Housing Price Prediction

Muhammed Shadil

Submitted for the Degree of Master of Science in

MSc Data Science and Analytics

Department of Computer Science
Royal Holloway University of London
Egham, Surrey TW20 0EX, UK

Declaration

This report has been prepared on the basis of my own work. Where other published and unpublished source materials have been used, these have been acknowledged.

Word Count: 12660

Student Name: Muhammed Shadil

Date of Submission: 30/08/2023

Signature:

Acknowledgement

I would like to express my sincere gratitude to all those who have supported and guided me throughout the duration of this project. Their contributions, encouragement, and insights have been invaluable in shaping the outcome of this work.

I extend my heartfelt thanks to my supervisor, **Volodya Vovk**, for their unwavering support, expertise, and guidance. Their mentorship has been instrumental in steering me through the challenges of this project.

I am also grateful to the **Royal Holloway University of London** for providing me with the necessary resources, facilities, and access to data that were crucial for the successful completion of this project.

I would like to thank my family and friends for their constant encouragement and understanding during this project. Their support motivated me to persevere through the various stages of this endeavour.

Lastly, I want to acknowledge all the authors and researchers whose work and publications I have consulted in the process of conducting this project. Their insights and findings have provided a strong foundation for the analysis and interpretation presented in this work.

Muhammed Shadil

100996119

Date: 28/08/2023

List of Figures

Figure 1 House prices over next five years 9
Figure 2 Features affecting house prices 10
Figure 3 Machine learning in house price prediction 11
Figure 4 Research methodology 16
Figure 5 Data preprocessing 17
Figure 6 Linear Regression 20
Figure 7 Ridge Regression 22
Figure 8 Polynomial Regression 24
Figure 9 Pair plot of features in the data 28
Figure 10 Correlation heatmap of the features in the data 30
Figure 11 Hist plot of features in the data 33

List of Tables

Table 1 Performance of linear regression model 34
Table 2 Performance of ridge regression model 36
Table 3 Performance of polynomial regression model 37
Table 4 Comparison of performance of all models 38

Table of Contents

Declaration 2
Acknowledgement 3
List of Figures 4
List of Tables 5
Abstract 8
Chapter 1: Introduction 9
1.1 Overview 9
1.2 History of House Price Prediction 9
1.3 Role of Machine Learning in House Price Prediction 11
1.4 Motivation 12
1.5 Research Questions 12
1.6 Aim 12
1.7 Objectives 13
Chapter 2: Literature Review 14
Chapter 3: Research Methodology 16
3.1 Dataset 16
3.2 Data Preprocessing 17
3.2.1 Handling Missing Values 17
3.2.2 Encoding Categorical Columns 18
3.2.3 Feature Selection 18
3.3 Exploratory Data Analysis 18
3.3.1 Pair Plot 18
3.3.2 Correlation Heatmap 18
3.3.3 Histograms 19
3.4 Building Machine Learning Models 19
3.4.1 Linear Regression Model Algorithm 20
3.4.2 Ridge Regression Model Algorithm 22
3.4.3 Polynomial Regression Model Algorithm 24
3.5 Coding Practices Followed 25
Chapter 4: Results 27
4.1 Hardware Requirements 27
4.2 Software Requirements 27
4.3 Exploratory Data Analysis 27
4.3.1 Pair Plot 28
4.3.2 Correlation Heatmap 29

4.3.3 Hist Plot of Features 32
4.4 Performance of Machine Learning Models 34
4.4.1 Performance of Linear Regression Model 34
4.4.2 Performance of Ridge Regression Model 35
4.4.3 Performance of Polynomial Regression Model 36
4.4.4 Comparison of Performance of all Machine Learning Models 38
Chapter 5: Professional Issues 39
5.1 Professional Issue - Privacy in Data Handling 39
5.2 Ethical Importance: 39
5.3 Practical Importance: 39
5.4 Example from the Public Domain: 39
Chapter 6: Self-Assessment 40
6.1 Reflection on the Project: 40
6.2 Project Journey: 40
6.3 Things that went right: 40
6.4 Things that went wrong: 40
Chapter 7: Conclusion 41
Chapter 8: Future Work 43
Chapter 9: How to Run Code in Google Colab ..... **Error! Bookmark not defined.**
Chapter 10: References 48
Appendix 51

Abstract

This project focuses on the development and evaluation of predictive models for housing price estimation using machine learning techniques. The study begins with data collection and preprocessing, where a real-world data set is cleaned and prepared for analysis. Exploratory data analysis (EDA) is conducted to gain insights into feature distributions and relationships.

Subsequently, three different regression models are implemented and assessed: Linear Regression, Ridge Regression, and Polynomial Regression. These models are trained on the pre-processed data, and hyperparameter tuning is performed to optimize their predictive performance.

The models are evaluated using the R-squared metric to measure their ability to predict housing prices accurately. Results indicate that all three models show promise, with R-squared values ranging from 0.6357 to 0.64, suggesting a strong correlation between the chosen features and housing prices.

This project demonstrates the effectiveness of machine learning in predicting housing prices and highlights the potential for further improvements in model performance. The methodologies employed here can serve as a foundation for future work in housing price prediction and related fields.

Chapter 1: Introduction

1.1 Overview

House price prediction plays a pivotal role in real estate markets, investment decisions, and financial planning. The ability to accurately estimate the value of a property based on various attributes empowers individuals, real estate agents, and policymakers to make informed choices (FOLGER, 2023). The housing market is characterized by its complexity, influenced by factors ranging from the size and location of the property to the number of bedrooms, bathrooms, and other amenities (Team, 2023). These features collectively contribute to the property's value, making it an ideal scenario for applying regression algorithms. In this project, we delve into the realm of predictive modelling by implementing two fundamental regression techniques, namely linear regression and ridge regression, to predict house prices. Our aim is to harness the power of data-driven methodologies to build models that can predict house prices based on a set of input features.

Figure 1 (Barber, 2022)

1.2 History of House Price Prediction

Predicting house prices has been a longstanding endeavour, driven by the need to anticipate and understand real estate market trends, make informed investment decisions, and assist individuals in making crucial financial choices. The history of house price prediction spans several decades, with technological advancements and data-driven methodologies reshaping the landscape of predictive modelling. This historical journey offers insights into the evolution of house price prediction and its significance in today's data-driven world.

Before the advent of modern data analysis and computational techniques, house price prediction largely relied on local expertise, intuition, and experience. Real estate agents, appraisers, and community members played a pivotal role in estimating property values based

on their familiarity with the local market, property conditions, and prevailing economic factors (Geerts, 2023). While these methods provided valuable insights, they often lacked the precision and scalability demanded by a growing and dynamic real estate sector.

Figure 2 (María Paz Cuturi, 2021)

The mid-20th century witnessed the emergence of statistical methods in house price prediction (Corsini, 2009). Linear regression, one of the earliest and simplest regression techniques, gained prominence (Valkov, 2019). Analysts started using statistical models to identify correlations between various property attributes and sale prices. These initial attempts laid the foundation for incorporating quantitative factors into price estimation, moving beyond qualitative assessments.

The digital revolution of the late 20th century transformed house price prediction. The proliferation of digital databases, computer systems, and data analysis tools enabled researchers to process and analyse vast amounts of real estate data (Saha, 2020). This shift catalysed the development of sophisticated predictive models that could consider multiple features simultaneously. Machine learning algorithms emerged as powerful tools for predicting house prices based on historical data patterns (Winky K.O. Ho, 2020).

In recent years, regression techniques have regained prominence due to their interpretability and ability to handle complex datasets. Linear regression, coupled with regularization techniques like ridge and lasso regression, gained traction. These methods strike a balance between incorporating multiple features and preventing overfitting (Paul, 2018).

As artificial intelligence continues to advance, predictive analytics in the real estate domain is poised for groundbreaking advancements (Frąckiewicz, 2023). Incorporating AI-driven algorithms, predictive models can consider an even broader range of factors, including sentiment analysis of market trends, macroeconomic indicators, and urban development plans. Moreover, the integration of geospatial data, coupled with AI-driven image analysis, promises to revolutionize property valuation by capturing the finer details of a property and its surroundings.

1.3 Role of Machine Learning in House Price Prediction

Figure 3 (Anon., n.d.)

In the realm of real estate, accurate property valuation is a critical aspect that drives investment decisions, market analysis, and financial planning. Machine learning, a subset of artificial intelligence, has emerged as a transformative force in the field of house price prediction. Leveraging algorithms, data analysis, and predictive modelling, machine learning techniques offer unparalleled insights into property values (Golchha, 2023), making them an indispensable tool for real estate professionals, investors, and homeowners. This article explores the multifaceted role of machine learning in the context of house price prediction.

Machine learning algorithms thrive on data, and the real estate market generates an abundance of it. By harnessing historical sales data, property features, location attributes, economic indicators, and more, machine learning models can uncover hidden patterns and relationships. This data-driven approach provides a holistic understanding of the factors influencing property values, empowering stakeholders with actionable insights.

House price prediction involves complex relationships between a myriad of variables (Max Hansson, 2023). Traditional methods often struggle to handle intricate interactions, leading to oversimplified models. Machine learning algorithms, on the other hand, excel at capturing nonlinear relationships (Ribeiro, 2023), enabling more accurate predictions by considering multiple factors simultaneously. This ability to handle complexity contributes to better predictive accuracy.

Determining which features have the most significant impact on property values is crucial for effective prediction. Machine learning models excel at identifying feature importance (Li, 2022). Through techniques such as feature selection and feature engineering, these models highlight attributes like location, square footage, amenities, and neighbourhood characteristics that play a pivotal role in determining property prices.

Incomplete or missing data is a common challenge in real estate datasets. Machine learning algorithms can handle missing data by imputing values based on patterns observed in the available data. This ensures that the predictive model is robust even when dealing with incomplete information.

The real estate market is dynamic, with trends and influences changing over time. Machine learning models can adapt to evolving market conditions by continuously learning from new data. This flexibility allows the models to remain relevant and accurate even as market dynamics shift.

As datasets grow larger and more complex, machine learning algorithms can handle the increased computational load. Whether dealing with thousands or millions of data points, these models scale seamlessly to accommodate large datasets, providing accurate predictions regardless of the dataset's size.

Hyperparameters significantly impact the performance of predictive models. Machine learning techniques offer tools for fine-tuning these hyperparameters, resulting in optimized models that deliver the best predictive accuracy. Techniques like grid search and random search help identify the best combination of hyperparameters for a given problem.

As artificial intelligence advances, the integration of cutting-edge techniques like natural language processing, sentiment analysis, and geospatial data analysis holds the promise of even more accurate and comprehensive property valuation. AI-driven insights can provide a deeper understanding of market sentiments, economic trends, and urban development impacts.

1.4 Motivation

The motivation behind this project lies in the practical significance of accurately predicting house prices, a crucial factor in real estate transactions and investments. By implementing linear regression and ridge regression models from scratch, we seek to bridge the gap between theoretical understanding and practical application, gaining insights into the mechanics of these algorithms. This project's educational value extends to skill development in data preprocessing, algorithm implementation, and hyperparameter tuning, contributing to our proficiency in predictive modelling. Furthermore, this endeavour holds the potential to empower stakeholders with better decision-making tools while fostering a deeper appreciation for the role of machine learning in addressing real-world challenges.

1.5 Research Questions

- How do different input features, including property attributes and location-related factors, contribute to the prediction of house prices?
- What is the impact of varying hyperparameters on the accuracy and stability of predictive models?

1.6 Aim

The aim of this project is to implement and explore regression-based predictive modelling techniques for house price prediction, focusing on understanding the interplay between property attributes and price fluctuations. Through the implementation of linear and ridge regression models, this project seeks to delve into the mechanics of these algorithms,

investigate their performance under different hyperparameters, and gain insights into the significance of various input features in influencing house prices.

1.7 Objectives

- **Literature Review:** Begin by thoroughly researching existing literature on house price prediction, regression techniques, and relevant machine learning concepts. This step will provide a comprehensive understanding of the background, challenges, and advancements in the field, guiding the project's direction.
- **Data Collection:** Identify and gather a suitable dataset that contains historical housing data, including features like property size, location, amenities, and sale prices. The quality and relevance of the dataset are crucial for training accurate predictive models.
- **Data Cleaning:** Clean the collected dataset to remove any inconsistencies, missing values, or outliers that might negatively impact the performance of the models. This step ensures that the dataset is ready for analysis and modelling.
- **Exploratory Data Analysis (EDA):** Conduct an exploratory analysis of the dataset to gain insights into the distribution of features, correlations, and trends. EDA helps identify potential patterns and relationships that can guide feature selection and model development.
- **Implementing Machine Learning Models:** Implement linear regression and ridge regression models from scratch using appropriate programming languages and libraries. The implementation should encompass key components like gradient descent, cost functions, and regularization, aligning with the mathematical foundations of these techniques.
- **Tuning the Hyperparameters:** Experiment with different hyperparameter values, such as learning rates and regularization parameters, to optimize the performance of the implemented models. Hyperparameter tuning aims to strike a balance between bias and variance, leading to models that generalize well to unseen data.
- **Evaluating the Performance:** Evaluate the performance of the implemented models using appropriate evaluation metrics such as mean squared error (MSE) or R-squared. Compare the performance of linear and ridge regression models across different hyperparameter settings.
- **Documenting the Report:** Compile a comprehensive report that documents the entire project process. This includes an introduction, background research, methodology, results, conclusions, and future work. The report explains the rationale behind each step, highlight key findings, and offer insights gained from the project.

