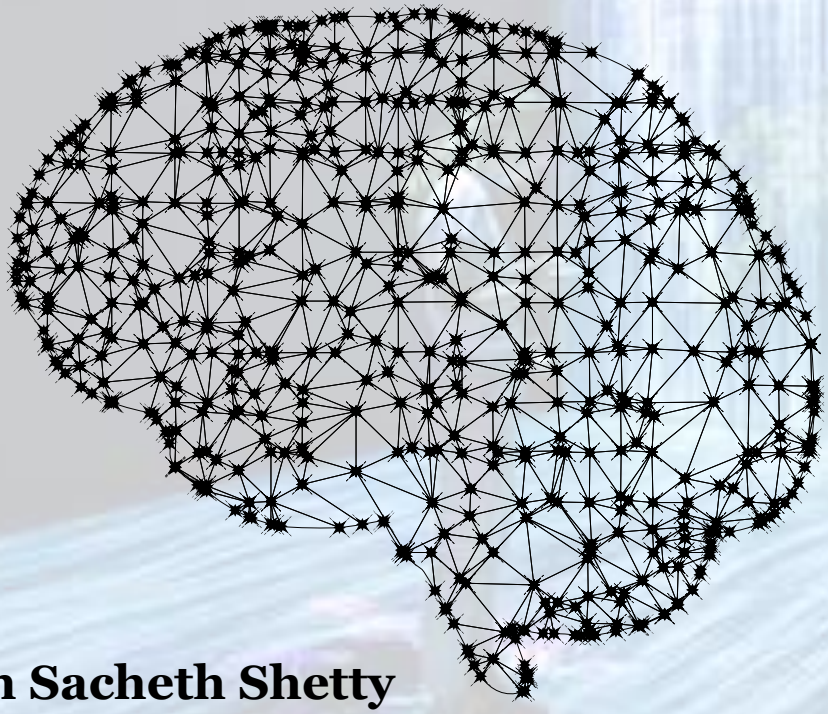
Aston University

BIRMINGHAM UK

MSc Business Analytics Dissertation

A comprehensive analysis of Feature engineering and Feature selection methods supported by a simulation study



Gagan Sacheth Shetty

210197062

Supervisor – Dr Shahin Ashkiani

Dissertation submitted to Aston Business School, Aston University

In partial fulfilment of requirements for MSc Business Analytics

Acknowledgements

I would like to extend my gratitude to Dr. Shahin Ashkiani, who made me realize the importance of feature engineering and feature selection for business applications along with many other important concepts while studying the Data mining module under him. Due to his insights and understanding, I could generate the idea of probing further into the topic of feature engineering and feature selection. Hence, I reached out to him to supervise me for guidance in this dissertation.

While working on this dissertation, Dr. Shahin's guidance has been very fruitful in forming the foundation of this study. Overall, in gaining a good understanding of machine learning and business analytics, I would like to thank all my professors who provided invaluable insights.

Abbreviations

IBM - International Business Machines
BRT - Boosted Regression Trees
SVM - Support Vector Machines
RF - Random Forest
MAE - Mean Absolute Error
FE - Feature Engineering
FS - Feature Selection
PCA - Principal Component Analysis
RMSE - Root Mean Squared Error
CSV - Comma Separated Value
RELU - Rectified Linear Activation Unit
TF - Term Frequency
IDF - Inverse Document Frequency
RFE - Recursive Feature Elimination
GAMS - Generalized Additive Models
MARS - Multivariate Adaptive Regression Spline models
NNMF - Non-Negative Matrix Factorization
RBF - Radial Basis Function
LASSO - Least Absolute Shrinkage and Selection Operation
CART - Classification And Regression Tree

TABLE OF CONTENTS

TABLE OF FIGURES & TABLES

1. Abstract

Feature engineering and feature selection are integral parts of machine learning, pushing the model's performance which is necessary for businesses in the modern day as they deal with a massive amount of data and are looking to leverage this. However, a comprehensive framework consisting of the need for feature engineering with a simulated proof and review of the various methods covering working mechanism, application, limitation, and methods to overcome limitations are missing in the research currently available.

This study aims to achieve the shortcomings mentioned above in the available research. The flow is set so that the concept of error, i.e., reducible and irreducible errors, is introduced in the first part of the study. Then with simulation using a toy dataset, proof of the inability of machine learning models to break the reducible error ceiling is shown. The power of feature engineering is then demonstrated where with the aid of transformation, the ceiling is broken with the newly introduced feature.

Further, the second part of the study delves deeper into the various methods available under feature engineering and feature selection with a discussion of important information such as mechanism, function, application, and limitations.

The ultimate goal of this research is to help machine learning practitioners quickly adopt and use feature engineering and feature selection methods which is currently a tedious process.

2. Introduction

2.1. Background of study

Businesses worldwide experience a massive influx of data. It is understood that companies worldwide are already generating around 2 million terabytes of data daily, with only 26% of companies worldwide estimated to inculcate a data-driven culture (Brown, 2021). Data is generated mainly from customers, day-to-day events collection, process flows, data from various internal and external stakeholders, website analytics, and historical data. The value of this data is now being realized and appreciated with the insights and opportunities gained by understanding the data and using it to predict future events.

Some examples of businesses effectively utilizing the data for predictive modelling are –

Managing quality control issues in a large-scale manufacturing system with predictive models such as Classification and Regression Trees (Decision trees) to predict the issues before occurring has one of the initial implementations of data mining in the business application by IBM in the 1990s (Chid Apté, 2019)

Another prominent and early use of machine learning for business can be traced back to the banking sector. The sector consists of an extensive database of user data on customer banking behaviour and finance engagement, this data of the user is leveraged effectively by banking for the following applications (Ostapchenya, 2021)

- Customer segmentation
- Credit risk and fraud prediction
- Feedback analysis

Service-providing businesses such as internet service providers, telephone companies, and electricity and gas provider companies use customer data and perform an analysis called churn analysis to predict when the customer is most likely to leave service based on various associated factors with regards to their behaviour (Chen, Li and Ge, 2011)

Ecommerce also has vast data mining applications (S and R, 2021) for significantly impacting the business's sales and performance. Some of the most prominent applications are

- Understanding market segmentation to identify key demographics
- Basket analysis for predicting auxiliary products to go with products already in the basket
- Forecasting sales for effective inventory management
- Detecting fraudulent behaviour of users who use fishy payment methods

The above examples of the application of data mining in business applications are only some of the representative examples; in reality, the application of machine learning in business applications is multifold, with new and innovative applications coming up rapidly with the rapid digitalization of business and with needs for a business to resort to innovative methods in a highly dynamic world

As we can summarize, the background of this study is the need for efficient machine learning implementation in a business context. The focus of this study is to discuss in detail the data transformation process or the raw material transformation process, which pushes the already

established machine learning models for further optimum performance. The focus is discussed in detail in the next section.

2.2. Focus of study

As discussed above in the background on the importance of machine learning in business applications, this study delves deeper into raw data transformation concepts rather than developing new algorithms, as substantial research on algorithms is already present.

Working on transforming the data, deriving suitable features (Feature engineering) and eliminating redundant features (Feature selection) help the model by making it more susceptible to the underlying relationship between the predictors and target (Kuhn and Johnson, 2019).

Feature engineering and Feature selection are fundamental steps in a machine learning process (Li, Ma and Xin, 2017). These steps directly result in further improvement of model performance. The complexity associated with these steps is that there are no set rules or guidelines for implementing the methods for different types of applications. Usually, the practitioner runs trial and error with the various available methods, and this process is usually one of the most challenging and time-consuming processes in a machine learning model development (Spanhol et al., 2016), (Christ et al., 2018)

Hence, this study focuses on understanding feature engineering and the feature selection process; the value targeted to be attained from this study is detailed in the below section.

2.3. The value obtained by this study

The value proposition for this study is to give data science practitioners an overall knowledge base of feature engineering and feature selection methods.

This study is designed to initially provide a complete overview of feature engineering necessity, i.e., the need to derive new features with the help of a simulation study. The study is explained with the concept of reducible and irreducible errors and how the irreducible error can be converted into reducible errors with the derivation of new features.

The subsequent section covers the various methods under feature engineering and feature selection. The basis of selecting these methods was such that a good insight into one of them

could give a good understanding of the variety of methods also covered by the same principle. For example, under the power transformation method, logit transformation and box–cox transformation are discussed in detail; going ahead, if the practitioner is exposed to more powerful transformation methods, they can easily understand the correct application and practical usage of the method.

2.4. Research aims and objective

2.4.1. Aim of the study

This research aims to understand the working principle of feature engineering and analyse various feature engineering and feature selection methods to help practitioners quickly implement and execute the data preprocessing steps.

2.4.2. The objective of the study

The defined aim for this research shall be met by achieving the below milestones

- Establish an understanding of the concepts of reducible error, irreducible error
- Understand the role of feature engineering in converting irreducible error to reducible error
- Validate the results obtained from the simulation study with different models, critically analyze the results and attempt to understand the behaviour of the model
- Select a wide variety of methods to cover the most prevalent family of methods under feature engineering
- Comprehensive analysis of 4 main types of feature selection categories and discuss methods in each category.

3. Literature review

3.1. What are feature engineering and feature selection?

Machine learning is the subject where the study of using computational efforts is used to understand and simulate human behaviour (Wang, Ma and Zhou, 2009). The various methods available in Machine learning are applied to a dataset to predict a target variable using a set of predictors.

A machine learning model can identify the underlying patterns in the data with the aid of predictors to accurately predict a variable quantitatively or qualitatively (Jain, Duin and Jianchang Mao, 2000). The machine learning process can be broadly divided into the steps below (Yufeng G, 2017).

1. Data collection
2. Data preparation
3. Choosing a model
4. Training a model
5. Evaluating the model
6. Hyperparameter tuning
7. Prediction

The primary focus of this paper shall be on the data preparation part, which is one of the most crucial parts of the machine learning process and can have a significant impact on the performance of the model (Brownlee, 2020)

Data preparation, also called data wrangling, is the process of tidying the data and mapping it into a different form which is more suitable or desirable for our machine learning model. Various steps involved in the data preparation process are as follows (Brownlee, 2020)

1. Cleaning the data
2. Feature selection
3. Feature engineering
4. Data transforms
5. Dimensionality reduction

Under the data preparation process in machine learning, we shall focus on two key aspects which will directly impact the resulting performance, i.e., Feature engineering and Feature selection.

Feature engineering is also considered in some research as an encompassing term that involves feature generation, feature transformation, feature selection, feature analysis and evaluation, as mentioned in (Dong and Liu, 2018). In this paper, feature engineering and feature selection are treated separately as mutually exclusive.

As quoted in the book by renowned authors – Introduction to statistical learning (James et al., 2021), the accurate prediction of the target variable is dependent on two factors which are

reducible and irreducible error. Reducible error is due to the underlying function for predicting the target. However, in real-life situations, the second factor of irreducible error is always present, which creates an upper bound to the prediction accuracy (Lee and Shin, 2020). Feature engineering aims to optimize the variables so that the function contributing to the reducible error is minimized as much as possible.

Feature engineering, as explained in the book Feature engineering and selection, A practical approach for predictive models (Kuhn and Johnson, 2019), is the process of manipulation of available predictors by re-representing them in a way that can be beneficial to machine learning model by either improving its performance or learning speed.

Feature selection is an essential step in the machine learning process which helps build uncomplicated and comprehensible models by improving the performance and making the data clean and more understandable, as introduced in the paper Feature selection: A data perspective (Li et al., 2018).

As summarized in the research mentioned above (Li et al., 2018), Feature selection discards redundant information or features that do not contribute significantly to the learning process of the machine learning algorithm. These unnecessary features may give the machine learning model a tough time finding an underlying pattern in the data when it does not exist (Khalid, Khalil and Nasreen, 2014).

3.2. Why do we do feature engineering and feature selection?

As iterated previously, Feature engineering and feature selection are one of the most critical processes in machine learning. They can impact the performance of the machine learning model.

Feature engineering help expose the underlying patterns of the dataset by manipulating the variables, as explained in detail in the next section and aid the algorithm in understanding more accurately and providing reliable predictions for the data.

Feature selection helps reduce the noise in the dataset by removing unnecessary and least significant variables which may not contribute to the prediction of the target variable. These variables may hinder the working of the algorithm, which is trying to understand the pattern by misleading it and making the learning process difficult (Khalid, Khalil and Nasreen, 2014).

As concluded in the paper – The impact of preprocessing steps on the accuracy of machine learning algorithms in sentiment analysis (Alam and Yao, 2018), an interesting study is carried out to quantify the impact of data preprocessing steps on machine learning algorithms. It was concluded that data preprocessing does have a significant effect on the performance accuracy of the Naïve Bayes algorithm as it had improved drastically after preprocessing was performed in the context of sentiment analysis.

The importance of feature selection methods has been iterated in the paper – A survey of feature selection methods (Chandrashekar and Sahin, 2014). Where comprehensive analysis of various feature selection methods was conducted on a dataset, it was concluded that applying feature selection methods would most certainly provide various benefits, such as identifying redundant variables, improved model prediction metrics, and data insights.

The paper - The effect of tuning, feature engineering, and feature selection in data mining applied to rainfed sugarcane yield modelling (Bocca and Rodrigues, 2016) makes a solid comparison of the methods in tuning, feature engineering and feature selection against Boosted Regression Trees (BRT), Support Vector Machines (SVM), and Random Forest (RF) machine learning algorithms against the performance metric Mean Absolute Error (MAE). The best performing model was found with the application of tuning, complete feature engineering and complete feature selection.

Hence it can be concluded from the various research evidence presented above that feature engineering and feature selection is inseparable from machine learning as they directly impact prediction quality and provide various added benefits to our understanding of the data. Using the right or most suitable methods under feature engineering and feature selection would require trial and error efforts and subject matter expertise in some situations. Hence FE and FS can be the driver of performance for a machine learning model that is used effectively.

3.3. How to do feature engineering and feature selection?

Various techniques currently known under feature engineering and feature selection are listed below, and some of these shall be taken into the comprehensive analysis section, which shall be explained in detail below.

In a dataset, the variables can be classified into two types which are the quantitative type or numerical variables and the qualitative type or categorical variables (Abs.gov.au, 2015); in this

article, various methods are discussed relating to both these variables and perform a comprehensive analysis against various metrics for a model.

