🏠 House Price Prediction - Project Report

# Table of Contents

1. Introduction

2. Problem Statement

3. Dataset Description

4. Tools & Technologies Used

5. Data Preprocessing

6. Exploratory Data Analysis (EDA)

7. Feature Engineering

8. Model Building

9. Model Evaluation

10. Hyperparameter Tuning

11. Predictions on Test Data

12. Conclusion

13. Limitations & Future Work

14. References

15. Appendix (Code Snippets)

# 1. Introduction

This project aims to develop a machine learning model for predicting house prices based on various features such as area, location, and condition of the house. Accurate house price prediction is essential for real estate buyers, sellers, and investors. With the availability of historical housing data, we can leverage regression algorithms to estimate future prices effectively. The real estate market is influenced by various socio-economic factors, and a data-driven approach helps to make informed decisions.

This project aims to develop a machine learning model for predicting house prices based on various features such as area, location, and condition of the house. Accurate house price prediction is essential for real estate buyers, sellers, and investors. With the availability of historical housing data, we can leverage regression algorithms to estimate future prices effectively. The real estate market is influenced by various socio-economic factors, and a data-driven approach helps to make informed decisions.

This project aims to develop a machine learning model for predicting house prices based on various features such as area, location, and condition of the house. Accurate house price prediction is essential for real estate buyers, sellers, and investors. With the availability of historical housing data, we can leverage regression algorithms to estimate future prices effectively. The real estate market is influenced by various socio-economic factors, and a data-driven approach helps to make informed decisions.

This project aims to develop a machine learning model for predicting house prices based on various features such as area, location, and condition of the house. Accurate house price prediction is essential for real estate buyers, sellers, and investors. With the availability of historical housing data, we can leverage regression algorithms to estimate future prices effectively. The real estate market is influenced by various socio-economic factors, and a data-driven approach helps to make informed decisions.

This project aims to develop a machine learning model for predicting house prices based on various features such as area, location, and condition of the house. Accurate house price prediction is essential for real estate buyers, sellers, and investors. With the availability of historical housing data, we can leverage regression algorithms to estimate future prices effectively. The real estate market is influenced by various socio-economic factors, and a data-driven approach helps to make informed decisions.

This project aims to develop a machine learning model for predicting house prices based on various features such as area, location, and condition of the house. Accurate house price prediction is essential for real estate buyers, sellers, and investors. With the availability of historical housing data, we can leverage regression algorithms to estimate future prices effectively. The real estate market is influenced by various socio-economic factors, and a data-driven approach helps to make informed decisions.

This project aims to develop a machine learning model for predicting house prices based on various features such as area, location, and condition of the house. Accurate house price prediction is essential for real estate buyers, sellers, and investors. With the availability of historical housing data, we can leverage regression algorithms to estimate future prices effectively. The real estate market is influenced by various socio-economic factors, and a data-driven approach helps to make informed decisions.

This project aims to develop a machine learning model for predicting house prices based on various features such as area, location, and condition of the house. Accurate house price prediction is essential for real estate buyers, sellers, and investors. With the availability of historical housing data, we can leverage regression algorithms to estimate future prices effectively. The real estate market is influenced by various socio-economic factors, and a data-driven approach helps to make informed decisions.

This project aims to develop a machine learning model for predicting house prices based on various features such as area, location, and condition of the house. Accurate house price prediction is essential for real estate buyers, sellers, and investors. With the availability of historical housing data, we can leverage regression algorithms to estimate future prices effectively. The real estate market is influenced by various socio-economic factors, and a data-driven approach helps to make informed decisions.

This project aims to develop a machine learning model for predicting house prices based on various features such as area, location, and condition of the house. Accurate house price prediction is essential for real estate buyers, sellers, and investors. With the availability of historical housing data, we can leverage regression algorithms to estimate future prices effectively. The real estate market is influenced by various socio-economic factors, and a data-driven approach helps to make informed decisions.

This project aims to develop a machine learning model for predicting house prices based on various features such as area, location, and condition of the house. Accurate house price prediction is essential for real estate buyers, sellers, and investors. With the availability of historical housing data, we can leverage regression algorithms to estimate future prices effectively. The real estate market is influenced by various socio-economic factors, and a data-driven approach helps to make informed decisions.

This project aims to develop a machine learning model for predicting house prices based on various features such as area, location, and condition of the house. Accurate house price prediction is essential for real estate buyers, sellers, and investors. With the availability of historical housing data, we can leverage regression algorithms to estimate future prices effectively. The real estate market is influenced by various socio-economic factors, and a data-driven approach helps to make informed decisions.

# 2. Problem Statement

The main objective of this project is to predict the selling prices of houses given a set of features describing the property. The dataset used in this project consists of numeric and categorical features. The target variable is 'SalePrice', which we aim to predict. The challenge includes handling missing data, encoding categorical features, and selecting the right machine learning algorithm to build an accurate model.

The main objective of this project is to predict the selling prices of houses given a set of features describing the property. The dataset used in this project consists of numeric and categorical features. The target variable is 'SalePrice', which we aim to predict. The challenge includes handling missing data, encoding categorical features, and selecting the right machine learning algorithm to build an accurate model.

The main objective of this project is to predict the selling prices of houses given a set of features describing the property. The dataset used in this project consists of numeric and categorical features. The target variable is 'SalePrice', which we aim to predict. The challenge includes handling missing data, encoding categorical features, and selecting the right machine learning algorithm to build an accurate model.

The main objective of this project is to predict the selling prices of houses given a set of features describing the property. The dataset used in this project consists of numeric and categorical features. The target variable is 'SalePrice', which we aim to predict. The challenge includes handling missing data, encoding categorical features, and selecting the right machine learning algorithm to build an accurate model.

The main objective of this project is to predict the selling prices of houses given a set of features describing the property. The dataset used in this project consists of numeric and categorical features. The target variable is 'SalePrice', which we aim to predict. The challenge includes handling missing data, encoding categorical features, and selecting the right machine learning algorithm to build an accurate model.

The main objective of this project is to predict the selling prices of houses given a set of features describing the property. The dataset used in this project consists of numeric and categorical features. The target variable is 'SalePrice', which we aim to predict. The challenge includes handling missing data, encoding categorical features, and selecting the right machine learning algorithm to build an accurate model.

The main objective of this project is to predict the selling prices of houses given a set of features describing the property. The dataset used in this project consists of numeric and categorical features. The target variable is 'SalePrice', which we aim to predict. The challenge includes handling missing data, encoding categorical features, and selecting the right machine learning algorithm to build an accurate model.

