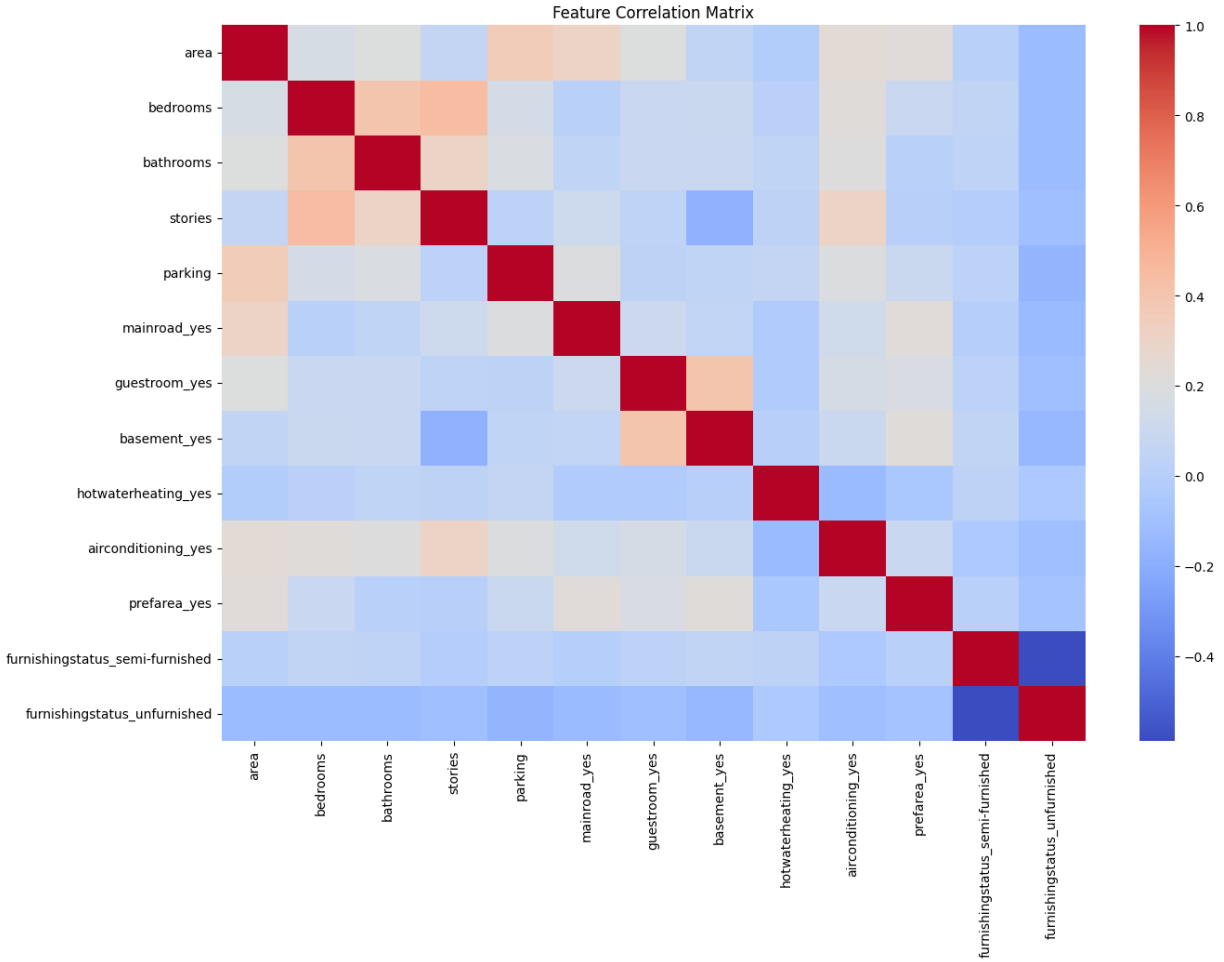


Fig. 5.5.1. Corraltion matrix of features

The correlation matrix reveals that the features with the strongest positive association to price are as follows:

1. Area, which serves as the most influential predictor of house price, as larger homes tend to command higher values;
2. Air Conditioning, which exerts a notably strong effect on price, suggesting that this amenity is highly prized in the housing market;
3. Bathrooms, which have a substantial impact on property value, reflecting their importance for comfort and functionality.