Each of the above objectives of this projects have been achieved and explained clearly in the subsequent chapters.

Chapter 2: Literature Review

The thesis (AKE, 2022) explores predictive models for property price changes and the underlying factors driving these changes. The study focuses on numerical property characteristics and spatial data, recognizing location's profound impact on property values. By investigating each data type's influence on prediction accuracy, the research highlights their significance in enhancing prediction models. A significant contribution of the thesis lies in demonstrating how software implementation can enhance predictive algorithms for house prices. The research emphasizes the potential of machine learning as an alternative technique for projecting housing values, asserting its viability alongside traditional regression methods. Utilizing the California House Price Prediction dataset, the study constructs models to predict median house values across the state. A range of regression models, including linear, ridge, lasso, Decision Tree, and Random Forest Regression, is applied. Results indicate the Random Forest model's superiority in terms of R-square value and Root Mean Square Error. Consequently, the study concludes that the Random Forest Model is the most suitable for predicting housing prices within this dataset, offering valuable insights for informed real estate decisions.

This research study (ALAN IHRE, 2019) utilized machine learning algorithms, like k-Nearest-Neighbours regression (k-NN) and Random Forest (RF) regression, to predict house prices based on features from the Ames housing dataset. The selection of these algorithms was influenced by prior research, with the aim of comparing their performance for this task. The implementation employed the scikit-learn Python library, computing error metrics to gauge the disparities between actual and predicted sales prices. Optimal hyperparameters were chosen for the algorithms, and the dataset underwent five-fold cross-validation to mitigate potential bias. Employing a grid search algorithm, the study determined the best subset of hyperparameters for both models to enhance prediction accuracy. Ultimately, the results demonstrated that the Random Forest consistently outperformed the k-NN algorithm, yielding smaller errors and proving to be a more suitable prediction model for house prices. However, the study acknowledged limitations, with a mean absolute error of approximately 9% from the mean price, suggesting practical usefulness mainly for basic valuations.

The study (Raul-Tomas Mora-Garcia, 2022) explores the application of machine learning algorithms for predicting house prices and assessing the impact of the COVID-19 pandemic on property values in a Spanish city. With the proliferation of digital technology, vast georeferenced datasets have become accessible, enabling the use of algorithms to discern patterns, make predictions, and facilitate decision-making. The research aims to identify optimal machine learning algorithms for house price prediction and quantify the pandemic's influence on prices. The study's methodology encompasses data preparation, feature engineering, hyperparameter tuning, model evaluation, and interpretation. Ensemble learning algorithms including Gradient Boosting Regressor, Extreme Gradient Boosting, Light Gradient Boosting Machine, random forest, and extra-trees regressor are employed and contrasted with a linear regression model. Georeferenced microdata from Alicante's real estate market, pre and post-COVID-19 declaration, in conjunction with auxiliary data like cadastre records, socioeconomic indicators, economic data, and satellite images, constitute the case study. Results indicate the superiority of machine learning algorithms over linear

models in capturing complexities within real estate data. Bagging-based algorithms exhibit overfitting tendencies, while boosting-based models display superior performance and reduced overfitting. Notably, this research contributes to Spanish real estate literature by utilizing machine learning and microdata to explore the pandemic's impact on house prices, marking one of the early endeavours of its kind.

The research (Ayushi Bhagat, 2023) delves into the dynamic real estate industry, acknowledging its price fluctuations and the potential application of machine learning to predict real estate prices with maximum accuracy. The study centres on employing appropriate machine learning algorithms to forecast real estate prices, encompassing essential parameters and geographical/statistical techniques. The paper outlines the functioning of a house pricing model post application of machine learning techniques and algorithms. Leveraging a dataset sourced from a reputable website, the study utilizes Linear Regression and the sklearn library to enhance accuracy. The research encompasses the model's structural aspects, including data cleaning, outlier handling, feature engineering, dimensionality reduction, hyperparameter tuning using grid search cv, k-fold cross-validation, and more.

The research (Quang Truong, 2019) centres around the House Price Index (HPI) as a tool to gauge housing price fluctuations. Acknowledging that housing prices are intricately linked to factors like location, area, and population, the study underscores the need for additional information beyond HPI to accurately predict individual housing prices. While traditional machine learning methods have been employed extensively for this purpose, the paper highlights a gap in assessing the performance of individual models and exploring lesser known but intricate models. In response, the research aims to comprehensively investigate the impact of various features on prediction methods by employing both traditional and advanced machine learning approaches. By examining multiple advanced models, the study intends to elucidate differences and nuances. Moreover, the research aims to validate diverse techniques in model implementation, with a particular focus on regression, offering an optimistic outlook for achieving accurate housing price predictions.

Chapter 3: Research Methodology

The methodology followed in this project encompassed several key steps to address the research objectives effectively. Beginning with data collection and preprocessing, the dataset underwent rigorous cleaning and transformation to ensure its suitability for modelling. Exploratory data analysis provided critical insights into feature distributions and correlations, informing subsequent model choices. Model development encompassed the implementation of Linear Regression, Ridge Regression, and Polynomial Regression, accompanied by hyperparameter tuning to enhance predictive accuracy. Model evaluation using the R-squared metric gauged their performance on unseen data. The analysis of results and comparisons between models facilitated the identification of their strengths and limitations. This methodology, which is explained in detail in this section, facilitated a comprehensive exploration of housing price prediction and laid the foundation for future work in refining model performance and exploring new predictive techniques.

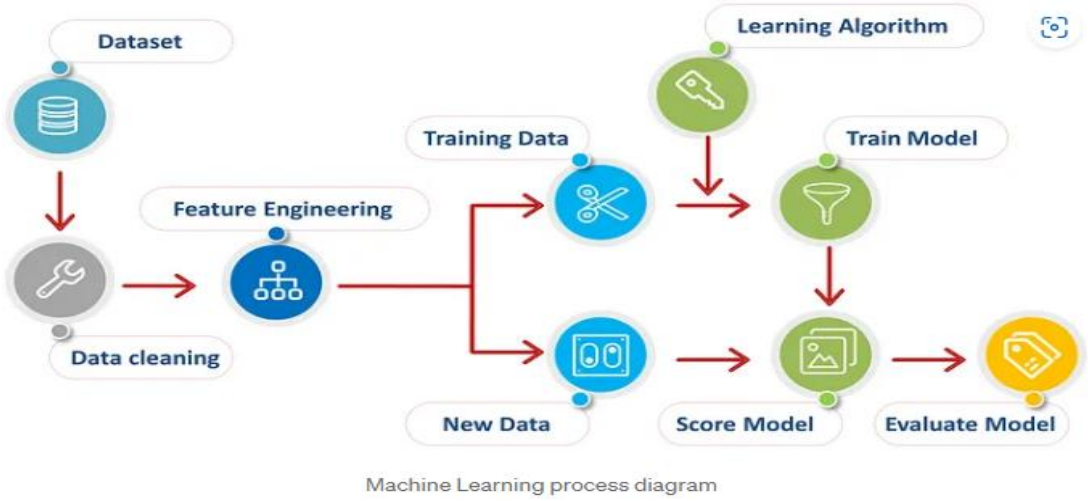


Figure 4 (Afroz Chakure,2019)

3.1 Dataset

The dataset chosen to conduct research is the Housing dataset which is available on Kaggle website. The dataset is open source and is free to use for any study. There is no need of any permissions to download and use the dataset. The dataset contains 20,640 rows and 10 columns. This is a substantial amount of data and is useful in conducting the research. The following is the list of features contained in the dataset.

- **longitude:** This feature represents the geographical longitude of the location of the housing unit. Longitude is a measure of how far east or west a location is on the Earth's surface.
- **latitude:** This feature indicates the geographical latitude of the housing unit's location. Latitude is a measure of how far north or south a location is on the Earth's surface.

- **housing_median_age:** This feature corresponds to the median age of houses in each area. It provides an insight into the age distribution of housing units.
- **total_rooms:** Total rooms refers to the total number of rooms in a housing unit, including bedrooms, living rooms, kitchens, etc.
- **total_bedrooms:** Total bedrooms represents the total number of bedrooms in a housing unit. This feature gives an idea of the accommodation capacity.
- **population:** Population indicates the total number of people residing in the area where the housing unit is located. It provides an understanding of the density of inhabitants.
- **households:** Households signify the total number of separate living spaces or families residing in the area. It can offer insights into the community size.
- **median_income:** Median income represents the median income of households in the area. This is an important socio-economic indicator that can influence housing prices.
- **median_house_value:** Median house value is the target variable of the dataset. It denotes the median value of houses in a specific area. This is the value you're trying to predict with your models.
- **ocean_proximity:** Ocean proximity categorizes the housing unit's location relative to the ocean. It could take values like 'near ocean,' 'near bay,' 'island,' etc. This categorical variable might provide insights into the geographical context.

3.2 Data Preprocessing

Data preprocessing steps like handling missing values, encoding categorical data, and dropping the irrelevant features have been carried out. They are explained below in detail.

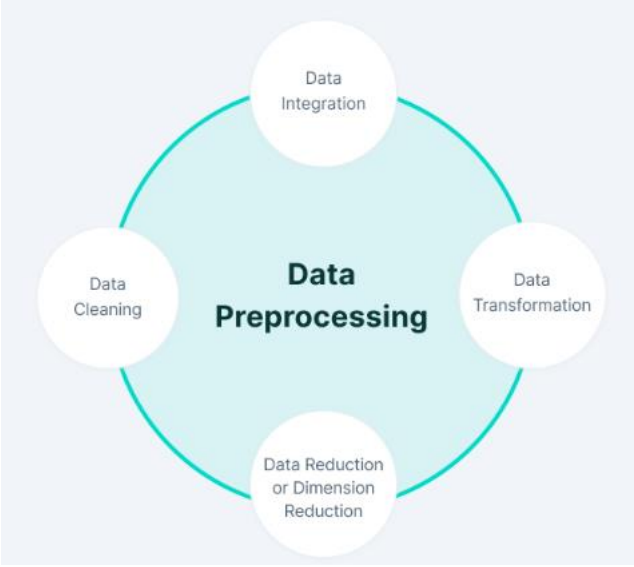


Figure 5 (Baheti, 2021)

3.2.1 Handling Missing Values

The initial step is to deal with any missing values in the dataset. The dataset contains missing values only in “total_bedrooms” column. 207 out of 20640 rows of this column are missing.

The rows with missing values have been dropped from the dataset. The decision to drop rows with missing values is since only around 1 percent of the rows have missing values, and it's assumed that the impact on the overall dataset will be minimal. Dropping rows with missing values ensures that the subsequent analysis and modelling are conducted on complete and reliable data.

3.2.2 Encoding Categorical Columns

The next step involves encoding the categorical column 'ocean_proximity' using OneHotEncoder. This column contains categorical data indicating the proximity of the housing unit to the ocean. OneHotEncoder converts categorical data into numerical values, creating separate columns for each category and assigning binary values (0 or 1) to each category.

3.2.3 Feature Selection

The target column for prediction is set to be the 'median_income' column, which represents the median income of households. Additionally, columns 'longitude' and 'latitude' are dropped from the dataset. This is done because, while these columns represent geographical coordinates, their numerical values alone might not have significant predictive power for the target variable and encoding them directly could lead to erroneous interpretations.

3.3 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is a crucial phase in the data analysis process, and it plays a pivotal role in shaping the direction and success of a data-driven project such as predicting house prices. EDA involves systematically examining and visualizing the data to gain insights, identify patterns, understand relationships between variables, and uncover potential anomalies or outliers.

3.3.1 Pair Plot

A pair plot is a powerful visualization tool that enables us to quickly identify correlations, distributions, and potential outliers in the data. By creating scatter plots for all possible pairs of numerical features, we have gained valuable insights into how different features interact with each other and their potential influence on the target variable ('median_income' in this case).

3.3.2 Correlation Heatmap

Creating a correlation heatmap is another crucial exploratory data analysis (EDA) technique that allows us to visualize and understand the relationships between numerical features in the dataset. A correlation heatmap provides a visual representation of the correlation coefficients between pairs of numerical features in your dataset. Correlation measures the strength and direction of the linear relationship between two variables. This heatmap helps us quickly identify which features are positively correlated, negatively correlated, or not correlated at

all. Correlation analysis is important because it can give us insights into multicollinearity, help select relevant features, and guide in the model building process.