The main objective of this project is to predict the selling prices of houses given a set of features describing the property. The dataset used in this project consists of numeric and categorical features. The target variable is 'SalePrice', which we aim to predict. The challenge includes handling missing data, encoding categorical features, and selecting the right machine learning algorithm to build an accurate model.

The main objective of this project is to predict the selling prices of houses given a set of features describing the property. The dataset used in this project consists of numeric and categorical features. The target variable is 'SalePrice', which we aim to predict. The challenge includes handling missing data, encoding categorical features, and selecting the right machine learning algorithm to build an accurate model.

The main objective of this project is to predict the selling prices of houses given a set of features describing the property. The dataset used in this project consists of numeric and categorical features. The target variable is 'SalePrice', which we aim to predict. The challenge includes handling missing data, encoding categorical features, and selecting the right machine learning algorithm to build an accurate model.

The main objective of this project is to predict the selling prices of houses given a set of features describing the property. The dataset used in this project consists of numeric and categorical features. The target variable is 'SalePrice', which we aim to predict. The challenge includes handling missing data, encoding categorical features, and selecting the right machine learning algorithm to build an accurate model.

The main objective of this project is to predict the selling prices of houses given a set of features describing the property. The dataset used in this project consists of numeric and categorical features. The target variable is 'SalePrice', which we aim to predict. The challenge includes handling missing data, encoding categorical features, and selecting the right machine learning algorithm to build an accurate model.

# 3. Dataset Description

The dataset includes training and test data. The training set contains 81 explanatory variables and one target variable: SalePrice. These features include physical attributes like overall quality, year built, number of rooms, and lot area. The test set contains the same features but lacks the target variable. Proper understanding of each feature, its data type, and potential impact on price is crucial for preprocessing and model building.

The dataset includes training and test data. The training set contains 81 explanatory variables and one target variable: SalePrice. These features include physical attributes like overall quality, year built, number of rooms, and lot area. The test set contains the same features but lacks the target variable. Proper understanding of each feature, its data type, and potential impact on price is crucial for preprocessing and model building.

The dataset includes training and test data. The training set contains 81 explanatory variables and one target variable: SalePrice. These features include physical attributes like overall quality, year built, number of rooms, and lot area. The test set contains the same features but lacks the target variable. Proper understanding of each feature, its data type, and potential impact on price is crucial for preprocessing and model building.

The dataset includes training and test data. The training set contains 81 explanatory variables and one target variable: SalePrice. These features include physical attributes like overall quality, year built, number of rooms, and lot area. The test set contains the same features but lacks the target variable. Proper understanding of each feature, its data type, and potential impact on price is crucial for preprocessing and model building.

The dataset includes training and test data. The training set contains 81 explanatory variables and one target variable: SalePrice. These features include physical attributes like overall quality, year built, number of rooms, and lot area. The test set contains the same features but lacks the target variable. Proper understanding of each feature, its data type, and potential impact on price is crucial for preprocessing and model building.

The dataset includes training and test data. The training set contains 81 explanatory variables and one target variable: SalePrice. These features include physical attributes like overall quality, year built, number of rooms, and lot area. The test set contains the same features but lacks the target variable. Proper understanding of each feature, its data type, and potential impact on price is crucial for preprocessing and model building.

The dataset includes training and test data. The training set contains 81 explanatory variables and one target variable: SalePrice. These features include physical attributes like overall quality, year built, number of rooms, and lot area. The test set contains the same features but lacks the target variable. Proper understanding of each feature, its data type, and potential impact on price is crucial for preprocessing and model building.

The dataset includes training and test data. The training set contains 81 explanatory variables and one target variable: SalePrice. These features include physical attributes like overall quality, year built, number of rooms, and lot area. The test set contains the same features but lacks the target variable. Proper understanding of each feature, its data type, and potential impact on price is crucial for preprocessing and model building.

The dataset includes training and test data. The training set contains 81 explanatory variables and one target variable: SalePrice. These features include physical attributes like overall quality, year built, number of rooms, and lot area. The test set contains the same features but lacks the target variable. Proper understanding of each feature, its data type, and potential impact on price is crucial for preprocessing and model building.

The dataset includes training and test data. The training set contains 81 explanatory variables and one target variable: SalePrice. These features include physical attributes like overall quality, year built, number of rooms, and lot area. The test set contains the same features but lacks the target variable. Proper understanding of each feature, its data type, and potential impact on price is crucial for preprocessing and model building.

The dataset includes training and test data. The training set contains 81 explanatory variables and one target variable: SalePrice. These features include physical attributes like overall quality, year built, number of rooms, and lot area. The test set contains the same features but lacks the target variable. Proper understanding of each feature, its data type, and potential impact on price is crucial for preprocessing and model building.

The dataset includes training and test data. The training set contains 81 explanatory variables and one target variable: SalePrice. These features include physical attributes like overall quality, year built, number of rooms, and lot area. The test set contains the same features but lacks the target variable. Proper understanding of each feature, its data type, and potential impact on price is crucial for preprocessing and model building.

# 4. Tools & Technologies Used

Python was used as the primary programming language due to its vast ecosystem of data science libraries. Key libraries include pandas for data manipulation, numpy for numerical operations, matplotlib and seaborn for data visualization, and scikit-learn for machine learning. The development environment was Visual Studio Code, and version control was managed using Git. These tools enabled efficient code development and analysis.

Python was used as the primary programming language due to its vast ecosystem of data science libraries. Key libraries include pandas for data manipulation, numpy for numerical operations, matplotlib and seaborn for data visualization, and scikit-learn for machine learning. The development environment was Visual Studio Code, and version control was managed using Git. These tools enabled efficient code development and analysis.

Python was used as the primary programming language due to its vast ecosystem of data science libraries. Key libraries include pandas for data manipulation, numpy for numerical operations, matplotlib and seaborn for data visualization, and scikit-learn for machine learning. The development environment was Visual Studio Code, and version control was managed using Git. These tools enabled efficient code development and analysis.

Python was used as the primary programming language due to its vast ecosystem of data science libraries. Key libraries include pandas for data manipulation, numpy for numerical operations, matplotlib and seaborn for data visualization, and scikit-learn for machine learning. The development environment was Visual Studio Code, and version control was managed using Git. These tools enabled efficient code development and analysis.

Python was used as the primary programming language due to its vast ecosystem of data science libraries. Key libraries include pandas for data manipulation, numpy for numerical operations, matplotlib and seaborn for data visualization, and scikit-learn for machine learning. The development environment was Visual Studio Code, and version control was managed using Git. These tools enabled efficient code development and analysis.

