**House Pricing Predictions**

Problem –

The primary problem we are addressing is predicting house prices based on property features. The goal is to develop a model that provides accurate price estimates, helping stakeholders in the real estate market.

Methodology –

* We use machine learning techniques to build a predictive model for house prices. The process involves data preprocessing, model selection, training, and evaluation.
* Programming language – python
* Model types - Random Forest Regressor, Decision Tree Regressor, Linear Regression

Dataset description –

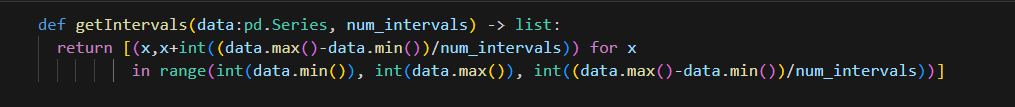
The dataset contains 21k rows of house sales details. Each sale have the following parameters – id, date, price, bedrooms, bathrooms, square footage of the home, square footage of the lot, floors, waterfront, view, condition, grade, footage of house apart from basement, square footage of the basement, Built Year, Year when house was renovated, zip code, Latitude coordinate, Longitude coordinate, Living room area in 2015, lot Size area in 2015.

Pre-processing (price prediction) –

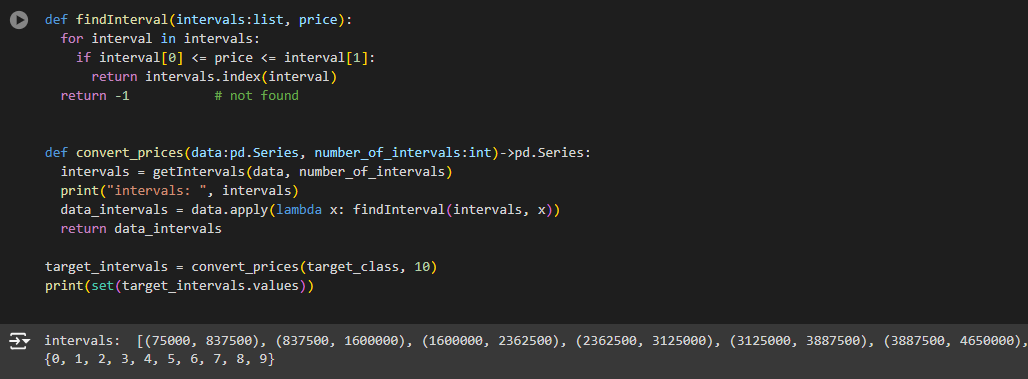
1. Drop unnecessary features – date, zip code, id.
2. Find each house location by its “lat” and “long” values using geopy library.
3. All the locations are in Washington, so they don’t contribute to the prediction process.
4. Drop the “lat” and “long” columns, so the data remain with 15 columns and the target column – price.
5. Separate the “price” column from the data – it will be the target class
6. Split the data to train and test using the method “train\_test\_split” with

test\_size = 0.2 and random\_state = 1.

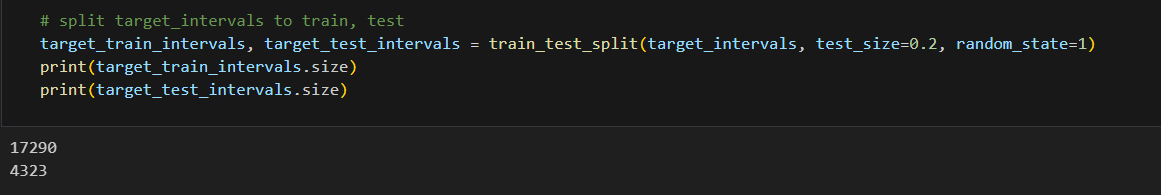
1. The train set contain 17290 rows and the test set contain 4323 rows.
2. Build a function that returns a list of ranges according to the requested number of intervals