3.3.3 Histograms

Creating histograms of each feature is a fundamental exploratory data analysis (EDA) technique that allows us to visualize the distribution of numerical features in the dataset. Histograms provide a visual representation of the frequency distribution of a single numerical feature. By plotting histograms for each feature, we can quickly understand the spread, central tendency, skewness, and potential presence of outliers in the data. This helps to identify the characteristics of each feature and assess whether any preprocessing steps are required before building predictive models.

3.4 Building Machine Learning Models

The dataset is split into training and test data, where test data is 30 percent of the total data. Machine learning models like Linear Regression, Ridge Regression, and Polynomial Regression are trained with training data, and these models are tested. The following are the algorithms of Linear Regression, Ridge Regression and Polynomial Regression implemented from scratch.

Here the gradient descent approach is used to update the weights and there by achieve the optimal values of weights which corresponds to the minimum mean squared error. There is also another way of finding the optimal weights, which is the Matrix form (closed form solution).

The matrix form provides a direct mathematical solution to finding the optimal model parameters, often denoted as θ (theta). It involves computing the matrix products and inverses to find the values of θ that minimize the cost function (often Mean Squared Error in linear regression).

The formula for θ in matrix form is $\theta = (X^T X)^{-1} X^T y$, where X is the feature matrix and y is the target variable.

It's efficient for small to moderately sized datasets because it doesn't require iterative updates.

The following are the reasons why I have chosen to implement gradient descent.

- **Scalability:** Gradient descent is highly scalable and can handle large datasets efficiently. In contrast, the matrix form becomes computationally expensive as the dataset size increases due to matrix inversion.
- **Memory Efficiency:** For very large datasets, storing the entire feature matrix in memory for matrix operations might not be feasible. Gradient descent operates on mini-batches or even single data points, making it more memory-efficient.
- **Flexibility:** Gradient descent can be adapted to various machine learning problems, including those where closed-form solutions do not exist. It's a versatile optimization method.

- **Online Learning:** For online learning scenarios, where data arrives sequentially, gradient descent is a natural choice. It can continuously update the model with incoming data, while the matrix form typically requires a full pass over the dataset.
- **Regularization:** When dealing with regularization terms like L1 or L2 regularization (as in Ridge or Lasso regression), gradient descent easily incorporates them into the cost function, allowing control over model complexity.

3.4.1 Linear Regression Model Algorithm

The linear regression model is implemented from scratch using a number of functions. They are linear _regression, normalize_features, gradient_descent, compute_cost, predict. Below is the detailed explanation of these functions.

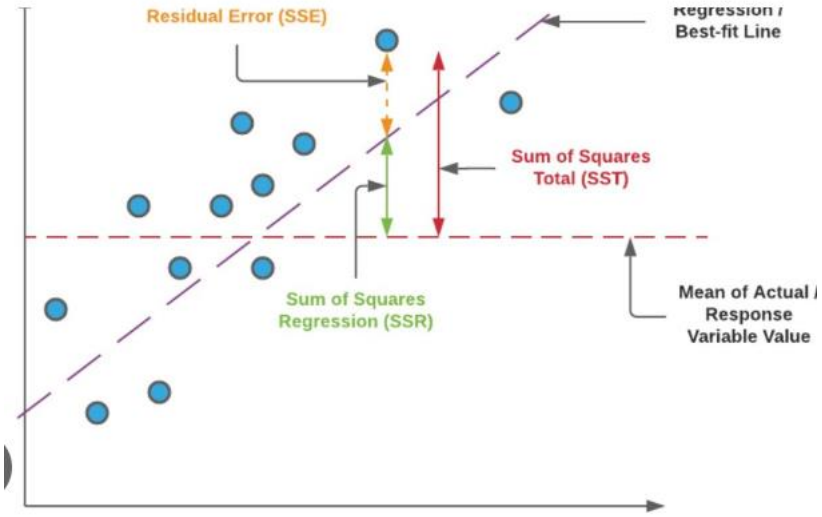


Figure 6 (Kumar, 2023)

Function: linear_regression

Input: Training dataset X_train and target values y_train, learning rate alpha, number of iterations num_iterations.

Output: Optimized parameter vector theta, mean and standard deviation of features (mean, std), cost history cost_history.

Functionality:

- Normalize the features using X_norm, mean, std = normalize_features(X_train).
- Insert a column of ones at the beginning of X_norm for the intercept term.
- Initialize the parameter vector theta with zeros.
- Initialize an empty list cost_history to track the cost function values over iterations.
- For each iteration i from 1 to num_iterations, do:
- Calculate predictions h by computing the dot product of X_norm and theta.
- Update theta using the gradient descent update rule:
- theta = theta - (alpha / m) * (X_norm.T.dot(h - y_train))

- Calculate the cost using the function compute_cost(X_norm, y_train, theta).
- Append the calculated cost to cost_history.
- Return theta, mean, std, and cost_history.

Function: normalize_features

Input: Features matrix X, optional mean mean, and optional standard deviation std.

Output: Normalized features matrix X_norm, computed mean mean, and computed standard deviation std.

Functionality:

- If mean is not provided, compute the mean of each column in X.
- If std is not provided, compute the standard deviation of each column in X.
- Normalize X using (X - mean) / std to obtain X_norm.
- Return X_norm, computed mean, and computed standard deviation.

Function: gradient_descent

Input: Features matrix X, target values y, parameter vector theta, learning rate alpha, number of iterations num_iterations.

Output: Optimized parameter vector theta, cost history cost_history.

Functionality:

- Initialize an empty list cost_history to track the cost function values over iterations.
- For each iteration i from 1 to num_iterations, do:
- Calculate predictions h by computing the dot product of X and theta.
- Update theta using the gradient descent update rule:
- theta = theta - (alpha / m) * (X.T.dot(h - y))
- Calculate the cost using the function compute_cost(X, y, theta).
- Append the calculated cost to cost_history.
- Return theta and cost_history.

Function: compute_cost

Input: Features matrix X, target values y, parameter vector theta.

Output: Cost value cost.

Functionality:

- Calculate predictions h by computing the dot product of X and theta.
- Compute the cost as (1 / (2 * m)) * np.sum((h - y) ** 2).
- Return the computed cost.

Function: predict

Input: New features matrix X, optimized parameter vector theta, mean mean, and standard deviation std.

Output: Predicted target values predictions.

- Normalize the new features using (X - mean) / std.
- Insert a column of ones at the beginning of normalized features for the intercept term.
- Calculate predictions by computing the dot product of normalized features and theta.
- Return the predicted values.

3.4.2 Ridge Regression Model Algorithm

The ridge regression model is implemented from scratch using a number of functions. They are ridge_regression, normalize_features, compute_ridge_cost, predict. Below is the detailed explanation of these functions.

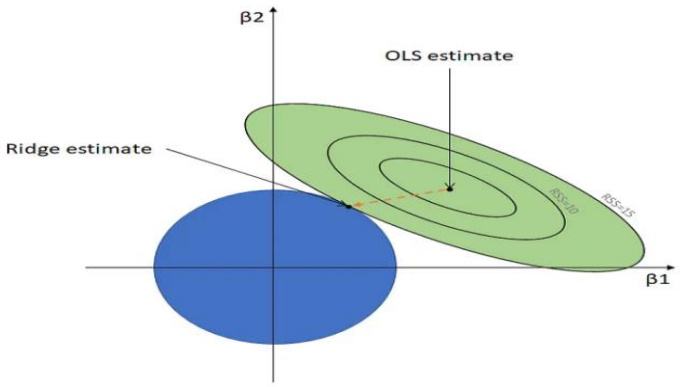


Figure 7 (Qshick,2019)

Function: ridge_regression

Input: Training dataset X_train and target values y_train, learning rate alpha, regularization parameter lambda_, number of iterations num_iterations.

Output: Optimized parameter vector theta, mean and standard deviation of features (mean, std), cost history cost_history.

Algorithm:

- Normalize the features using X_norm, mean, std = normalize_features(X_train).
- Insert a column of ones at the beginning of X_norm for the intercept term.
- Initialize m as the number of training samples and n as the number of features.
- Initialize the parameter vector theta with zeros.
- Initialize an empty list cost_history to track the cost function values over iterations.
- For each iteration i from 1 to num_iterations, do:
- Calculate predictions h by computing the dot product of X_norm and theta.
- Update the first element of theta using the gradient descent update rule:

- theta[0] = theta[0] - (alpha / m) * (X_norm[:, 0].dot(h - y_train))
- Update the remaining elements of theta using the ridge regression update rule:
- theta[1:] = theta[1:] * (1 - alpha * lambda_ / m) - (alpha / m) * (X_norm[:, 1:].T.dot(h - y_train))
- Calculate the cost using the function compute_ridge_cost(X_norm, y_train, theta, lambda_).
- Append the calculated cost to cost_history.
- Return theta, mean, std, and cost_history.

Function: normalize_features

Input: Features matrix X, optional mean mean, and optional standard deviation std.

Output: Normalized features matrix X_norm, computed mean mean, and computed standard deviation std.

Algorithm:

- If mean is not provided, compute the mean of each column in X.
- If std is not provided, compute the standard deviation of each column in X.
- Normalize X using (X - mean) / std to obtain X_norm.
- Return X_norm, computed mean, and computed standard deviation.

Function: compute_ridge_cost

Input: Features matrix X, target values y, parameter vector theta, regularization parameter lambda_.

Output: Cost value cost.

- Calculate predictions h by computing the dot product of X and theta.
- Compute the cost as (1 / (2 * m)) * np.sum((h - y) ** 2) + (lambda_ / (2 * m)) * np.sum(theta[1:] ** 2).
- Return the computed cost.

Function: predict

Input: New features matrix X, optimized parameter vector theta, mean mean, and standard deviation std.

Output: Predicted target values predictions.

Algorithm:

- Normalize the new features using (X - mean) / std.
- Insert a column of ones at the beginning of normalized features for the intercept term.

- Calculate predictions by computing the dot product of normalized features and theta.
- Return the predicted values.

3.4.3 Polynomial Regression Model Algorithm

Implementation of polynomial regression consists of polynomial_regression function and add_polynomial_features function. Initially the new features, which are the exponents of the original features are added to the dataset and ridge regression is applied on it. The implementation of ridge_regression is same as that explained in the above section.

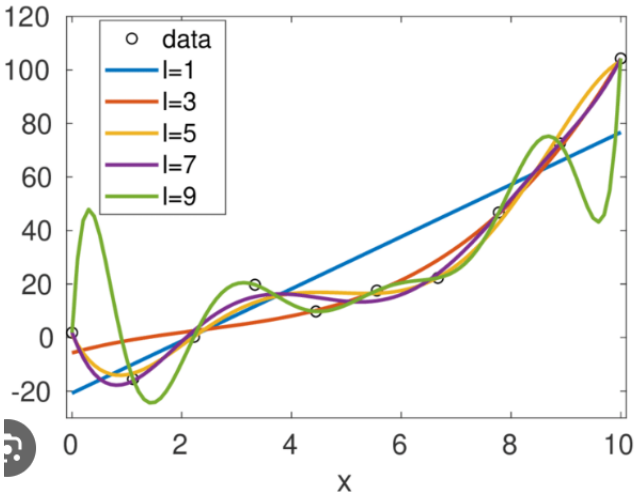


Figure 8 (Camporeale, n.d.)

Function: polynomial_regression

Input: Training dataset X_train and target values y_train, validation dataset X_validation, test dataset X_test, maximum polynomial degree max_degree, learning rate alpha, regularization parameter lambda_, number of iterations num_iterations.

Output: Optimized parameter vector theta, mean and standard deviation of features (mean, std), cost history cost_history.

Algorithm:

- Define the hyperparameter search space with alphas, lambdas, and num_iterations.
- Generate polynomial features up to the specified max_degree using the function add_polynomial_features(X, max_degree).
- Apply polynomial feature transformation to training, validation, and test datasets to get X_train_poly, X_validation_poly, and X_test_poly.
- For each combination of hyperparameters, do:
- Perform Ridge Regression on X_train_poly and y_train using the function ridge_regression(X_train_poly, y_train, alpha, lambda_, num_iteration).
- Predict the target values for the validation set using the function predict (X_validation_poly, theta, mean, std).

- Calculate the R2 score for the validation predictions using r2_score (y_validation, y_pred).
- Print the R2 score along with hyperparameter values.

Function: add_polynomial_features

Input: Features matrix X, maximum polynomial degree max_degree.

Output: Transformed features matrix X_poly.

Algorithm:

- Initialize X_poly as a copy of X.
- For each polynomial degree d from 2 to max_degree, do:
- Compute the element-wise power of X to the degree d.
- Stack the computed polynomial features horizontally to X_poly.
- Return the transformed features matrix X_poly.

