A logo for college computing

Description automatically generated

**Assessment Cover Page**

|  |  |
| --- | --- |
| *Student Full Name* | Marty Wickham |
| *Student Number* | sbs24069 |
| *Module Title* | Machine Learning |
| *Assessment Title* | CA1 |
| *Assessment Due Date* | 28/04/2024 |
| *Date of Submission* |  |

**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

[Introduction 1](#_Toc158384946)

[Chapter 1 1](#_Toc158384947)

[Chapter 1.1 1](#_Toc158384948)

[Chapter 1.1.1. 1](#_Toc158384949)

[References 2](#_Toc158384950)

# Machine Learning CA1

## Introduction:

For this project, I have chosen to develop and deploy two machine learning models that will be capable of accurately predicting property prices, based on universal features that are inherent in all properties. Ireland is currently in the middle of a serious housing crisis with house prices becoming increasingly more expensive, with “tens of thousands of young people locked out of the housing market” (Conneely, 2024). Many young people feeling like the dream of owning a home has never been so far out of reach. The overarching goal of this project will be to develop a model that will be capable of providing prospective homeowners with an increased measure of control in their housing purchases. Considering the unpredictable trends driven by external factors such as scarcity, inflation, and energy costs, this tool could provide valuable guidance for buyers and policy makers and an understanding of the valuations in the housing retail market. First-time buyers could benefit from a tool which could give them reasonably accurate estimates based on their individual requirements and preferences, offering insight into local area affordability, empower negotiation exchange and narrowing of the decision-making process based on means and or requirements. This tool could further guide those selling or renovating properties to choose their asking prices and renovation investments based on trends elucidated from a ML market valuation analysis. Predictive modelling techniques implementing prediction and classification algorithms have emerged as indispensable tools for navigating this complex domain, especially for the average individual outside of the housing industry. This project seeks to demonstrate the utility of these algorithms to ameliorate these challenges and provide actionable insights in the domain of property valuation and investment decision-making.

## Data Selection

To develop models capable of predicting property prices, I decided to use the “House Sales in King County, USA” dataset. This dataset contains house sale prices for King County, Seattle and includes homes sold between May 2014 and May 2015. My reasons for choosing this data are as follows:

* It is a dynamic dataset which provides a wide variety of property features to analyse.
* It is a very complete dataset that provides feature value for all observations.
* It is an exceptionally large dataset providing ample data on which to train the machine learning models.
* The dataset is well established as a reliable source evaluating for simple regression models ()

## Data Preprocessing

The dataset was imported as a Pandas data-frame object and analysed utilising the data-frame classes built in functionality. Firstly, I used the .head(20) method to give me a brief overview of what kind of data each feature contained, then I utilised the .describe() method to provide me with a numerical summary of all values contained in each feature. The .info() method allowed me to identify any categorical features and if they were any instances of NULL values. The results of this exploratory analysis revealed that the dataset is very clean containing no NULL values and only 1 categorical feature, date. Once I had identified that there were no NULL values to be cleaned or categorical features to be converted, I dropped the unnecessary features ‘id’ and ‘date’ from the data frame. After examining the data further, I noticed that there seemed to be a high number of 0 values in the column ‘yr\_renovated’. After performing a value count on this feature, I discovered that there were 20699 values of 0, so I decided to remove this column from the dataset as well. At this stage basic pre-processing of the data is complete and I could now move onto feature selection.

## Test-Train Split and Cross Validation

### Test-Train Split Sizes

Test-training split sizes play an important role in development and evaluation of machine learning models. Depending on the type of model being used it is important to experiment with different values for the training and testing split size, to ensure that the model can achieve the best performance possible. For instance, in some cases a lager training set can be important as it can provide the model with the required data to learn from effectively and achieve the desired performance and generalization. On the other hand, smaller training sets may result in underfitting, where the model has not learned well from the training data and is unable to acquire the underlying patterns in the training data resulting in high bias and poor performance. Overall, the most optimal test-training split used will depend on the specific nature of the dataset and the model deployed. In this project I have decide to test my algorithms over 4 different training splits of 15, 20, 25 and 30 percent. From my understanding, this should be a robust enough variation to ensure that my models can achieve optimal performance from a test-training split point of view.