Categorical variables have two or more categories, which may be ordinal or unordered nominal (lbe, Fundamentals of applied probability and random process second edition Chapter 8). Some machine learning algorithms can have a tough time interpreting text data unless presented in a suitable format since the inner working algorithm takes inputs in the form of numbers to find the optimum coefficient (D McGinnis et al., 2018). Hence, categorical variables are transformed into suitable numerical encoding before being passed onto the modelling stage. Some currently known and used methods are listed below for encoding categorical variables.
Categorical predictor encoding

1. Dummy variables for unordered categories
2. Feature hashing
3. Effect or likelihood encoding
4. Numerical encoding
5. Text – odds ratio
6. Word – qualitative assessment
7. Term frequency – Inverse document frequency statistic

Numerical variables quantify a specific characteristic of a record; as the name suggests, the data type is usually numerical, which can be an integer or float (Kaliyadan and Kulkarni, 2019). The need for numerical encoding arises since the underlying function may consider a different form of the variable; to aid in estimating this function, we use various encoding methods and try to improve the model's performance.

Numerical predictor encoding

1. Box-Cox transformation
2. Logit transformation
3. Arcsine transformation
4. Standardization – Centering
5. Standardization – Scaling
6. Data smoothing
7. Smoothing splines
8. Basis expansion
9. Categorization / Discretization

10. Principal Component Analysis
11. Kernel PCA
12. Partial Least squares
13. Autoencoders
14. Distance and depth features

Usually, in the data collection process, much-related data is collected based on intuition and subject matter expertise without knowing the impact of the variable on the target variable (Feature selection for classification (Dash and Liu, 1997). Hence efforts must be put in place to analyze and understand the relevance of the variables, and irrelevant features concerning target prediction are discarded for performance improvement. There are three broad categories consisting of many methods for performing this task as listed below; each of these methods are discussed in detail further in the article.

Feature selection generalization as discussed in the paper – A review of feature selection methods with applications (Jovic, Brkic and Bogunovic, 2015).

1. Wrapper methods

   These methods create a subset of features and evaluate their performance with a machine learning model, thus evaluating all combinations and selecting the best model (Talavera, 2005)
2. Embedded methods

   The machine learning algorithm consists of intrinsic methods that weight the features based on relevance or importance and use the most relevant predictors while performing prediction (Lal et al., 2022)
3. Filter methods

   Statistical measures of similarity, distance and consistency are employed in these methods to find out the best features concerning the target variable and hence give information on the retention or removal of the features before passing data through the machine learning algorithm (Talavera, 2005)

3.4. What is the current scope of available research in feature engineering and feature selection?

The current literature available in the field of feature engineering talks about various methods available depicting the transformation steps and the functional form of the feature engineering methods, as we can see in the article - Feature engineering tools and techniques for better

classification performance (Rawat and Khemchandani, 2017), here a general collection of the various transformation methods available is provided, highlighting the underlying functions

In the above research, the article provides numerous options for using methods; however, comprehensive analyses are not presented between the algorithms, and the application and context of usage of these feature engineering methods are unavailable.

Also, in the book by renowned authors, Feature engineering and feature selection, a practical approach for predictive models (Kuhn & Johnson, 2019), the methods for categorical encoding and numerical encoding are discussed in depth with examples of their applications on trial datasets

The above book discusses a vast collection of methods; a reader can gain a deeper understanding of the methods used in a practical context. However, it misses providing a comprehensive analysis, applicability to machine learning algorithms and context of usage.

The research – An empirical analysis of feature engineering for predictive modelling (Heaton, 2016) aims to compare performance with error as a metric with various numerical transformations against multiple machine learning models is limited regarding numerical predictors and error metrics for comparison.

The above research is restricted to numerical encoding methods and compares the model's performance based on error metrics. This research also misses being a comprehensive analysis with widely used methods and does not consist application of the methods with context.

In the research paper – Intelligent feature engineering for cybersecurity (Maxwell, Alhajjar and Bastian, 2019), a comparison of classification accuracy against selected feature engineering for categorical variables and various modelling algorithms has been presented in the context of cybersecurity. This paper provides a solid comparison and analysis of the change in behaviour of the model when encoding is conducted on the available variables.

The paper successfully compares the change in performance on the application of feature engineering methods to different variables, and this analysis is conducted in the context of cybersecurity; application and relevance to machine learning algorithms are not discussed

The paper – Performance comparison of Feature selection methods (Phyu and Oo, 2016) discusses the comparison of four existing feature selection methods: Information Gain,

Symmetric uncertainty, and Relief - F and an author proposed algorithm with its performance against Naïve Bayes and J48 classifier.

This paper fails to comprehensively analyse the three significant categories available under feature selection methods. Also, this analysis applies only to classification problems.

As we can summarize from the above research, a comprehensive analysis of the need for feature engineering and the various methods under feature engineering and feature selection is missing in a single paper. The exploration of the context of usage and application of feature engineering methods can be widely appreciated by any user of a machine learning model or someone performing research in related fields. This article aims to fill this gap and provide helpful information on the different FE and FS methods that rely on trial and error to find the most suitable approach.

One of the lesser probed areas in machine learning is the simulation study. The simulation study is the artificially creating of a dataset with randomly distributed noise and understanding the behaviour of various machine learning models. This process can give a greater insight into feature engineering. Machine learning models' behaviour can be tested effectively since the underlying connection between the predictor and target is already understood.

The study mentioned above also highlights the critical concept of error, which shall be explained as follows.

Assuming quantitative predictors and targets, we can assume the target to be X_1, X_2, X_3, X_4 ….. and the target to be Y. The underlying relationship between the predictors and the target can be depicted as (James, Witten et al., 2021)

$$Y = f(X) + E$$

Here the term ‘E’ is added to denote the error in the function. This term is generally introduced to denote the inability to perfectly predict an outcome due to factors out of the user's control. The error may be introduced due to the dataset observations, which may behave out of normal, or it is unrelated data being added to the dataset.

The error term can be crucial to understanding and unlocking the potential of a machine learning model. When the underlying pattern between the predictors and the target is incorrectly represented, any amount of hyperparameter tuning of the model shall render a maximum error value which cannot be further reduced. This error value can be considered a

ceiling to the reducible error. Feature engineering, with its operations over the features, usually helps expose the correct representation of the predictors. This aids in breaking the error ceiling posed by the false representation of the dataset and helps achieve further reduction in the error. The subsequent sections will demonstrate and prove this with a detailed simulation study.

Hence this study shall include this crucial element of a simulation study to empirically demonstrate the function of feature engineering in aiding the improvement of the machine learning model.

A notable mention of the importance of feature engineering is in the world-renowned online data science competition portal Kaggle. This portal consists of elite machine learning practitioners and data science enthusiasts who come together to solve and compete in some of the most complex problems in the field of data science. The performance of the teams or the individuals involved in the competition is based on various error metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and F1 – Score.

In a paper on the Kaggle competition – "Winning the Kaggle Algorithmic Trading Challenge with the Composition of Many models and Feature Engineering", the authors (De and Sugiyama, 2013) demonstrate how a prediction problem related to the prediction of short-term response of Order Driven Market following shocks in the liquidity market, was successfully won by efficiently using Feature engineering and Feature selection methods to improve the model and win the competition gradually. This paper depicts the importance of feature engineering, which can further differentiate an optimally working model from the best performing model to drive the accuracy of the predictions.

Hence, it can be concluded that a more detailed focus on feature engineering and feature selection methods can aid data science practitioners in efficiently executing data science problems and achieving more favourable outcomes.

4. Research Methodology

4.1. Simulation study methodology

As introduced in the previous section, a machine learning simulation study will be used in this study to understand and demonstrate the inner working mechanism of feature engineering. The objective of this section would be

- to demonstrate the reducible error and the irreducible error.

- The maximum extent to which the error can be reduced with hyperparameter tuning of the machine learning model – The reducible error ceiling
- Deriving new features out of the existing features to break the reducible error ceiling and gain better performance from the machine learning models
- Graphically represent the alignment of underlying function with the prediction of machine learning models without feature engineering and with feature engineering

4.1.1. Dataset design

The dataset considered for the simulation study is a regression problem. The context assumed for the study is housing price prediction, a common business prediction problem. The characteristics of the dataset are as follows.

Predictor: Initially, the dataset is constructed to have 3000 data points. The carpet area is selected as a random number from the value of 10 – 5000.

Noise: Randomly distributed noise is added with a mean of 0 (James et al., 2021) to the dataset to simulate a real-life scenario almost always containing environmental factors out of control.

Target: Since this is a regression problem, the target shall be a quantitative variable. The underlying relation between predictor and target is considered a square function multiplied by a factor. The factor here is 16. Hence the final function is calculated as follows

$$Y = f(X) + E$$

$$Price = 16*(carpet\_area)^2 + Noise$$

Noise is then deleted from the dataset, and the dataset is saved and loaded again so that the model cannot be exposed to noise and data leakage be avoided.

4.1.2. Validation dataset design

Care is taken into the design of the testing dataset such that there is no data leakage, i.e., there are no common values in the training set and Validation dataset.

Since the training dataset predictor consists of randomly selected values or carpet areas in the range of 10 – 5000. Validation set data points are values randomly selected from 5000 – 10000. There is no overlap between training and validation to ensure that data leakage does not occur since the machine learning model is already exposed to values in the range of 10 – 5000 in the

training process. It is also exposed to the resulting values from this pool of values. If a Validation set is taken as a subset of the above data points, the results may collude as the machine learning model might have overfitted to the results and can be displayed the same. Hence, care is taken such that an unknown predictor is presented to the machine learning model so that the result can be confidently assumed to be correct regarding the model's behaviour.

4.1.3. Graphical validation of resulting prediction accuracy with and without feature engineering

The performance of the model is evaluated with a superimposed scatterplot graph. The scatterplot consists of the independent variable – carpet area on the x-axis, while the dependent variable price shall be on the y-axis. The two scatterplots which are superimposed on each other are

i) The training set - predictor carpet area vs target house price

ii) Validation set - predictor carpet area vs predicted house price

Interpretation: We can assume the prediction power shall be confirmed to be of superior quality if the superimposed scatterplot consists of a continuous straight line running at 45^0 to the x-axis. This ideal straight line would mean that the underlying function is accurately followed to predict the validation set target. The below section demonstrates this with examples of how the super-imposed graphs look when the prediction is accurate and when it is not.

4.1.3.1. Graph depicting weak predicting power

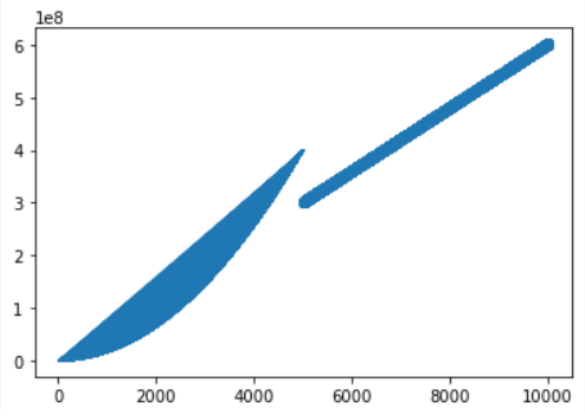


Figure 1 - Graph showing no continuity between actual and predicted

As we can see in the above graph, there is a disconnect between training and testing predictions, and they are not part of a continuous line. Hence, we can claim that the model depicting this graph shows weak performance.

4.1.3.2. Graph depicting strong predictive power

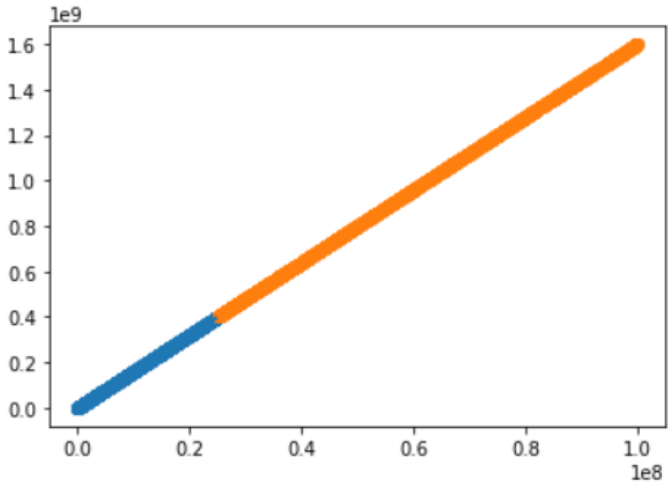


Figure 2 - Graph showing continuity between actual and predicted

The above graph shows a continuous line showing that the model has understood the underlying pattern in the data and has accurately predicted the values for the validation dataset. Hence, it can be claimed that the model performance depicted in the above graph is of superior quality.

4.1.4. The performance metric for quantitative comparison of models with and without feature engineering

The performance metric selected for the study is – Mean Absolute Error (MAE). The reason for this is as follows.

As this is a simulation study, we would need the error value to be as intuitive as possible and Mean Absolute Error is an intuitive performance metric (Yang and Sun, 2021). Since MAE is in the same units as the predictor and the changes in MAE are linear, it is easier to interpret this value to understand how the model performs. Another supporting reason for not employing the popular metric Root Mean Squared Error (RMSE) is that this metric causes inflation or heavily penalizes the error points situated further away from the actual values. This penalization can be non-ideal for our scenario as it beats the purpose of our simulation study, which is meant to be intuitive for clearly understanding the role of feature engineering.

4.1.5. Simulation procedure

The main procedure for the simulation is carefully designed to ensure the study can be trusted for its results. The simulation procedure is described in detail below.

This simulation is performed in the python programming platform with the aid of various packages such as pandas – Dataset management, NumPy – Dataset management, and matplotlib.pyplot – Plot resulting graphs to understand performance, SciKit-Learn – Library of machine learning models, GridSearchCV – Hyperparameter tuning with cross-validation to identify the best hyperparameters, joblib – to save models and load on the validation set

4.1.6. Testing for model performance without feature engineering with hyperparameter tuning

Step1: The dataset is saved as a CSV post design and loaded again.

Step2: Since there is no feature engineering involved in this test, the dataset is split into the X train consisting of the array of predictor values and the y train with target value - Price

Step3: Simple Linear regression model is loaded, and the model is fit on the defined X train, and y train and the model is saved.

Step4: The validation dataset, as defined previously in section 4.1.2, is split into X-validation and y-validation.

Step5: The saved model is then run to predict the values for the X test.