These three variables will be utilized as the principal predictors in the initial modeling phase, with the potential addition of other moderately correlated features to facilitate comparative performance analysis.
Moderate positive correlations are also evident among bathrooms, stories, and air conditioning, indicating that contemporary properties typically include these amenities collectively.
The relationship between semi-furnished and unfurnished furnishing statuses is mutually exclusive, with a correlation coefficient of -0.60.
Principal Component Analysis (PCA) was subsequently applied to the training feature set to reduce dimensionality while retaining maximum information (variance), and the resulting cumulative explained variance plot is depicted in Figure 5.5.2.

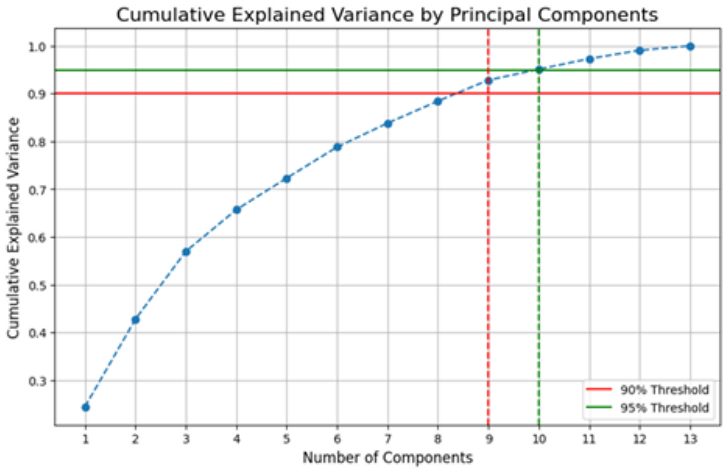


Fig. 5.5.2. CEV plot of PCA

The primary objective of applying PCA for dimensionality reduction is to eliminate noise, as components exhibiting low variance predominantly capture noise rather than meaningful structure. Additionally, this process decreases model complexity, which in turn shortens training time and enhances interpretability for complex models. Given the initial presence of 13 features, the selection of 10 principal components enables the retention of approximately 95% of the original dataset's variance, while excluding 3 dimensions. This approach effectively balances the preservation of relevant information with the reduction of computational demands.

Part V: Bonus section

Multicollinearity refers to a condition where two or more independent variables in a regression model are highly correlated with each other. This issue complicates the interpretation of the model's coefficients, as it becomes difficult for the model to distinguish the individual impact of each feature on the target variable.

Multicollinearity occurs when two or more independent variables in a regression model exhibit high correlation, making it difficult for the model to disentangle and estimate the unique contribution of each feature to the target variable; as a result, the reliability of coefficient interpretation is compromised.

The Variance Inflation Factor (VIF) quantifies the extent to which the variance of a regression coefficient is inflated due to multicollinearity, and is computed for each feature by regressing that feature against all other features within the model.

$$VIF_i = \frac{1}{1 - R_i^2}$$

A feature with $VIF > 5$or $VIF > 10$typically indicates problematic multicollinearity. 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.

```
vif_data = pd.DataFrame()
vif_data["Feature"] = X_scaled.columns
vif_data["VIF"] = [variance_inflation_factor(X_scaled.values, i) for i in range(X_scaled.shape[1])]

print("\nVIF values for features:")
display(vif_data.sort_values(by="VIF", ascending=False))
```

```
VIF values for features:
```

	Feature	VIF
12	furnishingstatus_unfurnished	1.695279
11	furnishingstatus_semi-furnished	1.587195
3	stories	1.547346
1	bedrooms	1.468454
7	basement_yes	1.378603
0	area	1.366038
2	bathrooms	1.285334
6	guestroom_yes	1.272195
9	airconditioning_yes	1.265309
4	parking	1.231844
5	mainroad_yes	1.191841
10	prefarea_yes	1.150189
8	hotwaterheating_yes	1.041776

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. RFE: 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.

Optional 2

```
model = LinearRegression()
rfe = RFE(estimator=model, n_features_to_select=10)  # You can adjust number as needed
rfe.fit(X_scaled, y_train)

selected_features = X_train.columns[rfe.support_]
print("\nTop 10 selected features using RFE:")
print(selected_features.tolist())
```

```
Top 10 selected features using RFE:
['area', 'bathrooms', 'stories', 'parking', 'mainroad_yes', 'basement_yes', 'hotwaterheating_yes', 'airconditioning_yes', 'prefarea_yes', 'furnishingstatus_unfurnished']
```

```
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.