3.5 Coding Practices Followed

The following coding practices have been strictly followed in this project.

Modularization:

- Code has been broken down into logical functions and classes for specific tasks, such as data preprocessing, model implementation, and evaluation.
- Functions are kept focused and perform a single task, making it easier to read and debug.

Comments and Documentation:

- Adequate comments have been added to explain the purpose of functions, the logic behind complex code blocks, and important variable usage.
- Function docstrings are provided to explain inputs, outputs, and functionalities of functions.

Meaningful Variable and Function Names:

- Variable and function names are chosen descriptively to convey their purpose. This enhances code readability and understanding.

Consistent Indentation:

- Proper indentation is maintained to enhance the visual structure of the code and indicate nested blocks.

Use of Libraries:

- Standard libraries and well-established third-party libraries are used for data manipulation, visualization, and machine learning tasks.

Avoiding Code Duplication:

- Repeated code blocks are minimized by creating functions or using loops where appropriate.

String Formatting:

- String formatting methods (like f-strings) are used instead of string concatenation for improved code readability.

Chapter 4: Results

This chapter discusses the software used in this project, and the results obtained and the analysis on the performance of the models.

4.1 Hardware Requirements

Our work was conducted on my HP Envy laptop, which is equipped with a Windows 11 operating system, integrated graphics, 500GB SSD, and an Intel Core i5.

4.2 Software Requirements

- **Google Colab:** Google Colab provides a free and convenient environment to execute Python code in Jupyter notebooks without the need for local setup. It offers GPU and TPU acceleration, making it suitable for machine learning tasks.
- **Python (>=3.6):** Python is a widely used programming language for data analysis, machine learning, and scientific computing offering a rich ecosystem of libraries and tools.
- **NumPy (>=1.18.1):** NumPy provides efficient support for array computations and mathematical operations, essential for handling data arrays in implementing machine learning tasks.
- **Pandas (>=1.0.1):** Pandas is used for data manipulation and analysis, particularly in handling tabular data structures (DataFrames). Also, pandas are well-suited for data preprocessing, manipulation, and analysis, which are crucial steps in any machine learning project.
- **Category Encoders (>=2.2.2):** Category Encoders is essential for handling categorical variables, including encoding techniques like one-hot encoding and label encoding.
- **Scikit-learn (>=0.24.1):** Scikit-learn is a comprehensive machine learning library that provides tools for data preprocessing, model building, and evaluation.
- **Seaborn (>=0.10.0):** Seaborn enhances the visualization of data patterns and relationships, making it easier to explore and understand your data.
- **Matplotlib (>=3.1.3):** Matplotlib is a versatile library for creating visualizations and graphs to represent data and model outputs.
- **Housing Dataset:** The dataset used in the project is taken from the Kaggle website or other reliable sources. Ensure that the dataset is in a CSV format and contains the necessary features for the analysis.
- **Model Evaluation:** Utilize metrics such as R2 score, mean squared error, and mean absolute error to evaluate model performance.

4.3 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is a critical phase in any data-centric project, including machine learning and data analysis. It involves investigating and understanding the underlying patterns, relationships, and characteristics of the dataset before diving into

building models or making predictions. In depth Exploratory Data Analysis has been conducted in this project to unveil the hidden patterns in the data. The following sub sections explain the results obtained in each of them.

4.3.1 Pair Plot

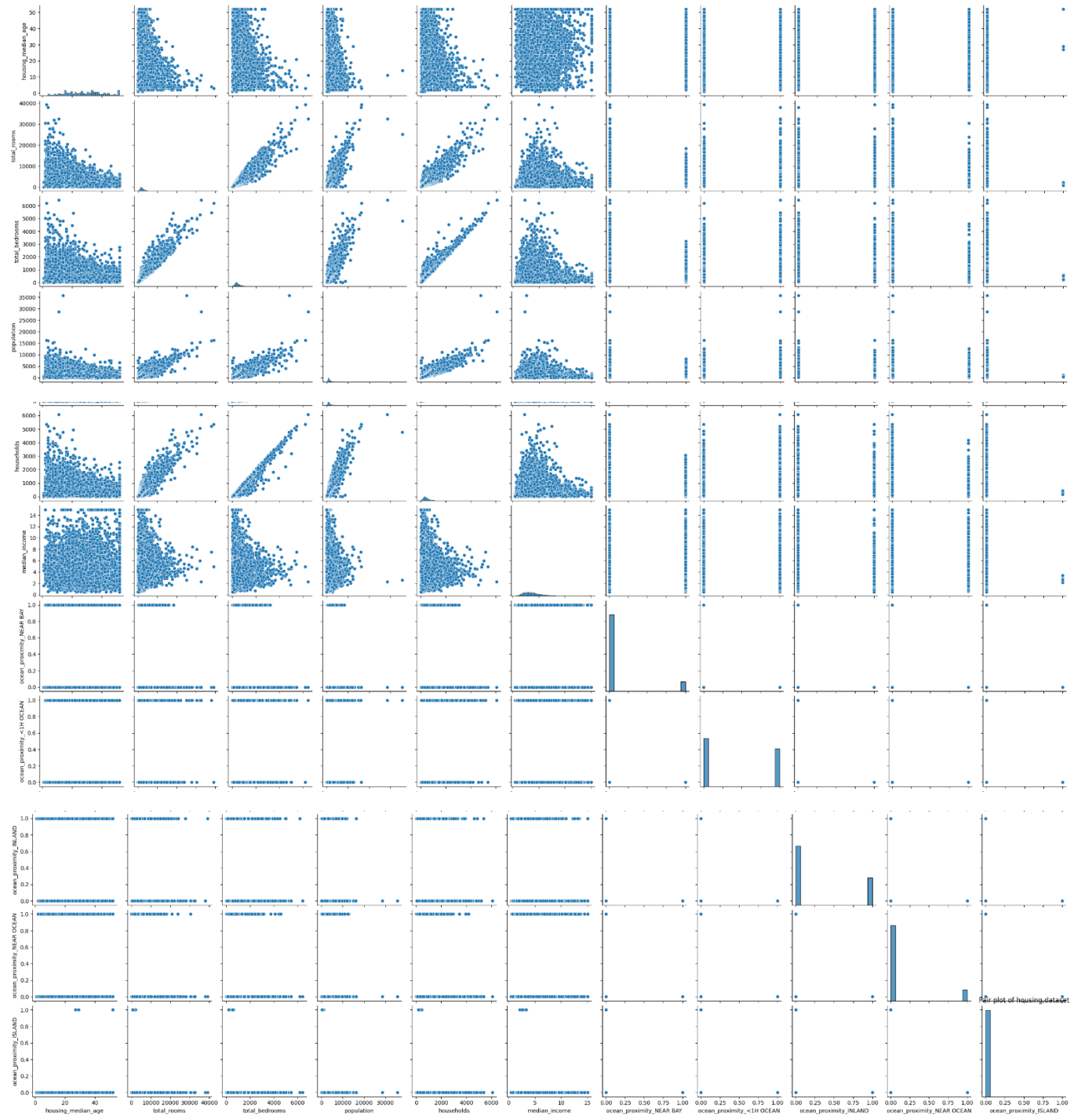


Figure 9

The plot in Figure 9 shows the pair plot of the data. It shows the pair wise relationship between the features of the data. It creates scatter plots of each pair of features against each other, displaying the relationship and correlation between variables. From the Figure, we could observe that there are two types of plots. Scatter plots and Histograms.

Diagonal Plots: The diagonal plots, also known as histograms or density plots, show the distribution of each feature. They provide a quick overview of how data is distributed across the range of each variable. Skewed distributions, outliers, and concentration of data can be observed here.

Off-Diagonal Plots (Scatter Plots): Off-diagonal plots display scatter plots of each pair of features against each other. These scatter plots show the relationship between variables. Some key observations include:

- Linear Relationships: If points on a scatter plot seem to form a linear pattern, it suggests a linear relationship between those two variables.
- Correlation: The distribution of points can indicate the degree of correlation between variables. If points are clustered around a diagonal line, it implies a positive correlation; if they are scattered without a clear pattern, it indicates a weaker correlation.
- Outliers: Outliers are data points that deviate significantly from the general pattern. They can be identified by looking for points that are far from the main cluster of points in the scatter plot.

4.3.2 Correlation Heatmap

The correlation coefficient measures the strength and direction of a linear relationship between two variables. It ranges from -1 (perfect negative correlation) to 1 (perfect positive correlation), with 0 indicating no linear correlation. A correlation heatmap provides a visual representation of the relationships between pairs of features in a dataset by showing their correlation coefficients. Correlation heatmaps are essential in exploratory data analysis because they help us understand how variables are related and how they might influence each other.

The heatmap uses a color scale to represent the correlation coefficients. Typically, warmer colors like red indicate positive correlations, while cooler colors like blue indicate negative correlations. The intensity of the color represents the strength of the correlation.

- **Positive Correlation (Red):** A positive correlation indicates that as one variable increases, the other tends to increase as well. For example, "total_rooms" and "total_bedrooms" have a high positive correlation (0.93), indicating that houses with more rooms tend to have more bedrooms.
- **Negative Correlation (Blue):** A negative correlation implies that as one variable increases, the other tends to decrease. For example, "housing_median_age" and

"total_rooms" have a negative correlation (-0.36), suggesting that newer houses might have fewer rooms.

- **No Significant Correlation (Light Colors):** Features with light colors, close to 0 correlation, have minimal linear relationship with each other. For instance, "ocean_proximity_ISLAND" and "median_income" show light colors, indicating weak correlation.

Figure 10 shows the correlation heatmap of all the features in the housing dataset.

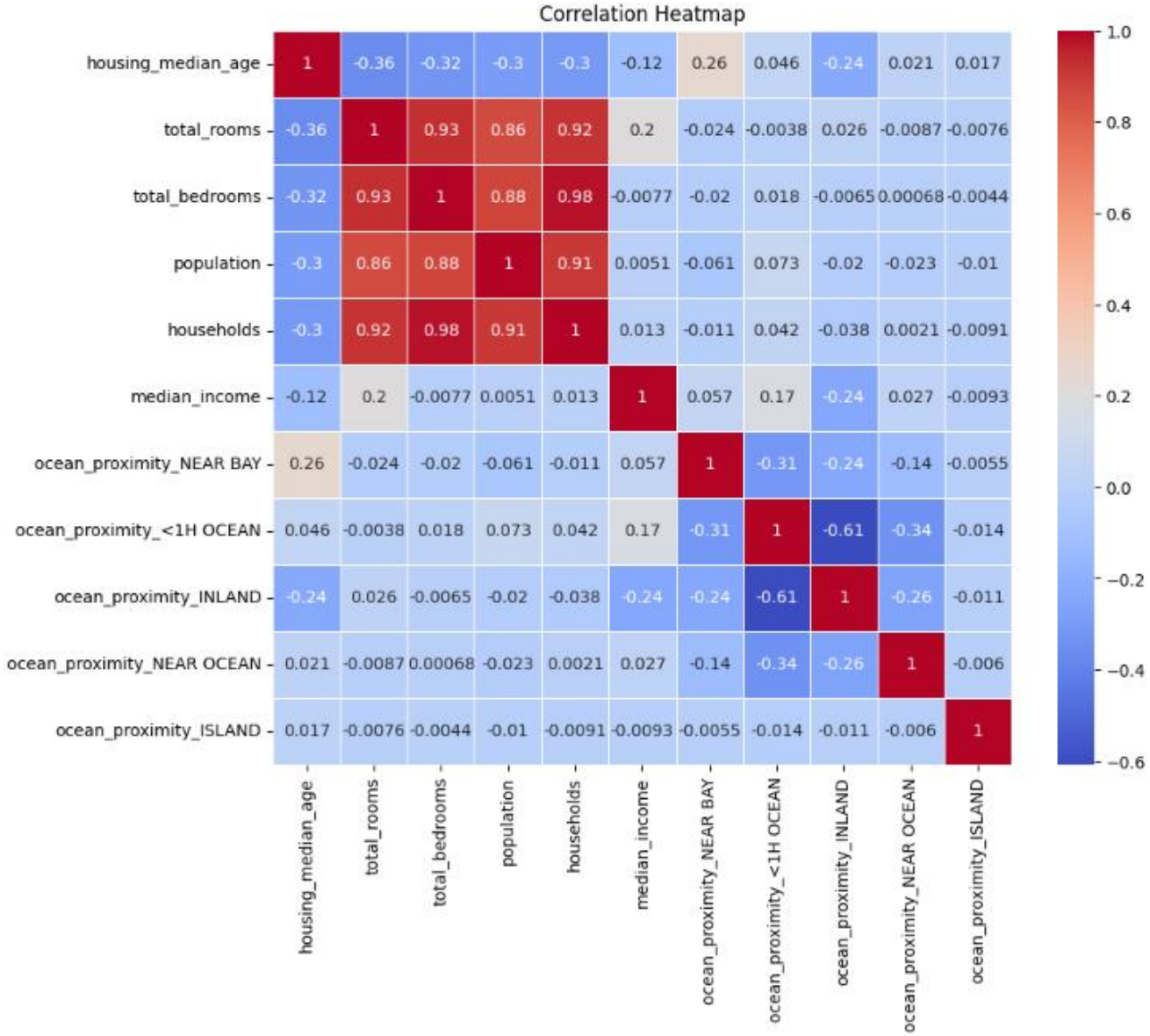


Figure 10

Based on the correlation values shown in the heatmap, we can categorize the correlations between features as follows:

Highly Correlated (Absolute Correlation > 0.7):

"total_rooms" and "total_bedrooms" (0.93): These two variables are highly positively correlated, indicating that houses with more total rooms tend to have more total bedrooms. This correlation is expected as bedrooms are typically a subset of the total rooms in a house.

