Final Year Project Report

Full Unit - Final Report

Comparison of Ridge Regression and K Nearest Neighbours Algorithms

Fabio Eugenio dos Santos de Sampaio Doria

A report submitted in part fulfilment of the degree of

BSc (Hons) in Computer Science

Supervisor: Nicolo Colombo

Department of Computer Science
Royal Holloway, University of London

April 12, 2024

Declaration

This report has been prepared on the basis of my own work. Where other published and unpublished source materials have been used, these have been acknowledged.

Word Count:

Student Name: Fabio Eugenio dos Santos de Sampaio Doria

Date of Submission: 12/04/2024

Signature: Fabio Eugenio dos Santos de Sampaio Doria

Table of Contents

Abstract

For most people buying a house will be one of their most important and expensive economic decisions that they will take in their lives.[1] Because of this it would be logical to say that being able to accurately predict the prices of said houses would be of extreme value to people. One possible way to make these predictions would be to create a machine learning model that, given a certain amount of features from each house, would be able to create an accurate prediction of their price.[2] There are a wide range of different machine learning algorithms that could be used to solve this problem. However, this project will focus on two: Ridge Regression and Decision Trees.

In this project I plan on implementing both of these algorithms and comparing their performance on one another to see which one is more effective at resolving the Boston housing problem. To measure their accuracy I will use two metrics specifically used for regression problems, mean absolute error (MAE) and root mean squared error (RMSE). MAE returns the average residual of the predictions that each model makes, this is useful as it can be directly compared against the other model for differences.[3] However, RMSE returns the square root of the average residual of the predictions. This highlights larger errors caused by the model which is practical as it helps to differentiate the performance of both algorithms.[4]

The dataset that will be used to judge the different algorithms is called the Boston housing problem datasets. The dataset is comprised of 506 entries, each having a total of 14 features which describe a property inside of the Boston Massachusetts area.[5] Two variations of this dataset will be used, one with full 14 features and another with a lower number 5. Two models will be created, one with each algorithm and each of these models will be trained and tested on both of these datasets. Their performance will then be analysed, compared, and finally conclusions will be drawn from the results in order to define the effectiveness both algorithms.

Project Specification

During the process of creating this report, my aims are to analyse and compare the performance of the ridge regression and K nearest neighbour algorithms. This is to be done by scoring them based off of accuracy metrics and their usability as models. I also plan to test these algorithms under different conditions such as with datasets of varying numbers of features and different training set sizes. I also plan to explore the relationship between the parameters of these algorithms and their performance.

In order to achieve these goals I will first study the theoretical background knowledge necessary to understand how both of these algorithms function. I will also learn how to identify situations that the ridge regression algorithm should perform well under, such as multicolinear data. After gathering this knowledge I will then create a program with a GUI that is capable of importing and preprocessing the datasets that I will use to examine the algorithms with. This program will also be able to create machine learning models the implement both ridge regression and K nearest neighbour algorithms which can be run on the datasets to collect results on their performances. This program will be able to take user input in order to explore how different scenarios effect the models performances.

With respect to my future career, this project will give me invaluable experience on understanding, implementing, building, ruining and analysing machine learning algorithms in real life situations that I can apply in my future career. It will help me further develop my mathematical skills and understanding as I conduct research on these algorithms which I will be able to apply to other areas of my life and work.

Chapter 1: **Introduction**

Since humans began roaming the earth there have always been three main things that we have needed to survive: food, water, and shelter. In modern developed civilisations food and water has become relatively easy to come by with affordable versions of both being available to most people. However, shelter seems to only become more expensive with time while income stays the same. This can be seen in the UK where the median price of residential houses increased by 14% while income decreased by 1% from 2020 to 2021, making the issue of housing one of the most prevalent ones in recent times.[6] Also, With the advent of the internet and its capability of delivering large scale of information to users, the options of houses available are in-numerous. Because of this it can be an extremely overwhelming task to look for a suitable place to live that both fits the budget but is also reasonably priced considering the aspects and features of the house. House prices also play a significant role in the economy, one of these is being linked to consumer spending. If a homeowner knows that the value of his home increases then he feels confident and is likely to spend more in goods or services or to pay off their debts. If they know the value has decreased then the opposite occurs, homeowners become less confident meaning they are likely to spend less and save more.[7] Taking all of these factors into consideration it can be said that having an accurate method of estimating house prices is of importance from an individual looking for a home all the way to governments trying to predict what economic measures they should take next.

The first step in order to generate accurate estimates is to gather and analyse data on past houses which have been sold and try to find a pattern regarding their value. Specifically between the price they were sold for and features that describe the houses, such as the size of the house, area it is located in, and so forth. This analysis used to be done through traditional statistical methods, however, recently a new approach has emerged called machine learning. Here a machine is responsible for processing and learning from the data it is given in order to make predictions on new data autonomously.[8] When it comes to predicting house prices the machine learning model is given data consisting of past sales of houses where each house sold is also accompanied with a number of features that describe it as mentioned before. The system can then learn how these features influenced the price of the houses and use this information to make predictions on new houses based only on its corresponding features.[9]

Chapter 2: **Theoretical Background**

Within machine learning there are two main methodologies utilised when trying to solve a problem, supervised and unsupervised learning. Unsupervised learning is used when the data being analysed does not need to be labelled, but instead needs to be sorted into groups by their features. On the other hand, supervised learning is concerned with providing labels to unlabelled data in an dataset, such as a list of houses without a price attached to them.[10] Within supervised learning there is once again two different types of problems that occur: classification and regression problems. Classification problems occur when the list of possible classifications is finite such as identifying a handwritten digit as its correct number. However, in the Boston housing problem the list of possible labels is infinite as they could be any price, i.e., any real number.[11] This is called a regression problem and will be the focus of the research paper.

2.1 Simple & Multiple Linear Regression

Regression as a type of machine learning problem comes from the statistical method of regression, which has the goal of determining the relationship between a dependent and independent variable(s).[12] This is then expanded to create regression models which can be formally defined as using an independent variable 'x' to predict a dependent variable 'y'.[13] With the Boston housing problem the independent variable x are the features of each house such as, per capita crime rates per town, while the dependent variable y are the prices of the houses.[14]

2.1.1 Simple Linear Regression

There are many variations of the regression algorithm that can be used in a machine learning model, however simple linear regression is arguably the most simplistic ones and serves as a base for the others. The equation used in a simple linear regression model has the form:

$$y = \beta_0 + \beta_1 x + \epsilon$$

where y and x are the independent and dependent variable respectively. β_0 is the intercept of the line created by the equation with the y-axis, this is known as the constant term as it does not change. β_1 is the regression coefficient which defines the slope of the line. And lastly ϵ is the random error term or the random deviation which represents the difference between the predicted value and the real-life value.[15, 16]

The goal of this machine learning method is to fit the best possible line to the given data, which is achieved by finding the optimal weights of the regression coefficient β_1 and β_0, i.e., the parameters of the model. For this a cost function is used which calculates the difference between the predicted value and the expected values of every sample in the training dataset and returns it as a single real number.[17] This cost is then minimised by modifying the parameters, when the cost function cannot be further minimised then the most optimal coefficients have been achieved.[18] The most commonly used cost function for linear regression is the Sum of Squares Error (SSE) function

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y_i})^2$$

where y_i is the observed true label and $\hat{y}_i$ is the predicted label. The predicted label variable can also be substituted with the linear regression equation to get another version of the cost function.

$$SSE = \sum_{i=1}^{n}(y_i - \beta x_i)^2$$

2.1.2 Multiple Linear Regression

With the basic theoretical framework on regression being set out, it is now necessary to expand it in order to fit a practical use as a machine learning model that can be applied to real-world datasets. In reality, when creating a model their is most of the time going to be more than simply one feature that will define the outcome of the model. For this, multiple linear regression is used which has the same form as the previously seen linear regression but is extended by adding in more terms to account for the multiple features.[19]

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p + \epsilon$$

In this new equation there are now p dependent variables, for each of the features present in the data being used. Each dependent variable also has a their own β coefficient, which represents the change on y given a one unit change the coefficients x.[20] This is the equation describing the process of predicting one value y given p number(s) of dependent variables. However, this can be expanded to solve so that n number of predictions can be made given p dependent variables in a data set.

$$\begin{aligned} y_1 &= \beta_0 + \beta_1 x_{11} + \beta_2 x_{12} + ... + \beta_p x_{1p} + \epsilon \\ y_2 &= \beta_0 + \beta_1 x_{21} + \beta_2 x_2 2 + ... + \beta_p x_2 p + \epsilon \\ &\vdots \\ y_n &= \beta_0 + \beta_1 x_{n1} + \beta_2 x_{n2} + ... + \beta_p x_{np} + \epsilon \end{aligned}$$

During this report the matrix notation will be used primarily when it comes to manipulating and solving equations. Therefore, the system of equations above describing multiple linear regression can also be displayed in this form which is much easier to work with with and to visualise. It is written as

$$\boldsymbol{y} = \boldsymbol{X\beta} + \boldsymbol{\epsilon},$$

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \boldsymbol{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & ... & x_{1p} \\ 1 & x_{21} & x_{22} & ... & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & ... & x_{np} \end{bmatrix}, \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix},$$

where $\boldsymbol{y}$ is a vector of dimensions $n \times 1$ with n being the number of predictions being made. $\boldsymbol{y}$ is a matrix of dimensions $n \times p$ where p is the number of dependent variables. The row of this matrix are made up of the values of the data sample, with each column in the row being a different feature for the given sample. Its first column is filled with ones so that β_0 is always present in the equations after the matrix multiplication. $\boldsymbol{\beta}$ is a vector of dimensions $p \times 1$ that holds all the coefficients for the dependent variables. It is also important to note that features can easily be added or removed from this model, this will be used when comparing the performance of the two algorithms on datasets with different numbers of features.

The cost function that was discussed earlier can also be put into matrix notation which will make the task of minimizing it much easier. It can be written as

$$SSE = (\boldsymbol{y} - \boldsymbol{X\beta})^T(\boldsymbol{y} - \boldsymbol{X\beta})$$

In order to square the residuals before summing them, the vector transpose is multiplied by the vector containing the residuals. This will first multiply each residual with itself squaring them, before they are all added up.

2.2 Ridge Regression

The Ridge Regression method is a different way of estimating the parameters for a multiple linear regression model, specifically in order to deal with issues that can arise from using data that suffers from multicollinearity.[21] Before looking at how Ridge Regression addresses this issue it is first necessary to understand what is multicollinearity and the effects that it can have on a linear regression model.

2.2.1 Issues Caused by Multicollinearity

As said before, Linear Regression models utilise the Sum of Squares Error method to estimate the value of the regression coefficients. However, the reliability of these coefficient estimations could be severely impacted if the data being used to train the model suffers from multicollinearity, which is when two or more independent variables in the model are correlated.[22] This impacts the reliability of the estimations as linear regression models define the regression coefficient to represent a mean change in the dependant variable for each unit change in the independent variable when all other independent variables remain the same, and by definition this cannot occur if two or more independent variables are correlated.[23] This multicollinearity causes the model to become unstable because a small change in one variable will suddenly effect another causing a big change in the prediction. It also makes the model harder to interpret as the regression coefficients do not necessarily reflect the significance of only that specific feature.[24] Finally, it can lead to overfitting which is when a model displays a high level of accuracy on the training dataset, but performs badly when it comes to the testing set. This happens because the model is trained to specifically to the individual data points in the training set instead of the general trend of the points.Then when it tries to predict a data sample not from that set it accuracy is poor.[25] In this section of the report we will only mention the effects of multicollinearity on regression models, the task of determining if multicollinearity is present within a dataset will be the topic of its own section.

2.2.2 Ridge Regression Solution

The Ridge Regression algorithm solves these issues caused by multicollinearity by applying a processes to the model called regularisation, specifically L2 Regularisation. Regularisation is the process of 'regularising' the regression coefficients by trying to make them as small as they need to be.[26] The effect this has on the model is of generalisation, meaning that it wont be as accurate with its predictions in the training set, but it will work better in the long run with predictions on new unseen data. In more precise terms it can be said that a small amount of bias is added into the model which in return decreases the amount of variance which causing the poor performance.[27]

L2 regularisation achieves this by enhancing the SSE equation by adding a penalty term to the end of it. This term is the summation of squared weights of each feature, multiplied by 'a' that defines how 'harsh' the penalty term should be. Finding the optimum value for 'a' is critical as it defines the overall performance of the equation, if it is zero then the model once again uses just SSE and the higher 'a' is the more generalised and less accurate the model will be.[28]

$$SSE + a||\beta||^2 = \sum_{i=1}^{n}(y_i - \beta x_i)^2 \ + a\sum_{j=1}^{P}\beta_j^2.$$

This equation can be represented in matrix notation aswell,

$$SSE_{ridge} = (\boldsymbol{y} - \boldsymbol{X\beta})^T(\boldsymbol{y} - \boldsymbol{X\beta}) + a(\boldsymbol{\beta^T\beta})$$

This equation is then minimised and with its β values being the most optimal coefficients that will be used to help prevent overfitting in the model.

2.2.3 Deriving Ridge Estimator

In order to find the best regression coefficients for our model, we first need to derive the Ridge Regression estimator of β. This equation can then be solved for β which will yield the most optimal coefficients for the model. This estimator will then be implemented into the program in the python programming language, where it will be used to estimate the most optimal regression coefficients for any data set given to the model. In this paper the estimator is derived from the aforementioned L2 regularisation function, specifically in its matrix notation

$$(\boldsymbol{y} - \boldsymbol{X\beta})^T(\boldsymbol{y} - \boldsymbol{X\beta}) + a(\boldsymbol{\beta^T\beta})$$

This equation first needs to be expanded before it can be solved for β

$$\boldsymbol{y^Ty} - \boldsymbol{y^TX\beta} - \boldsymbol{\beta^TX^Ty} + \boldsymbol{\beta^TX^TX\beta} + a(\boldsymbol{\beta^T\beta})$$

Here the terms $\boldsymbol{y^TX\beta}$ and $\boldsymbol{\beta^TX^Ty}$ are both equal to each other, following the rules of transpose, and also a scalar as this is the result of the dot product between its elements. Therefore,

$$\boldsymbol{y^Ty} - 2\boldsymbol{\beta^TX^Ty} + \boldsymbol{\beta^TX^TX\beta} + a(\boldsymbol{\beta^T\beta})$$

To find the point where this function has its lowest value we need to take its derivative with respect to β, as this will show us how the value output by the cost function changes with respect to β

$$\frac{\mathrm{d}(SSE_{ridge})}{\mathrm{d}\beta} = -2X^Ty + 2X^TX\beta + 2a\beta$$

This differential equation can now be set to zero and solved for β, giving us our ridge estimator for the model.

$$\begin{aligned} 0 &= -2X^Ty + 2X^TX\beta + 2a\beta \\ 0 &= -X^Ty + X^TX\beta + a\beta \\ X^Ty &= X^TX\beta + a\beta \\ X^Ty &= \beta(X^TX + a\boldsymbol{I}) \\ \beta &= (X^TX + a\boldsymbol{I})^{-1}X^Ty \end{aligned}$$

This equation can now be solved for β using the information available from a training set, giving us the optimal regression coefficients for our ridge regression model. Where the rows of matrix $\boldsymbol{X}$ are the data samples of the training set and its columns being the values of the features corresponding to each sample. And vector y containing the labels of the corresponding data samples.

2.3 K Nearest Neighbours

The K nearest neighbours algorithm is very different compared to the aforementioned ridge regression algorithm. However, for this report I shall be using it as a benchmark to compare with the performance of the ridge regression algorithm. K nearest neighbours is a relatively simple and intuitive machine learning algorithm compared to must others. It is a supervised learning method that can be used to solve both classification and regression problems, the latter version will be used in this report. We will take a look at the differences between the two algorithms, but first we must understand specifically how k nearest neighbours works and how it computes its predictions.

