Housing Prices

Anonymous

Submitted for the Degree of Master of Science in

Data Science and Analytics

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

August 29, 2024

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 Where other published and unpublished source materials have been used, these have been acknowledged.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: 14,535

Student Name: Anonymous

Date of Submission: 5th September 2024

Candidate number: 2405616

Abstract

Predicting house prices is a common measure used in real estate analysis for price determination and investment strategy, as well market forecasting and economic assessment. We examine a number of machine learning approaches for predicting housing prices from the Boston Housing and Ames Housing datasets. Here, we want to see how different regression models viz Linear Regression, Ridge, Decision Tree and Random Forest forecast the housing prices as well.

The research starts with a detailed literature review, which emphasizes information related to real-estate and the necessity of using machine learning in it includes methods used for predicting prices. The methodology section of Table 2 explains the data preprocessing such as scaling features and handling missing values to ensure optimum performance. The different regression techniques are applied and analysed using the metrics MSE, RMSE, R^2. On the test sets, our results show that ensemble methods (Random Forests and Bagging Regressors) perform significantly better at prediction than traditional linear models. These models provide better accuracy and generalisation, as can be seen with lower values of MSE & RMSE for each one along with R^2 scores. The project also contains an inflation analysis of the Ames Housing dataset which converts sale prices to 2010 dollars so that readers can more easily understand house price trends historically. The comparative analysis between the different models will give a lot of insights into their strong and weak points. Since these models are capable enough to capture complex interactions and non-linearity between independent variables, they can easily outperform simpler ones like Linear Regression. This is then enriched even further with the inflation adjustment which shows how price changes historically effect what you are buying today.

In addition to contributing new knowledge how machine learning can improve housing price prediction, our project suggests some practical implication for the real estate professionals and researchers. This enables a more consistent evaluation between models and opens avenues for future research, in particular because the housing market analysis community keeps introducing novel model variants into practice.

INDEX

1. Introduction

1.1 Motivation:
Housing price prediction is a task that can have severe impacts on many aspects of both the economy and personal finance. Prediction of housing prices is significant for different stakeholders including prospective home buyers, real estate developers and landlords, financial institutions that underwrite or fund mortgages as well policy makers to judge the market. As a home buyer : anybody can take an informed decision with the future prices when buying properties. It can also help mortgage lenders better understand the risk associated with home loans, which in turn affects the terms and conditions of financing options offered to borrowers [3].

Additionally as housing prices are an important economic indicator that shows us the strength of a country's economy. Changes in home price levels and increases or decreases should be a strong indicator of economic conditions, influencing consumer spending all the way to government fiscal policy. For example, the fact that housing prices are increasing could mean an increase in economic growth, evidenced by increased demand for housing as well and disease among consumers confidence. The decrease may also be indicative of recession or a slowdown down with less consumer spending (11).

When it comes to urban planning and development, precise housing price forecast is a cornerstone for sustainable advancement. Forecasters, city and regional planners and developers use these predictions to help them decide where resources should be allocated, new infrastructure built or plan for future housing needs. Predicting housing prices can help them identify the areas where growth is most likely to occur and conversely which parts are past their prime, enabling more efficient planning [2]. This project seeks to create machine learning models that are able to predict housing prices in 2 separate datasets, the Boston and Ames Housing markets. In addition to the academic theories, these models can inform policy for urban planning in real estate and economy[1].

1.2 Aims and Objectives:
This project focus on the Boston and Ames Housing Markets aiming to create predictive models with machine learning process. Using these techniques, the project aspires to do comparative study among various models and decide which one work best in prediction of housing prices. [2].

Specifically, the objectives of this project are:

- **Data Preprocessing:** Clean, normalize and preprocess the Boston as well Ames datasets ready for machine learning modelling. It is an important step for data preprocessing before the models are trained to make sure that they perform well by filling missing values, scaling features or fixing any inconsistency in the data. [22].

- **Develop Models:** Train a variety of machine learning models such as Linear Regression, Decision Tree, Ridge Regression. Support Vector regression and Random Forest to predict house prices. After training and testing each one of the models with the datasets, they never scored equally well; a good way to score those methods are through key performance metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE) and R-squared — R^2 [4].
- **Comparative Analysis:** Researchers can then conduct a Comparative Analysis when evaluating the models to see which one produces predictions that are most in line with expectations. To do so, we will evaluate the predictive capability of these models and whether their complexity is justifiable, in relation to how interpretable they are as well as if they can be used for generalizing data at deployment. [5].
- **Inflation Adjustment Analysis:** Use the Ames dataset to A) Get a breakdown of how old housing prices would fit in today (eg. all 2010 houses into current economy), and filter those batches B) Produce error measurements based on adjusted Zillow predictions. This analysis would give us a clear picture of what the real value change for years passed by accounting inflation [3].

As such, the project success will yield important lessons for applying machine learning to real estate price prediction that can inform both academic and commercial approaches [1].

1.3 Contributions to Future Career :
The project delivered in this post has provided me with highly communicable skills that will greatly improve my future prospects and enrich the work I do, especially when it comes to data science, machine learning, real estate analysis. Being able to deploy machine learning models on real-world datasets like the ones examined in this work is a profitable and highly applicable task that more industries are requesting for [4]. Well, for a start this project incorporated basic data preprocessing which is the most important thing in any data science project. The better we are at cleaning and preparing data, the more reliable our models will be. These general skills are critical for just about any role in the realm of data analytics or machine learning, as is evident when predictive modelling falls down due to inadequate data quality [22].

But also, having implemented these models to then compare has helped me understand the power of them specifically and respective gaps. This especially applies to verticals such as financial services, healthcare and technology where predictive accuracy plays a critical part in strategic decision making [5].
It also turned me into a better problem-solver and critical thinker. Creating a reliable model is just as technical an effort, but it requires qualitative skill at interpreting results and error solving to retrain models. These are highly sought after skills for any analytical role, particularly in high- data environments like actuarial science, economics and business intelligence [8].

Additionally, conclusions based on historical trends in inflation-adjusted housing prices shed light into the repartee between macroeconomic factors and real estate markets. These dynamics are important for urban planning, real estate investment and economic policy roles where long-term trends and market behaviour is crucial [3].
All in all, this project gave me a complete skill set that I can quickly use toward my future career. My skillset a technical prowess, ability to think analytically and practical exposure to real-world data has equipped me with the requisite knowledge for success in this field that is rapidly changing [9].

2. Background Research

2.1 Literature Review:
The prediction of housing prices is an important research area as it concerns a lot stakeholders including home-owners, investors and policy-makers [3]. Early research predominantly used statistical models like linear regression and these formulations paved the way to establish associations of variables such as location, crime rates or educational facilities with housing [4]. However, these models were bounded by their ability to model the complex and often non-linear relationships in real estate data [5].
House price prediction is a crucial research domain as it benefits many people including homeowners, investors and policy makers [3]. Its early research primarily relied on statistical models such as linear regression, providing the hustle for conducting non-causal driven studies of unaccustomed variables like location-crime rates-educational facilities with housing [4]. However, these models were limited in their ability to represent the complexity and non-linearity that is presented by real estate data [5].

Data pre-processing is also a vital step in predictive modelling. Hyper parameters Pre-processing techniques (Normalization, Standardization and Feature Selection once) as important to improve the performance of a model [22]. Literatures have demonstrated that the methods of addressing missing and outliers could achieve so this will be favourable to make predictions about data [10]. Indeed, imputation methods and transformation of skewed variables to the normal scale or removing bias in data cleaning may well increase a model ' s robustness and accuracy [22].

Decision tree regression: A decision tree regression model forms part of non-parametric supervised learning and is utilized in both classification as well as a type of the dependent variable. Prediction by making use simple decisions based on data features. Some of the reasons why this model is routinely used are: it can handle both categorical and continuous variables, its relatively simple method gives an understanding behind predictions [15].
It works on recursive splitting of data based the value of input features. In terms of a machine learning approach, the tree structure consists of nodes for decisions similar to feature (or attributes such as yes and no), leaf nodes that contain output. Note: At each node the split is done on some feature over MSE (mean square error) type of criteria which tries to minimise within variance in all subsets produced by every split [3].

Why Decision Tree Regression Works:

- **Non-linear Relationships:** Decision trees can model non-linear relationships between features and the target variable. In some cases, this attribute might improve the assignment when a linear model fails. So they can only be used where the dependent variable is best approximated by a stepwise rather than linear function – as opposed to regression techniques [15].

- **Intuition:** Decision trees are very easy to interpret the so we say that model is interpretable. All edges that lead from the root and terminal nodes represent an unconditional statement about a variable. Therefore, decision trees are very explainable models.
- **Pruning**: Decision Trees are good at handling data, but they keep on growing without caring overfit the available training dataset. The model could be overfitting, meaning that its complexity is causing it not to generalize well to new observations or noise. By pruning, we can modify the decision tree because pruning is a method that requires us to get rid of all those nodes which contributed very little towards the model representation power[15]. Pre-pruning and post-pruning could be applied to the pruning.
- **Machine Bias-Variance**: Decision trees by tweaking with the depth of the tree can balance out bias-variance trade-off. Shallow trees tend to be more biased but have lower variance. Conversely, deep trees reduce bias and increase variance that can be controlled by pruning or different regularization techniques.

What does Ridge Regression do:
It is a regularization method of linear regression, and it has two main features that can solve multicollinearity problem or estimation bias in Ordinary Least Squares (OLS) model because predictors are highly correlated. Ridge Regression adds a regularization term to the objective function which penalizes large weights and thereby keeps your model stable[18].
Normally in Ridge regression model, the RSS is modified by multiplying with penalty which is equal to square of coefficient values. A cost function is a mathematical representation of how far we are from the true value.

$$\text{Cost Function} = \text{RSS} + \lambda \sum_{j=1}^{p} \beta_j^2$$

where λ lambda if the regularization parameter to control how much you fine tune penalty This penalty term acts as a way of punishing high-valued coefficients, making the model more general and avoiding overfitting [18].

Why Ridge Regression Works

- Multicollinearity: Ridge Regression is good with multicollinear datasets because it penalizes the coefficients and provides coefficient estimates a stability. This results in more reproducible and interpretable models [18].
- Bias-Variance: This adds bias but is relatively effective at reducing variance, thereby lowering the out-of-sample prediction error [19].
- If we were to colour our interpretation of the above expression with a Bayesian brush, then it would be as if we are simply placing some priors on their coefficients… For starters, suppose that each coefficient is normally distributed (around 0), and has an inverse-variance proportional to λ. This interpretation justifies the bias introduced by the regularization as it enforces a soft constraint on coefficients [5].

- Geometric interpretation: RR constrains coefficients onto a hypersphere, keeping them from becoming too large, an issue faced in high-dimensional data scenarios where the number of predictors exceeds the observations [19].

A Support Vector Machines (SVM): can be used in an indirect way to training our model named SVR, the SVM algorithm is applied with modification so it become powerful machine learning family called support vector regression. The first one is used for predicting continuous values, it works great with regression tasks. While remaining at least ε careless of the particular target values, (SVR) tries to determine a function which curves less than possible from actual target attributes [26].

How does Support Vector Regression Works:
The idea behind SVR is entirely based on the previous version of SVM that we know as Classifier. As it projects the high-dimensional feature space into a space having lower dimensions, which provides an opportunity for the model to capture complex relationships between features and target variable that would not be possible in original feature-space. SVR attempts to search for a linear hyperplane in this higher-dimensional space so that it fits the data within epsilon [17].

- It uses Kernel Trick and Non-Linearity- SVR can map data into a higher dimension space allowing more complex relationships between variables, however the computation is costly. The selection of the kernel function (linear, polynomial and radial basis functions in this option) plays a main role so that the model can adapt certain patterns from data [17].
- **Margin Maximization**: Similar to SVM, SVR seeks to maximize the margin between the predicted values and the actual data points, which helps in improving the generalization of the model. This is particularly useful in ensuring that the model performs well on unseen data, reducing the risk of overfitting [26].
- **Bias-Variance trade-off**: SVR is good when we want to balance out bias and variance. The weights of the margin tolerance and regularization parameter (C) are both used to control overfitting: negatively biasing our model through complexity against way better ability to generalize as they trade against each other. Striking this balance is instrumental to obtain a model that does not overfit or underfit the data [26].
- **Convex Optimization**: he optimization problem is convex in SVR so that it has a unique global solution. This property makes the model stable and reliable even in confronting difficult, high-dimensional data sets[17].

Random Forest Regression It is an ensemble method from machine learning, using Decision Trees and takes the average gain of Tree predictions. This method combines bagging (Bootstrap Aggregating) of multiple models and randomness in feature selection that attempts to reduce variance and overfitting results [6].

How does Random Forest Regression Works:
This Algorithm is generated great number of decision trees for predicting the target values and outputting the average prediction of all individual trees. Every tree in the ensemble is constructed from a bootstrap sample of data (sample drawn with replacement), and at every split within each tree, only use a random subset of features. This randomness decreases the correlation between trees, which lowers variance of an ensemble relative to that off individual trees [6].

- **Bagging:** Training each tree on a random subset of the data helps in variance reduction without significant blowup in bias. This is due to the fact that a single decision tree can be sensitive to noise within the training data, but averaging many trees decreases this sensitivity [14].
- **Out-of-Bag Error**: That brings us to another interesting point about Random Forests — OOB error estimation. Given that each tree is trained on a bootstrap sample, roughly one-third of the data not used can be employed as a test to measure how will our model performs leading to an embedded cross-validation procedure without having another validation set [6].
- **Overfitting:** as opposed to individual decision trees, Random Forests averages predictions of many classifications or regression that tends to underperform on the data only upon which it has been trained. If the model is made too complex, overfitting results and instead of fitting to an underlying pattern in a data series [14], it begins to fit noise.
- **Bias-Variance**: The bias of Random Forests generally increases due to the fact that it is averaging the results from many trees and this inadvertently decreases variance. Well yes, this averaging slight bias but the overall mean squared error drops due to less variance which makes a dependable and generalizable model [6].
- **Feature Importance:** Random Forests give the best estimation of feature importance. Through considering the distribution of change in node impurity (e.g., Gini impurity or variance) to train a sensible model, it is able to rank features based on their predictive importance [25].
- **Dealing with high dimensions data:** Random Forest performs well even though the datasets have a ridiculous number of features and some may not mean at all. Random Forest can survive in a high-dimensional space or with many irrelevant features since it grows trees on random subsets of the feature dimensions [25].

2.3 Dataset Overview
One of the most famous datasets in machine learning, which is often used as a benchmark by regression models [13], production-scaling recommender systems. The data set of 506 entries representing doctors from a particular sub region or town in Boston. It comprises of 13 important features like CRIM (crime rate), INDUS (proportion of nonretail business acres per town), NOX(Nitric oxides concentration), RM(average number of rooms per dwelling). The target variable is the median value

of owner-occupied homes (MEDV) in thousands of dollars in any case, this dataset is ideal to test different regression models for your project due its size and number of features of which is required a good understanding in order to extract the relevant ones or make it not just overfit on them. It is a good example of size and with already covers most well-known features that are heavily used individually, hence makes it an ideal dataset to analyze different machine learning algorithms in details [13].

On the other hand, then Ames Housing dataset which includes 2,930 entries with 80 features describing aspects of residential homes in Ames Iowa [7]. These features are variables like the Overall Quality of material and finish, Year house was built in and the square footage living area The dependent variable in this dataset is the sale price of the houses. This dataset is more difficult for machine learning models because its higher dimensionality and complexity. It's truly an invaluable resource when you want to test the scalability and robustness of these models in your project. Including the Ames dataset with increase the opportunity to work on more advanced feature engineering and selection, as well check your models generalizability for a complex problem [7].

I needed both datasets to ensure all machine learning models were being evaluated across the board in my project. The Boston dataset was used as a baseline model, given its straighter forward structure and interpretability. As a contrast, opportunities to test the abilities of models and feature uses with their strengths considering balance at those constraints in an urban area using Ames dataset [13].

3. Methodology

3.1 Boston Housing Dataset:

3.1.1 Overview of the Boston Dataset:

The Boston Housing Dataset is a classic and very widely used dataset in machine learning, specially often seen in the context of regression tasks. Dataset Overview The dataset has 506 instances, a combination of 14 features and the target[13]. It comprises 13 features such as CRIM (crime rate per capita), ZN — proportion of residential land zoned for large plots, INDUS — proportion of non-retail business acres per town, and RM - average number of rooms to dwelling etc. The target variable includes the median value of owner-occupied homes (MEDV) measured in $1,000s. It is a popular dataset for benchmarking regression models since its size is relatively small and the properties of it are well-documented [13].

