A logo for college computing

Description automatically generated

**Assessment Cover Page**

|  |  |
| --- | --- |
| *Student Full Name* | Alina Marrulyn Lozano Flores |
| *Student Number* | 2024068 |
| *Module Title* | Machine Learning (10 ETCS) |
| *Assessment Title* | Prediction of house pricing |
| *Assessment Due Date* | 21-04-2024 |
| *Date of Submission* | 21-04-2024 |

**Declaration**

By submitting this assessment, I confirm that I have read the CCT policy on academic misconduct and understand the implications of submitting work that is not my own or does not appropriately reference material taken from a third party or other source.

I declare it to be my own work and that all material from third parties has been appropriately referenced.

I further confirm that this work has not previously been submitted for assessment by myself or someone else in CCT College Dublin or any other higher education institution.

**Contents (1486 words)**

[Introduction *(91 words)* 2](#_Toc164633739)

[Data Cleaning 2](#_Toc164633740)

[Exploratory data analysis – EDA *(120 words)* 2](#_Toc164633741)

[Dummy variable (112 words) 3](#_Toc164633742)

[Identifying Outliers in area and price (65 words) 4](#_Toc164633743)

[Heatmap/ Correlation matrix (55 words) 5](#_Toc164633744)

[**Scaling our Data** (46 words) 5](#_Toc164633745)

[Our Machine Learning Models (695 words) 5](#_Toc164633746)

[Splitting 6](#_Toc164633747)

[Random Forest Regressor 6](#_Toc164633748)

[Feature importance 7](#_Toc164633749)

[Results of our RFR 7](#_Toc164633750)

[Results with Hypertuning 8](#_Toc164633751)

[Support Vector Regression 10](#_Toc164633752)

[Results 10](#_Toc164633753)

[Results with Hypertuning 11](#_Toc164633754)

[Cross Validation (37 words) 12](#_Toc164633755)

[Conclusions (265 words) 13](#_Toc164633756)

[Bibliography 14](#_Toc164633757)

[Github 14](#_Toc164633758)

# Introduction *(91 words)*

Data analysis is widely used to predict housing prices. Regressor Machine learning models can help to predict prices, and indicate the likely price at which a property will sell.

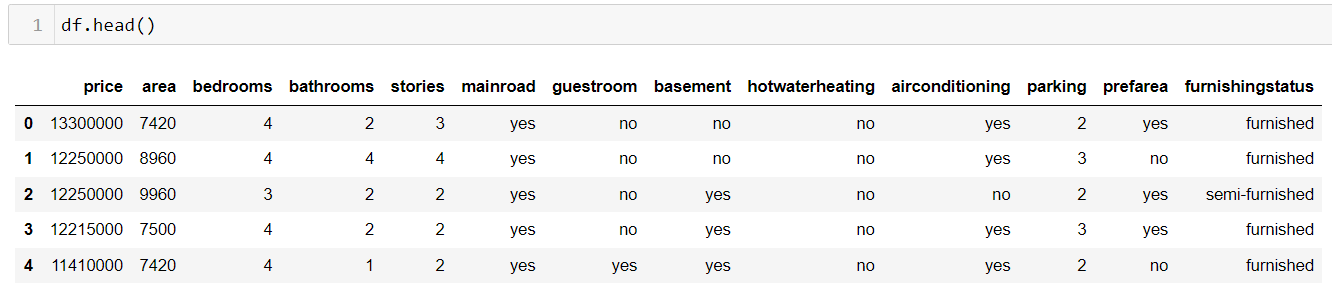
This dataset contains 13 variables, and this project’s goal is to predict prices. We used two methods, (1) Random Forest Regressor and (2) Support Vector Regression (SVR), comparing the results through Cross Validation (CV), to determine which model works better.

We will have three training splits, with a 20, 25 and 30 percent of test size, with the intention to compare results.

