# Feb 2023 – SB+ - MSc in Data Analytics

Author: Sajjan Bhattarai

e-mail: [sbs23010@student.cct.ie](mailto:sbs23010@student.cct.ie)

Student ID: sbs23010

# Abstract

*This report presents an analysis of the relationship between average prices of new houses and economic factors across Ireland. It is based on the publicly available datasets from Department of Housing, Local Government and Heritage. The study utilizes Statistical Probability Distribution and Linear Regression models to explore the significance of numerous factors such as housing supply, planning permissions, housing construction cost index, consumer price index (CPI), mortgage interest rate index, unemployment rate, Gross National Product (GNP), and so on. The results show that construction cost, mortgage rate, unemployment, CPI and GNP are the prominent predictors of new house prices. Whereas other factors such as housing supply and planning permissions showed to have weaker impact than thought initially. The study aims to contribute to the literature on predicting and planning housing market in Ireland. Limitations and directions for further research are also discussed.*

Table of Contents

[Feb 2023 – SB+ - MSc in Data Analytics 1](#_Toc132402695)

[Abstract 2](#_Toc132402696)

[Introduction 4](#_Toc132402697)

[Materials and Methods 5](#_Toc132402698)

[Selection of Programming Language 5](#_Toc132402699)

[Project Management Framework 6](#_Toc132402700)

[Research Understanding 6](#_Toc132402701)

[Data Understanding 7](#_Toc132402702)

[Data Preparation 9](#_Toc132402703)

[Modeling 12](#_Toc132402704)

[Evolution 18](#_Toc132402705)

[Deployment 18](#_Toc132402706)

[Results 19](#_Toc132402707)

[Descriptive Statistics 19](#_Toc132402708)

[Exploratory Data Analysis 19](#_Toc132402709)

[Model Development 19](#_Toc132402710)

[Discussion/Conclusions 21](#_Toc132402711)

[References 22](#_Toc132402712)

# Introduction

The housing market refers to the buying and selling of residential properties such as houses and apartments. The condition of housing market can have a significant impact on the overall economy, and standard of living for normal people (Organization for Economic Co-operation and Development, 2023). One key factor affecting house prices is supply and demand. Additionally, other macro-economic factors such as inflation, Gross National Product (GNP), Per Capita Income and so on also impact housing prices. Conversely, since the large proportion of total economic asset is held in housing market, dramatic changes in this sector can have considerable effect on the consumer behaviour and overall economic outlook (Kenny G., 1998, p. 2).

Utilising Statistics, Data Analytics and Machine Learning to analyse Housing related datasets can be a good application. Such approach helps to identify or reaffirm the relationships between different economic factors and housing prices. Machine Learning model such as Regression and Classification can be useful to predict house prices and determine affordability respectively.

# Materials and Methods

## Selection of Programming Language

In today’s world, there are dozens if not hundreds of programming languages, each having their own appeal for different use cases. These languages can be classified in several ways based on the criteria such as paradigm, syntax, abstraction, domain, popularity and so on. It is important to evaluate the use case against these criteria before selecting the programming language(s) for a project.

For this study, 4 programming languages were considered initially: Python, JavaScript, PHP and Java. The primary reason being the prior exposure and experience of working with them. After a preliminary analysis, the later 3 languages were eliminated because a common disadvantage with them is the lack of relevant libraries for data analytics and machine learning. Additionally, in case of JavaScript and PHP, they are mainly useful for web-application development as client-side and server-side scripting respectively (Wellens, 2015). Even though these languages might not be well-suited for data science projects, these can still be supplementary towards overall data analytics projects by aiding to build interactive web visualization, dashboards, etc.

After further research, 2 programming languages were shortlisted: Python and R. Python being currently one of the most widely used programming languages for data science, and is known for its simplicity, versatility, and collection of wide range of useful libraries. Some of the most popular data science libraries like Pandas, PySpark, NumPy, Matplotlib, Scikit-learn, Seaborn, etc. are the testament to Python’s usefulness (Navlani, Fandango and Idris, 2021, p. 7). On the other hand, R is yet another one of the most popular programming languages in the field of computing statistics, data visualization, and data science. As more companies are becoming data-driven, the user base of R is also expanding fast and is already supported by over 2 million users globally (Koushik and Ravindran, 2016, p. 1). R also has a rich ecosystem of libraries and tools for data science. Some of the examples include ggplot2, dplyr, tidyr, caret, purrr, etc. Similar to Python, R can also be used with notebook applications like Jupyter and Zeppelin.

Both Python and R have significant level of resemblance when compared against their paradigm, abstraction, domain, and popularity. In regards to syntax, Python has slight edge among beginners because of its more straightforward and readable syntax. Regarding domain and interoperability, R is more specialized for statistical analysis and has superior data manipulation/exploratory tools. Whereas Python has better interoperability with other languages and tools. When it comes to data collection from web as in this study, Python’s *requests* library allows easier access. In terms of data modeling also, Python has standard libraries for the purpose, whereas R lacks behind. That said, R is better positioned when performing exploration and visualization on large datasets (IBM Cloud Team, 2021).

At the end of the research, Python was selected as the desired programming language for this study. The main reasons for this decision were the prior experience, commonality in academic study, exposure to relevant libraries/tools, and overall time constraint. On the technical level, having easily readable syntax, ability to read data from web, and availability of encoding, scaling, and modeling libraries added more points for Python.

After selecting Python as the desired programming language, couple of Python libraries were also considered: Pandas and PySpark. Pandas provides Structured tabular data for analysis, and is heavily utilized in Data Analytics/Science, and Machine Learning projects. However, it can be limited by its dependence on a single machine, and may not be well-suited for analyzing very large datasets (SparkByExamples, 2023). PySpark on the other hand is a Python based library built on top of Apache Spark, and leverages distributed system. Unlike the Pandas Dataframe, PySpark Dataframe is a wrapper on Resilient Distributed Dataset (RDD) and considered to be fault-tolerant. These attributes make PySpark better suited for large datasets and Streaming workloads. That being said, for the scope of this analysis, Pandas will be more than enough and is more applicable due to its ease of use and deployment.

## Project Management Framework

In this study, 2 common frameworks were considered:

* Cross Industry Standard Process (CRISP-DM)  
  It was developed by a consortium of many companies involved in data mining, and is the most widely adopted framework for developing data mining solutions (Kotu and Deshpande, 2014). It involves iterative phases for business understanding, data understanding, modeling phase, evolution phase and deployment phase. A real-life application can be an e-commerce company seeking to develop a recommendation system to suggest relevant products to customers based on their purchase history. The project would start with understanding of business requirements and objectives. It would then have phases to gather and understand data that represent previous purchase details. Next phases would involve building, training and testing the predictive Models, and evolving them till deployment.
* Sample, Explore, Modify, Model and Assess (SEMMA)  
  It was developed by Statistical Analysis System (SAS) and is similar to CRISP-DM in many ways. It contains 5 iterative phases: sample, explore, modify, model and assess. A credit card company building a model for Fraud Detection can be a real-life example of this.

CRISP-DM was selected for this study because it provides a more comprehensive framework for the entire data mining/analysis process, from understanding the business or research requirements to final deployment of the model (Ghavami, 2019). Each of its phases provide guidance on specific tasks and deliverables, which are required for a successful project. Whereas SEMMA doesn’t cover the business understanding part, and focuses more on the modeling.