It is a crucial dataset for investigating real-world data problems, such as nonlinearity and heteroscedasticity. Additionally, a concern with the MEDV feature (the target variable) is its notably skewed distribution having many values around $50,000 or so triggering that there may have been capping occurring during the data collection process [13].

3.1.2 Data Preprocessing for Boston:

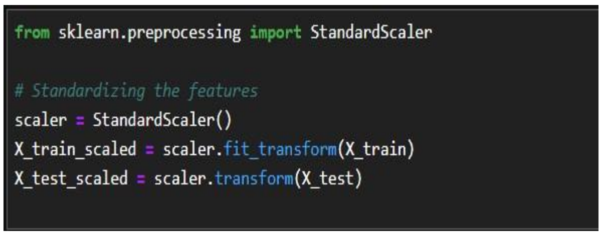
Before we can even model the Boston Housing Dataset, Data Preprocessing is a very important step. We did some preprocessing:

Handling Missing Values:

Checking Missing values in Boston Housing Dataset using the. isnull(). The dataset is given through sum() method, revealing that there are no missing entries across any of its features. The lack of missing data additionally facilitated the preprocessing stage, utilizing imputation strategies for NA handling such as filling in absent values with mean or median [22].

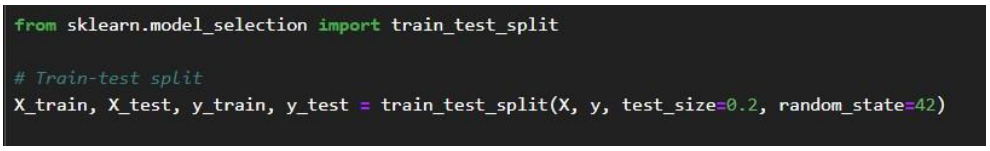
Standardize:

The features have been normalized by Standard Scalar from Scikit-learn with mean 0 and variance of 1. It keeps all values of features cantered around zero and within the same order of magnitude. This is important in practice given that algorithms, such as Ridge Regression and Support Vector Machines (SVM) are scale sensitive [20]. Here the code for Normalization is given Below:



```
from sklearn.preprocessing import StandardScaler

# Standardizing the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

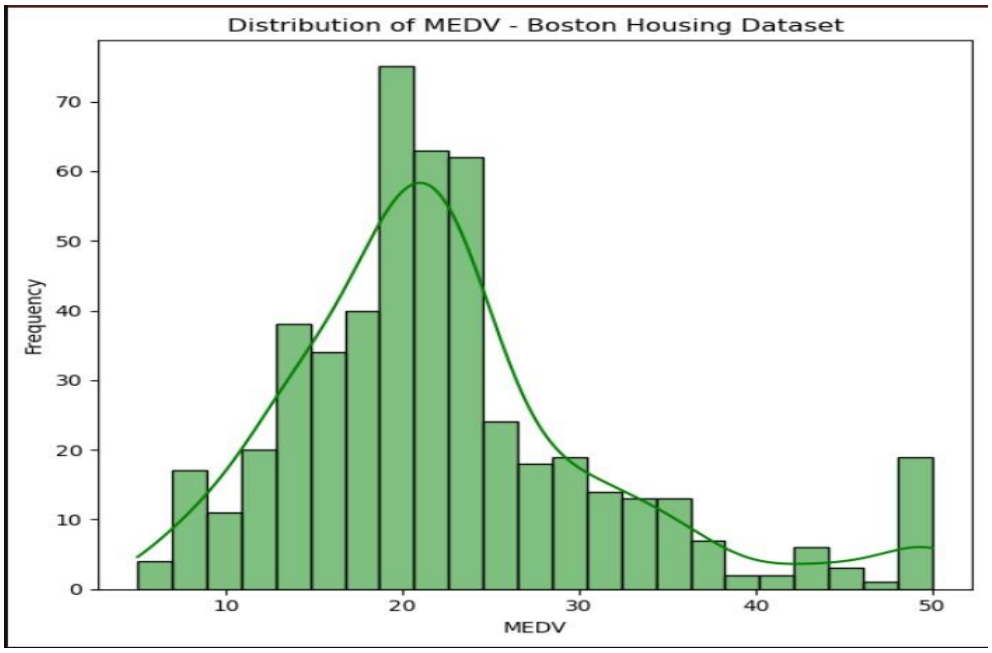
Train-Test Split:
It was then split the data into 80 % training and 20% testing. This split enables the evaluation of how well model will perform on data that it has not seen, important measure for generalization [27]. The split was done using the following code:



```
from sklearn.model_selection import train_test_split

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

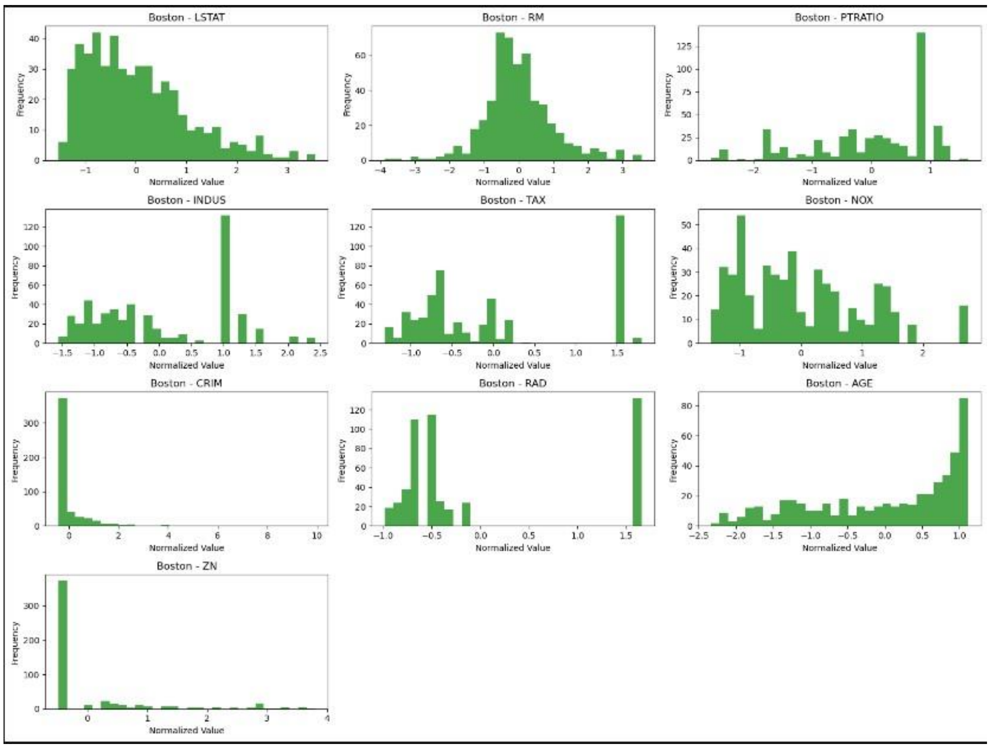
Feature Scaling:
Feature scaling in addition to normalizing data is done because it makes sure that all the features have an equal contribution towards predicting and none of them gets an upper hand over others just because it has large ranges. During model training, efficient gradient descent is necessary for this to work and hence proper scaling [20].

Distribution of MEDV:
The target value MEDV distribution was explored and it turned out slightly right-skewed with many houses gathered around 50,000$. This ceiling effect indicates that the dataset might not represent high-value homes well and this could affect how regression models will perform [13].



Top 10 Features for Boston:
The top 10 features that is most correlated with MEDV was found via a correlation analysis. This is a critical step to help improve the interpretability and performance of your model. RM, LSTAT and PTRATIO are shown to correlate with the other

response variables in order of connectivity that represents three features that strongly affect median home value. [13]. Visualize the correlation matrix as below:



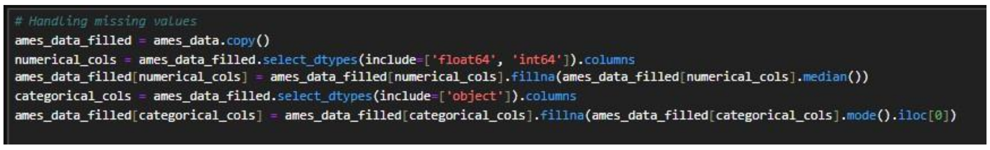
Distribution and Correlation Analysis:
It was done so by plotting other detailed visualizations which included histograms and correlation matrix graphs (of MEDV vs all features) to understand the distribution of a feature individually as well as among different factors affecting target value that is MEDV. These insights were important for subsequent dimensionality reduction efforts, as they served a guide in selecting the most pertinent features [13].

3.1.3 Ames Housing Dataset:
The Ames Housing dataset provides a more extensive and complicated counterpart to the Boston Housing dataset. This dataset contains 2,930 records of home sales in Ames, Iowa sold over the years from 2006-2010 each record is comprised of approximately 80 features that describe various aspects about a house such as quality material used to build the property size living area year built how close its nearness to amenities etc. In this dataset, the sale price of cabins Mills in dollars [7] is target variable. This dataset is particularly well-suited for predictive modelling as it contains a large number of housing features and exhibits challenges that are commonplace in real-world data, such as multicollinearity and missing values [7].

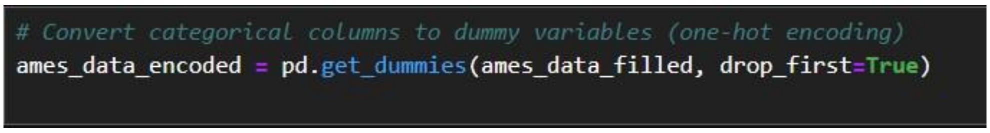
There is a mix of features in the Ames dataset, from categorical variables like 'Neighborhood' and 'BldgType', to numerical ones such as our dependent variable 'GrLivArea' (above ground living area) or another independent one called 'GarageArea'. Altogether, these features permit a comprehensive analysis of what contributes to the housing price and therefore could serve as a great dataset for practice in training machine learning models, especially those more suited to regression tasks. The target variable, SalesPrice, is right skewed (having few properties with high value) This property is important because it has implications on the choice of model and feature preprocessing such as transforming target variable to try a better normal distribution properties which may lead for improved predictive performance [7].

Because of its size and complexity, the Ames Housing dataset has become a cottage industry for testing regression models [7], often as one data points in machine learning competitions or exercises on using ancient ML techniques to make predictions.

3.1.4. Preprocessing for Ames Dataset
Data processing is essential in a machine learning pipeline to clean up the dataset and make it well-suited for model training. Ames Housing: A dataset listed in Kaggle that had missing values, feature scaling and encoding of nominal data [7].
Data pre-processing Information: Step1 Missing Values Identification and Handling The dataset consists of numerical and categorical features, each requiring its own method to deal with missing data. Imputation of missing values in numerical features was performed by replacing them with median over all non-missing observations, it is a robust method that avoid biases due to outliers [22]. Some of the data missing is just input error, i.e., none. we can easily fill it using intuitive value possible like take median or mode to do this while keeping most information as in LotFrontage[7].

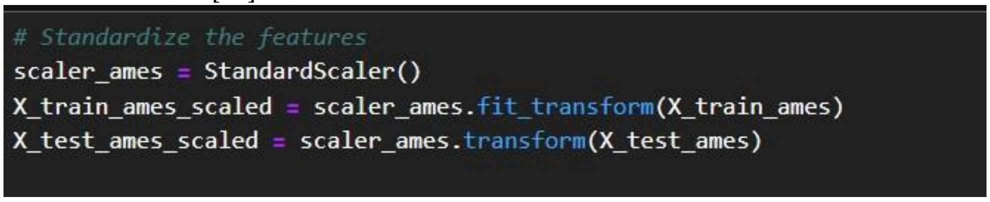

```
# Handling missing values
ames_data_filled = ames_data.copy()
numerical_cols = ames_data_filled.select_dtypes(include=['float64', 'int64']).columns
ames_data_filled[numerical_cols] = ames_data_filled[numerical_cols].fillna(ames_data_filled[numerical_cols].median())
categorical_cols = ames_data_filled.select_dtypes(include=['object']).columns
ames_data_filled[categorical_cols] = ames_data_filled[categorical_cols].fillna(ames_data_filled[categorical_cols].mode().iloc[0])
```

For categorical features, the most frequent category was used for imputation, as this method is effective in preserving the mode of the data distribution [22]. For instance, missing entries in the 'Electrical' feature were filled using the mode, ensuring that the most common electrical system type remains predominant in the dataset [7]
Next, One-hot encoding was used to convert the categorical variables into numerical. This method creates binary columns for every category, which is especially useful to functions needing numerical input like linear regression or support vector machines [20].



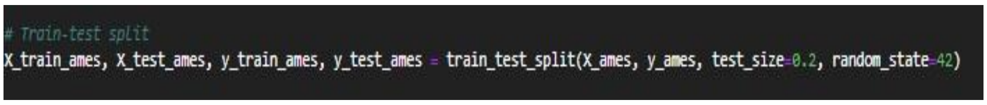
```
# Convert categorical columns to dummy variables (one-hot encoding)
ames_data_encoded = pd.get_dummies(ames_data_filled, drop_first=True)
```

Next, it was essential to perform feature scaling since some that depended on the scale of input data: ridge regression and support vector regression. The features were standardized such that they had a mean of zero and standard deviation one so as to prevent some data points from affecting the classification performance significantly more than others [20].



```
# Standardize the features
scaler_ames = StandardScaler()
X_train_ames_scaled = scaler_ames.fit_transform(X_train_ames)
X_test_ames_scaled = scaler_ames.transform(X_test_ames)
```

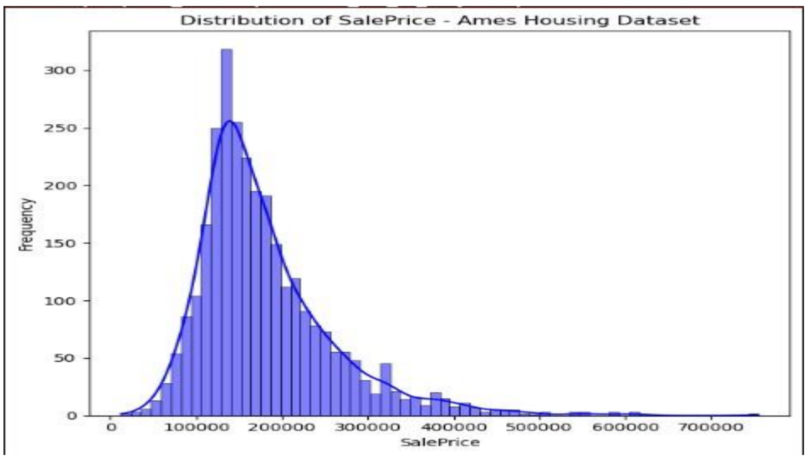
The dataset was then split into training and testing sets using 80% of the data for training, and the remaining 20% used to test. This step will help you to evaluate your model on unseen data as opposed to the training/validation dataset and hence gives a more accurate idea of how well it generalizes [27]. We visualized the preprocessing steps to verify transformations were working with SalesPrice — distribution plot and top 10 correlated feature variables [7].



```
# Train-test split
X_train_ames, X_test_ames, y_train_ames, y_test_ames = train_test_split(X_ames, y_ames, test_size=0.2, random_state=42)
```

These are the preprocessing steps that make sure Ames Housing dataset was ready for training machine learning models by handling common problems such as missing data and uneven feature scales, all important to achieve accurate and robust predictive models [7].