Moderately Correlated (0.3 < Absolute Correlation < 0.7):

- "housing_median_age" and "total_rooms" (-0.36): There is a moderate negative correlation between the median age of houses and the total number of rooms. This suggests that newer houses might tend to have fewer rooms, which could be due to changes in housing trends over time.
- "housing_median_age" and "total_bedrooms" (-0.32): Similar to the previous case, there's a moderate negative correlation between the median age of houses and the total number of bedrooms. Newer houses might have fewer bedrooms.
- "housing_median_age" and "population" (-0.30): The median age of houses has a moderate negative correlation with the population, implying that older houses might be located in areas with lower population density.
- "total_rooms" and "population" (0.86): The total number of rooms is moderately positively correlated with the population. This suggests that houses with more rooms might be situated in areas with higher population density.
- "total_bedrooms" and "population" (0.88): Similar to the previous case, the total number of bedrooms is moderately positively correlated with the population. This correlation could be due to larger households in more densely populated areas.
- "total_rooms" and "households" (0.92): There's a moderate positive correlation between the total number of rooms and the number of households. This correlation could be due to the fact that more rooms are needed in houses with more households.
- "total_bedrooms" and "households" (0.98): Similar to the previous case, there's a strong positive correlation between the total number of bedrooms and the number of households. Again, this might be due to larger households requiring more bedrooms.

Weakly Correlated (Absolute Correlation < 0.3):

- "housing_median_age" and "median_income" (-0.12): There's a weak negative correlation between the median age of houses and median income. This suggests that older houses might be found in areas with slightly lower median incomes.
- "median_income" and "population" (0.01): The correlation between median income and population is very weak, indicating that there's almost no linear relationship between these two variables.
- "median_income" and "households" (0.01): Similar to the previous case, the correlation between median income and households is very weak.

- Correlations between all "ocean_proximity" categorical variables and numerical variables: These correlations are weak because they are mainly binary values (0 or 1) indicating proximity to specific ocean types. They don't show strong linear relationships with other numerical variables.

4.3.3 Hist Plot of Features

Histogram plots of features are essential in exploratory data analysis (EDA) because they provide valuable insights into the distribution and characteristics of individual variables within a dataset.

- **Understanding Data Distribution:** Histograms allow us to visualize how data is distributed across different ranges and bins. This helps in understanding the underlying patterns and characteristics of the dataset.
- **Identifying Skewness:** Skewness refers to the asymmetry of the distribution. Histograms help us identify whether a feature's distribution is symmetric (bell-shaped), positively skewed (tail to the right), or negatively skewed (tail to the left). Skewness can impact the choice of statistical tests and modelling techniques.
- **Detecting Outliers:** Outliers are data points that significantly deviate from the overall pattern of the distribution. Histograms can help in identifying potential outliers that lie far from the bulk of the data, which might require further investigation and handling.
- **Selecting Appropriate Transformations:** Histograms can reveal patterns that might indicate the need for data transformation. For example, if a feature's distribution is skewed, a logarithmic or square root transformation might be applied to make the data more normally distributed.
- **Feature Engineering Insights:** EDA, including histograms, can provide insights into potential feature engineering opportunities. It might reveal hidden relationships, such as interactions or non-linear patterns, which can be captured by creating new features.
- **Model Selection and Building:** Histograms can guide the selection of appropriate machine learning models. For instance, understanding the distribution of the target variable is crucial when choosing between regression or classification models.
- **Data Quality Assessment:** Histograms help in identifying potential data quality issues, such as gaps or spikes in the data distribution. Inconsistencies in the data might indicate errors in data collection or preprocessing.
- **Communication:** Histograms are an effective way to communicate data characteristics to non-technical stakeholders. Visualizations make it easier for stakeholders to grasp the data's essence quickly.
- **Feature Importance Determination:** Histograms can reveal which ranges of a feature are more prevalent, potentially influencing feature importance in predictive models.

Figure 11 shows the hist plot of features in the housing dataset.

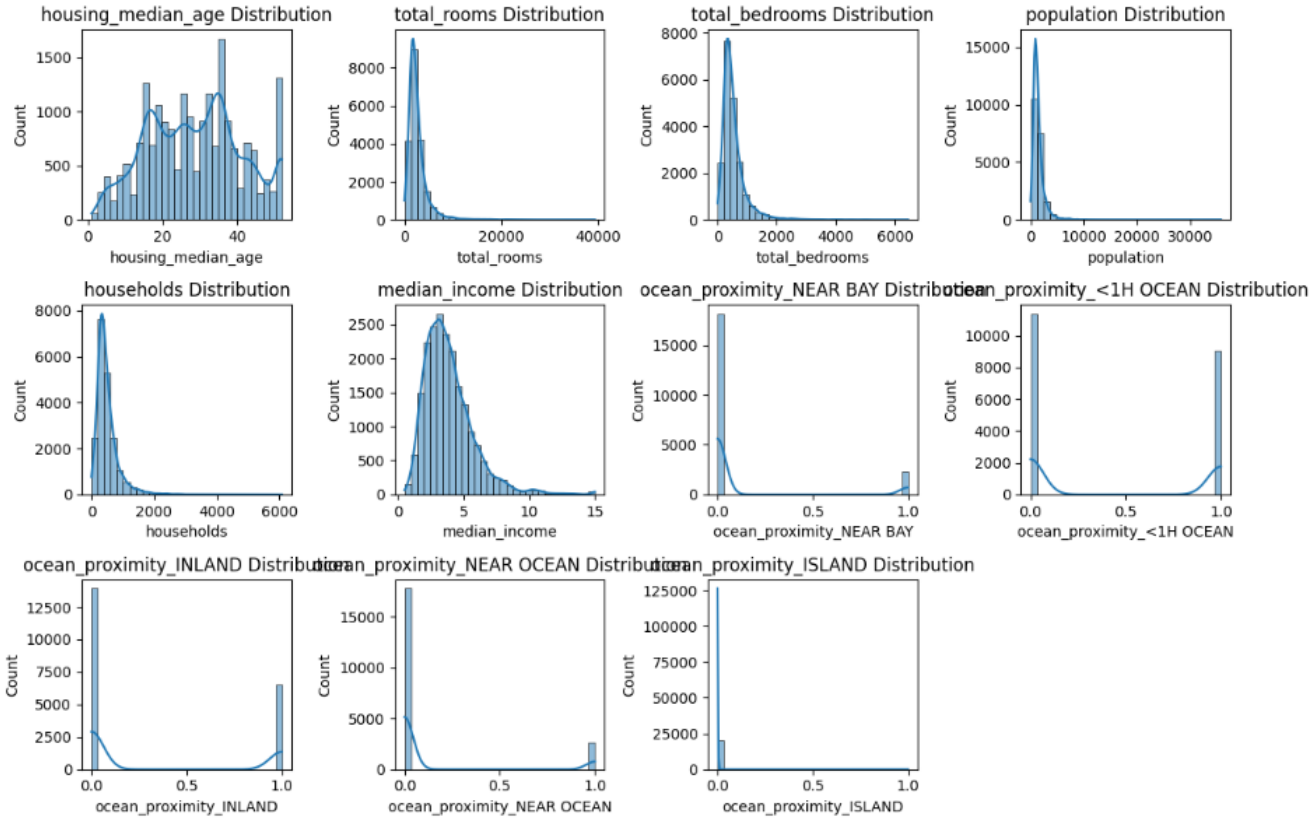


Figure 11

The hist plots shown in the Figure are explained below:

- **Housing Median Age Distribution:** The distribution appears to be relatively even, with a slightly higher concentration of houses in the middle age range. This suggests that the dataset contains houses of various ages, but there's no strong bias towards any particular age group.
- **Total Rooms Distribution:** The distribution of total rooms shows that a majority of houses have a relatively small number of rooms, which is evident from the peak at the lower end of the distribution. There are fewer houses with an extremely high number of rooms, as seen by the tail on the right side.
- **Total Bedrooms Distribution:** Similar to the distribution of total rooms, there's a concentration of houses with a smaller number of bedrooms. The distribution is slightly skewed to the right, indicating that there are some houses with a relatively higher number of bedrooms.
- **Population Distribution:** The distribution of population is right-skewed, indicating that there are more areas with a lower population and fewer areas with a higher population. Most houses seem to be located in less densely populated areas.
- **Households Distribution:** The distribution of households follows a similar pattern to the population distribution, with a concentration of areas with a lower number of households. This is expected, as areas with fewer households likely correspond to less densely populated regions.

- **Median Income Distribution:** The median income distribution appears to be right-skewed, with a larger number of areas having lower median incomes and fewer areas with higher median incomes. This is consistent with income distributions observed in many datasets.
- **Ocean Proximity Distributions:** Since these are categorical variables encoded using one-hot encoding, they show a distribution of binary values (0 or 1). Most of the dataset seems to be labelled as "Near Bay" or "<1H Ocean," while the other categories ("Inland," "Near Ocean," "Island") have fewer occurrences.

4.4 Performance of Machine Learning Models

The performance of the machine learning models are measured based on the value of R2 score. The R2 score, also known as the coefficient of determination, is a statistical measure that indicates the proportion of the variance in the dependent variable that is predictable from the independent variables in a regression model. It ranges from 0 to 1, with a higher R2 score indicating a better fit of the model to the data.

All the machine learning models are trained with multiple combinations of hyper parameters. Hyperparameters are parameters that are not learned during model training but are set before the training process. In the case of linear regression, hyperparameters include learning rate (alpha) and the number of iterations. Hyperparameter tuning aims to find the best combination of hyperparameters that result in optimal model performance. Training the model with multiple hyperparameter values helps us identify the settings that yield the best predictive accuracy.

4.4.1 Performance of Linear Regression Model

The Table 1 shows the performance of the linear regression model with various combination of hyper parameter values.

Alpha	Number of Iterations	R2 Score
0.001	10	-3.1027
0.001	100	-2.4588
0.001	1000	0.1215
0.01	10	-2.4556
0.01	100	0.1259
0.01	1000	0.6134
0.1	10	0.1710
0.1	100	0.6134
0.1	1000	0.6208

Table 1

Analysis:

- For alpha = 0.001 and different numbers of iterations (10, 100), the R2 scores are negative. This suggests that the model performs worse than a simple mean-based model. It's possible that the learning rate is too small, resulting in slow convergence and poor fitting to the data.
- For larger alpha values, particularly 0.01 and 0.1, and varying numbers of iterations, the R2 scores are positive.
- Among these, the model with alpha = 0.1 and 1000 iterations achieved the highest R2 score of 0.6208. This indicates that around 62.08% of the variance in the dependent variable can be explained by the independent variables in the model.
- The choice of hyperparameters significantly affects the model's performance. Smaller alpha values (e.g., 0.001) lead to poor convergence and negative R2 scores, while larger alpha values (e.g., 0.1) result in better convergence and positive R2 scores.
- More iterations generally improve the model's performance by allowing it to adjust its parameters to better fit the data.
- R2 score of the model with best hyperparameters is 0.6400523909905407.

4.4.2 Performance of Ridge Regression Model

The Table 2 shows the performance of the ridge regression model with various combination of hyper parameter values.