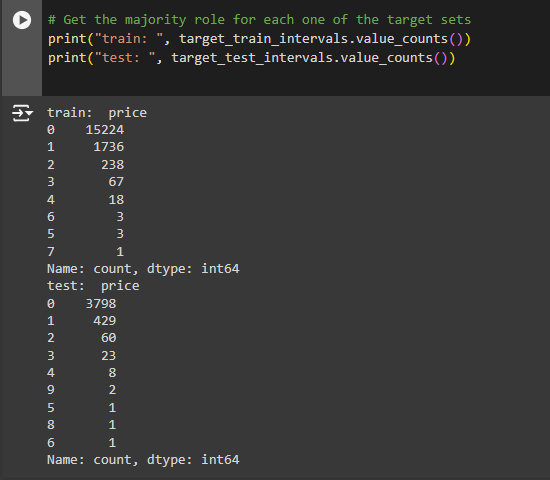
1. Building functions that go through the entire series and change the values ​​to their intervals according to the ranges



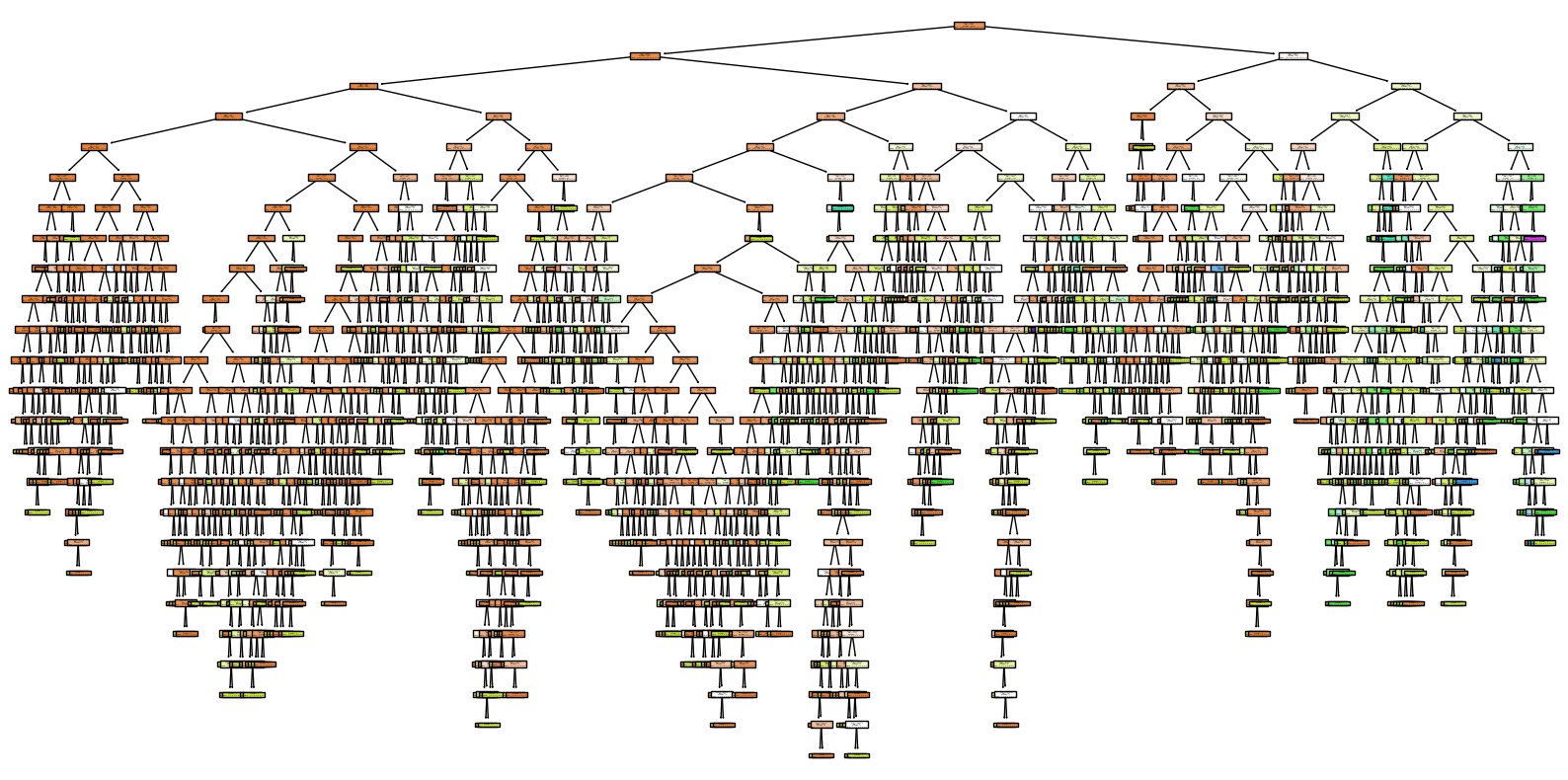
1. Split the new series (the interval one) to fit to the train data and test data