```
lr_model = LinearRegression()
lr_model.fit(X_train_scaled, y_train)

results = []
results.append(evaluate_model(lr_model, X_train_scaled, X_test_scaled, y_train, y_test, "Multiple Linear Regression"))
```

```
📊 Multiple Linear Regression Performance:
MAE:  880919.5116
RMSE: 1134300.2555
R²:   0.6815
```

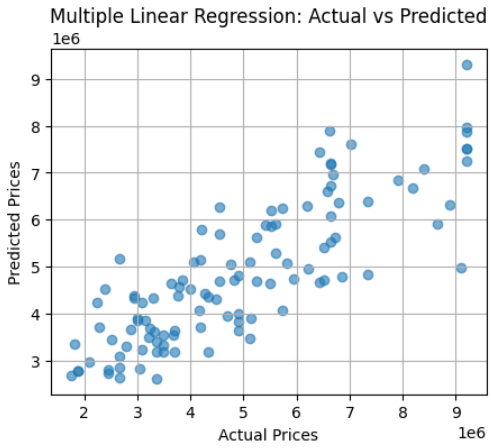


Fig. 5.6.1. Multiple linear regression trained model metrics and output

2. Ridge Regression:

A Ridge Regression model was trained and evaluated using cross-validation (RidgeCV) to find

2. Ridge Regression

```
ridge_model = Ridge(alpha=1.0)
ridge_model.fit(X_train_scaled, y_train)

results.append(evaluate_model(ridge_model, X_train_scaled, X_test_scaled, y_train, y_test, "Ridge Regression"))
```

```
📊 Ridge Regression Performance:
MAE:  880661.5574
RMSE: 1134317.1759
R²:   0.6815
```

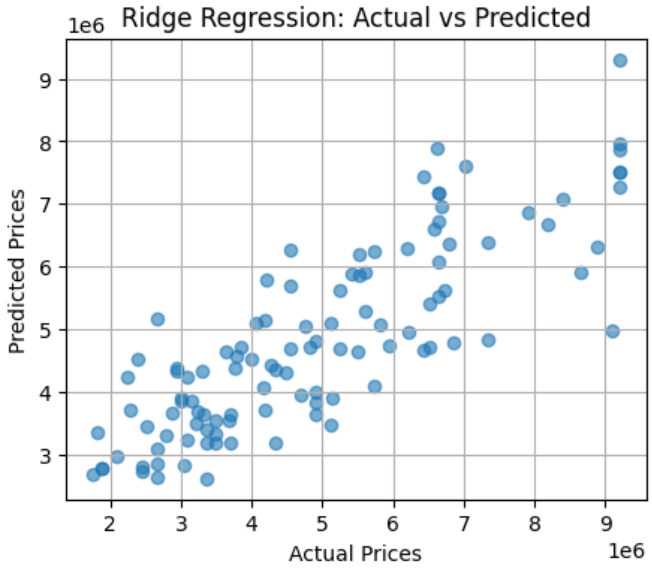


Fig. 5.6.2. Ridge regression trained model metrics and output

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.

3. Lasso Regression

```
lasso_model = Lasso(alpha=0.001, max_iter=10000)
lasso_model.fit(X_train_scaled, y_train)

results.append(evaluate_model(lasso_model, X_train_scaled, X_test_scaled, y_train, y_test, "Lasso Regression"))
```