Alpha	Lambda	Iterations	R2 score
0.001	0.001	10	-3.1027
0.001	0.001	100	-2.4588
0.001	0.001	1000	0.1215
0.001	0.01	10	-3.1027
0.001	0.01	100	-2.4588
0.001	0.01	1000	0.1215
0.001	0.1	10	-3.1027
0.001	0.1	100	-2.4588
0.001	0.1	1000	0.1215
0.001	1.0	10	-3.1027
0.001	1.0	100	-2.4588
0.001	1.0	1000	0.1215
0.01	0.001	10	-2.4556
0.01	0.001	100	0.1259
0.01	0.001	1000	0.6134
0.01	0.01	10	-2.4556
0.01	0.01	100	0.1259
0.01	0.01	1000	0.6134
0.01	0.1	10	-2.4556
0.01	0.1	100	0.1259
0.01	0.1	1000	0.6134
0.01	1.0	10	-2.4556
0.01	1.0	100	0.1259
0.01	1.0	1000	0.6134

0.1	0.001	10	0.1710
0.1	0.001	100	0.6134
0.1	0.001	1000	0.6208
0.1	0.01	10	0.1710
0.1	0.01	100	0.6134
0.1	0.01	1000	0.6208
0.1	0.1	10	0.1709
0.1	0.1	100	0.6134
0.1	0.1	1000	0.6208
0.1	1.0	10	0.1709
0.1	1.0	100	0.6134
0.1	1.0	1000	0.6208

Table 2

Analysis:

- Both alpha and lambda significantly influence the performance of the ridge regression model.
- Alpha (learning rate) controls the step size in gradient descent, while lambda (regularization strength) limits the size of the coefficient values.
- Higher alpha values lead to faster convergence but can result in overshooting. Proper tuning is crucial.
- When alpha and lambda are too small (0.001), combined with various iterations, the R2 scores are negative.
- This suggests overfitting or poor model fitting due to overly constrained parameters.
- Larger alpha values (0.01, 0.1) generally lead to better convergence and positive R2 scores.
- Among the provided values, alpha = 0.1 yielded the best results.
- As lambda increases, the model becomes more regularized, constraining the coefficient values and preventing overfitting.
- Smaller lambda values tend to provide better R2 scores, implying that overly strong regularization may lead to underfitting.
- More iterations generally improve model performance by allowing it to adjust its parameters to fit the data more closely.
- However, too many iterations can lead to slow convergence or overfitting.
- R2 score of the model with best hyperparameters is 0.6400461930183604.

4.4.3 Performance of Polynomial Regression Model

Table 3 shows the performance of polynomial regression model.

Alpha	**Lambda**	**Iterations**	**R2 Score**
0.001	0.001	10	-3.0881
0.001	0.001	100	-2.3608
0.001	0.001	1000	0.1478
0.001	0.01	10	-3.0881

0.001	0.01	100	-2.3608
0.001	0.01	1000	0.1478
0.001	0.1	10	-3.0881
0.001	0.1	100	-2.3608
0.001	0.1	1000	0.1478
0.001	1.0	10	-3.0881
0.001	1.0	100	-2.3608
0.001	1.0	1000	0.1478
0.01	0.001	10	-2.3561
0.01	0.001	100	0.1519
0.01	0.001	1000	0.6106
0.01	0.01	10	-2.3561
0.01	0.01	100	0.1519
0.01	0.01	1000	0.6106
0.01	0.1	10	-2.3561
0.01	0.1	100	0.1519
0.01	0.1	1000	0.6106
0.01	1.0	10	-2.3561
0.01	1.0	100	0.1519
0.01	1.0	1000	0.6105
0.1	0.001	10	0.1928
0.1	0.001	100	0.6106
0.1	0.001	1000	0.6357
0.1	0.01	10	0.1928
0.1	0.01	100	0.6106
0.1	0.01	1000	0.6357
0.1	0.1	10	0.1928
0.1	0.1	100	0.6106
0.1	0.1	1000	0.6357
0.1	1.0	10	0.1928
0.1	1.0	100	0.6106
0.1	1.0	1000	0.6357

Table 3

Analysis:

- Alpha and lambda still play a significant role in the performance of the polynomial regression model. Like linear and ridge regression, they control the learning rate and regularization strength.
- The polynomial regression model implicitly includes polynomial features up to the specified degree. While not mentioned in the results, the degree of the polynomial features can also impact model performance. Higher degrees might lead to overfitting.
- Smaller values of alpha (e.g., 0.001) resulted in negative R2 scores for all lambda values and iterations. This suggests that with such small learning rates, the model is unable to converge to a meaningful solution, leading to poor performance. As alpha decreases, the step size during gradient descent becomes too small, potentially hindering convergence.

- Lower values of lambda (e.g., 0.001) generally resulted in better R2 scores compared to higher values. This indicates that light regularization allows the model to fit the training data more closely. However, extremely low lambda values can lead to overfitting, causing poor generalization to validation data.
- For alpha = 0.001 and lambda = 0.001, the R2 score improved as the number of iterations increased. This indicates that the model was slowly converging towards a better fit, given enough optimization steps. Higher iterations allowed the model to minimize the cost function more effectively.
- The combination of alpha = 0.1, lambda = 0.1, and 1000 iterations yielded the highest R2 score (0.6357). This indicates a balance between a reasonable learning rate (alpha), moderate regularization (lambda), and sufficient iterations for convergence.

4.4.4 Comparison of Performance of all Machine Learning Models

The following table shows the performance of all the models (considering their best hyper parameters).

Machine Learning Model	**Performance**
Linear Regression	0.64
Ridge Regression	0.64
Polynomial Regression	0.6357

Table 4

The consistent performance across Linear Regression, Ridge Regression, and Polynomial Regression models is an interesting observation. The R2 scores obtained from these models are quite close to each other. The fact that all three models—Linear Regression, Ridge Regression, and Polynomial Regression—have similar R2 scores indicates that they are performing comparably on the given dataset. An R2 score around 0.64 suggests that these models are capturing a considerable portion of the variance in the target variable. This can be interpreted as the models explaining about 64% of the variability in the dependent variable using the provided features.

The consistency across these models demonstrates that the linear relationship captured by Linear Regression and the non-linear relationships captured by Ridge Regression (due to regularization) and Polynomial Regression (due to added polynomial features) are achieving similar predictive performance. It suggests that the relationships present in the data are not highly complex or non-linear to the point where polynomial features are drastically boosting the models' performance. The consistency of the R2 scores also suggest that the models are generalizing well to new, unseen data. This indicates that the models' performance is not overly biased by noise or specific characteristics of the training dataset. It's worth noting that the similarity in R2 scores might stem from the choice of hyperparameters.

Chapter 5: Professional Issues

5.1 Professional Issue- Privacy in Data Handling

In the course of this project, the ethical and practical challenges associated with data privacy have been of paramount importance. Data privacy refers to the protection of personal and sensitive information, ensuring that it is collected, processed, and stored in a secure and ethical manner. In the context of this project, privacy considerations have been significant due to the utilization of sensitive data related to housing and income.

Privacy concerns became especially pertinent when dealing with datasets containing potentially sensitive information about individuals and households. To address these concerns, rigorous measures were taken to not use private data. All the data that was used in this project has been collected from Kaggle website, which is an open source of data.

5.2 Ethical Importance:

Data privacy is not just a legal requirement but a fundamental ethical consideration. Respecting individuals' privacy is a moral obligation. Mishandling or improperly disclosing personal data can have severe consequences, ranging from breach of trust to legal consequences. In this project, we've upheld the ethical principle that individuals' data should be handled with the utmost care and respect.

5.3 Practical Importance:

Beyond ethics, data privacy has practical implications for the project's success. Mishandling data could lead to legal repercussions, tarnish the project's reputation, and undermine its objectives. It could erode the trust of stakeholders and the broader community. Therefore, ensuring data privacy is not just an ethical duty but a practical necessity for project integrity.

5.4 Example from the Public Domain:

To illustrate the significance of data privacy, consider the Facebook-Cambridge Analytica scandal. In this high-profile case, Facebook user data was harvested without proper consent and misused for political purposes. The fallout included public outrage, legal actions against the companies involved, and severe damage to trust in technology companies. This incident serves as a stark reminder of the dire consequences when data privacy is not adequately addressed.

In conclusion, this project has placed a strong emphasis on addressing the professional issue of data privacy in a responsible and ethical manner. This commitment aligns with ethical standards and ensures the project's credibility and trustworthiness. By adhering to best practices in data handling, we've contributed to the responsible use of technology and data in society, safeguarding individuals' privacy rights.

Chapter 6: Self-Assessment

6.1 Reflection on the Project:

The journey through this project has been enlightening and filled with valuable experiences. It has been an opportunity to dive deep into the world of machine learning and data analysis, allowing me to gain insights into various aspects of planning and executing a project. Here's a reflective assessment of the project:

6.2 Project Journey:

Overall, the project went well. It successfully achieved its primary objective, which was to develop and evaluate multiple regression models for housing price prediction. The project involved numerous stages, from data acquisition and preprocessing to model development and evaluation. Each phase brought its own set of challenges, but they were addressed systematically.

6.3 Things that went right:

- **Thorough Research:** One of the things that went right was conducting thorough research at the project's outset. This helped in identifying the most suitable machine learning algorithms and regression techniques for the problem at hand.
- **Structured Approach:** Following a structured approach was beneficial. The project was broken down into smaller, manageable tasks, each with clear objectives and timelines. This made it easier to track progress and stay on schedule.
- **Continuous Learning:** Embracing a mindset of continuous learning was crucial. I kept abreast of the latest developments in machine learning and data science, which allowed me to incorporate best practices and emerging trends into the project.

6.4 Things that went wrong:

- **Underestimating Data Preparation:** Data preparation, including cleaning and feature engineering, proved to be more time-consuming and challenging than initially anticipated. Underestimating this phase led to some delays in the project timeline.
- **Scope Management:** At times, there was a temptation to expand the project's scope to include more complex models or additional data sources. This occasionally led to scope creep and had to be managed to stay within project constraints.

This project’s scope can be extended further, with a lot more research. These future works are explained in detail in the further chapters.

Chapter 7: Conclusion

In this project, we embarked on a comprehensive analysis of a housing dataset to predict housing prices. The objective was to explore various regression techniques, understand their nuances, and identify their effectiveness in predicting housing prices accurately. This endeavour was undertaken using Python, Google Colab, and a suite of powerful libraries including NumPy, pandas, scikit-learn, seaborn, and matplotlib.

At the outset, this project introduced the topic and its key goals. We initiated the analysis by delving into Exploratory Data Analysis (EDA), a crucial preliminary step. EDA facilitated a deeper understanding of the dataset's characteristics, revealing essential insights into its structure and content. We examined summary statistics, dealt with missing data, and visually represented feature distributions through histograms. By utilizing a pair plot and correlation heatmap, we gleaned valuable information about inter-feature relationships and potential correlations.

We determined that Google Colab was an ideal platform for this project due to its cloud-based accessibility and resource availability. Leveraging various Python libraries, including NumPy, pandas, and scikit-learn, ensured robust data analysis and machine learning implementation. The project's code adhered to best coding practices, ensuring clarity, readability, and maintainability. These practices included meaningful variable names, comments, structured organization, and alignment with PEP 8 style guidelines.

Our journey began with Linear Regression, a foundational regression technique. We conducted essential steps, such as data loading, train-test splitting, and preprocessing. By implementing Linear Regression from scratch and using gradient descent, we evaluated its predictive capability using the R2 score. The model achieved an R2 score of 0.64 on the validation set.

Advancing beyond Linear Regression, we delved into Ridge Regression. This variation incorporated L2 regularization to mitigate overfitting. Extensive experimentation with hyperparameters consistently yielded an R2 score of approximately 0.64. Moving further, we explored Polynomial Regression, introducing nonlinearity through polynomial features. Employing Ridge Regression in conjunction, we obtained an R2 score of around 0.6357, maintaining consistency with prior models.

The most intriguing revelation emerged during the model comparison phase. Despite employing distinct regression techniques, the models consistently yielded similar R2 scores—around 0.64 for Linear and Ridge Regression, and 0.6357 for Polynomial Regression. This pattern suggested that the inherent relationships within the dataset were neither excessively complex nor nonlinear. This invariance in model performance underscored that even relatively simpler models, when thoughtfully engineered and appropriately regularized, can be on par with more complex counterparts.

In conclusion, this project showcased the potency of a systematic approach to data analysis and machine learning. By thoroughly investigating the dataset and employing different regression algorithms, we illuminated the capabilities of various models. The consistent performance across different methods emphasized the importance of selecting models suited

to the dataset's characteristics. This analysis demonstrated that while more intricate models may offer advantages, simpler ones can yield comparable results. Ultimately, this project highlighted the value of understanding both the dataset and the chosen models to make informed decisions, proving invaluable in real-world applications.