2.3.1 Differences from Ridge Regression

One of the key differences between ridge regression and the K nearest neighbours algorithm is the assumptions that they both make about the data they are predicting on. K nearest neighbours is a non-parametric algorithm, meaning that it does not rely on any assumptions about the underlying distribution of the data it is predicting on. This is the opposite of ridge regression that assumes the relation between the feature and target variables is specifically linear. This property of K nearest neighbours allows it to be a very versatile algorithm as it can be applied to many more datasets than the ridge regression could be and still perform well. On the other hand if ridge regression is applied to a dataset that does not follow a linear functional form then it will perform poorly. A problem with K nearest neighbours however is that for every prediction it makes it needs to go through the entire training set. For smaller datasets this is not to big a problem but for larger datasets it can become extremely computational expensive, especially in comparison to the ridge regression algorithm. For a ridge regression model it only needs to be fitted once to the training set and can then use its coefficients to create any other predictions. For this reason, ridge regression is considered computationally efficient and scales well with large datasets contrary to k nearest neighbours.

As mentioned before K nearest neighbours can be used to solve both regression and classification tasks, once again adding to its versatility as an algorithm compared to other regression algorithms. This ability to solve both types of problems means that an K nearest neighbours model made to solve classification problems can be easily transformed to solve regression problems if needed. A final difference between the two algorithms is how interpretable they are when it comes understanding how well each feature describes the target variable. Regression algorithms, such as ridge, tend to be high in interpretability as one can look at the magnitude of the coefficients of each feature variable to tell how impactful it is in predicting the target variable. On the other hand K nearest neighbours predicts based off of computing euclidean distances, this means that their is no simple way of determining what feature was more significant is causing a closer distance between the two samples.

2.3.2 How K Nearest Neighbours Predicts

The main difference between the two however lies in how k nearest neighbours predicts its target values, for regression this is done by first computing the distance between all the training data points and the new unseen data sample. The model then proceeds to choose the k nearest data points to the unseen sample, their target variables are then averaged and this is given as the predicted target value of the given sample. There are a couple of different ways to calculate this distance, however, for this report I shall be using the euclidean distance as it is the most common and straight forward method as the main aim of this report is to analyse the performance of the ridge regression algorithm. The euclidean distance between two data points, $\underline{u}$ and $\underline{v}$, in a euclidean space of n dimension can be described as

$$d(\underline{u}, \underline{v}) = \sqrt{\sum_{i}^{n} = 1(v_i - u_i)^2}$$

As said before, this distance must be computed for every data point inside of the training set. These distances are then all compared with one another to find the k smallest distances. In big O notation this computation of euclidean distance has a time complexity of O(n), where p is the number of dimensions of the dataset. Therefore, the time complexity of predicting one sample is O(np), where p is the number of samples within the training set.

2.3.3 Choosing the Parameter K

Similar to the alpha parameter in ridge regression, the K nearest neighbour algorithm also has a parameter K which is the number of neighbours taken into account when predicting the target value of the new sample. Just like alpha, K is a hyper parameter meaning that it is chosen by us to control how the model learns from its training data. How K is chosen is critical to building a successful model, the value chosen directly affects how general the model fits its training data. If K is equal to one than it will score perfectly when predicting on the training set, however, it will perform poorly on the test set as the model is overfitting. This happens as the chosen predicted value is coming from only one point in the training set, which in most cases will not be exactly the same as the unseen sample. Therefore, as

K increase the more general the model will become, if too high then the model will suffer from underfitting as it does not capture enough details within the training set. Usually the parameter K is set to 3, however, just like for the ridge regression model we will also be using cross validation to find the most optimal value for K.

2.4 Diagnosing Multicollinearity

As outlined in the previous section, multicollinearity can seriously impact the performance of regression models if present within their training set. Because of this it is important to know how to identify multicollinearity within a dataset, so that the correct measures can be taken to deal with it. In the case of this report, we need to make sure our datasets suffer from multicollinearity so that we can analyse the effectiveness of the ridge regression algorithm in combating it as this is its main purpose. Before looking for multicollinearity we first need to understand how to describe and find patterns within data mathematically, this can be done through the concept of variance, covariance, eigenvalues, and eigenvectors.

2.4.1 Variance, Covariance & Correlation

As said previously, before looking at collinearity we must first understand the concept of variance and what it tells us about a dataset. Variance is the measure of how much each data point differs from the mean of the whole dataset. This can be computed by taking the difference between each point and the mean of the dataset, squaring these results and then averaging them. Therefore,

$$VAR(X) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2,$$

and

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i.$$

From these equations we can see that if the data is more spread a out then the datasets variance score will be higher the average distance from the points to the mean will be larger, and vice versa. The measure of variance is also 1-dimensional, as it only quantifies the spread of data in regards to a single feature, e.g., the grades of all students in a class.

The concept of variance is useful as it tells us how a 1-dimensional dataset is spread out over an axes. However, to find collinearity we need to, by definition, at least be able to compare one or more different features with each other to see if they are correlated. This brings forward the concept of covariance, which can be simply defined as a measure of how the mean of two variables vary with respect to each other. This is given by

$$COV(X,Y) = \frac{1}{N}\sum_{i=1}^{M}(x_i - \mu_x)(y_i - \mu_y).$$

Many observations can be made from this equation, firstly if you calculate the covariance between a dimension and itself you simply get its variance. It is also important to note the sign of the resulting computation, if positive then as the values of one dimension increase so do the values of the other, i.e., they are positively correlated. The same goes for the opposite, if the result is negative then as one dimension increases the other decreases, i.e., they are negatively correlated. Finally, if the covariance is zero then a change in one dimension does not affect the other in any whatsoever, i.e., there is no linear correlation between the two. This links directly with our definition of collinearity as it occurs when one feature is correlated with another.

However, there is a problem of using only covariance to look for collinearity, it only indicates the direction of the relationship between the two variables and not the magnitude of the relationship. As we are dealing with values recorded in the real world it is expected that some of the different features will be correlated, however, the issue is how correlated they are and if it is enough to cause issue in a regression model. Therefore, in order to determine collinearity we would also need a measure of how correlated two variables are to decide if they in fact are collinear.

This brings us to the concept of correlation which is an extension of the covariance measure we have just seen. Correlation, just like covariance, captures the direction of the linear relationship but it also describes the strength of said relationship as a value between -1 and +1. Correlation is also a function of covariance and can be computed as follows:

$$CORR(X, Y) = \frac{COV(X, Y)}{\sqrt{VAR(X)VAR(Y}}.$$

There are many different ways to compute correlation, however, for the purposes of this report we will be using the Pearson's correlation coefficient which is achieved by computing the covariance of two variables over the squared product of each variables variances as seen above. This measure of correlation will now allow us to see if any two variables from a dataset are correlated and to what extent, this is a very useful tool that can be used to analyse a dataset and determine if it suffers from collinearity.

2.4.2 Covariance Matrix

Considering the observations from the correlation measure, a way to determine if a dataset sufferers from multicollinearity is to compute the correlation scores of all its features with respect to each other and analyse their values. This can be done by first building the covariance matrix $\boldsymbol{C}$ of dimensions $(d \times d)$ and then transforming it into a correlation matrix. The covariance will also be important later so it is important to know how to compute it which is defined as

$$\boldsymbol{C} = \begin{bmatrix} COV(Dim_1, Dim_1) & ... & COV(Dim_1, Dim_d) \\ \vdots & \ddots & \vdots \\ COV(Dim_d, Dim_1) & ... & COV(Dim_d, Dim_d) \end{bmatrix}$$

where Dim_i and Dim_j are the different dimensions being compared for that given position of the covariance matrix. An important thing to note is that this matrix is of dimensions $(d \times d)$ where d is the number of features or dimensions that the dataset has. Another observation is that the covariance matrix is symmetric by definition as $COV(Dim_i, Dim_j) = COV(Dim_j, Dim_i)$. Where all values along the diagonal will be equal to the variance of each dimension, as the covariance of a dimension with itself is simply its variance, $COV(Dim_i, Dim_i) = VAR(Dim_i)$.

This whole process of creating a covariance matrix can be generalized through the use of matrix notation to simplify the computations needed to reach the final resulting matrix. For this part I shall refer to the initial data matrix as $\boldsymbol{D}$ which has dimensions $M \times F$. The rows of this matrix are the individual data samples from the dataset while each column represents a feature that describes each sample. This can be visualised as a matrix that is composed of F vectors stacked horizontally, where F is the number of features that describes the dataset.

$$\boldsymbol{D} = \begin{bmatrix} \vdots & \vdots & ... & \vdots \\ \underline{t}_1 & \underline{t}_2 & ... & \underline{t}_F \\ \vdots & \vdots & ... & \vdots \end{bmatrix}.$$

The first step to building the covariance matrix is to center the data so that the mean of all feature vectors $\underline{t}$ is equal to zero. This is done by taking the mean of each feature vector $\underline{t}$ and then subtracting it from every corresponding feature vectors elements. To do this we define a new matrix $\boldsymbol{X}$ of the same dimension $M \times F$ which is the centered version of the original data matrix $\boldsymbol{D}$, it is described as

$$\boldsymbol{X} = \begin{bmatrix} \vdots & \vdots & ... & \vdots \\ \underline{x}_1 & \underline{x}_2 & ... & \underline{x}_F \\ \vdots & \vdots & ... & \vdots \end{bmatrix}, \; x_{ij} = t_{ij} - \mu_i,$$

where

$$\mu_i = \frac{1}{M}\sum_{j=1}^{M} t_{ij}.$$

Now that we have the centered version of our data matrix we can see some similarities with our equation for covariance. Every element in $\boldsymbol{X}$ is equal to the first bracket of the summation operation to compute variance, which is the original value minus the mean. In order to make it equal to the brackets of the covariance equation we just need to multiply the transpose of the centered matrix by itself so that we multiply the correct terms with themselves to create the whole of the contents of the summation symbol for covariance. The only thing left to achieve the covariance would be to divide the previous matrix multiplication by the reciprocal of the total sample size minus 1. Therefore, the covariance matrix of $\boldsymbol{X}$ is described as

$$\boldsymbol{C_X} = \frac{1}{M-1} X X^T.$$

2.4.3 Correlation Matrix

To get from the covariance matrix to the correlation matrix there is only one step needed. Just like to transform the covariance equation into the correlation equation we need to divide each entry of $\boldsymbol{C_X}$ by the square root of the product of the variance scores for that specific entry. The correlation matrix $\boldsymbol{\rho}$ can be describe as

$$\boldsymbol{\rho} = \begin{bmatrix} 1 & ... & CORR(Dim_1, Dim_d) \\ \vdots & \ddots & \vdots \\ CORR(Dim_d, Dim_1) & ... & 1 \end{bmatrix}, \ \rho_{ij} = \frac{COV(Dim_i, Dim_j)}{\sqrt{VAR(Dim_i)VAR(Dim_j)}}.$$

Some important properties of the correlation matrix that we can observe is that it is symmetric just like the covariance matrix. However, one difference between the two is that the diagonal of the correlation matrix is all 1s. This happens because the diagonal of the matrix is computed by taking the correlation of the same variable and intuitively a variable is always fully correlated with itself. Also as mentioned with the correlation score, this matrix will only contain values between -1 and 1.

From this matrix alone we are already able to see if there is any collinearity present between the feature variables of its corresponding dataset. Simply analyse the value of each element of the correlation matrix, not including the values that sit on the matrices diagonal. If any of these values are close to plus or minus one then the two corresponding feature variables of that correlation score are reasonably correlated to be considered collinear.

One important questions that needs to be answered for this approach to work though is at what value does the correlation indicate collinearity? The answer is that their is no objective and exactly defined boundary. A common interpretation of the correlation values is that anything above +0.75 or below -0.75 is considered a high degree of correlation, and therefore a good indicator that collinearity is present within the dataset. If a value of above +0.90 or below -0.90 occurs then those variables will be considered a very high degree of correlation, and therefore an extremely strong indicator of collinearity within the dataset.

A side note to be made of the correlation matrix is that in the same way that it can be used to determine collinearity between feature variables, it can also be used describe how well each feature variable predicts the target variable. By definition, if a feature variable is correlated to the target variable then it describes it to a certain extent which can be seen by computing its correlation score with the target variable. In the correlation matrix this can easily be visualised by looking at the row or column of the target variable and observing all the correlation scores on that axis which correspond to the correlation scores all the feature variables against the target. In this report we will use any value above 0.50 to say that a feature variable plays a reasonable part in describing the target variable.

2.4.4 Problems with Correlation Matrix

Even though computing the correlation matrix helps us determine if collinearity is present within the dataset, in some cases this technique is not adequate enough to determine if multicollinearity is present within a dataset. The issue is that multicollinearity can still occur between 3 or more feature variables. This relationship would not show up in the covariance matrix as it only computes the relationship between two feature variables at a time. Another shortcoming is that correlation only measures the linear relations between 2 variables, any non-linear relationships would not be detected.

A variable could be related to more than two other variables but not be directly correlated to them when considered individually. This can become more prevalent in datasets with bigger numbers of features as they are more likely to interact with each other in less obvious ways due to the amount of dimensions that the data lives in.

A way to mitigate some of these problems is with the covariance matrix, specifically, by decomposing it so that we determine its principal components and further analyse the relationship between the covariance values and how they change between themselves. This decomposition into components can be made through the use of eigenvectors and eigenvalues which will allow us to find the variance between these covariance scores to better determine if multicollinearity is present.

2.4.5 Eigenvalues and Eigenvectors

As mentioned before, eigenvalues and eigenvectors can help in describing properties of our datasets. However, first we must understand what they are exactly and what they can tell us about a matrix. Firstly, we can define an eigenvalue λ as a scalar quantity of a non-zero eigenvector $\underline{u}$, such that

$$\boldsymbol{A}\underline{u} = \lambda\underline{u}$$

where $\boldsymbol{A}$ is a matrix of dimensions $n \times n$. This means that the matrix $\boldsymbol{A}$ stretches the eigenvector $\underline{u}$ by the scalar amount of the eigenvalue λ.[29]

This can be visualised geometrically if the matrix $\boldsymbol{A}$ is plotted in n dimensions. An eigenvector $\underline{u}$ of a matrix $\boldsymbol{A}$ can be interpreted as a direction within the plot in which the data of the matrix is spread along this direction by a degree specified by its eigenvalue λ. A matrix of dimensions $n \times n$ will have $\lambda_1, ..., \lambda_n$ eigenvalues and eigenvectors, one pair describing each dimension.

An important property of eigenvectors is that they are all linearly independent of one another. This is allows them to describe the dimensional space in which their matrix exists in, as a set of n linearly independent vectors defines a n-dimensional space. This means that the original matrix $\boldsymbol{A}$ can be described by it eigenvectors, signifying an axes were the values are spread across, and its eigenvalues, which describe the spread of the data across said axes. Therefore, we call each eigenvector of matrix $\boldsymbol{A}$ a component of $\boldsymbol{A}$, i.e., an axes that describes the matrix $\boldsymbol{A}$. And the corresponding eigenvalue the variance of the matrix along said axes.

2.4.6 Eigenvalues of Covariance Matrix

This understanding of eigenvectors and eigenvalues allows us to draw important similarities with the definition of variance, which is the measure of spread between data points from a set. The similarity is that eigenvalues can be seen as a measure of variance for our components in regards to one another, where a higher eigenvalue demonstrates a larger variance between one component and another and lower eigenvalues as a lower variance. This is significant as it directly tells us how correlated our features are with themselves which is the cause of multicollinearity.