```
📊 Lasso Regression Performance:
MAE:  880919.5117
RMSE: 1134300.2559
R²:   0.6815
```

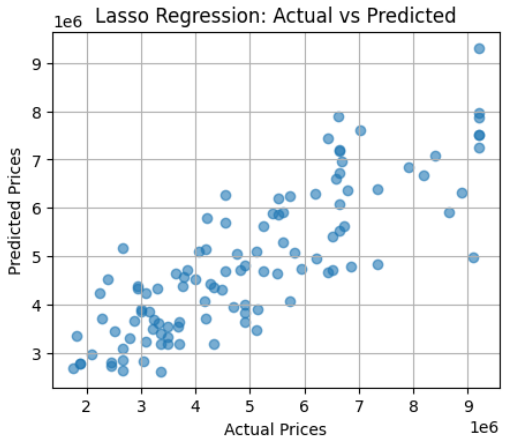
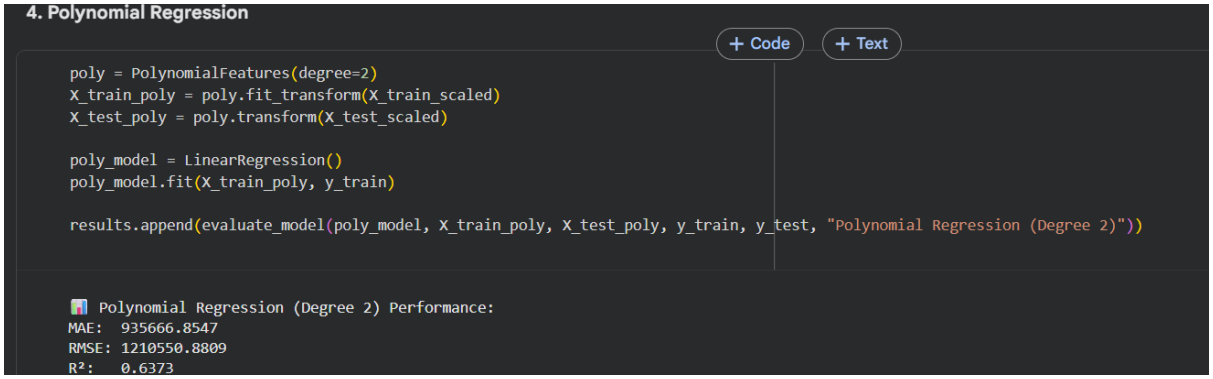


Fig. 5.6.3. Lasso regression trained model metrics and output

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 nth 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.



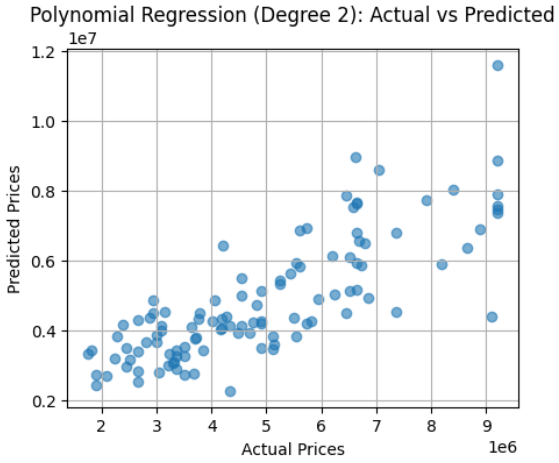


Fig. 5.6.4. Polynomial regression trained model metrics and output

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.

```
5. Multi-Layer Perceptron (MLP)

mlp_model = MLPRegressor(hidden_layer_sizes=(64, 32),
                         activation='relu',
                         solver='adam',
                         learning_rate='adaptive',
                         max_iter=500,
                         random_state=42)

mlp_model.fit(X_train_scaled, y_train)

results.append(evaluate_model(mlp_model, X_train_scaled, X_test_scaled, y_train, y_test, "MLP Regressor"))
```

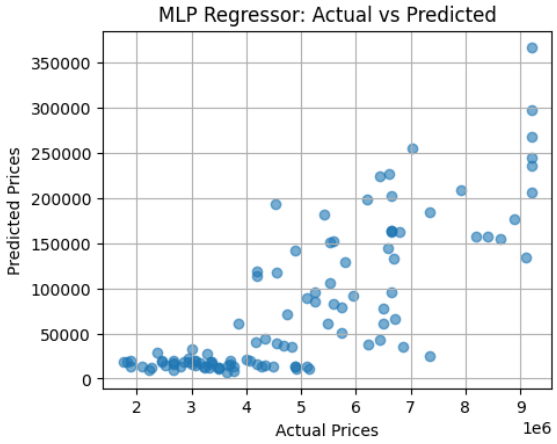


Fig. 5.6.5.MLP trained model metrics and output

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:

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

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:

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

(Optional) 6. Elastic-Net Regression

```
elastic_model = ElasticNet(alpha=0.001, l1_ratio=0.5, max_iter=10000)
elastic_model.fit(X_train_scaled, y_train)

results.append(evaluate_model(elastic_model, X_train_scaled, X_test_scaled, y_train, y_test, "Elastic-Net Regression"))
```