# Data Cleaning

# Exploratory data analysis – EDA *(120 words)*

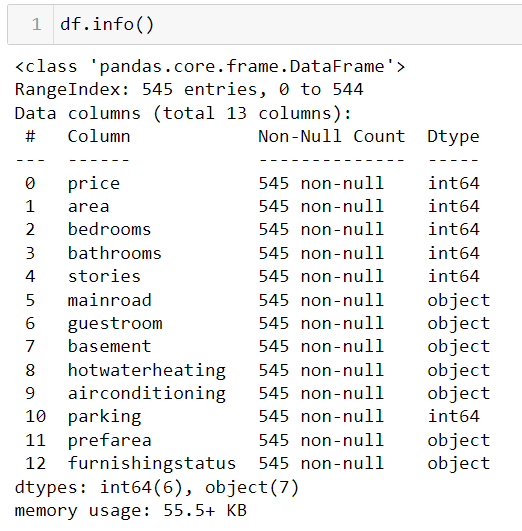
For this analysis our data set is called Housing in format csv, with the function “df.head()” we can see the first five columns:



*Fig 1*

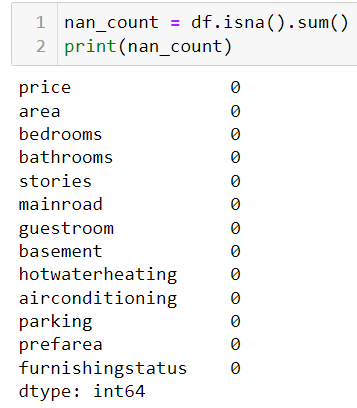
As we can see in figure 1, we have 13 columns, six numerical values, and seven category columns. With the function info, *“df.info()”,* we observe 545 rows or samples.

With the function *“info()”*, we see the type of columns we have.



*Fig. 2*

We do not see any numerical column as a ‘Dtype object,’ meaning that our dataset does contain text in numerical columns; with the function “nan\_count = df.isna().sum()” we can corroborate that there are not *nan* values (Fig. 3).



*Fig. 3*

Our dataset is therefore clean, and we can proceed to scale our values and apply Dummy Variable Encoding.

# Dummy variable (112 words)

For categorical data we should convert into numbers, we use *Dummy Variable Encoding* to convert our categories into 0 or 1, with dummy encoding the category columns are split into new columns: for example, the column “mainroad” had two values, *yes or no,* after we apply the *dummy variable*, the column is splitted in 2 new columns: “mainroad\_yes” and “mainroad\_no”.

If the row had mainroad, the column “mainroad\_yes” has the number 1=True and “mainroad\_no” is has the number 0=False. This function in phyton is called “pd.get\_dummies(df)”.

In figure 4, we now have 21 columns, since five of our category columns are split into 2 columns, with the column furnishing status into three.