### Research Understanding

This study is based on the publicly available datasets from Department of Housing, Local Government and Heritage (Government of Ireland, 2023). As part of the data exploration, datasets containing below information were reviewed:

* Average house prices
* Annual manual information
* Supply of housing land
* Planned permissions
* House registrations
* Mortgage loan approvals
* Unemployment rates
* National economic indicators like GDP and GNP
* Financial interest rates

The objectives of this research have been to understand the relationship between housing market and macro-economic factors, and to come up with a model to predict or forecast the outcome.

### Data Understanding

The initial summarization of the above datasets can be found in Jupyter Notebook (data-exploration-notebook.ipynb). Some of the key takeaways from the data exploration were:

1. All these datasets represent historical information i.e. observations over years, quarters or even months in some cases.
2. They also have varying coverage of period. For instance, average house prices dataset contains records from 1975 to 2015. Whereas another dataset for supply of housing lands contains records from 2000 to 2012 only. Similar discrepancies can be observed in other datasets as well.
3. The datasets also have varying degree of granularity and size. Some datasets are divided into areas, quarters/months, agency, and so on, thus allowing them to have bigger size. On the other hand, some datasets only contain statistics on national and annual level.

#### Discrete Distribution

One of the datasets contains home mortgage loan approvals for new houses and other houses. In other words, for every loan approval in this dataset, it is either for new house or other house type. In total, there were approximately 42.8% of loan approvals for purchasing new houses. Then, defining X to be 1 if the loan approval is for new house and 0 if it is for other house, the X would be a Bernouli(0.43) random variable. The expected value and variance for X would be 0.43 and 0.43(1-0.43)=0.245, respectively.

Below diagram shows the number of loan approvals for new and other houses over the recorded years. There are some outliers present in the data as 3 of the years had exceptionally high loan approvals for new houses.

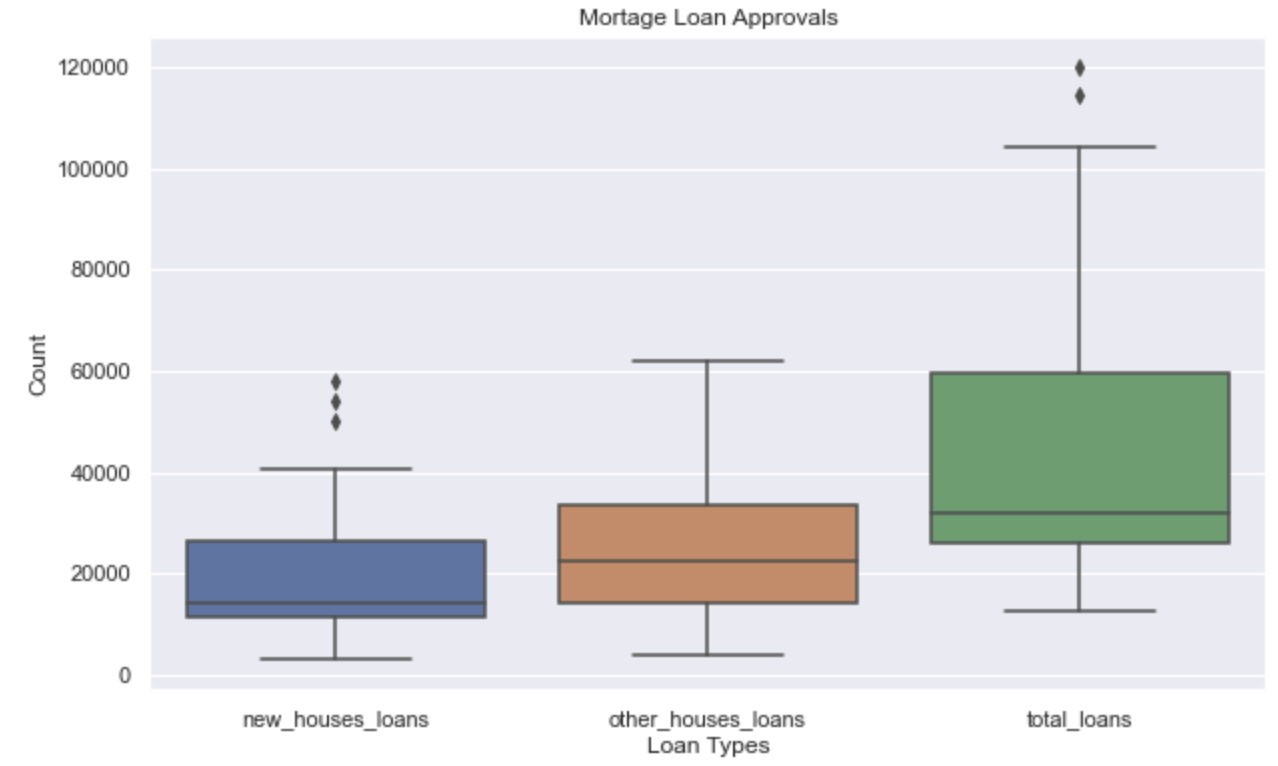


Figure 1: Mortgage Loan Approvals by House Types

Binomial distribution can also be used to model n independent and identical Bernouli trials, each having probability p of success that is picking loan approval for new house (Wang, 2011, p. 11,12). If X is a binomial(n,p) random variable, then X is the number of successes out of a total of n trials. Supposing that each of the next 10 loan approvals has probability 0.43 of being for new house, the X would be binomial(10, 0.43) random variable. Similarly, for 20 loan approvals, X would be binomial(20, 0.43). For reference, below plots show the Binomial probability distribution when using sample size of 10 and 20 respectively.

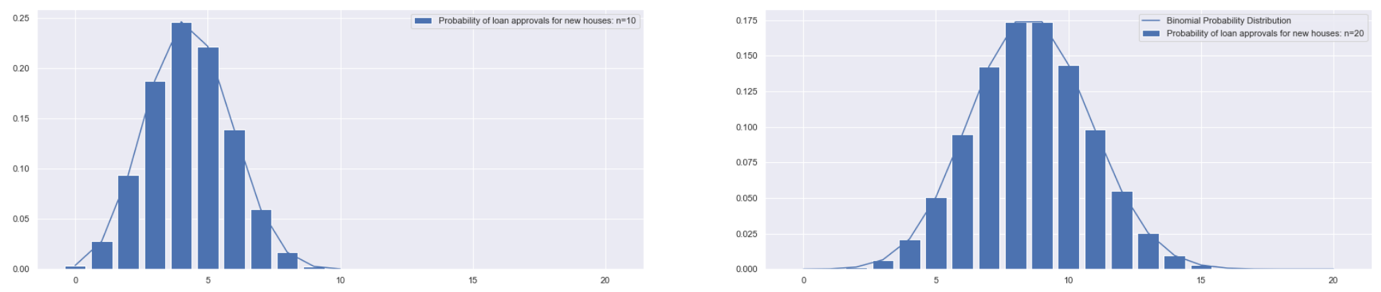


Figure 2: Probability of Binomial Distribution

Taking **n** as 20, following probabilities of exact number of successes can be calculated:

* **Probability of exact 5 loan approvals for new houses:** 0.05057989119846802
* **Probability of exact 10 loan approvals for new houses:** 0.14350071462048797

#### Approximation using Normal Distribution

Supposing that a random variable X is a binomial(n,p), X can be approximated by normal distribution if n is large and p is not near 0 or 1. A common rule of thumb is to ensure that both np and n(1-p) are greater than 5. In order to get better approximation, added 0.5 to x or subtracted 0.5 from x, where number 0.5 is called the continuity correction factor (OpenStax, 2020). Taking **n** as 20, following probabilities can be calculated using Binomial and Normal distribution:

* **Probability of less than equal to 12 loan approvals for new houses:**

**Binomial**: 0.9613247396908604

**Normal**: 0.9619347204273013

* **Probability of more than 12 loan approvals for new houses:**

**Binomial**: 0.0386752603091396

**Normal**: 0.03806527957269866

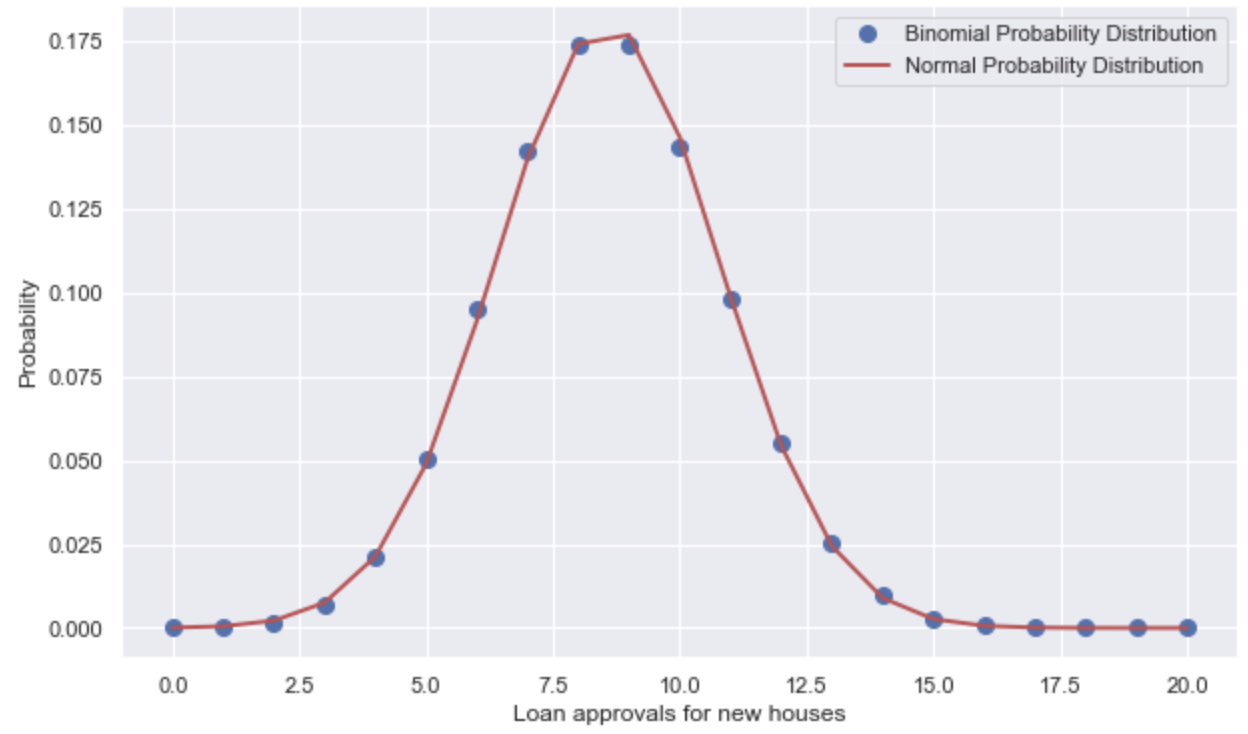


Figure 3: Probability of Binomial and Normal Distributions

### 

### Data Preparation

###### Loading Data

After the desired datasets were identified in data.gov.ie website, they were loaded into Pandas Dataframes using *read\_csv()* method. Initial approach was to download the files locally and then read them from Pandas. As the study progressed, more datasets were explored, so took an approach of directly reading from the web URL without downloading. While reading different datasets, main challenges were to parse the header lines and encoding correctly.

###### Exploring and Cleaning Data

With the Dataframe, several exploratory or describe methods were performed such as *shape* to display number of rows/columns, *head()* or *tail()* or *iloc[[0,-1]]* to display corresponding rows, *info()* to display columns and data-types, *isnull().sum()* to check null values, *duplicated().sum()* to display duplicates, *describe()* to get statistics, and so on. These methods provided insight into size, layout and any immediate concern within the data.

For reference, the loan approval dataset *loan\_approvals\_df* can be taken as an example.It had an empty last row, which was addressed by slicing the dataframe *loan\_approvals\_df[:-1].* It also had a multi-indexed schema and partially readable feature names, which was addressed by defining a list of readable feature names and setting *loan\_approvals\_df.columns* to it. Similarly, its features had incorrect datatypes and some had comma in thousand place, which was corrected by utilizing *astype()* and *replace(‘,’, ‘’)* methods. On the statistics side, there were considerable differences between Mean and 50th percentile (Median), and between 75th percentile and Maximum for *new\_houses\_loans*. This implied that there were both skew and outliers present for this feature. Using *skew()* method returned skewness of 1.32, which represents a positively skewed or right-skewed distribution (Greco, 2021). And to identify outliers, methods like Interquartile Range (IQR) or z-score were used which returned 3 outliers. The above Figure 1 also confirms the same visually. Normally, such skewness and outliers in a dataset can be balanced using transformation techniques like Box-Cox, Logarithmic, Outlier removal, etc. However, for this particular dataset, the columns are closely related as in *new house loans + other house loans = total house loans*. Hence, transforming the data wasn’t preferred, and decided to continue using it as is.

Additionally, other methods used during data preparation were:

* groupby() for grouping data by desired columns
* *merge()* for joining Dataframes
* *set\_index()* to set custom index and *reset\_index()* to reset index to default
* *transpose()* to reverse the position of columns and rows
* *melt()* to massage a Dataframe into a format
* *numpy.arange()* to generate a NumPy array of range
* *rename()* to change the column names
* *drop()* to remove the column(s) or row(s) using axis flag
* *append()* or *insert()* items into a list

Preparing a dataset for House Registrations *total\_house\_registration\_df* was an interesting task. This dataset has multi-indexed columns with years as first layer and null/quarter/months in second layer. Such layout occurred because the data was collected in yearly basis in the starting years, then in quarterly basis and finally in monthly basis in recent years. Similarly, the areas/counties are listed as part of rows. The objective was to fetch the annual house registrations for the national level. Therefore, the first action taken on this was to create a list of columns that represented ‘County’ and ‘TOTAL’ values. Then, a new Dataframe was created by selecting this list of columns, and dropped the last empty line. Next, a NumPy array of years from 1978 to 2015 was created and set the columns to it because it was known that the data belonged to these years. These actions significantly simplified the data preparation because the resulting Dataframe only contained the yearly total values for each county. Lastly, a series of methods including *transpose(), reset\_index(), rename(), astype(), replace()* and Dataframe slicing were applied to prepare the final version that could be merged with other datasets.

As it happens with all real-life datasets, the collected datasets here also have differences in their structure, data types, granularity of features, duration of observations, unit of measurement, and so on. Therefore, preparing each of them required a separate set of techniques and methods as per the overall objective. Some set of methods were common to all of them, while some actions were applied specifically to particular datasets. Most of the datasets needed to be sliced on columns, while some datasets needed to be sliced on rows after checking the uniqueness of values using *unique()* method. An example of the later was applied on the dataset for National Economic Statistics *gdp*\_df. The Gross Domestic Product (GNP) at constant and Gross National Product (GNP) at constant were of interest to the study (Central Statistics Office, 2023). Hence, *unique()* method was used to get all the statistic labels in the dataset, and then *str.contains()* method was used to filter the observations containing the desired statistics. Provided that the dataset contained quarterly data, the *TLIST(Q1)* column was divided by 10 and stored the quotient as *year* column. Next, the data was grouped by columns *Statistic Label* and *year* to get the average value of *VALUE*. A visualization was also created using Seaborn to display the trend of GDP and GNP across the years.