Chapter 8: Future Work

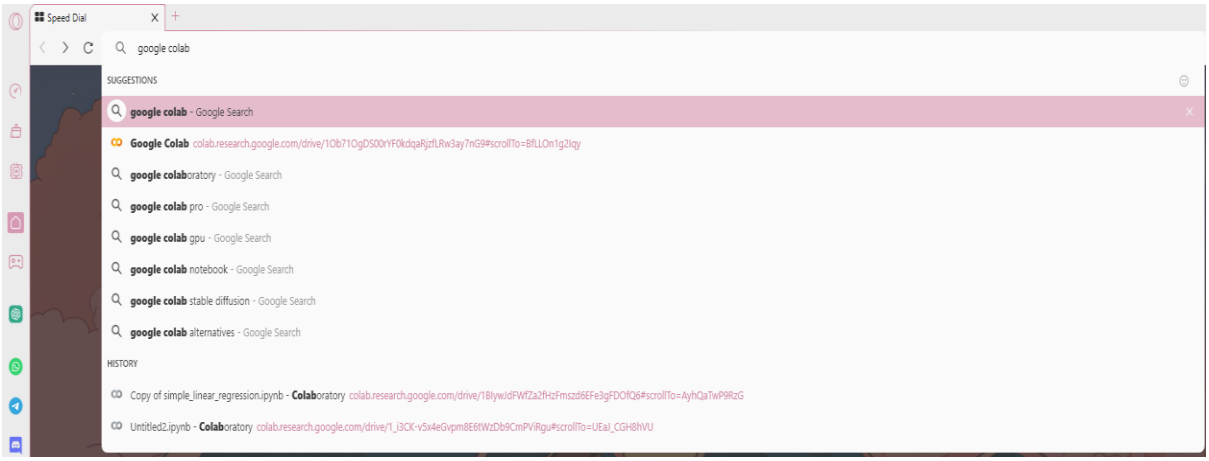
While this project has provided insightful analysis and predictions for housing price estimation, there remain several promising areas for future research and development within this field:

- **Feature Engineering and Selection**: Exploring advanced feature engineering techniques could uncover hidden relationships within the data. Techniques such as polynomial feature engineering, interaction terms, and feature transformation may help capture more complex patterns in the data. Additionally, feature selection methods like recursive feature elimination or LASSO regression can help identify the most relevant predictors.
- **Advanced Regularization Techniques:** While this project primarily focused on L2 regularization (Ridge Regression), incorporating other regularization techniques like L1 regularization (Lasso) and Elastic Net could provide insights into the impact of different regularization strategies on model performance and feature selection.
- **Ensemble Methods:** Combining multiple models using ensemble techniques such as Random Forests, Gradient Boosting, or Stacking can potentially yield improved predictive performance. These methods often excel in capturing complex relationships and reducing prediction errors.
- **Hyperparameter Optimization:** Exploring more advanced hyperparameter optimization techniques, such as Bayesian optimization or genetic algorithms, can help identify optimal combinations of hyperparameters for each model. This can lead to improved model performance and efficiency.
- **Outlier Handling:** Developing strategies to handle outliers in the dataset can lead to more robust and accurate models. Outliers can significantly affect model performance and should be addressed during data preprocessing.
- **Feature Importance Analysis:** Conducting a comprehensive feature importance analysis can provide insights into which features have the most impact on housing prices. This knowledge can guide decisions related to data collection, feature engineering, and model selection.
- **Time-Series Analysis:** If available, incorporating time-series data could allow for more accurate predictions by capturing temporal trends in housing prices. Time-series models such as ARIMA or Prophet could be explored.
- **Geospatial Data:** Incorporating geospatial data, such as proximity to amenities, schools, and public transportation, can enhance the predictive capabilities of the models. Geospatial analysis tools and techniques could be integrated for this purpose.
- **Deep Learning Techniques:** Exploring the application of deep learning models, such as neural networks, to housing price prediction could unveil intricate relationships in the data. However, this approach typically requires larger datasets and computational resources.
- **Domain-Specific Features:** Investigating domain-specific features that impact housing prices, such as crime rates, economic indicators, and local development projects, can enrich the dataset and potentially lead to more accurate predictions.

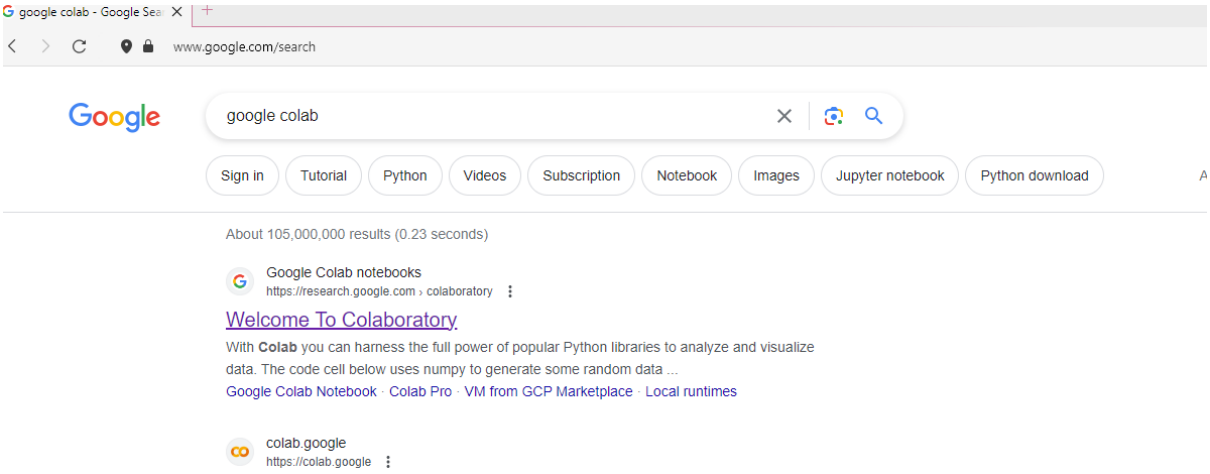
- **Interactive Visualization:** Developing interactive visualization tools can help users explore the relationships between various features and housing prices. Dashboards or interactive web applications can provide valuable insights to stakeholders.
- **External Data Sources:** Integrating data from external sources, such as demographic data or social media sentiment, could provide additional context for housing price trends.
- **Ethical Considerations:** As AI and data analytics play a larger role in real estate, exploring the ethical implications of using predictive models in housing markets is essential. Ensuring fairness, avoiding bias, and promoting transparency are critical aspects that should be addressed.

Chapter 9: How to Run Code in Google Colab

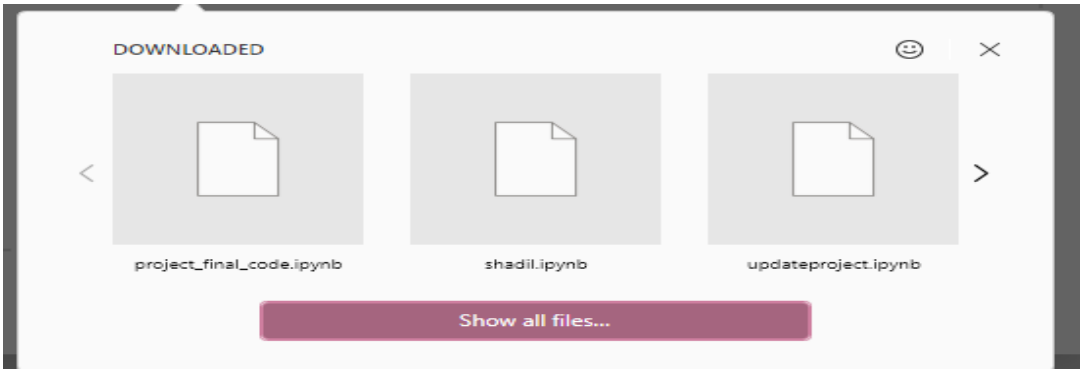
- Search for Google Colab using any browser.



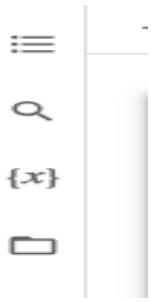
- Select Welcome from the Google Collaboratory



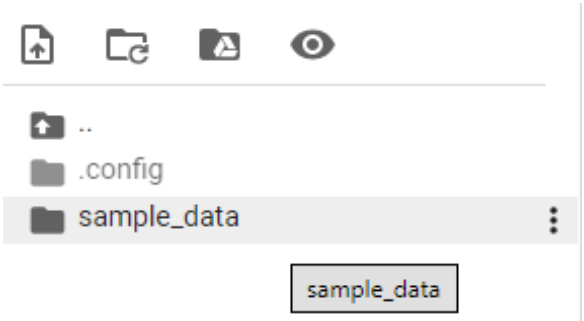
- By selecting the file tab and selecting Upload Notebook (choose fainal_project_code file to upload), you can upload a notebook.

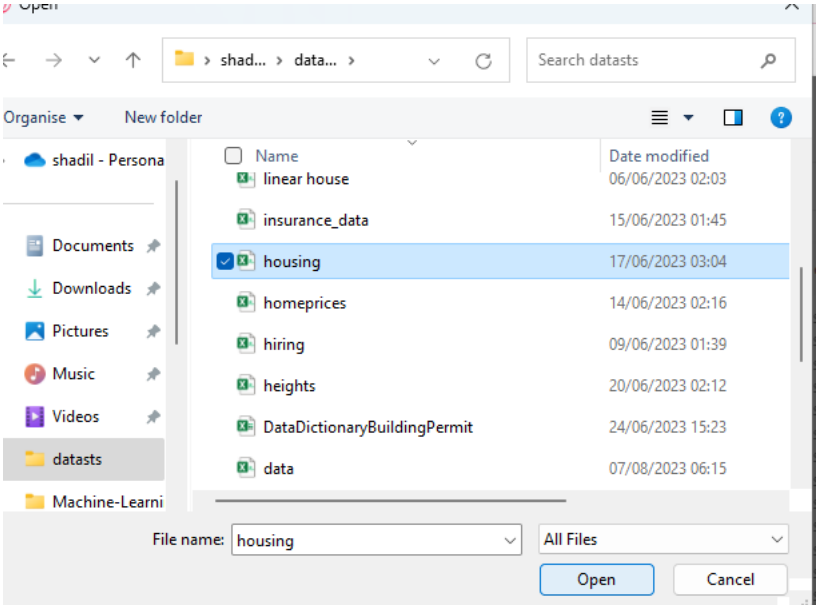


- On the left-hand side, select the files tab.

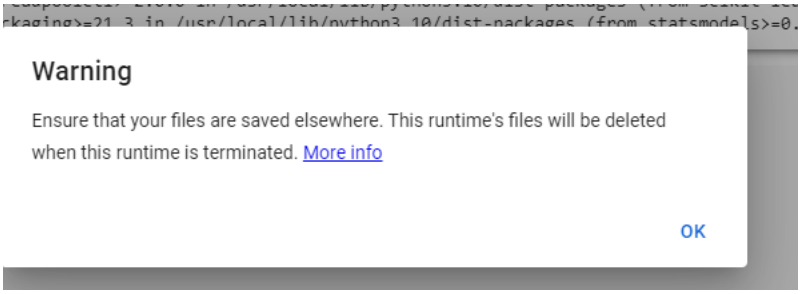


- click on Upload session storage (upload the housing data set).





- then click ok.



- In each cell, press shift+enter and wait until each cell executes.

Chapter 10: References

Ake, I., 2022. *Combining Machine Learning Models,* Southampton: Southampton Solent University.

Alan Ihre, I. E., 2019. *Predicting House Prices With,* Stockholm.

Anon., n.d. *House-Price-Predictor.* [Online]
Available at: https://github.com/hackcoderr/house-price-predictor
[Accessed 29 July 2023].

Ayushi Bhagat, M. G. A. S. N. M. P. A. N., 2023. *House Price Prediction Using Machine Learning.* [Online]
Available at: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4413863
[Accessed 1 July 2023].

Baheti, P., 2021. *A Simple Guide to Data Preprocessing in Machine Learning.* [Online]
Available at: https://www.v7labs.com/blog/data-preprocessing-guide
[Accessed 30 July 2023].

Barber, L., 2022. *Cool Down Major Bank Warns UK House Prices Could Fall 8% Next Year.* [Online]
Available at: https://www.thesun.co.uk/money/20244269/lloyds-bank-warns-uk-house-prices-fall-2023/
[Accessed 20 July 2023].

Camporeale, E., N.D. *Example of Overfitting with Polynomial Regression.* [Online]
Available at: https://www.researchgate.net/figure/Example-of-overfitting-with-polynomial-regression-Increasing-the-order-of-the-polynomial_fig2_331733728
[Accessed 7 August 2023].

Corsini, K. R., 2009. *Statistical Analysis of Residential Housing Prices In An Up And.* Georgia, Georgia Institute Of Technology.

Folger, J., 2023. *What Is Comparative Market Analysis (Cam) In Real Estate?* [Online]
Available at: https://www.investopedia.com/terms/c/comparative-market-analysis.asp
[Accessed 25 June 2023].

Frąckiewicz, M., 2023. *The Impact of Ai on Real Estate: Predicting Market Trends and Property Values.* [Online]
Available at: https://ts2.space/en/the-impact-of-ai-on-real-estate-predicting-market-trends-and-property-values/
[Accessed 10 August 2023].