In an ideal dataset all of the eigenvalues from its covariance matrix would be larger than zero and of similar size. This demonstrates that its components all describe the dataset in equally different ways from one another. On the other hand, if some of its components have very high eigenvalues and other have very low eigenvalues, then only some features describe the data really well and uniquely, while others describe the data poorly and redundantly. This means that one or more feature(s) describe the data in the same way that another feature does, by definition this means that the values of the features are dependent to some degree on each other.

A way to quantify this difference is by the use of condition indices, this is computed for each component by calculating the square root of the largest eigenvalue divided by the smallest eigenvalue. Depending on the size of this condition index we can tell if the dataset is likely suffering from multicollinearity. In this report any values between 10 and 30 are considered significant, values over 30 are considered to show that the dataset suffers from severe multicollinearity.

Chapter 3: **Method**

3.1 Data Selection

In order to gather data on the performance of the different machine learning algorithms we first need to choose the datasets that they will be trained on. To ensure that their performance is fully analysed, each algorithm will be trained on 3 different datasets so that they can be analysed under a wide range of conditions. The idea is to check if there is a difference in performance from the algorithms when they are trained on datasets with different quantities of features.

It is expected that the Ridge Regression model will perform equally or better as the number of features increases while the K-nearest neighbors algorithm will suffer. This is due to the possibility of the datasets with more features suffering from highers levels of co-linearity. Ridge Regression should not be greatly affected by this as it implements the penalty term to reduces the effects of co-linearity, however, K-nearest neighbors does not have such a mechanism.

To try and observe this difference each dataset will have approximately 10, 50, and 100 features in order to capture any difference in performance. The 3 datasets chosen for this report are the Boston Housing Dataset with 13 features[30], the Student Performance dataset with 30 features[31], and the Communities and Crime Database with 127 features[32].

3.1.1 Boston Housing Dataset

The Boston Housing Dataset contains information describing houses in the area of Boston, Massachusetts, where each sample is a house and each column is a feature for said house. The dataset is relatively small with only 506 samples. Each of these samples are described by 14 features with the last one "MEDV" being the median value of the home in \$1000 which is our target variable:

1. CRIM - per capita crime rate by town
2. ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
3. INDUS - proportion of non-retail business acres per town.
4. CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
5. NOX - nitric oxides concentration (parts per 10 million)
6. RM - average number of rooms per dwelling
7. AGE - proportion of owner-occupied units built prior to 1940
8. DIS - weighted distances to five Boston employment centres
9. RAD - index of accessibility to radial highways
10. TAX - full-value property-tax rate per \$10,000
11. PTRATIO - pupil-teacher ratio by town

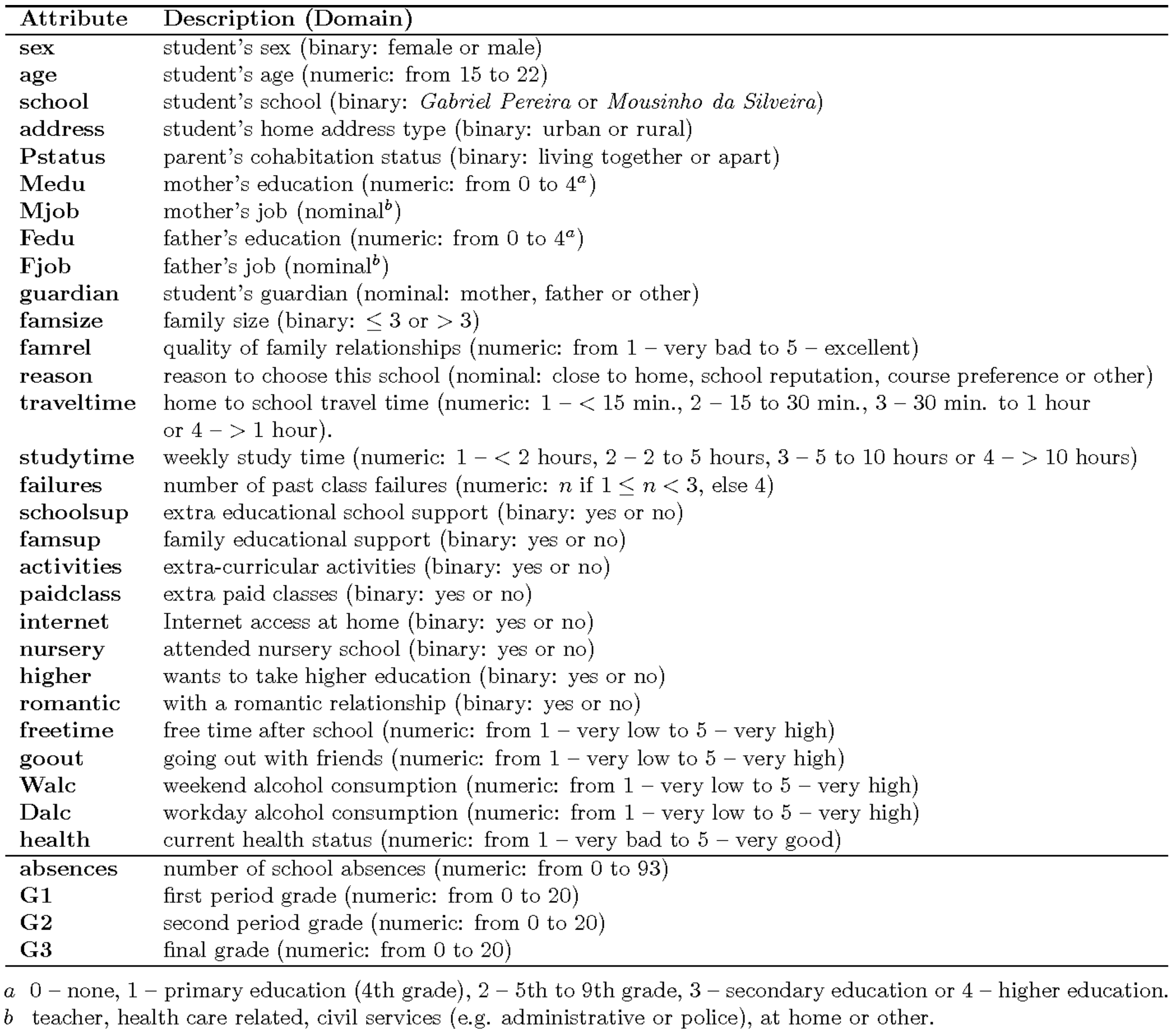
12. B - 1000(Bk - 0.63)$\hat{2}$ where Bk is the proportion of blacks by town
13. LSTAT - % lower status of the population
14. MEDV - Median value of owner-occupied homes in $1000's

An important observation of the values for the target variable is that they seem to be censored at 50, i.e., $50,000. This seems to be the case as the exact value of 50 for MEDV is present in 16 separate samples, while in 15 other samples the value is between 40 and 50. However, the original authors did not specify if they did any sort of censoring when collecting the original data.

3.1.2 Student Performance Dataset

This dataset consists of the information collected from secondary students of two separate public schools in Portugal between the year 2005 and 2006. The information from this dataset came from two main sources: past school report cards, and responses from a questionnaire with closed questions. Each student is graded three separate times in a year with their 3rd grade constituting the final grade. The recorded grades came from the mathematics and Portuguese language classes as these are mandatory and therefore taken by all students. On the other hand, the questionnaire asked questions built to capture demographic, social/emotional, and school related information from each student.[33]

The dataset consists of 649 samples, i.e., students, each of which are described by 33 features with the last 3 being the report grades. The individual descriptions of the features are



Attribute	Description (Domain)
sex	student's sex (binary: female or male)
age	student's age (numeric: from 15 to 22)
school	student's school (binary: *Gabriel Pereira* or *Mousinho da Silveira*)
address	student's home address type (binary: urban or rural)
Pstatus	parent's cohabitation status (binary: living together or apart)
Medu	mother's education (numeric: from 0 to 4^a)
Mjob	mother's job (nominal^b)
Fedu	father's education (numeric: from 0 to 4^a)
Fjob	father's job (nominal^b)
guardian	student's guardian (nominal: mother, father or other)
famsize	family size (binary: ≤ 3 or > 3)
famrel	quality of family relationships (numeric: from 1 – very bad to 5 – excellent)
reason	reason to choose this school (nominal: close to home, school reputation, course preference or other)
traveltime	home to school travel time (numeric: 1 – < 15 min., 2 – 15 to 30 min., 3 – 30 min. to 1 hour or 4 – > 1 hour).
studytime	weekly study time (numeric: 1 – < 2 hours, 2 – 2 to 5 hours, 3 – 5 to 10 hours or 4 – > 10 hours)
failures	number of past class failures (numeric: n if $1 \leq n < 3$, else 4)
schoolsup	extra educational school support (binary: yes or no)
famsup	family educational support (binary: yes or no)
activities	extra-curricular activities (binary: yes or no)
paidclass	extra paid classes (binary: yes or no)
internet	Internet access at home (binary: yes or no)
nursery	attended nursery school (binary: yes or no)
higher	wants to take higher education (binary: yes or no)
romantic	with a romantic relationship (binary: yes or no)
freetime	free time after school (numeric: from 1 – very low to 5 – very high)
goout	going out with friends (numeric: from 1 – very low to 5 – very high)
Walc	weekend alcohol consumption (numeric: from 1 – very low to 5 – very high)
Dalc	workday alcohol consumption (numeric: from 1 – very low to 5 – very high)
health	current health status (numeric: from 1 – very bad to 5 – very good)
absences	number of school absences (numeric: from 0 to 93)
G1	first period grade (numeric: from 0 to 20)
G2	second period grade (numeric: from 0 to 20)
G3	final grade (numeric: from 0 to 20)

a 0 – none, 1 – primary education (4th grade), 2 – 5th to 9th grade, 3 – secondary education or 4 – higher education.
b teacher, health care related, civil services (e.g. administrative or police), at home or other.

Figure 3.1: The preprocessed student related variables[33]

An observation from the descriptions of the features, I expect to encounter multicollinearity in this dataset especially between the first two grades and the final grade. This intuitively makes sense as a student who has already done well in a test is more likely to do well again in another test than not and vice versa. However, this will be analysed later in depth.

3.1.3 Communities and Crime Dataset

This third and final dataset consist of real socio-economic data collected in 1990 US Census, law enforcement data from the 1990 US LEMAS survey, and crimedata from the 1995 FBI UCR.[32] This information describes certain aspects of different communities within the US, such as family income, percentages of ethnicities, and housing costs. It also has information describing the strength of law enforcement in these communities such as number of officers and police cars. Finally, the dataset contains the crime statistics for each of the communities as well, with values describing murder, robbery, arson and per capita violent crimes. This equates to a total of 147 features:

- 125 features: Predictive, data on communities and law enforcement
- 4 features: Non-predictive, data that identifies the communities

- 18 features: Possible target variables, crime indecies for each community

The authors also outline a possible limitation of this dataset which is that a lot of the police data from the LEMAS survey is not present for many of the communities. This is due to the survey only being taken by departments with 100 plus officers which was not the case for many of these communities. Also the per capita crime features were computed using FBI data from 1995, where the other features were built from 1990. This can lead to issues when predicting crimes per capital as their might be a natural difference between the feature and target variables that cannot necessarily be captured by the model.

3.2 Data Analysis

After conducting a surface analyses of the datasets and their descriptions it is now time to apply some data visualisation techniques in order to get a more sophisticated view of our datasets and if there seems to be any patterns inside of them.

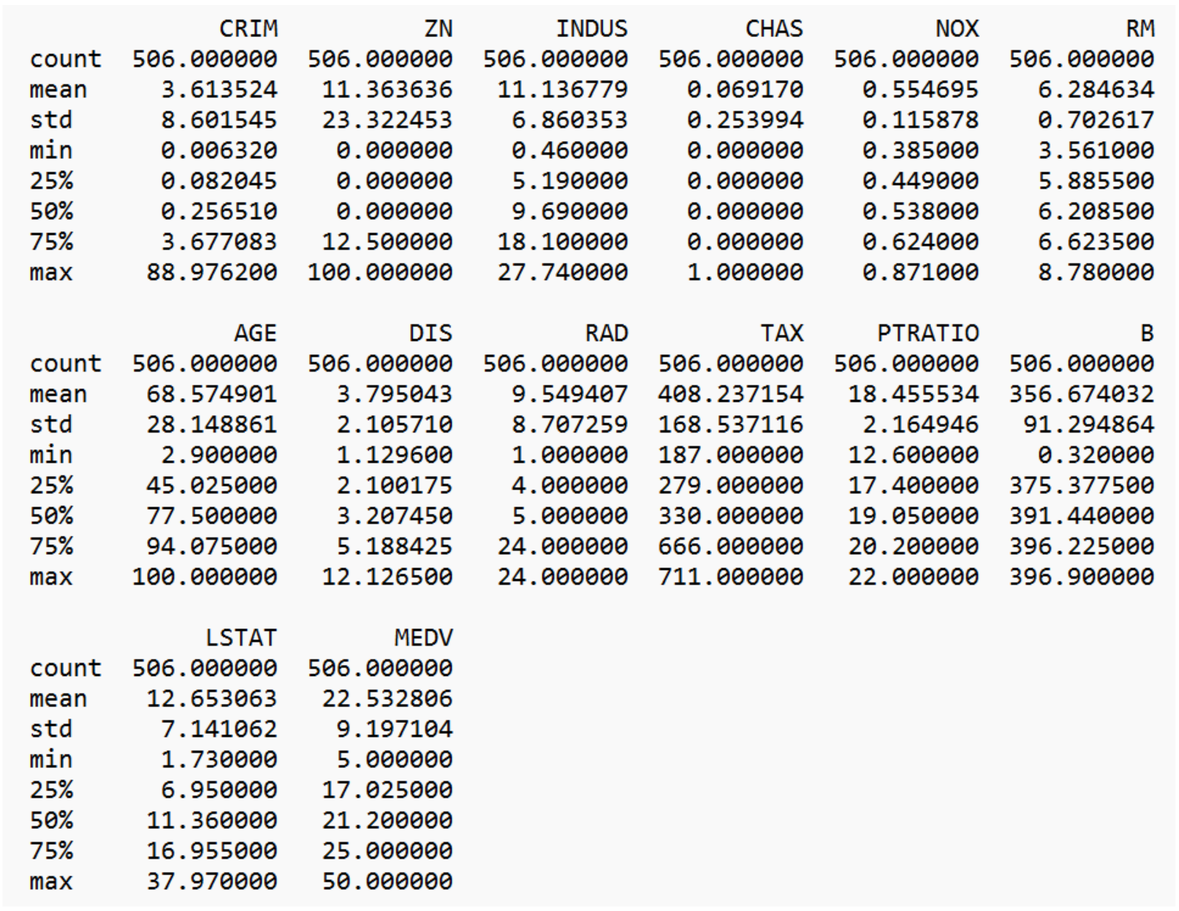
One of these techniques is the describe() function from the pandas python library, which returns the count (number of non-null observations), mean, standard deviation, minimum, and maximum values of each feature in the dataset. It also returns how many values in each column are below the the 75, 50, and 25 percentile. These values can give us some insight on any patterns present within each feature and if any modifications need to be made to remove irregularities from the datasets.

We will also be employing the techniques discussed in the theoretical section such as building each datasets covariance matrix and computing their eigenvalues and eigenvectors to determine their condition numbers to see if they suffer from multicollinearity. I will also be creating their correlation matrices to see if any feature variables seem to be directly correlated with one another. This correlation matrix will be displayed as a heat map so that it is easier and more intuitive to see what variables are correlated with each other and the magnitude of this correlation. The heat map works by displaying higher correlation scores in the matrix as brighter colours and lower correlation scores as darker colours, this makes it easier to understand the matrix and what entries are more significant just by visual inspection.