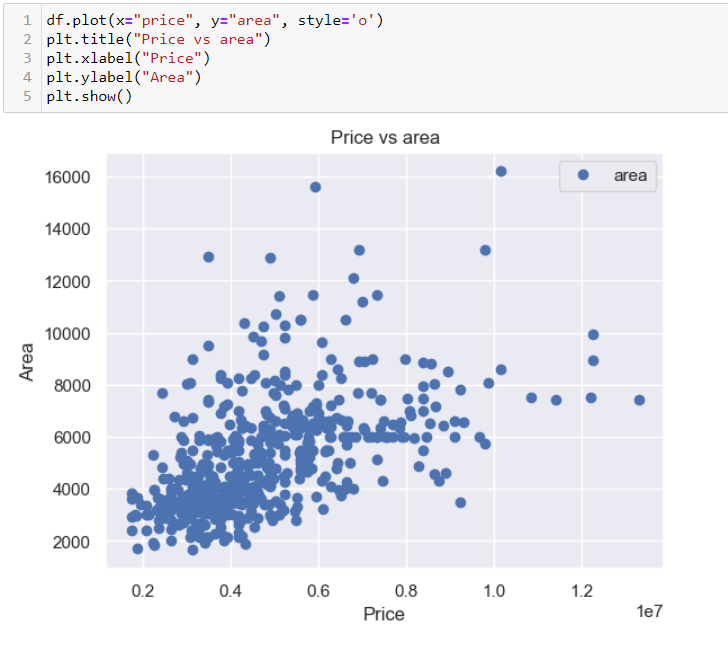
*Fig.4*

# Identifying Outliers in area and price (65 words)

In figure 5, we see the correlation between price and area; the first graph shows the correlation with the outliers, and the second graph shows the correlation without outliers.

Our graphic shows a relationship between area and prices, but there are other factors influencing the final price since the correlation is not strictly linear.

Removing the outliers reduce the area values from 16,000 to 11,000.

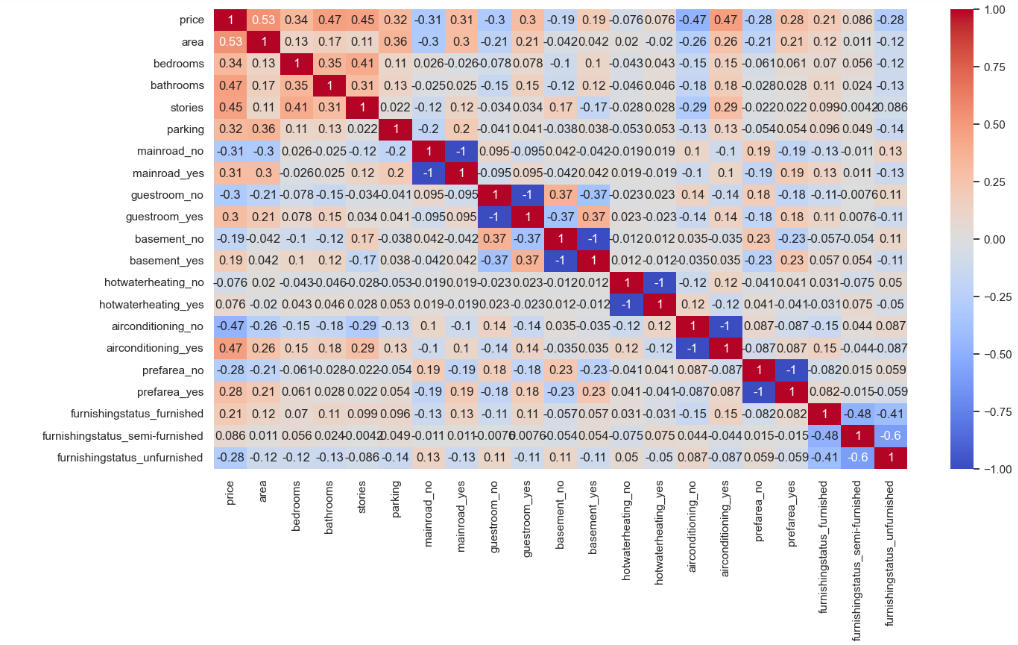


*Fig. 5*

# Heatmap/ Correlation matrix (55 words)

Heatmaps help us to observe datasets through the use of colors. They help us to observe correlations among features that we cannot easily detect by simply observing tables. They can identify trends and atypical values.

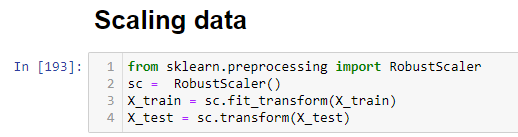
According to our heatmap, the most explanatory variable to predict prices is the area, followed by the feature ‘bedrooms’ and ‘bathrooms’.

****

*Fig.6*

# **Scaling our Data** (46 words)

We used RobustScaler for our analysis. This ensures that the statistical features of our dataset are on the same scale. RobustScaler is similar to the StandarScaler but the RobustScaler uses the median and quartiles, instead of the mean and variance. This helps us to remove outliers.