Distribution of SalePrice:

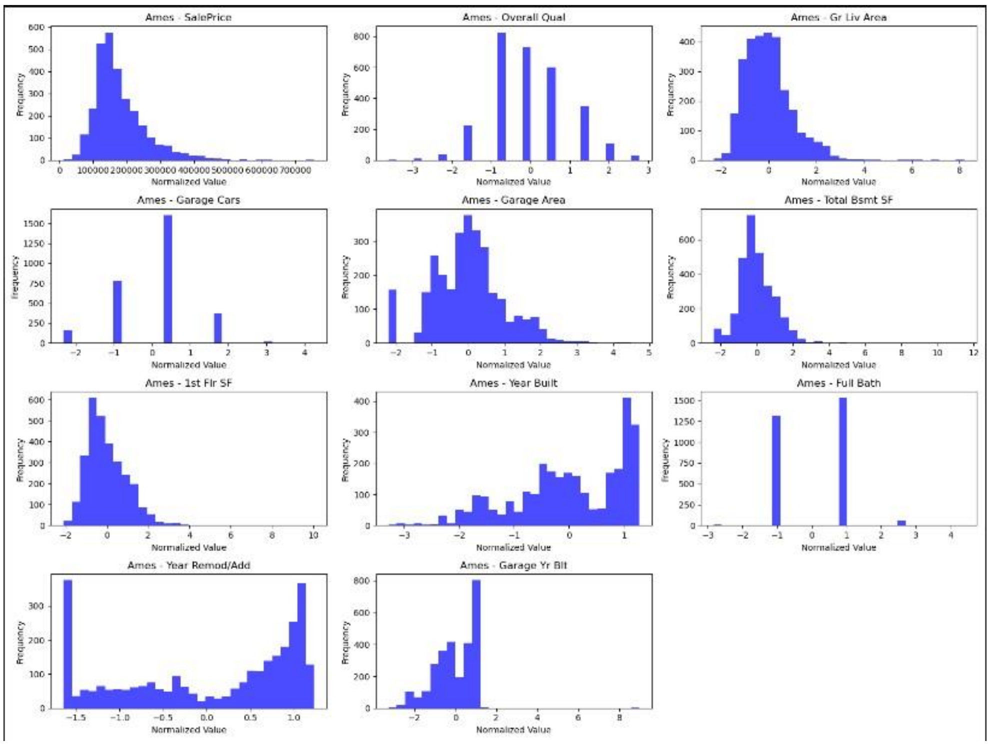


Target distribution of Sales Price:

In Ames Housing dataset (the first graph) The distribution is very skewed to the right, with a large number of home prices clustering around $100–200k. The positive skew shows that most homes are priced at under $400,000 and some as high a s$7000. This skewness can be problematic for regression models, especially ones that assume a normally distributed target variable (e.g. linear regression). This could result in lower accuracy since the model cannot entirely learn about this distribution, because of it not being normally distributed! This issue can be alleviated by normalizing the target variable or using models that are robust to skewness, such as tree-based models [7].

Normalized Distribution of Outlier: Top Features:

The second set of graphs shows the normalized distribution in which various SalePrice correlated features are displayed. Some of the features have a weak correlation with our dependant feature SalePrice, for example Overall Qual, Gr Liv Area ,Garage Cars, Garage Area and many others. It is important to normalize features so that each feature contributes equally to the model predictions also when measured on different scales. For example, Overall Qual and Gr Liv Area are two of the most powerful predictors of SalePrice as expected since higher-quality construction tends to come with larger living areas which usually translates into a nice price for buyers eventually [7]. We normalize these features as this will make our feature then comparable on the same scale which raises any kind of model performance, not only linear but also tree-based models [20].



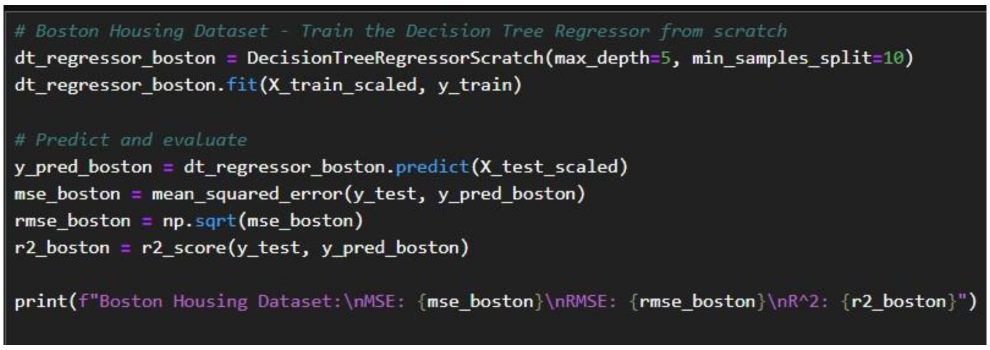
Overall, these analyses serve to highlight the necessity of being aware not only of how the target variable is distributed and scales, but also its construct scaling in relation to that predictor when building predictive models. A proper treatment of these can drastically improve the accuracy and trustworthiness of home prices prediction models [7].

3.2. Model Implementation:

3.2.1. Decision Tree Regression:

Boston Housing Dataset: One of the classic datasets used for regression tasks, 13 features such as number of rooms, property age etc and target value price. Decision Tree Regression, a non-linear model can best capture the interactions between these features recursively trying to split data at nodes based on feature values which minimize the Mean Squared Error (MSE). Decision Trees each split in the tree on a decision rule, which is very intuitive and easy to understand. Therefore they are one of the most used classifiers for this dataset along with KNN based approaches [13].

By applying that algorithm for the Boston Housing dataset, we can see how Decision Tree Regression regressor is iteratively selecting features and thresholds to split at each node such that it aims to a reduction in MSE. This process is done recursively until the maximum depth of tree or minimum number samples per split level are reached. In a smaller dataset like Boston Housing, tuning hyperparameters such as max_depth and min_samples_split is crucial in managing the complexity of the tree better to prevent overfitting [5].



```python
# Boston Housing Dataset - Train the Decision Tree Regressor from scratch
dt_regressor_boston = DecisionTreeRegressorScratch(max_depth=5, min_samples_split=10)
dt_regressor_boston.fit(X_train_scaled, y_train)

# Predict and evaluate
y_pred_boston = dt_regressor_boston.predict(X_test_scaled)
mse_boston = mean_squared_error(y_test, y_pred_boston)
rmse_boston = np.sqrt(mse_boston)
r2_boston = r2_score(y_test, y_pred_boston)

print(f"Boston Housing Dataset:\nMSE: {mse_boston}\nRMSE: {rmse_boston}\nR^2: {r2_boston}")
```

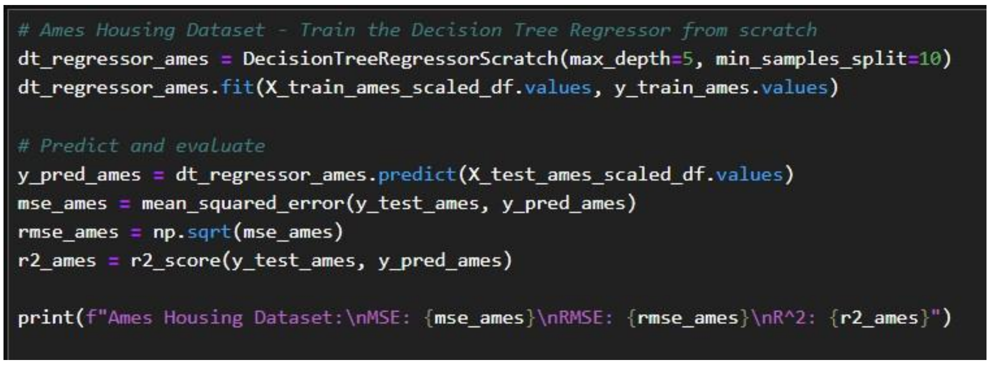
Overfitting and Model Evaluation:

Since the algorithm tries to perfectly split at every possible point in our training data, we end up with a model that fits too well on the training data and performs poorly on new (unseen) records. For the Boston Housing dataset, using a shallow tree with finetuned hyperparameters can act as regularization and makes your model more robust to generalization errors (e. g., overfitting). Finally, pruning strategies c We eventually solve them by removing most of that model structure post-training which is eliminating the branches relating to poor predictive performance and thus gain further generalization edges [5].

Performance:
Performance is often evaluated through metrics like MSE, Root Mean Squared Error (RMSE), and R^2 on the Boston Housing dataset for example. This also explains how effective is a particular model at predicting housing prices based on the current features. E.g., a well-tuned Decision Tree model will give good RMSE to compete with complex models and also interpretability of decision tree [20].

Conclusion:
To sum up, Decision Tree Regression is a robust and interpretable way of modelling the Boston Housing dataset. Advantages simplicity, interpretability but must tune regular right to not overfit by using these techniques, it is ensured that the model can learn from input data effectively as well without breaking its capability to generalize for new unseen samples [5].

Ames Housing Dataset:
The Ames Housing dataset is a more complicate and larger version of the Boston Housing dataset, containing 79 features representing attributes (include metadata) for residential homes in Ames Iowa. The complexity gives breeding to more detailed and accurate predictions that can be made in the dataset using Decision Tree Regression Algorithm. For this, we need decision trees as they can work with both numerical and categorical data and are interpretable which is useful in understanding the relationship between features/target variable [7]
The Decision Tree model in the Ames Housing dataset works on similar principle that of splitting data recursively at nodes according to threshold feature values which gives minimum MSE. However, the model is very sensitive on other hyperparameters such as max_depth and min_samples_split/leaf because of higher dimensionality and complexity in dataset. These parameters help compute the tree from growing too hard and overfitting. Since the Ames dataset is larger, they also fine-tune these hyperparameters by using a more sophisticated technique called CrossValidation to make sure their model generalizes well to new data [20].


```python
# Ames Housing Dataset - Train the Decision Tree Regressor from scratch
dt_regressor_ames = DecisionTreeRegressorScratch(max_depth=5, min_samples_split=10)
dt_regressor_ames.fit(X_train_ames_scaled_df.values, y_train_ames.values)

# Predict and evaluate
y_pred_ames = dt_regressor_ames.predict(X_test_ames_scaled_df.values)
mse_ames = mean_squared_error(y_test_ames, y_pred_ames)
rmse_ames = np.sqrt(mse_ames)
r2_ames = r2_score(y_test_ames, y_pred_ames)

print(f"Ames Housing Dataset:\nMSE: {mse_ames}\nRMSE: {rmse_ames}\nR^2: {r2_ames}")
```

Feature Engineering and Ensemble Methods:
With feature engineering, Decision Trees built on Ames will do better than they would have done otherwise. For instance, categorical encoding, continuous variables

normalization and feature selection can all gave us a great improvement in the accuracy of our model. Also, for example how tree-ensemble methods like Random Forests or Gradient Boosting Machines need the design of each method to construct a forest bundle; because though it's easy that an individual Decision Tree model has both high projection and low variance [20].

Challenges and Overfitting:
So although the Ames Housing dataset hardly measures up to our expectations from learning in class, this does not mean that Decision Tree Regression cannot be an accurate housing price predictor. That is why due to this trade-off of a fair amount of interpretability and handling complex interactions between features, Random Forest became the top choice for many data scientists. But these advantages come at a price, as the added complexity of the Ames dataset reveals limitations in simple Decision Trees, especially with regard to overfitting (the model was not regularized or tuned) [5].

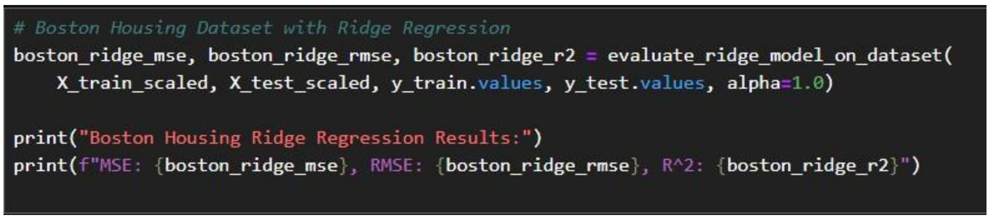
Conclusion:
So to wrap it up, unlike Boston's dataset that is not very complex in comparison with Ames Housing history data set, Decision Tree Regression offers a good way of modelling these complicated relationships. Success lies in manipulating your feature set, tuning hyperparameters maybe using ensemble methods to eke out more performance improvements. As a result, by making adjustments based on these factors Decision Tree Regression is able to deliver precise and interpretable predictions necessary for in the real-world scenarios such as housing price prediction [5].

3.2.2 Ridge Regression

Boston Housing Dataset:

Ridge Regression and Multicollinearity:-
Ridge Regression is a regularized form of linear regression that works really well when you have multicollinearity in the dataset, or even worse, your number features (n) are way too more-than-required observations. This regularization technique adds a penalty term (controlled by the hyperparameter alpha) to the loss function, which shrinks less important feature coefficients causing weaker features shrink close zero and minimises model complexity preventing overfitting [18].

Application to the Boston Housing Dataset:
Ridge Regression on the Boston Housing Dataset can alleviate some of these issues associated with having many features in a dataset. The data set may have variables that are highly correlated (like average number of rooms per dwelling, property tax rates and etc.), in such settings multicollinearity can make the coefficient estimates unstable in a linear regression model. To solve this problem, Ridge Regression can be used since it specifies a penalty to the large coefficient values (which improves generalizing to unseen data) [5].



```
# Boston Housing Dataset with Ridge Regression
boston_ridge_mse, boston_ridge_rmse, boston_ridge_r2 = evaluate_ridge_model_on_dataset(
    X_train_scaled, X_test_scaled, y_train.values, y_test.values, alpha=1.0)

print("Boston Housing Ridge Regression Results:")
print(f"MSE: {boston_ridge_mse}, RMSE: {boston_ridge_rmse}, R^2: {boston_ridge_r2}")
```

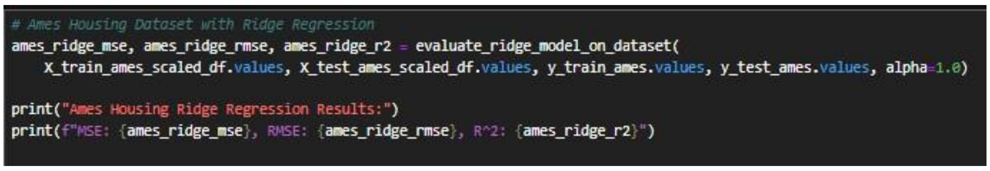
Implementation of Ridge Regression and Standardization:
In the implementation of Ridge Regression on Boston Housing dataset, it is necessary to scale all features together because otherwise there will be no guarantee that regularized term would apply uniformly for each column. The model is then trained with regularized normal equation that minimizes the MSE and also affects the magnitude of coefficients. It is used with predictive modelling and it checks the model evaluation by metrics like MSE, RMSE or R^2 for testing its accuracy & goodness of fit on r-Squared (R2) [5].

Outcome Results and Model Tuning:
In practice, working on the Boston Housing dataset we can verify that this Ridge Regression model results in a more stable and performative predictive model, possibly compared to an ordinary linear regression. Model. This is clear as the RMSE and R-squared values have increased, so our model should be capable of generalizing to new data better. Besides, it allows us to set alpha level as a hyperparameter in order find out the best compromise between predicting power and overfitting for specific characteristics of dataset [18].

Conclusion and Benefits:
Summary: Ridge Regression provides a strong foundation for modelling the Boston Housing dataset due to multicollinearity problems. With the introduction of regularization, we balance model bias and variance that ultimately provide decent results for accurate predictions needed in real estate application like housing price estimation [5].

Ames Housing Dataset
With its 79 features, the more complex Ames Housing dataset is a challenge for regression models because of high dimensionality (potentially leading to overfitting). Ridge Regression is very appropriate on this dataset because it introduces a regularization term to mitigate the influence of individual variables by shrinkage, thus preventing overfitting [18].

When applying Ridge Regression to the Ames Housing dataset, this regularization factor introduced by alpha parameter plays in our favour as it penalizes large coefficients. This is particularly valuable in a high-dimensional dataset such as Ames, at which the variance of overfitting was quite large considering many observations. Ridge Regression therefore effectively shrinks the influence of less useful predictors so that model builds are focused more heavily on using and correctly weighing important indicators 18 to make accurate, generalisable predictions.[18]



```
# Ames Housing Dataset with Ridge Regression
ames_ridge_mse, ames_ridge_rmse, ames_ridge_r2 = evaluate_ridge_model_on_dataset(
    X_train_ames_scaled_df.values, X_test_ames_scaled_df.values, y_train_ames.values, y_test_ames.values, alpha=1.0)

print("Ames Housing Ridge Regression Results:")
print(f"MSE: {ames_ridge_mse}, RMSE: {ames_ridge_rmse}, R^2: {ames_ridge_r2}")
```

Model Training and Standardization:
In the case of Ridge Regression on Ames data set, our first step is to standardize all features so that regularization can be fairly applied across variables. This is normalized using normal equation with regularization and the model will be trained on this updated version of data where alpha term specifies the severity of normalization. Such metrics are used to see how accurately your model can predict considering new dataset, so the overall performance of a model is validated using MSE RMSE and R-squared which gives us measurement related with the prediction tendencies.