3.2.1 For Boston Housing

Description of features

The first thing to be done is run the pandas describe function on the Boston housing datasets pandas Dataframe object to get the aforementioned values for its features. The result of the function is as follows:



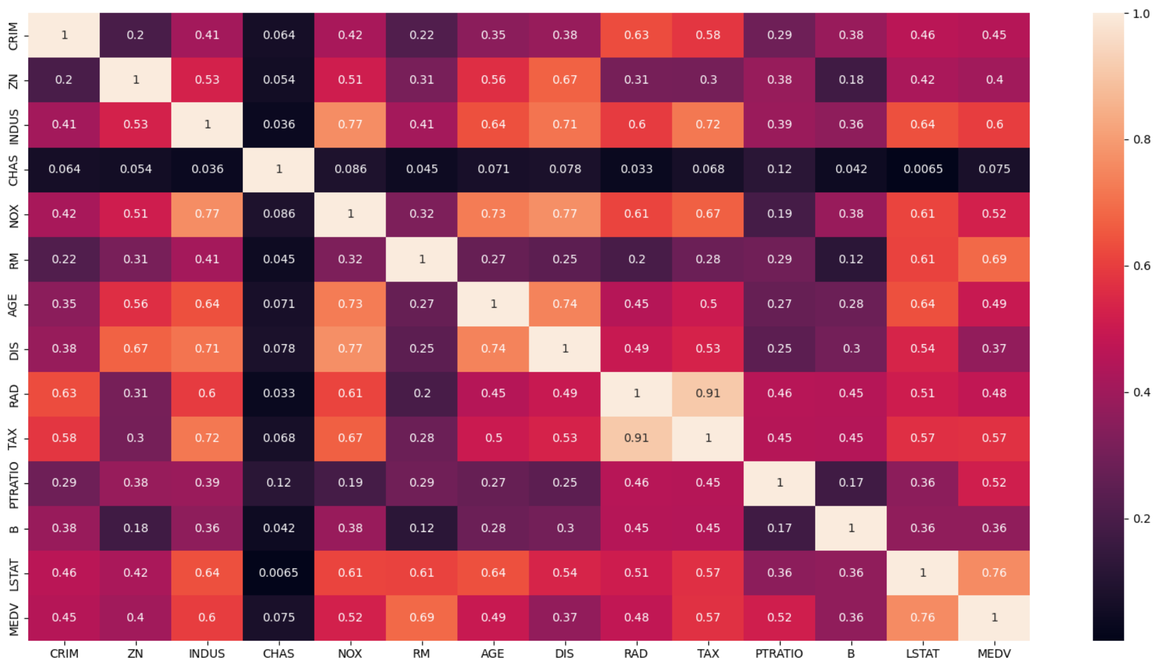
```
             CRIM          ZN       INDUS        CHAS         NOX          RM
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean     3.613524   11.363636   11.136779    0.069170    0.554695    6.284634
std      8.601545   23.322453    6.860353    0.253994    0.115878    0.702617
min      0.006320    0.000000    0.460000    0.000000    0.385000    3.561000
25%      0.082045    0.000000    5.190000    0.000000    0.449000    5.885500
50%      0.256510    0.000000    9.690000    0.000000    0.538000    6.208500
75%      3.677083   12.500000   18.100000    0.000000    0.624000    6.623500
max     88.976200  100.000000   27.740000    1.000000    0.871000    8.780000

              AGE         DIS         RAD         TAX     PTRATIO           B
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean    68.574901    3.795043    9.549407  408.237154   18.455534  356.674032
std     28.148861    2.105710    8.707259  168.537116    2.164946   91.294864
min      2.900000    1.129600    1.000000  187.000000   12.600000    0.320000
25%     45.025000    2.100175    4.000000  279.000000   17.400000  375.377500
50%     77.500000    3.207450    5.000000  330.000000   19.050000  391.440000
75%     94.075000    5.188425   24.000000  666.000000   20.200000  396.225000
max    100.000000   12.126500   24.000000  711.000000   22.000000  396.900000

            LSTAT        MEDV
count  506.000000  506.000000
mean    12.653063   22.532806
std      7.141062    9.197104
min      1.730000    5.000000
25%      6.950000   17.025000
50%     11.360000   21.200000
75%     16.955000   25.000000
max     37.970000   50.000000
```

From these values, two different features standout from the others due to their 25 and 50 percentiles values being equal to zero. These are the ZN and CHAS which are the proportion of residential land zoned for lots over 25,00 sq.ft and the dummy Charles River value that denotes if a property bound the river or not. The reason that these percentile values are zero can probably be accredited to both of these features being either conditional or boolean in nature. These values could be an indication that their corresponding variables will not be very useful in predicting the target variable.

In this description it is also possible to note the max value of the MEDV variable being equal to exactly 50.00. This does follow the theory that the original authors censored the max value of the prices of the properties. If censoring did in fact occur then it might be beneficial to the model to remove any samples that have their target values equal to 50 as these do not follow the same relationship between feature and target variables as the other samples do. Due to this possiblity, before any further analysis is made I shall remove any sample from the dataset that has 50.00 as its MEDV value.

Correlation Matrix

Now lets compute the correlation matrix of the Boston housing dataset to see if any features might be correlated with one another in a way that might indicate the presence of multicollinearity.



	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
CRIM	1	0.2	0.41	0.064	0.42	0.22	0.35	0.38	0.63	0.58	0.29	0.38	0.46	0.45
ZN	0.2	1	0.53	0.054	0.51	0.31	0.56	0.67	0.31	0.3	0.38	0.18	0.42	0.4
INDUS	0.41	0.53	1	0.036	0.77	0.41	0.64	0.71	0.6	0.72	0.39	0.36	0.64	0.6
CHAS	0.064	0.054	0.036	1	0.086	0.045	0.071	0.078	0.033	0.068	0.12	0.042	0.0065	0.075
NOX	0.42	0.51	0.77	0.086	1	0.32	0.73	0.77	0.61	0.67	0.19	0.38	0.61	0.52
RM	0.22	0.31	0.41	0.045	0.32	1	0.27	0.25	0.2	0.28	0.29	0.12	0.61	0.69
AGE	0.35	0.56	0.64	0.071	0.73	0.27	1	0.74	0.45	0.5	0.27	0.28	0.64	0.49
DIS	0.38	0.67	0.71	0.078	0.77	0.25	0.74	1	0.49	0.53	0.25	0.3	0.54	0.37
RAD	0.63	0.31	0.6	0.033	0.61	0.2	0.45	0.49	1	0.91	0.46	0.45	0.51	0.48
TAX	0.58	0.3	0.72	0.068	0.67	0.28	0.5	0.53	0.91	1	0.45	0.45	0.57	0.57
PTRATIO	0.29	0.38	0.39	0.12	0.19	0.29	0.27	0.25	0.46	0.45	1	0.17	0.36	0.52
B	0.38	0.18	0.36	0.042	0.38	0.12	0.28	0.3	0.45	0.45	0.17	1	0.36	0.36
LSTAT	0.46	0.42	0.64	0.0065	0.61	0.61	0.64	0.54	0.51	0.57	0.36	0.36	1	0.76
MEDV	0.45	0.4	0.6	0.075	0.52	0.69	0.49	0.37	0.48	0.57	0.52	0.36	0.76	1

From looking at the heat map of the correlation matrix one pair of variables stand out in particular, the RAD and TAX features. From the correlation matrix we can see that their correlation scores are 0.91 which is very close to 1 in terms of correlation. This is a very strong indicator that theses two variables are most likely collinear to an extent that it could cause severe issues in a regular regression model that does not have a regularisation term.

From the heat map we can also identify some other variables that seem to be significantly correlated with others. These may not be to the same extent as the RAD and TAX features are but their correlation scores are still high enough to be considered possibly problematic. Theses are variable pairs with correlation scores above 0.70, such as: INDUS with NOX, DIS, and TAX. NOX with AGE, and DIS and finally AGE with DIS. Even though the correlation of these variables are not very high they are still high enough to indicate a presence of significant multicollinearity, especially due to the amount of variable pairs with theses high values, i.e., $\rho \geq 0.70$.

The correlation matrix can also give us a good idea of what variables are most likely to play a bigger part in helping us predict the target variable. This can be seen by analysing the correlation score of the feature variables against the target variable, values over 0.50 indicate that those variables will play an important part in the prediction process and should have high regression coefficient values. These variables are INDUS, NOX, RM, TAX, PTRATIO, and LSTAT, with LSTAT being very important due to its correlation score of 0.76 with the target variable.

Eigenvalues & CIs of Covariance Matrix

To further analyse this dataset and determine if it suffers from multicollinearity lets compute the covariance matrix and take its eigenvalues and eigenvectors to see the variance of this matrix. Initially, we can build a scree plot of the computed eigenvalues to have a general idea of the scale and distributions of the eigenvalues. This will help us understand how many features actually contribute to predicting the target variable and if these features are few or if they all equally contribute.

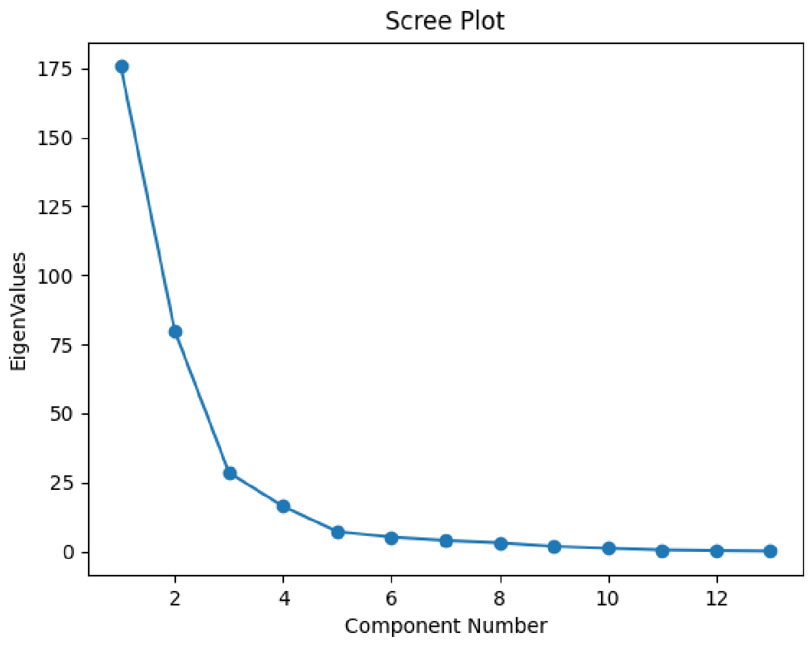
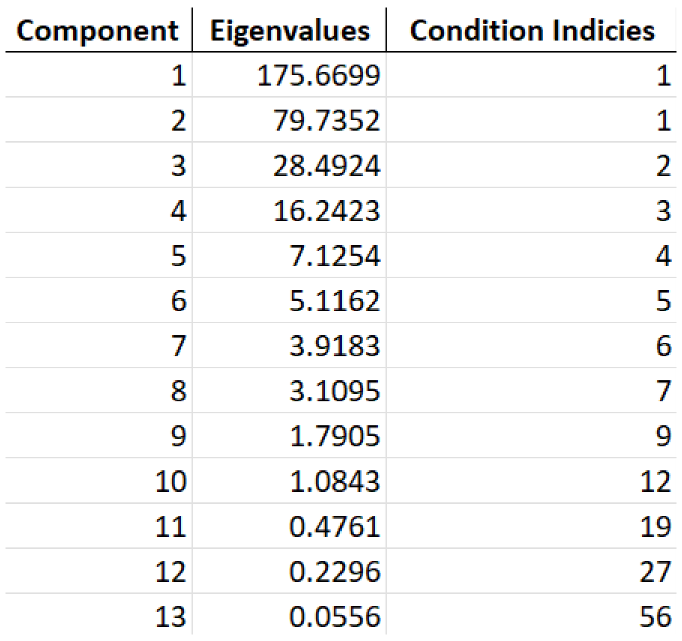


Figure 3.2: Scree Plot of Boston Housing Dataset

From the graph we can see that only about the first 4 components really play a significant role in describing the median house prices due to their large eigenvalues, especially seen in the first component. In the graph we can also observe a large drop in size of values after the first component which keeps dropping significantly until the 5th component. After this point the eigenvalues are all of similar size and relatively small compared to the first few. This is a good initial sign of the presence of multicollinearity within the dataset as very small eigenvalues implies that their is multicollinearity between the feature variables.

Now we compute the condition indices of each eigenvalue and put them together in a table to analyse in closer detail.



Component	Eigenvalues	Condition Indicies
1	175.6699	1
2	79.7352	1
3	28.4924	2
4	16.2423	3
5	7.1254	4
6	5.1162	5
7	3.9183	6
8	3.1095	7
9	1.7905	9
10	1.0843	12
11	0.4761	19
12	0.2296	27
13	0.0556	56

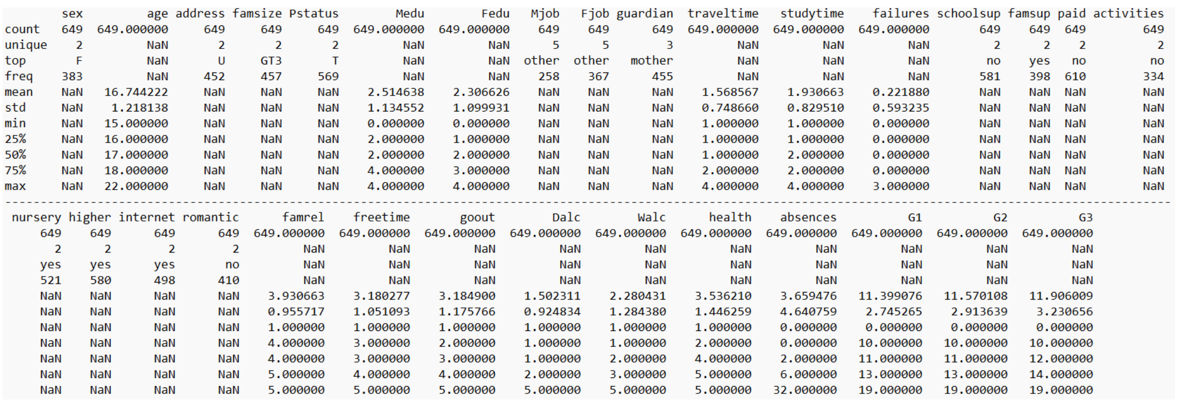
Figure 3.3: Eigenvalues and Condition Indices for Boston Housing Dataset

From the table we can see the difference in variance between the different components and especially how the last three values are very close to 0. This on its own is already a strong indicator of multicollinearity within the data, however, by looking at the condition indices we get an even better measure. From the table, four components come to attention when looking at the condition indices, these being the 10th, 11th, 12th, and 13th components as their values are all above 10 which is threshold we are using to consider the presence multicollinearity to some extent. The 13th component indicates particularly strong multicollinearity within the dataset with an eigenvalue of above 50.