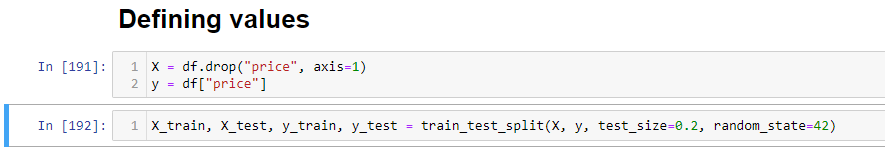
****

*Fig. 7*

# Our Machine Learning Models (695 words)

## Splitting

For this project we split our data 3 times, with test sizes of 20%,25% and 30%, after which we will compare our results. In figure 8 we observe that our test size is 0.2 (20%).

****

*Fig. 8*

We used two methods, Support Vector Regressor (SVR) and Random Forest Regressor. After comparison, SVR is generally preferable for predicting prices. RFR is good for testing because it is based on decisions and hierarchy, and is therefore used for classification and regression.

## Random Forest Regressor

Random Forest Regressor (RFR), uses regressor trees, rather than classification trees.

A random forest is an algorithm combining results of multiple decision trees, therefore improving overall performance and controlling over-fitting.

Each decision tree predicts the house price base in our values. The result gives us the mean price of all outputs.

On figure 9 we see the parameters of our code:

***N\_estimators .-*** *The number of decision trees in our code is 300*

***Max\_depth*** *.- The maximum possible depth of each tree is 5*

***Max\_features.-*** *The maximum number of features the model will consider determining a split, on this case square root.*

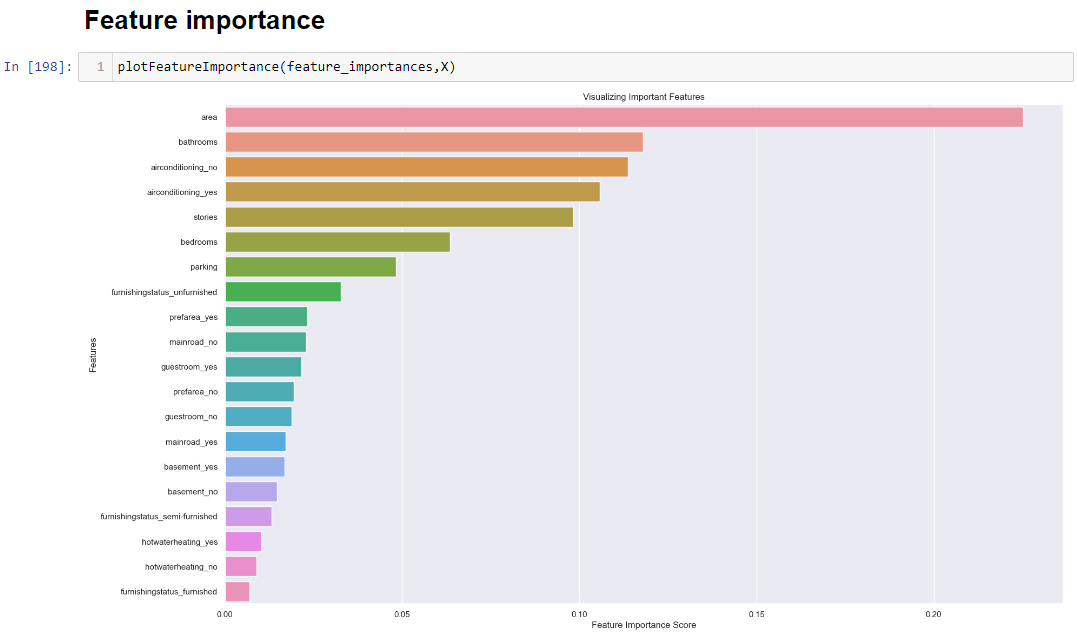
***Random state.-*** *We define the random samples used to shuffle the data before splitting it. In this case is 42*

****

*Fig. 9*

### Feature importance

The graph below (fig.10), shows that the most important feature driving our dependent variable is the area, followed by number of bathrooms.