Handling Multicollinearity: Advantages
A big reason why people use Ridge Regression on the Ames Housing dataset, is that it helps you deal with multicollinearity which can be a real problem in large datasets. The addition of the regularization term helps to stabilize coefficient estimates, this makes the model less sensitive to fluctuations in data and more generalizable when given new make it models unseen observations [18].

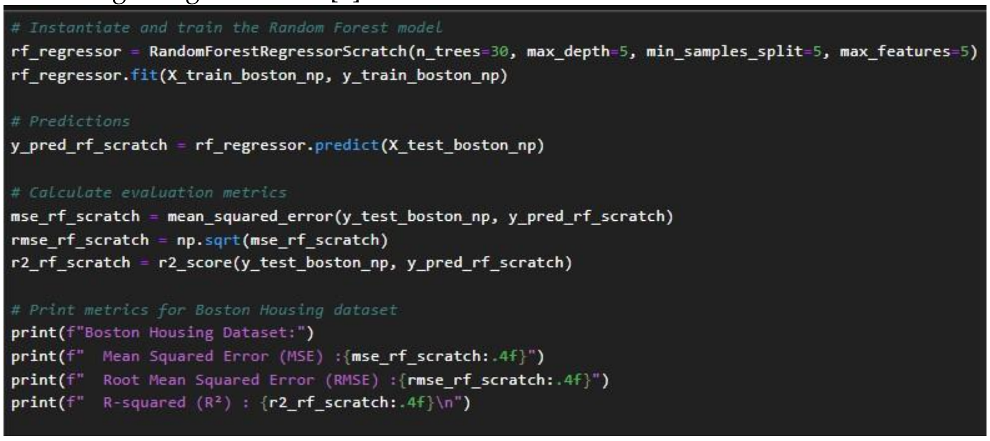
Outcomes In Practice and Some Adjustments:
The application of Ridge regression to the Ames Housing dataset in practice yields a model that has little RPMSE, MASE and WAPE; this suggests it is able to deal well with the complexity within our data. The model can be fine-tuned to strike the best balance between bias and variance, both being accurate and generalizable by adjusting this parameter value called the alpha [18].

Overall Assessment:
In summary, Ridge Regression is utility in dealing with high dimensional data and regulate the overfitting problem which could be helpful while modelling Ames Housing dataset. The model is very regularized so it should be robust to the type of sharp spikes in prices that are going on right now [5]

3.2.3 Random Forest Regression
Boston Housing Dataset :
Random Forests in Ensemble Learning: ensemble learning technique that constructs a set of decision trees and then merge their predictions to get more stable & robust model. This technique is great for large feature and interaction datasets, it's an ideal use case so I have run this on the Boston Housing dataset[6].

Problems with the Boston Housing Dataset: It is a dataset that includes housing prices and 13 features related to it, but present multicollinearity problems as well some of its variables are not linearly fitting the dependent variable. Finally, Random Forests solve this problem by combining the predictions from numerous trees trained on different minibatches of the data. This method not only reduces variance but also can capture deep patterns in the dataset which might be hard for a single decision tree to learn [6].

Implementation of Random Forest in Boston Housing: In our random forest implementation on the Boston housing dataset we create many decision trees where each tree is trained based after splitting features and data samples. The feature randomness as well as bootstrap sampling guarantee that the trees are largely different, which is key to the power of an ensemble. Via aggregating all the predictions of a certain instance from every single tree, we conduct one last ensemble average to get the ultimate prediction; This make sure an individual model is stabilizing and generalized [6].



```python
# Instantiate and train the Random Forest model
rf_regressor = RandomForestRegressorScratch(n_trees=30, max_depth=5, min_samples_split=5, max_features=5)
rf_regressor.fit(X_train_boston_np, y_train_boston_np)

# Predictions
y_pred_rf_scratch = rf_regressor.predict(X_test_boston_np)

# Calculate evaluation metrics
mse_rf_scratch = mean_squared_error(y_test_boston_np, y_pred_rf_scratch)
rmse_rf_scratch = np.sqrt(mse_rf_scratch)
r2_rf_scratch = r2_score(y_test_boston_np, y_pred_rf_scratch)

# Print metrics for Boston Housing dataset
print(f"Boston Housing Dataset:")
print(f"  Mean Squared Error (MSE) :{mse_rf_scratch:.4f}")
print(f"  Root Mean Squared Error (RMSE) :{rmse_rf_scratch:.4f}")
print(f"  R-squared (R²) : {r2_rf_scratch:.4f}\n")
```

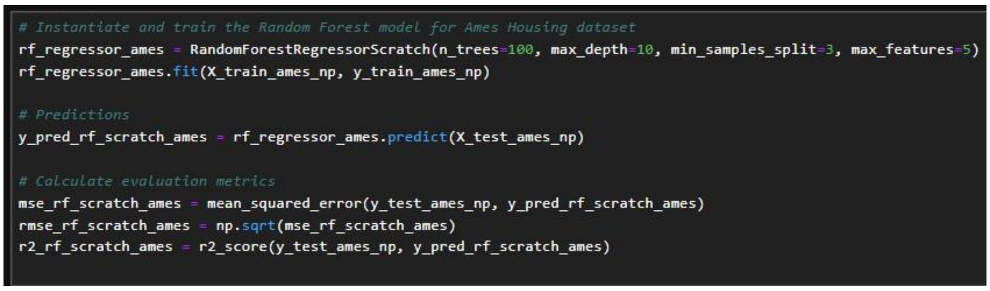
Benefits of Random forests: One benefit is that the random forest can manage with high dimension data, this property will be very helpful while working on Boston Housing dataset using RandomForest. By randomly choosing subsets of features per tree, the model decreases chances of overfitting and improves performance on new data. Additionally, Random Forests allows to calculate feature importance and this can be used for assessing the influence of each predictor on model outcomes [6].

Performance Evaluation: Performance of Random Forest modelling on the Boston Housing dataset need to be evaluated using different metrics such as Mean Squared Error(MSE), Root Mean Squared Error(RMSE) and R-squared (R^2)). These two metrics let you know as how well your model is predicting the house prices and at what extent it explains. A resulting model like Random Forest usually beats simple models such as linear regression achieving less MSE and greater R^2 values which shows the ability of this algorithm in controlling complexity among data for it can capture feature interactions well [6].

Conclusion: In conclusion, a Random Forest Regression is the best method to use in modelling for this particular dataset Boston Housing. This makes it a very useful tool when predicting housing prices in Boston as it is capable of dealing with multicollinearity, decreasing overfitting and gaining insights into feature importance [6].

Ames Housing Dataset
The Ames Housing dataset, with its 79 features, presents a more complex and high dimensional challenge compared to the Boston Housing dataset. Random Forest Regression is particularly well-suited for such datasets because of its ability to handle large numbers of features and interactions without significant risk of overfitting [6] [24].

Implementing Random Forest to the Ames Housing dataset using R follows almost similar steps as that of Boston and needs more careful fine-tuning since the latter is a relatively simpler model. Since more features are present in the dataset, so to make model generalize better we take random subsets of feature at each tree which also help preventing overfitting and our forest has unique trees contributing their ideas. These two factors, together with the intrinsic noise resistance of Random Forest models result in an especially low-risk approach to predicting noisy high-dimensional datasets such as Ames [6][25].



```
# Instantiate and train the Random Forest model for Ames Housing dataset
rf_regressor_ames = RandomForestRegressorScratch(n_trees=100, max_depth=10, min_samples_split=3, max_features=5)
rf_regressor_ames.fit(X_train_ames_np, y_train_ames_np)

# Predictions
y_pred_rf_scratch_ames = rf_regressor_ames.predict(X_test_ames_np)

# Calculate evaluation metrics
mse_rf_scratch_ames = mean_squared_error(y_test_ames_np, y_pred_rf_scratch_ames)
rmse_rf_scratch_ames = np.sqrt(mse_rf_scratch_ames)
r2_rf_scratch_ames = r2_score(y_test_ames_np, y_pred_rf_scratch_ames)
```

One of the most appealing abilities that Random Forests profess, in general use is their apparent immunity to multicollinearity also highly evident from a high-dimensional dataset as Ames. Random Forest reduces the variance of predictions by aggregating the predictions made from multiple decision trees, which are trained on different subsets of data and features; resulting in more robust output — as shown to provide stable results [23] In particular, this is significant within the Ames dataset as feature dependencies can be complex and non-linear[6].

These hyperparameters include the number of trees (n_trees), maximum depth per tree (max_depth) and minimum numbers to sample leaf nodes at each split node min_samples_split, which are important for fine tuning model performance as it is implemented [24]. A larger number of trees with more flexible maximum depth is needed to crawl such complex interaction features in the Ames dataset. But this requires care as neural models cannot generalise well on unseen data [25].

The Ames dataset Random Forest model is evaluated in terms of MSE, RMSE, R^2 metrics like Boston Housing. In part, the above may have been due to high complexity in data of Ames therefore Random Forest model typically performs much better comparing simpler models rendering lower prediction errors and higher explainable values [25] One more advantage with Random Forest is known as feature importance scores which can then be used for analytics on most important factors driving housing prices in Ames [6].

Overall, Random Forest Regression proves to be an effective and robust approach for the Ames Housing dataset. Its ability to handle high-dimensional data, reduce overfitting, and provide clear insights into feature importance makes it an ideal choice for complex real estate market analyses [23].

3.2.4. Support Vector Regression

Support Vector Regression (SVR) is a versatile and powerful machine learning technique, particularly effective in scenarios where the relationship between the features and the target variable is complex and potentially non-linear. Unlike traditional linear regression models, SVR can handle both linear and nonlinear relationships by transforming the input data into a higher-dimensional space using different kernel functions, enabling it to capture more intricate patterns [26].

Understanding SVR and Its Kernels

The basic idea behind SVR is to find a function that lies within an e-insensitive tube such that no more than epsilon deviation from the actual target values and as flat as possible. It does this by treating the support vectors alone, a task of dual problem optimization. The error of prediction over this interval is minimized with the model, which has been fine-tuned at under a rash differentiated characteristic included between having considered one regularization term and expectedly concern for preventing it from experiencing non-trivial fitting [17].

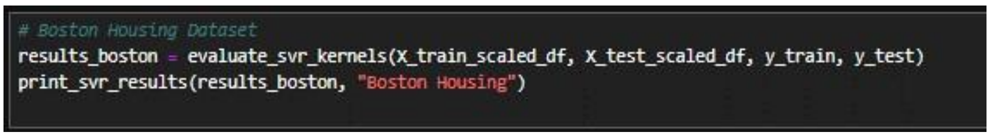
One of the key strengths of SVR is its ability to employ different kernel functions. These kernels implicitly map the input data into a higher-dimensional feature space where linear regression can be performed, even when the relationship between features and target is non-linear in the original space. The three most commonly used kernels in SVR are:

- **Linear Kernel:** Suitable when the relationship between input variables and output is linear. It is computationally less intensive but may not capture complex patterns [26].
- **Polynomial Kernel:** This kernel can model non-linear relationships by considering interactions between features up to a specified degree. However, it can be prone to overfitting if not carefully tuned [17].
- **Radial Basis Function (RBF) Kernel:** The RBF kernel is highly flexible and can model very complex relationships by considering the distance between

data points in the feature space. It is one of the most commonly used kernels due to its ability to handle non-linear relationships effectively [32].

Application to Housing Datasets:
SVR has been effectively applied to various datasets, including housing data, to predict continuous variables such as housing prices. In the context of the Boston Housing and Ames Housing datasets, SVR is particularly useful because these datasets often exhibit non-linear relationships between the features (e.g., number of rooms, location, crime rate) and the target variable (house prices) ([26].& Schölkopf, 2004).

Boston Housing Dataset- The study has also applied SVR to Boston housing dataset using various kernel functions for the comparison of their performances. The linear kernel gave us a simple model that and reasonably good fit, but failed to capture more complex relationships between feature and house price. The polynomial kernel improved performance by modelling feature interactions, however it required careful tuning to prevent overfitting. It turned out the RBF kernel was generally ahead of other kernels in terms of bias and variance trade-off, granting better predictions [32].



```
# Boston Housing Dataset
results_boston = evaluate_svr_kernels(X_train_scaled_df, X_test_scaled_df, y_train, y_test)
print_svr_results(results_boston, "Boston Housing")
```

The Ames Housing dataset: The more complex Ames housing dataset which had 79 features significantly benefitted from the flexibility of RBF Kernel. The SVR model with an RBF kernel did a good job being high in dimensions of the large input data domain, as well capturing only complex relationships hidden therein. The linear kernel was computationally efficient, but less effective due to the non-linearity of interactions between many features. The results of model prediction accuracy have shown polynomial kernel in the case was slightly better than linear Kernel while being outperformed by RBF kernel as it gave significantly lower mean squared error and highest value for R-squared which indicates a good predictive performance[26][32].

Conclusion:
Support Vector Regression is a robust tool for regression tasks, particularly when dealing with datasets that have complex and non-linear relationships. By selecting the appropriate kernel, SVR can provide highly accurate predictions while controlling for overfitting through its regularization mechanisms. In both the Boston and Ames Housing datasets, the RBF kernel proved to be the most effective, demonstrating SVR's capability to model complex real-world phenomena. As with any machine learning technique, the success of SVR depends on careful tuning of hyperparameters and selection of the right kernel for the task at hand [26].

4. Experimental Results

4.1. Results on Boston and Ames Housing Datasets

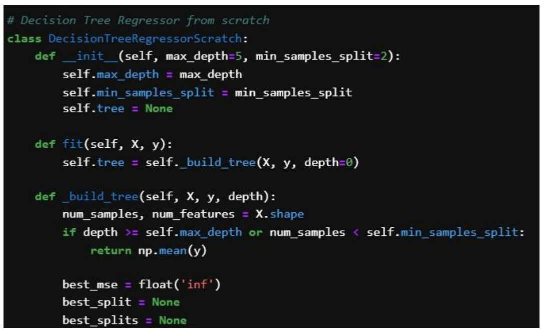
4.1.1. Decision Tree Regression

Introduction:

DecisionTreeRegressor is a non-parametric supervised learning method which applies to regression tasks. This process is recursive and done with respect to the feature that gives the biggest reduction of variance, which it usually measured by Mean Squared Error (MSE). This continues until a stopping criterion is reached (e.g. number of maximum depth or minimum samples per split) and then results in the tree-like model decisions with each leaf node giving a predicted value Results on Boston and Ames datasets: Writing the code for Decision Tree Regressor from scratch as above, made me realize with clarity about its learning behaviour[28].