3.2.2 For Student Performance Dataset

Description of Features

Before running the describe function on the dataset we first need to drop the 'school' and 'reason' features as these are just identifying variables and non-predictive. After this we run the pandas describe function to see if any values stand out in particular which can give us insight on the features and what values they contain.



```
        sex         age address famsize Pstatus        Medu        Fedu   Mjob   Fjob guardian  traveltime   studytime    failures schoolsup famsup paid activities
count   649  649.000000     649     649     649  649.000000  649.000000    649    649      649  649.000000  649.000000  649.000000       649    649  649        649
unique    2         NaN       2       2       2         NaN         NaN      5      5        3         NaN         NaN         NaN         2      2    2          2
top       F         NaN       U     GT3       T         NaN         NaN  other  other   mother         NaN         NaN         NaN        no    yes   no         no
freq    383         NaN     452     457     569         NaN         NaN    258    367      455         NaN         NaN         NaN       581    398  610        334
mean    NaN   16.744222     NaN     NaN     NaN    2.514638    2.306626    NaN    NaN      NaN    1.568567    1.930663    0.221880       NaN    NaN  NaN        NaN
std     NaN    1.218138     NaN     NaN     NaN    1.134552    1.099931    NaN    NaN      NaN    0.748660    0.829510    0.593235       NaN    NaN  NaN        NaN
min     NaN   15.000000     NaN     NaN     NaN    0.000000    0.000000    NaN    NaN      NaN    1.000000    1.000000    0.000000       NaN    NaN  NaN        NaN
25%     NaN   16.000000     NaN     NaN     NaN    2.000000    1.000000    NaN    NaN      NaN    1.000000    1.000000    0.000000       NaN    NaN  NaN        NaN
50%     NaN   17.000000     NaN     NaN     NaN    2.000000    2.000000    NaN    NaN      NaN    1.000000    2.000000    0.000000       NaN    NaN  NaN        NaN
75%     NaN   18.000000     NaN     NaN     NaN    4.000000    3.000000    NaN    NaN      NaN    2.000000    2.000000    0.000000       NaN    NaN  NaN        NaN
max     NaN   22.000000     NaN     NaN     NaN    4.000000    4.000000    NaN    NaN      NaN    4.000000    4.000000    3.000000       NaN    NaN  NaN        NaN
------------------------------------------------------------------------------------------------------------------------------------------------------------------
 nursery higher internet romantic      famrel    freetime       goout        Dalc        Walc      health    absences          G1          G2          G3
     649    649      649      649  649.000000  649.000000  649.000000  649.000000  649.000000  649.000000  649.000000  649.000000  649.000000  649.000000
       2      2        2        2         NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN
     yes    yes      yes       no         NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN
     521    580      498      410         NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN
     NaN    NaN      NaN      NaN    3.930663    3.180277    3.184900    1.502311    2.280431    3.536210    3.659476   11.399076   11.570108   11.906009
     NaN    NaN      NaN      NaN    0.955717    1.051093    1.175766    0.924834    1.284380    1.446259    4.640759    2.745265    2.913639    3.230656
     NaN    NaN      NaN      NaN    1.000000    1.000000    1.000000    1.000000    1.000000    1.000000    0.000000    0.000000    0.000000    0.000000
     NaN    NaN      NaN      NaN    4.000000    3.000000    2.000000    1.000000    1.000000    2.000000    0.000000   10.000000   10.000000   10.000000
     NaN    NaN      NaN      NaN    4.000000    3.000000    3.000000    1.000000    2.000000    4.000000    2.000000   11.000000   11.000000   12.000000
     NaN    NaN      NaN      NaN    5.000000    4.000000    4.000000    2.000000    3.000000    5.000000    6.000000   13.000000   13.000000   14.000000
     NaN    NaN      NaN      NaN    5.000000    5.000000    5.000000    5.000000    5.000000    5.000000   32.000000   19.000000   19.000000   19.000000
```

Figure 3.4: Output of describe method on Student dataset

From this breakdown the first thing we see is that the dataset has a number of categorical variables which are denoted by feature descriptions that have a non-null value in their unique statistic, this value indicates the size of the categorical group. There are 15 of these variables and each of them will need to be encoded in order for further data analysis. To do this I will be using the dummy variable method which creates n-1 new columns for every categorical variable, where n is the number of unique groups within in that specific column. These new columns will represent one of these categories and a row will have a 1 if that samples is part of said group and 0 if it does not, n-1 is used to prevent the creation of redundant variables that can cause unnecessary multicollinearity. After the encoding and dropping of the original categorical columns our dataset now has 38 features.

Also, from the descriptions of the last 3 grade features: G1, G2, and G3 we can see that their means and standard deviations are very similar to each other. This is a good indicator that their will be a high correlation between these variables, this also makes sense logically as students who do well usually continue to do well and vice-versa.

Correlation Matrix

After the encoding we can now compute the correlation matrix of the new dataset to observe how correlated the features are with each other and if they are to a degree that might indicate collinearity. Due to the size of the correlation matrix we will first display the heat map without the correlation values and then if any areas are identified as highly correlated than they can be further investigated by inspecting their values.

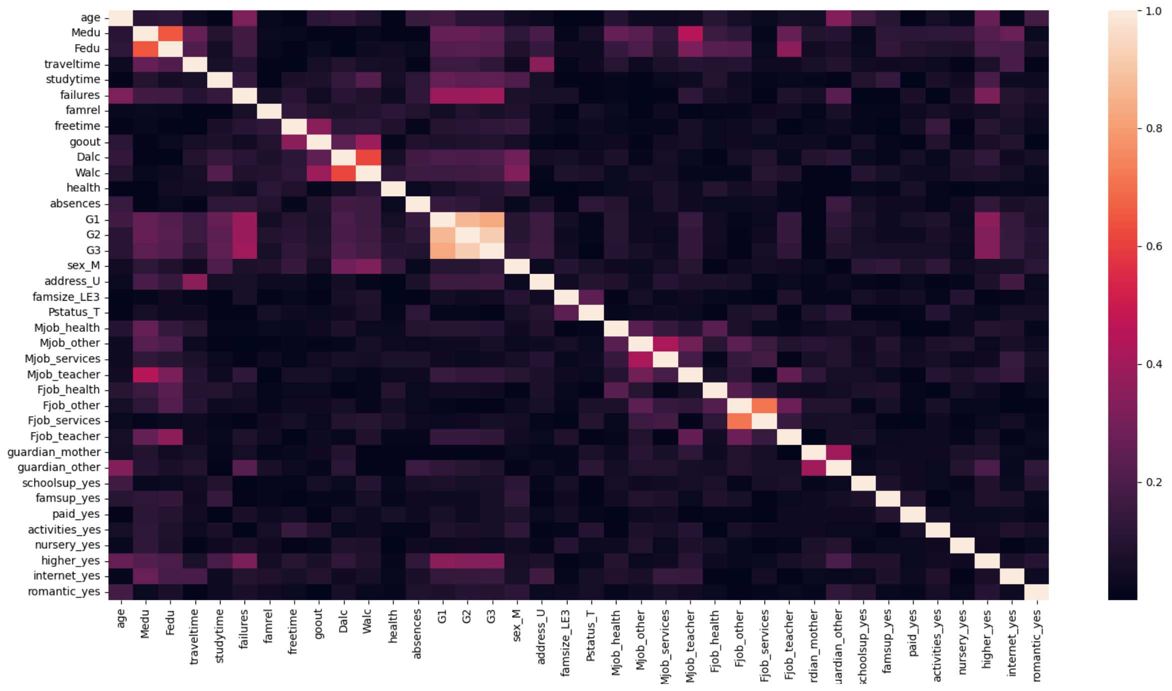


Figure 3.5: Heat map of correlation matrix of student dataset without values

From this initial heat map there are 4 sections that seem to have high correlation values, with one of these showing strong signs of collinearity. This occurs between the G1, G2, and G3 variables as we predicted. The other areas appear between the Fedu and Medu, Walc and Dalc, and Fjob_other and Fjob_services variables.

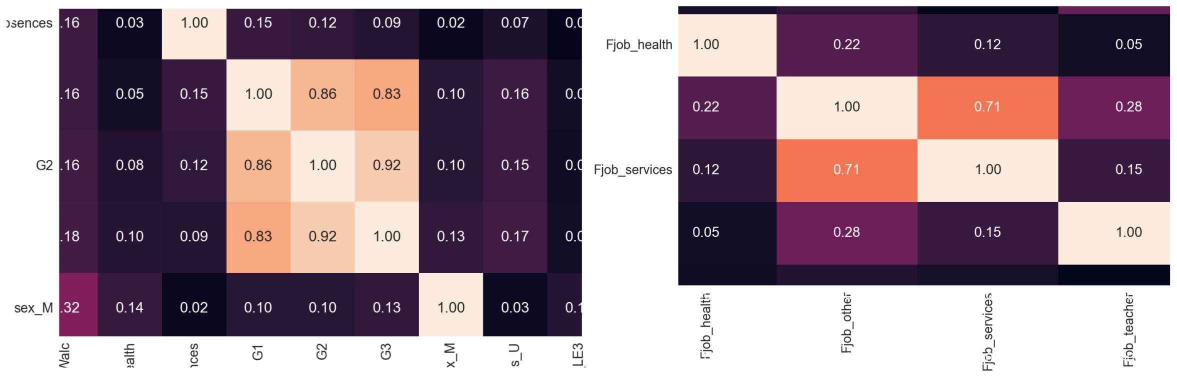


Figure 3.6: Significant correlation values from heat map

After closer inspection only 2 of these identified areas had correlation scores equal to or above 0.70, our boundary for significant correlation. The biggest of which are the grade scores 1 and 2, where G1 and G2 had a correlation score of 0.86. The scores of G1 and G2 are above 0.70 which indicate a high correlation between themselves which can be a cause for concern as collinearity is highly likely to be present within the dataset. In the other section we see that the features Fjob_other and Fjob_services also have a high collinearity score of 0.71 which can further exacerbate the collinearity within the dataset.

Finally, we can also use the heat map to see if any features are significantly correlated to the target variable of G3. This can give us a good indication of what variables will be most important when predicting the target and also how well the data describes the target variable. However, from the heat map we see that no correlations score are really significant except for the correlation of G1 and G2 with G3, where G1 and G3 has a score of 0.83, and G2 and G3 a score of 0.92. This is considered a high and very high correlation score between the first 2 grades and the final. This shows that the model will rely heavily on these two features, it can also be an indicator that the model will have difficulties learning from the dataset as the overall correlation between the features and target variables are low.

Eigenvalues & CIs of Covariance Matrix

Just like for the previous dataset, we can now compute the eigenvalues of this dataset and generate its condition indices.

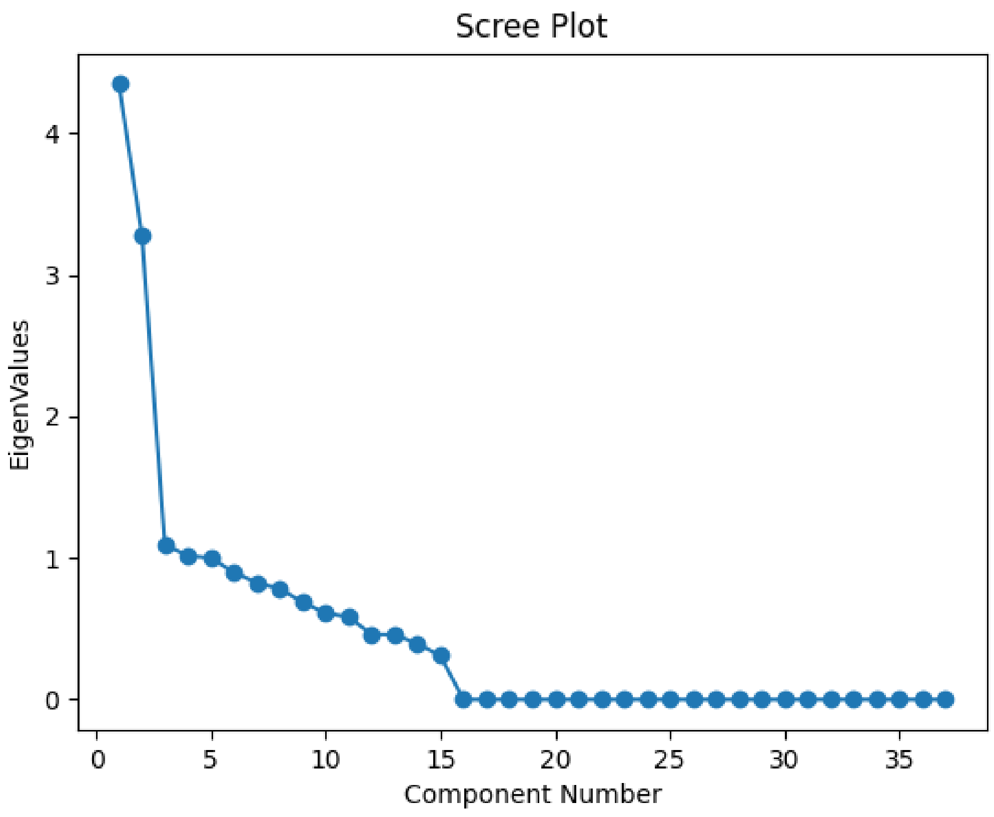
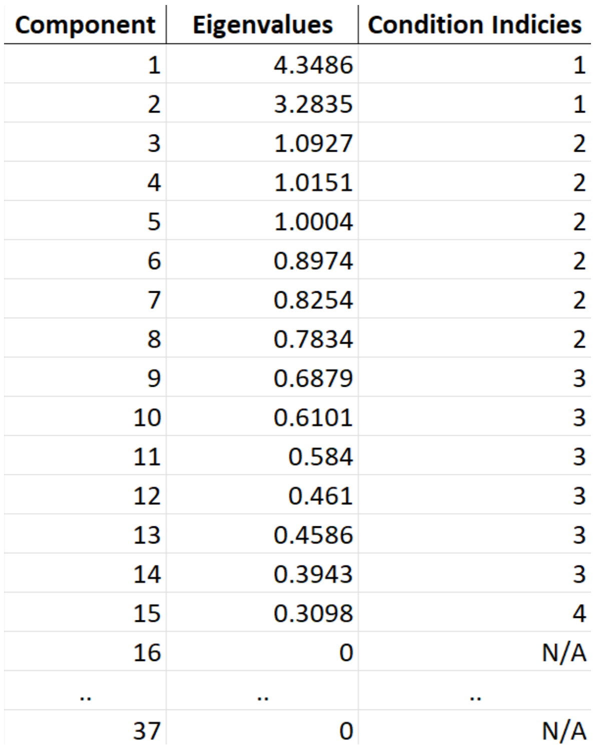


Figure 3.7: Scree plot of Student Dataset

This plot is interesting as, just like in the last dataset, their is a steep drop off after the first couple of components. However, after this the next eigenvalues stay at roughly the same values which is then followed by another final drop off. The first observation to be made from this is the first 3 principal components and how they seem to describe most of the variance within the dataset. This can be attributed to the 3 grade variables seen in the correlation matrix. Even though there is a reasonable drop after these 3 components, the size of the eigenvalues are much lower than those of the last dataset. This will result in smaller condition indices as the difference between each component with the principal component will not be as large.



Component	Eigenvalues	Condition Indicies
1	4.3486	1
2	3.2835	1
3	1.0927	2
4	1.0151	2
5	1.0004	2
6	0.8974	2
7	0.8254	2
8	0.7834	2
9	0.6879	3
10	0.6101	3
11	0.584	3
12	0.461	3
13	0.4586	3
14	0.3943	3
15	0.3098	4
16	0	N/A
..	..	..
37	0	N/A

Figure 3.8: Eigenvalues and Condition Indices for Student Dataset

From the table we can see that in fact our condition indices values are small and under our boundary of significance of 10. However, something very interesting occurs between component 16 till 37 and that is that their eigenvalues are all equal to 0. This indicates that the covariance matrix is not of full rank and therefore, collinearity is present between many of its features. This result was probably caused by the large amount of dummy variables created when encoding our categorical data. This will probably result in a generally poor performance in both of our models when predicting on this data, especially paired with the low correlation values with the target variable. However, the ridge regression algorithm should still perform better as it will generalise the model heavily in an attempt to achieve better results, using cross validation will most likely find a large alpha value as the optimal hyper parameter.

3.2.3 Crime Dataset

Description of features

Like on the previous datasets we will also need to drop some features that are not relevant to our models before running the describe method. This includes non-predictive variables that were mainly used to identify the communities that took part in the study. Also, the other possible target variables will need to be dropped so that only the "ViolentCrimesPerPop" target variable remains. After we are left with our relevant feature variables and target variable we can run the describe function.