Python was used as the primary programming language due to its vast ecosystem of data science libraries. Key libraries include pandas for data manipulation, numpy for numerical operations, matplotlib and seaborn for data visualization, and scikit-learn for machine learning. The development environment was Visual Studio Code, and version control was managed using Git. These tools enabled efficient code development and analysis.

Python was used as the primary programming language due to its vast ecosystem of data science libraries. Key libraries include pandas for data manipulation, numpy for numerical operations, matplotlib and seaborn for data visualization, and scikit-learn for machine learning. The development environment was Visual Studio Code, and version control was managed using Git. These tools enabled efficient code development and analysis.

Python was used as the primary programming language due to its vast ecosystem of data science libraries. Key libraries include pandas for data manipulation, numpy for numerical operations, matplotlib and seaborn for data visualization, and scikit-learn for machine learning. The development environment was Visual Studio Code, and version control was managed using Git. These tools enabled efficient code development and analysis.

Python was used as the primary programming language due to its vast ecosystem of data science libraries. Key libraries include pandas for data manipulation, numpy for numerical operations, matplotlib and seaborn for data visualization, and scikit-learn for machine learning. The development environment was Visual Studio Code, and version control was managed using Git. These tools enabled efficient code development and analysis.

Python was used as the primary programming language due to its vast ecosystem of data science libraries. Key libraries include pandas for data manipulation, numpy for numerical operations, matplotlib and seaborn for data visualization, and scikit-learn for machine learning. The development environment was Visual Studio Code, and version control was managed using Git. These tools enabled efficient code development and analysis.

Python was used as the primary programming language due to its vast ecosystem of data science libraries. Key libraries include pandas for data manipulation, numpy for numerical operations, matplotlib and seaborn for data visualization, and scikit-learn for machine learning. The development environment was Visual Studio Code, and version control was managed using Git. These tools enabled efficient code development and analysis.

Python was used as the primary programming language due to its vast ecosystem of data science libraries. Key libraries include pandas for data manipulation, numpy for numerical operations, matplotlib and seaborn for data visualization, and scikit-learn for machine learning. The development environment was Visual Studio Code, and version control was managed using Git. These tools enabled efficient code development and analysis.

# 5. Data Preprocessing

Data preprocessing is a critical step to clean and prepare the dataset for modeling. It involved handling missing values using imputation techniques, encoding categorical variables using one-hot encoding, and feature scaling where necessary. Some features with a high percentage of missing values were dropped. The dataset was also split into training and validation sets to evaluate model performance before final testing.

Data preprocessing is a critical step to clean and prepare the dataset for modeling. It involved handling missing values using imputation techniques, encoding categorical variables using one-hot encoding, and feature scaling where necessary. Some features with a high percentage of missing values were dropped. The dataset was also split into training and validation sets to evaluate model performance before final testing.

Data preprocessing is a critical step to clean and prepare the dataset for modeling. It involved handling missing values using imputation techniques, encoding categorical variables using one-hot encoding, and feature scaling where necessary. Some features with a high percentage of missing values were dropped. The dataset was also split into training and validation sets to evaluate model performance before final testing.

Data preprocessing is a critical step to clean and prepare the dataset for modeling. It involved handling missing values using imputation techniques, encoding categorical variables using one-hot encoding, and feature scaling where necessary. Some features with a high percentage of missing values were dropped. The dataset was also split into training and validation sets to evaluate model performance before final testing.

Data preprocessing is a critical step to clean and prepare the dataset for modeling. It involved handling missing values using imputation techniques, encoding categorical variables using one-hot encoding, and feature scaling where necessary. Some features with a high percentage of missing values were dropped. The dataset was also split into training and validation sets to evaluate model performance before final testing.

Data preprocessing is a critical step to clean and prepare the dataset for modeling. It involved handling missing values using imputation techniques, encoding categorical variables using one-hot encoding, and feature scaling where necessary. Some features with a high percentage of missing values were dropped. The dataset was also split into training and validation sets to evaluate model performance before final testing.

Data preprocessing is a critical step to clean and prepare the dataset for modeling. It involved handling missing values using imputation techniques, encoding categorical variables using one-hot encoding, and feature scaling where necessary. Some features with a high percentage of missing values were dropped. The dataset was also split into training and validation sets to evaluate model performance before final testing.

Data preprocessing is a critical step to clean and prepare the dataset for modeling. It involved handling missing values using imputation techniques, encoding categorical variables using one-hot encoding, and feature scaling where necessary. Some features with a high percentage of missing values were dropped. The dataset was also split into training and validation sets to evaluate model performance before final testing.

Data preprocessing is a critical step to clean and prepare the dataset for modeling. It involved handling missing values using imputation techniques, encoding categorical variables using one-hot encoding, and feature scaling where necessary. Some features with a high percentage of missing values were dropped. The dataset was also split into training and validation sets to evaluate model performance before final testing.

Data preprocessing is a critical step to clean and prepare the dataset for modeling. It involved handling missing values using imputation techniques, encoding categorical variables using one-hot encoding, and feature scaling where necessary. Some features with a high percentage of missing values were dropped. The dataset was also split into training and validation sets to evaluate model performance before final testing.

Data preprocessing is a critical step to clean and prepare the dataset for modeling. It involved handling missing values using imputation techniques, encoding categorical variables using one-hot encoding, and feature scaling where necessary. Some features with a high percentage of missing values were dropped. The dataset was also split into training and validation sets to evaluate model performance before final testing.

Data preprocessing is a critical step to clean and prepare the dataset for modeling. It involved handling missing values using imputation techniques, encoding categorical variables using one-hot encoding, and feature scaling where necessary. Some features with a high percentage of missing values were dropped. The dataset was also split into training and validation sets to evaluate model performance before final testing.

# 6. Exploratory Data Analysis (EDA)

EDA was conducted to understand the data distribution, detect outliers, and identify correlations between variables. Histograms, box plots, and scatter plots were used to visualize numerical data, while bar plots were used for categorical variables. A correlation heatmap revealed strong predictors of sale price. This step helped in feature selection and understanding which variables might need transformation.

EDA was conducted to understand the data distribution, detect outliers, and identify correlations between variables. Histograms, box plots, and scatter plots were used to visualize numerical data, while bar plots were used for categorical variables. A correlation heatmap revealed strong predictors of sale price. This step helped in feature selection and understanding which variables might need transformation.

EDA was conducted to understand the data distribution, detect outliers, and identify correlations between variables. Histograms, box plots, and scatter plots were used to visualize numerical data, while bar plots were used for categorical variables. A correlation heatmap revealed strong predictors of sale price. This step helped in feature selection and understanding which variables might need transformation.