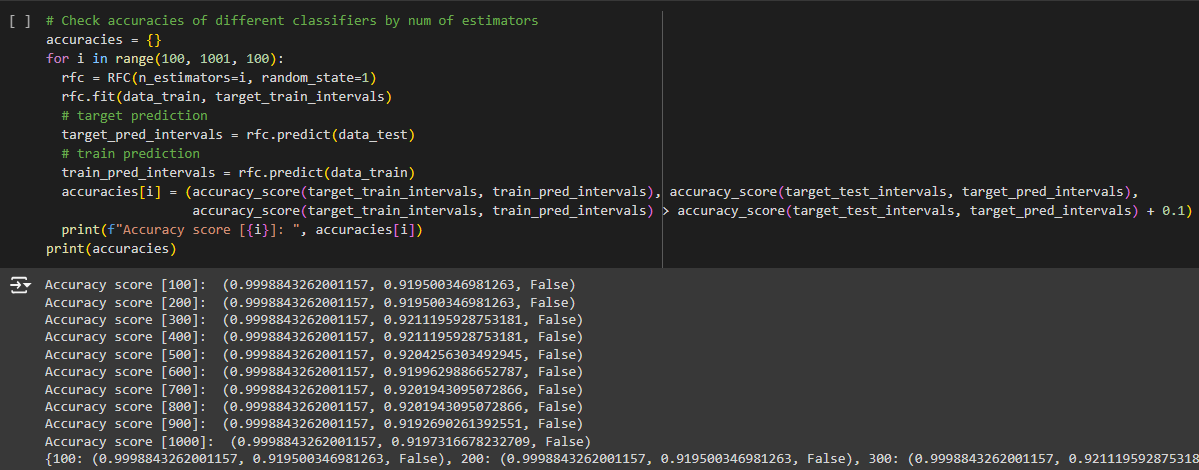
1. Get the majority rule of each one of the subsets.



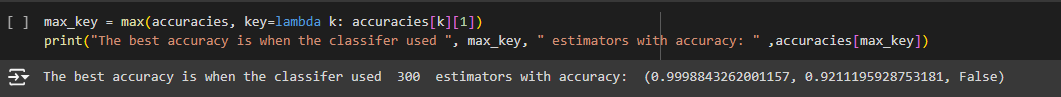
1. Initialize the RFC model and fit to the train data.



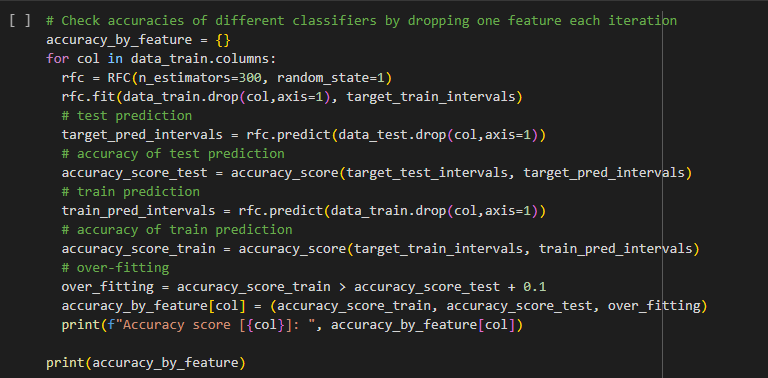
1. Predict the test data and the train data using “predict” function.
2. Get the test accuracy score– 0.919500346981263, train accuracy score – 0.9998843262001157 -> No over-fitting :).
3. Checking the dependency between the number of the model’s estimators and the accuracy with for loop and dictionary that contains all the results.



1. The best score is :



1. Checking the influence between each feature and the accuracy with for loop and dictionary that contains all the results.



