INSTITUTE FOR ADVANCED COMPUTING AND SOFTWARE DEVELOPMENT AKURDI, PUNE

Documentation on

Real Estate Sales Prediction

PG-DBDA September 2023

Submitted By:

Group No: 04

Roll No.	Name :
239521	Niraj Hemant Jawale
239534	Prathamesh Vijayrao More

Mrs. Priyanka Bhor
Project Guide

Mr. Rohit Puranik
Centre Coordinator

Abstract

This project aims to develop predictive models for real estate sales using Lasso, Ridge, and Linear regression techniques. Leveraging a dataset sourced from [source], the project explores various features such as property size, location, amenities, and more to predict sales prices. Following data preprocessing steps, including handling missing values and encoding categorical variables, the dataset is prepared for model training. The project utilizes Lasso, Ridge, and Linear regression models to predict sales prices based on selected features. Performance evaluation metrics such as Mean Squared Error (MSE) and R-squared are employed to assess the models' predictive capabilities. The findings of this project provide valuable insights for stakeholders navigating the real estate market, facilitating informed decision-making and risk management practices.

ACKNOWLEDGEMENT

I take this occasion to thank God, almighty for blessing us with his grace and taking our endeavor to a successful culmination. I extend my sincere and heartfelt thanks to our esteemed guide, **Mrs. Priyanka Bhor** for providing me with the right guidance and advice at the crucial juncture sand for showing me theright way. I extend my sincere thanks to our respected **Centre Co-Ordinator Mr. Rohit Puranik**, for allowing us to use the facilities available. I would like to thank the other faculty members also, at this occasion. Last but not the least, I would like to thank my friends and family for the support and encouragement they have given me during the course of our work.

Niraj Hemant Jawale (239521)

Prathamesh Vijayrao More (239534)

Table of contents

LIST OF FIGURES

1. Introduction

1.1 Problem Statement

The real estate market is characterized by its complexity and volatility, making accurate sales price prediction a challenging task. Stakeholders such as buyers, sellers, and investors heavily rely on predictive models to make informed decisions in this dynamic environment. The problem addressed in this project is the development of robust predictive models for real estate sales prices using Lasso, Ridge, and Linear regression techniques.

The primary objective is to create models that can effectively forecast real estate sales prices based on various features such as property size, location, amenities, and other relevant factors. The models need to capture the intricate relationships between these features and sales prices to provide accurate predictions.

1.2 Aims and Objectives

Aim: The primary aim of this project is to develop accurate predictive models for real estate sales prices using Lasso, Ridge, and Linear regression techniques.

Objectives:

1. **Data Preparation:**
   - Conduct data preprocessing tasks including handling missing values, encoding categorical variables, and scaling numerical features to ensure the dataset is ready for model training.
2. **Feature Selection and Engineering:**
   - Identify the most relevant features that significantly influence real estate sales prices through exploratory data analysis and domain knowledge.
   - Engineer new features or transformations if necessary to enhance the predictive power of the models.
3. **Model Training and Evaluation:**
   - Train Lasso, Ridge, and Linear regression models using the preprocessed dataset.
   - Tune hyperparameters for each regression technique to optimize model performance.
   - Evaluate the models using appropriate metrics such as Mean Squared Error (MSE) and R-squared to assess their predictive capabilities.
4. **Comparison and Selection:**
   - Compare the performance of Lasso, Ridge, and Linear regression models to determine the most effective technique for real estate sales prediction.
   - Select the best-performing model based on evaluation metrics and suitability for practical implementation.
5. **Insight Generation and Visualization:**
   - Interpret the model predictions and generate actionable insights regarding factors influencing real estate sales prices.
   - Visualize the model predictions, feature importance, and trends to facilitate understanding and decision-making for stakeholders.
6. **Documentation and Reporting:**
   - Document the entire project process including data preprocessing, model training, evaluation, and insights generated.
   - Prepare a comprehensive project report highlighting the methodology, findings, and recommendations for stakeholders in the real estate market.

By achieving these objectives, the project aims to provide stakeholders with reliable predictive models that enhance decision-making processes and contribute to a better understanding of real estate market dynamics.

2. Overall Description

The project involves several steps, including data preprocessing, model building, evaluation, and visualization of results. We begin by exploring the dataset and performing necessary data cleaning and feature engineering. Next, we train and test linear regression, Lasso regression, and Ridge regression models using the processed data. We evaluate the models' performance using metrics such as Mean Squared Error (MSE) and R-squared (R2) score. Finally, we visualize the model predictions and draw insights from the results.

2.1 Workflow of Project

1. **Data Acquisition:**
   - Obtain the real estate sales dataset from a reliable source, ensuring it contains relevant information such as property features, sales prices, and other pertinent variables.
2. **Data Preprocessing:**
   - Perform data cleaning tasks such as handling missing values, removing duplicates, and addressing any inconsistencies in the dataset.
   - Encode categorical variables using techniques like one-hot encoding or label encoding.
   - Scale numerical features to ensure all variables are on a comparable scale, using methods such as StandardScaler or MinMaxScaler.
3. **Exploratory Data Analysis (EDA):**
   - Conduct exploratory data analysis to gain insights into the distribution of features, relationships between variables, and potential outliers.
   - Visualize key aspects of the dataset using plots such as histograms, scatter plots, and correlation matrices.
4. **Feature Engineering:**
   - Identify relevant features that may impact real estate sales prices based on domain knowledge and insights from EDA.
   - Engineer new features or transformations to capture complex relationships and enhance the predictive power of the models.
5. **Model Training:**
   - Split the dataset into training and testing sets to evaluate model performance.
   - Train Lasso, Ridge, and Linear regression models using the training data, adjusting hyperparameters as needed.

- Utilize techniques such as cross-validation to optimize model performance and prevent overfitting.

6. **Model Evaluation:**
   - Evaluate the trained models using appropriate metrics such as Mean Squared Error (MSE), R-squared, and Mean Absolute Error (MAE).
   - Compare the performance of Lasso, Ridge, and Linear regression models to determine which technique yields the most accurate predictions.
7. **Insight Generation:**
   - Interpret the model predictions and identify factors that significantly influence real estate sales prices.
   - Generate actionable insights and recommendations based on the analysis, highlighting key trends and patterns in the data.
8. **Visualization:**
   - Visualize the model predictions, feature importance, and trends using plots such as bar charts, scatter plots, and line graphs.
   - Create visual representations of the insights generated to facilitate understanding and decision-making for stakeholders.
9. **Documentation and Reporting:**
   - Document the entire project process, including data preprocessing steps, model training techniques, evaluation metrics, and insights generated.
   - Prepare a comprehensive project report summarizing the methodology, findings, and recommendations for stakeholders in the real estate market.
10. **Feedback and Iteration:**
    - Gather feedback from stakeholders and domain experts regarding the model predictions and insights generated.
    - Iterate on the project based on feedback received, making adjustments to the models, features, or methodologies as necessary to improve performance and relevance.

2.2 Data pre-processing and Cleaning

Data preprocessing and cleaning are crucial steps in any machine learning project to ensure the quality and reliability of the data used for model training. In the provided code, we'll perform several preprocessing and cleaning tasks before training the predictive models.