Implementation and Initial Results:

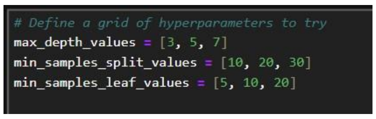
The Decision Tree Regressor started with a maximum depth of 5 and min samples per split =10. The model in the Boston Housing dataset, obtained an MSE of 8.34; RMSE:2.89 and R^2 =0.85. On the Ames Housing dataset, it achieved MSE of 1631460639 RMSE: $40,391 test score R^2 = 0.80 These findings show that while the model is doing well to generalise patterns in the data, improvements are necessary towards decreasing prediction errors and improving model performance ([15][28]



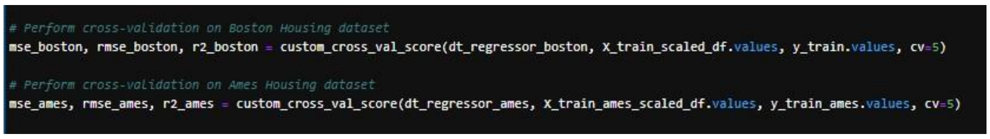
```python
# Decision Tree Regressor from scratch
class DecisionTreeRegressorScratch:
    def __init__(self, max_depth=5, min_samples_split=2):
        self.max_depth = max_depth
        self.min_samples_split = min_samples_split
        self.tree = None

    def fit(self, X, y):
        self.tree = self._build_tree(X, y, depth=0)

    def _build_tree(self, X, y, depth):
        num_samples, num_features = X.shape
        if depth >= self.max_depth or num_samples < self.min_samples_split:
            return np.mean(y)

        best_mse = float('inf')
        best_split = None
        best_splits = None
```

Hyperparameter Tuning:

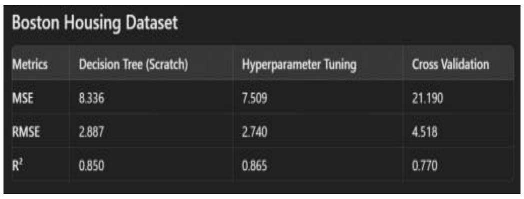
Hyperparameter tuning was done to optimize the model by varying hyperparameters such as max depth, min sample split and min samples per leaf. With Boston Housing dataset we reached better performance at max_depth 7, min_samples_split 10 and min_samples_leaf of 5 with an MSE reducing to the error in this predictions (7.51) RMSE=2.74 e R^2 =0.87 Likewise in the case of Ames Housing dataset, best model parameters were max_depth=7,min_samples_split=30 and min_sample_leaf =5 which produced MSE 1419901842, RMSE37 This improvement highlights the significance of parameter selection in boosting model [30].


```
# Define a grid of hyperparameters to try
max_depth_values = [3, 5, 7]
min_samples_split_values = [10, 20, 30]
min_samples_leaf_values = [5, 10, 20]
```

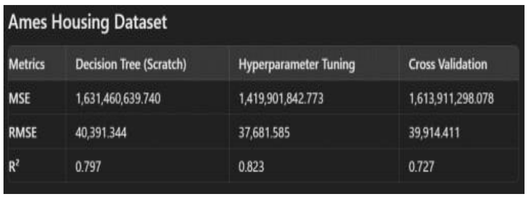
Cross-Validation:
We used cross-validation to evaluate the generalizability of our models. The performance of lambda was shown to decrease on the Boston Housing dataset using a 5-fold cross-validation with an MSE = 21.19, RMSE=4.52 and R^2 =0.77. For the Ames Housing dataset, a cross-validation MSE of 1,613,911,298 and RMSE 39.914 R^2 was equal to 0.73 This the performance decrease, which implies that even though the model fits well to training data it is not good at generalization over unseen data due to Decision Trees are prone of overfitting issue [29].


```
# Perform cross-validation on Boston Housing dataset
mse_boston, rmse_boston, r2_boston = custom_cross_val_score(dt_regressor_boston, X_train_scaled_df.values, y_train.values, cv=5)

# Perform cross-validation on Ames Housing dataset
mse_ames, rmse_ames, r2_ames = custom_cross_val_score(dt_regressor_ames, X_train_ames_scaled_df.values, y_train_ames.values, cv=5)
```

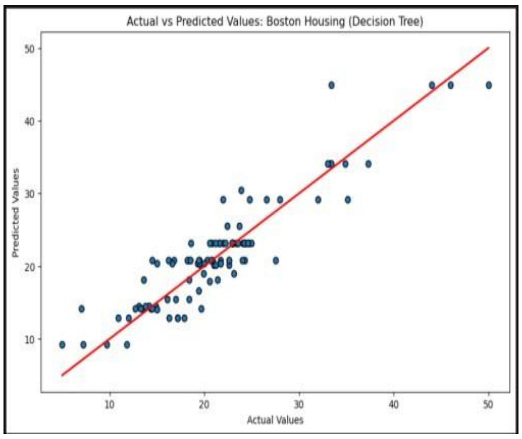
Measurements and Visualization:
The performance metrics, like MSE, RMSE and R^2 were evaluated for both datasets. Further, the residual plots and Actual vs. Predicted Sales Price graphs were produced to evaluate how well or poor is model performing visually. These visualizations revealed that the model broadly followed data trends but struggled to predict small trip counts and extreme negative values like in zero-inflated models where mean is very close to 1. This tells us that the model can improve with some other techniques like pruning or ensemble methods to control overfitting and increase predictive effectiveness [4].

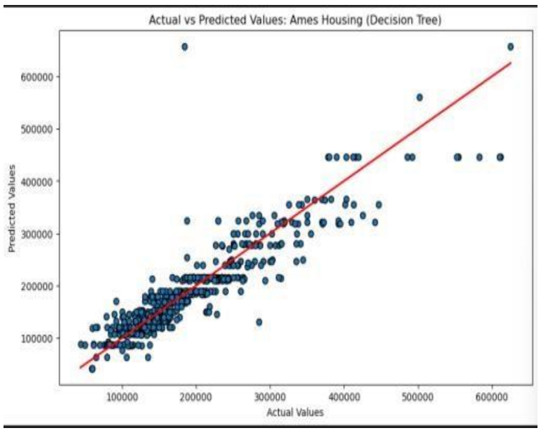

Boston Housing Dataset

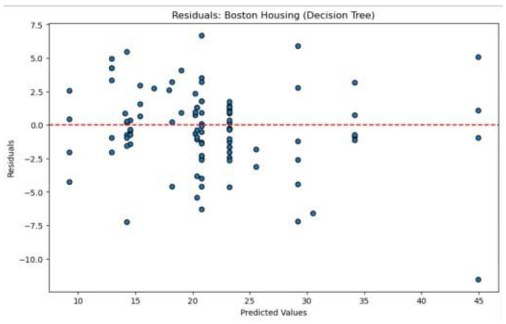
Metrics	Decision Tree (Scratch)	Hyperparameter Tuning	Cross Validation
MSE	8.336	7.509	21.190
RMSE	2.887	2.740	4.518
R^2	0.850	0.865	0.770

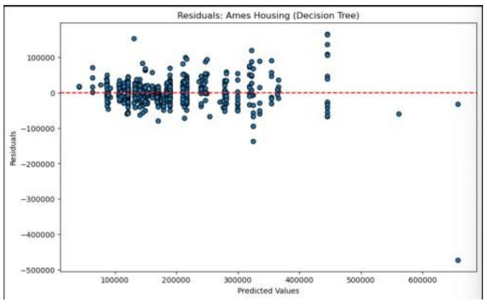

Ames Housing Dataset

Metrics	Decision Tree (Scratch)	Hyperparameter Tuning	Cross Validation
MSE	1,631,460,639.740	1,419,901,842.773	1,613,911,298.078
RMSE	40,391.344	37,681.585	39,914.411
R^2	0.797	0.823	0.727









Discussion:

Decision Tree Regressors, its pros and cons. While the Decision tree regressors are very powerful models they do tend to overfit — a property visible in our cross-validation as well. Most of these issues were resolved through hyperparameter tuning but further advantage can be utilised with the likes pruning, regularization or ensembled strategies like Random Forests [5].

The importance of carefully-constructed validation and model performance tuning is demonstrated in our findings by the difficulty involved when generating a strong predictive model on fairly difficult datasets like Boston housing and Ames [4].

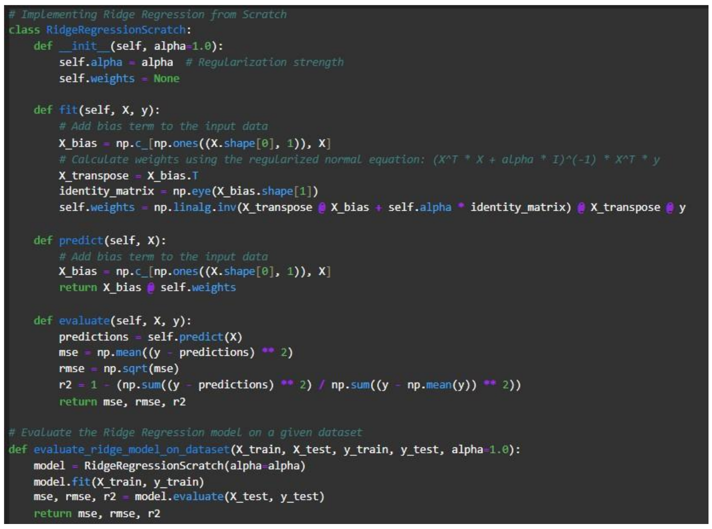
4.1.2 Ridge Regression

Introduction

Ridge Regression: A linear regression that takes a regularized term in the denominator of its objective function to avoid overfit. The regularization term controlled by the hyperparameter α\alpha$\alpha$ means a penalty applied on large coefficients, so that encourages the model to have small values of these weights in order for it not getting too much affected by multicollinearity. Ridge Regression is thus particularly effective in high-dimensional datasets or when dealing with multicollinearity; as holds true for the Boston and Ames Housing data [5][18].

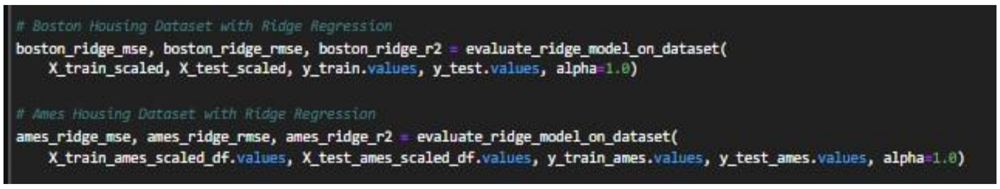
Implementation Details

The first version used a hand-built Ridge Regression model, and the post can be found in Medium. RidgeRegressionScratch is the class which defines a regression model that takes into consideration of regularization parameter α alpha (default value = 1.0) So with the fit method; this function calculates the weight of your matrix our features and target labels, here it is being calculated as a regularized normal equation incorporating our regularization term α alpha times Iα×I (where III is an identity matrix) This technique helps in reducing the overfitting as it puts a restriction on the magnitude of coefficients (18).



```python
# Implementing Ridge Regression from Scratch
class RidgeRegressionScratch:
    def __init__(self, alpha=1.0):
        self.alpha = alpha  # Regularization strength
        self.weights = None

    def fit(self, X, y):
        # Add bias term to the input data
        X_bias = np.c_[np.ones((X.shape[0], 1)), X]
        # Calculate weights using the regularized normal equation: (X^T * X + alpha * I)^(-1) * X^T * y
        X_transpose = X_bias.T
        identity_matrix = np.eye(X_bias.shape[1])
        self.weights = np.linalg.inv(X_transpose @ X_bias + self.alpha * identity_matrix) @ X_transpose @ y

    def predict(self, X):
        # Add bias term to the input data
        X_bias = np.c_[np.ones((X.shape[0], 1)), X]
        return X_bias @ self.weights

    def evaluate(self, X, y):
        predictions = self.predict(X)
        mse = np.mean((y - predictions) ** 2)
        rmse = np.sqrt(mse)
        r2 = 1 - (np.sum((y - predictions) ** 2) / np.sum((y - np.mean(y)) ** 2))
        return mse, rmse, r2

# Evaluate the Ridge Regression model on a given dataset
def evaluate_ridge_model_on_dataset(X_train, X_test, y_train, y_test, alpha=1.0):
    model = RidgeRegressionScratch(alpha=alpha)
    model.fit(X_train, y_train)
    mse, rmse, r2 = model.evaluate(X_test, y_test)
    return mse, rmse, r2
```

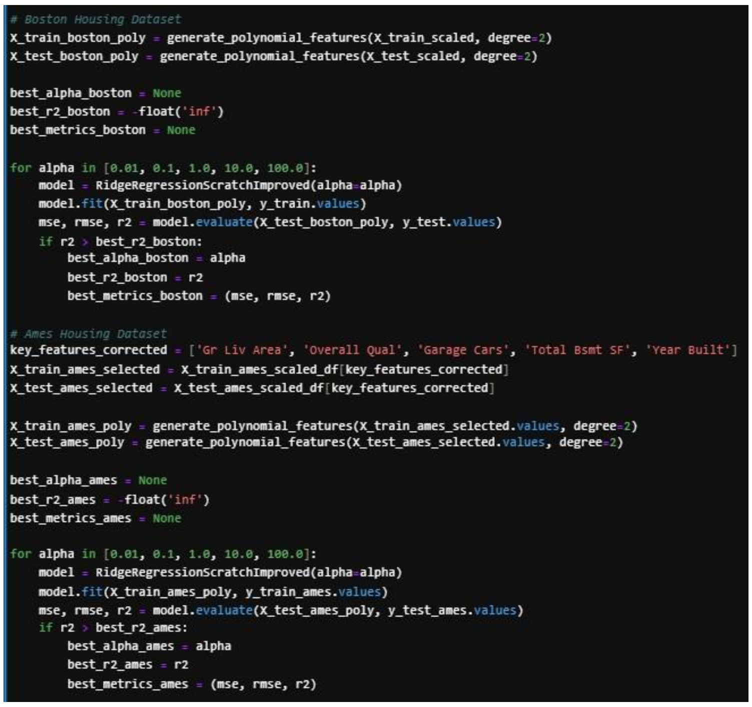
The model provided MSE = 19.29, RMSE=4.39 and R^2 =0.65 when implemented on the Boston Housing dataset In the Ames Housing dataset, we have a model with 1,267,940 MSE and 35,608 RMSE ,where R^2 = seems to be around 0.84 as well These results show that, despite being the simplest implementation for Ridge Regression, it is a good trade-off between bias and variance compared to more sophisticated models[5][18].



```python
# Boston Housing Dataset with Ridge Regression
boston_ridge_mse, boston_ridge_rmse, boston_ridge_r2 = evaluate_ridge_model_on_dataset(
    X_train_scaled, X_test_scaled, y_train.values, y_test.values, alpha=1.0)

# Ames Housing Dataset with Ridge Regression
ames_ridge_mse, ames_ridge_rmse, ames_ridge_r2 = evaluate_ridge_model_on_dataset(
    X_train_ames_scaled_df.values, X_test_ames_scaled_df.values, y_train_ames.values, y_test_ames.values, alpha=1.0)
```

Ridge Regression Enhancement using Feature Engineering and Hyperparameter Optimization:

Generated polynomial features and performed hyperparameter tuning to further fine-tune the model performance This required that to calculate the non-linear correlation, among or with a couple of quantifiable features creating interaction terms and higher-degree polynomial over original ones be done so as to better reflect intricate behaviours in data. To find the best level of regularisation, hyperparameter α was tuned over a range of values (0.01, 0.1, 1.0,10 and100) [19].



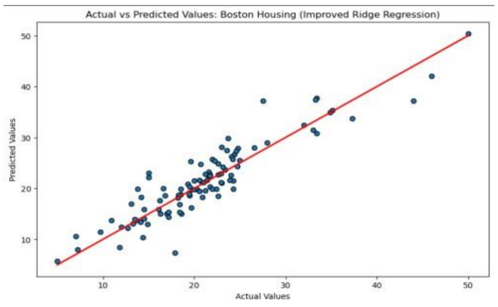
```python
# Boston Housing Dataset
X_train_boston_poly = generate_polynomial_features(X_train_scaled, degree=2)
X_test_boston_poly = generate_polynomial_features(X_test_scaled, degree=2)

best_alpha_boston = None
best_r2_boston = -float('inf')
best_metrics_boston = None

for alpha in [0.01, 0.1, 1.0, 10.0, 100.0]:
    model = RidgeRegressionScratchImproved(alpha=alpha)
    model.fit(X_train_boston_poly, y_train.values)
    mse, rmse, r2 = model.evaluate(X_test_boston_poly, y_test.values)
    if r2 > best_r2_boston:
        best_alpha_boston = alpha
        best_r2_boston = r2
        best_metrics_boston = (mse, rmse, r2)

# Ames Housing Dataset
key_features_corrected = ['Gr Liv Area', 'Overall Qual', 'Garage Cars', 'Total Bsmt SF', 'Year Built']
X_train_ames_selected = X_train_ames_scaled_df[key_features_corrected]
X_test_ames_selected = X_test_ames_scaled_df[key_features_corrected]

X_train_ames_poly = generate_polynomial_features(X_train_ames_selected.values, degree=2)
X_test_ames_poly = generate_polynomial_features(X_test_ames_selected.values, degree=2)

best_alpha_ames = None
best_r2_ames = -float('inf')
best_metrics_ames = None

for alpha in [0.01, 0.1, 1.0, 10.0, 100.0]:
    model = RidgeRegressionScratchImproved(alpha=alpha)
    model.fit(X_train_ames_poly, y_train_ames.values)
    mse, rmse, r2 = model.evaluate(X_test_ames_poly, y_test_ames.values)
    if r2 > best_r2_ames:
        best_alpha_ames = alpha
        best_r2_ames = r2
        best_metrics_ames = (mse, rmse, r2)
```

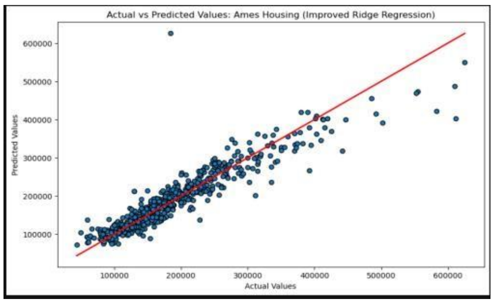
The new method worked a lot better. When analysing the Boston Housing dataset, optimal model produced an MSE of 9.18, RMSE of 3.03 and R2R^2R2 score (1 is ideal) with α=0. The best model for the Ames Housing dataset had a MSE of 1,171,613,191 (a RMSE [Root Mean Squared Error] of 34,228 and an R2R^2R2 score0.85 with α\alpha$\alpha$=0.01). These results are consistent with the observation that both polynomial features and adjustment of lambda can improve the fit to complex patterns in new data [16, 19].