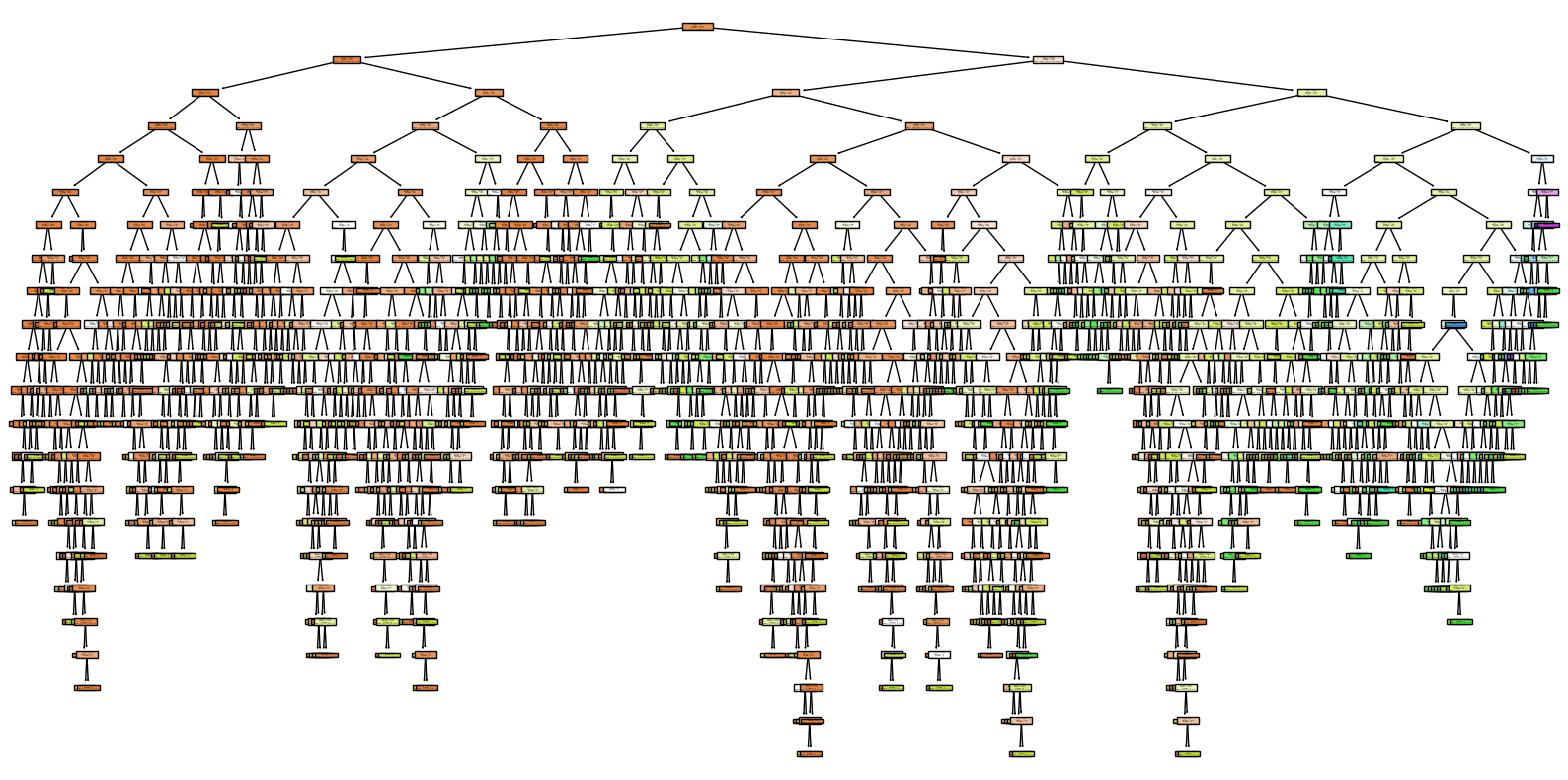
The results:

A computer screen with numbers and symbols

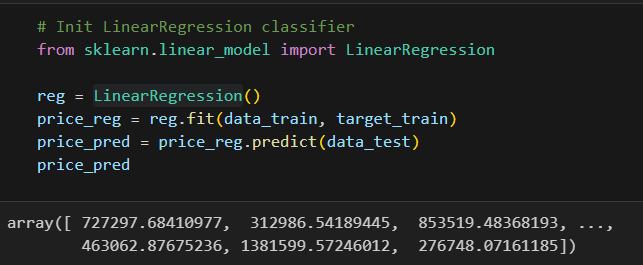
Description automatically generated

According to the results, no feature has massive influence on the accuracy score, so we don’t need to drop non of the features to improve the prediction skills.