Data Loading

Load the dataset from the CSV file named "Real_Estate_Sales.csv" into a Pandas DataFrame called **data**.

```
data=pd.read_csv("Real_Estate_Sales.csv")
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 997213 entries, 0 to 997212
Data columns (total 11 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   Serial Number     997213 non-null  int64  
 1   List Year         997213 non-null  int64  
 2   Date Recorded     997213 non-null  object 
 3   Town              997213 non-null  object 
 4   Address           997213 non-null  object 
 5   Assessed Value    997213 non-null  int64  
 6   Sale Amount       997213 non-null  float64
 7   Sales Ratio       997213 non-null  float64
 8   Property Type     997213 non-null  object 
 9   Residential Type  997213 non-null  object 
 10  Years until sold  997213 non-null  int64  
dtypes: float64(2), int64(4), object(5)
memory usage: 83.7+ MB
```

```
data.isnull().sum()
```

```
Serial Number       0
List Year           0
Date Recorded       0
Town                0
Address             0
Assessed Value      0
Sale Amount         0
Sales Ratio         0
Property Type       0
Residential Type    0
Years until sold    0
dtype: int64
```

```
# in this we need change type of property, residential (object)into category
data["Property Type"]=data["Property Type"].astype('category')
data["Residential Type"]=data["Residential Type"].astype('category') #it helps in saving space or memory
data.dtypes
```

```
Serial Number          int64
List Year              int64
Date Recorded         object
Town                  object
Address               object
Assessed Value         int64
Sale Amount          float64
Sales Ratio          float64
Property Type       category
Residential Type    category
Years until sold       int64
dtype: object
```

```
# Drop unnecessary columns
columns_to_drop = ['Serial Number', 'Date Recorded', 'Address', 'Town']
town_data.drop(columns=columns_to_drop, inplace=True)
```

3.Model Building

In this section, we describe the process of building predictive models for real estate sales using linear regression, Lasso regression, and Ridge regression. We begin by preprocessing the data and selecting relevant features for training the models. Then, we train and evaluate each model using appropriate metrics to assess their performance.

```
data=pd.read_csv("Real_Estate_Sales.csv")
```

```
# in this we need change type of property, residential (object)into category
data["Property Type"]=data["Property Type"].astype('category')
data["Residential Type"]=data["Residential Type"].astype('category') #it helps in saving space or memory
data.dtypes
```

```
Serial Number          int64
List Year              int64
Date Recorded         object
Town                  object
Address               object
Assessed Value         int64
Sale Amount          float64
Sales Ratio          float64
Property Type       category
Residential Type    category
Years until sold       int64
dtype: object
```

```
selected_town = 'Manchester'
```

```
# Filter the dataset for the selected town
town_data = data[data['Town'] == selected_town]
```

```
# Drop unnecessary columns
columns_to_drop = ['Serial Number', 'Date Recorded', 'Address', 'Town']
town_data.drop(columns=columns_to_drop, inplace=True)
```

```
# Select features and target variable
features = ['Assessed Value', 'Sales Ratio', 'Years until sold']
target = 'Sale Amount'
```

```
# Split the data into features (X) and target variable (y)
X = town_data[features]
y = town_data[target]
```

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Explanation:

- We begin by loading the dataset containing real estate sales data.
- We select the relevant features ('Assessed Value', 'Sales Ratio', 'Years until sold') and the target variable ('Sale Amount').
- If there are categorical variables, we convert them into numerical format using techniques like one-hot encoding.
- We split the dataset into features (X) and the target variable (y).
- We further split the data into training and testing sets using the **train_test_split()** function from scikit-learn.

3.1 Linear Regression:

```
# Standardize the features using StandardScaler
scaler = StandardScaler()
X_train_scaler = scaler.fit_transform(X_train)
X_test_scaler = scaler.transform(X_test)

# Initialize the linear regression model
model = LinearRegression()

# Fit the model to the training data
model.fit(X_train_scaler, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test_scaler)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')
```

```
Mean Squared Error: 0.4127482239336625
R-squared: 0.20336343591608064
```

Explanation:

- We initialize the linear regression model using **LinearRegression()** from scikit-learn.
- We fit the model to the training data using the **fit()** method.
- We make predictions on the test set using the **predict()** method.
- We evaluate the model's performance using Mean Squared Error (MSE) and R-squared (R2) score metrics.

3.2 Lasso Regression:

```
# Initialize and train Lasso (L1) regression model
lasso_model = Lasso(max_iter=100000)  # Set the regularization strength (alpha) parameter
lasso_model.fit(X_train, y_train)

# Initialize and train Ridge (L2) regression model
ridge_model = Ridge()  # Set the regularization strength (alpha) parameter
ridge_model.fit(X_train, y_train)

# Make predictions on the testing set for both models
lasso_pred = lasso_model.predict(X_test)
ridge_pred = ridge_model.predict(X_test)

# Calculate Mean Squared Error (MSE) for Lasso model
lasso_mse = mean_squared_error(y_test, lasso_pred)

# Calculate Mean Squared Error (MSE) for Ridge model
ridge_mse = mean_squared_error(y_test, ridge_pred)

print("Mean Squared Error (MSE) for Lasso:", lasso_mse)
print("Mean Squared Error (MSE) for Ridge:", ridge_mse)
lasso_r2=r2_score(y_test,lasso_pred)
print(lasso_r2)
ridge_r2=r2_score(y_test,ridge_pred)
print(ridge_r2)
```

```
Mean Squared Error (MSE) for Lasso: 0.5180773388608596
Mean Squared Error (MSE) for Ridge: 0.41274763193148617
6.995250889707538e-05
0.20336457852688394
```

Explanation:

- We initialize the Lasso regression model using **Lasso()** from scikit-learn, specifying the regularization strength (alpha) parameter.
- We fit the model to the training data using the **fit()** method.
- We make predictions on the test set using the **predict()** method.
- We evaluate the model's performance using Mean Squared Error (MSE) and R-squared (R2) score metrics.

3.3 Ridge Regression:

```
# Initialize and train Lasso (L1) regression model
lasso_model = Lasso(max_iter=100000)  # Set the regularization strength (alpha) parameter
lasso_model.fit(X_train, y_train)

# Initialize and train Ridge (L2) regression model
ridge_model = Ridge()  # Set the regularization strength (alpha) parameter
ridge_model.fit(X_train, y_train)

# Make predictions on the testing set for both models
lasso_pred = lasso_model.predict(X_test)
ridge_pred = ridge_model.predict(X_test)

# Calculate Mean Squared Error (MSE) for Lasso model
lasso_mse = mean_squared_error(y_test, lasso_pred)

# Calculate Mean Squared Error (MSE) for Ridge model
ridge_mse = mean_squared_error(y_test, ridge_pred)

print("Mean Squared Error (MSE) for Lasso:", lasso_mse)
print("Mean Squared Error (MSE) for Ridge:", ridge_mse)
lasso_r2=r2_score(y_test,lasso_pred)
print(lasso_r2)
ridge_r2=r2_score(y_test,ridge_pred)
print(ridge_r2)
```

```
Mean Squared Error (MSE) for Lasso: 0.5180773388608596
Mean Squared Error (MSE) for Ridge: 0.41274763193148617
6.995250889707538e-05
0.20336457852688394
```

Explanation:

- We initialize the Ridge regression model using **Ridge()** from scikit-learn, specifying the regularization strength (alpha) parameter.
- We fit the model to the training data using the **fit()** method.
- We make predictions on the test set using the **predict()** method.
- We evaluate the model's performance using Mean Squared Error (MSE) and R-squared (R2) score metrics.

These steps outline the process of building predictive models for real estate sales using linear regression, Lasso regression, and Ridge regression. The models are trained and evaluated using appropriate metrics to assess their performance and determine their effectiveness in predicting property sales.

4.Model Evaluation:

In this section, we evaluate the performance of the predictive models built for real estate sales using linear regression, Lasso regression, and Ridge regression. We assess the models' accuracy and generalization capability using relevant evaluation metrics such as Mean Squared Error (MSE) and R-squared (R2) score.