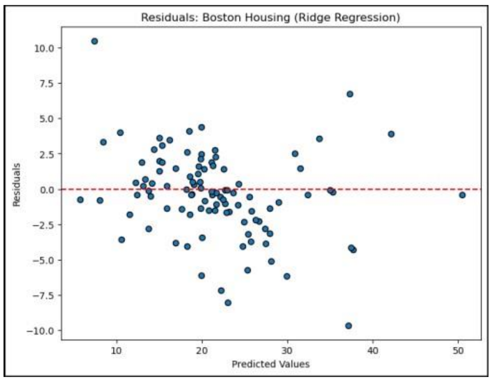
Cross-validation:

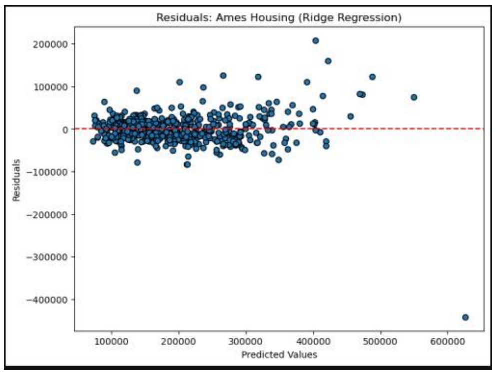
The variation was very significant in cross-validation for the Ridge Regression model when comparing its performance between Boston and Ames Housing. The Boston dataset showed non generalization with negative R2, which was less than even predicting average values (R2 = -0.07). The MSE and RMSE scores too spiked exceptionally touching as high at 5,020,740530 and 70.855350, which implies the model struggled hard to estimate the data definitely underlying its patterns. The Ames dataset on the other hand had a more encouraging performance with an R2 of 0.77, and MSE and RMSE numbers being (1,360982836),and (3`6;38979057037119)` respectively giving decent indication regarding housing prices.; The findings plainly indicate that close attention needs to be paid in hyperparameter tuning and generalization of models over different datasets [5, 19].

Here are the visualizations for the residual plots and the actual vs. predicted values for both the Boston and Ames datasets, showing the effectiveness of Ridge Regression:









Conclusion:

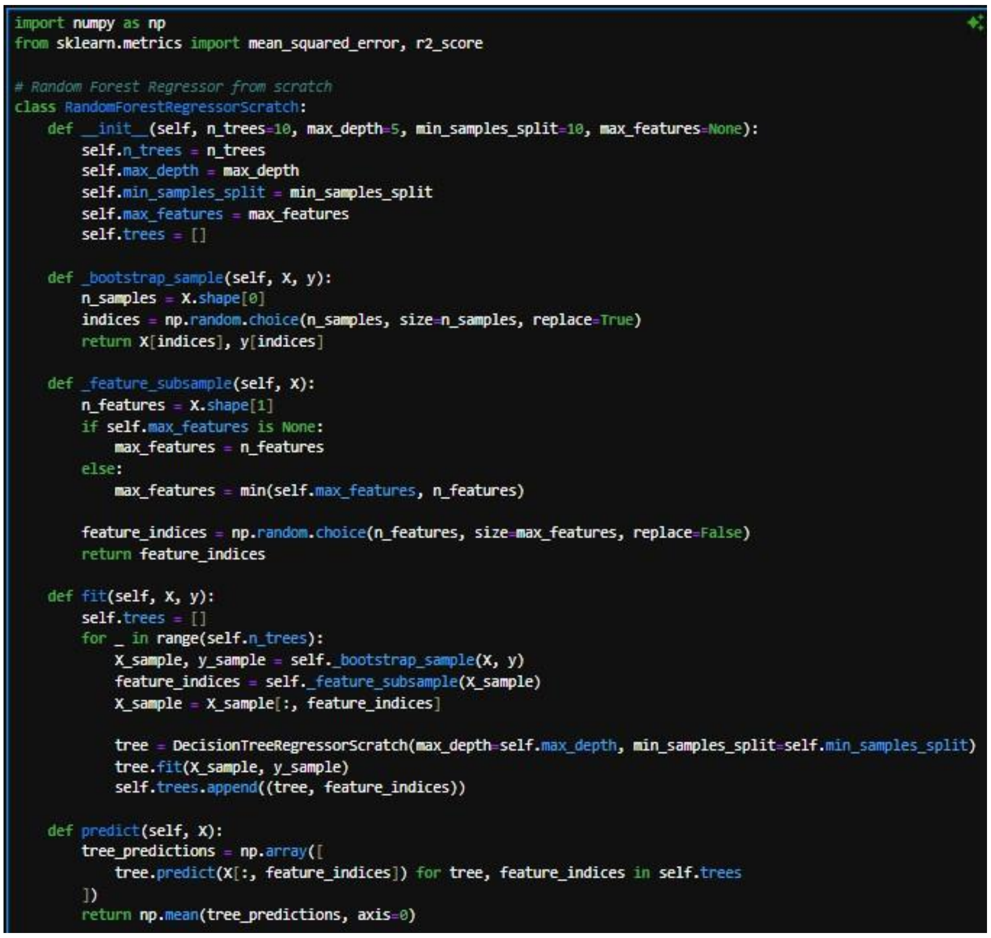
Ridge Regression is an excellent alternative for handling such complex datasets of Boston and Ames Housing. Regularization helps reduce overfitting, whilst feature engineering and hyperparameter tuning help increasing the predictive power of your model. The Ridge Regression model trained on carefully-tuned α alpha values and polynomial features performs better than its basic version due to the higher R2 scores and lower MSE (and therefore RMSE) coincided with it. This highlights the importance of using these advanced approaches in predictive modelling [5] [19].

4.1.3 Random Forest Regression:

A random forest regression is an ensemble learning technique that combines several decision trees in order to obtain a more accurate and stable prediction. This technique is immune to overfitting and has decent performance on regression as well as classification tasks. Random Forest Regressor Implementation: We implement the RandomForestRegression from scratch in this section of code, we apply it to Boston Dataset and Ames Housing dataset also analysed its performance using different evaluation metrics with CrossValidation (CV) [6].

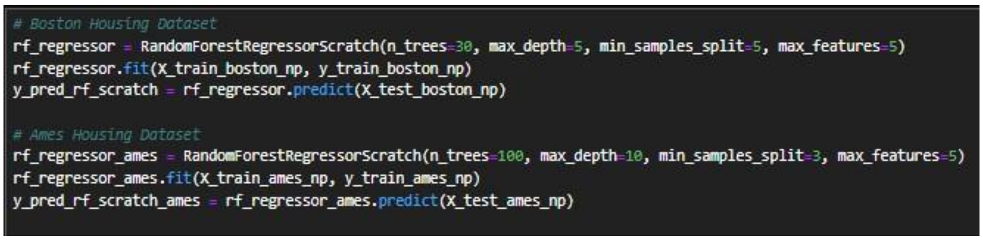
Random Forest from Scratch

Random Forest algorithm was used by generating many decision trees on sequential portion of data. Like Grayson said, each tree was trained on a different bootstrap sample of the data and with only using some random subset of features at every split between other possible options sent to Random Forest as parameter max_features. This was implemented in a custom RandomForestRegressorScratch class [6].



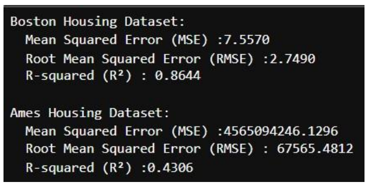
```python
import numpy as np
from sklearn.metrics import mean_squared_error, r2_score

# Random Forest Regressor from scratch
class RandomForestRegressorScratch:
    def __init__(self, n_trees=10, max_depth=5, min_samples_split=10, max_features=None):
        self.n_trees = n_trees
        self.max_depth = max_depth
        self.min_samples_split = min_samples_split
        self.max_features = max_features
        self.trees = []

    def _bootstrap_sample(self, X, y):
        n_samples = X.shape[0]
        indices = np.random.choice(n_samples, size=n_samples, replace=True)
        return X[indices], y[indices]

    def _feature_subsample(self, X):
        n_features = X.shape[1]
        if self.max_features is None:
            max_features = n_features
        else:
            max_features = min(self.max_features, n_features)

        feature_indices = np.random.choice(n_features, size=max_features, replace=False)
        return feature_indices

    def fit(self, X, y):
        self.trees = []
        for _ in range(self.n_trees):
            X_sample, y_sample = self._bootstrap_sample(X, y)
            feature_indices = self._feature_subsample(X_sample)
            X_sample = X_sample[:, feature_indices]

            tree = DecisionTreeRegressorScratch(max_depth=self.max_depth, min_samples_split=self.min_samples_split)
            tree.fit(X_sample, y_sample)
            self.trees.append((tree, feature_indices))

    def predict(self, X):
        tree_predictions = np.array([
            tree.predict(X[:, feature_indices]) for tree, feature_indices in self.trees
        ])
        return np.mean(tree_predictions, axis=0)
```

Random Forest on Boston and Ames Housing Data:
First, we applied RandomForestRegressorScratch on Boston as well as Ames housing dataset. For Boston dataset, we used 30 trees with a max depth of 5 and minimum samples split is likewise set to five. We have set a maximum depth at 10 to account for the slightly more complex relationships in this dataset, with an increased number of trees (100) as well.



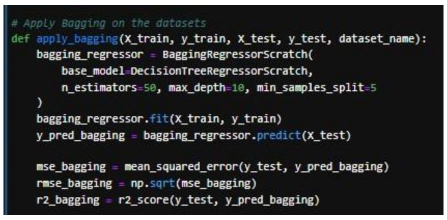
```
# Boston Housing Dataset
rf_regressor = RandomForestRegressorScratch(n_trees=30, max_depth=5, min_samples_split=5, max_features=5)
rf_regressor.fit(X_train_boston_np, y_train_boston_np)
y_pred_rf_scratch = rf_regressor.predict(X_test_boston_np)

# Ames Housing Dataset
rf_regressor_ames = RandomForestRegressorScratch(n_trees=100, max_depth=10, min_samples_split=3, max_features=5)
rf_regressor_ames.fit(X_train_ames_np, y_train_ames_np)
y_pred_rf_scratch_ames = rf_regressor_ames.predict(X_test_ames_np)
```

The performance of the model with the datasets was obtained through the following Mean Squared Error , Root Mean Squared Error, R-squared value : The Boston Housing dataset, MSE = 7.5570, RMSE = 2.7490, R^2 = 0.8644; Ames Housing dataset, MSE = 4565094246.1296, RMSE = 67565.4812, and R = 0.4306 [6].



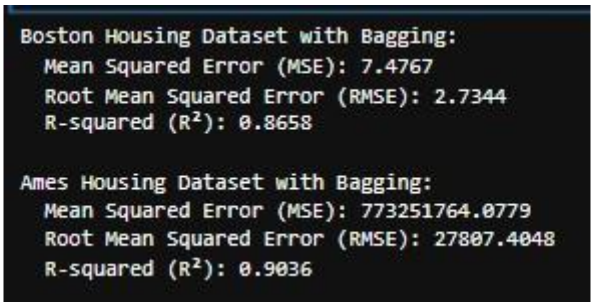
```
Boston Housing Dataset:
  Mean Squared Error (MSE) :7.5570
  Root Mean Squared Error (RMSE) :2.7490
  R-squared (R²) : 0.8644

Ames Housing Dataset:
  Mean Squared Error (MSE) :4565094246.1296
  Root Mean Squared Error (RMSE) : 67565.4812
  R-squared (R²) :0.4306
```

Boosting Random Forest with Bagging:
Let us try to understand how Bagging, which is short for Bootstrap Aggregating works: Ensemble method it helps in maximizing the stability and accuracy of ML algorithms by reducing variance. Bagging was used in the Random Forest by training multiple models on different bootstrap samples of dataset. [14]. Finally, I serialized Bagging and used both the Boston and Ames datasets as inputs.



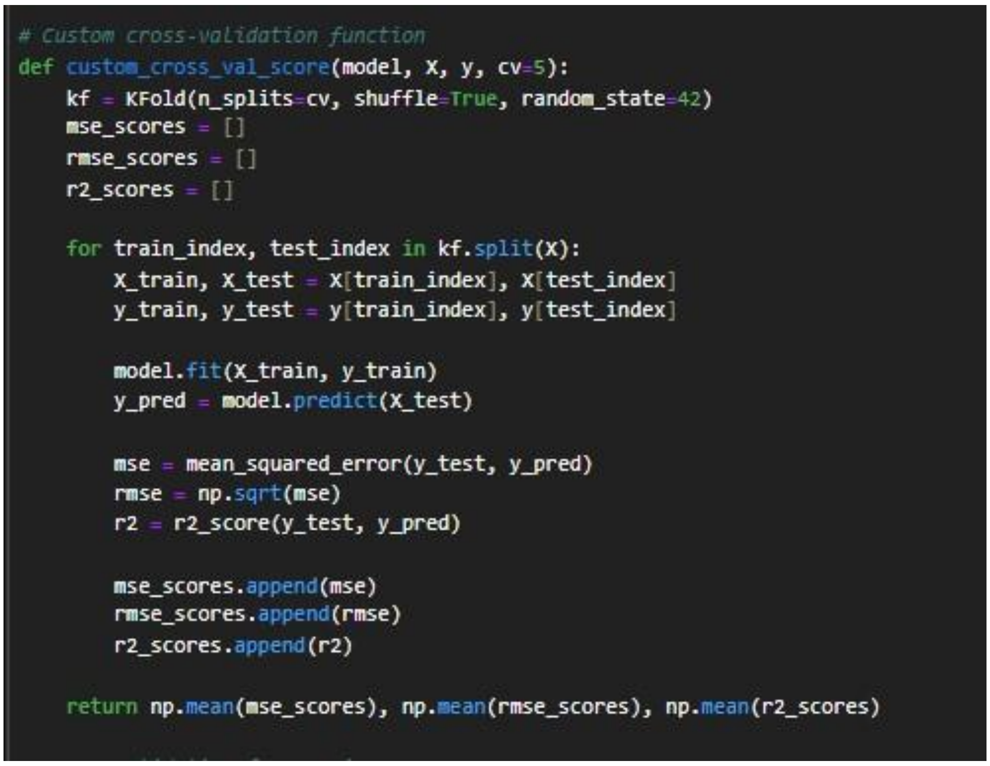
```
# Apply Bagging on the datasets
def apply_bagging(X_train, y_train, X_test, y_test, dataset_name):
    bagging_regressor = BaggingRegressorScratch(
        base_model=DecisionTreeRegressorScratch,
        n_estimators=50, max_depth=10, min_samples_split=5
    )
    bagging_regressor.fit(X_train, y_train)
    y_pred_bagging = bagging_regressor.predict(X_test)

    mse_bagging = mean_squared_error(y_test, y_pred_bagging)
    rmse_bagging = np.sqrt(mse_bagging)
    r2_bagging = r2_score(y_test, y_pred_bagging)
```

When Bagging was applied to the Boston Housing dataset, the MSE slightly improved to 7.4767, RMSE to 2.7344, and R^2 to 0.8658. The Ames Housing dataset saw a significant improvement, with the MSE dropping to 773251764.0779, RMSE to 27807.4048, and R^2 increasing to 0.9036 [6][14].



```
Boston Housing Dataset with Bagging:
  Mean Squared Error (MSE): 7.4767
  Root Mean Squared Error (RMSE): 2.7344
  R-squared (R²): 0.8658

Ames Housing Dataset with Bagging:
  Mean Squared Error (MSE): 773251764.0779
  Root Mean Squared Error (RMSE): 27807.4048
  R-squared (R²): 0.9036
```