Step6: The superimposed scatterplot graph discussed in section 4.1.3 is created to visually understand the model's predictive performance.

Step7: From SciKit, learn the models –

i) Multilayer Perceptron – Regressor (Neural network)
ii) Support Vector Machine – Regressor
iii) LASSO regressor
iv) Ridge regressor
v) Random forest regressor
vi) K – Nearest neighbours regressor

The above models are imported to test for perceptiveness of feature engineering, and sequentially the below procedure is followed for all the above models

Step8: Available Hyperparameters are collected, and possible options for these hyperparameters are passed onto grid search.

Step9: The best combination of the hyperparameter is determined based on the error metric selected, which is the mean absolute error in our case.

Step10: The trained model is saved, which consists of the most optimal hyperparameters in the local folder.

Step11: The saved model is loaded, and prediction is run on the validation dataset; the performance metric mean absolute error is calculated for the best model in the validation dataset, and the score is saved.

Step12: The superimposed scatterplot graphs are plotted to understand the model performance visually, as explained in section 4.1.3

Step13: The score for all the models is recorded for comparison

4.1.7. Testing for model performance with feature engineering and hyperparameter tuning

Step14: The dataset is loaded, and feature engineering is carried out, i.e., square transformation is performed on the predictor to derive a new feature.

Step14: The original feature using which the new feature was derived is then dropped

Step15: X train and y train are split for training the model

Step16: All the same steps from Step7 to Step10 are carried out. The key difference here is the training dataset which shall consist of feature engineered predictor

Step17: The model trained on feature engineered dataset is saved in the local folder

Step18: The validation dataset is loaded, and the same data preprocessing step is carried out for the validation dataset, which in this case refers to the squaring of the predictor.

Step19: The loaded model is run to predict the results for the validation dataset, and the predicted values are compared against the actual values against the Mean Absolute Error metric.

Step20: The superimposed scatterplot graph is then plotted to understand the predictive performance of the model visually

Step21: The results of the model performance with feature engineering are tabulated

Step22: The scores and the plotted scatterplot graphs are evaluated to understand the behaviour of the machine learning models with and without the presence of feature engineering

Each machine learning model summarises the resulting Mean Absolute Error values with feature engineering and without feature engineering. They are critically analyzed to understand the behaviour of the machine learning model.

4.2. Methodology for Survey of feature engineering methods

Feature engineering methods and their compatibility over various datasets is a subject where with certainty, we cannot claim that a specific feature engineering method is the best method. The reason is that the underlying relationship between the predictor and target can be very dynamic in different datasets. Hence, machine learning practitioners usually employ the trial–and–error method for identifying the suitable feature engineering method. Since feature engineering is critical to a machine learning process, much time may be devoted to finding the best method. This survey aims to enrich user knowledge with all the commonly known methods and various essential information regarding the methods. This crucial information can aid in the employment of feature engineering methods.

The main objective of this section is to bring together a wide array of feature engineering methods and analyze and understand it critically by

- Classifying it into the correct category of the type of transformation
  This section is vital for understanding the broader category under which a feature engineering method falls. Some examples of classification are Learning type (supervised, Unsupervised), Type of transformation (1:1, 1: Many, Many: Many)
- The working mechanism of the method
  This section shall introduce the transformation mechanism or the function to which the variable is subjected in the feature engineering method. This information is critical for users who are subject matter experts or are regular practitioners in the field who understand the data in depth. This information shall benefit these members who can apply the suitable method based on the knowledge of the underlying function.

- The function of the method

  After understanding the internal working mechanism of the feature engineering method, this section helps take understanding a step further by providing a deeper insight into the functioning of the feature engineering method.
- Application of the feature engineering method

  The feature engineering methods usually are suited for specific applications more than others; these applications are explained in this section. Also, the pros and cons of using the feature engineering method under discussion are explained in detail here.
- Limitations to the feature engineering method

  The feature engineering methods cover a wide range of applications. However, due to the internal working mechanism, there are limitations to input that can be passed onto methods for transformation. An example of this scenario is log transformation which is a popular feature engineering method used to make the data normally distributed and to reduce the outliers. Due to the internal mechanism of the logarithmic function, the inputs that are passed are supposed to be non–negative values. Necessary restrictions such as above are highlighted here, and workaround around this limitation, if known, are also mentioned.
- Methods of overcoming the limitation of the feature engineering method

  This section details the known ways of overcoming the limitation discussed in the previous section, if available.

Overall, this research methodology is designed in such a way that initially, with a simulation study, the need for feature engineering is demonstrated with the concept of reducible and irreducible error. The various useful methods are discussed in detail in feature engineering and feature selection.

5. Research outcome

5.1. Simulation study

5.1.1. Simple linear regression

5.1.1.1. Without feature engineering

Actual coefficient of the predictor: 16

Predicted coefficient of the predictor: 60225.81

Superimposed scatter plot for visually understanding prediction performance:

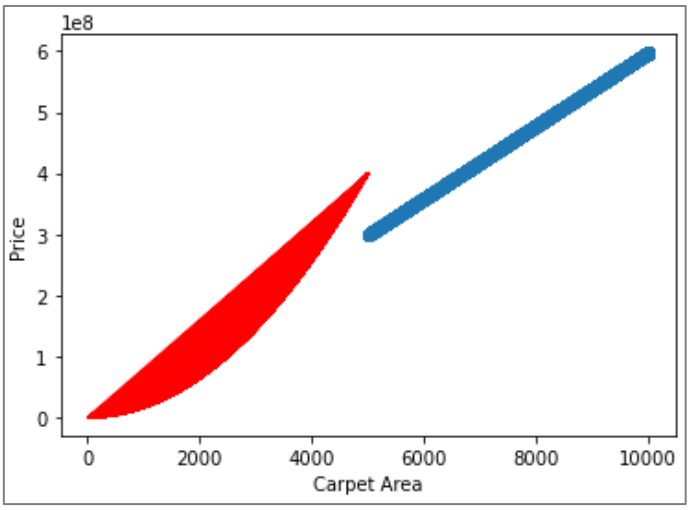


Figure 3 - Output graph for linear regression without feature engineering

Interpretation of results: This graph shows the weak performance of the model. It can be seen that the linear regression model does not perform very well when the data is not transformed. There is a break in the graph with no continuity between the actual and predicted values. The expectation is that the superimposed scatterplot shall show a continuous straight line. We can see a big difference in the predicted value vs actual coefficient, confirming the model's poor performance.

Summary: This is expected as the predictor to target relationship is not linear but quadratic. The model does not perceive this very well.

5.1.1.2. With feature engineering

Actual coefficient of the predictor: 16

Predicted coefficient of the predictor: 16.0000000000001

Superimposed scatter plot for visually understanding prediction performance:

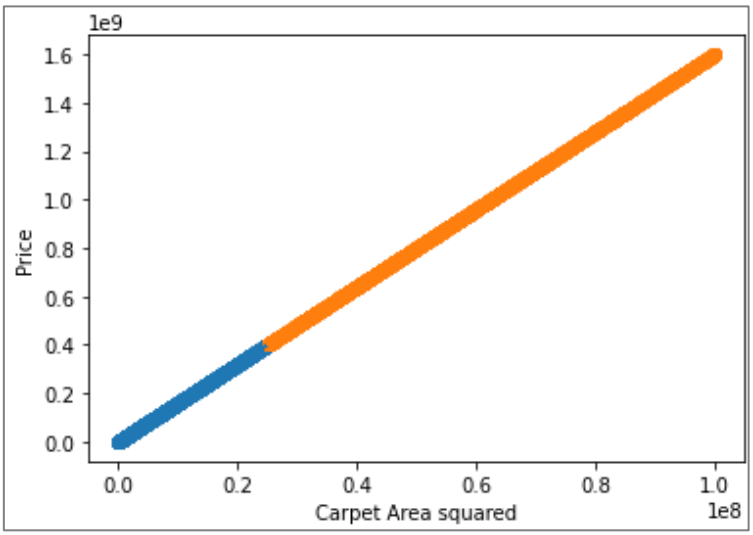


Figure 4 - Output graph for linear regression with feature engineering

Interpretation of results: The graph demonstrates strong predictive performance. We can see continuity between the predicted and actual values due to the straight line passing along the diagonal of the graph. The linear relationship is also demonstrated between the transformed and actual values. The strong performance is also demonstrated with the actual and predicted value having the same value with an error of $1x10^{-13}$.

Summary: The performance of the linear regression model got a good boost when the transformation was performed on the predictor. The transformation helped establish linear relations between the predictor and target, which the model could easily capture and interpret to predict accurate values successfully.

5.1.2. Multilayer perceptron regressor – Neural network

5.1.2.1. Without feature engineering, with hyperparameter tuning

Hyperparameter grid optimized:

The overall parameters considered which are applicable for the Neural Network model out of which the most optimal parameter is chosen are

'Hidden layer sizes' represents the x number of neurons present in a total of y layers (x,y)

'Activation' is the parameter that selects the function responsible for the hidden layer. The options available are rectified linear unit function (RELU), the hyperbolic tan function (tanh), linear identity function, and logistic sigmoid function.

'Solver' refers to the penalizing or the optimization function for the weights, the options available are quasi-Newton methods(lbfgs), stochastic gradient descent(sgd), enhanced stochastic gradient-based optimization(adam)

Hidden layer sizes - (100, 4), (300,4), (500,4)

Activation - identity, relu, tanh, sigmoid

Solver - lbgfs, sgd, adam

Mean absolute error for validation set: 1002181993.58

Superimposed scatter plot for visually understanding prediction performance:

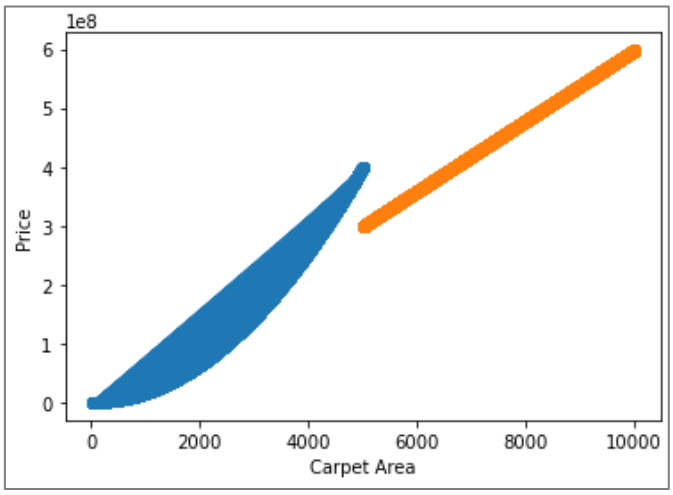


Figure 5 - Output graph for multilayer perceptron without feature engineering

Interpretation of results: We can understand from the above graph that the feed-forward neural network does not do an efficient job of accurately predicting the target variable. There is a disconnect from the actual relationship to the predicted values. The weak performance is also reflected in the performance metric Mean Absolute Error values, which are very high.

Summary: The dataset without transformation is not perceived very well by the Multilayer Perceptron model. The relationship estimated is similar to linear regression, where the target is assumed to be linearly related to the predictor while, in reality, they are not linearly related.

5.1.2.2. With feature engineering, with hyperparameter tuning

Hyperparameter grid optimization: The same set of hyperparameters, as discussed in section 5.1.2.1. are passed to this model to select the most optimal set.

Mean absolute error for validation set: 57.67

Superimposed scatter plot for visually understanding prediction performance:

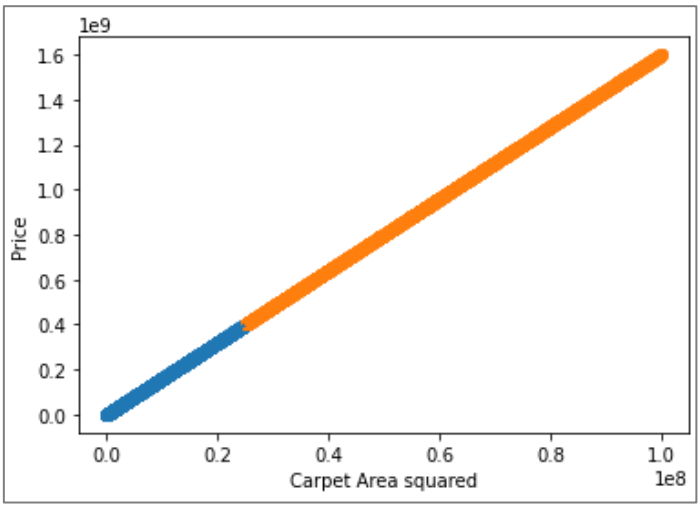


Figure 6 - Output graph for multilayer perceptron with feature engineering

Interpretation of results: The graph demonstrates excellent performance by the Multilayer Perceptron model since there is a continuous straight-line present connecting the predicted values to the actual values from the training set. The strong performance is also confirmed by the value of the performance metric MAE which is only 57.67. This value is a massive improvement over the model prediction without feature engineering.

Summary: The transformation of the predictor aided the neural network model in rightly predicting the target variable. Hence, we can understand that the neural network model is perceptive to transformation and performing feature engineering is beneficial for improving the model.

5.1.3. Support Vector Machine – Regressor

5.1.3.1. Without feature engineering, with hyperparameter tuning

Hyperparameter grid optimized:

The overall parameters considered which are applicable for the Support Vector Machine model out of which the most optimal parameter is chosen are

'Kernel', this parameter defines the kernel function based on which the optimization is performed. The various options available are radial basis function(rbf), polynomial(poly), linear

'Gamma', this parameter defines the kernel coefficient for the selected function. This is generally a float value.