EDA was conducted to understand the data distribution, detect outliers, and identify correlations between variables. Histograms, box plots, and scatter plots were used to visualize numerical data, while bar plots were used for categorical variables. A correlation heatmap revealed strong predictors of sale price. This step helped in feature selection and understanding which variables might need transformation.

EDA was conducted to understand the data distribution, detect outliers, and identify correlations between variables. Histograms, box plots, and scatter plots were used to visualize numerical data, while bar plots were used for categorical variables. A correlation heatmap revealed strong predictors of sale price. This step helped in feature selection and understanding which variables might need transformation.

EDA was conducted to understand the data distribution, detect outliers, and identify correlations between variables. Histograms, box plots, and scatter plots were used to visualize numerical data, while bar plots were used for categorical variables. A correlation heatmap revealed strong predictors of sale price. This step helped in feature selection and understanding which variables might need transformation.

EDA was conducted to understand the data distribution, detect outliers, and identify correlations between variables. Histograms, box plots, and scatter plots were used to visualize numerical data, while bar plots were used for categorical variables. A correlation heatmap revealed strong predictors of sale price. This step helped in feature selection and understanding which variables might need transformation.

EDA was conducted to understand the data distribution, detect outliers, and identify correlations between variables. Histograms, box plots, and scatter plots were used to visualize numerical data, while bar plots were used for categorical variables. A correlation heatmap revealed strong predictors of sale price. This step helped in feature selection and understanding which variables might need transformation.

EDA was conducted to understand the data distribution, detect outliers, and identify correlations between variables. Histograms, box plots, and scatter plots were used to visualize numerical data, while bar plots were used for categorical variables. A correlation heatmap revealed strong predictors of sale price. This step helped in feature selection and understanding which variables might need transformation.

EDA was conducted to understand the data distribution, detect outliers, and identify correlations between variables. Histograms, box plots, and scatter plots were used to visualize numerical data, while bar plots were used for categorical variables. A correlation heatmap revealed strong predictors of sale price. This step helped in feature selection and understanding which variables might need transformation.

EDA was conducted to understand the data distribution, detect outliers, and identify correlations between variables. Histograms, box plots, and scatter plots were used to visualize numerical data, while bar plots were used for categorical variables. A correlation heatmap revealed strong predictors of sale price. This step helped in feature selection and understanding which variables might need transformation.

EDA was conducted to understand the data distribution, detect outliers, and identify correlations between variables. Histograms, box plots, and scatter plots were used to visualize numerical data, while bar plots were used for categorical variables. A correlation heatmap revealed strong predictors of sale price. This step helped in feature selection and understanding which variables might need transformation.

# 7. Feature Engineering

New features were created by combining or transforming existing ones. For example, TotalBathrooms was created by summing full and half bathrooms. Some variables were log-transformed to reduce skewness. Categorical variables were encoded using one-hot encoding. Irrelevant features with little predictive power were dropped. These transformations enhanced model performance and interpretability.

New features were created by combining or transforming existing ones. For example, TotalBathrooms was created by summing full and half bathrooms. Some variables were log-transformed to reduce skewness. Categorical variables were encoded using one-hot encoding. Irrelevant features with little predictive power were dropped. These transformations enhanced model performance and interpretability.

New features were created by combining or transforming existing ones. For example, TotalBathrooms was created by summing full and half bathrooms. Some variables were log-transformed to reduce skewness. Categorical variables were encoded using one-hot encoding. Irrelevant features with little predictive power were dropped. These transformations enhanced model performance and interpretability.

New features were created by combining or transforming existing ones. For example, TotalBathrooms was created by summing full and half bathrooms. Some variables were log-transformed to reduce skewness. Categorical variables were encoded using one-hot encoding. Irrelevant features with little predictive power were dropped. These transformations enhanced model performance and interpretability.

New features were created by combining or transforming existing ones. For example, TotalBathrooms was created by summing full and half bathrooms. Some variables were log-transformed to reduce skewness. Categorical variables were encoded using one-hot encoding. Irrelevant features with little predictive power were dropped. These transformations enhanced model performance and interpretability.

New features were created by combining or transforming existing ones. For example, TotalBathrooms was created by summing full and half bathrooms. Some variables were log-transformed to reduce skewness. Categorical variables were encoded using one-hot encoding. Irrelevant features with little predictive power were dropped. These transformations enhanced model performance and interpretability.

New features were created by combining or transforming existing ones. For example, TotalBathrooms was created by summing full and half bathrooms. Some variables were log-transformed to reduce skewness. Categorical variables were encoded using one-hot encoding. Irrelevant features with little predictive power were dropped. These transformations enhanced model performance and interpretability.

New features were created by combining or transforming existing ones. For example, TotalBathrooms was created by summing full and half bathrooms. Some variables were log-transformed to reduce skewness. Categorical variables were encoded using one-hot encoding. Irrelevant features with little predictive power were dropped. These transformations enhanced model performance and interpretability.

New features were created by combining or transforming existing ones. For example, TotalBathrooms was created by summing full and half bathrooms. Some variables were log-transformed to reduce skewness. Categorical variables were encoded using one-hot encoding. Irrelevant features with little predictive power were dropped. These transformations enhanced model performance and interpretability.

New features were created by combining or transforming existing ones. For example, TotalBathrooms was created by summing full and half bathrooms. Some variables were log-transformed to reduce skewness. Categorical variables were encoded using one-hot encoding. Irrelevant features with little predictive power were dropped. These transformations enhanced model performance and interpretability.

New features were created by combining or transforming existing ones. For example, TotalBathrooms was created by summing full and half bathrooms. Some variables were log-transformed to reduce skewness. Categorical variables were encoded using one-hot encoding. Irrelevant features with little predictive power were dropped. These transformations enhanced model performance and interpretability.

New features were created by combining or transforming existing ones. For example, TotalBathrooms was created by summing full and half bathrooms. Some variables were log-transformed to reduce skewness. Categorical variables were encoded using one-hot encoding. Irrelevant features with little predictive power were dropped. These transformations enhanced model performance and interpretability.

# 8. Model Building

Multiple regression models were trained including Linear Regression, Decision Trees, Random Forests, and Gradient Boosting. Each model was evaluated using training and validation data. Linear Regression served as a baseline. Random Forest and Gradient Boosting performed better due to their ability to handle non-linear relationships. Model selection was based on evaluation metrics and cross-validation results.

Multiple regression models were trained including Linear Regression, Decision Trees, Random Forests, and Gradient Boosting. Each model was evaluated using training and validation data. Linear Regression served as a baseline. Random Forest and Gradient Boosting performed better due to their ability to handle non-linear relationships. Model selection was based on evaluation metrics and cross-validation results.