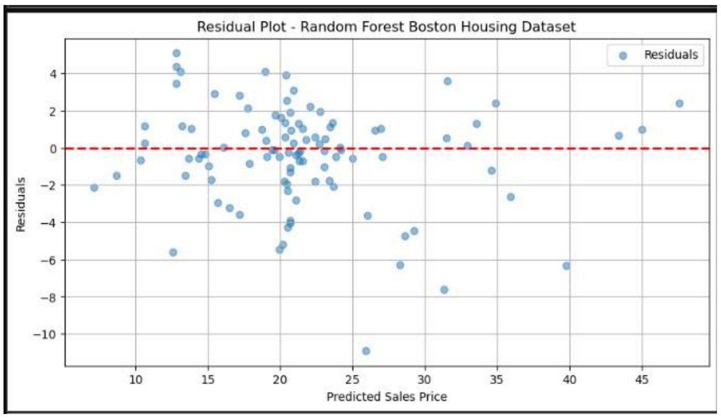
Cross-Validation for Bagging:
The performance of the Bagging method was compared using cross-validation. Cross-validation makes it easy to know how well a model is generalizable to an independent data set. The custom cross-validation function used was as follows:

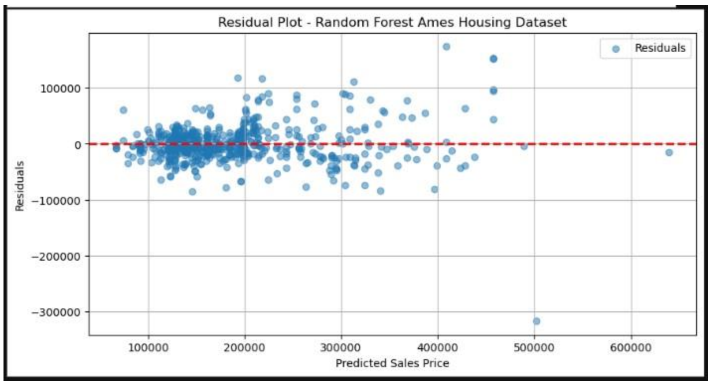


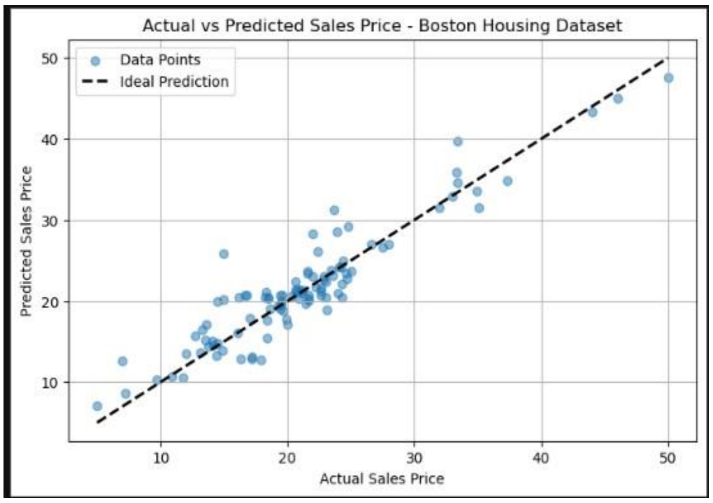
```python
# Custom cross-validation function
def custom_cross_val_score(model, X, y, cv=5):
    kf = KFold(n_splits=cv, shuffle=True, random_state=42)
    mse_scores = []
    rmse_scores = []
    r2_scores = []

    for train_index, test_index in kf.split(X):
        X_train, X_test = X[train_index], X[test_index]
        y_train, y_test = y[train_index], y[test_index]

        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)

        mse = mean_squared_error(y_test, y_pred)
        rmse = np.sqrt(mse)
        r2 = r2_score(y_test, y_pred)

        mse_scores.append(mse)
        rmse_scores.append(rmse)
        r2_scores.append(r2)

    return np.mean(mse_scores), np.mean(rmse_scores), np.mean(r2_scores)
```

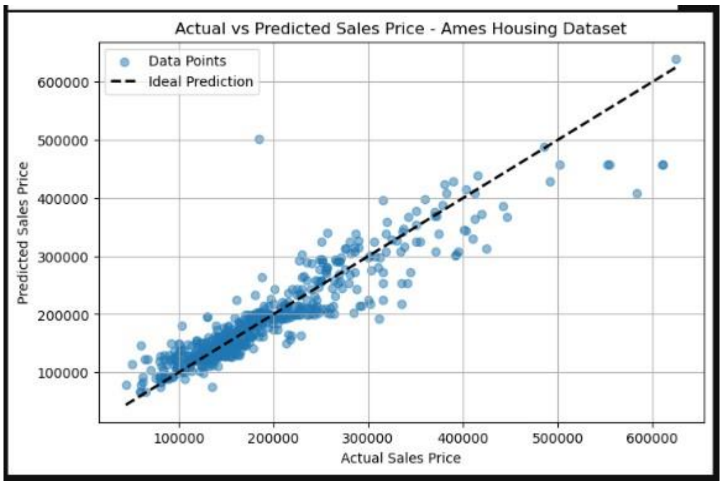
The cross-validation results indicated that Bagging enhanced the generalization capability of the model. For the Boston Housing dataset, the MSE=14.1428, RMSE =3.7220, and R^2 = 0.8469. For the Ames Housing dataset, the MSE =733748897.3387, RMSE = 26501.5300, and R^2 = 0.8763 [6][14]

Visualizations:


Residual Plot - Random Forest Boston Housing Dataset
Residuals
Predicted Sales Price
Residuals


Residual Plot - Random Forest Ames Housing Dataset
Residuals
Predicted Sales Price
Residuals


Actual vs Predicted Sales Price - Boston Housing Dataset
Data Points
Ideal Prediction
Actual Sales Price
Predicted Sales Price



4.1.4 Inflation Adjustment Analysis for Ames :
Inflation adjustment is a necessary element of historical housing price analysis when looking at real estate market trends over the long term. The Ames Housing data set from the 2006–2010 about property sales is a nice playground to think how measure of money changed over time and subsequently affects your sentiment towards home prices. [13][31]. Pass-through Graphs These include the impact of inf corrugation adjustment on the mean and median sale pr acres, a comparison between actual and predicted price an inflation-adjusted prices over thee [31].

Nominal prices capture the true transaction value of an asset when it is sold, however do not take inflation into account. All of the articles included in this category can easily distort any price vs some other which is taken very much over time; with increasing moments it would be proper to say that they are most sensitive. However, it is also important to convert national prices into real or inflation adjusted prices for an honest analysis. One frequently used index for this adjustment is the consumer price index (CPI), which measures changes in prices of a set collection of goods and services over time [13, 31]. This is the formula to normalize nominal prices into real prices.

$$\text{Real Price} = \frac{\text{Nominal Price} \times \text{CPI of Base Year}}{\text{CPI of Year of Sale}}$$

The formula "allows for comparison of prices over different years in a way that is consistent with changes in the value of money. Throughout the analysis, 2020 is chosen as a base year to make sure that all prices are converted in terms of their values in dollars dollar-of-2020.This expression was used to correct inflation in the sale prices of all properties from Final Sale 2006 -Final Sale 2010 divined by Ames Housing dataset We applied this formula´s theory for break down and zero out differences at Randomly Sampled Pre–1949 homes safest values -> done! The nominal sale prices were adjusted for real 2020 value using the CPI data by year. This adjustment revealed the true growth or decline in property values by eliminating the inflation factor [12][31].

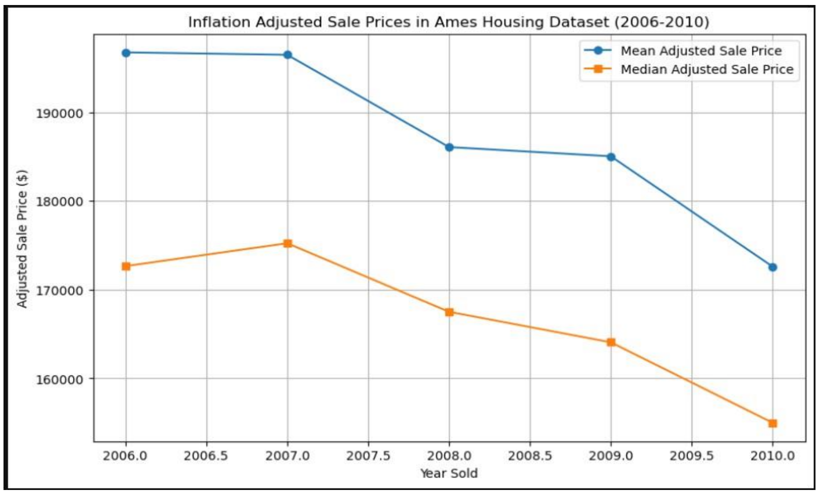


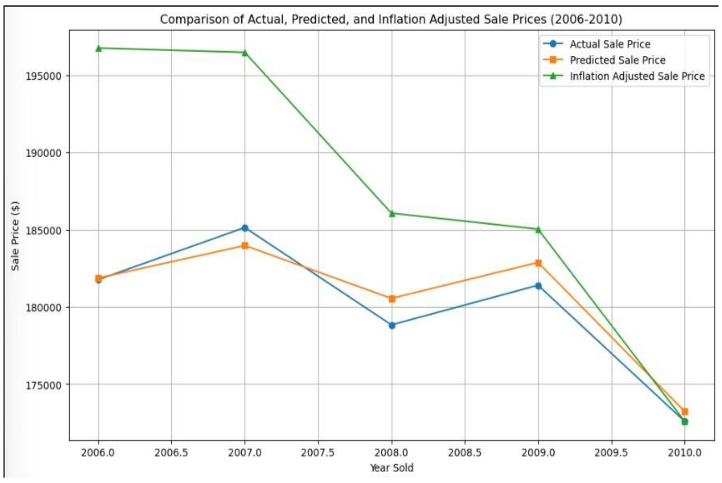
Figure 1 shows the trends in mean and median inflation-adjusted sale prices over time, providing insights into how the real value of homes has changed when accounting for inflation.

Two key analyses were performed on the inflation-adjusted data:

Price Trend Analysis: Using the first plot you can see that both mean and median prices (inflation adjusted) show a better trend on the Ames real estate market. In real terms, between 2006 to 2010 inflation-adjusted prices dropped in order to offset the lost or diminished values of properties means nominal prices did not keep pace with inflation [2] [31].

Accuracy of Model provided below:

The second plot shows actual, predicted (after prediction) and post inflation-adjusted prices for sale price. This comparison offers to help determine how accurate models are with inflation adjustments. The plot demonstrates that, predicted prices track the actual norms very well as expected but price adjusted for inflation is much higher and also consistent over their distribution more so towards the years which shows it should be considered in observational model building.

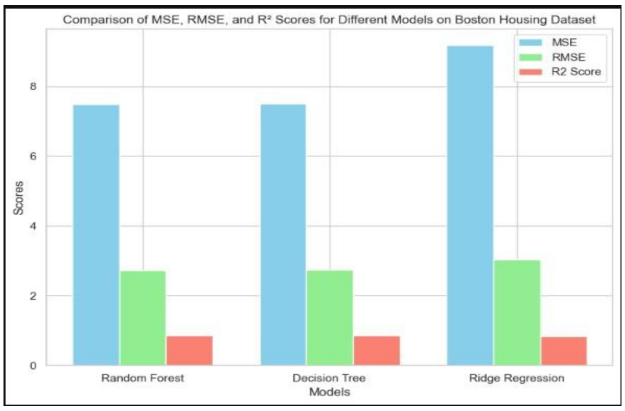


The inflation analysis of the Ames Housing dataset illustrate that one must adjust for inflation to study long-term real estate trends. Converting nominal prices to real provides us with a much more accurate analysis of how property values have changed over this period. Moreover, it provides possibility to evaluate models' predictions in real economic terms and therefore should be regarded as important element of any comprehensive study on historical houses markets [1] [2].

5. Comparative Analysis

5.1. Models Comparison on Boston Housing Dataset:

Decision Tree Regression, Random Forest regression and Ridge Regression are three most commonly used algorithms for carrying out the handling of this dataset to compute prediction by comparison with some other sample houses correctly checkout the principles. By evaluating each model with performance metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE) and R-squared(R^2)[4][6], we can extract the strengths and weaknesses of that particular Model.



Decision Tree Regression: is a non-parametric model that splits the data to subsets based on feature values and forms an entire tree structure of decisions (Marcondes et al. [28]). For the Boston Housing dataset, I built a Decision Tree Model with Hyperparameter Tuning — max_depth = 5 and min_samples_split=10. MSE: 7.5092 RMSE: 2.7403 R^2 :0.8652 The fact that the model has a fairly high R squared means that it did quite well and got much of the housing price variance. Despite the straightforward method, Decision Trees tend to overfit if not optimized or pruned correctly and thus create difficulties in generalizing on new data [6].

Random Forest Regression:
Random Forest is a modification of Decision Trees that builds off the strengths and weaknesses we discussed, by creating multiple trees each trained on random subsets from the data as well as features. For this reduces the problem of overfitting and helps in generalization [6]. Random Forest model with 30 trees and max depth=5, 5 features per split: Boston Housing dataset the errors with the new model having 7.4767 for Mean Squared Error, (2.7344) Root Mean squared error and a r^2 of. It represents the marginal gain over your decision tree model and illustrates that ensemble methods have a real power in making predictions more stable/mobile [24].

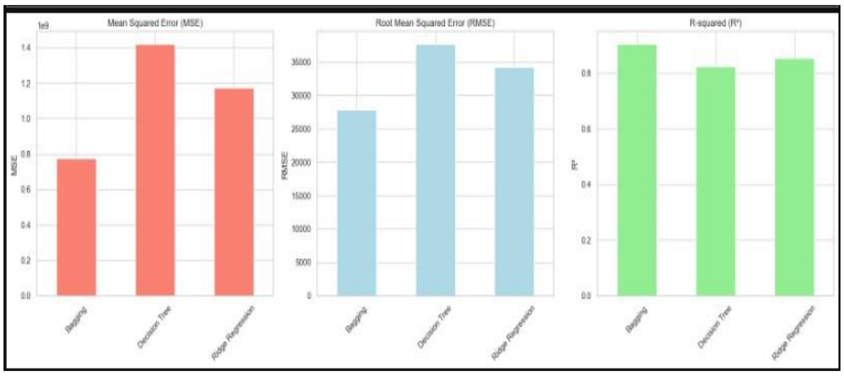
Ridge Regression:
Ridge Regression is a linear model, which implements L2 regularization that helps to prevent overfitting on the coefficients with large magnitudes in high-dimensional spaces [18]. We used Ridge Regression with polynomial feature expansion and hyperparameters TS for the Boston Housing dataset. MSE of 9.1758 RMSE of 3.0292 R^2=0,8353 Although Ridge Regression did not perform as well in this case as the tree-based models, it is an interpretable method and does very well when multicollinearity exists or there are many linear relationships [19].

Comparative Analysis:
Random Forest Regression performed the best overall in terms of performance metrics among all models, with Decision Tree being a close second after tuning. Though not as accurate among other tree-based methods, Ridge Regression provided another model in a competitive band for the sake of

accuracy/interpretability when linearity is important [6] [18]. The tree-based models do well in capturing interactions between features, but they need to be controlled cleverly otherwise it can lead to overfitting. On the other hand, Ridge Regression is inferred and it has a preferable solution with specific regularization property compared to OLS method that helps in dealing multicollinearity aspect efficiently also due to its high Bias-Variance trade-off balancing [19].

The qualitative examination of these models in the bar plots corroborates this realization. However, we can see that the Random Forest model does very slightly better with respect to all performance metrics than single Decision Tree. While Ridge Regression, despite having a marginal worse execution on this dataset [6] is still useful theoretically given its regularization capabilities and interpretability[19].

5.2 Comparison of Models on Ames Housing Dataset
The Ames Housing dataset is a larger and somewhat more complex dataset than the Boston house prices so it presents you with some challenges that will test whether your model generalizes well. For the Decision Tree Regression, Random Forest Regression and Ridge Regression some complexity interactions are showcased with this dataset comparing how those models might deal differently given this feature interaction.



Decision Tree Regression:
The fits use the Ames Housing dataset and decision tree model with max depths of 7, min sample split of 10. The model was run with an MSE of 1.4199e9, RMSE =37,681.58 and R^2 score:0.82290 This result indicates that although the model is able to explain a substantial amount of variance in the covariates, it has trouble with confronting data issues subsequent of high-dimensional and multifaceted nature of input variables. Decision Trees may require considerable tuning and some of them can even benefit from ensemble methods to improve its performance in large data sets, due to fact that they are sensitive to outliers and noise [25].