```
📊 Elastic-Net Regression Performance:
MAE:  880862.9935
RMSE: 1134303.3942
R²:   0.6815
```

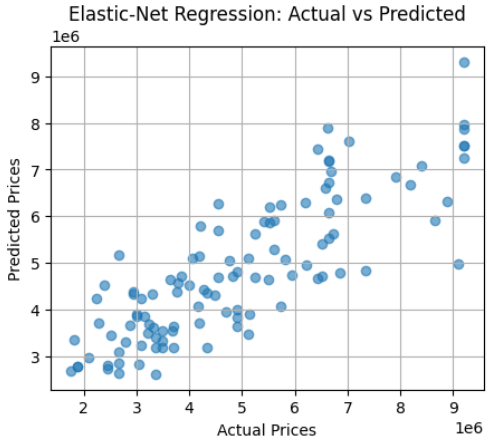


Fig. 5.6.6. Elastic-Net trained model metrics and output

7. Conclusion

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

Model	R^2	MAE	Description
Linear Regression	0.6815	880919	Baseline linear model with strong and stable performance
Lasso Regression	0.8615	880661	Automatically adjusts coefficients and removes less important features
Ridge Regression	0.6815	880919	Prevents overfitting, but slightly less accurate
Polynomial Regression	0.6373	935666	Increased model complexity, slight overfitting
MLP Regressor	-5.7364	4840170	Achieves highest accuracy, but requires more tuning and computation time
Elastic Net Regression	0.8615	880862	Balances Ridge and Lasso effects, but slightly weaker performance

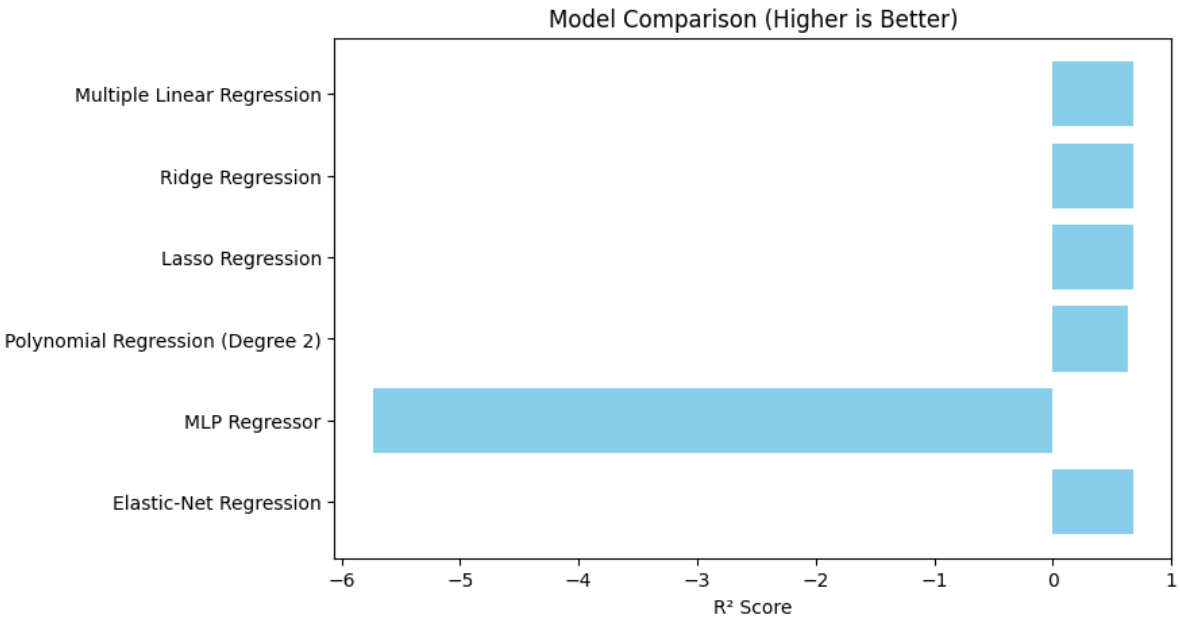
The linear models (Linear, Lasso, Ridge) all show similar performance which means this indicates that the data are approximately linear, and simple models are quite sufficient.
Elastic Net strikes a good balance between L1 and L2 regularization, but it did not yield significant improvement on this particular 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. And these results with codes are in 5.6.7.

7. Compare All Models