From this statistical information we can make some observations about the dataset. Firstly, we can confirm the limitations stated by the authors that a lot of the data related to the law enforcement of each community is missing from the count statistic of the police related feature names. Also, looking at the count value from the target and "OtherPerCap" variables we see that some rows have these missing. This means that both the police features and the samples missing the target and "OtherPerCap" variables need to be removed from the dataset to clean it before building the correlation matrix.

Correlation Matrix

After cleaning the data we can display the heat map of the correlation matrix to examine how the features interact with each other. Similar to the previous dataset, because of the number of features the heat map will initially be displayed without values or labels.

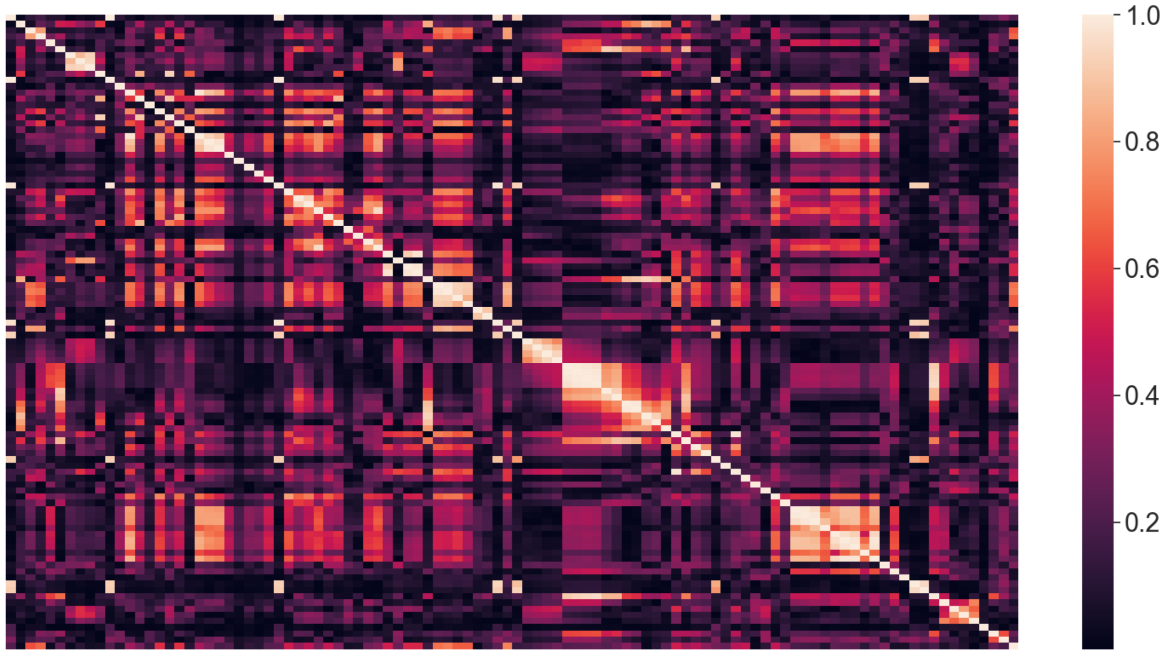


Figure 3.9: Heat map of correlation matrix of crime dataset without values

From this initial heat map we can see that there are several different instances of correlations scores that will indicate high and very high correlation between the variables. These proved to be too many to analyse by hand when trying to zoom into the heat map to get the specific values. However, general observations can still be made such as the presence of many highly correlated variables indicating that collinearity is present thought the dataset. We can also see that along the bottom row we can see some significant correlation variables between the different features and the target variable. This is an indicator that the dataset does describe the target variable to a certain extent and that our model should be able to perform reasonably well.

Eigenvalues & CIs of Covariance Matrix

Due to the limitations recently seen of having a very large number of features we will need to rely on the eigenvalues and condition indices of our covariance matrix further examine our dataset. This is a great example of how useful this technique can be when dealing with these larger and more complex datasets. Firstly, lets compute the covariance matrix and take its eigenvalues to see how much each component describes the target variable, we will once again use a scree plot to visualise these values.

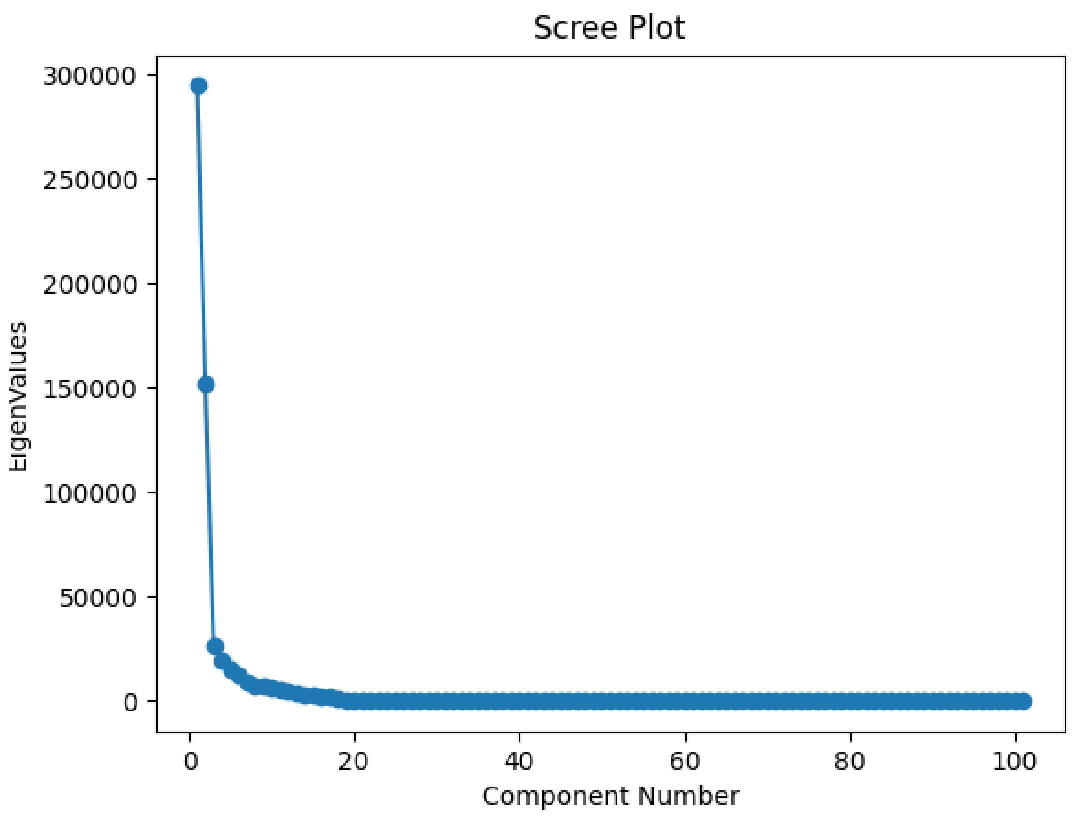
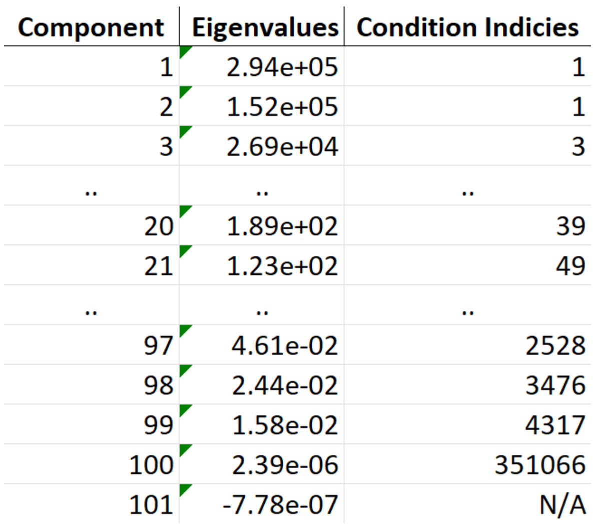


Figure 3.10: Scree plot of Crime Dataset

From the plot we can observe a huge drop between the first 3 eigenvalues and the rest of the components. The difference between the magnitudes of these values also indicates that these 3 components play an overwhelming part in describing the target value. From this graph we can tell that the condition indices will all be very high simply due to the size of the first 3 values. Due to the number of components I will only be showing some important values from the table.



Component	Eigenvalues	Condition Indicies
1	2.94e+05	1
2	1.52e+05	1
3	2.69e+04	3
..	..	..
20	1.89e+02	39
21	1.23e+02	49
..	..	..
97	4.61e-02	2528
98	2.44e-02	3476
99	1.58e-02	4317
100	2.39e-06	351066
101	-7.78e-07	N/A

Figure 3.11: Eigenvalues and Condition Indices for Crime Dataset

Looking at our table we can confirm that the condition indices are extremely high for this dataset. The first 19 components still have their indices below 30, however, from component 20 till 100 all the indices are of massive size indicating that the dataset suffers from severe multicollinearity. Another interesting result worth noting is the eigenvalue of component 101, and that it is a very small negative number. This is most likely due to a over/underflow error when computing the eigenvalues of the covariance matrix due to its size and it possibly being ill-conditioned. This can cause very small eigenvalues to become negative which is probably what happened in this scenario. Similar to the previous dataset, the ridge regression model will most likely choose a very high alpha hyper parameter value to try and generalise the model as much as possible to counteract the severe multicollinearity within the dataset.

3.3 Data Preprocessing

Now that we have chosen and become familiar with our datasets we need to do some data preprocessing before they can be used to train and test the machine learning models. The reason for preprocessing the datasets is to try and ensure that we get an as accurate and reliable as possible model that is not negatively affected by any inconsistencies found within the data it uses. This task ranges from removing missing or inconsistent data found in the set to applying normalisation techniques in order to standardise the data for better performance.

3.3.1 Normalisation

Normalisation is a vital step in the preprocessing process, this is especially true in the case of ridge regression and K nearest neighbours which will be explained later. Firstly, normalisation is the task of transforming all the features in a dataset so that they are all on a similar scale.[34] This creates features which numerical magnitudes are uniform across all other features. If the features are not measured on the same scale then they will end up not contributing equally to the training of the model which allows for bias to be introduced into the model.

This is twofold in a ridge regression model as this causes its penalty term to also be biased against particular features further destabilising the model. This is also important for the K nearest neighbours algorithm as it depends on measuring the euclidean distance of the test data point with its nearest neighbours. The position of this point is based on the size of its features and if they operate on different scales then bias will also be introduced into its model.

Standard Scalar

In this report I have chosen to utilize the StandardScaler() function, that is provided through the sklearn python library, to normalise each dataset before being used by the ridge regression model to ensure that that the best results are obtained from the algorithm. The goal of the standard scalar is to transform all features from a set of data so that their mean is equal to zero and their variance equal to 1, i.e., bring all features to the same scale. This is done in two steps: first, shift each feature down by its mean, then divide each feature by its standard deviation.[35]

I have chosen the standard scalar method of normalisation for the ridge regression model due to how it regularises the features with its penalty term. If the mean of each feature is not set to zero in the dataset, then the penalty term will bias the model away from the average data point which is what we are trying to achieve. Also if the variance of all features is not set to 1 then the penalty term will penalize some features more than others just due to the magnitude of their values and not necessarily because they are less important predictors.[36]

I chose this same normalization method for the K nearest neighbours model due to it relying on distance measures, Euclidean distance in the case of this report, to determine the similarity between data points. If the features have different scales then features with larger scales may dominate the distance calculation and create bias within the model. Therefore, normalizing the data via the standard scaler ensures that all features contribute equally to the distance computation by scaling them to a similar range.

Avoiding Data Snooping

Data snooping occurs when some of the test set ends up being used to train a model. This negatively impacts the model as its performance is likely to be artificially inflated compared to how it would really perform on new unseen data samples. This can happen obviously if rows of data from the test set are present within the training set, however, the test set can also leak into the model through other subtler ways. One of these is through improper handling of data when applying normalisation methods.

The correct way to normalise our datasets is to first find the transformation, which will lead to the specific normalisation, from the training set. This transformation is then applied to both the training set and the test set, this ensures that the test set has no influence on the values in the training set.

3.4 Accuracy Metrics

An important part of creating a machine learning model is having a means of objectively measuring the accuracy of the predictions made by the model. This is necessary as without this metric it would be challenging to evaluate the effectiveness of the model, making it difficult to implement meaningful modifications and observe if these changes have any positive effect. There are different metrics for measuring the accuracy of a model predicting on regression versus classification problems, as in classification problems the prediction is either correct or incorrect while in regression problems a numerical value is being predicted which can be closer or farther from the target value.

3.4.1 R^2 Score

For this report I have chosen to use the R-squared (R^2) statistical measure as a means to determine the goodness of fit of my Ridge Regression model. The R^2 score describes the variability in the labels predicted by the model, based on the features that were used to train it. This score is represented as a number between 0 and 1. A score of 0 describes a model that has not learnt any relationship between the features and the labels of the dataset, while a score of 1 means that the features perfectly describe the value of the labels. Any score x between 0 and 1 describe a variation of $(x \times 100)\%$ in the predicted labels given the features used to train the model.[37]

When evaluating the performance of the models a higher R^2 score will demonstrate a more accurate and higher performing model. It is important to note that the R^2 score will be computed on the predictions made from the test set and not the training set. This is done in order to analyse how well the models perform on unseen data, as the main point of Ridge Regression is to generalize the model making it more effective on new data which the model has not been trained on. This score value will be the main method to objectively analyse and compare the effectiveness of the Ridge Regression and K Nearest Neighbours models on each dataset.

To ensure that the scores are statistically valid I will compute them 5 times on each dataset for each model. Each time the random state of the train_test_split() method will be changed, resulting in different data samples within the training and test sets. The mean of these score will then be taken along with their standard error in order to create a general score along with uncertainties for the results.

3.4.2 Result Visualisation

Because this report aims to explore many different conditions under which the algorithms perform, there will also be different ways of displaying the results. This is done as one way of displaying them might be more appropriate than the other in regards to readability and how easy it is to visualise trends within the gathered data.

Table Display

The R^2 accuracy scores of these models, along with their standard errors, will be recorded and displayed in a table in order to compare the results of the algorithm on each dataset versus one another. This table will consist of two columns, one for each algorithm, and three rows, one for each dataset. This way of displaying the results was chosen as it makes it easy to quickly look and understand how the scores of each algorithm differ from each other on the same tasks, i.e., performance on each dataset.

Predicted vs True Plot

Along with the table explained above, there will also be plots for each model and dataset pair that describe how the predicted target values made by the model compare to the true values from the dataset. This is displayed as a predicted vs true graph, where each point is a sample from the test set. This point on the x-axis denotes what is the true target value of that sample, i.e., the target value stated in the dataset, and on the y-axis the target value predicted by the model for that sample is shown. This means that with a perfect accuracy score all the points on the graph would sit perfectly on a diagonal line going through the origin of both axes. Plotting the results in this manner is useful as it allows for intuitive visualisation of the models performance. When comparing the plot of two algorithms together one simply needs to identify the model which has most of its points closer to the diagonal of the graph.

Accuracy vs Parameter Plot

Finally, I will explore how different parameter values affect the performance of the models. To display this I will use a plot similar to the one described above, but instead the x-axis will represent the varying values set to be the parameters (alpha and K). Once again there will be two graphs, one for each algorithm and 3 separate lines will be plotted in each that represent their performances on different datasets with different parameters.

3.5 Parameter Tuning

As mentioned in the theoretical section, the cross validation technique will be used to find the best values for the hyperparameters alpha and K for the ridge regression and K nearest neighbour models respectively. This is done to ensure that both algorithms are given the best setups so that their performances are as optimal as possible, with only the algorithm themselves defining how well they score. In machine learning cross validation is a way to quickly assess the performance of a model on unseen data, this can be extended so that many models are created and evaluated with different parameters. The parameter of the model with the best score will be considered the most optimal parameter for that specific algorithm and dataset, and will be used as our parameter when gathering our results.