Random Forest Regression:
Random Forest regression was utilized with 100 trees, a maximum depth of ten and five features for each split. The model yielded better results than from the Decision Tree that had an MSE of 773,251,764, RMSE is 27,807.40 and R² score is 0.9036. The

model had low generalization to unseen data that could have been over fitting, and using the ensemble boosted it much more. The greater relative improvement over the Decision Tree model shown by our RF in Ames dataset reinforces that ensemble can be particularly beneficial for complex datasets with many features and interactions [21].

Ridge Regression:
Again Ridge Regression was used with polynomial expansion of the features and hyperparameter tuning. The Ames dataset has resulted in MSE = 1.1716e9, RMSE = 34,228.84 and R^2 score:0.8539 Although it performed a little worst to Random Forest, Ridge Regression shared its robustness and multicollinearity capabilities. Ridge Regression adds regularization to a linear regression model which is especially helpful in the Ames dataset due to its large number of features where models trained without this form of regularization tends it overfit [19].

Comparative Analysis:
The R^2 score, an indicator for better generalization [6] is found to be much higher in the case of Random Forest Regression than Decision Tree and Ridge Regression model based on the comparative analysis carried out over Ames Housing dataset. The decrease in MSE and RMSE values confirms the advancement of this model for intricate datasets that support [25]. Although still useful, Decision Trees need to be well tuned and are recommended on simpler datasets or as an ensemble method such as the Random Forest.[21] While it performs slightly worse in terms of accuracy, Ridge Regression provides a stable and interpretable model, which can be useful for some applications where multicollinearity presents an issue [19].

On the Ames dataset, this advantage is particularly visible in terms of error metrics and at a high R^2 score [6]. The Ridge Regression model performs somewhat less well than Random Forest, but is highly competitive for this task; moreover the light-scale insight ability of its coefficients and regularization benefits [18]. Decision Trees, albeit serving as a good baseline, greatly benefit from ensembling approaches with dual data-structures [21].

6. Self-Assessment

During this project, I have learned a lot in the implementation of machine learning models on real-world datasets especially when it comes to predicting house prices. This required me to handle two well-known datasets namely the Boston Housing dataset and Ames Housing dataset. Both the datasets came with their own set of challenges and opportunities that really helped me to have a hands-on experience in different modelling techniques, data preprocessing, evaluation metrics which made my grip stronger on machine learning pipeline.

ML Algorithms Understanding:

Building and trying multiple machine learning algorithms on the datasets - A very deep depth of familiarisation. I began with Decision Tree Regression as it is non parametric and allows complex interactions between features to be captured by recursively splitting the data. When I implemented Decision Trees on house price prediction for both the Boston as well as Ames dataset not only did it help me in learning how trees could work with numerical and categorical features, but also its shortcomings like tendency to overfit if hyperparameters are left unchanged.

To counter balance the overfitting, I played around with random forest regression which is an ensemble learning method that fits a number of decision tree regressors on various sub-samples of the dataset and use averaging to improve accuracy. A large number of trees in this approach improved the performance by averaging predictions across many trees such that helped reduce variance. This was especially clear when we applied Random Forests to the Ames dataset which demonstrated how ensemble methods could help in high dimensionality data with large number of features. Tuning hyperparameters such as the number of trees and maximum depth once again made me realize how much these parameters affect Model complexity, overfitting and more.

Part of the project is also on implementing Ridge Regression, which was a linear model with L2 regularization. This was especially useful for the Boston Housing dataset since it suffered from multicollinearity where some features were highly correlated with each other. Ridge Regression (penalizing large coefficients) led to a more robust and thorough model with predictions that were less likely to overfit compared to basic linear regression techniques. This taught me about the delicate balance between bias and variance, as well as how regularization can help you walk that tightrope with your model.

I also used Support Vector Regression (SVR) with different kernel namely linear, polynomial and radial basis function(RBF). One of the reasons why I was able to model clearly so many feature interactions in Ames Dataset as SVR provided me with capability handle both linear relationship and non-linear relationships. The RBF kernel especially performed better than both the linear and polynomial kernels, as one would expect of it to perform best at capturing non-linear patterns in data. So this experience reinforced the importance of choosing kernel in SVR and also gave a good understanding about how different kinds of kernels can affect model accuracy.

Why Data Preprocessing & Feature Engineering is important for a model:
A large part of the work went down into data pre-processing which was found to be one key ingredient for successful model building. Data Cleaning : I cleaned the data by dealing with missing values, normalized features and encoded categorical variables for both of our datasets. This was especially true in the Ames dataset where the data as complex and varied. Using the appropriate methods to impute missing values (e.g., median for numeric variables and mode of categorical ones) allows machine learning models to better learn from data without being misled.

Another important step in preprocessing was feature scaling as Ridge Regression and SVR are some of the algorithms that require features to be scaled. This would bring all features down to the same scale, also ensuring that every input feature contributes roughly equally to each prediction made by our model (so a real-valued number of much larger range does not get interpreted as having more important significance than any other numerical value in an individual training example).

I carried on a feature engineering procedure to increase the predictive capacity of models. This also included production of Polynomial Feature, Ridge Regression to help the model capture non-linear relationships between Features and Target variable. Identifying the most important features again through correlation analysis and expert knowledge was key to making our models interpretable, but also directed us towards better modelling performance. The analysis showed that certain features such as 'RM' (effectively the average number of rooms per dwelling and 'LSTAT' (% lower status population), in the Boston dataset, and may be used to predict body mass index feature are important predictors. 'OverallQual' : Overall material finish quality, 'GrLivArea': Above grade living area which can also consider important variables based on this study.

On the other hand: Using sophisticated tools
It also gave us practice doing more advanced things like bagging and cross-validation. Bagging, as implemented through Random Forests, has the distinct advantage of both decreasing variance and making predictions more robust. In the Ames dataset, with its naturally challenging high dimensionality due to diverse categorical variables recorded down by MEALS, this method was especially helpful. Bagging averaged the predictions of a number of decision trees — making not only an increase in its generalization to unseen data but also bringing it closer to improved accuracy.

Cross Validation: Cross-validation is another important technique used to evaluate how the model will perform. Such a split is employed at the outset of parameter estimation in order to obtain an initial estimate and measure confidence, where multi-fold replication provides higher reliability. Using this method greatly helped in avoiding overfitting, and gave a smooth idea about how the model would perform on new unseen data. Using CrossValidation also illustrates the necessity of robust estimation in machine learning, such that results are replicable across train-test splits and not overfitted to specific runs.

Barriers and Opportunities:
This success was not without its share of challenges to elucidate areas needing improvement. One of the main difficulties was being able to optimize hyperparameters efficiently. This becomes very much tricky when on one hand you have computational efficiency, and on other high level complex models like Random Forests or SVR. Tuning of Parameters The process included tuning parameters like

the number of trees in Random Forests, regularization strength in Ridge Regression and kernel parameters at Support Vector Regressor (SVR) which needed attention to detail was cumbersome and also involved a bit of Hit & Trial.

Also dealing with the complexity of Ames dataset containing many features as well and needed more powerful preprocessing and feature selection techniques. I was able to solve most of these problems, but implementing some automated pipeline for selecting and tuning models would be a great step forward especially if we are dealing with big datasets or complex models. This process can be accelerated by implementing tools like automated machine learning (AutoML) frameworks in future projects.

We also drove home the need for extensive model validation. Our results demonstrated that cross-validation is a good method to estimate the performance of ensemble regressions, but as seen in our analysis using both datasets (Boston and Ames) there are considerable differences between them, so we used other strategies which were specific for each section. I think this has underscored the need for techniques to accommodate with some of these facets based on specifics in a data set.

Conclusion:
In general, this project was a tremendously enlightening discovery that completed the whole idea of integrating machine learning models with real-world datasets. This includes various algorithms, data pre-processing & feature engineering and so on so forth from a full end to end machine learning workflow. The problems that we faced during the implementation of the project have identified some areas more to be worked on such as hyperparameters parameters tuning, model validation etc. I hope to expand on these in the coming projects, grow as a data scientist and use more sophisticated approaches with even larger datasets.

7. References

1. Bokhari, S., & Geltner, D. (2011). Loss aversion and anchoring in commercial real estate pricing: Empirical evidence and price index implications. Real Estate Economics, 39(4), 635-670. https://doi.org/10.1111/j.1540-6229.2011.00307.x
2. Goodman, A. C., & Thibodeau, T. G. (2008). Where are the speculative bubbles in US housing markets? Journal of Housing Economics, 17(2), 117-137 https://doi.org/10.1016/j.jhe.2008.02.001

3. Glaeser, E. L., Gyourko, J., & Saks, R. E. (2005). Why is Manhattan so expensive? Regulation and the rise in housing prices. The Journal of Law and Economics, 48(2), 331-369. https://doi.org/10.1086/429979
4. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning: with Applications in R. Springer Publishing. https://doi.org/10.1007/978-1-4614-7138-7
5. Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Series in Statistics. Https://doi.org/10.1007/978-0-387-84858-7
6. Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5-32. https://doi.org/10.1023/A:1010933404324
7. De Cock, D. (2011). Ames, Iowa: Alternative to the Boston Housing Data as an End of Semester Regression Project. Journal of Statistics Education. https://doi.org/10.1080/10691898.2011.11889627
8. Domingos, P. (2012). A few useful things to know about machine learning. Communications of the ACM, 55(10), 78-87. https://doi.org/10.1145/2347736.2347755
9. Murphy, K. P. (2012). Machine Learning: A Probabilistic Perspective. MIT Press. Https://doi.org/10.7551/mitpress/8916.001.0001
10. Witten, I. H., Frank, E., & Hall, M. A. (2016). Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann. https://doi.org/10.1016/C2014-0-03407-7
11. Pindyck, R. S., & Rubinfeld, D. L. (2013). Microeconomics. Pearson Education.
12. Fotheringham, A. S., Brunsdon, C., & Charlton, M. (2017). Geographically weighted regression: the analysis of spatially varying relationships. John Wiley & Sons. https://doi.org/10.1002/9780471721444
13. Harrison, D., & Rubinfeld, D. L. (1978). Hedonic prices and the demand for clean air. Journal of Environmental Economics and Management, 5(1), 81-102. https://doi.org/10.1016/0095-0696(78)90006-2
14. Breiman, L. (1996). Bagging predictors. Machine Learning, 24(2), 123-140. https://doi.org/10.1007/BF00058655
15. Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). Classification and regression trees. Wadsworth International Group.
16. Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. Journal of Statistical Software, 33(1), 1-22. https://doi.org/10.18637/jss.v033.i01
17. Cristianini, N., & Shawe-Taylor, J. (2000). An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press. Link https://doi.org/10.1017/CBO9780511801389
18. Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. Technometrics, 12(1), 55-67. https://doi.org/10.1080/00401706.1970.10488634
19. Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B (Statistical

Methodology), 67(2), 301-320. https://doi.org/10.1111/j.1467-9868.2005.00503.x

20. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825-2830. https://jmlr.org/papers/v12/pedregosa11a.html
21. Cutler, D. R., Edwards Jr, T. C., Beard, K. H., Cutler, A., Hess, K. T., Gibson, J., & Lawler, J. J. (2007). Random forests for classification in ecology. Ecology, 88(11), 2783-2792. https://doi.org/10.1890/07-0539.1
22. Little, R. J. A., & Rubin, D. B. (2019). Statistical Analysis with Missing Data. Wiley. https://doi.org/10.1002/9781119482260
23. Biau, G. (2012). Analysis of a random forests model. Journal of Machine Learning Research, 13, 1063-1095. https://jmlr.org/papers/v13/biau12a.html
24. Liaw, A., & Wiener, M. (2002). Classification and regression by randomForest. R news, 2(3), 18-22. https://CRAN.R-project.org/doc/Rnews/
25. Louppe, G. (2014). Understanding random forests: From theory to practice. arXiv preprint arXiv:1407.7502. https://arxiv.org/abs/1407.7502
26. Smola, A. J., & Schölkopf, B. (2004). A tutorial on support vector regression. Statistics and Computing, 14(3), 199-222. https://doi.org/10.1023/B:STCO.0000035301.49549.88
27. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press. https://www.deeplearningbook.org/
28. Loh, W. Y. (2011). Classification and regression trees. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 1(1), 14-23. https://doi.org/10.1002/widm.8
29. Arlot, S., & Celisse, A. (2010). A survey of cross-validation procedures for model selection. Statistics Surveys, 4, 40-79. https://doi.org/10.1214/09-SS054
30. Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. Journal of Machine Learning Research, 13(Feb), 281-305. https://jmlr.org/papers/v13/bergstra12a.html
31. Bureau of Labor Statistics. (2020). Consumer Price Index (CPI) data. U.S. Department of Labor. https://www.bls.gov/cpi/
32. Basak, D., Pal, S., & Patranabis, D. C. (2007). Support vector regression. Neural Information Processing, 17(3), 203-224. https://doi.org/10.1080/10691898.2011.11889627

8. How to use the Project

This project aims to build and test prediction models for house prices, estimating of the Boston (Massachusetts) real estate market and Ames(Iowa), through advanced techniques of machine learning. This is a step-by-step guideline, on how to better navigate and use the different parts of the project for both practical tasks as well further research.

1. Understanding the Project Structure:

- **Data Preprocessing:** This is an important module where the datasets (Boston Housing & Ames Housing) will be pre-processed before being fed into training. The preprocess includes missing impute, normalize feature and one hot encoding They are plain Python scripts that automate these tasks so the data is clean, consistent and ready for analysis.

- **Model Development:** This contains use of various machine learning models such as Decision Tree Regressor, Random Forest Regressor and Ridge Regression. They are made using Python and Scikit-learn so you can edit them as needed. After preprocessing, the models are trained and tested on that data; this is where performance metrics also come into play.

- **Hyperparameter Tuning:** Hyperparameter Tuning: For maximizing model performance, hyperparameters are tuned (like n_estimators and max_depth in our case.) Hyperparameters are values which direct how an algorithm learns: for instance, the maximum depth of tree in Decision Trees, number of trees in Random Forests and regularization parameter is Ridge Regression. Grid Search and Cross-Validation processes are using to identify the best hyperparameters which reduces prediction errors and generalizes well.

- **Model Evaluation:** Performance evaluation of each model with MSE, RMSE, and R^2 (M) Finally, we check residual plots and actual vs. predicted plots to judge the model input as well as output distribution visually for some scope of improvements if required in those inputs or outputs.

- **Inflation Adjustment Analysis:** Since inflation has affected housing prices over the many years of the Ames Housing dataset, we include an analysis to adjust for these effects. From this chart where the real and nominal prices are shown side by side, after we convert all values to 2020 dollars using CPI you get an insight into longer-term price trends that would not be visible from just looking at current or historic pricing.

2. Running the Project:

In order to work with this project you will need the following:

- **Prerequisite:** Have Python and the necessary libraries installed. The main dependency are pandas, numpy, scikit-learn and matplotlib. The requirements are provided, so you can install these libraries. txt file.
- **Load Data:** Put Boston and Ames datasets to proper directory The format of the files is identical to the training data, so they will be automatically loaded and processed by your data preprocessing script just like with training.
- **Train the Model:** You must train the model by running a script containing all regression models. Scripts are broken down so they can be run on their one, meaning you do not have to deal with all packages when only working in 1 model. In the console, you'll see training results including performance metrics and can save these for more analysis later.
- **Evaluate and Visualisation:** Work provides an evaluation script to find the metrics as well. These will be the outputs that you will use to check if your models perform well, or otherwise are overfitting / underfitting.
- **Inflation Adjustment Analysis:** If you are using Ames dataset, perform inflation adjustment script to convert nominal prices into real priced. You don't have to do this (we only really need the most recent data) but I would recommend downloading all of them as it'll help with testing.

3. Extending the Project:

- **This project is designed to be easily extendable:**
  For Adding new models: You can add the other machine learning algorithms by just following this Model Development Section and structure. This approach is extensible to adding new models — add corresponding scripts and preprocess the data in a similar way, then put them into evaluation.

- **Personalized Preprocessing:** The data preprocessing can be changed to introduce more manipulations or attend other challenges in different datasets.

- **Dynamically Adaptable to Real Estate Datasets:** Project is easy to adapt for other real estate datasets. Make sure the new datasets are of similar format as from Boston and Ames for compatibility with preprocessing and modelling scripts that exist.

- **Conclusion:**
  This project is a complete machine learning model for housing price prediction. By walking through the above steps, you should be able to train and evaluate different regression models while also learning about model performance and what is driving housing prices. Whether it's used for academic research, professional development or practical applications in real estate this project is here as a sturdy platform to kick off further exploration and experimentation.