'C' is the regularization parameter. The strength of regularization can be inversely related to the value of 'C' chosen. This is generally a float value

'Epsilon' helps select the tube width around the hyperplane estimated. This is generally a float value

Kernel - poly, rbf, linear

Gamma - 10^{-7}, 10^{-4}

C - 1.5, 10

Epsilon - 0.1, 0.2, 0.5, 0.3

Mean absolute error for validation set: 6024359126033.198

Superimposed scatter plot for visually understanding prediction performance:

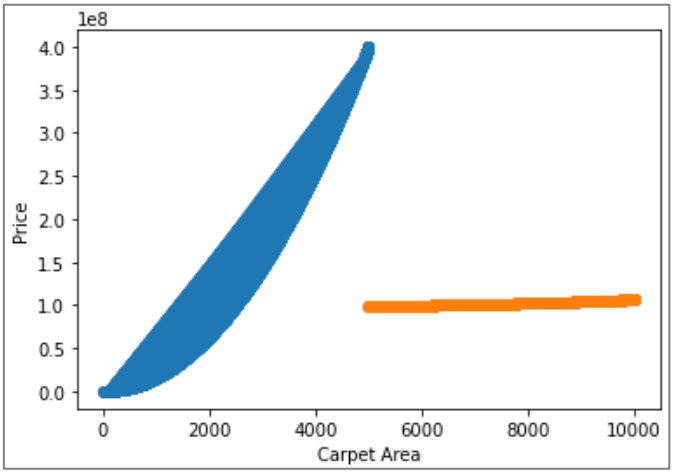


Figure 7 - Output graph for Support Vector Machine without feature engineering

Interpretation of results: The above graph demonstrates the weak predictive power of the Support Vector Machine Regressor model as there is no continuity between the actual values line and the validation dataset prediction. We can observe that the model predicts the value in a small range of values around 1 x 10^8 . The performance metric MAE also suggests a similar trend quantitatively by having a large value.

Summary: The predictive power of non-transformed predictor with the underlying relationship between predictor and target being square by the Support Vector Machine - regressor model is found to be weak

5.1.3.2. With feature engineering, with hyperparameter tuning

Hyperparameter grid optimized:

The same set of hyperparameters, as discussed in section 5.1.3.1. are passed to this model to select the most optimal set.

Mean absolute error for validation set: 1501508389.89

Superimposed scatter plot for visually understanding prediction performance:

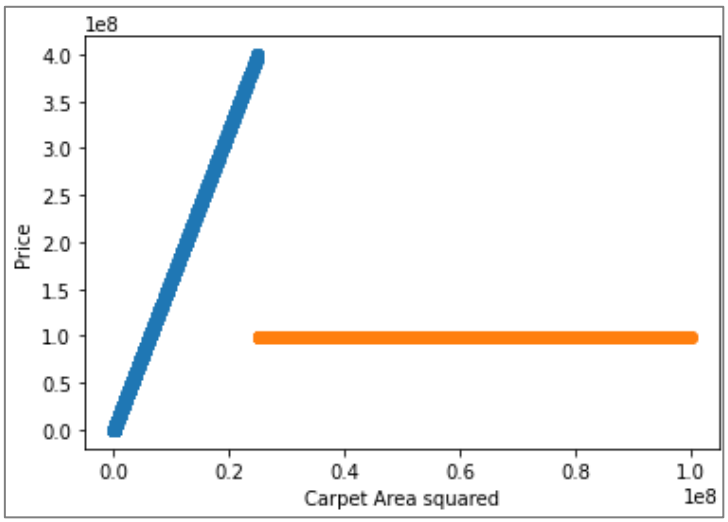


Figure 8 - Output graph for Support Vector Machine with feature engineering

Interpretation of results: The graph clearly shows no continuity between the actual target and predicted values. This behaviour is surprising as even with the assistance of a feature engineering model fails to predict the values accurately. This behaviour may be attributed to the dataset and the underlying relationship type considered for the simulation study.

Summary: Support Vector Regressor in the context of the simulation study for the dataset consisting of 1 predictor, 5000 data points, the underlying relationship being square, does not perform well when feature engineering is performed to aid the model and improve the performance.

5.1.4. LASSO regressor model

5.1.4.1. Without feature engineering, with hyperparameter tuning

Hyperparameter grid optimized:

The overall parameters considered which are applicable for the LASSO regressor model out of which the most optimal parameter is chosen are

'Alpha', this parameter controls the strength of regularization by multiplying the penalization factor L1. This value is generally non–negative, and the float value

Alpha – 0.02, 0.024, 0.025, 0.026, 0.03

Mean absolute error for validation set: 863272428.88

Superimposed scatter plot for visually understanding prediction performance:

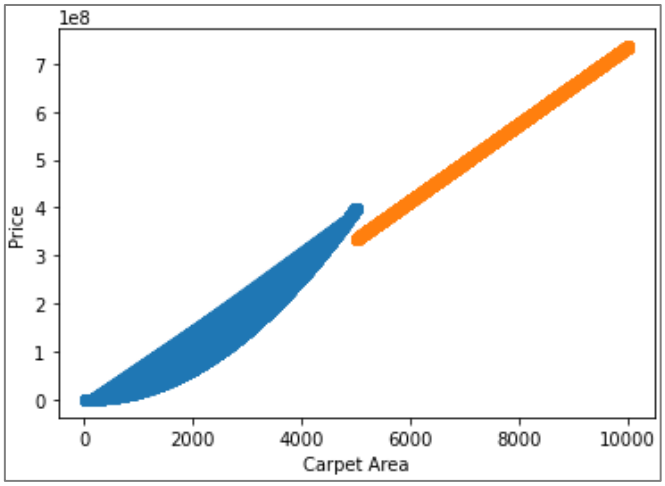


Figure 9 - Output graph for LASSO model without feature engineering

Interpretation of results: This is not a well-performing model as there is a disconnect between the actual and the predicted values, which is supposed to be a straight line. The model attempts to establish a linear relationship between the predictor and the target while it is related quadratically to the target variable. The performance metric MAE also confirms having a very high value.

Summary: LASSO regressor model does not perform very efficiently without feature engineering. It tries to establish a linear relationship between the predictor and the target.

5.1.4.2. With feature engineering, with hyperparameter tuning

Hyperparameter grid optimized:

The same set of hyperparameters, as discussed in section 5.1.4.1. are passed to this model to select the most optimal set.

Mean absolute error for validation set: 1.17×10^{-8}

Superimposed scatter plot for visually understanding prediction performance:

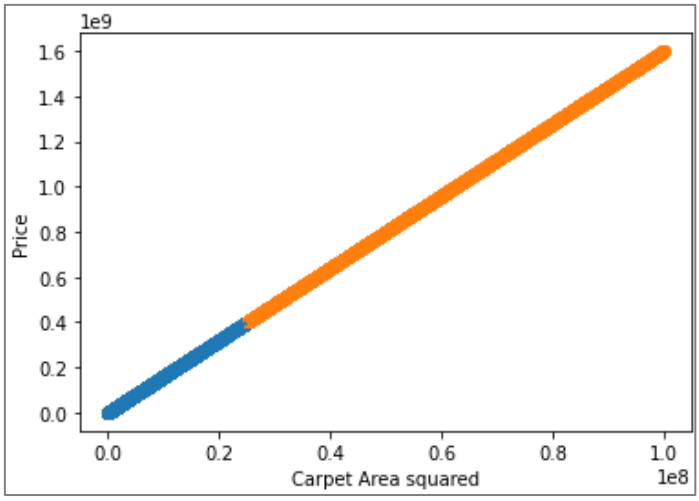


Figure 10 - Output graph for LASSO model with feature engineering

Interpretation of results: The graph depicts the strong predictive performance of the LASSO regression model since there is continuity between the actual and predicted values. This performance shows that the model has understood the proper underlying relation and correctly estimated the coefficient's value. Quantitatively this is proved with the performance metric MAE error value being 0. Feature engineering square transformation has aided in the boost of performance for the model.

Summary: LASSO regression model is one of the best performing models aided with feature engineering in rightly predicting the target variable. This performance is supported by the error MAE value being very close to 0.

5.1.5. Ridge regressor model

5.1.5.1. Without feature engineering, with hyperparameter tuning

Hyperparameter grid optimized:

The overall parameters considered which are applicable for the Ridge regressor model out of which the most optimal parameter is chosen are

'Alpha', this parameter controls the strength of regularization by multiplying the penalization factor L2. This value is generally non–negative, and float value

Alpha – 0.02, 0.024, 0.025, 0.026, 0.03

Mean absolute error for validation set: 867856937.97

Superimposed scatter plot for visually understanding prediction performance:

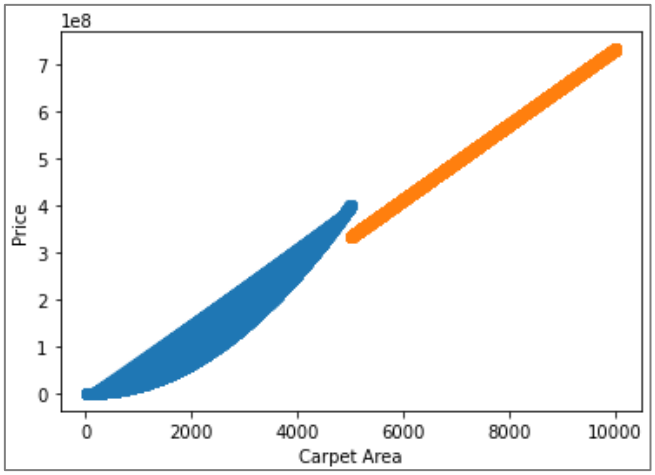


Figure 11 - Output graph for Ridge model without feature engineering

Interpretation of the results: The graph shows a disconnect between actual and predicted values. The trend closely follows the LASSO regression model, as is expected. The algorithm tries to establish a linear relationship while the underlying one is quadratic. The performance metric MAE value is also substantial, confirming the observations in the graph

Summary: Ridge regression model does not perform very efficiently without the aid of feature engineering, for the underlying relationship between predictor and target is quadratic

5.1.5.2. With feature engineering, with hyperparameter tuning

Hyperparameter grid optimized:

The same set of hyperparameters, as discussed in section 5.1.5.1. are passed to this model to select the most optimal set.

Mean absolute error for validation set: 8.71×10^{-9}

Superimposed scatter plot for visually understanding prediction performance:

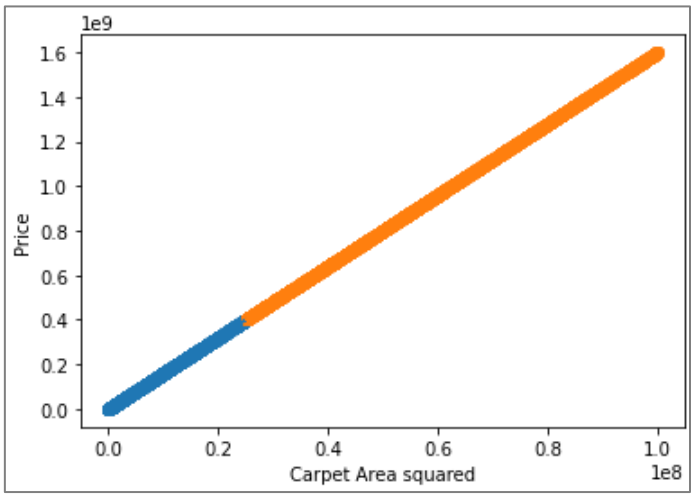


Figure 12 - Output graph for Ridge model with feature engineering

Interpretation of the result: The graph depicts the strong predictive performance of the ridge regressor model. This performance is also quantitatively proved, with the error value being very close to 0. The performance of the Ridge regression model is very similar to the LASSO model, as is expected. The constraints applied to the coefficients are similar in lasso regression, considering the penalty factor.

Summary: Ridge regression model aided with feature engineering successfully accurately predicts the target variable with a high degree of accuracy. Along with the LASSO regression model, these are both best-performing models.

5.1.6. Random forest regressor model

5.1.6.1. Without feature engineering, with hyperparameter tuning

Hyperparameter grid optimized:

The overall parameters considered which are applicable for the Random Forest regressor model out of which the most optimal parameter is chosen are

'Number of estimators', this parameter determines the total number of regression trees that are to be considered in the forest. This generally accepts an integer value

'Max depth' determines the max number of nodes that can be considered in a tree. This generally accepts an integer value

'Maximum features' determines the maximum number of features that can be considered in each tree node. This generally accepts integer or float values. Also, some options available are the square root of the total features (sqrt) or Automatically select the max features (Auto)

'Minimum samples leaf' is the parameter that sets the lower limit to the number of samples that can be considered in each node. This generally accepts integer or float value

'Minimum samples split' is the parameter that determines the lower limit to the number of samples needed to split the node. This generally accepts integer or float value

Number of estimators - 500, 600

Max depth - 15, 20, 30

Max features - auto, sqrt

Min samples leaf - 1, 2, 4

Min samples split - 2, 5, 10

Mean absolute error for validation set: 1200209385.63

Superimposed scatter plot for visually understanding prediction performance:

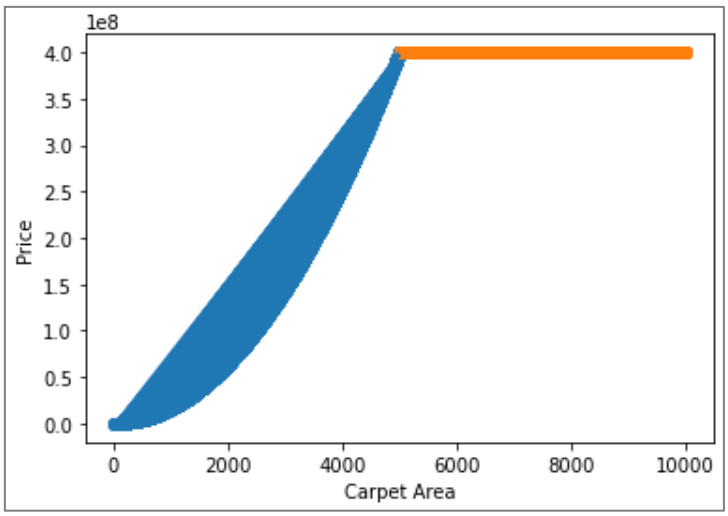


Figure 13 - Output graph for Random Forest model without feature engineering

Interpretation of results: The above graph clearly shows that the model is not performing well as a discontinuity is observed in the prediction and actual values. This behaviour is also confirmed by a very high value of the performance metric MAE. We can observe that the model predicts a single value for the validation dataset. This phenomenon is explained in the next section.

Summary: We can see that the random forest regressor fails to predict the correct value. The reason that causes this phenomenon is due to the inner working mechanism of the model. In

our simulation study design, we have constructed it so that the validation dataset does not have any value common to the training dataset. This design is done to avoid exposing the model to results and hence data leakage, which may occur due to overfitting of the model. Since random forest is an ensemble of multiple decision trees with previously unknown values in the training set, extrapolation of new value fails, and the model tries to predict an average value of the unknown value known to the model. Hence, we can observe that the model predicts a constant value. (Zhang, Nettleton and Zhu, 2019)

5.1.6.2. With feature engineering, with hyperparameter tuning

Hyperparameter grid optimized:

The same set of hyperparameters, as discussed in section 5.1.6.1. are passed to this model to select the most optimal set.