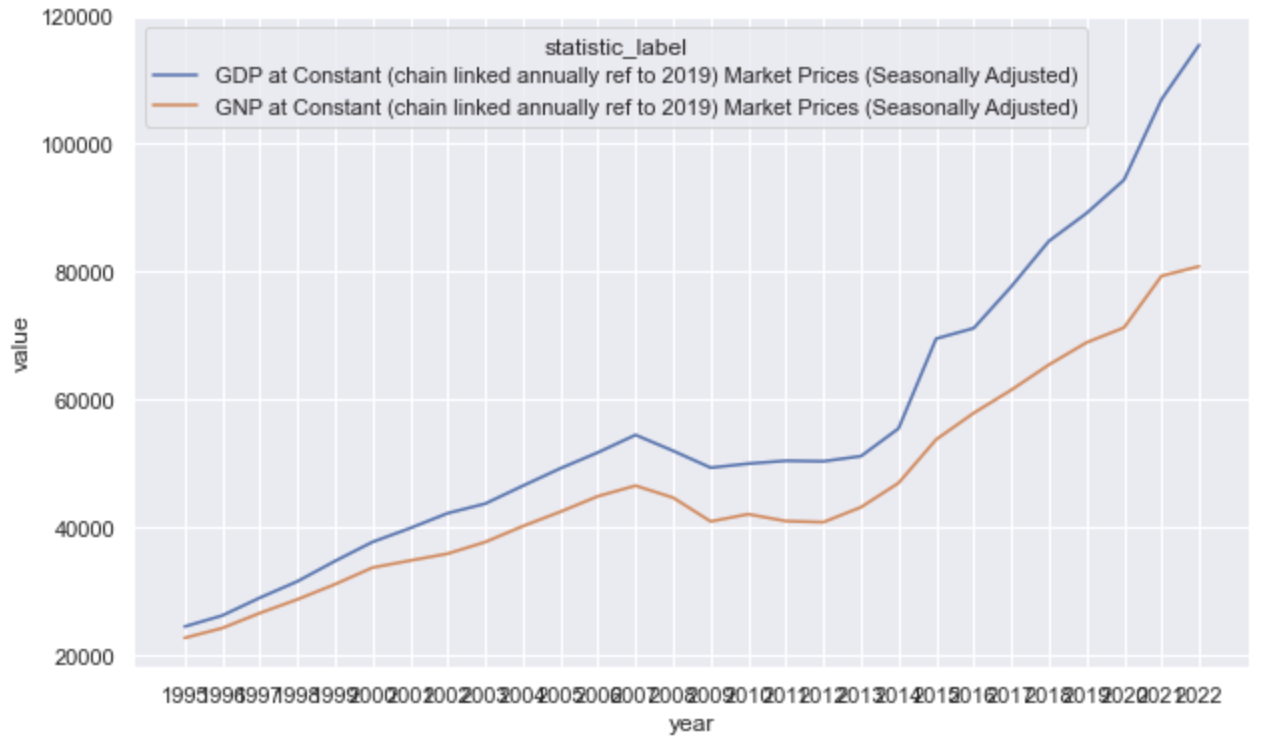


Figure 4: GDP and GNP of Ireland between 1995 and 2022

About visualization, dataset for Annual Market Information Indices *annual\_market\_information\_indices* is another example. It includes index values for some of the key market factors from 1990 to 2015. It provides insight into how factors affecting the housing market are evolving over the years, and can be considered a very useful information in further modeling. Below visualization was created using Seaborn. Same could be created using Matplotlib pyplot’s *plot()* method as well, however with bit more lines of code. The Dataframe was massaged into a format using *melt()* in order to facilitate classification of index values with Seaborn’s *hue* parameter. Line chart was utilized in these scenarios because it is easier to display the trend over the years. Other charts like Grouped Bar charts were also considered, though they could make the visualization more cluttered and hard to understand.



Figure 5: Annual national economic indices

###### Combining the Prepared Datasets

The final task during data preparation phase was to combine all the datasets together into a common dataset. This was done using Pandas *merge()* method. All the datasets have common column *year*, which was converted to *string* data-type. Using *str* type for *year* also prevented it from being interpreted by *describe()* method. For most of the datasets, default type of *inner* *merge* was used. Whereas for a dataset that had partial overlap on the join column values, *outer merge* was performed that resulted in some *NaN* values. Similar to initial data exploration, descriptive methods like *describe(), isnull()* and *duplicated()* were applied on the final dataset for high-level insight. It showed that there were some null values for columns *avg\_earnings\_of\_adult\_workers, permissions, permission\_percent* and *total\_decisions*. These will be taken into consideration during feature selection during Modeling phase.

### Modeling

Given the nature of available datasets and observed constraints, performing a prediction on average house prices was decided to be an area of interest for Machine Learning. Since most of the collected independent features are quantitative and the target feature *avg\_price* is also quantitative, it was decided to be a Regression problem and corresponding models were chosen (Krishna Kumar, Arvind, and Dharmendra Singh, 2020). The fact that the target variable was available in the dataset, Supervised Learning was the desired technique.

###### Feature Selection

To proceed with Regression Models, the first requirement was to select the desirable features from the finalized dataset. There are several ways to do this such as Correlation-based selection, Wrapper-based selection, Filter-based selection and Recursive Feature selection (Fagroud et al., 2022). For the purpose of this study, Recursive feature selection and Correlation were utilized. Initially recursive approach was taken, model tests were performed by removing the features and evaluating the results in each iteration. Later, the correctness of the selected features was double-checked using Correlation-based selection.