Multiple regression models were trained including Linear Regression, Decision Trees, Random Forests, and Gradient Boosting. Each model was evaluated using training and validation data. Linear Regression served as a baseline. Random Forest and Gradient Boosting performed better due to their ability to handle non-linear relationships. Model selection was based on evaluation metrics and cross-validation results.

Multiple regression models were trained including Linear Regression, Decision Trees, Random Forests, and Gradient Boosting. Each model was evaluated using training and validation data. Linear Regression served as a baseline. Random Forest and Gradient Boosting performed better due to their ability to handle non-linear relationships. Model selection was based on evaluation metrics and cross-validation results.

Multiple regression models were trained including Linear Regression, Decision Trees, Random Forests, and Gradient Boosting. Each model was evaluated using training and validation data. Linear Regression served as a baseline. Random Forest and Gradient Boosting performed better due to their ability to handle non-linear relationships. Model selection was based on evaluation metrics and cross-validation results.

Multiple regression models were trained including Linear Regression, Decision Trees, Random Forests, and Gradient Boosting. Each model was evaluated using training and validation data. Linear Regression served as a baseline. Random Forest and Gradient Boosting performed better due to their ability to handle non-linear relationships. Model selection was based on evaluation metrics and cross-validation results.

Multiple regression models were trained including Linear Regression, Decision Trees, Random Forests, and Gradient Boosting. Each model was evaluated using training and validation data. Linear Regression served as a baseline. Random Forest and Gradient Boosting performed better due to their ability to handle non-linear relationships. Model selection was based on evaluation metrics and cross-validation results.

Multiple regression models were trained including Linear Regression, Decision Trees, Random Forests, and Gradient Boosting. Each model was evaluated using training and validation data. Linear Regression served as a baseline. Random Forest and Gradient Boosting performed better due to their ability to handle non-linear relationships. Model selection was based on evaluation metrics and cross-validation results.

Multiple regression models were trained including Linear Regression, Decision Trees, Random Forests, and Gradient Boosting. Each model was evaluated using training and validation data. Linear Regression served as a baseline. Random Forest and Gradient Boosting performed better due to their ability to handle non-linear relationships. Model selection was based on evaluation metrics and cross-validation results.

Multiple regression models were trained including Linear Regression, Decision Trees, Random Forests, and Gradient Boosting. Each model was evaluated using training and validation data. Linear Regression served as a baseline. Random Forest and Gradient Boosting performed better due to their ability to handle non-linear relationships. Model selection was based on evaluation metrics and cross-validation results.

Multiple regression models were trained including Linear Regression, Decision Trees, Random Forests, and Gradient Boosting. Each model was evaluated using training and validation data. Linear Regression served as a baseline. Random Forest and Gradient Boosting performed better due to their ability to handle non-linear relationships. Model selection was based on evaluation metrics and cross-validation results.

Multiple regression models were trained including Linear Regression, Decision Trees, Random Forests, and Gradient Boosting. Each model was evaluated using training and validation data. Linear Regression served as a baseline. Random Forest and Gradient Boosting performed better due to their ability to handle non-linear relationships. Model selection was based on evaluation metrics and cross-validation results.

# 9. Model Evaluation

Model performance was assessed using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared. Cross-validation was used to ensure that the model generalizes well. Residual plots and prediction accuracy charts were also analyzed. The final model was selected based on the lowest RMSE and highest R-squared score.

Model performance was assessed using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared. Cross-validation was used to ensure that the model generalizes well. Residual plots and prediction accuracy charts were also analyzed. The final model was selected based on the lowest RMSE and highest R-squared score.

Model performance was assessed using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared. Cross-validation was used to ensure that the model generalizes well. Residual plots and prediction accuracy charts were also analyzed. The final model was selected based on the lowest RMSE and highest R-squared score.

Model performance was assessed using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared. Cross-validation was used to ensure that the model generalizes well. Residual plots and prediction accuracy charts were also analyzed. The final model was selected based on the lowest RMSE and highest R-squared score.

Model performance was assessed using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared. Cross-validation was used to ensure that the model generalizes well. Residual plots and prediction accuracy charts were also analyzed. The final model was selected based on the lowest RMSE and highest R-squared score.

Model performance was assessed using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared. Cross-validation was used to ensure that the model generalizes well. Residual plots and prediction accuracy charts were also analyzed. The final model was selected based on the lowest RMSE and highest R-squared score.

Model performance was assessed using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared. Cross-validation was used to ensure that the model generalizes well. Residual plots and prediction accuracy charts were also analyzed. The final model was selected based on the lowest RMSE and highest R-squared score.

Model performance was assessed using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared. Cross-validation was used to ensure that the model generalizes well. Residual plots and prediction accuracy charts were also analyzed. The final model was selected based on the lowest RMSE and highest R-squared score.

Model performance was assessed using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared. Cross-validation was used to ensure that the model generalizes well. Residual plots and prediction accuracy charts were also analyzed. The final model was selected based on the lowest RMSE and highest R-squared score.

Model performance was assessed using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared. Cross-validation was used to ensure that the model generalizes well. Residual plots and prediction accuracy charts were also analyzed. The final model was selected based on the lowest RMSE and highest R-squared score.

Model performance was assessed using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared. Cross-validation was used to ensure that the model generalizes well. Residual plots and prediction accuracy charts were also analyzed. The final model was selected based on the lowest RMSE and highest R-squared score.

Model performance was assessed using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared. Cross-validation was used to ensure that the model generalizes well. Residual plots and prediction accuracy charts were also analyzed. The final model was selected based on the lowest RMSE and highest R-squared score.

# 10. Hyperparameter Tuning

To optimize the model, hyperparameters were tuned using GridSearchCV. Parameters such as number of estimators, max depth, and learning rate were tested for models like Random Forest and XGBoost. The best set of parameters was selected based on cross-validated performance. Tuning significantly improved the accuracy and robustness of the final model.

To optimize the model, hyperparameters were tuned using GridSearchCV. Parameters such as number of estimators, max depth, and learning rate were tested for models like Random Forest and XGBoost. The best set of parameters was selected based on cross-validated performance. Tuning significantly improved the accuracy and robustness of the final model.

To optimize the model, hyperparameters were tuned using GridSearchCV. Parameters such as number of estimators, max depth, and learning rate were tested for models like Random Forest and XGBoost. The best set of parameters was selected based on cross-validated performance. Tuning significantly improved the accuracy and robustness of the final model.