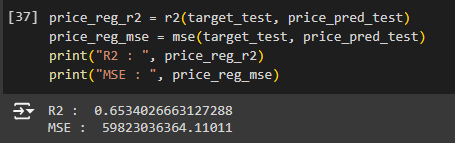
1. Initialize a Decision Tree Classifier and fit it with the data.



1. Predict the test data and the train data using “predict” function.
2. Get the test accuracy score– 0.8924358084663429, train accuracy score – 0.9998843262001157 -> the distance between the accuracies is 0.107 so we assume there is no over-fitting.
3. Initialize and fit a Linear Regression model to predict the exact prices.
4. The prediction gave an array with float numbers, so we apply the round method to get the integer values

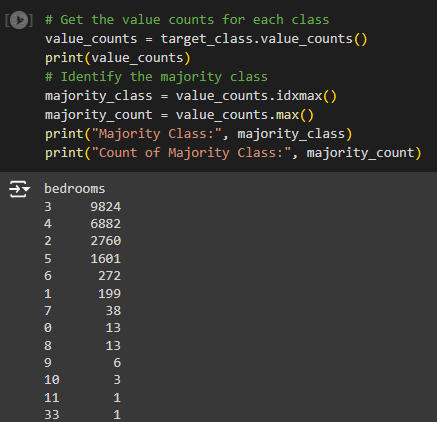


1. Predict the r2 and the mse:



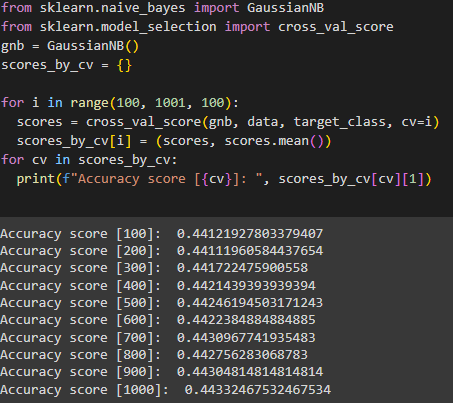
Pre-processing (bedroom prediction) –

1. Reopen the dataset to make a prediction on the bedroom column.
2. Drop the unnecessary columns (id, zipcode, lat, long, date).
3. Find the distribution of the values in the target class.

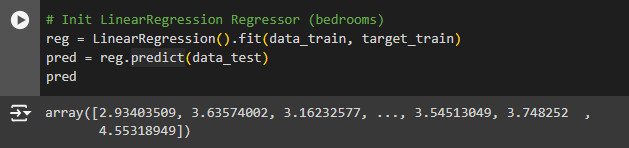


1. Checking the influence of different number of cv on the accuracy scores.

First, we run the for loop on numbers between 3 to 10 and the results was around 0.42. Next, we run the for loop on numbers between 100 to 1000 with steps of 100.



1. Split the data to train and test.
2. Initialize Linear Regressor model and fit to the train data.
3. Predict the test data – the model returned all the results as float numbers, so we used the round function.