Mean absolute error for validation set: 1200180295.16

Superimposed scatter plot for visually understanding prediction performance:

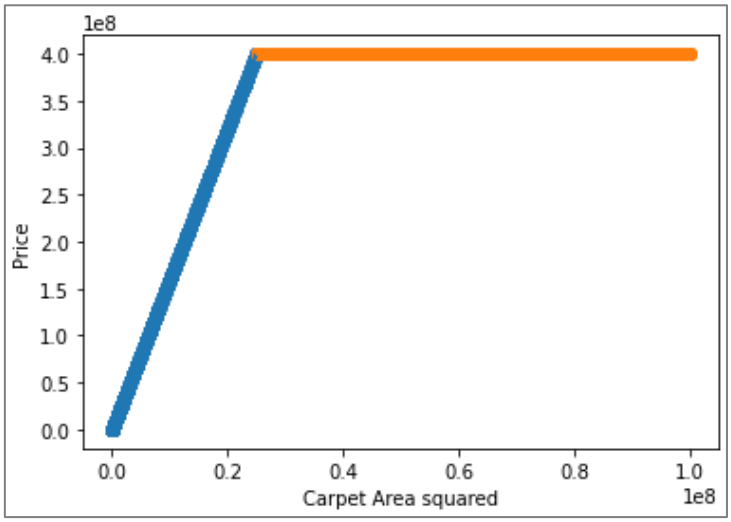


Figure 14 - Output graph for Random Forest model with feature engineering

Interpretation of the results: We can observe from the above graph that even with feature engineering, the random forest regressor model fails to predict the correct values for the target variables appropriately. This conclusion is drawn since there is no continuity between the actual values function and the predicted. This behaviour is also quantitatively confirmed with a very high value of performance metric MAE.

Summary: The random forest fails to predict the correct value for the target variable irrespective of the applied transformation via feature engineering. This behaviour is due to the shortcomings in the random forest regressor model, which is explained in detail in section 5.1.7.1

5.1.7. K – Nearest neighbours' regressor model

5.1.7.1. Without feature engineering, with hyperparameter tuning

Hyperparameter grid optimized:

The overall parameters considered which are applicable for the K Nearest neighbours regressor model out of which the most optimal parameter is chosen are
'Weights' parameter determines the weighing function used. Some of the available options are unform weights where all points considered by the model are weighted equally, distance weights with an inverse relation to the distance

'Algorithm' parameter determines the computing algorithm to find the nearest neighbour. The options for this parameter are an automatic selection of the best algorithm and a brute force search-based algorithm.
'Number of neighbours' determines the total number of neighbours to be considered in the vicinity of the query point. This generally accepts integer values

Weights - uniform, distance

Algorithm - auto, brute

Number of neighbors - 5, 7, 10, 15, 20, 25, 30, 33, 36, 40

Mean absolute error for validation set: 1201229351.37

Superimposed scatter plot for visually understanding prediction performance:

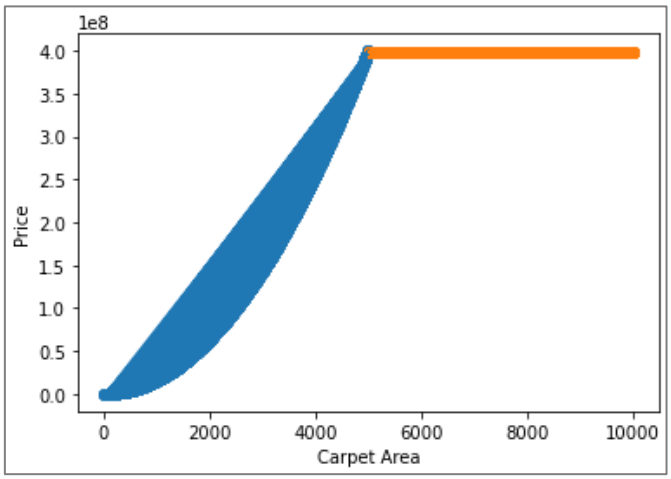


Figure 15 - Output graph for KNN model without feature engineering

Interpretation of results: The above clearly shows the failure of K neighbour regressor models as there is no continuity between the actual and predicted values. This behaviour is also confirmed by the high value of the performance metric MAE. Moreover, we can see that the model predicts a single value for all the predictors. The reason for this behaviour is explained in the next section.

Summary: K Neighbors does not perform well in identifying the underlying relationship between the non-feature engineered predictor and the target variable. The reason for this specific behaviour of the K neighbours' model, where we see the single prediction of the target variable, is due to the internal working mechanism of the K neighbours' model. To understand this further, the way K neighbours work is that the model, based on its training experience, tries to estimate the new values. However, in our case, where the dataset is designed so that the training and validation dataset are mutually exclusive, this causes the model to estimate a single value based on the maximum value it previously encountered.

5.1.7.2. With feature engineering, with hyperparameter tuning

Hyperparameter grid optimized:

The same set of hyperparameters, as discussed in section 5.1.7.1. are passed to this model to select the most optimal set.

Mean absolute error for validation set: 1200575331.18

Superimposed scatter plot for visually understanding prediction performance:

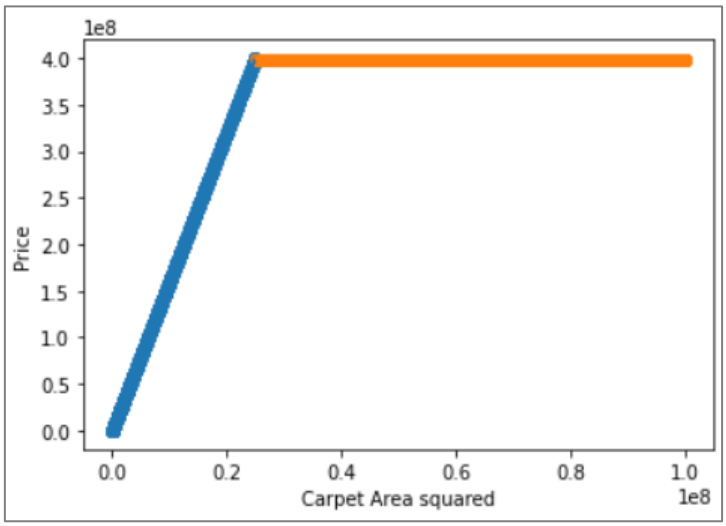


Figure 16 - Output graph for KNN model with feature engineering

Interpretation of results: From the above graph, the predictive performance of the K nearest neighbours is not strong as there is no strong continuity along the diagonal. The model predicts a single value for the validation set. The performance is also validated by the performance metric MAE having a very high value.

Summary: Due to the internal working mechanism of K nearest neighbours, as explained in section 5.1.8.1, the model does not work well even with the aid of feature engineering.

5.1.8. Conclusion of the simulation study

Regression models	**Error with hyperparameter tuning but no feature engineering**	**Error with feature engineering**
Linear regression	60225.81	1E-12
Neural network - Multilayer perceptron	1002181994	57.67
Support vector machine	6.02436E+12	1501508390
LASSO	863272428.9	1.17E-08
Ridge	867856938	0.00000000871
Random forest regressor	1200209386	1200180295
KNN	1201229351	1200575331

Table 1 - Comparison of MAE values for all considered models

5.1.8.1. Reducible error ceiling

The simulation study proved the presence of the maximum extent of error value which can be optimized without the aid of feature engineering. From the observations in the table, we can conclude that none of the models taken into consideration could break the ceiling and achieve the optimal result without the aid of feature engineering under the constraints of the simulation with a single predictor with the underlying relationship being quadratic and normally distributed error.

When the feature engineering was applied, a drastic performance improvement was noticed in the models, which could easily pick up the underlying relationship, and accurately predict the coefficient of the variable. This improvement shows that irreducible errors for models without feature engineering were converted to reducible errors. The function was further optimized for the correct prediction of the target variable. This study demonstrates the inner working of feature engineering. It proves how it is beneficial and essential to apply feature engineering to predictive modelling when there is a significant amount of error which cannot be further reduced with hyperparameter tuning. This study signifies the importance of feature engineering in machine learning, which can sometimes be the critical driver of performance.

5.1.9. Model behaviour

The model behaviour observed as a by-product of the study is worth mentioning. There are four models – Linear regression, Neural network – Multilayer perceptron, LASSO regression, and Ridge regression, which are perceptive to the underlying relationship between predictor and target when feature engineering is applied. There are three models – Support vector machine, Random Forest regressor & K Nearest Neighbors which are not performing efficiently. The main reason, as explained above, can be attributed to 2 reasons, mainly

i) The dataset design: The dataset has been carefully designed such that there is no overlapping between the training and validation dataset, and nowhere in the process is there exposure to the validation dataset. Hence the validation dataset consists of new values which are not previously trained. This design has been done so that the model successfully interprets the underlying relationship and the model does not overfit or data leakage occurs.

ii) The inner working mechanism of the model: As explained in sections 5.1.7 & 5.1.8, the inner working mechanism for estimating the predictor varies heavily

from model to model. Hence, we can understand the failure of these models even with feature engineering due to this reason.

5.2. Feature engineering and feature selection – Survey

5.2.1. Numerical predictor encoding

5.2.1.1. Box-Cox transformation

Category of feature engineering: 1:1 transformation – Individual predictor transformation resulting in a single newly derived feature

Learning type: Unsupervised

Python package for implementation (scikit-learn, 2022**):**

Module: Sklearn

Function: `"sklearn.preprocessing.power_transform()"`

Parameter: `'method': 'box-cox'`

The working mechanism of the method

This method belongs to Tukey's family of power transformation methods initially introduced (Tukey, 1957).

$$y_i^{(\lambda)} = \begin{cases} y_i^{\lambda}; & \lambda \neq 0 \\ \log y_i; & \lambda = 0 \end{cases}$$

This transformation was modified by Box & Cox in 1964 to take into consideration the discontinuity that happens at λ = 0. Hence the Box & Cox transformation was finally formulated as follows (Sakia, 2022)

$$y_i^{\lambda} = \begin{cases} (y_i^{\lambda} - 1)/\lambda; & \lambda \neq 0 \\ \log y_i; & \lambda = 0 \end{cases}$$ for the unknown value of λ $$\mathbf{y}^{(\lambda)} = (y_1^{(\lambda)}, y_2^{(\lambda)}, \ldots, y_n^{(\lambda)})' = \mathbf{X}\boldsymbol{\theta} + \boldsymbol{\varepsilon}$$

The lambda value (λ) is defined to calculate the most optimal value for normal distribution. It is estimated using the maximum likelihood technique. (Kuhn and Johnson, 2019)

The function of the method

Improve the model's predictive performance by making the distribution of the required variable, which is not normally distributed and make it normally distributed.

Why make data normally distributed? – In statistical techniques, the errors are assumed to be distributed normally for the hypothesis tests as they aid in creating confidence intervals. Hence, the target variable transformation aim is to aid in the normalization of the error values. The second key reason for making values normally distributed is to improve the model's predictive performance.

Application of the feature engineering method

This method works efficiently for continuous variables having a considerable amount of skewness.

Limitations of the feature engineering method

The method applies only to non–negative numbers due to the presence of a log function in the working mechanism of the transformation

Methods of overcoming the limitation of the feature engineering method

Add constant value to ensure all values are non–negative

5.2.1.2. Logit transformation

Category of feature engineering: 1:1 transformation – Individual predictor transformation resulting in a single newly derived feature

Learning type: Unsupervised

Python package for implementation (scikit-learn, 2022**):**

Module: Scipy

Function: **"`scipy.special.logit()`"**

The working mechanism of the method

This transformation uses a logarithm of the odds ratio. The proportion value is divided over (1 – Proportion), and the log of the resulting value is taken. This transformation can be formulated as below (Baum, 2008)

$$logit(p) = ln\left(\frac{p}{1 - p}\right)$$

Where p is the probability,

p/ (1 – p) is the odds ratio

ln is the natural log

The function of the method

This method helps transform variables between values 0 & 1 from negative to positive infinity (Ramasubramanian and Singh, 2016). This transformation can aid in modelling as the values in a smaller scale may be ignored by the model, and the feature may not be contributing actively to the performance in its original scale

Application of the feature engineering method

This method shall be helpful when the variable is in proportions or percentages.

Limitations of the feature engineering method

By looking at the function, we can understand that the method cannot be used in 2 scenarios. 1st scenario is the log of a negative number which is undefined. Hence the result of (1 – p) cannot be negative. 2nd scenario is zero division error. The result of (1-p) cannot be zero.

Methods of overcoming the limitation of the feature engineering method

The above limitations can be overcome by adding or subtracting a small value from the proportion for all values in the variable to ensure that result of (1-p) is always nonnegative and greater than zero.

5.2.1.3. Standardization

Category of feature engineering: 1:1 transformation – Individual predictor transformation resulting in a single newly derived feature

Learning type: Unsupervised

Python package for implementation (scikit-learn, 2022**):**

Module: sklearn

Function: `"sklearn.preprocessing.StandardScaler"`

The working mechanism of the method

Two essential methods will be discussed in this section, with their contexts of usage and the key differences that can aid the data science practitioner in selecting the appropriate method.

Z – score standardization (Feature scaling): Is the process of rescaling the values of the variables such that they shall end up having the statistical properties of a standard normal distribution

$\mu = 0$ and $\sigma = 1$, where μ is mean of the data and σ is standard deviation

The z – score for the corresponding value of x is calculated with the below formula (Raschka, 2014)

$$z = \frac{x - \mu}{\sigma}$$

Min – Max scaling (Normalization): This method normalizes the data to a defined fixed range by the user. The data is converted to the defined range with the aid of the below formula (Krishna and Sahu, 2015)

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Where X_{norm} Is the normalized output for the corresponding value of X, X_{max} Is the user-defined maximum value, X_{min} It is the user-defined minimum value.

The function of the method

Z – score standardization (Feature scaling): Performing this method helps attain the statistical properties of the standard normal distribution, which machine learning models desire. Standardizing also helps when multiple variables comprise different measuring units by making them comparable. Some machine learning models consisting of gradient descent optimization, such as logistic regression, support vector machines and multilayer perceptron, have the general requirement of having variables in a standardized format.

Min – Max scaling (Normalization): This method results in a smaller standard deviation value, and another side effect is the outlier elimination. Since the values are capped at the minimum and maximum values, the outliers get suppressed.