To optimize the model, hyperparameters were tuned using GridSearchCV. Parameters such as number of estimators, max depth, and learning rate were tested for models like Random Forest and XGBoost. The best set of parameters was selected based on cross-validated performance. Tuning significantly improved the accuracy and robustness of the final model.

To optimize the model, hyperparameters were tuned using GridSearchCV. Parameters such as number of estimators, max depth, and learning rate were tested for models like Random Forest and XGBoost. The best set of parameters was selected based on cross-validated performance. Tuning significantly improved the accuracy and robustness of the final model.

To optimize the model, hyperparameters were tuned using GridSearchCV. Parameters such as number of estimators, max depth, and learning rate were tested for models like Random Forest and XGBoost. The best set of parameters was selected based on cross-validated performance. Tuning significantly improved the accuracy and robustness of the final model.

To optimize the model, hyperparameters were tuned using GridSearchCV. Parameters such as number of estimators, max depth, and learning rate were tested for models like Random Forest and XGBoost. The best set of parameters was selected based on cross-validated performance. Tuning significantly improved the accuracy and robustness of the final model.

To optimize the model, hyperparameters were tuned using GridSearchCV. Parameters such as number of estimators, max depth, and learning rate were tested for models like Random Forest and XGBoost. The best set of parameters was selected based on cross-validated performance. Tuning significantly improved the accuracy and robustness of the final model.

To optimize the model, hyperparameters were tuned using GridSearchCV. Parameters such as number of estimators, max depth, and learning rate were tested for models like Random Forest and XGBoost. The best set of parameters was selected based on cross-validated performance. Tuning significantly improved the accuracy and robustness of the final model.

To optimize the model, hyperparameters were tuned using GridSearchCV. Parameters such as number of estimators, max depth, and learning rate were tested for models like Random Forest and XGBoost. The best set of parameters was selected based on cross-validated performance. Tuning significantly improved the accuracy and robustness of the final model.

To optimize the model, hyperparameters were tuned using GridSearchCV. Parameters such as number of estimators, max depth, and learning rate were tested for models like Random Forest and XGBoost. The best set of parameters was selected based on cross-validated performance. Tuning significantly improved the accuracy and robustness of the final model.

To optimize the model, hyperparameters were tuned using GridSearchCV. Parameters such as number of estimators, max depth, and learning rate were tested for models like Random Forest and XGBoost. The best set of parameters was selected based on cross-validated performance. Tuning significantly improved the accuracy and robustness of the final model.

# 11. Predictions on Test Data

After finalizing the model, the test dataset was processed using the same preprocessing pipeline. Predictions were generated for the test set and saved in CSV format. These predictions can be used for real-world applications such as estimating property values or participating in housing price prediction competitions like those on Kaggle.

After finalizing the model, the test dataset was processed using the same preprocessing pipeline. Predictions were generated for the test set and saved in CSV format. These predictions can be used for real-world applications such as estimating property values or participating in housing price prediction competitions like those on Kaggle.

After finalizing the model, the test dataset was processed using the same preprocessing pipeline. Predictions were generated for the test set and saved in CSV format. These predictions can be used for real-world applications such as estimating property values or participating in housing price prediction competitions like those on Kaggle.

After finalizing the model, the test dataset was processed using the same preprocessing pipeline. Predictions were generated for the test set and saved in CSV format. These predictions can be used for real-world applications such as estimating property values or participating in housing price prediction competitions like those on Kaggle.

After finalizing the model, the test dataset was processed using the same preprocessing pipeline. Predictions were generated for the test set and saved in CSV format. These predictions can be used for real-world applications such as estimating property values or participating in housing price prediction competitions like those on Kaggle.

After finalizing the model, the test dataset was processed using the same preprocessing pipeline. Predictions were generated for the test set and saved in CSV format. These predictions can be used for real-world applications such as estimating property values or participating in housing price prediction competitions like those on Kaggle.

After finalizing the model, the test dataset was processed using the same preprocessing pipeline. Predictions were generated for the test set and saved in CSV format. These predictions can be used for real-world applications such as estimating property values or participating in housing price prediction competitions like those on Kaggle.

After finalizing the model, the test dataset was processed using the same preprocessing pipeline. Predictions were generated for the test set and saved in CSV format. These predictions can be used for real-world applications such as estimating property values or participating in housing price prediction competitions like those on Kaggle.

After finalizing the model, the test dataset was processed using the same preprocessing pipeline. Predictions were generated for the test set and saved in CSV format. These predictions can be used for real-world applications such as estimating property values or participating in housing price prediction competitions like those on Kaggle.

After finalizing the model, the test dataset was processed using the same preprocessing pipeline. Predictions were generated for the test set and saved in CSV format. These predictions can be used for real-world applications such as estimating property values or participating in housing price prediction competitions like those on Kaggle.

After finalizing the model, the test dataset was processed using the same preprocessing pipeline. Predictions were generated for the test set and saved in CSV format. These predictions can be used for real-world applications such as estimating property values or participating in housing price prediction competitions like those on Kaggle.

After finalizing the model, the test dataset was processed using the same preprocessing pipeline. Predictions were generated for the test set and saved in CSV format. These predictions can be used for real-world applications such as estimating property values or participating in housing price prediction competitions like those on Kaggle.

# 12. Conclusion

This project demonstrated the complete machine learning pipeline from data exploration to model deployment for predicting house prices. The final model showed high accuracy and robustness. Important insights were derived from the data regarding factors that influence house prices. The project highlights the potential of data science in solving real-world business problems.

This project demonstrated the complete machine learning pipeline from data exploration to model deployment for predicting house prices. The final model showed high accuracy and robustness. Important insights were derived from the data regarding factors that influence house prices. The project highlights the potential of data science in solving real-world business problems.

This project demonstrated the complete machine learning pipeline from data exploration to model deployment for predicting house prices. The final model showed high accuracy and robustness. Important insights were derived from the data regarding factors that influence house prices. The project highlights the potential of data science in solving real-world business problems.

This project demonstrated the complete machine learning pipeline from data exploration to model deployment for predicting house prices. The final model showed high accuracy and robustness. Important insights were derived from the data regarding factors that influence house prices. The project highlights the potential of data science in solving real-world business problems.

This project demonstrated the complete machine learning pipeline from data exploration to model deployment for predicting house prices. The final model showed high accuracy and robustness. Important insights were derived from the data regarding factors that influence house prices. The project highlights the potential of data science in solving real-world business problems.