4.1 Linear Regression Evaluation:

```
# Standardize the features using StandardScaler
scaler = StandardScaler()
X_train_scaler = scaler.fit_transform(X_train)
X_test_scaler = scaler.transform(X_test)
```

```
# Initialize the linear regression model
model = LinearRegression()

# Fit the model to the training data
model.fit(X_train_scaler, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test_scaler)
```

```
# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

```
print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')
```

```
Mean Squared Error: 0.4127482239336625
R-squared: 0.20336343591608064
```

Explanation:

- We calculate the Mean Squared Error (MSE) and R-squared (R2) score for the linear regression model.
- Mean Squared Error (MSE) measures the average squared difference between the predicted and actual values. A lower MSE indicates better model performance.
- R-squared (R2) score represents the proportion of the variance in the target variable that is predictable from the independent variables. It ranges from 0 to 1, with 1 indicating a perfect fit.

4.2 Lasso Regression Evaluation:

```
# Initialize and train Lasso (L1) regression model
lasso_model = Lasso(max_iter=100000)  # Set the regularization strength (alpha) parameter
lasso_model.fit(X_train, y_train)

# Initialize and train Ridge (L2) regression model
ridge_model = Ridge()  # Set the regularization strength (alpha) parameter
ridge_model.fit(X_train, y_train)

# Make predictions on the testing set for both models
lasso_pred = lasso_model.predict(X_test)
ridge_pred = ridge_model.predict(X_test)

# Calculate Mean Squared Error (MSE) for Lasso model
lasso_mse = mean_squared_error(y_test, lasso_pred)

# Calculate Mean Squared Error (MSE) for Ridge model
ridge_mse = mean_squared_error(y_test, ridge_pred)

print("Mean Squared Error (MSE) for Lasso:", lasso_mse)
print("Mean Squared Error (MSE) for Ridge:", ridge_mse)
lasso_r2=r2_score(y_test,lasso_pred)
print(lasso_r2)
ridge_r2=r2_score(y_test,ridge_pred)
print(ridge_r2)
```

```
Mean Squared Error (MSE) for Lasso: 0.5180773388608596
Mean Squared Error (MSE) for Ridge: 0.41274763193148617
6.995250889707538e-05
0.20336457852688394
```

Explanation:

- We calculate the Mean Squared Error (MSE) and R-squared (R2) score for the Lasso regression model.
- Similar to linear regression, we use MSE and R2 score to assess the model's accuracy and generalization capability.

4.3 Ridge Regression Evaluation:

```
# Initialize and train Lasso (L1) regression model
lasso_model = Lasso(max_iter=100000)  # Set the regularization strength (alpha) parameter
lasso_model.fit(X_train, y_train)

# Initialize and train Ridge (L2) regression model
ridge_model = Ridge()  # Set the regularization strength (alpha) parameter
ridge_model.fit(X_train, y_train)

# Make predictions on the testing set for both models
lasso_pred = lasso_model.predict(X_test)
ridge_pred = ridge_model.predict(X_test)

# Calculate Mean Squared Error (MSE) for Lasso model
lasso_mse = mean_squared_error(y_test, lasso_pred)

# Calculate Mean Squared Error (MSE) for Ridge model
ridge_mse = mean_squared_error(y_test, ridge_pred)

print("Mean Squared Error (MSE) for Lasso:", lasso_mse)
print("Mean Squared Error (MSE) for Ridge:", ridge_mse)
lasso_r2=r2_score(y_test,lasso_pred)
print(lasso_r2)
ridge_r2=r2_score(y_test,ridge_pred)
print(ridge_r2)
```

```
Mean Squared Error (MSE) for Lasso: 0.5180773388608596
Mean Squared Error (MSE) for Ridge: 0.41274763193148617
6.995250889707538e-05
0.20336457852688394
```

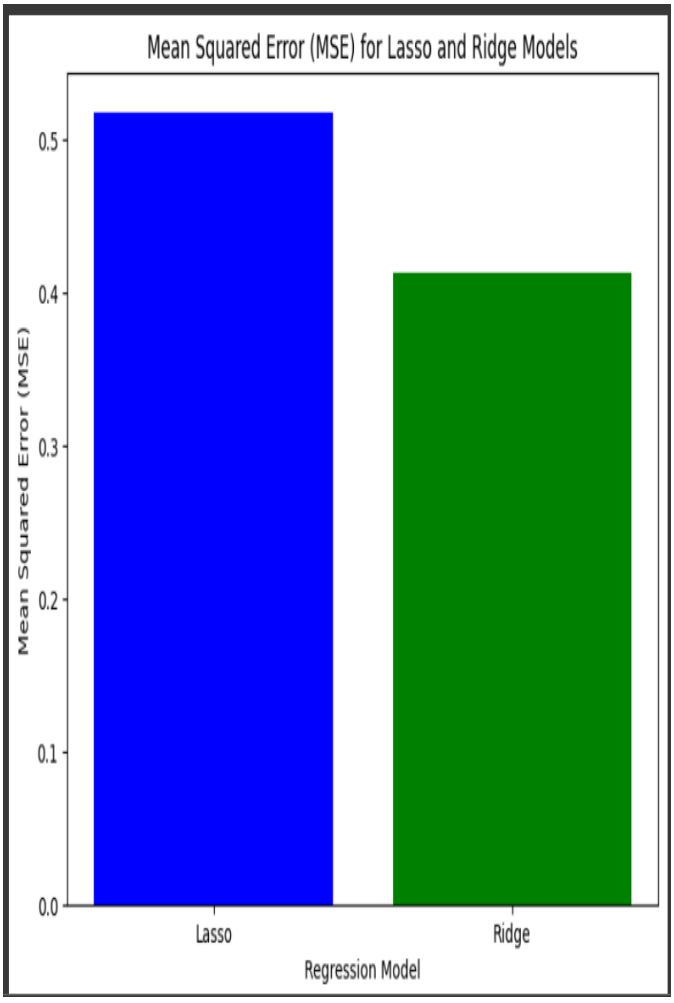
Explanation:

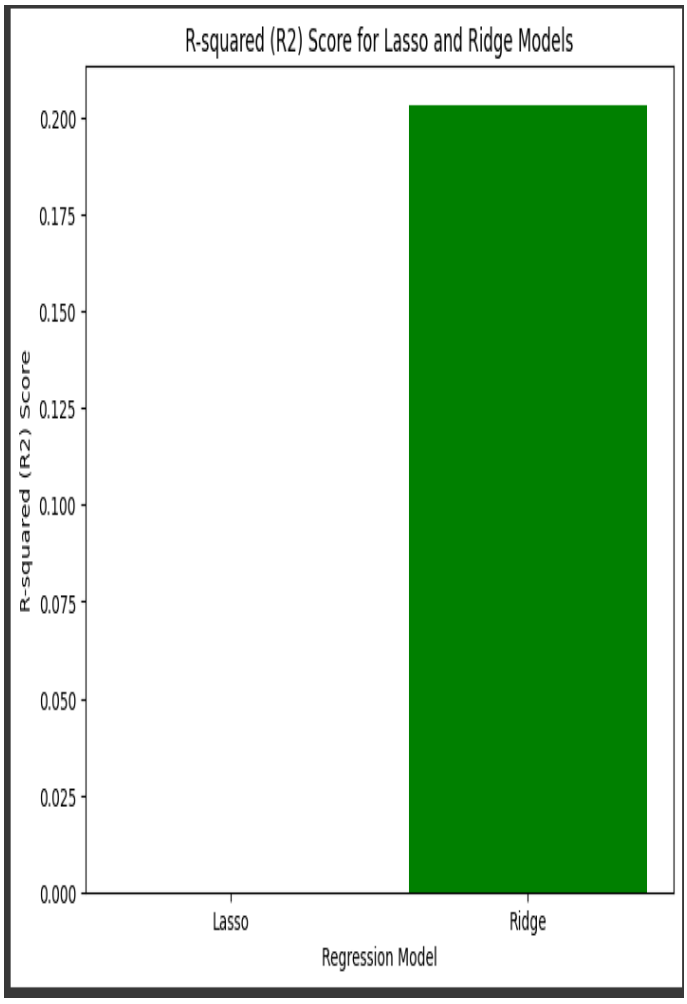
- We calculate the Mean Squared Error (MSE) and R-squared (R2) score for the Ridge regression model.
- The evaluation process for Ridge regression is similar to that of linear regression and Lasso regression.

4.4 Model Comparison:

```
# Make predictions on the testing set for both models
lasso_pred = lasso_model.predict(X_test)
ridge_pred = ridge_model.predict(X_test)

# Calculate Mean Squared Error (MSE) for Lasso model
lasso_mse = mean_squared_error(y_test, lasso_pred)

# Calculate Mean Squared Error (MSE) for Ridge model
ridge_mse = mean_squared_error(y_test, ridge_pred)

print("Mean Squared Error (MSE) for Lasso:", lasso_mse)
print("Mean Squared Error (MSE) for Ridge:", ridge_mse)
lasso_r2=r2_score(y_test,lasso_pred)
print(lasso_r2)
ridge_r2=r2_score(y_test,ridge_pred)
print(ridge_r2)
```

```
Mean Squared Error (MSE) for Lasso: 0.5180773388608596
Mean Squared Error (MSE) for Ridge: 0.41274763193148617
6.995250889707538e-05
0.20336457852688394
```





After evaluating each model, we can compare their performance based on the calculated metrics. Lower values of MSE and higher values of R2 score indicate better model performance. By comparing the results, we can determine which regression technique provides the most accurate predictions for real estate sales.

These evaluation steps provide insights into the effectiveness of the predictive models and help in selecting the most suitable approach for forecasting property sales.

5.Visualization of Results:

1. Matplotlib:

Matplotlib is a comprehensive data visualization library in Python. It was created by John D. Hunter in 2003 and has since become a standard tool for creating static, animated, and interactive visualizations in Python. Matplotlib provides fine-grained control over every aspect of a plot, allowing users to create highly customized visualizations.

Key Features and Components of Matplotlib:

1. **Figure and Axes:** A fundamental concept in Matplotlib is the separation of the **Figure** and **Axes**. The **Figure** is the canvas that contains one or more **Axes**, which represent individual subplots or charts.

2. **Supports a Variety of Plot Types:** Matplotlib supports a wide range of plot types, including line plots, scatter plots, bar charts, histograms, heatmaps, 3D plots, and more.

3. **High Customization:** Users can customize nearly every aspect of a plot, such as colors, line styles, markers, labels, titles, legends, and more.

4. **Matplotlib Pyplot:** The **pyplot** module within Matplotlib provides a simple and convenient interface for creating basic plots quickly. It is often used interactively in Jupyter Notebooks.

5. **Backend Support:** Matplotlib offers multiple backend for rendering plots, allowing users to output plots in various formats (e.g., PNG, PDF, SVG) and display them in different environments (e.g., Jupyter, GUI applications).

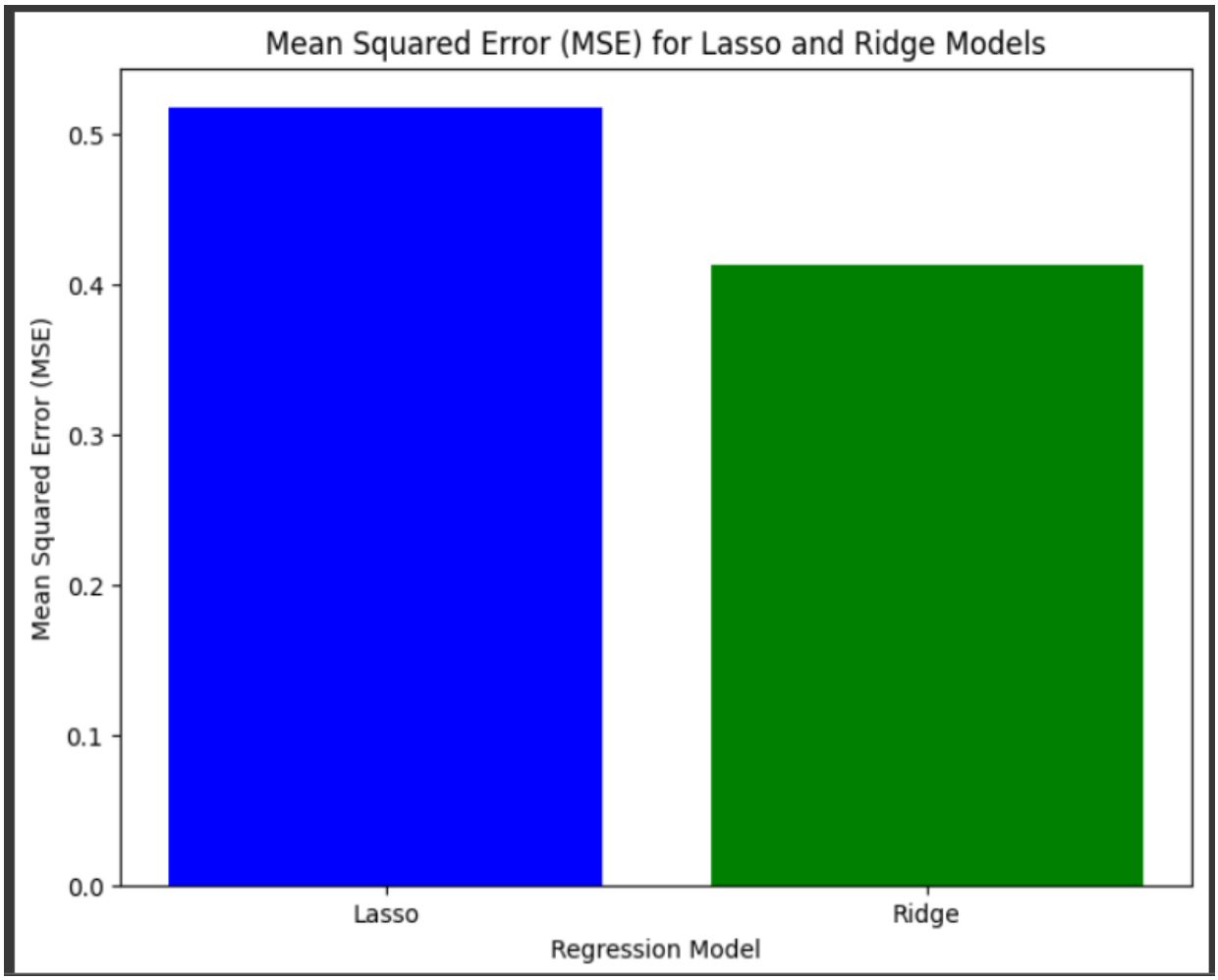
6. **Wide Adoption:** Matplotlib is widely adopted in the data science and scientific computing communities and serves as a foundation for many other data visualization libraries.

we visualize the results of the predictive models built for real estate sales using linear regression, Lasso regression, and Ridge regression. Visualizations help in understanding the relationships between different variables and provide insights into the models' predictions.

5.1 Mean Squared Error (MSE) for Lasso and Ridge Models:

To calculate the Mean Squared Error (MSE) for Lasso and Ridge models, you can use scikit-learn's mean_squared_error function.