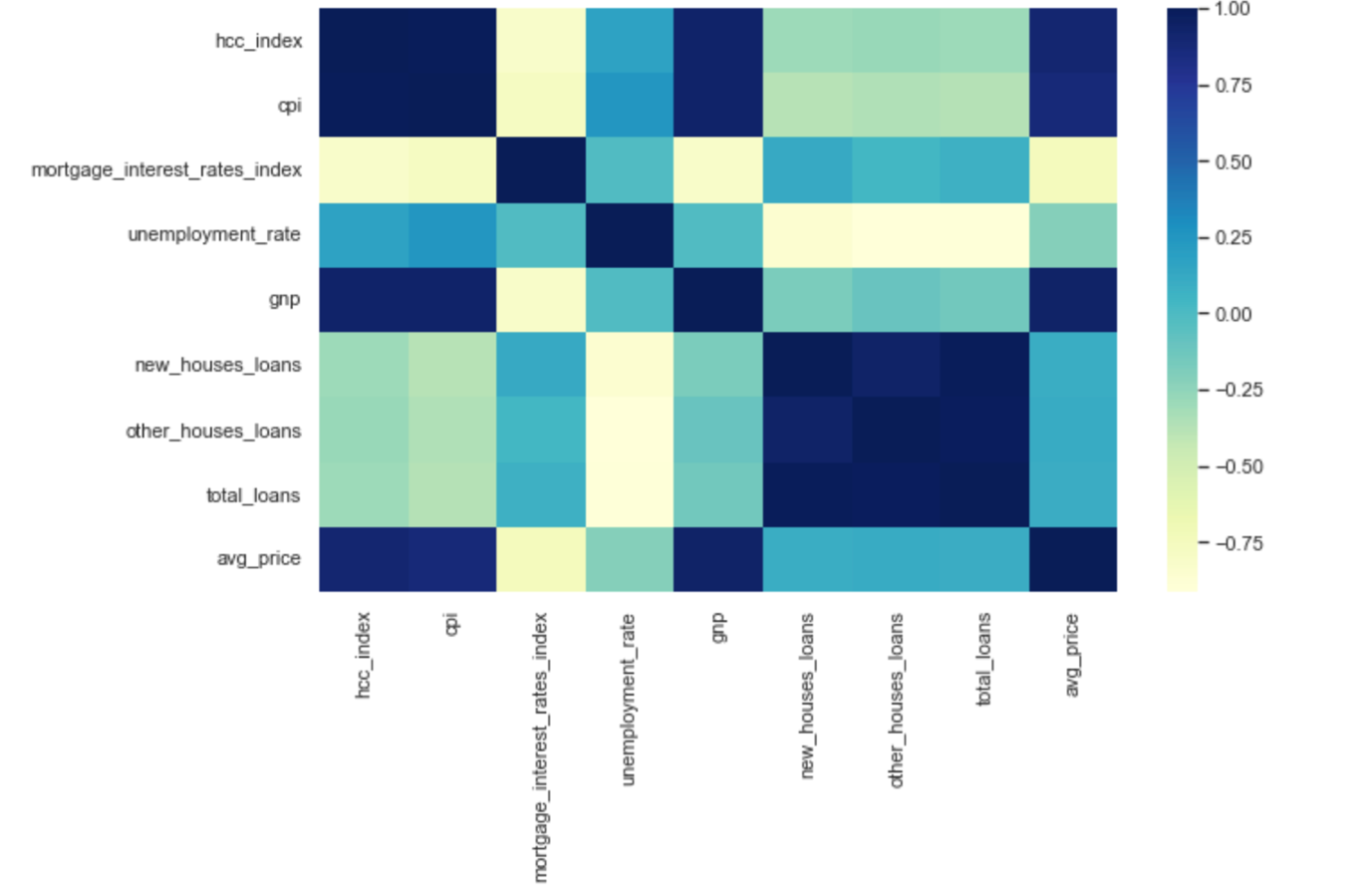


Figure 6: Correlation Matrix showing Features

This resulted in the selection of 5 features: *hcc\_index, cpi, mortgage\_interest\_rates\_index, unemployment\_rate, and gnp*. These map to Housing Construction Cost index, Consumer Price Index, Mortgage Interest Rates Index, Unemployment Rate and Gross National Product respectively. While performing the feature selection, it was also learned that *corr()* method and ***LinearRegression Model*** don’t work on null values. Therefore, it was decided to not utilize several of the missing features such as *permissions, permission\_percent, avg\_earnings\_of\_adult\_workers* and *total\_decisions*. Additionally, another feature *new\_house\_price\_index* was also not used because it is basically another form of average price i.e. target variable and misled prediction to 100% accuracy.

###### Data Scaling

Among the selected features, most of them are indices that are in the range of hundreds. The unemployment rate is in the range of single digits. And the GNP is in the scale of tens of thousands. Therefore, it can be beneficial to transform the features into a common scale for the Modeling performance, convergence and to avoid bias towards any particular feature(s) (Brownlee, 2020). In the context of this study, 2 scalers will be considered:

* MinMaxScaler:

Default scale between 0 and 1. It is more applicable when the range of the data is important. In other words, it is a normalization technique.

* StandardScaler:

Default scale to have mean as 0.0 and divide by the standard deviation to give the standard deviation of 1. It is well-suited when the mean and variance of the data are important. In other words, it is standardization technique.

Depending on the nature of dataset and experimentation results, none or either or both of these techniques can be utilized. Depending on the use case, it may also be desirable to normalize data after it has been standardized.

###### Regression Models

Linear Regression

For the purpose of this study, Linear Regression Model aka Ordinary Least Square (OLS was the first approach because there exists a linear relationship between the dependent and independent variables or predictors. It is a simple and commonly used method for regression analysis. This model can be understood as a line running through the feature space that minimizes the distance between the line and the data points. By looking at the coefficients of each feature, it can be identified how much the predicted value for target would change for each unit change in the predictor. On the other hand, Linear Regression models can only capture the linear relationships, while the real-life scenarios are generally more complex. Additionally, linear models aren’t good at capturing the interactions between variables easily (Blanchard, Behera and Bhatnagar, 2019).

***LinearRegression()*** class provided from Python *sklearn.linear\_model* module was used in this analysis. Using this model with default parameters on the dataset, the coefficients, model intercept, training/test set scores and prediction residuals were calculated.

|  |  |
| --- | --- |
| **Feature** | **Coefficient** |
| **hcc\_index** | 283.50130154249683 |
| **cpi** | 2263.539826690218 |
| **mortgage\_interest\_rates\_index** | 315.81102167871484 |
| **unemployment\_rate** | -8072.3573422923955 |
| **gnp** | 2.2171269681230115 |
| **Model Intercept** | -198617.4886957663 |

And the training set score and test set scores were 0.951 and 0.954 respectively.

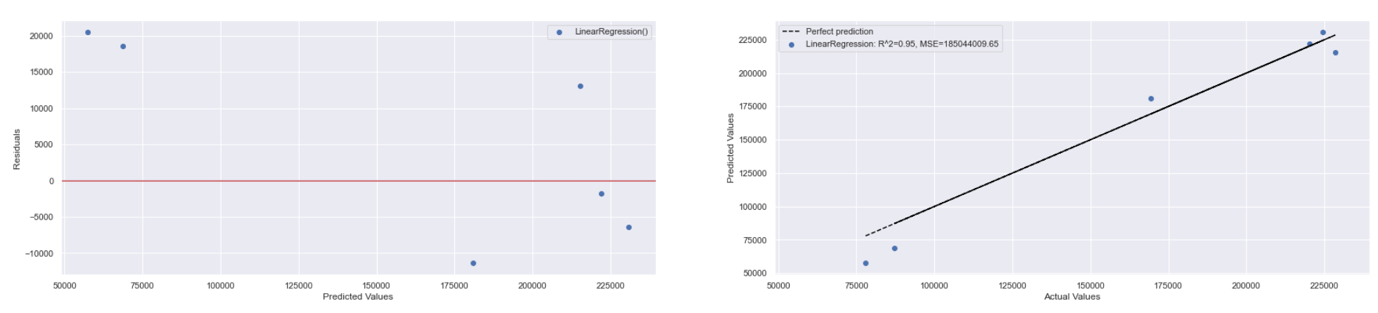


Figure 7: Prediction Results of Linear Regression

This is already a decent training and testing score. However, prior to arriving at this result, this model was returning much poorer or underfitting results because the feature selection wasn’t performed and the *inner merge()* on all datasets was resulting in much fewer observations. After selecting the preferred features through recursive tests and Correlation-based selection, the remaining features are more related and the observations also increased a bit.

Ridge Regression

Ridge Regression or L2 Regularization is a linear regression model that uses the coefficients not just for training the data, but also to fit an additional constraint. It adds “squared magnitude” of coefficient as penalty term to the loss function. The alpha parameter can be used to balance between simplicity and training set performance.