This project demonstrated the complete machine learning pipeline from data exploration to model deployment for predicting house prices. The final model showed high accuracy and robustness. Important insights were derived from the data regarding factors that influence house prices. The project highlights the potential of data science in solving real-world business problems.

This project demonstrated the complete machine learning pipeline from data exploration to model deployment for predicting house prices. The final model showed high accuracy and robustness. Important insights were derived from the data regarding factors that influence house prices. The project highlights the potential of data science in solving real-world business problems.

This project demonstrated the complete machine learning pipeline from data exploration to model deployment for predicting house prices. The final model showed high accuracy and robustness. Important insights were derived from the data regarding factors that influence house prices. The project highlights the potential of data science in solving real-world business problems.

This project demonstrated the complete machine learning pipeline from data exploration to model deployment for predicting house prices. The final model showed high accuracy and robustness. Important insights were derived from the data regarding factors that influence house prices. The project highlights the potential of data science in solving real-world business problems.

This project demonstrated the complete machine learning pipeline from data exploration to model deployment for predicting house prices. The final model showed high accuracy and robustness. Important insights were derived from the data regarding factors that influence house prices. The project highlights the potential of data science in solving real-world business problems.

This project demonstrated the complete machine learning pipeline from data exploration to model deployment for predicting house prices. The final model showed high accuracy and robustness. Important insights were derived from the data regarding factors that influence house prices. The project highlights the potential of data science in solving real-world business problems.

This project demonstrated the complete machine learning pipeline from data exploration to model deployment for predicting house prices. The final model showed high accuracy and robustness. Important insights were derived from the data regarding factors that influence house prices. The project highlights the potential of data science in solving real-world business problems.

# 13. Limitations & Future Work

While the model performs well, it has limitations such as reliance on historical data which may not reflect future trends. Feature selection and data quality greatly influence predictions. Future work includes incorporating external data such as crime rates, school quality, and economic indicators. Deployment as a web application for end-users is another possible extension.

While the model performs well, it has limitations such as reliance on historical data which may not reflect future trends. Feature selection and data quality greatly influence predictions. Future work includes incorporating external data such as crime rates, school quality, and economic indicators. Deployment as a web application for end-users is another possible extension.

While the model performs well, it has limitations such as reliance on historical data which may not reflect future trends. Feature selection and data quality greatly influence predictions. Future work includes incorporating external data such as crime rates, school quality, and economic indicators. Deployment as a web application for end-users is another possible extension.

While the model performs well, it has limitations such as reliance on historical data which may not reflect future trends. Feature selection and data quality greatly influence predictions. Future work includes incorporating external data such as crime rates, school quality, and economic indicators. Deployment as a web application for end-users is another possible extension.

While the model performs well, it has limitations such as reliance on historical data which may not reflect future trends. Feature selection and data quality greatly influence predictions. Future work includes incorporating external data such as crime rates, school quality, and economic indicators. Deployment as a web application for end-users is another possible extension.

While the model performs well, it has limitations such as reliance on historical data which may not reflect future trends. Feature selection and data quality greatly influence predictions. Future work includes incorporating external data such as crime rates, school quality, and economic indicators. Deployment as a web application for end-users is another possible extension.

While the model performs well, it has limitations such as reliance on historical data which may not reflect future trends. Feature selection and data quality greatly influence predictions. Future work includes incorporating external data such as crime rates, school quality, and economic indicators. Deployment as a web application for end-users is another possible extension.

While the model performs well, it has limitations such as reliance on historical data which may not reflect future trends. Feature selection and data quality greatly influence predictions. Future work includes incorporating external data such as crime rates, school quality, and economic indicators. Deployment as a web application for end-users is another possible extension.

While the model performs well, it has limitations such as reliance on historical data which may not reflect future trends. Feature selection and data quality greatly influence predictions. Future work includes incorporating external data such as crime rates, school quality, and economic indicators. Deployment as a web application for end-users is another possible extension.

While the model performs well, it has limitations such as reliance on historical data which may not reflect future trends. Feature selection and data quality greatly influence predictions. Future work includes incorporating external data such as crime rates, school quality, and economic indicators. Deployment as a web application for end-users is another possible extension.

While the model performs well, it has limitations such as reliance on historical data which may not reflect future trends. Feature selection and data quality greatly influence predictions. Future work includes incorporating external data such as crime rates, school quality, and economic indicators. Deployment as a web application for end-users is another possible extension.

While the model performs well, it has limitations such as reliance on historical data which may not reflect future trends. Feature selection and data quality greatly influence predictions. Future work includes incorporating external data such as crime rates, school quality, and economic indicators. Deployment as a web application for end-users is another possible extension.

# 14. References

1. Kaggle Housing Price Dataset  
2. Scikit-learn Documentation  
3. Python Data Science Handbook by Jake VanderPlas  
4. Blogs and tutorials on feature engineering and model selection  
5. Machine learning courses and webinars

1. Kaggle Housing Price Dataset  
2. Scikit-learn Documentation  
3. Python Data Science Handbook by Jake VanderPlas  
4. Blogs and tutorials on feature engineering and model selection  
5. Machine learning courses and webinars

1. Kaggle Housing Price Dataset  
2. Scikit-learn Documentation  
3. Python Data Science Handbook by Jake VanderPlas  
4. Blogs and tutorials on feature engineering and model selection  
5. Machine learning courses and webinars

1. Kaggle Housing Price Dataset  
2. Scikit-learn Documentation  
3. Python Data Science Handbook by Jake VanderPlas  
4. Blogs and tutorials on feature engineering and model selection  
5. Machine learning courses and webinars

1. Kaggle Housing Price Dataset  
2. Scikit-learn Documentation  
3. Python Data Science Handbook by Jake VanderPlas  
4. Blogs and tutorials on feature engineering and model selection  
5. Machine learning courses and webinars

1. Kaggle Housing Price Dataset  
2. Scikit-learn Documentation  
3. Python Data Science Handbook by Jake VanderPlas  
4. Blogs and tutorials on feature engineering and model selection  
5. Machine learning courses and webinars

1. Kaggle Housing Price Dataset  
2. Scikit-learn Documentation  
3. Python Data Science Handbook by Jake VanderPlas  
4. Blogs and tutorials on feature engineering and model selection  
5. Machine learning courses and webinars

1. Kaggle Housing Price Dataset  
2. Scikit-learn Documentation  
3. Python Data Science Handbook by Jake VanderPlas  
4. Blogs and tutorials on feature engineering and model selection  
5. Machine learning courses and webinars

1. Kaggle Housing Price Dataset  
2. Scikit-learn Documentation  
3. Python Data Science Handbook by Jake VanderPlas  
4. Blogs and tutorials on feature engineering and model selection  
5. Machine learning courses and webinars