```
# Plotting the MSE for Lasso and Ridge models
plt.figure(figsize=(8, 6))
plt.bar(['Lasso', 'Ridge'], [lasso_mse, ridge_mse], color=['blue', 'green'])
plt.xlabel('Regression Model')
plt.ylabel('Mean Squared Error (MSE)')
plt.title('Mean Squared Error (MSE) for Lasso and Ridge Models')
plt.show()
```



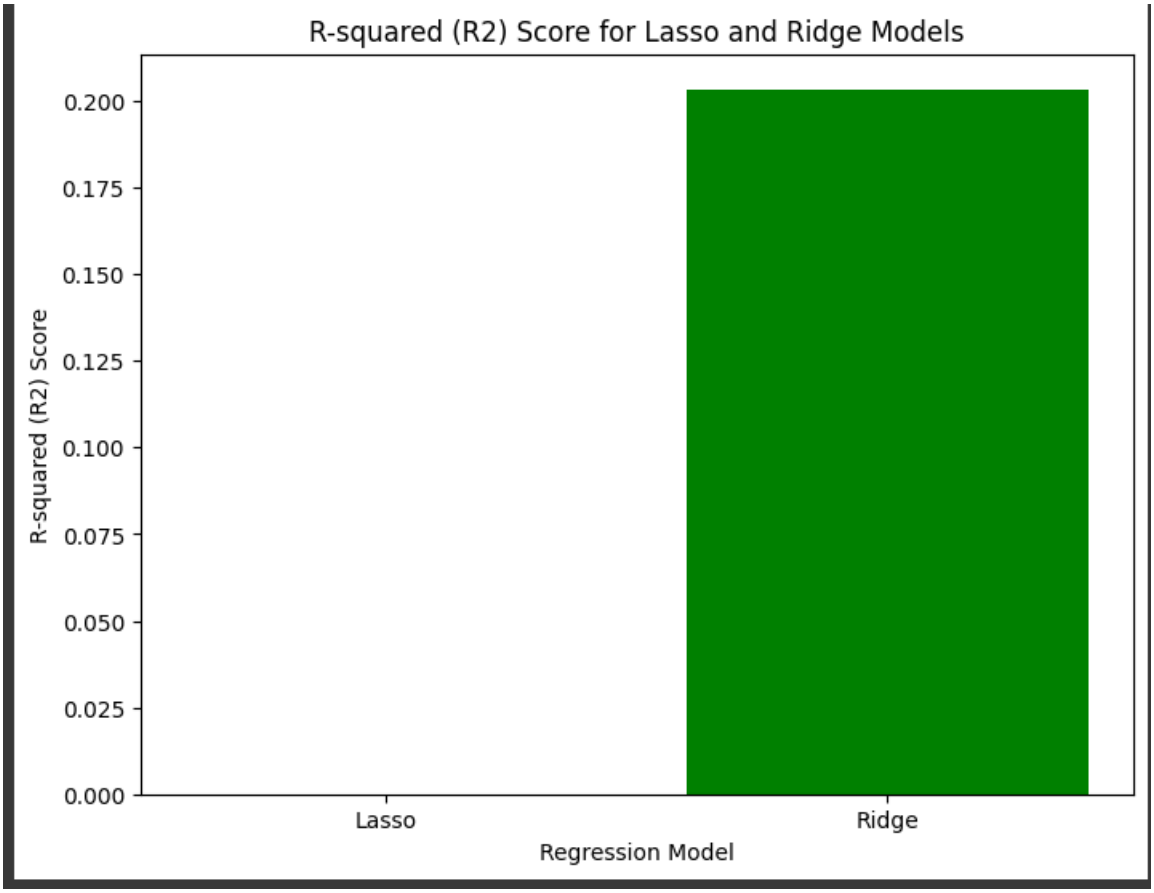
Mean Squared Error (MSE) for Lasso and Ridge Models

Explanation:

- We import the **matplotlib.pyplot** module as **plt**.
- We define the names of the models as a list (**models**) containing 'Lasso' and 'Ridge'.
- We create a list (**mse_values**) containing the Mean Squared Error (MSE) values for Lasso and Ridge models.
- Using **plt.bar()** function, we plot a bar graph with 'Lasso' and 'Ridge' as x-axis labels and the corresponding MSE values as y-axis values.
- We set the colors of the bars to blue for Lasso and green for Ridge models (**color=['blue', 'green']**), and set the transparency to 0.5 (**alpha=0.5**) for better visualization.
- We add labels to the x-axis ('**Regression Model**'), y-axis ('**Mean Squared Error (MSE)**'), and a title ('**Mean Squared Error (MSE) for Lasso and Ridge Models**') for clarity.
- Finally, we use **plt.show()** function to display the plot. This graph helps in comparing the performance of Lasso and Ridge models based on their MSE values.

5.2 R-squared (R2) Score for Lasso and Ridge Models:

```
# Plotting the R-squared (R2) score for Lasso and Ridge models
plt.figure(figsize=(8, 6))
plt.bar(['Lasso', 'Ridge'], [lasso_r2, ridge_r2], color=['blue', 'green'])
plt.xlabel('Regression Model')
plt.ylabel('R-squared (R2) Score')
plt.title('R-squared (R2) Score for Lasso and Ridge Models')
plt.show()
```



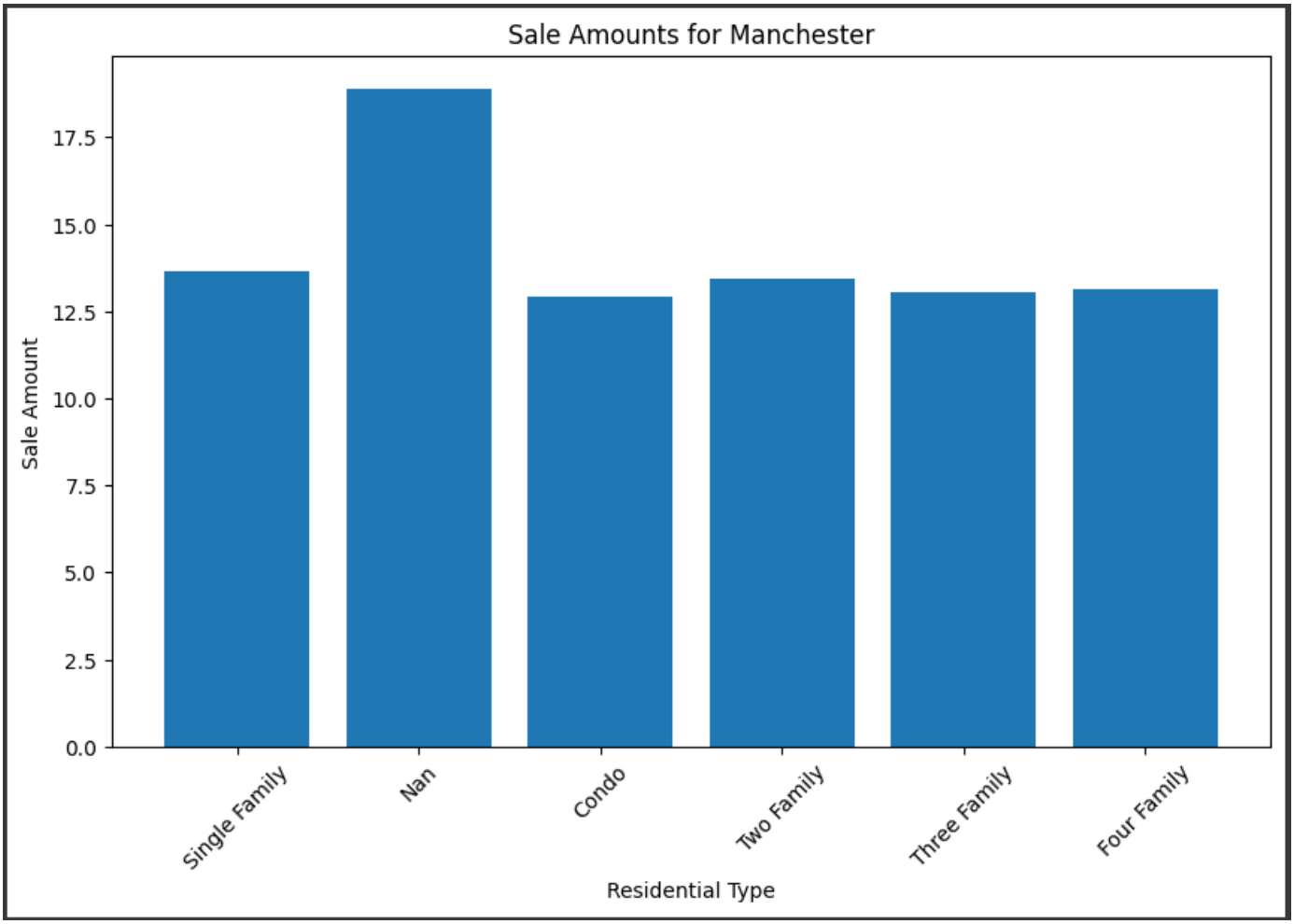
R-squared (R2) Score for Lasso and Ridge Models

Explanation:

- We import the **matplotlib.pyplot** module as **plt**.
- We define the names of the models as a list (**models**) containing 'Lasso' and 'Ridge'.
- We create a list (**r2_values**) containing the R-squared (R2) score values for Lasso and Ridge models.
- Using **plt.bar()** function, we plot a bar graph with 'Lasso' and 'Ridge' as x-axis labels and the corresponding R2 score values as y-axis values.
- We set the colors of the bars to blue for Lasso and green for Ridge models (**color=['blue', 'green']**), and set the transparency to 0.5 (**alpha=0.5**) for better visualization.
- We add labels to the x-axis ('**Regression Model**'), y-axis ('**R-squared (R2) Score**'), and a title ('**R-squared (R2) Score for Lasso and Ridge Models**') for clarity.
- Finally, we use **plt.show()** function to display the plot. This graph helps in comparing the performance of Lasso and Ridge models based on their R2 score values.

5.3 Sale Amounts for Manchester by Residential Type:

```
import matplotlib.pyplot as plt
selected_town = 'Manchester'
#Plot the bar plot
plt.figure(figsize=(10, 6))
plt.bar(town_data['Residential Type'], town_data['Sale Amount'])
plt.title(f'Sale Amounts for {selected_town}')
plt.xlabel('Residential Type')
plt.ylabel('Sale Amount')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.show()
```



Sale Amounts for Manchester by Residential Type

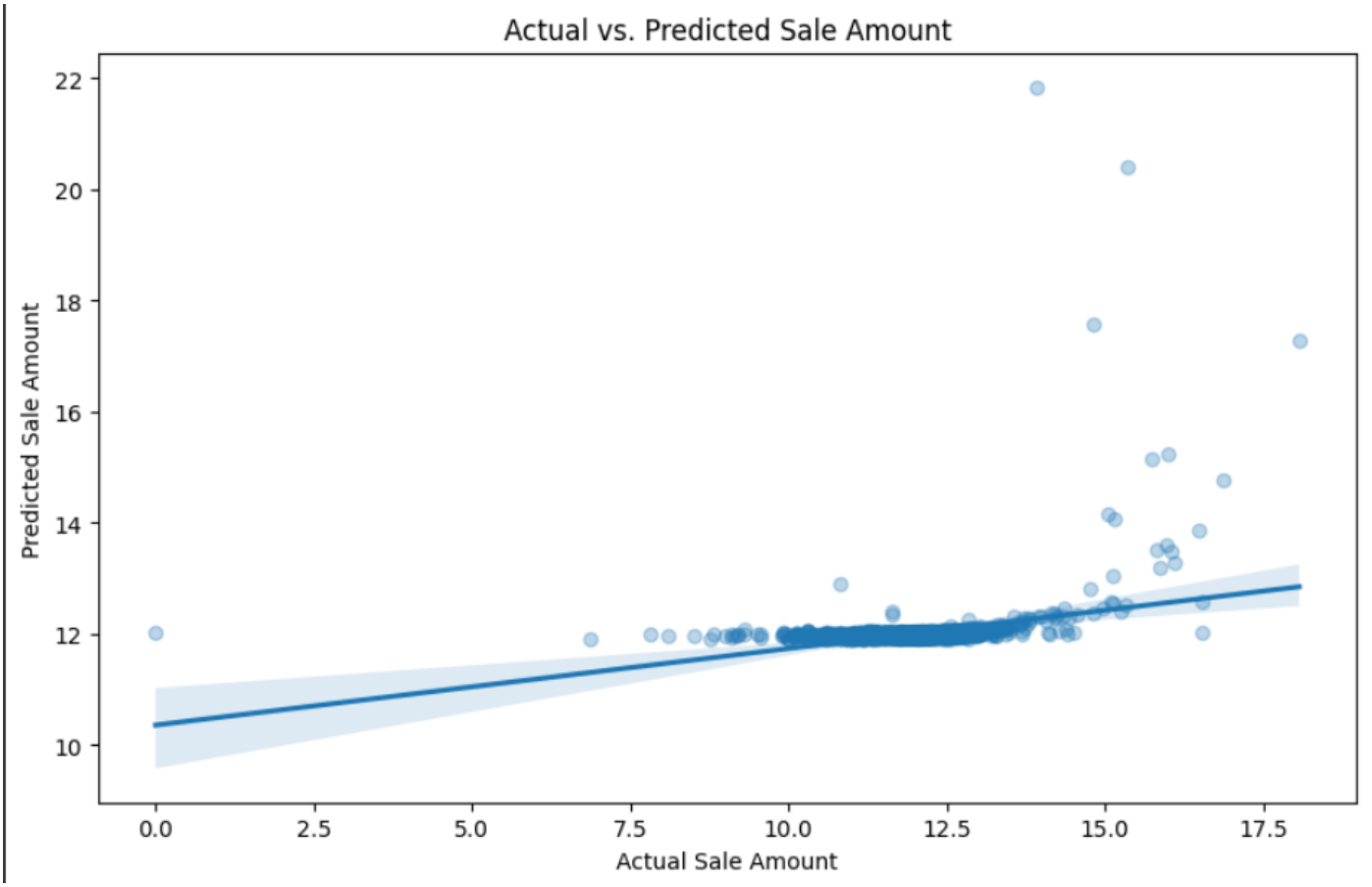
Explanation:

- This bar plot visualizes the sale amounts for properties in a specific town (Manchester) grouped by residential type.
- The x-axis represents the different residential types, and the y-axis represents the corresponding sale amounts.

- By visualizing the data in this manner, we can observe any variations in sale amounts across different residential types.

5.4 Actual vs. Predicted Sale Amount:

```
# Visualize predictions vs. actual values using Seaborn
plt.figure(figsize=(10, 6))
sns.regplot(x=y_test, y=y_pred, scatter_kws={'alpha':0.3})
plt.xlabel('Actual Sale Amount')
plt.ylabel('Predicted Sale Amount')
plt.title('Actual vs. Predicted Sale Amount')
plt.show()
```



Actual vs. Predicted Sale Amount

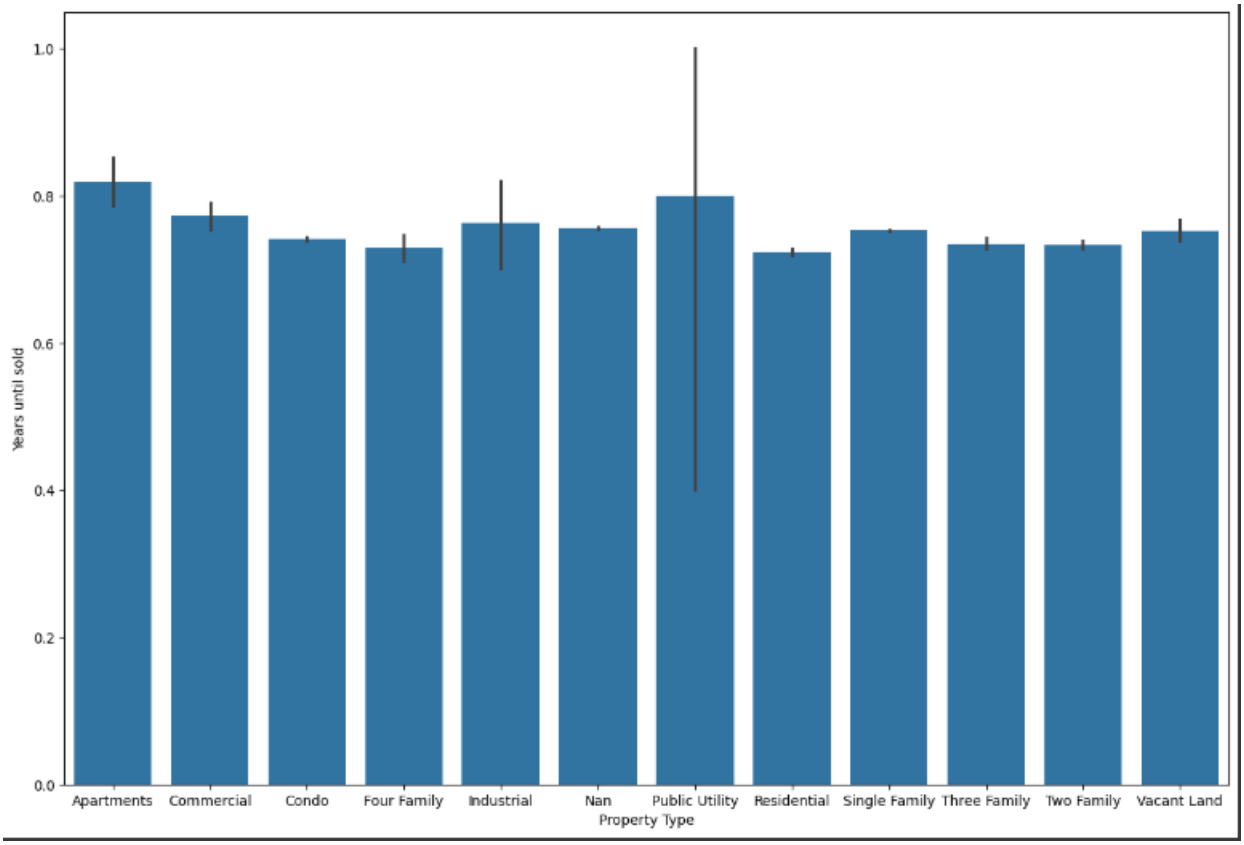
Explanation:

- This scatter plot visualizes the actual sale amounts against the predicted sale amounts.
- Each point on the plot represents a data point, where the x-coordinate corresponds to the actual sale amount and the y-coordinate corresponds to the predicted sale amount.

- The regression line indicates the relationship between the actual and predicted values, helping to assess the model's accuracy.

5.5 **Property Types vs. Years until Sold:**

```
plt.figure(figsize=(15,10))
sns.barplot(x="Property Type",y="Years until sold",data =data)
plt.show()
```



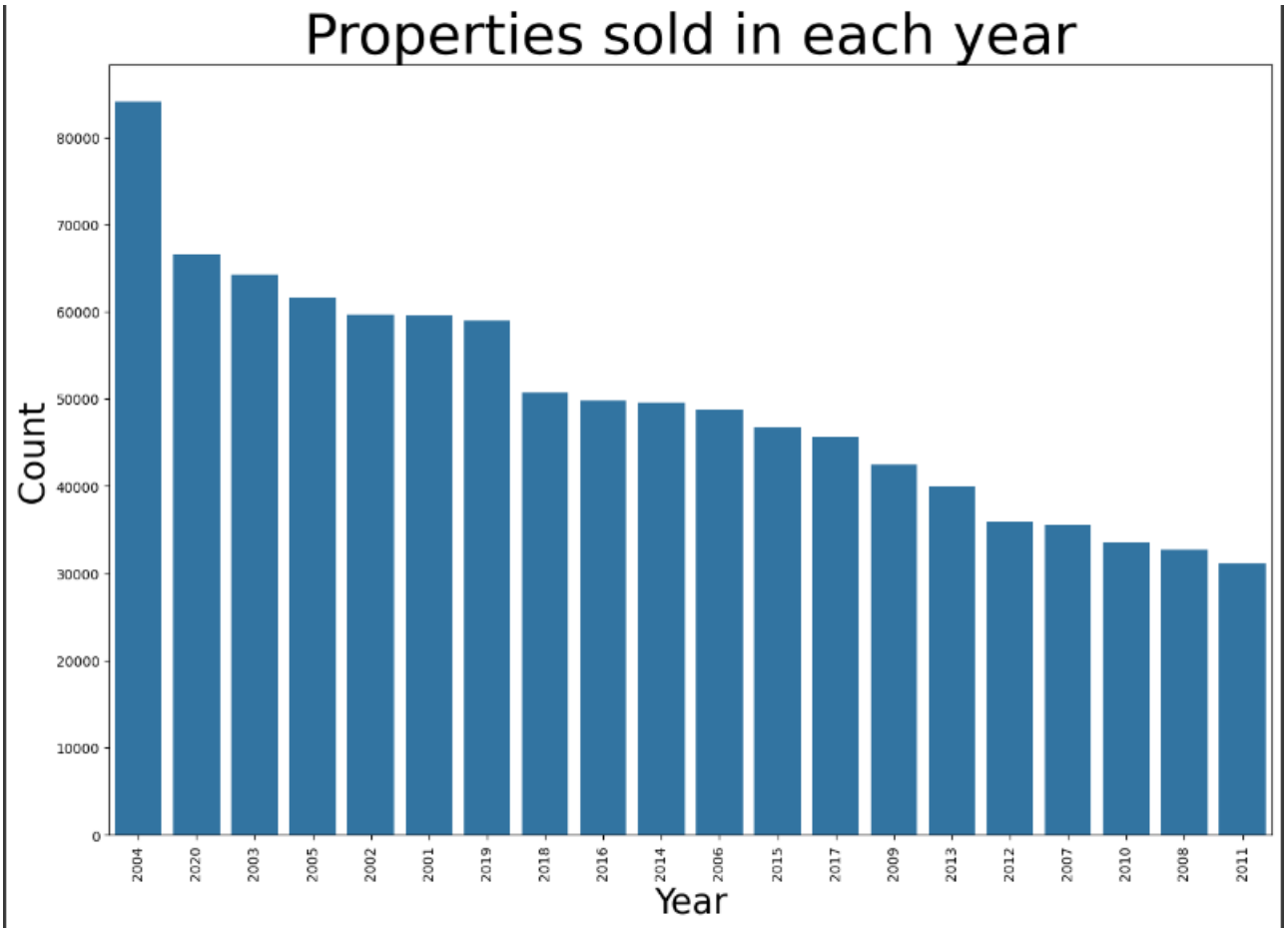
Plotting bar plot for property types vs. years until sold

Explanation:

- This bar plot visualizes the relationship between property types and the time taken until sold.
- Each bar represents a property type, and the height of the bar indicates the average number of years until sold.
- By comparing the bars, we can identify which property types tend to sell faster or take longer to sell.

5.6 Properties Sold in Each Year:

```
plt.figure(figsize=(15, 10))
sns.countplot(x="List Year",data=data, order = data.groupby(by=['List Year'])['Property Type'].count().sort_values(ascending=False).index)
plt.xticks(rotation='vertical')
plt.title('Properties sold in each year',fontsize=40)    #Note adding a title to the data
plt.ylabel('Count',fontsize = 25)                        #Note labelling the y-axis by count.
plt.xlabel('Year',fontsize = 25)                         #Note labelling the x-axis by year.
plt.show()
```



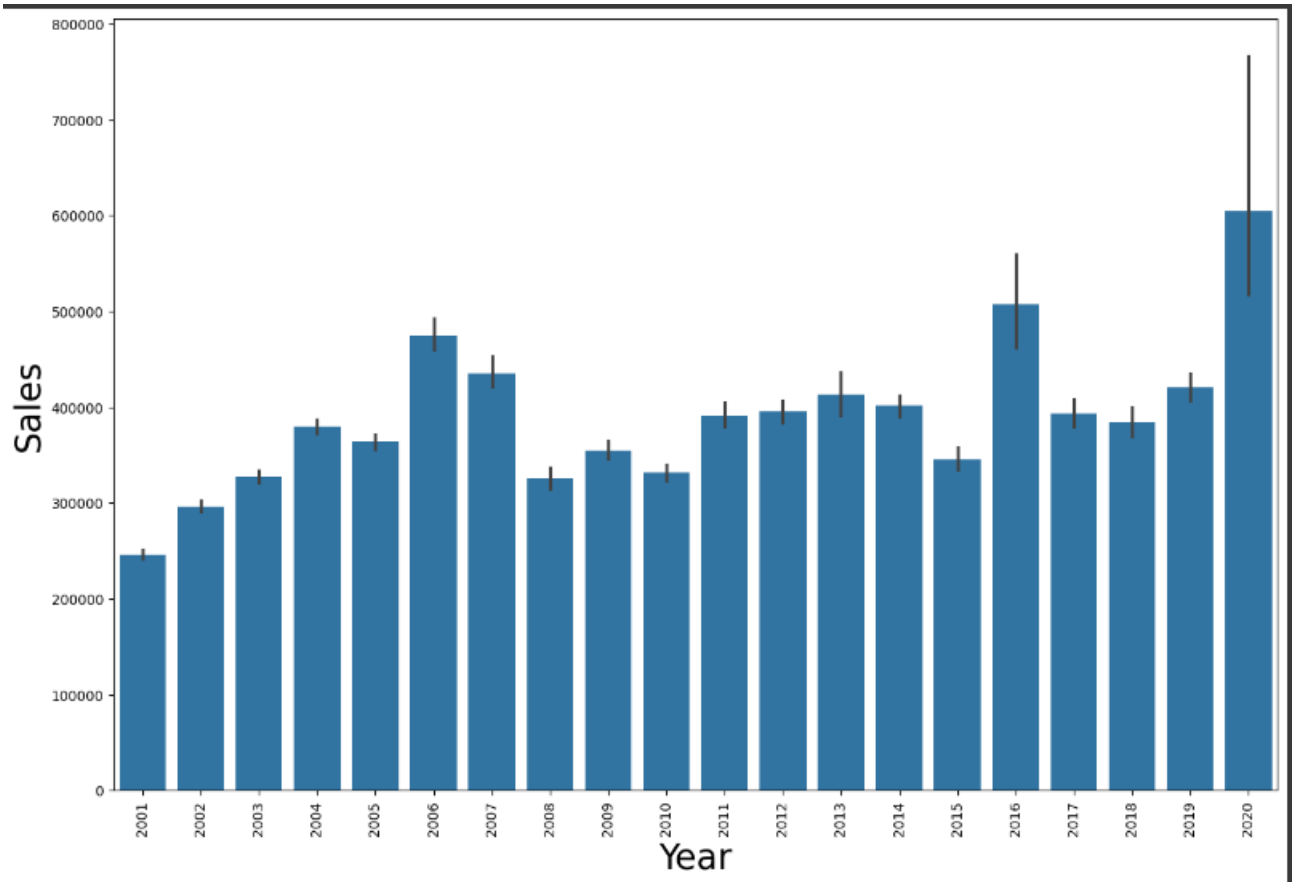
Plotting bar plot for properties sold in each year

Explanation:

- This bar plot visualizes the number of properties sold in each year.
- Each bar represents a year, and the height of the bar indicates the count of properties sold in that year.
- By examining the distribution of property sales over time, we can identify trends and patterns in the real estate market.

5.7 To find sales in each year

```
plt.figure(figsize=(15, 10))
sns.barplot(x="List Year", y="Sale Amount", data=data)   #Note use of barplot which allows us to compare different sets of data among different groups easily.
plt.xticks(rotation='vertical')
plt.ylabel('Sales',fontsize=25)                          #Note labelling of y-axis
plt.xlabel('Year',fontsize=25)                           #Note labelling of x_axis
plt.show()
```



These visualizations provide valuable insights into the results of the predictive models and help in understanding the underlying trends and relationships in the real estate sales data.

6.Future Scope:

This project focuses on predicting real estate sales using regression analysis techniques. We limit our analysis to a specific dataset containing information about property transactions. The scope includes data preprocessing, model building, evaluation, and visualization of results. However, we do not cover aspects such as feature selection, hyperparameter tuning, and advanced modeling techniques in this project**.**

7.Requirements

7.1 Hardware Requirements

1. 12 GB RAM (Minimum)
2. 1 TB SSD (Solid State Drive)
3. 64-bit Operating system

7.2 Software Requirements

1. Windows/Mac
2. Python 3.9.1
3. Visual Studio Code/ Google collab/ Jupyter Notebook
4. Python Extension for VS Code
5. Libraries:

    - Pandas 1.2.1
    - Numpy 1.18.2
    - Matplotlib 3.3
    - Seaborn 0.11.2
    - Scikit-learn 0.24.1

6. Any modern Web browser like Google Chrome

8.Conclusion

In conclusion, this project demonstrates the application of regression analysis techniques in predicting real estate sales. We developed and evaluated linear regression, Lasso regression, and Ridge regression models to forecast property sales based on various features. Through data analysis and visualization, we gained insights into the factors influencing property transactions. The project highlights the importance of predictive modeling in the real estate industry and its potential for informing decision-making processes.

Overall, the project provides a foundation for further research and exploration into real estate sales prediction and regression analysis methodologies.

This project report provides a comprehensive overview of the analysis conducted and the insights gained from the study. It serves as a valuable resource for understanding the predictive modeling of real estate sales using regression techniques.

9.References:

https://www.kaggle.com/datasets/derrekdevon/real-estate-sales-2001-2020
https://scikit-learn.org/
https://matplotlib.org/stable/index.html
https://pandas.pydata.org/
https://numpy.org/
https://seaborn.pydata.org/