```
results_df = pd.DataFrame(results)
display(results_df.sort_values(by="R2", ascending=False))

plt.figure(figsize=(8, 5))
plt.barh(results_df["Model"], results_df["R2"], color='skyblue')
plt.xlabel("R² Score")
plt.title("Model Comparison (Higher is Better)")
plt.gca().invert_yaxis()
plt.show()
```

	Model	MAE	RMSE	R2
0	Multiple Linear Regression	8.809195e+05	1.134300e+06	0.681517
2	Lasso Regression	8.809195e+05	1.134300e+06	0.681517
5	Elastic-Net Regression	8.808630e+05	1.134303e+06	0.681515
1	Ridge Regression	8.806616e+05	1.134317e+06	0.681507
3	Polynomial Regression (Degree 2)	9.356669e+05	1.210551e+06	0.637259
4	MLP Regressor	4.840170e+06	5.216730e+06	-5.736385

5.6.7. Compare all Models



5.6.7. Compare all Models (Visualization)

Part VII: Using MLP with Feature Selector

To begin, the original regression problem of predicting continuous price values was reformulated into a four-class classification challenge. This approach required the neural network (MLP) to discover and learn decision boundaries in the feature space that best separate different price levels. The continuous target variable, Price, was discretized into four categories: Low, Medium-Low, Medium-High, and High.

After the neural network was trained as a classifier, the final classification output layer was removed. The activations from the last hidden layer—the new set of 32 abstract, nonlinear features—were then extracted. These 32 high-level features replaced the original 13 input features and were used as input variables for the final predictive model.

We using MLP for feature selection in pic 5.7.1

Model: "functional"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 13)	0
dense (Dense)	(None, 128)	1,792
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8,256
dropout_1 (Dropout)	(None, 64)	0
last_hidden (Dense)	(None, 32)	2,080
output (Dense)	(None, 4)	132

Total params: 12,260 (47.89 KB)
Trainable params: 12,260 (47.89 KB)
Non-trainable params: 0 (0.00 B)

5.7.1. Model Functionality

And after 12 epochs with extract from the last hidden layer I evaluate the model:

```
MLP Classifier Accuracy on test set: 0.6330275229357798

Classification Report (MLP):
              precision    recall  f1-score   support

           0       0.71      0.78      0.75        32
           1       0.37      0.53      0.43        19
           2       0.53      0.38      0.44        24
           3       0.83      0.74      0.78        34

    accuracy                           0.63       109
   macro avg       0.61      0.60      0.60       109
weighted avg       0.65      0.63      0.64       109

Confusion Matrix (MLP):
 [[25  7  0  0]
 [ 5 10  3  1]
 [ 4  7  9  4]
 [ 1  3  5 25]]
```

5.7.2. Model evaluation results

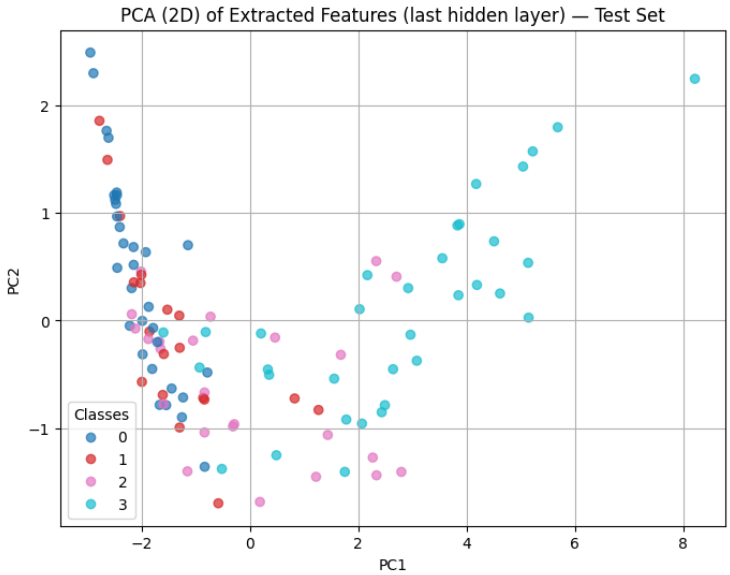
At least I compare all model's accuracy:

	Model	Accuracy
6	MLP_Classifier	0.633028
3	Polynomial	0.550459
0	LinearRegression	0.495413
2	Lasso	0.495413
1	Ridge	0.495413
5	ElasticNet	0.495413
4	MLPRegressor	0.293578

5.7.3. Compare all model's accuracy

MLP_Classifier has performed well with the 32 new features and it's the best model in the extracted set.

And visualize extracted features (PCA to 2D) as shown in Figure 5.5.1.



5.7.1. PCA Extracted