1. Kaggle Housing Price Dataset  
2. Scikit-learn Documentation  
3. Python Data Science Handbook by Jake VanderPlas  
4. Blogs and tutorials on feature engineering and model selection  
5. Machine learning courses and webinars

1. Kaggle Housing Price Dataset  
2. Scikit-learn Documentation  
3. Python Data Science Handbook by Jake VanderPlas  
4. Blogs and tutorials on feature engineering and model selection  
5. Machine learning courses and webinars

1. Kaggle Housing Price Dataset  
2. Scikit-learn Documentation  
3. Python Data Science Handbook by Jake VanderPlas  
4. Blogs and tutorials on feature engineering and model selection  
5. Machine learning courses and webinars

# 15. Appendix (Code Snippets)

Below is an example code snippet for preprocessing data:  
  
import pandas as pd  
from sklearn.impute import SimpleImputer  
from sklearn.preprocessing import OneHotEncoder  
  
# Load data  
train\_df = pd.read\_csv('train.csv')  
  
# Impute missing values  
imputer = SimpleImputer(strategy='median')  
numeric\_df = train\_df.select\_dtypes(include=['number'])  
train\_df[numeric\_df.columns] = imputer.fit\_transform(numeric\_df)

Below is an example code snippet for preprocessing data:  
  
import pandas as pd  
from sklearn.impute import SimpleImputer  
from sklearn.preprocessing import OneHotEncoder  
  
# Load data  
train\_df = pd.read\_csv('train.csv')  
  
# Impute missing values  
imputer = SimpleImputer(strategy='median')  
numeric\_df = train\_df.select\_dtypes(include=['number'])  
train\_df[numeric\_df.columns] = imputer.fit\_transform(numeric\_df)

Below is an example code snippet for preprocessing data:  
  
import pandas as pd  
from sklearn.impute import SimpleImputer  
from sklearn.preprocessing import OneHotEncoder  
  
# Load data  
train\_df = pd.read\_csv('train.csv')  
  
# Impute missing values  
imputer = SimpleImputer(strategy='median')  
numeric\_df = train\_df.select\_dtypes(include=['number'])  
train\_df[numeric\_df.columns] = imputer.fit\_transform(numeric\_df)

Below is an example code snippet for preprocessing data:  
  
import pandas as pd  
from sklearn.impute import SimpleImputer  
from sklearn.preprocessing import OneHotEncoder  
  
# Load data  
train\_df = pd.read\_csv('train.csv')  
  
# Impute missing values  
imputer = SimpleImputer(strategy='median')  
numeric\_df = train\_df.select\_dtypes(include=['number'])  
train\_df[numeric\_df.columns] = imputer.fit\_transform(numeric\_df)

Below is an example code snippet for preprocessing data:  
  
import pandas as pd  
from sklearn.impute import SimpleImputer  
from sklearn.preprocessing import OneHotEncoder  
  
# Load data  
train\_df = pd.read\_csv('train.csv')  
  
# Impute missing values  
imputer = SimpleImputer(strategy='median')  
numeric\_df = train\_df.select\_dtypes(include=['number'])  
train\_df[numeric\_df.columns] = imputer.fit\_transform(numeric\_df)

Below is an example code snippet for preprocessing data:  
  
import pandas as pd  
from sklearn.impute import SimpleImputer  
from sklearn.preprocessing import OneHotEncoder  
  
# Load data  
train\_df = pd.read\_csv('train.csv')  
  
# Impute missing values  
imputer = SimpleImputer(strategy='median')  
numeric\_df = train\_df.select\_dtypes(include=['number'])  
train\_df[numeric\_df.columns] = imputer.fit\_transform(numeric\_df)

Below is an example code snippet for preprocessing data:  
  
import pandas as pd  
from sklearn.impute import SimpleImputer  
from sklearn.preprocessing import OneHotEncoder  
  
# Load data  
train\_df = pd.read\_csv('train.csv')  
  
# Impute missing values  
imputer = SimpleImputer(strategy='median')  
numeric\_df = train\_df.select\_dtypes(include=['number'])  
train\_df[numeric\_df.columns] = imputer.fit\_transform(numeric\_df)

Below is an example code snippet for preprocessing data:  
  
import pandas as pd  
from sklearn.impute import SimpleImputer  
from sklearn.preprocessing import OneHotEncoder  
  
# Load data  
train\_df = pd.read\_csv('train.csv')  
  
# Impute missing values  
imputer = SimpleImputer(strategy='median')  
numeric\_df = train\_df.select\_dtypes(include=['number'])  
train\_df[numeric\_df.columns] = imputer.fit\_transform(numeric\_df)

Below is an example code snippet for preprocessing data:  
  
import pandas as pd  
from sklearn.impute import SimpleImputer  
from sklearn.preprocessing import OneHotEncoder  
  
# Load data  
train\_df = pd.read\_csv('train.csv')  
  
# Impute missing values  
imputer = SimpleImputer(strategy='median')  
numeric\_df = train\_df.select\_dtypes(include=['number'])  
train\_df[numeric\_df.columns] = imputer.fit\_transform(numeric\_df)

Below is an example code snippet for preprocessing data:  
  
import pandas as pd  
from sklearn.impute import SimpleImputer  
from sklearn.preprocessing import OneHotEncoder  
  
# Load data  
train\_df = pd.read\_csv('train.csv')  
  
# Impute missing values  
imputer = SimpleImputer(strategy='median')  
numeric\_df = train\_df.select\_dtypes(include=['number'])  
train\_df[numeric\_df.columns] = imputer.fit\_transform(numeric\_df)

Below is an example code snippet for preprocessing data:  
  
import pandas as pd  
from sklearn.impute import SimpleImputer  
from sklearn.preprocessing import OneHotEncoder  
  
# Load data  
train\_df = pd.read\_csv('train.csv')  
  
# Impute missing values  
imputer = SimpleImputer(strategy='median')  
numeric\_df = train\_df.select\_dtypes(include=['number'])  
train\_df[numeric\_df.columns] = imputer.fit\_transform(numeric\_df)

Below is an example code snippet for preprocessing data:  
  
import pandas as pd  
from sklearn.impute import SimpleImputer  
from sklearn.preprocessing import OneHotEncoder  
  
# Load data  
train\_df = pd.read\_csv('train.csv')  
  
# Impute missing values  
imputer = SimpleImputer(strategy='median')  
numeric\_df = train\_df.select\_dtypes(include=['number'])  
train\_df[numeric\_df.columns] = imputer.fit\_transform(numeric\_df)