****

*Fig.10*

### Results of our RFR

With a test size of 20%, the results of our metrics show that our training data has a score of 0.68; however, our test data has a score of 0.57, which means that our data could be slightly over-fitted, and the accuracy is low, at only 0.57. (Fig.11)

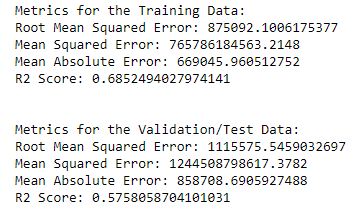
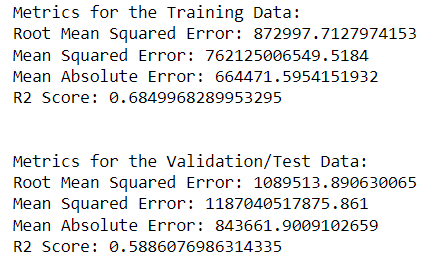


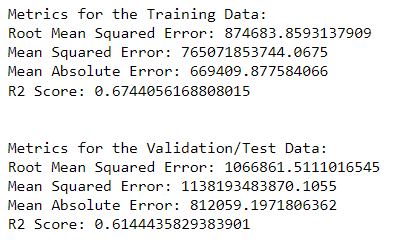
Fig. 11

With a test size of 25%, the results of our metrics show that our training data has a score of 68%;, however, our test data has a score of 58%, meaning that our data could be slightly over-fitted. (Fig. 12).



*Fig. 12*

With a test size of 30%, the results of our metrics show that our training data has a score of 0.67; however, our test data has a score of 0.61 (Fig. 13).



*Fig.13*

### Results with Hypertuning

The hyper parameter optimization allows us to optimise the parameters for our model.

The results, with a test *size of 20*%, after tuning the hyper parameters are:

*N\_estimators*: 700

*Max\_depth*: 7

*Max\_features*.- sqrt

*Random state.-*  35

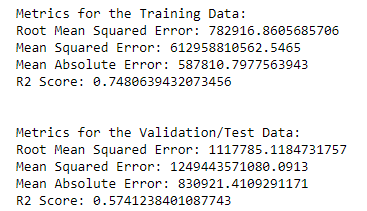
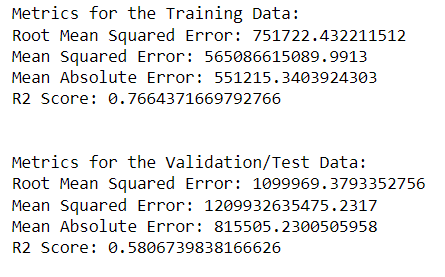


Fig. 14

The results of our metrics (Fig.14) show that our training data has a score of 0.74; however, our test data has a score of 0.57, this means that our data is overfitted.

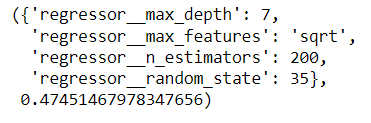
The results, with a test *size of 25*%, after tuning the hyper parameters are:



*Fig.15*

The results of our metrics (Fig.15) show that our training data has a score of 0.76; however, our test data has a score of 0.58, meaning that our data is overfitted.

The results, with a test *size of 25*%, after tuning the hyper parameters are:



*Fig. 16*

*N\_estimators*: 200

*Max\_depth*: 7

*Max\_features*.- sqrt

*Random state.-*  35

The results, with a test *size of 30*%, after tuning the hyper parameters are 0.60. and for the test data 0.54, meaning that the result is less overfitted.