Application of the feature engineering method

Z – score standardization (Feature scaling): The machine learning models use distance measures such as K nearest neighbours. It is essential that this feature engineering be performed since all the features are needed to be contributing to an equivalent measure for appropriate prediction. (Ahsan et al., 2021)

For component analysis such as Principal Component Analysis and Kernel Component Analysis which studies the direction of variance, the scales need to be similar for equal emphasis between all variables.

It is important to note that in tree-based algorithms such as decision trees and random forests, the scale of the variables does not matter. Hence these algorithms are indifferent to the feature scaling methods due to the internal working mechanism where splits are determined irrespective of their similarity to other variables

Min – Max scaling (Normalization): This method is essential for applications such as image processing, which needs pixels between a defined range of 0 and 255 (Krishna and Sahu, 2015). Also, it is worth noting that some neural network algorithms require the data to be normalized in the range of zero to one.

5.2.1.4. Non – Linear feature creation with Basis expansion

Category of feature engineering: 1: Many transformations – Individual predictor transformation resulting in multiple new features

Learning type: Unsupervised

Python package for implementation (scikit-learn, 2022**):**

Module: sklearn

Function: **“sklearn.preprocessing.PolynomialFeatures”**

The working mechanism of the method

This method defines nonlinear functions so that regression models can capture the function. The method expands on a single predictor to multiple new predictors of various polynomial multiple polynomial functions (Kuhn and Johnson, 2019). An example of basis expansion of a polynomial spline of the third degree (cubic spline) is as follows

$$f(x) = \sum_{i=1}^{3} \alpha_i f_i(x) = \alpha_1 x + \alpha_2 x^2 + \alpha_3 x^3$$

Where α is the coefficient of the derived predictors, which can be estimated by the regression. x is the value of the predictor variable.

A more complex function covers many splines, such as piecewise polynomials, natural splines, and smoothing.

The function of the method

The feature engineering discussed above is mainly used to describe non-linearities so that the model can fit into regression models. Basis expansion effectively helps detect the underlying relationship by exposing the various forms (family of expressions) of the predictor variable via the available splines. This method helps the model fit a variety of shapes to provide strong predictive performance. The balance between the overfitting and flexibility of the model should be taken care of while building the model.

Application of the feature engineering method

This method is suitable for regression which is a simple yet potent method which is highly intuitive for understanding the variation in predictors that lead to the predictor value. However, as the name suggests, linear regression is restricted to identifying linear patterns in the data. To overcome this barrier, basis expansion can effectively consider where the predictors are transformed so that the algorithm can estimate the transformed variables linearly. Hence, the feature engineering method of basis expansion can utilize the simple yet one of the most powerful algorithms in machine learning. (Drury, 2017)

Basis expansion can also be used effectively in univariate time series forecasting (Oreshkin et al., 2019). It has been seen to offer flexibility and perform well on a wide array of forecasting problems.

Limitations of the feature engineering method

The main limitation of this method is the usage of various complex splines, which provide a high level of flexibility, leading to the overfitting of the model

Methods of overcoming the limitation of the feature engineering method

Care is taken while implementing complex functions such that the flexibility of the functions is compromised to avoid the overfitting of the model.

5.2.1.5. Principal Component Analysis (PCA)

Category of feature engineering: Many: Many transformations – Multiple predictors transformed to result in multiple new features

Learning type: Unsupervised

Python package for implementation (scikit-learn, 2022**):**

Module: sklearn

Function: **"`sklearn.decomposition.PCA`"**

The working mechanism of the method

This method reduces the dimensionality of the dataset by projecting existing data orthogonally into lower dimensional spaces. (Shlens, 2014)

This method works by creating new components in an iterative sequence where the first component explains the highest variance of the data, and the subsequent component explains the variance of the remaining data. A graph of explained variance against the number of features is constructed to understand the percentage variance change (Harrington et al., 2005). Based on the graph, the noise element in the data set, which explains variance close to 0%, is eliminated.

As discussed in section 5.2.1.3, standardisation of the features must be carried out for the best performance of the component analysis.

The principal component values can be derived from multiple methods. The most used method is performing eigen decomposition of the covariance matrix. This method is based on linear algebra. (Abdi & Williams, 2010)

Application of the feature engineering method

There are multiple functions of Principal Component Analysis, which are not only restricted to feature engineering purposes. Performing this analysis helps reduce the dimensionality, which can help visualize the problem by reducing a large number of variables into 2- or 3 -dimensional space.

Component analysis can help denoise the dataset by eliminating the components that do not contribute significantly to explaining the variance in the dataset. Performing PCA and reducing the dimensionality can help aid the machine learning model by improving the performance and speed, which may be bogged down with the massive number of variables.

Limitations of the feature engineering method

There are three main concerns while using principal component analysis, which are -

Loss of interpretability with the resulting components - The initial variables would be defined as the characteristics of the target variable, which can be interpreted and understood after the machine learning model identifies the underlying relationship. However, this power is lost when PCA is performed as the components are linear combinations of variables to explain the highest amount of variance.

Loss of information - while eliminating variables that do not significantly contribute to explaining the variance, the accompanying data is also eliminated with it, compromising the improvement of the model's performance.

The linear relationship is assumed based on the linear correlation between the predictor variables. The model fails when the genuine underlying relationship is nonlinear.

Methods of overcoming the limitation of the feature engineering method

Usage of PCA has to be avoided when interpretation is the main requirement of constructing the machine learning model, i.e., to understand the contribution of the predictors to the target variable quantitatively.

Regarding information loss, the compromise between data loss and explained variance must be carefully selected so that reasonable data loss, as determined by the practitioner, is compromised with model performance.

5.2.1.6. Kernel Principal Component Analysis

Category of feature engineering: Many: Many transformations – Multiple predictors transformed to result in multiple new features

Learning type: Unsupervised

Python package for implementation (scikit-learn, 2022**):**

Module: sklearn

Function: **“`sklearn.decomposition.KernelPCA`”**

The working mechanism of the method

This method augments the shortcomings of Principal Component Analysis by being compatible with the underlying nonlinear relationship between the predictor and the target variable. PCA successfully works for predictor variables which are linearly correlated.

Internally the kernel functions are added along with internal PCA mathematical functions. These kernel functions trick the standard PCA method to expand the predictor space dimension to include nonlinear functions (Schölkopf et al., 1998) (Shawe-Taylor and Cristianini, 2004).

Application of the feature engineering method

This method is helpful for data where the underlying relationship may not be linear, and there is a weak linear correlation between the predictors. Another scope of usage is when the standard Principal Component Analysis does not give optimal results. Different kernels can be substituted and attempted to obtain the optimal representation of the predictor to improve the predictive model's performance.

Limitations of the feature engineering method

This method also carries over the limitations similar to Principal Component Analysis as explained in section 5.2.1.5, i.e., loss of information and data loss, loss of interpretability.

Methods of overcoming the limitations

Similar methods as suggested in section 5.2.1.5 for the principal component analysis shall be followed for kernel Principal Component Analysis

5.2.2. Categorical predictor encoding

5.2.2.1. Dummy variables creation

Category of feature engineering: Handling unordered categories in a variable with fewer categories

Learning type: Unsupervised

Python package for implementation (scikit-learn, 2022**):**

Module: sklearn

Function: **"`sklearn.preprocessing.OneHotEncoder`"**

The working mechanism of the method

The internal working of this function can be described as follows, one of the categories in the variable is taken as a reference, contrast is dropped, and the rest of the categories are converted into new binary variables (Kunanbayev, Temirbek and Zollanvari, 2021). The working of this method is demonstrated below

Types of fruits
Apple Reference
Orange
Banana
Mango
Guava

Orange	**Banana**	**Mango**	**Guava**
0	0	0	0
1	0	0	0
0	1	0	0
0	0	0	1

Table 2 - Demonstration of the working mechanism of One Hot Encoding

The function of the method

Algorithms such as linear regression, neural networks and more models usually seek inputs in numerical format as they cannot process categorical features directly due to their inner workings. Hence the categorical variable must be quantified suitably for the feature to be considered in the machine learning model (Gupta and Asha, 2020). For categorical variables with unordered categories, the method of using dummy variables is one of the most common methods used in the machine learning data pre-processing feature engineering phase. It is also worth noting that the number of features dramatically increases when this is applied to all the categorical variables in the dataset. The new features that will be added are equal to the total categories present in all the variables minus the reference contrast variable.

Application of the feature engineering method

This method applies to models that cannot accept categorical features, such as linear regression, LASSO, Ridge, Neural networks and deep learning models, and Support Vector Machines (SVM). The methods such as Classification And Regression Trees (CART) and random forest can at least consider the representation of the categorical variable.

Limitations of the feature engineering method

One of the critical limitations of this method occurs when there are many categories in a categorical variable. When all the categories are converted into binary dummy variables except the reference contrast, the total number of features increases heavily. This added feature burdens the machine learning model, making the data complicated and cumbersome to understand, which may increase the computational power and time needed.

Overcoming the limitations of the feature engineering method

More feature engineering methods are better suited to handle more categories, such as feature hashing; this shall be explained in detail in the next section.

5.2.2.2. Feature hashing

Category of feature engineering: Handling unordered categories in a variable with many categories

Learning type: Unsupervised

Python package for implementation (scikit-learn, 2022**):**

Module: sklearn

Function: `"sklearn.feature_extraction.FeatureHasher"`

The working mechanism of the method

The inputs required are categories and the total number of resulting features needed. There is no need for a contrast function. The initial categories in the features are reduced into a lower number of hashes derived from the text data. The hashing process is deterministic, and since there is a reduction in the number of categories, there are possibilities of clashes occurring within the categories, i.e., the same categories being mapped into one hash feature (Weinberger et al., 2022).

The above-discussed hash features are then converted into dummy binary variables

The function of the method

This method is applicable for categorical variables consisting of a very long list of categories, such as the zip code of a country/state.

Applications of the feature engineering method

This feature engineering method also helps in a problem usually faced when dealing with large numbers of categories. A category with a lower frequency of occurrence in the categorical variable is more likely to be omitted from the dataset, as resampling may omit the categories occurring less (Kuhn and Johnson, 2019).

Limitations of the feature engineering method

The hashes created via this method are generally irreversible, i.e., there is no way of knowing the original values used to create these hashes.

Due to the reduction in the categories taken into consideration, the downside is the collision of categories, i.e., the grouping of categories into a single hash value. This grouping leads to a loss of data.

Overcoming the limitations of the feature engineering method

The hash functions can be signed, i.e., values such as -1, +1 and 0 can be created to indicate the various values occurring for the same hash code. Taking this step helps avoid the collision of the data points with a single hash function.

5.2.2.3. Likelihood encoding

Category of feature engineering: Handling unordered categories in a variable with many categories

Learning type: Supervised

Python package for implementation (scikit-learn, 2022**):**

Module: sklearn

Function: `"sklearn.linear_model.LogisticRegression"`

The working mechanism of the method

This is a supervised feature engineering method in which the training includes the target variable. Logistic regression is performed to identify the odds ratio of each category against the categorical target. The odds ratio can be defined as the strength of association between 2 events (Clayton, 1974). It represents the odds or the chance of event A happening in the presence of event B. The odds ratio can be defined as

$$OR = \frac{p}{1-p}$$

Here OR is the odds ratio, p is the odds of the events happening in the exposed group and 1 – p is the odds of happening in the non-exposed group.

The function of the method

This method helps quantify categories of categorical variables. This method is supervised, which can be worth considering while attempting to improve the predictive model's performance against other categorical variables encoding methods.

Application of the feature engineering method

This method applies to the classification type of machine learning problems consisting of categorical variables.

Limitations of the feature engineering method

This method is mainly applicable to the classification type of models

Overcoming the limitations of the feature engineering method

More supervised methods, such as simple generalized linear models, use both logistic and linear regression to encode the categorical values of the feature.

5.2.2.4. Feature creation from text data tf-idf statistics

Category of feature engineering: Handling unordered categories in a variable with many categories

Learning type: Supervised

Python package for implementation (scikit-learn, 2022**):**

Module: sklearn

Function: **"sklearn.feature_extraction.text.TfidfTransformer"**

The working mechanism of the method

The term frequency (tf) is calculated by calculating the number of times a word occurs in literature or a document. The inverse document frequency (idf) value is the normalization weight in the complete collection of documents. These statistics can be calculated with the below formula (Liu et al., 2018)

$$tf = \frac{Total\ occurence\ of\ term\ m\ in\ a\ document\ d}{Total\ number\ of\ terms\ in\ the\ document\ d}$$

$$idf = log_e \frac{Total\ documents}{Number\ of\ documents\ with\ term\ t\ within}$$

The function of the method

The text is analyzed to identify keywords that may influence the prediction of the outcome. The statistic helps quantify the key text such that if any correlation exists between the occurrence or the non-occurrence of the text with the target variable prediction, it is exposed, and the modelling process is aided with performance improvement.

Application of the feature engineering method

This method is handy when the dataset consists of open-ended answers from the user, such as a survey or feedback. These answers can reflect the reason for predicting the outcome, hence being a crucial predictor, which would be ignored otherwise if not for tf – idf statistics.

Limitations of the feature engineering method

The identification of relevant keywords which may have an impact on the predictive performance can be a very time-consuming procedure. Adding many text data, especially when there is a vast number of observations, can make the dataset huge.

Another critical issue is that when the model is exposed to unseen or new data, the method shall be invalid for these cases as the statistic would not be available for the fresh set of words.

As only the words are considered for analysis, a sequence of words which may mean different things may be omitted from the analysis.

Overcoming the limitations of the feature engineering method

Stemming and lemmatization can be done before calculating tf-idf statistics so that misinterpretation of the sequence of words is considered.

Contexts where new data shall be expected to be predicted, should be avoided while using this feature engineering method.

5.2.3. Feature selection methods

Feature selection is an integral part of the machine learning model due to reasons mainly (Kuhn & Johnson, 2019)

A. Machine learning models such as Support Vector Machines and Artificial Neural Networks do not work efficiently in the presence of predictors, which contribute weakly to the prediction. In fact, in some cases, they affect negatively by reducing the performance score of the model.
B. Regression-based models such as linear regression and logistic regression consist of multicollinearity assumption, which states the predictors are assumed not to be correlated with each other. However, feature selection can be performed to eliminate correlated variables and disassociated by eliminating the rest and selecting the most important.
C. Excess predictors non - contributing to the prediction of the model can be detrimental to the data collection efforts. Resources such as time and money might have been spent on collecting the data only to be eliminated in the modelling process, hence feature selection helps determine these detrimental variables and guide the practitioner to avoid collecting this information, saving money and time.