RidgeRegression() class present in *sklearn.linear\_model* module was used and tested with different values for alpha.

|  |  |
| --- | --- |
| **Feature** | **Coefficient** |
| **hcc\_index** | 371.73045353157175 |
| **cpi** | 2083.4828261620855 |
| **mortgage\_interest\_rates\_index** | 352.563543884394 |
| **unemployment\_rate** | -7890.477457577715 |
| **gnp** | 2.353134125739427 |
| **Model Intercept** | -195793.09995994548 |

Lasso Regression

Lasso (Least Absolute Shrinkage and Selection Operator) Regression or L1 Regularization is yet another method of regularizing a linear regression. It restricts coefficients to be near zero, and can consequently cause some coefficients to become exactly zero. Such approach can reveal the most important features of the model, and can completely ignore some features. It adds “absolute value of magnitude” of coefficient as penalty term to the loss function.

LassoRegression() class present in *sklearn.linear\_model* module was used and tested with different values for alpha.

|  |  |
| --- | --- |
| **Feature** | **Coefficient** |
| **hcc\_index** | 283.7408479965949 |
| **cpi** | 2263.0483615517596 |
| **mortgage\_interest\_rates\_index** | 315.890974008202 |
| **unemployment\_rate** | -8071.898463787299 |
| **gnp** | 2.217483884978038 |
| **Model Intercept** | -198607.8947310528 |

###### Comparing the Modeling Results

Below diagram represents the prediction residuals in first chart, and predicted vs actual values on the second chart.

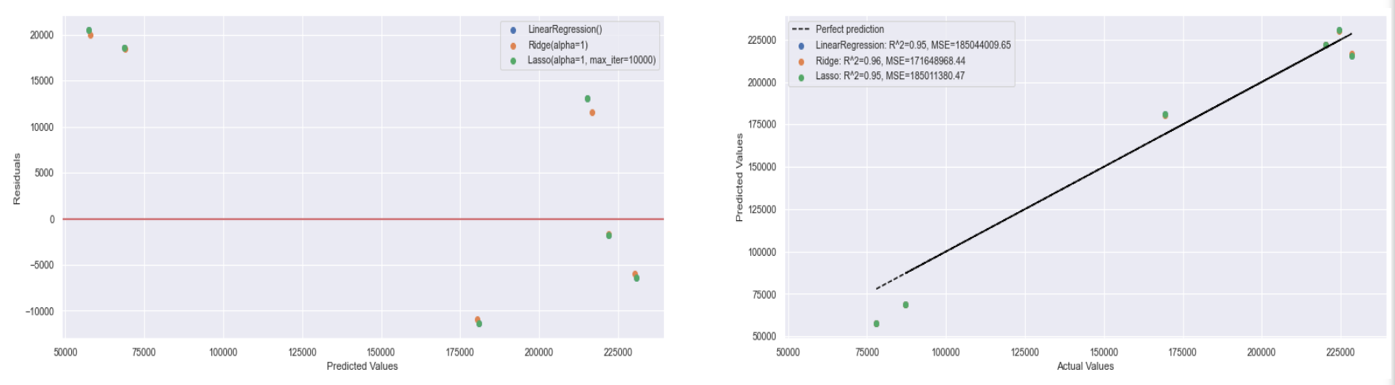


Figure 8: Prediction Results of Different Linear Regression Models

And following are the training and testing scores of above models with default parameters:

| **Model** | **Training Score** | **Test Score** |
| --- | --- | --- |
| LinearRegression() | 0.950895 | 0.954067 |
| Ridge(alpha=1) | 0.950866 | 0.957392 |
| Lasso(alpha=1, max\_iter=10000) | 0.950895 | 0.954075 |

It shows almost identical results from all 3 linear models.

Similarly, below represents the Modeling results after applying MinMaxScaler and StandardScaler on the dataset:

| **Scaler** | **Model** | **Training Score** | **Test Score** |
| --- | --- | --- | --- |
| MinMaxScaler() | LinearRegression() | 0.950895 | 0.954067 |
| MinMaxScaler() | Ridge(alpha=1) | 0.878361 | 0.940361 |
| MinMaxScaler() | Lasso(alpha=1, max\_iter=10000) | 0.950895 | 0.954155 |
| StandardScaler() | LinearRegression() | 0.950895 | 0.954067 |
| StandardScaler() | Ridge(alpha=1) | 0.947555 | 0.978712 |
| StandardScaler() | Lasso(alpha=1, max\_iter=10000) | 0.950895 | 0.954107 |

As seen in the above table, the training/testing scores remained unchanged for the most part. However, some models, mainly Ridge showed degraded training score when scaling or normalizing the data with MinMaxScaler. Whereas Ridge returned improvement in test score when scaling or standardizing the data with StandardScaler.

###### Cross-Validation and Hyper-parameter Tuning

It is important to check and verify the problem of selecting wrong test sets while training and testing a model. Cross-validation (CV) is a valid method for such purpose (Bonaccorso, 2020). It is a resampling technique that involves splitting the data into multiple chunks or folds, and using each fold as a test set while training the model on the remaining folds. K-Fold is one of the common cross-validation techniques for linear regression. Similarly, tools like GridSearchCV provides a combined functionality of cross-validation and hyper-parameter tuning. It allows training and evaluation of the model on different hyper-parameter settings for each fold. It then returns the best estimator or hyper-parameters based on the average performance across all folds.

GridSearchCV() class was imported from *sklearn.model\_selection* module, and tested with all 3 models. For LinearRegression, it showed that the best hyper-parameter was *normalize=True*. The best hyper-parameters for Ridge and Lasoo were *alpha=0.1* and *alpha=1000* respectively.