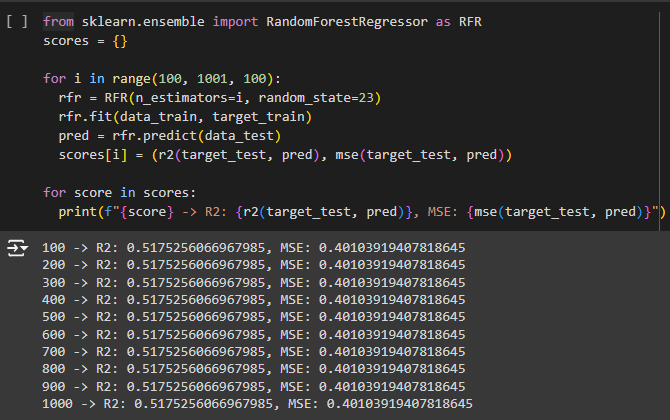


1. Get the R2 and mse scores of the model.

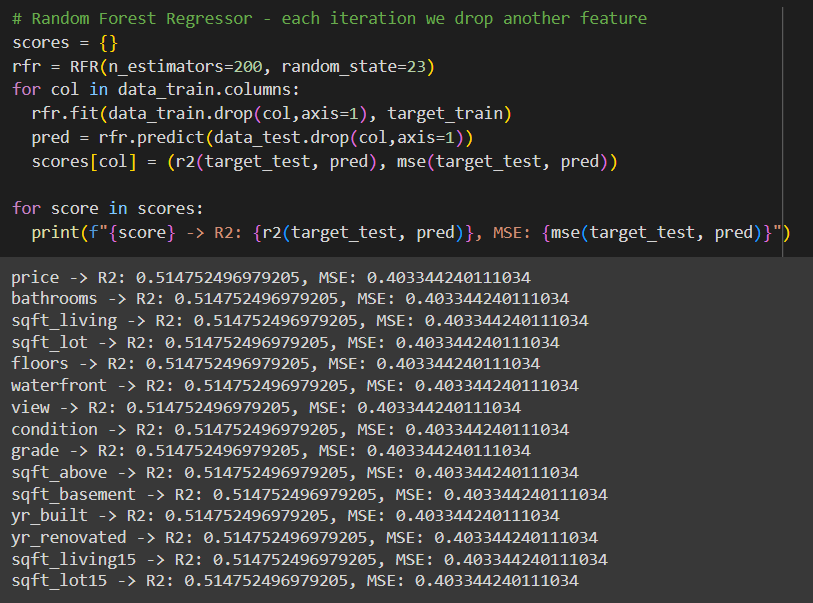
A screenshot of a computer program

Description automatically generated

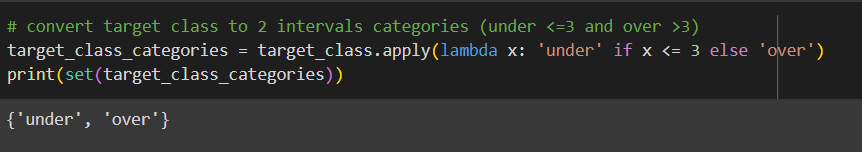
1. Using a for loop, we tried to test if the number of estimators affects the r2 and mse of the model.



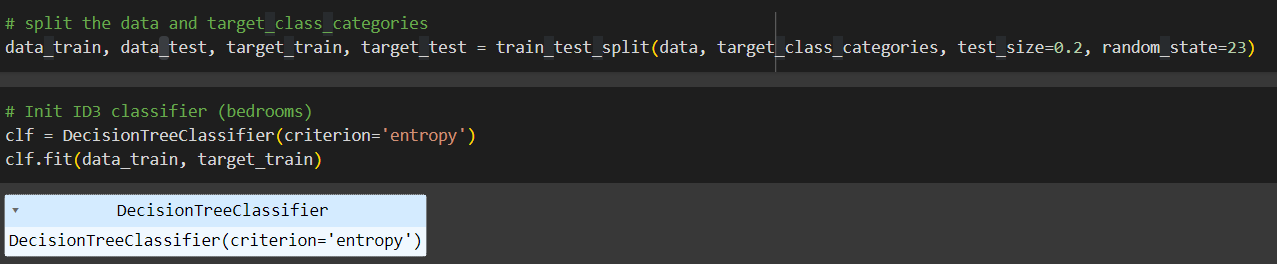
1. Using a for loop, we tried to test if there is a feature that affects the prediction more than the other features.



1. According to the values distribution in the ‘bedrooms’ class, we chose to divide the data to 2 main intervals – under 3 (<=3) and over 3 (>3).
2. Now, apply the changes to the target class.



1. Split the data to train, test subsets.
2. Initialize the ID3 classifier and fit with the train set.



A group of people in a line

Description automatically generated

1. A screen shot of a computer code

   Description automatically generatedPredict the train and test subset using ‘predict’ function.
2. Calculate the accuracy of the model and check whether the model suffers from over-fitting.