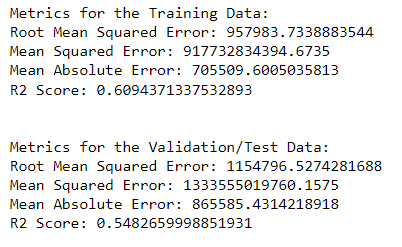


Fig. 17

*N\_estimators*: 600

*Max\_depth*: 5

*Max\_features*.- sqrt

*Random state.-*  60

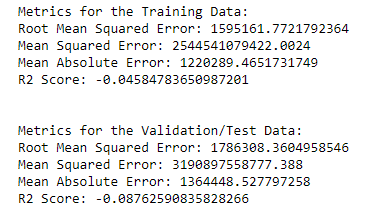
## Support Vector Regression

A linear regression calculates the closest 'fit' (typically the least squared) in the database. The best way of describing this fit is 'R2', which should be as high as possible, with the maximum being 1.00. For use as a forecasting tool, the average forecast Standard deviation is an indicator of usefulness.

This model is generally used to predict prices.

### Results

The results with a test *size of 20%* are:



*Fig. 18*

The results with a test *size of 25%* are:

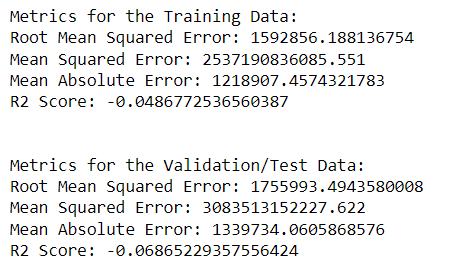
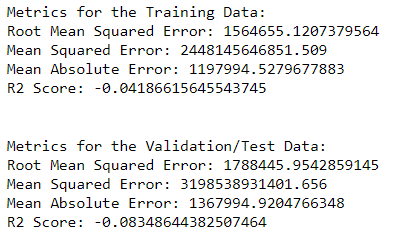


Fig.19

The results with a test *size of 30%* are:

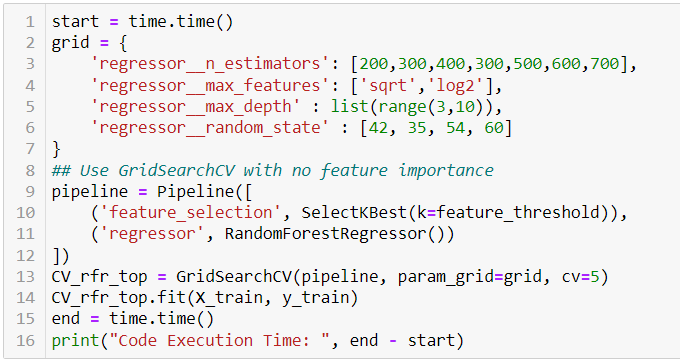


*Fig.20*

As we can see our result is negative, we can use hypertuning (GridSearchCV) to find the best parameters.

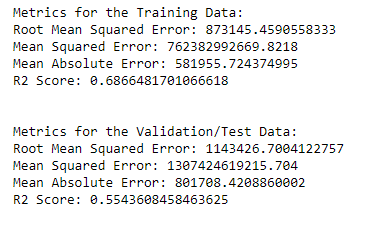
### Results with Hypertuning

GridSearchCV operates an internal cross-validation technique; to get the result, we calculate the score for each combination of parameters on the grid. We can therefore define the parameters in our grid.