### Test-Train Split Sizes Results

#### Linear Regression:

A screenshot of a computer code

Description automatically generated A screenshot of a white screen

Description automatically generated

#### kNN Regression:

A screenshot of a computer program

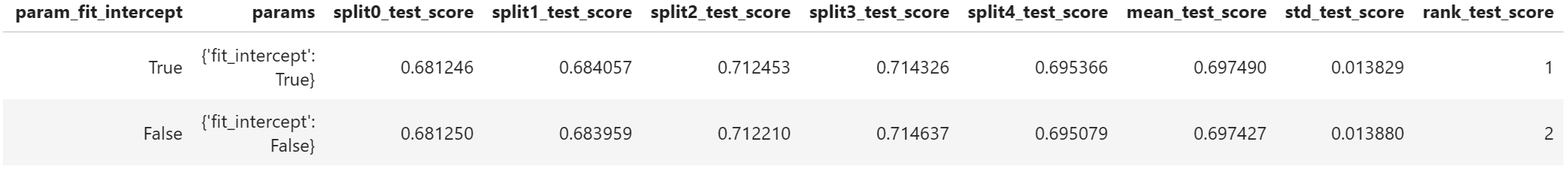
Description automatically generated

The results of my test-training splits show that the most optimum split values for the linear regression and kNN regression models are 25% and 15% respectively. Although comparing all of this results indicates that there was not a huge performance increase to be gained by using different splits on this data. I would speculate that this is due to the overall size of the dataset which contains over 21,000 observations and is more than enough data for either algorithm to be trained with.

In addition to this, I have employed cross validation within my model training splits. Cross validation is the concept of creating multiple different training sets which are compromised of variations in the selection of randomly chosen records for training splits. In the event where there are imbalances in the representation of features across the dataset, this can have an impact on model performance, as a random selection of data for a training split may result in the model learning patterns and trends that do not accurately represent the data.

Cross validation can help to mitigate the consequences of this, by creating variability in the distribution of observations within training subsets. For my linear regression model, I have opted to choose 8 cross validation splits. I believe this should be more than enough splits to account for any imbalances in the data as the dataset I have chosen is very well recorded and maintained with features being well represented across the dataset. For my kNN regression model I have opted only for 5 splits due to the performance overheads required by the model which result in much slower prediction times.

Linear regression cv splits:



The results of my splits in the linear regression model show a variation in accuracy between 68 – 71% which are a good indication that dataset used is well balanced.

## Hyperparameter Tuning

In machine learning models, hyperparameters are parameters which “express important properties of the model such as its complexity or how fast it should learn”. Hyperparameters can be thought of as configuration settings for a model which are set before the model is trained. They are ultimately responsible for controlling how the model learns from the data it is trained with. Some of examples this are the number of neighbours to search in a kNN regression, the kernel size in a support vector machine or the number of neurons in a layer of a neural network. Hyperparameter tuning, is the process of adjusting these configuration variables to identify the optimal hyperparameter values which will result in the best performance for a model (GeeksForGeeks, 2019).

For this project I have decided to use GridSearchCV as the method for controlling hyper parameter tuning. GridSearchCV is a hyperparameter tuning approach that allows a model developer to provide a specified list of relevant hyperparameters and values to be employed during model training. Using the specified list, GridSearchCV will fit the model using all possible combinations of hyperparameters in a systematic brute force approach. Each performance outcome of the model is recorded, and the model with the best performance will be chosen as the hyperparameters to be used.” The approach is called GridSearchCV, because it searches for the best set of hyperparameters from a grid of hyperparameters values” (GeeksForGeeks, 2019). Aside from the key advantage of producing the best accuracy for a model, another key advantage of the approach is that it naturally incorporates cross validation splits discussed previously, with a default value of 5 cv splits if another is not specified. Although this approach will identify the best performance for a model, a primary disadvantage of the approach is the time and space complexity of the search method. As an exhaustive approach is employed this increase the computational overhead required to run the search, which may be a limiting factor. As an outcome of this, devices with less computational power may result in longer times for results.

## Results Interpretation

### Linear Regression

An R-squared score of 0.7 on the training data and 0.69 on the test data indicates that linear regression model explains approximately 70% and 69% of the variance in the target variable in the training and test data respectively. A higher R-squared value indicates that the model explains a larger proportion of the variance in the target variable. As both the training and test R-squared values are relatively high, and the difference is small, it suggests that the model is not significantly overfitting or underfitting. This suggests that the linear regression model provides a decent fit to the data and indicates that the model generalizes well to unseen data. If training R-squared was much higher than the test R-squared scores, this would suggest overfitting. If both scores were low this would suggest underfitting.

While the R-squared value of the model was relatively good, other metrics for the model such as the mean absolute error and the root mean squared error are very high at 126863.168 and 210130.343 respectively. This could indicate that the models’ predictions are not as accurate as the R-squared value would suggest. These results could possibly be explained by a significant number of outliers in the data. For example, the maximum price value in the dataset is 7,700,000 which is extremely above the mean value of

### kNN Regression

# References