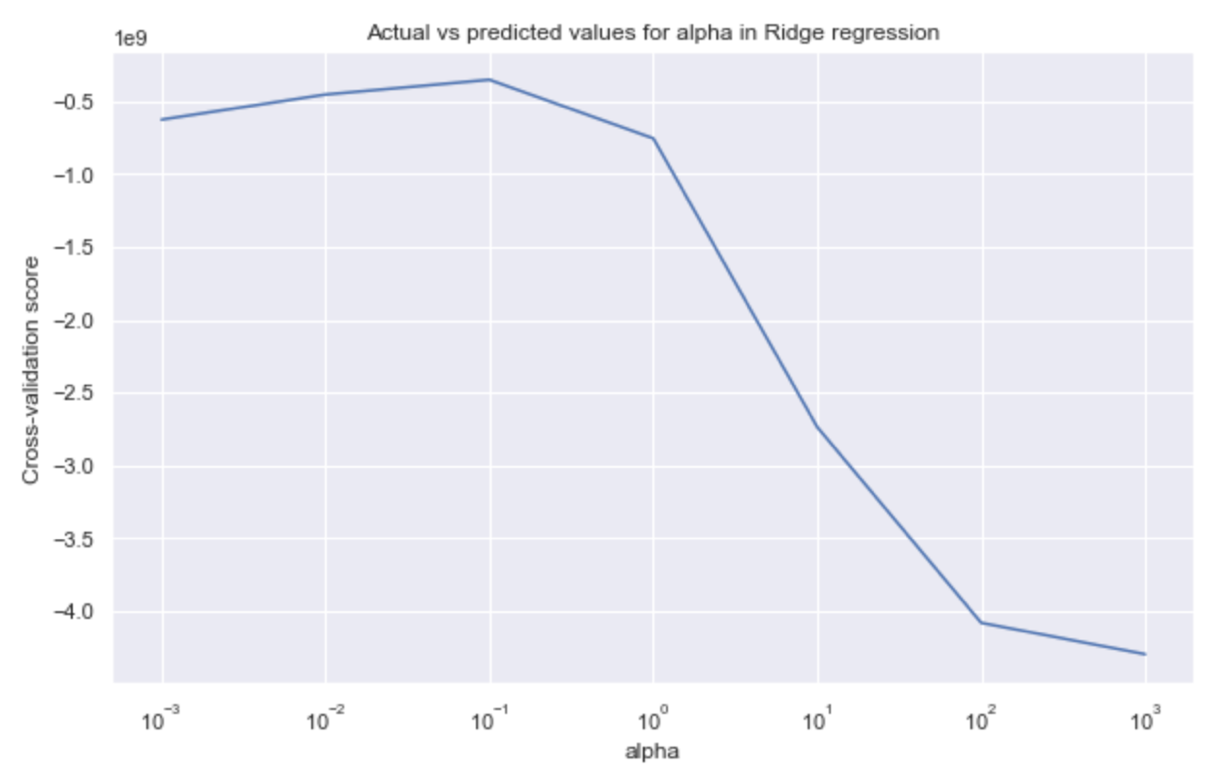


Figure 9: Cross-validation of Ridge for alpha values

Above graph shows how the cross-validation scores or mean test scores were changing for different alpha values. For this particular model, using *alpha=0.1* was the best parameter. Below diagram shows the prediction results of default model parameters and the best estimator/hyper-parameter returned by GridSearchCV on the scaled data with MinMaxScaler:



Figure 10: Prediction Results of Models with Default and Best Hyper-parameters

The model prediction results verified the outcome of GridSearchCV. There was slight improvement in Ridge and Lasso Model’s accuracy when using the best hyper-parameters. Whereas for the LinearRegression, the results remained unchanged even when using the best hyper-parameters. Below table outlines the modeling scores:

| **Scaler** | **Model** | **Training Score** | **Test Score** |
| --- | --- | --- | --- |
| **With Default parameters** | | | |
| MinMaxScaler() | LinearRegression() | 0.950895 | 0.954067 |
| MinMaxScaler() | Ridge(alpha=1) | 0.878361 | 0.940361 |
| MinMaxScaler() | Lasso(alpha=1, max\_iter=10000) | 0.950895 | 0.954155 |
| **With best hyper-parameters returned by GridSearchCV** | | | |
| MinMaxScaler() | LinearRegression(normalize=True) | 0.950895 | 0.954067 |
| MinMaxScaler() | Ridge(alpha=0.1) | 0.948004 | 0.977470 |
| MinMaxScaler() | Lasso(alpha=1000.0, max\_iter=10000) | 0.942553 | 0.98432 |

### Evolution

Based on the learning from Data Preparation and Modeling phases, the developed model(s) were further evaluated using another target *inflation\_adjusted\_price* in lack of additional data. It was derived from the previous target *avg\_price* and *cpi*. In a way, this can be considered a more reliable indicator of how the prices were evolving over the years. When testing the model against this target, the training set score reduced from 0.95 to 0.91, while the test score increased from 0.95 to 0.98.

Additionally, while performing iterative experiments during modeling phase, there were too many redundant lines of code and blocks of code and outputs. After few experiments, it became increasing difficult to effectively write code with different configuration, evaluate changes and tune/optimize the model. Therefore, the code blocks were rewritten into Python functions that could take multiple arguments, and execute the code conditionally. This approach made it more efficient and time-effective when making a series of experiments with different parameters and tunings.

### Deployment

After collecting the model scores for all the major tests, the highest scoring combination of model, target and scaler were found to be:

**For combined highest training and test score:**

| **Target** | **Scaler** | **Model** | **Training Score** | **Test Score** |
| --- | --- | --- | --- | --- |
| avg\_price | MinMaxScaler() | LinearRegression(normalize=True) | 0.950895 | 0.954067 |

**For highest test score:**

| **Target** | **Scaler** | **Model** | **Training Score** | **Test Score** |
| --- | --- | --- | --- | --- |
| inflation\_adjusted\_price | StandardScaler() | Lasso(alpha=1000.0, max\_iter=10000) | 0.917561 | 0.993352 |

Based on the above outcome, LinearRegression with *normalize=True* on a scaled data by MinMaxScaler can be considered the best performing model and deployed accordingly.

# 

# Results

## Descriptive Statistics

The finalized dataset contained 21 observations with 18 features. While this is a very small number of observations, the modeling had to be continued due to the lack of further data. The summary statistics of data are presented in below table:

|  | **count** | **mean** | **std** | **min** | **25%** | **50%** | **75%** | **max** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **hcc\_index** | 21.0 | 173.619048 | 34.828833 | 114.000000 | 140.0 | 186.000 | 203.000000 | 208.000000 |
| **cpi** | 21.0 | 146.047619 | 21.622849 | 111.000000 | 126.0 | 150.000 | 165.000000 | 170.000000 |
| **mortgage\_interest\_rates\_index** | 21.0 | 40.190476 | 11.732941 | 27.000000 | 30.0 | 36.000 | 45.000000 | 64.000000 |
| **unemployment\_rate** | 21.0 | 8.470238 | 4.003077 | 3.908333 | 4.5 | 7.375 | 11.958333 | 14.666667 |
| **gnp** | 21.0 | 38172.119048 | 8082.960758 | 22634.500000 | 33646.0 | 40760.250 | 43106.250000 | 53701.750000 |
| **new\_houses\_loans** | 21.0 | 26116.476190 | 17551.295837 | 2960.000000 | 6991.0 | 29220.000 | 39399.000000 | 58104.000000 |
| **other\_houses\_loans** | 21.0 | 36211.380952 | 16697.388504 | 9874.000000 | 27436.0 | 34408.000 | 48250.000000 | 61933.000000 |
| **total\_loans** | 21.0 | 62327.857143 | 33723.668182 | 12834.000000 | 31897.0 | 64652.000 | 88747.000000 | 120037.000000 |
| **avg\_price** | 21.0 | 211693.869048 | 71469.007661 | 77974.250000 | 169383.5 | 228216.500 | 249002.000000 | 321615.750000 |

## Exploratory Data Analysis

An exploratory data analysis on the dataset can be summarized as:

* The distribution of target variable was bit negatively skewed or skewed towards the left.
* Features hcc\_index, cpi, mortgage\_interest\_rates\_index, unemployment\_rate, and gnp were highly correlated and were selected as predictors.
* Based on Correlation-based selection, feature *gnp* was identified as the most important feature for the target variable.

## Model Development

The dataset was trained on 3 Machine Learning Regression models: Linear Regression, Ridge Regression and Lasso Regression. GridSearchCV with 5 folds were used for cross-validation and hyper-parameter tuning. The result summary are presented in table below:

| **Scaler** | **Model** | **Training Score** | **Test Score** | **R-Squared** | **Mean Absolute Error** | **Mean Squared Error** |
| --- | --- | --- | --- | --- | --- | --- |
| MinMaxScaler | LinearRegression | 0.950895 | 0.954067 | 0.95 | 11942.43 | 1.850440e+08 |
| MinMaxScaler | Ridge | 0.948004 | 0.977470 | 0.98 | 8436.71 | 9.076367e+07 |
| MinMaxScaler | Lasso | 0.942553 | 0.984324 | 0.98 | 7596.22 | 6.315118e+07 |

From the results, it can be seen that LinearRegression performed the best in terms of training/test scores and R-squared value. However, Ridge and Lasso also performed very closely, or even better on some metrics.