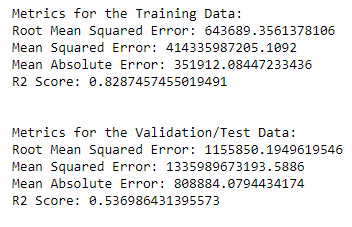
*Fig.21*

The results, with a test *size of 20*%, after tuning the hyper parameters are:

****

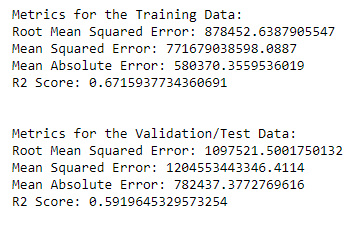
*Fig. 22*

The results, with a test *size of 25*%, after tuning the hyper parameters are:



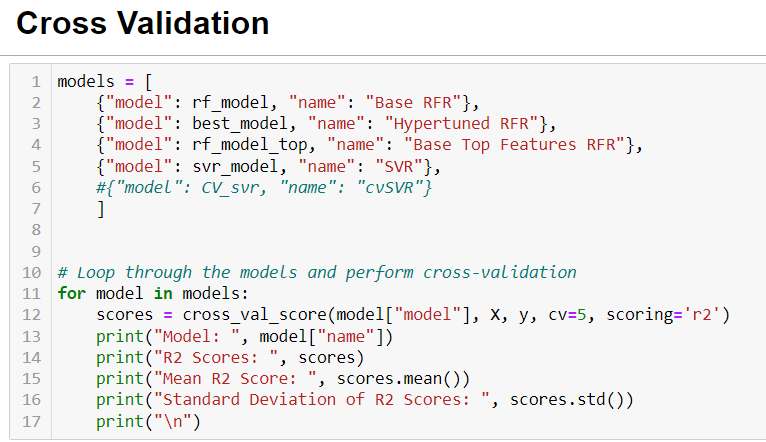
*Fig.23*

The results, with a test *size of 30*%, after tuning the hyper parameters are:



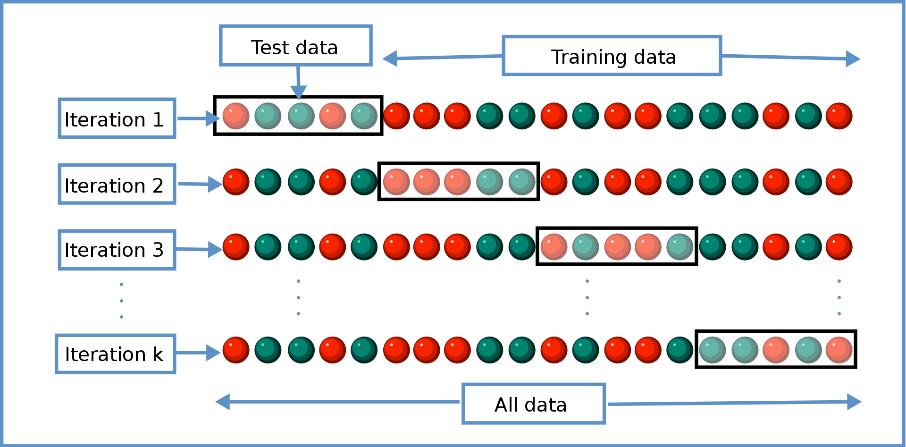
*Fig. 24*

# Cross Validation (37 words)



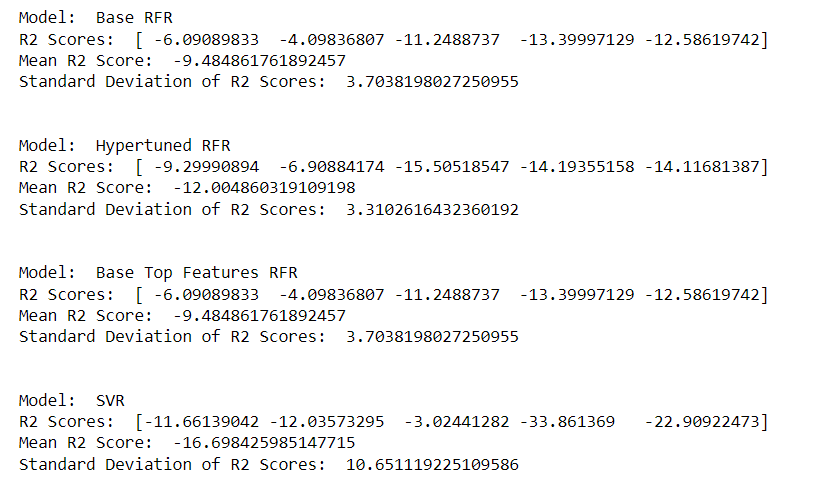
*Fig. 25*

Finally, we used cross-validation from the library scikit learn, *the cross\_val\_score* function uses different parts of our data set, and with different iterations changes the test data to get a better average, and therefore improve our results.



*Fig 26.*

Cross validation based in a test size of 30% and 5 iterations:



# Conclusions (265 words)

These models allow us to experiment, and explore ML concepts and metrics. The choice of ML models is driven by recommendations for price prediction, and they are used for housing. The use of hyperparameters is useful to determine optimal parameters for SVR, but in RFR they gave us overfitting.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 20% | | 25% | | 30% | |
|  | Train data | Test data | Train data | Test data | Train data | Test data |
| RFR | 0.68 | 0.57 | 0.68 | 0.58 | 0.67 | 0.61 |
| RFRT | 0.74 | 0.57 | 0.76 | 0.58 | 0.60 | 0.54 |
| SVR | -0.04 | -0.08 | -0.04 | -0.06 | -0.04 | -0.08 |
| SVRT | 0.68 | 0.55 | 0.82 | 0.53 | 0.67 | 0.59 |

.

Hypertuning RFR has a high train data score (0.74), with a split of 75% trained data and 25% test data - but the test data score is only 0.57, meaning that our model is overfitted.

However, RFR with a split of 30% for testing, obtained a score of 0.67 for our trained data, and a score of 0.61 for our test data. This is the best result obtained since the scores are close.

The results of our Support Vector Regressor was negative, but after the hyperparameters we see a dramatic improvement; we see the best result with a split of 30% for testing, with an score of 0.67 for our trained data, and 0.59 for the test data.

In conclusion, best results were given by RFR, with a split of 30%, because the difference between the trained data and the test data is lowest. We can further experiment with other ML learning models to improve the results.

We recommended to increase the dataset sample to increase the probability of an accurate prediction.

# Bibliography

Suresh Kumar Mukhiya and Ahmed, U. (2020). *Hands-on exploratory data analysis with Python : perform EDA techniques to understand, summarize, and investigate your data*. Birmingham, Uk: Packt Publishing.

Bhavanasi , S.S., Navroop, S. and Munoz Ebratt, C. (2021). How can you use a heat map to visualize Machine Learning performance? [online] www.linkedin.com. Available at: https://www.linkedin.com/advice/3/how-can-you-use-heat-map-visualize-machine-learning-xgtnc#:~:text=Heat%20maps%20in%20Machine%20Learning%20can%20be%20used%20to%20identify [Accessed 21 Mar. 2024].

scikit-learn (2018). *3.2.4.3.2. sklearn.ensemble.RandomForestRegressor — scikit-learn 0.20.3 documentation*. [online] Scikit-learn.org. Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html>.

Sellahewa, K. (2023). *Random Forest Explained! (Regression and Classification Tasks)*. [online] www.linkedin.com. Available at: https://www.linkedin.com/pulse/random-forest-explained-regression-classification-tasks-sellahewa/ [Accessed 20 Apr. 2024].

Team, G.L. (2023). Hyperparameter Tuning with GridSearchCV. [online] Great Learning Blog: Free Resources what Matters to shape your Career! Available at: https://www.mygreatlearning.com/blog/gridsearchcv/#:~:text=GridSearchCV%20is%20also%20known%20as [Accessed 21 Apr. 2024].

# Github

Link: https://github.com/2024068/Machine-Learning-CA1