Alejandro Gleason Méndez UT EID ag77698

Task 3 - Data Science for Engineers

Taking into account the knowledge acquired during the last week, I came to the conclusion that this task would be solved much better with a Decision Tree rather than with Multiple Linear Regression, so this is the perfect opportunity to redo the model and increase the accuracy in predictions. But why does a Decision Tree work better? Mainly because of two reasons: Firstly, calculating the relations between variables is hard, specially hard when they are categorical, so while I did a good job at attempting to find correlation with Pearson's coefficient and with chi-squared, it still had a lot of room for improvement. And secondly, decision trees are easier to conceptualize and a great option for starters.

So let's go down the path of developing a good Decision Tree for this task, which according to CareerFoundry, is useful when breaking down complex data into more manageable parts (Hillier, W., 2020). And since I am just starting in the Data Science field, and I have over 20 variables for this model, I will follow down the path of the Decision Tree.

Preparing the data

For my model development, I decided to drop redundant variables such as Latitude and Longitude (since they are already contained on Suburb), unique variables such as Address and useless variables such as date, using the .drop() method.

Cleaning the data is still a very important step since I am going to be tweaking the parameters and I do not want the results to be affected by empty rows or wrong data types. So I followed the same process of filling empty numerical values with the median and deleting rows which held empty cells. I too removed cells that had 0 as land size assuming they represented an error. The data set ended up with 10,270 rows from the original 13,580 count, which is still a significant count.

It was also a good opportunity to rename the data frame misspelled columns using the rename() method.

	Suburb	Rooms	Type	Price	Method	Seller	Distance
count	10272	10272.000000	10272	1.027200e+04	10272	10272	10272.000000
unique	305	NaN	3	NaN	5	237	NaN

Image 1: New row count

I finally encoded categorical variables using sklearn LabelEncoder(), which is suitable for labels with values between 0 and n_classes - 1 (scikit learn, 2020); this makes the variable more manageable.

```
Suburb              int64
Rooms               int64
Type                int64
Price               int64
Method              int64
Seller              int64
Distance          float64
Postcode            int64
Bedroom             int64
Bathroom            int64
Car               float64
Landsize            int64
BuildingArea      float64
YearBuilt         float64
CouncilArea         int64
RegionName          int64
PropertyCount       int64
dtype: object
```

Image 2: Data types after encoding

Feature Engineering

As usual, choosing features is quite difficult, there is a whole subject on that, nevertheless I tried to leverage some sklearn tools to perform the combinations of the independent variables in the model. The tool that worked best for me was to do a correlation matrix heatmap, to search for linear relationships between the dependent and independent variables to combine them, and to search for linear relationships between the independent variables to avoid multicollinearity.

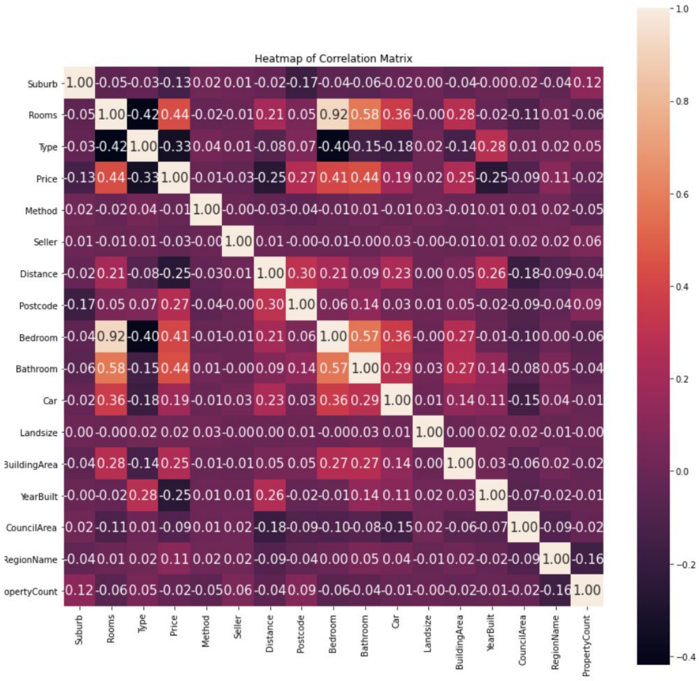

Image 3: Heatmap of Correlation Matrix

The map helped me develop my first models, which had a Mean Absolute Error (MAE) of around 25%, but I decided I could do better and developed a few more proposals to stick with the one with the lowest MAE. Note that the obtained MAE is before hyperparameter tuning.