3.5.1 Cross-Validation

This method works by firstly splitting the original dataset into a training and test set. The idea of using cross-validation to find optimal model parameters is to first choose a set of possible values for your parameters, then for each parameter crate a model with those values and then run cross-validation on the training set only with said value. This process is repeated for all possible parameters in the specified set, with the best score and corresponding parameters being saved. At the end of this process a new model is built with the best parameters from the loop. This model is then trained on the whole training set and tested on the training set, the value of the new models score and the best cross validation score is compared to see if those parameters are truly the best.

This strategy of first splitting the dataset and applying the cross-validation to only the training set is done to prevent data snooping as using the test set to even find parameters means that it is being used to develop your model. This technique will be done for each model on all datasets to find their most optimal parameters for each dataset. This value will be used when computing their r-squared scores which will be used when comparing the two algorithms.

Chapter 4: **Experiments & Results**

The aim of this study is to analyse the performance of the ridge regression algorithm and compare it to the performance of the K nearest neighbours algorithm. This will be done by constructing two different models, one using ride regression and the other with k nearest neighbours, and then applying them to 3 different datasets with different numbers of features. Each model will be run five times on every dataset with different random splits to ensure that the resulting accuracy scores are statistically significant, the standard error will also be computed and recorded. This is done to see if changing the number of features effects the performance in any way of these two algorithms.

Along with running the models on datasets with different quantities of features, I will also experiment with different split sizes of the training and test sets. As mentioned before in the theoretical section, the size of the training set can lead to overfitting problems if it is too small as the model is not able to generalise due to too few training data points. In these situations the ridge regression algorithm may perform to a high degree, especially in comparison to the k nearest neighbours model.

I will explore how different parameter values affect the performance of these two algorithms on the dataset. This is to see how big a part does parameter selection play a role in the performance of these two algorithms and to have an idea how the accuracy changes with respect to the parameter values.

Lastly, observations will be made on the general usability of these algorithms and how they differ from each other. This includes how computationally efficient they are and also how interpretable they are, in the sense of how does each algorithm help us understand the relationship between the feature and target variables.

4.1 Number of Features

As mentioned above the mean accuracy scores for each algorithm on each dataset are computed and displayed in the following table, where the scores in bold are the most accurate from that dataset. These values were achieved when using a train test split of 0.75 for the training set. This is the default value used by sklearn and will be used to get this initial comparison of the algorithms.

	Ridge	KNN
Boston Dataset	0.7345 ± 0.0210	**0.7896** ± 0.0117
Student Dataset	**0.8441** ± 0.0074	0.5173 ± 0.0213
Crime Dataset	0.5652 ± 0.0186	**0.5776** ± 0.0108

Table 4.1: Results when training set size is set to 0.75

From the table above we can see that the ridge regression algorithm performed better than K nearest neighbours on only one of the datasets, this being the student dataset with around 30 variables. The other two datasets Boston and Crime, with around 10 and 100 features respectively, returned better scores with the K nearest neighbour model compared to the ridge model.

Looking at these initial results it seems that their is not much correlation between number of features and how well the ridge regression model performs. It was expected that as the number of features increases so does the performance of the ridge regression algorithm. Due to it being able to generalise the complex model which is caused by the higher dimensionality of the data. However, this relation can still be further explored when running the same predictions but when using different ratios of training and test splits.

From the initial results, it also seems that the K nearest neighbours algorithm out performs the ridge regression algorithm on these datasets which suffer from multicollinearity. However, looking closer at the difference between the score of the ridge and K nearest neighbour models we see that in the instances that K nearest neighbours had a higher score it was by a relatively small amount. And when predicting on the second dataset the ridge model vastly out performed the K nearest neighbours model. Therefore, on average it can be said that both algorithms performed roughly at the same level. However, more results must be attained in order to draw more concrete conclusions.

4.1.1 Different Training Set Sizes

To further examine and compare the performance of these two model I will compute and display the accuracy scores of these algorithms on the datasets when using different training and test set splits sizes. There will be 3 tables where a train test split size of 0.75 (same as above), 0.50, and 0.25 was used to train and score the models.

	Ridge	KNN
Boston Dataset	0.7345 ± 0.0210	**0.7896** ± 0.0117
Student Dataset	**0.8441** ± 0.0074	0.5173 ± 0.0213
Crime Dataset	0.5652 ± 0.0186	**0.5776** ± 0.0108

Table 4.2: Results when training set size is set to 0.75

	Ridge	KNN
Boston Dataset	0.7371 ± 0.0110	**0.7587** ± 0.0065
Student Dataset	**0.8094** ± 0.0038	0.4809 ± 0.0112
Crime Dataset	0.4862 ± 0.0649	**0.5818** ± 0.0045

Table 4.3: Results when training set size is set to 0.50

	Ridge	KNN
Boston Dataset	0.6497 ± 0.0307	**0.7106** ± 0.0118
Student Dataset	**0.5845** ± 0.1094	0.4138 ± 0.0175
Crime Dataset	0.4229 ± 0.0639	**0.5383** ± 0.0066

Table 4.4: Results when training set size is set to 0.25

By looking at the tables we can see that as the training size decreases so does the accuracy of both models on almost all occasions. This is expected as with less training data the models have less information to learn from to make predictions. However, against my expectations the ridge model did not outperform the K nearest neighbours model as the size of the training set decreased. Both algorithms suffered from the smaller training set but the K nearest neighbour model seems to have stayed relatively close to its previous levels. The ridge model on the other hand has had a big drop in accuracy, especially when the training split was equal to 0.25 as seen by the results from the table.

Another observation from the tables are the standard errors of each algorithm and how they vary depending on the training split size. The K nearest neighbours model seems to be stable as it has and maintains a low standard error even as the training set gets smaller. In regards to the ridge model it produced a low standard error when given the default training set split, similar to that of the K nearest neighbours model. However, the error starts to drastically increase as the split size decrease, this is especially apparent in the 3 table with errors as high as 0.1094.

Computational Complexity

A final but very important observation about these two algorithms is their respective run times and computational complexities to implement. The ridge regression algorithm only needs to compute its regression coefficients once from the training set, after which it can quickly apply it to any new samples to create a prediction. K nearest neighbours on the other hand needs to go through the whole training set every time to calculate the distances between them and the new sample to compute its prediction. Because of this K nearest neighbours can take a significantly longer time to compute its predictions compared to the ridge algorithm. This time is even more exacerbated by the size and dimensionality of the dataset being applied to the model.

In regards to the first dataset both algorithms had reasonably similar runtimes but a difference is already present with ridge taking around 2 seconds and K nearest neighbours 7. This small difference is due to the Boston dataset being relatively small and low dimensional with only 13 features. In the second dataset, this difference becomes even larger as the ridge model continues around 2 seconds while K nearest neighbours now takes 14 seconds to complete all computations. This is most likely due to the increase in features present within the dataset as the algorithm needs to compare every single feature variable of a sample in order to compute the euclidean distance. Finally, when predicting on the third dataset the K nearest neighbours algorithm took 2 minute and 4 seconds to predict on the test set. This might not seem as much but compared to the once again almost instantaneous runtime of the ridge algorithm it is an extremely large and significant result.

This difference in runtimes could be very problematic depending on the application of the model. For example, if hundreds of thousands of predictions need to be made then the K nearest neighbours algorithm is simply to slow and would not be able to perform adequately. The ridge regression algorithm on the other hand would be optimal for these situations as it is extremely fast with only sacrificing a minimal degree of accuracy provided that it has enough training data. However, in situations where you require very few and precise predictions, such as some scientific applications, then K nearest neighbours can be beneficial as it has time to complete its computations and gain a deeper understanding of the patterns within the data.

4.2 Different Hyperparameter Values

Now we shall explore the relationship between the values of the hyperparameters of each model and how this affects their accuracy. Our data will consist of the R-squared scores of each algorithm on each dataset with a ranging values for their own respective hyperparameters. The default train test split will be used to gather these results of training set = 0.75.

4.2.1 K Nearest Neighbours

We will start by examining the hyperparameter of the K nearest neighbour algorithm and how assigning it different values affects the accuracy of the model on the different datasets.

The hyperparameter K specifies how many of the closest neighbours to the new sample should be included when predicting the new target variable.

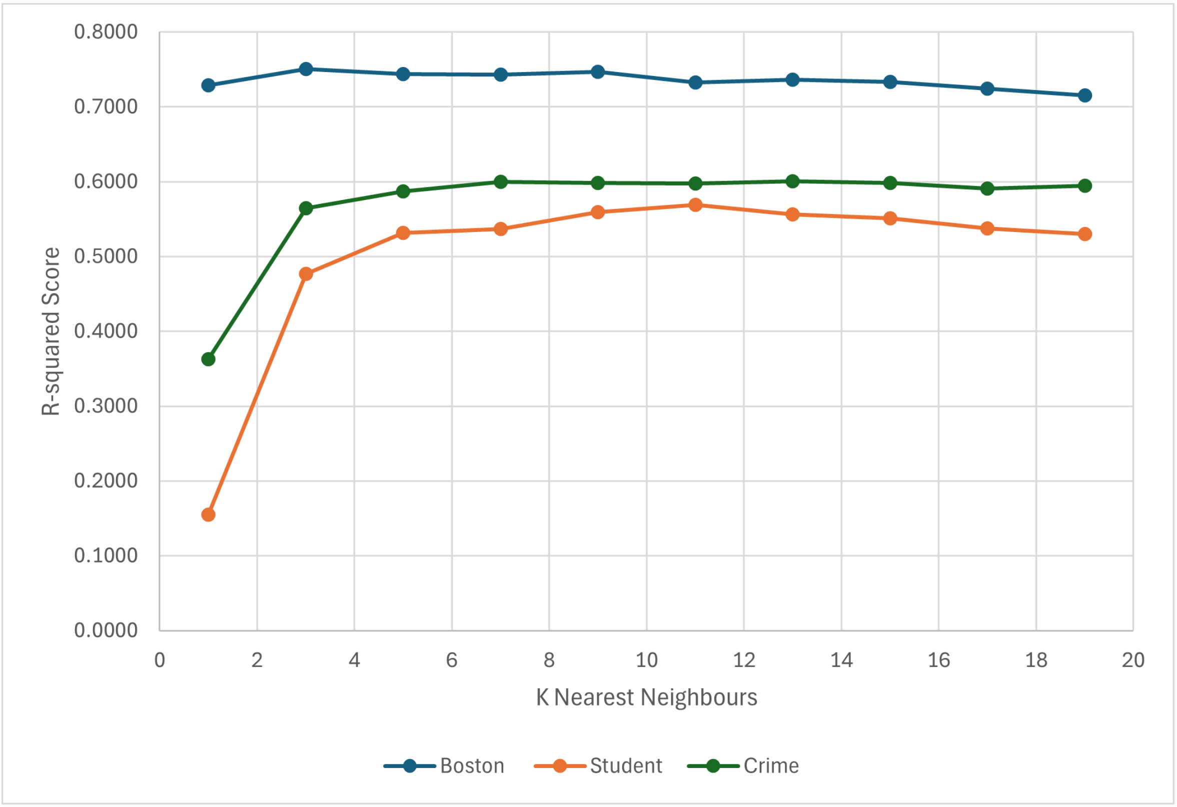


Figure 4.1: Plot of how accuracy varies with different K values

This graph is very useful and gives us a lot of insight on how K affects the accuracy of the regression model. Firstly, one thing that occurs equally in all datasets is the general pattern between the parameter value and the score. At k equals 1 the model performs poorly on all datasets in comparison to the other values, this can be seen by all score going up significantly from k = 1 to k = 3. This occurs due to the model overfitting at K = 1 as an exact value from the training set is used as the prediction for the unknown sample. Also, after K = 3 the performance of the model on the datasets plateaus to a certain extent and further increasing the value of K does not have much effect on accuracy. Further increasing K can result in slightly better performance, however, simply using a value of 3 or 5 regardless of the dataset usually will mean a close to optimal performance.

4.2.2 Ridge Regression

Moving on to the ridge regression algorithm, we will now plot the same graph having alpha on the x-axis instead of K. The hyperparameter of the ridge regression algorithm is fundamentally different from that of the K nearest neighbours algorithm. It works by dictating how strong the penalty term should be, i.e., by how much should the model be generalised.

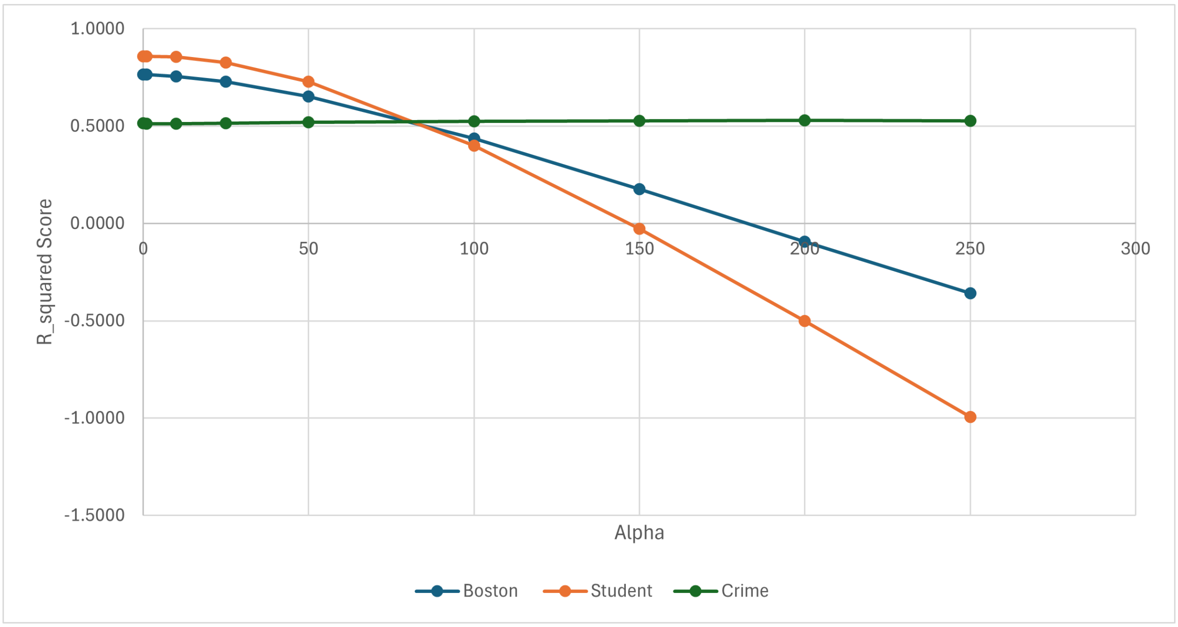


Figure 4.2: Plot of how accuracy varies with different alpha values

Firstly, comparing this graph to the previous one we can see that it has a completely different shape and relationship between the models hyperparameter and accuracy. Its slightly hard to see from the graph due to the different scales of alpha and how they interact with each dataset, but the pattern is that for each graph there is an alpha value were max accuracy is achieved. If alpha is set to a value either smaller or larger than this optimal value the models performance will decrease. This graph also shows the importance of finding the optimal alpha value for a dataset and how quickly the models accuracy can suffer if a wrong value is chosen. This graph can also help us understand our data and how much each dataset sufferers from multicollinearity. Datasets who suffer less will have optimal alpha values closer to zero as less generalisation needs to be done, demonstrated in the Boston dataset that was the dataset which least suffered from multicollinearity. The crime dataset on the other hand suffered greatly from multicollinearity as seen by the eigenvalues of its covariance matrix. This same conclusion can confirmed through the graph as its optimal alpha value is around 150, while Bostons optimal alpha is around 0.01.