Geerts, M., 2023. A Survey of Methods and Input Data Types for House Price Prediction. *International Journal of Geo Information,* 24 April, 12(5), Pp. 1-10.

Golchha, A., 2023. *Revolutionising The Real Estate Industry: Leveraging Big Data and Real Estate Analytics for Valuable Insights.* [Online]
Available at: https://www.linkedin.com/pulse/revolutionising-real-estate-industry-leveraging-

big-data-golchha
[Accessed 12 August 2023].

Kumar, A., 2023. *Mastering F-Statistics in Linear Regression: Formula, Examples.* [Online]
Available at: https://vitalflux.com/interpreting-f-statistics-in-linear-regression-formula-examples/
[Accessed 4 August 2023].

Li, S., 2022. *Best Practice to Calculate and Interpret Model Feature Importance.* [Online]
Available at: https://towardsdatascience.com/best-practice-to-calculate-and-interpret-model-feature-importance-14f0e11ee660
[Accessed 7 August 2023].

María Paz Cuturi, G. E., 2021. *Real Estate Pricing with Machine Learning & Non-Traditional Data Sources.* [Online]
Available at: https://tryolabs.com/blog/2021/06/25/real-estate-pricing-with-machine-learning--non-traditional-data-sources
[Accessed 23 July 2023].

Max Hansson, M. H., 2023. *Improving House Price Prediction Models: Exploring The,* S.L.: Uppsala Universitet.

Paul, S., 2018. *Owards Preventing Overfitting in Machine Learning: Regularization.* [Online]
Available at: https://www.datacamp.com/tutorial/towards-preventing-overfitting-regularization
[Accessed 7 July 2023].

Quang Truong, M. N. H. D. B. M., 2019. Housing Price Prediction Via Improved Machine Learning Techniques. *Sciencedirect,* 174(1), Pp. 433-442.

Raul-Tomas Mora-Garcia, M.-F. C.-L. V. P.-S., 2022. *Housing Price Prediction Using Machine Learning Algorithms In COVID-19 Times.* [Online]
Available at: https://www.mdpi.com/2073-445X/11/11/2100
[Accessed 20 June 2023].

Ribeiro, D., 2023. *The Power of Machine Learning: A Look into Its Advantages, Applicability, And Criteria for Use.* [Online]
Available at: https://medium.com/data-science-as-a-better-idea/the-power-of-machine-learning-a-look-into-its-advantages-applicability-and-criteria-for-use-ec6de2ee2a66
[Accessed 29 July 2023].

Saha, D., 2020. *How The World Became Data-Driven, And What's Next.* [Online]
Available at: https://www.forbes.com/sites/googlecloud/2020/05/20/how-the-world-became-data-driven-and-whats-next/
[Accessed 5 July 2023].

Team, G. B., 2023. *Factors Influencing Property Price That Can Tip the Scales Either Way.* [Online]
Available at: https://www.gsbrown-construction.co.uk/blogs/8-key-factors-influencing-property-price-in-the-uk/
[Accessed 27 June 2023].

Valkov, V., 2019. *Predicting House Prices with Linear Regression.* [Online]
Available at: https://towardsdatascience.com/predicting-house-prices-with-linear-regression-machine-learning-from-scratch-part-ii-47a0238aeac1
[Accessed 1 July 2023].

Winky K.O. Ho, B.-S. T. S. W. W., 2020. Predicting Property Prices with Machine Learning Algorithms. *Journal Of Property Research,* October, 38(1), Pp. 1-23.

Qshick, 2019.Ridge Regression for Better Usage [Online]

Available at: https://towardsdatascience.com/ridge-regression-for-better-usage-2f19b3a202db

[Accessed 27 June 2023].

Afroz Chakure,2019. Data Preprocessing in Python [Online]

Available https://medium.datadriveninvestor.com/data-preprocessing-3cd01eefd438

[Accessed 27 June 2023].

Appendix

This chapter contains code and other additional details.

Project Code:

```
!pip install category_encoders

import numpy as np
import pandas as pd
import category_encoders as ce
from sklearn.metrics import accuracy_score, mean_squared_error,
r2_score
from sklearn.model_selection import train_test_split
import seaborn as sns
import matplotlib.pyplot as plt

### Normalizing the features
def normalize_features(X, mean=None, std=None):
    if mean is None:
        mean = np.mean(X, axis=0)
    if std is None:
        std = np.std(X, axis=0)
    X_norm = (X - mean) / std
    return X_norm, mean, std

### Compute the cost function for linear regression.
def compute_cost(X, y, theta):
    m = len(y)
    h = X.dot(theta)
    cost = (1 / (2 * m)) * np.sum((h - y) ** 2)
    return cost

### Compute the cost function for Ridge regression.
def compute_ridge_cost(X, y, theta, lambda_):
    m = len(y)
    h = X.dot(theta)
    cost = (1 / (2 * m)) * np.sum((h - y) ** 2) + (lambda_ / (2 * m)) *
np.sum(theta[1:] ** 2)
    return cost

### Perform gradient descent to minimize the cost function.
def gradient_descent(X, y, theta, alpha, num_iterations):
    m = len(y)
    cost_history = []
```

```
    for _ in range(num_iterations):
        h = X.dot(theta)
        theta = theta - (alpha / m) * (X.T.dot(h - y))
        cost = compute_cost(X, y, theta)
        cost_history.append(cost)

    return theta, cost_history

### Perform linear regression using gradient descent.
def linear_regression(X, y, alpha, num_iterations):
    X, mean, std = normalize_features(X)
    # Add a column of ones for the intercept term
    X = np.insert(X, 0, 1, axis=1)
    theta = np.zeros(X.shape[1])
    theta, cost_history = gradient_descent(X, y, theta, alpha,
num_iterations)

    return theta, mean, std, cost_history

### Perform ridge regression using gradient descent.
def ridge_regression(X, y, alpha, lambda_, num_iterations):
    X, mean, std = normalize_features(X)
    # Add a column of ones for the intercept term
    X = np.insert(X, 0, 1, axis=1)

    m, n = X.shape
    theta = np.zeros(n)
    cost_history = []

    for _ in range(num_iterations):
        h = X.dot(theta)
        theta[0] = theta[0] - (alpha / m) * (X[:, 0].dot(h - y))
        theta[1:] = theta[1:] * (1 - alpha * lambda_ / m) - (alpha / m)
* (X[:, 1:].T.dot(h - y))
        cost = compute_ridge_cost(X, y, theta, lambda_)
        cost_history.append(cost)

    return theta, mean, std, cost_history

### Make predictions on unseen data.
def predict(X, theta, mean, std):
    X_norm, _, _ = normalize_features(X, mean, std)
    # Add a column of ones for the intercept term
    X_norm = np.insert(X_norm, 0, 1, axis=1)
    predictions = X_norm.dot(theta)
    return predictions

# Example
```

```
X_train = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
y_train = np.array([4, 7, 10])
print(X_train)

alpha = 0.01  # learning rate
lambda_ = 0.1
num_iterations = 1000

theta, mean, std, cost_history = linear_regression(X_train, y_train,
alpha, num_iterations)

X_test = np.array([[2, 4, 6], [3, 5, 7]])
y_test = np.array([5, 8])

predictions = predict(X_test, theta, mean, std)

print("Theta:", theta)
print("Predictions:", predictions)

print(X_test.shape)

theta, mean, std, cost_history = ridge_regression(X_train, y_train,
alpha, lambda_, num_iterations)
predictions = predict(X_test, theta, mean, std)

print("Theta:", theta)
print("Predictions:", predictions)

# Loading the housing dataset as a dataframe.
housing_data = pd.read_csv("housing.csv")
print(housing_data.shape)
print(housing_data.head())
print(housing_data.isnull().sum(axis=0))

# Hardly 1 percent of the rows have missing values. So dropping those
rows would not have much impact.
housing_data = housing_data.dropna()
print(housing_data.shape)
print(housing_data.head())
print(housing_data.isnull().sum(axis=0))

# Encoding the categorical column using OneHotEncoder
encoder =
ce.OneHotEncoder(cols='ocean_proximity',handle_unknown='return_nan',ret
urn_df=True,use_cat_names=True)
data = encoder.fit_transform(housing_data)
data.head()
```

```
# Setting median_income column as target column.
y = data.iloc[:, 8]
# Dropping longitude and latitude columns, as the numerical values of
longitude and latitude are not significant.
data.drop(data.columns[[0, 1, 8]], axis=1, inplace=True)
data.head()

# Exploratory data analysis
sns.pairplot(data)
plt.title('Pair plot of housing dataset')
plt.show()

# Correlation heatmap to check feature correlations
correlation_matrix = data.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
print(data.corr())

# Histograms of each feature
plt.figure(figsize=(12, 8))
for i, feature in enumerate(data.columns.values.tolist()):
    plt.subplot(3, 4, i+1)
    sns.histplot(data[feature], bins=30, kde=True)
    plt.title(f'{feature} Distribution')
plt.tight_layout()
plt.show()

# Splitting the data into train and test datasets.
X_train, X_test, y_train, y_test = train_test_split(data, y,
test_size=0.30)

# Splitting the data into train and validation datasets.
X_train, X_validation, y_train, y_validation =
train_test_split(X_train, y_train, test_size=0.20)

# Define hyperparameter search space
alphas = [0.001, 0.01, 0.1]
lambdas = [0.001, 0.01, 0.1, 1.0]
num_iterations = [10, 100, 1000]

# Finding the best hyperparameters
for alpha in alphas:
  for num_iteration in num_iterations:
```

```
    theta, mean, std, cost_history =
linear_regression(X_train.to_numpy(), y_train.to_numpy(), alpha,
num_iteration)
    y_pred = predict(X_validation.to_numpy(), theta, mean, std)
    r2 = r2_score(y_validation, y_pred)
    print(f'The R2 score of the linear regression model with alpha:
{alpha} and number of iterations: {num_iteration} = {r2}')

# Testing the linear regression model with the best hyperparameters on
test data.
theta, mean, std, cost_history = linear_regression(X_train.to_numpy(),
y_train.to_numpy(), 0.1, 1000)

# Predicting the values of test data
y_pred = predict(X_test.to_numpy(), theta, mean, std)

# Evaluating the model
r2 = r2_score(y_test, y_pred)

# Printing the results
print("R2 score of the model with best hyperparameters on test data =
", r2)

# Finding the best hyperparameters
for alpha in alphas:
  for lambda_ in lambdas:
    for num_iteration in num_iterations:
      theta, mean, std, cost_history =
ridge_regression(X_train.to_numpy(), y_train.to_numpy(), alpha,
lambda_, num_iteration)
      y_pred = predict(X_validation.to_numpy(), theta, mean, std)
      r2 = r2_score(y_validation, y_pred)
      print(f'The R2 score of the ridge regression model with alpha:
{alpha}, lambda: {lambda_}, and number of iterations: {num_iteration} =
{r2}')

# Testing the ridge regression model with the best hyperparameters on
test data.
theta, mean, std, cost_history = ridge_regression(X_train.to_numpy(),
y_train.to_numpy(), 0.1, 1.0, 1000)

# Predicting the values of test data
y_pred = predict(X_test.to_numpy(), theta, mean, std)

# Evaluating the model
r2 = r2_score(y_test, y_pred)

# Printing the results
```

```
print("R2 score of the model with best hyperparameters on test data =
", r2)


# Adding new features to the dataset whose values are the exponential
values of the existing feature values.
def add_polynomial_features(X, max_degree):
    # Add polynomial features up to the specified degree.
    X_poly = X.copy()
    for d in range(2, max_degree+1):
        X_poly = np.hstack((X_poly, X**d))
    return X_poly

X_train_poly = add_polynomial_features(X_train, 2)
X_validation_poly = add_polynomial_features(X_validation, 2)
X_test_poly = add_polynomial_features(X_test, 2)


# Finding the best hyperparameters for polynomial regression model
for alpha in alphas:
  for lambda_ in lambdas:
    for num_iteration in num_iterations:
      theta, mean, std, cost_history = ridge_regression(X_train_poly,
y_train.to_numpy(), alpha, lambda_, num_iteration)
      y_pred = predict(X_validation_poly, theta, mean, std)
      r2 = r2_score(y_validation, y_pred)
      print(f'The R2 score of the polynomial regression model with
alpha: {alpha}, lambda: {lambda_}, and number of iterations:
{num_iteration} = {r2}')

# Training the Polynomial regression model with the train data
X_train_poly = add_polynomial_features(X_train, 2)
theta, mean, std, cost_history = ridge_regression(X_train_poly,
y_train.to_numpy(), 0.1, 0.0001, 1000)

# Predicting the values of test data
X_test_poly = add_polynomial_features(X_test, 2)
y_pred = predict(X_test_poly, theta, mean, std)

# Evaluating the model
r2 = r2_score(y_test, y_pred)

# Printing the results
print("R2 score of the polynomial regression model with best
hyperparameters = ", r2)
```