Parameters used	MAE
Rooms, Distance, Bathroom, Car, Postcode, Landsize, BuildingArea, PropertyCount, Type, Method, Seller, CouncilArea, RegionName	21.98%
Rooms, Distance, Bedroom, Bathroom, Car, Landsize, BuildingArea, Suburb, Type, Method, YearBuilt	25.88%
Suburb, Rooms, Type, Method, Distance, Postcode, Bathroom, Car, Landsize, BuildingArea, YearBuilt	21.99%
Suburb, Rooms, Type, Method, Distance, Bathroom, Car, Landsize, BuildingArea, YearBuilt, CouncilArea, RegionName, PropertyCount	22.53%

Suburb, Rooms, Type, Method, Seller, Distance, Postcode, Bedroom, Bathroom, Car, Landsize, BuildingArea, YearBuilt, CouncilArea, RegionName, PropertyCount	22.16%

Model development

I started using a Decision Tree Regressor since it is a great model to get started. When doing that, I encountered the existence of both Classifiers and Regressors on sklearn, and for people who do not know their difference, as me, here it is: A classifier works for predicting a label, which means, mapping the class within which a target variable would most likely fall (Brownlee, J., 2017), while regression works for predicting a continuous variable, such as price.

So with that in mind, I first used a Decision Tree Regression and then I moved to a Random Forest Regressor, which frequently leads to better results since predictions are evaluated in multiple trees and not just in one.

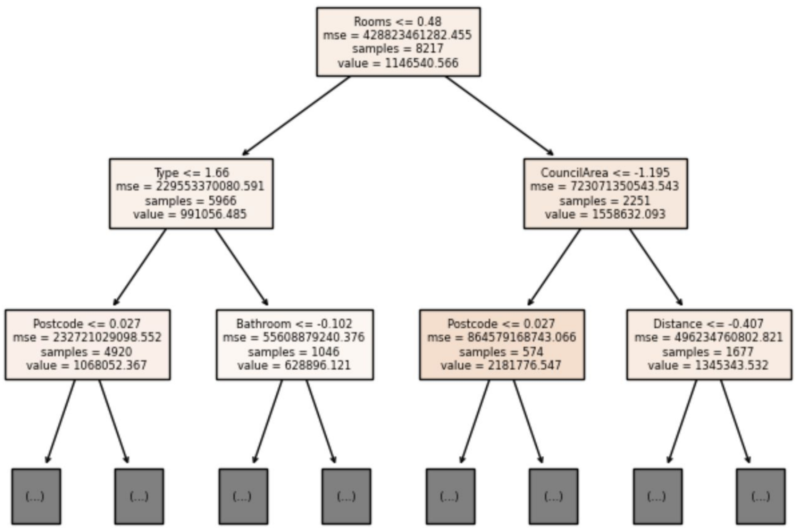

Image 4: Decision Tree Plot

Finally, I stepped up the game and did hyper parameter tuning for both models, managing to reduce the MAE in both cases. Hyper parameter tuning goes about modifying parameters in the estimator, not in the model variables. There are two common techniques to do this: GridSearchCV and RandomizedSearchCV, the first one creates a grid with all parameters and performs k-fold validation for each point in the grid, while the second one does not try all hyperparameters (so it is less expensive); so considering the computer power I have access, I stick with the second. It would be great to test the first one with a GPU, which is available on Google's platform Colab. Below you can find a table with my final models and its MAEs.

Model Specification	**MAE**

Decision Tree Regressor WITHOUT hyperparameter tuning	21.98%
Decision Tree Regressor WITH hyperparameter tuning	18.12%
Random Forest Regressor WITHOUT hyperparameter tuning	15.68%
Random Forest Regressor WITH hyperparameter tuning	15.12%

References

1. W. Hillier, "What is a decision tree and how is it used?," 11-Aug-2020. [Online]. Available: https://careerfoundry.com/en/blog/data-analytics/what-is-a-decision-tree/#:~:text=Decision%20trees%20are%20extremely%20useful,%2C%20data%20classification%2C%20and%20regression. [Accessed: 08-Feb-2021].
2. scikit learn, "sklearn.preprocessing.LabelEncoder," *scikit*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html. [Accessed: 08-Feb-2021].
3. J. Brownlee, "Difference Between Classification and Regression in Machine Learning," Machine Learning Mastery, 21-May-2019. [Online]. Available: https://machinelearningmastery.com/classification-versus-regression-in-machine-learning/. [Accessed: 08-Feb-2021].