Performing feature selection is broadly classified into four main categories

1. Filter methods
2. Wrapper methods
3. Embedded methods
4. Hybrid methods

All these categories and essential methods with their comprehensive analysis shall be discussed in detail in the following sections

5.2.3.1. Filter methods

This method mainly works on the standard characteristics of the data used for training without any dependence on other predictors (Sánchez-Maroño, Alonso-Betanzos and Tombilla-Sanromán, 2022). The methods covered under this category are usually less taxing on computational power than other feature selection categories.

For a dataset consisting of many features, the filter method can be combined with other methods such that an initial level of preprocessing is performed before, and better computationally, more taxing methods are applied.

5.2.3.2. Chi-Square test

Category of feature engineering: Filter method

Learning type: Supervised

Python package for implementation (scikit-learn, 2022**):**

Module: sklearn

Function: `"sklearn.feature_selection.chi2"`

`"sklearn.feature_selection.SelectKBest"`

The working mechanism of the method

The method uses the statistical technique chi-square test. This test is performed to understand whether two variables are independent or dependent. The formula for the chi-square test is as below (Franke, Ho and Christie, 2011)

$$X_c^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

Here c denotes the total degree of freedom, O denotes the values observed & E denotes the expected value.

The function of the feature selection method

This method works based on the output value obtained for the chi-square value. If the chi-square value is minimal, that would mean the difference between the observed and the expected value is less, hence independent of each other, as the values are close to each other. If the value

is very high, this would mean there is a significant difference between the observed and expected value hence showing their dependency on each other (Thaseen, Kumar and Ahmad, 2018).

Hence, the features most dependent on the target variable are selected as they are most relevant in providing us with the prediction.

Application of this method

This method applies to the categorical features in the dataset

5.2.3.3. Correlation coefficient

Category of feature selection: Filter method

Learning type: Supervised

Python package for implementation (scikit-learn, 2022**):**

Module: Pandas

Function: "`pandas.DataFrame.corr()`"

The working mechanism of the method

This method uses the statistical technique of finding a correlation coefficient between 2 variables by quantifying the strength of their linear relationship. The formula can be defined below (Asuero, Sayago and Gonzalez, 2007)

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 (y_i - \bar{y})^2}}$$

Here, r is the correlation coefficient

x_i is the value of the variable x

$\bar{x}$ is the mean value of the variable x

y_i is the value of the variable y

$\bar{y}$ is the mean value of the variable y

The function of the feature selection method

The above-described correlation coefficient method provides us with the quantitative strength of the linear relationship between the variables within and with the target variable. For feature selection consideration, we consider the correlation between the target variable and the other variables (Hsu and Hsieh, 2010). A minimum correlation coefficient value threshold is kept below which the variables are discarded from the dataset.

Application of the feature selection method

The Pearson correlation method explained above is applicable for features containing continuous values and does not apply to categorical features.

5.2.3.4. Wrapper methods

As we have already understood, feature engineering and feature selection is mostly a trial and error process which may work in some scenarios and may not work in others. The wrapper method is a kind of brute force method which iterates through the possible combinations of feature subsets from all the features and, based on a defined greedy or non-greedy mechanism, evaluates the model performance and identifies the best possible combination of features (Maldonado and Weber, 2009).

As expected, these methods involve the usage of much computational power. Since there is extensive training involved in identifying the most optimal features amongst all the features, this method also comes with a high chance of overfitting.

5.2.3.5. Greedy wrapper method - Recursive feature elimination RFE (Backwards selection)

Category of feature selection: Wrapper method

Learning type: Supervised

Python package for implementation (scikit-learn, 2022**):**

Module: sklearn

Function: `"sklearn.feature_selection.RFE"`

The working mechanism of the method

This method considers feature importance determining method as an input, and the total number of features that need to remain in the model. It initially runs the model to identify the feature importance value and trims the features at the bottom of the list with low importance

values. The model is run iteratively with the subsequent subsets, and the features are trimmed continuously until the user determines the target number of features (Chen and Jeong, 2007).

The function of the feature selection method

This method works backwards by initially having all the features and ending up with reduced features. This method finds a local best setting and immediately fixates on identifying the best set of features. This method is called greedy since the search progresses forward by identifying the best feature ranking out of the remaining (Zhou et al., 2014). These methods do not consider comparing the overall performance of the model between features of low and high importance.

Application of the method

As discussed in the previous section, this method fails to work efficiently with features of varying importance levels. The interaction within the features is most likely going to be ignored by this model. Hence if previously known regarding the within feature interactions, this feature selection method is to be avoided.

5.2.3.6. Non – Greedy wrapper method - Genetic Algorithms

Category of feature selection: Wrapper method

Learning type: Supervised

Python package for implementation (scikit-learn, 2022**):**

Module: genetic_selection

Function: **“`GeneticSelectionCV()`”**

The working mechanism of the method

This method takes inspiration from the biological principles of evolution theory. Evolution is simulated through the genetic algorithm, and ultimately the fittest or the best set of features are left surviving, which provides the best performance of the model. Initially, a population of a set of features is defined; further, multiple subsets of features are created from this population, and their performance is determined with a machine learning model. Then a tournament of the subsets is conducted to identify the features that would progress onto the next generation. A mutation is also introduced to mimic the genetic evolution process by introducing and removing random features in the 2nd generation of features. The sequence of the above steps is

carried out over multiple generations to ultimately end up with the most optimal set of features (Babatunde et al., 2014).

The function of the method

This method has a unique way of evaluating and determining the best set of features for the model. This method is highly versatile as it has improved model performance (Warnes, 2021) compared to baseline performance and other feature selection methods.

5.2.3.7. Embedded method - LASSO regression

Category of feature selection: Embedded method

Learning type: Supervised

Python package for implementation (scikit-learn, 2022**):**

Module: sklearn

Function: `"sklearn.linear_model.Lasso()"`

The working mechanism of the method

Embedded methods are those where the machine learning models have an inbuilt mechanism for eliminating the least contributing features by using penalty functions that quantify the feature importance and filter out the unwanted (Lal et al., 2022).

This method can also be considered a combination of filter and wrapper methods as iteratively, feature importance is evaluated, and the least important features are filtered out based on a threshold value.

The function of the method

L1 regularization value is considered inside the LASSO regression, which calculates the penalty value based on the coefficients' magnitude (Muthukrishnan and Rohini, 2016).

As these methods are embedded within the models, they are utilized natively by the model by default. The penalizing function can be changed with hyperparameter tuning based on the available options. These methods' efforts are significantly less as they are directly integrated into the machine learning model. There is no need for added coding efforts other than passing the right penalizing parameter to the model.

5.2.3.8. Hybrid feature selection methods

The methods covered in this section utilise the advantages of filters and wrapper methods (Hsu, Hsieh and Lu, 2011). Filter methods are rapid for executing the task on a large dataset; however, they are not the best set of methods for providing the best performance of the models. Wrapper methods are excellent in identifying the best set of features. However, due to their mechanism in which they work iteratively in identifying the best feature subset out of all the features, they take a lot of computational power and time. By combining filter and wrapper methods, hybrid methods are created where the model is initially fed into the filter method to perform higher-level filtering of the features. Post this step, the features are fed into the wrapper method to identify the most optimal set of features that gives the model's best performance. As we can understand, this method uses the filter method to reduce the burden on the wrapper method, which ultimately carries out the final selection of the relevant features.

6. Conclusion

The significance and the role of feature engineering in developing machine learning models are emphasized heavily in multiple works of literature. However, a simulation study does little to quantify and understand this significance in detail. The research presented above attempts to bridge this gap initially by providing an in-depth simulation study to understand the inner working mechanism of feature engineering. Hence, the need for deriving new features to drive machine learning model performance proceeds to detail a wide array of available feature engineering and feature selection methods to make practitioners well equipped and provide more insight into the usage of the methods.

6.1. Summary of findings

The study has been enriched with multiple findings concerning the objectives before starting research (Section 2.4). Some key findings of this research that can be summarized generally include

- Demonstration of the concept of reducible error and irreducible error without feature engineering, wherein reducible error refers to an error value that can be optimized with modelling hyperparameter tuning efforts, and irreducible error the error value which cannot be driven further down even after exhaustive various modelling type selection and extensive hyperparameter tuning

- Understood the working mechanism of feature engineering by simulating an underlying relationship between predictor and target and proving the need for feature engineering by running a machine learning model with and without feature engineering and comparing the error value.
- Comprehensive detail of primary feature engineering methods with crucial information on the working mechanism, function, application, limitation, and methods to overcome limitations. The suitable python package was identified with the function details for quick implementation of the said feature engineering methods.
- Comprehensive detail of all the categories of the feature selection method, with discussion on the benefits and drawbacks of the usage of various methods, a brief look into the inner working mechanism, function, and application of the methods.

Some added findings from research that initially were not within the scope of objectives, although they were identified as part of the research which is of significance, are listed below

- Machine learning models such as Random Forest regressor, K Nearest Neighbors regressor and Support Vector Machine regressor fail to identify the pattern and predict the values it has not been exposed to in the training process. The reason can be attributed to the internal working mechanisms as detailed in the study.

6.2. Recommendations and future work

Machine learning simulation is a knowledge-enriching experience that can be conducted to understand the behaviour of models and their exposure to various datasets with varying underlying relationships hence recommendation #1 is to expand the machine learning simulation that is demonstrated in the above research to more applications such as

- Identifying the behaviour of one type of machine learning algorithm to the various feature engineering methods. This research can be done to understand the susceptibility of a machine learning model to various underlying relationships and how the efficiency increases with feature engineering methods.
- The above simulation study considers a single predictor. More complex underlying relationships can be studied with multiple predictors with complex functions to see how different feature engineering methods help detect the underlying relationship.

- The simulation study conducted in this research has normally distributed errors. The study of the behaviour of models when exposed to various error values and their impact on the performance of machine learning models can be understood in detail.

The list compiled above is not exhaustive on the various feature engineering and feature selection methods researched from other literature on the topic. The recommendations are as below

- Cover more sophisticated feature engineering methods under continuous feature transformation such as Generalized Additive Models (GAMS), Multivariate Adaptive Regression Spline models (MARS), Non–Negative Matrix Factorization (NNMF) under categorical feature transformation such as Word – qualitative assessment, polynomial contrast encoding and more.
- The study discusses in detail the categories of feature selection methods, mainly supervised methods, less popular unsupervised methods, and their application can be studied in detail

6.3. Contributions to knowledge

Feature engineering and selection methods are often the most challenging and time-consuming processes in machine learning model development and deployment. This difficulty can be attributed to less understanding of the usage context and the time taken to identify many methods with their correct usage. This study aims to make a data science practitioner better versed in understanding the need and necessity of feature engineering and expose the practitioner to the wide range of available methods and a comprehensive summary for quick adoption of the said methods. The literature focuses on limited feature engineering methods, and discussions are based on a precisely defined dataset. Hence the contribution to knowledge expected from this study is a simulation-based study of feature engineering and a comprehensive understanding of feature engineering methods and feature selection methods.

7. Self-reflection

Working on this research has been an intellectually enriching experience as being from a different background and new to the field of machine learning. My previous knowledge was mainly theoretical, with no insight into the suitable usage of feature engineering and feature selection methods.

The previous understanding was mainly limited to random and limited application of the various feature engineering and selection method, which I have discovered to be wrong in many cases throughout this research. The understanding I have gained currently includes the time, effort and energy that can be saved by knowing the essential information related to these methods.

Working on simulated datasets has also been a very enriching experience as we usually do not have any insight or understanding of the underlying relationship between predictors and targets. Knowing the relationship and controlling it effectively to understand more the working of machine learning is something I would use further to improve my knowledge in machine learning and feature engineering.

8. Reference list

Abs.gov.au. (2015). *Statistical Language - Quantitative and Qualitative Data*. [online] Available at: https://www.abs.gov.au/websitedbs/D3310114.nsf/Home/Statistical+Language+-+quantitative+and+qualitative+data [Accessed 25 Aug. 2022].

Ahsan, M., Mahmud, M., Saha, P., Gupta, K. and Siddique, Z. (2021). Effect of Data Scaling Methods on Machine Learning Algorithms and Model Performance. *Technologies*, [online] 9(3), p.52. doi:10.3390/technologies9030052.

Alam, S. and Yao, N. (2018). The impact of preprocessing steps on the accuracy of machine learning algorithms in sentiment analysis. *Computational and Mathematical Organization Theory*, [online] 25(3), pp.319–335. doi:10.1007/s10588-018-9266-8.

Asuero, A.G., Sayago, A. and Gonzalez, A.G. (2007). *The Correlation Coefficient: An Overview*. [online] Critical Reviews in Analytical Chemistry. Available at: https://www.tandfonline.com/doi/full/10.1080/10408340500526766?casa_token=WwixNuCw4G0AAAAA%3AHyU6lxvvbYSADr9akJIzbbtP_wK74im15ZMh38bbdA08UzlrnfsilFUpHaqZjvMMzHGPR3ov7cWr1Bw [Accessed 28 Aug. 2022].

Babatunde, O.H., Armstrong, L., Leng, J. and Diepeveen, D. (2014). *A Genetic Algorithm-Based Feature Selection*. [online] Research Online. Available at: https://ro.ecu.edu.au/ecuworkspost2013/653/ [Accessed 29 Aug. 2022].

Baum, C.F. (2008). Stata Tip 63: Modeling Proportions. *The Stata Journal: Promoting communications on statistics and Stata*, 8(2), pp.299–303. doi:10.1177/1536867x0800800212.

Bocca, F.F. and Rodrigues, L.H.A. (2016). The effect of tuning, feature engineering, and feature selection in data mining applied to rainfed sugarcane yield modelling. *Computers and Electronics in Agriculture*, [online] 128, pp.67–76. doi:10.1016/j.compag.2016.08.015.

Brownlee, J. (2020). *Data Preparation for Machine Learning Data Cleaning, Feature Selection, and Data Transforms in Python*. [online] Available at: http://14.99.188.242:8080/jspui/bitstream/123456789/14557/1/Data%20Preparation%20for%20Machine%20Learning%20-%20Data%20Cleaning%2C%20Feature%20Selection%2C%20and%20Data%20by%20Jason%20Brownlee.pdf.