# Discussion/Conclusions

After performing the exploratory data analysis, feature engineering and machine learning modeling, several conclusions can be drawn. Firstly, it is crucial to thoroughly analyse the input dataset(s), and gain solid understanding using descriptive statistics and visualization. Doing domain knowledge research also plays a key role on this part. Equipped with this insight, the business and data requirements can be determined correctly.

Secondly, the data collection/preparation and feature selection can go hand-in-hand. Knowing which features are most important predictors for the target can aid in future data preparation and modeling efforts. In case of this dataset, GNP was the most important feature with correlation value of 0.9366.

Finally, based on the results of multiple ML models, LinearRegression was found to be best suited for this analysis. That said, the Ridge and Lasso models also performed very closely. The performance improvement was also observed when utilizing data scaling techniques, and the best hyper-parameters. This highlights the importance and usefulness of data scaling, split selection and hyper-parameter tuning.

In conclusion, the analysis suggests that feature GNP is an important predictor for national average house prices, and the Linear Regression model may be the best choice for predicting this variable in future use cases. Nevertheless, further study and experimentation are needed to explore the generalization ability of the findings. More importantly, it’ll be imperative to gather datasets with more relevant observations along with additional features.

# References

* Organization for Economic Co-operation and Development (2023). *Housing and the economy*. Available at: <https://www.oecd.org/housing/topics/housing-economy/> (Accessed: 12 April 2023).
* Kenny G. (1998). *‘The Housing Market and the Macroeconomy: Evidence from* Ireland’, pp. 2-3. [online] Available at: <https://www.centralbank.ie/docs/default-source/publications/research-technical-papers/1rt98---the-housing-market-and-the-macroeconomy-evidence-from-ireland-(kenny).pdf?sfvrsn=4> (Accessed: 12 April 2023).
* Government of Ireland (2023). Datasets, Publisher: Department of Housing, Local Government and Heritage. Available at: <https://data.gov.ie/dataset?q=houses&organization=department-of-housing-local-government-and-heritage&sort=score+desc%2C+metadata_created+desc&theme=Housing> (Accessed: 12 April 2023).
* SparkByExamples (2023). *Pandas vs PySpark DataFrame With Examples*. Available at: <https://sparkbyexamples.com/pyspark/pandas-vs-pyspark-dataframe-with-examples/> (Accessed: 14 April 2023).
* Krishna Kumar, M., Arvind, P. and Dharmendra Singh, R. (2020) Predictive Analytics Using Statistics and Big Data: Concepts and Modeling. Singapore: Bentham Science Publishers Ltd. Available at: https://search.ebscohost.com/login.aspx?direct=true&db=e250xww&AN=2710400&site=ehost-live (Accessed: 12 April 2023).
* Kotu v. and Deshpande B. (2014) Predictive Analytics and Data Mining : Concepts and Practice with RapidMiner. Amsterdam: Morgan Kaufmann. Available at: https://search.ebscohost.com/login.aspx?direct=true&db=e250xww&AN=919334&site=eds-live (Accessed: 12 April 2023).
* Ghavami P. (2019) Big Data Analytics Methods : Analytics Techniques in Data Mining, Deep Learning and Natural Language Processing. Berlin: De Gruyter. Available at: https://search.ebscohost.com/login.aspx?direct=true&db=e250xww&AN=2333425&site=eds-live (Accessed: 12 April 2023).
* Wang, S. (2011) Everything You Need to Know About Frequently Used Probability Distributions. [Newmarket, Ont.]: BrainMass Inc (Need to Know Series). Available at: https://search.ebscohost.com/login.aspx?direct=true&db=e020mww&AN=529713&site=eds-live (Accessed: 13 April 2023).
* OpenStax (2020). 6.4: Normal Approximation to the Binomial Distribution, LibreTexts Statistics, 22 October 2020. Available at: <https://stats.libretexts.org/Courses/Las_Positas_College/Math_40%3A_Statistics_and_Probability/06%3A_Continuous_Random_Variables_and_the_Normal_Distribution/6.04%3A_Normal_Approximation_to_the_Binomial_Distribution> (Accessed: 13 April 2023).
* Paul Wellens (2015) Practical Web Development : Learn CSS, JavaScript, PHP, and More with This Vital Guide to Modern Web Development. Birmingham, UK: Packt Publishing (Community Experience Distilled). Available at: https://search.ebscohost.com/login.aspx?direct=true&db=e250xww&AN=1046853&site=eds-live (Accessed: 13 April 2023).
* Avinash Navlani, Armando Fandango and Ivan Idris (2021) Python Data Analysis : Perform Data Collection, Data Processing, Wrangling, Visualization, and Model Building Using Python. Birmingham: Packt Publishing. Available at: https://search.ebscohost.com/login.aspx?direct=true&db=e250xww&AN=2725992&site=eds-live (Accessed: 13 April 2023).
* Raja B. Koushik and Sharan Kumar Ravindran (2016) R Data Science Essentials. Birmingham: Packt Publishing (Community Experience Distilled). Available at: https://search.ebscohost.com/login.aspx?direct=true&db=e250xww&AN=1151007&site=eds-live (Accessed: 13 April 2023).
* IBM Cloud Team (2021). *‘Python vs. R: What’s the Difference?’*, IBM, 23 March 2021. Available at: <https://www.ibm.com/cloud/blog/python-vs-r> (Accessed: 13 April 2023).
* Christopher Greco (2021) Data Analytics : Systems Engineering - Cybersecurity - Project Management. Dulles, Virginia: Mercury Learning & Information. Available at: https://search.ebscohost.com/login.aspx?direct=true&db=e020mww&AN=2730671&site=eds-live (Accessed: 13 April 2023).
* Central Statistics Office (2023). *Current and Constant Prices*. Available at: <https://www.cso.ie/en/interactivezone/statisticsexplained/nationalaccountsexplained/currentandconstantprices/> (Accessed: 28 March 2023).
* Fagroud, F.Z. et al. (2022) ‘Connected Devices Classification using Feature Selection with Machine Learning’, IAENG International Journal of Computer Science, 49(2), pp. 445–452. Available at: https://search.ebscohost.com/login.aspx?direct=true&db=edo&AN=157247549&site=eds-live (Accessed: 14 April 2023).
* Brownlee, J. (2020). ‘*How to Use StandardScaler and MinMaxScaler Transforms in Python’*, Machine Learning Mastery, 10 June 2020. Available at: <https://machinelearningmastery.com/standardscaler-and-minmaxscaler-transforms-in-python/> (Accessed: 07 April 2023).
* Tommy Blanchard, Debasish Behera and Pranshu Bhatnagar (2019) Data Science for Marketing Analytics : Achieve Your Marketing Goals with the Data Analytics Power of Python. Birmingham, UK: Packt Publishing. Available at: https://search.ebscohost.com/login.aspx?direct=true&db=e020mww&AN=2094757&site=eds-live (Accessed: 14 April 2023).
* Giuseppe Bonaccorso (2020) Mastering Machine Learning Algorithms : Expert Techniques for Implementing Popular Machine Learning Algorithms, Fine-tuning Your Models, and Understanding How They Work, 2nd Edition. [N.p.]: Packt Publishing. Available at: https://search.ebscohost.com/login.aspx?direct=true&db=e250xww&AN=2366459&site=eds-live (Accessed: 14 April 2023).