Brown, T. (2021). *Who is Using Big Data in Business?* [online] ITChronicles. Available at: https://itchronicles.com/big-data/who-is-using-big-data-in-business/ [Accessed 24 Aug. 2022].

Chandrashekar, G. and Sahin, F. (2014). A survey on feature selection methods. *Computers & Electrical Engineering*, [online] 40(1), pp.16–28. doi:10.1016/j.compeleceng.2013.11.024.

Chen, X. and Jeong, J.C. (2007). Enhanced recursive feature elimination. *Sixth International Conference on Machine Learning and Applications (ICMLA 2007)*. [online] doi:10.1109/icmla.2007.35.

Chen, Y., Li, B. and Ge, X. (2011). Study on Predictive Model of Customer Churn of Mobile Telecommunication Company. *2011 Fourth International Conference on Business Intelligence and Financial Engineering*. [online] doi:10.1109/bife.2011.112.

Chid Apté (2019). *The Role of Machine Learning in Business Optimization*. [online] Open-Review. Available at: https://openreview.net/forum?id=rkWCAqbd-S [Accessed 22 Aug. 2022].

Christ, M., Braun, N., Neuffer, J. and Kempa-Liehr, A.W. (2018). Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package). *Neurocomputing*, [online] 307, pp.72–77. doi:10.1016/j.neucom.2018.03.067.

Clayton, D.G. (1974). *Some odds ratio statistics for the analysis of ordered categorical data*. [online] https://academic.oup.com/biomet/article-abstract/61/3/525/249460. Available at: https://doi.org/10.1093/biomet/61.3.525.

Critical Reviews in Analytical Chemistry. (2022). *The Correlation Coefficient: An Overview*. [online] Available at: https://www.tandfonline.com/doi/full/10.1080/10408340500526766?casa_token=WwixNuCw4G0AAAAA%3AHyU6lxvvbYSADr9akJIzbbtP_wK74im15ZMh38bbdA08UzlrnfsilFUpHaqZjvMMzHGPR3ov7cWr1Bw [Accessed 28 Aug. 2022].

DASH, M. and LIU, H. (1997). Feature selection for classification. *Intelligent Data Analysis*, [online] 1(1-4), pp.131–156. doi:10.1016/s1088-467x(97)00008-5.

DE, M. and SUGIYAMA, M. (2013). Winning the Kaggle Algorithmic Trading Challenge with the Composition of Many Models and Feature Engineering. *IEICE TRANSACTIONS on*

Information and Systems, [online] E96-D(3), pp.742–745. doi:https://search.ieice.org/bin/pdf_link.php?fname=e96-d_3_742&category=D&lang=E&year=2013&abst=.

D McGinnis, W., Siu, C., S, A. and Huang, H. (2018). Category Encoders: a scikit-learn-contrib package of transformers for encoding categorical data. *The Journal of Open Source Software*, [online] 3(21), p.501. doi:10.21105/joss.00501.

Dong, G. and Liu, H. eds., 2018. *Feature engineering for machine learning and data analytics*. CRC Press.

Drury, M. (2017). *A Comparison of Basis Expansions in Regression*. [online] Github.io. Available at: http://madrury.github.io/jekyll/update/statistics/2017/08/04/basis-expansions.html [Accessed 15 Aug. 2022].

Franke, T.M., Ho, T. and Christie, C.A. (2011). The Chi-Square Test. *American Journal of Evaluation*, [online] 33(3), pp.448–458. doi:10.1177/1098214011426594.

Gupta, H. and Asha, V. (2020). Impact of Encoding of High Cardinality Categorical Data to Solve Prediction Problems. *Journal of Computational and Theoretical Nanoscience*, [online] 17(9), pp.4197–4201. doi:10.1166/jctn.2020.9044.

Harrington, P. de B., Vieira, N.E., Espinoza, J., Nien, J.K., Romero, R. and Yergey, A.L. (2005). Analysis of variance–principal component analysis: A soft tool for proteomic discovery. *Analytica Chimica Acta*, [online] 544(1-2), pp.118–127. doi:10.1016/j.aca.2005.02.042.

Heaton, J. (2016). An empirical analysis of feature engineering for predictive modeling. *SoutheastCon 2016*. [online] doi:10.1109/secon.2016.7506650.

Hsu, H.-H. and Hsieh, C.-W. (2010). Feature Selection via Correlation Coefficient Clustering. *Journal of Software*, 5(12). doi:10.4304/jsw.5.12.1371-1377.

Hsu, H.-H., Hsieh, C.-W. and Lu, M.-D. (2011). Hybrid feature selection by combining filters and wrappers. *Expert Systems with Applications*, [online] 38(7), pp.8144–8150. doi:10.1016/j.eswa.2010.12.156.

Ibe, O., 2014. *Fundamentals of applied probability and random processes*. Academic Press.

Jain, A.K., Duin, P.W. and Jianchang Mao (2000). Statistical pattern recognition: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, [online] 22(1), pp.4–37. doi:10.1109/34.824819.

James, G., Witten, D., Hastie, T. and Tibshirani, R. (2021). An Introduction to Statistical Learning. *SpringerLink*. [online] doi:10.1007-978-1-0716-1418-1.

Jovic, A., Brkic, K. and Bogunovic, N. (2015). A review of feature selection methods with applications. *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. [online] doi:10.1109/mipro.2015.7160458.

J Toby Mordkoff (2016). *The Assumption(s) of Normality*. [online] Available at: http://www2.psychology.uiowa.edu/faculty/mordkoff/GradStats/part%201/I.07%20normal.pdf.

Kaliyadan, F. and Kulkarni, V. (2019). Types of variables, descriptive statistics, and sample size. *Indian Dermatology Online Journal*, [online] 10(1), p.82. doi:10.4103/idoj.idoj_468_18.

Khalid, S., Khalil, T. and Nasreen, S. (2014). A survey of feature selection and feature extraction techniques in machine learning. *2014 Science and Information Conference*. [online] doi:10.1109/sai.2014.6918213.

Krishna and Sahu, K.K. (2015). Normalization: A Preprocessing Stage. *arXiv.org*. [online] doi:10.48550/arXiv.1503.06462.

Kuhn, M. and Johnson, K. (2019). *Feature Engineering and Selection: A Practical Approach for Predictive Models*. [online] Feat.engineering. Available at: http://www.feat.engineering/ [Accessed 24 Aug. 2022].

Kunanbayev, K., Temirbek, I. and Zollanvari, A. (2021). Complex Encoding. *2021 International Joint Conference on Neural Networks (IJCNN)*. [online] doi:10.1109/ijcnn52387.2021.9534094.

Lal, T.N., Chapelle, O., Weston, J. and Elisseeff, A. (2022a). Embedded Methods. *Feature Extraction*, [online] pp.137–165. doi:10.1007/978-3-540-35488-8_6.

Lal, T.N., Chapelle, O., Weston, J. and Elisseeff, A. (2022b). Embedded Methods. *Feature Extraction*, [online] pp.137–165. doi:10.1007/978-3-540-35488-8_6.

Lee, I. and Shin, Y.J. (2020). Machine learning for enterprises: Applications, algorithm selection, and challenges. *Business Horizons*, [online] 63(2), pp.157–170. doi:10.1016/j.bushor.2019.10.005.

Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R.P., Tang, J. and Liu, H. (2018). Feature Selection. *ACM Computing Surveys*, [online] 50(6), pp.1–45. doi:10.1145/3136625.

Liu, Q., Wang, J., Zhang, D., Yang, Y. and Wang, N. (2018). Text Features Extraction based on TF-IDF Associating Semantic. *2018 IEEE 4th International Conference on Computer and Communications (ICCC)*. [online] doi:10.1109/compcomm.2018.8780663.

Maldonado, S. and Weber, R. (2009). A wrapper method for feature selection using Support Vector Machines. *Information Sciences*, [online] 179(13), pp.2208–2217. doi:10.1016/j.ins.2009.02.014.

Marimuthu, S., Mani, T., Sudarsanam, T.D., George, S. and Jeyaseelan, L. (2022). Preferring Box-Cox transformation, instead of log transformation to convert skewed distribution of outcomes to normal in medical research. *Clinical Epidemiology and Global Health*, [online] 15, p.101043. doi:10.1016/j.cegh.2022.101043.

Maxwell, P., Alhajjar, E. and Bastian, N.D. (2019). Intelligent Feature Engineering for Cybersecurity. *2019 IEEE International Conference on Big Data (Big Data)*. [online] doi:10.1109/bigdata47090.2019.9006122.

Muthukrishnan, R. and Rohini, R. (2016). LASSO: A feature selection technique in predictive modeling for machine learning. *2016 IEEE International Conference on Advances in Computer Applications (ICACA)*. [online] doi:10.1109/icaca.2016.7887916.

Ostapchenya, D. (2021). *The Role of Big Data in Banking : How do Modern Banks Use Big Data?* [online] Finextra Research. Available at: https://www.finextra.com/blogposting/20446/the-role-of-big-data-in-banking--how-do-modern-banks-use-big-data [Accessed 22 Aug. 2022].

Phyu, T.Z. and Oo, N.N. (2016). Performance Comparison of Feature Selection Methods. *MATEC Web of Conferences*, [online] 42, p.06002. doi:10.1051/matecconf/20164206002.

Ramasubramanian, K. and Singh, A. (2016). Feature Engineering. *Machine Learning Using R*, [online] pp.181–217. doi:10.1007/978-1-4842-2334-5_5.

Raschka, S. (2014). *About Feature Scaling and Normalization*. [online] Dr. Sebastian Raschka. Available at: https://sebastianraschka.com/Articles/2014_about_feature_scaling.html [Accessed 15 Aug. 2022].

Rawat, T. and Khemchandani, V. (2017). Feature Engineering (FE) Tools and Techniques for Better Classification Performance. *International Journal of Innovations in Engineering and Technology*, 8(2). doi:10.21172/ijiet.82.024.

Sakia, R.M. (1992). The Box-Cox Transformation Technique: A Review. *The Statistician*, [online] 41(2), p.169. doi:10.2307/2348250.

Sánchez-Maroño, N., Alonso-Betanzos, A. and Tombilla-Sanromán, M. (2022). Filter Methods for Feature Selection – A Comparative Study. *Intelligent Data Engineering and Automated Learning - IDEAL 2007*, [online] pp.178–187. doi:10.1007/978-3-540-77226-2_19.

Shlens, J. (2014). A Tutorial on Principal Component Analysis. *arXiv.org*. [online] doi:10.48550/arXiv.1404.1100.

S, M. and R, G. (2021). Data mining in Ecommerce platforms for product - ProQuest. *Proquest.com*. [online] doi:10.5958/2321-581X.2021.00001.5.

Solorio-Fernández, S., Carrasco-Ochoa, J.A. and Martínez-Trinidad, J.Fco. (2019). A review of unsupervised feature selection methods. *Artificial Intelligence Review*, [online] 53(2), pp.907–948. doi:10.1007/s10462-019-09682-y.

Song, F., Guo, Z. and Mei, D. (2010). Feature Selection Using Principal Component Analysis. *2010 International Conference on System Science, Engineering Design and Manufacturing Informatization*. [online] doi:10.1109/icsem.2010.14.

Spanhol, F.A., Oliveira, L.S., Petitjean, C. and Heutte, L. (2016). Breast cancer histopathological image classification using Convolutional Neural Networks. *2016 International Joint Conference on Neural Networks (IJCNN)*. [online] doi:10.1109/ijcnn.2016.7727519.

Talavera, L. (2005). An Evaluation of Filter and Wrapper Methods for Feature Selection in Categorical Clustering. *Lecture Notes in Computer Science*, [online] pp.440–451. doi:10.1007/11552253_40.

Tfidf.com. (2022). *Tf-idf :: A Single-Page Tutorial - Information Retrieval and Text Mining*. [online] Available at: http://www.tfidf.com/#:~:text=Thus%2C%20the%20term%20frequency%20is,how%20important%20a%20term%20is. [Accessed 17 Aug. 2022].

Thaseen, I.S., Kumar, Ch.A. and Ahmad, A. (2018). Integrated Intrusion Detection Model Using Chi-Square Feature Selection and Ensemble of Classifiers. *Arabian Journal for Science and Engineering*, [online] 44(4), pp.3357–3368. doi:10.1007/s13369-018-3507-5.

Tukey, J.W. (1957). *On the Comparative Anatomy of Transformations on JSTOR*. [online] Jstor.org. Available at: https://www.jstor.org/stable/2237223#metadata_info_tab_contents [Accessed 26 Aug. 2022].

Wang, H., Ma, C. and Zhou, L. (2009). A Brief Review of Machine Learning and Its Application. *2009 International Conference on Information Engineering and Computer Science*. [online] doi:10.1109/iciecs.2009.5362936.

Weinberger, K., Dasgupta, A., Langford, J., Smola, A. and Attenberg, J. (2022). *Feature hashing for large scale multitask learning | Proceedings of the 26th Annual International Conference on Machine Learning*. [online] ACM Other conferences. Available at: https://dl.acm.org/doi/abs/10.1145/1553374.1553516?casa_token=A8vhsXLSJ6QAAAAA:0ri91r4PDVd9AJw_DFgcBpklLBd1DvC8yngY-HdLc85lgE17iVVYJAjK1aUPYtmX_1aXNuPhyR89H28g [Accessed 28 Aug. 2022].

Yang, C. and Sun, B. (2021). Additive requirement ratio estimation using trend distribution features. *Modeling, Optimization, and Control of Zinc Hydrometallurgical Purification Process*, [online] pp.63–82. doi:10.1016/b978-0-12-819592-5.00014-4.

Yufeng G (2017). *The 7 Steps of Machine Learning - Towards Data Science*. [online] Medium. Available at: https://towardsdatascience.com/the-7-steps-of-machine-learning-2877d7e5548e [Accessed 24 Aug. 2022].

Zhang, H., Nettleton, D. and Zhu, Z. (2019). *Regression-Enhanced Random Forests*. [online] ResearchGate. Available at: https://www.researchgate.net/publication/332603876_Regression-Enhanced_Random_Forests [Accessed 26 Aug. 2022].

Zhou, Q., Zhou, H., Zhou, Q., Yang, F. and Luo, L. (2014). Structure damage detection based on random forest recursive feature elimination. *Mechanical Systems and Signal Processing*, [online] 46(1), pp.82–90. doi:10.1016/j.ymssp.2013.12.013.