4.3 Conclusions

Referring back to our original objective of analysing and comparing the performance of the ridge regression and K nearest neighbours algorithms, I believe that they both have their specific use cases where they can outperform the other. As seen from our results of R-squared scores and different datasets with different number of features I believe, in terms of pure accuracy, that the K nearest neighbours algorithm outperforms the ridge algorithm. This is due to the unique way that K nearest neighbour computes its predictions using euclidean distance opposed to linear regression. This technique allows the model to capture very detailed information and relationships between feature variables from the dataset and apply it when create new predictions. The ridge regression algorithm managed to perform to a close level when given enough training data, however, with smaller training sets the accuracy decreased as well and high standard errors were observed indicating that the model had become unstable.

With this in mind, another important aspect of performance is usability and how practical it is to implement these algorithms to solve real world problems. In this aspect, the ridge algorithm out performed the K nearest neighbours algorithm. This is due to the computational complexity and runtime of the K nearest neighbours algorithm, causing it to become very slow on bigger and more complex datasets. Ridge regression on the other hand can create predictions almost instantly after having fitted its training data only once, and if given enough training data it can perform accurately to a similar standard of the K nearest neighbours algorithm.

Finally, in regards to each of their hyperparameters the K nearest neighbours parameter K is relatively straightforward to optimise as not much variance in accuracy occurs after passing the value of 3 or 5. This makes it easier to implement as no specific or exhaustive fine-tuning is necessary to obtain accurate results. However, finding the optimal alpha parameter of the ridge algorithm on the other hand is extremely important in order to build and accurate model. This value is hard to find however, and can vary greatly across different datasets. At the same time, this parameter is what allows the algorithm to perform accurately and quickly on what are very large and complex datasets where other algorithms would suffer on.

In conclusion, both algorithms have their own applications to specific real world situations. Overall, K nearest neighbours is more accurate than ridge and can be applied to very complex datasets which require precise predictions. Ridge on the other hand is very good at dealing with severe multicollinearity and still not have its runtime affected by this, making it very useful in situations where a large amount of predictions must be made where many features and samples are involved.

Chapter 5: **Professional Issues**

Ethics is and always has been an intrinsically aspect of computer science, however, now a days it has become an absolute necessity to consider and discuss especially due to the recent advancements in machine learning and how prevalent it have become within our society. Due to the huge amounts of data available to train new machine learning models it is necessary to be conscious and aware of the possible dangers this could pose to our society. In this section I will talk specifically about how personal data being used to train models could cause unintentional prejudice within machine learning models and how this can pose a threat to the way our societies view machine learning and AI.

This issue first came to my attention when I was researching the Boston Housing dataset introduced in the project description of my course. After reading up about it I came across some articles which raised several ethical considerations that need to be acknowledged and addressed.

Firstly, privacy concerns emerge when working with datasets that contain information about individuals or households. Although the Boston Housing Dataset doesn't include explicit personal identifiers, it still comprises attributes like median home value, crime rates, and socio-economic characteristics, which could indirectly reveal sensitive information about specific communities or individuals. Not only is this an ethical consideration it is also a very important legal one that can come with severe penalties if breached. Therefore, ensuring data anonymization and implementing strict access controls is crucial to mitigate potential privacy risks.

The biggest issue with the dataset is the historical context in which it was collected and how this introduces ethical implications related to fairness and bias. The data reflects housing dynamics during a period marked by racial segregation and discrimination. Thus, if not handled carefully, the models trained on this dataset might perpetuate or exacerbate existing biases, leading to unfair outcomes, particularly for marginalized groups. This is why awareness about ethical issues is important in this field as these possible biases can be introduced in many similar datasets and models even if the computer scientist had no intention whatsoever to do so.

Another ethical consideration lies in the potential misuse of the dataset and the models trained on it. Deploying predictive models based on biased or flawed data in real-world applications, such as housing policy decision-making or resource allocation, can have detrimental consequences, amplifying existing inequalities and perpetuating social injustices. To combat this it is of upmost importance to be transparent with what data you are working with and the connotation that your work might have. In my case this means recognising that this dataset has ethical considerations and that I am using it for the purpose of investigating performance of machine learning algorithms and not any specific real world application.

Chapter 6: **Software Engineering Method**

When creating my program I followed the standard software engineering practices by using the DevOps application GitLab to store and organise my directories and files. I used the creation of feature branches to keep track of my work and commit messages to describe and document my progress. When writing my code I made sure to annotate it with comments to improve readability and to document my program.

Chapter 7: **Self Evaluation**

In general I believe that I succeeded in creating an informative and exploitative report. During the whole process of learning, coding, writing, and discussing I felt that I was always bettering myself and learning from mistakes I committed along the way. My results were not exactly what I had predicted from my research, however, I still managed to draw meaningful conclusions from the results that I can apply in other situations. I also managed to successfully explore and learn from my experiments with different parameters and how they are related to the models performance. Furthermore, I increased my knowledge on the area of mathematics, specifically in regards to diagnosing multicollinearity and the use of eigenvalues and vectors which have a huge possible area of different applications that I can use in the future.

Some of the issues that I had with my report was definitely time management as this was the first time I had a project of this magnitude to complete. However, I managed to catch and correct myself before getting to close to the deadline which meant I was still able to create a careful and accurate piece of work. In regards to my results I believe that if I had repeated each experiment 10 times with 10 random states, instead of 5, I might have gotten results which were more in line with what I had predicted would happen. That being said it may have been overambitious for me to try and analyse 3 different datasets side by side. If I had stuck to one or two I might have been able to run more through and in depth experiments with the regression algorithms, possibly resulting in more definitive conclusions.

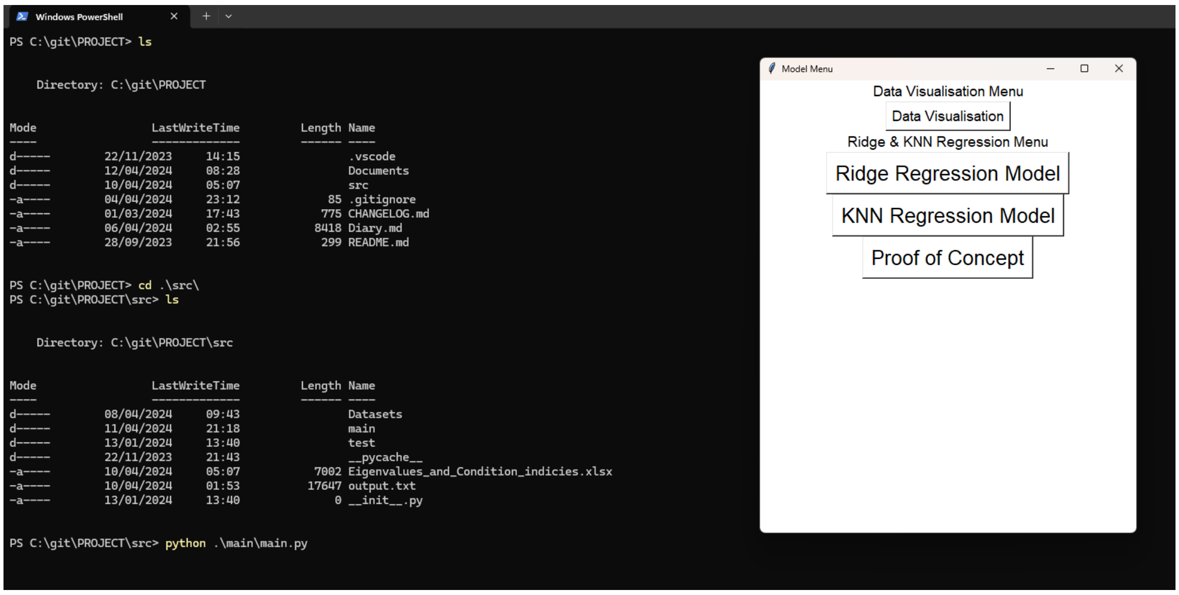
Currently my next goals are to pursue a job in the field of computer science, specifically machine learning or data science in order to get some real world experience. However, I also plan at some point to come back to university and achieve a postgraduate in machine learning as I both love how this field works and also because I believe that it will define our future and I would love to be at the tip of that spear.

Appendices

Appendix A: **Running the Program**

In order to run the program I created for this report you first need to install python, I used 3.10.8, and some libraries, these include: tkinter, numpy, matplotlib, pandas, seaborn, IPython, openpyxl, and scikit-learn.

After installing and opening the submitted folder, open a command prompt within the directory. Next move into the src directory, from here you can run the main.py file which will initiate the GUI. This process is demonstrated in the image below



```
PS C:\git\PROJECT> ls


    Directory: C:\git\PROJECT


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        22/11/2023     14:15                .vscode
d-----        12/04/2024     08:28                Documents
d-----        10/04/2024     05:07                src
-a----        04/04/2024     23:12             85 .gitignore
-a----        01/03/2024     17:43            775 CHANGELOG.md
-a----        06/04/2024     02:55           8418 Diary.md
-a----        28/09/2023     21:56            299 README.md


PS C:\git\PROJECT> cd .\src\
PS C:\git\PROJECT\src> ls


    Directory: C:\git\PROJECT\src


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        08/04/2024     09:43                Datasets
d-----        11/04/2024     21:18                main
d-----        13/01/2024     13:40                test
d-----        22/11/2023     21:43                __pycache__
-a----        10/04/2024     05:07           7002 Eigenvalues_and_Condition_indicies.xlsx
-a----        10/04/2024     01:53          17647 output.txt
-a----        13/01/2024     13:40              0 __init__.py


PS C:\git\PROJECT\src> python .\main\main.py
```

Figure A.1: Starting GUI from folder

Also here is a link to a video of the code demonstration:

https://www.youtube.com/watch?v=oFB4y_odrpk

Appendix B: **Proof of Concept**

With this background research it is now possible to implement these algorithms as a program from scratch that can be run on a dataset to return real predictions. However, before committing to a full size project I believe it would be beneficial to create a proof of concept (POC) program where the algorithm can be tested on a smaller scale. There are many reasons for doing this, one of them is that a POC program can give insight into the feasibility of the project. If it shows very poor results than it might be better to go back to the planning stage and try to restructure the project and tackle it from another angle. Another reason is that when creating the POC program it is easier to identify any challenges that can arise during implementation. This is valuable as its easier to deal with these challenges when working on a smaller scale, solving these issues now also means that it will be less complicated to deal with them again when they appear in the full implementation. It also creates improved documentation as the issues can be clearly explained and written down, allowing others to have a better chance of re-creating the results of the report. Finally, implementing a POC before the final program is a great way of starting the process of getting to that final stage. It also makes the project aims a lot clearer as you can branch of into many different areas after having achieved the POC such as building a GUI, implementing optimisation processes, developing data visualisation capabilities. This would be much harder if the main algorithm still had not been proven to work.

B.1 Choosing the Dataset

As mentioned before the aim of the proof of concept program is to start small and simple, because of this I have chosen to not utilise the Boston housing dataset for this program. This is due to it having 13 different features and over 500 entries, not only does this mean there is a lot more space for things to go wrong but it also that it is quite unfeasible to show a written example due to the number of variables and entries. Therefore, I have decided to use a dataset that is available through the sklearn library which is called the Linnerud dataset.[38] The dataset was collected from twenty middle-aged men at a fitness club and consists of three features which are the number of reps completed for a given exercise: sit-ups, chins, and jumps. This dataset is multi-output which means in this case that it has 3 different dependent variables: weight, waist, and pulse but for this proof of concept the waist label was chosen to be the dependant variable as it seemed to have a stronger linear relationship compared to the other two features.

B.2 Implementing Ridge Regression

I would consider this to be the most challenging part of the proof of concept as not only is the understanding of the principles behind Ridge Regression needed but also the mathematical knowledge of how to optimise the coefficients and the technical ability to translate it into functioning code. After my research, I decided that the best way, at this stage of the project, to optimise the cost function is to represent the multiple linear regression line in a matrix form.

$$\hat{y} = X\hat{\beta} + \epsilon$$

Where, $\hat{y}$ is a vector containing the predictions made, X is a matrix containing the independent variables, and $\hat{\beta}$ is another vector that contains all the coefficients corresponding to the values in X^T. By using this form it is possible to create an equation, through algebraic manipulation and calculus, where we can solve for $\hat{\beta}$ which will give us all of the coefficients where the SSE_{L2} function is the smallest.[39]

$$\hat{\beta}(a) = (X^T X + aI)^{-1} X^T y$$

This equation can be easily solved using python and the numpy library as we have all the values need. The following snippet of code is how this equations looks in my program, it sits inside of the fit method and is used when fitting the model to the training set.

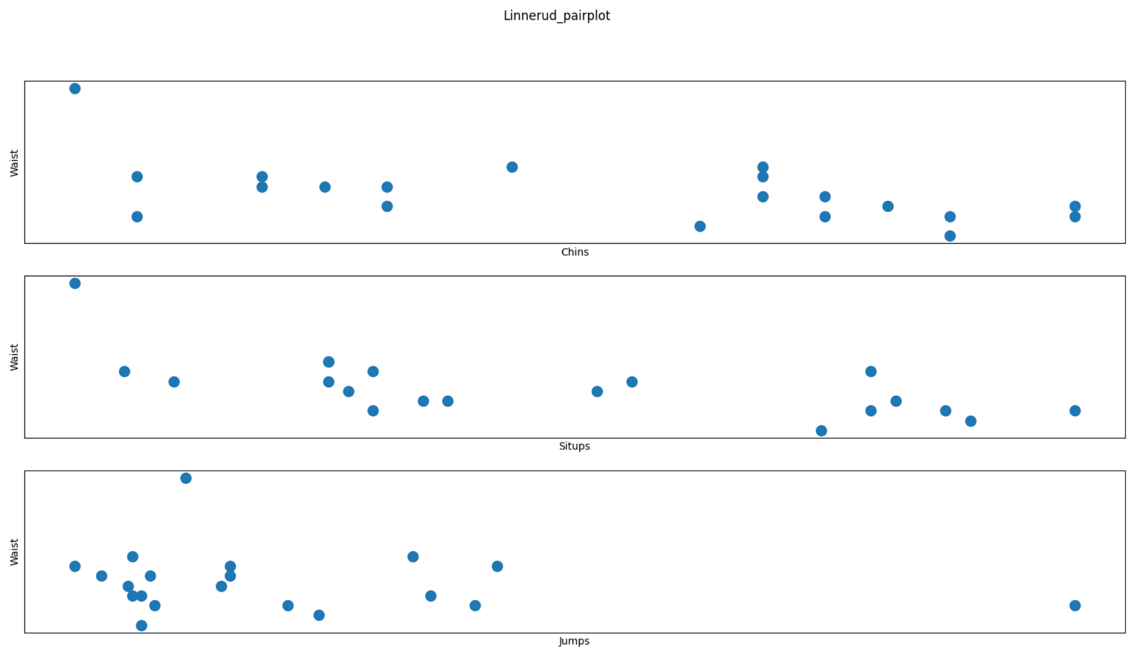
```
def fit(self, X_train, y_train):
    """Fits the regression model to the training data."""
    self.X_train = X_train
    self.y_train = y_train

    # Stores sizes of the training set matrix
    self.m = X_train.shape[0]
    self.n = X_train.shape[1]

    # Identity matrix needed for beta_ridge_hat computation
    I = np.identity(self.n)

    self.beta_ridge_hat = ((inv((self.X_train.T).dot(self.X_train)
    + self.penalty * I)).dot(self.X_train.T)).dot(y_train)
```

Now that the optimisation equation has been completed all that is needed is to create a predict method that takes a set of data points, runs the multiple regression formula with the optimised coefficients, and returns its own predictions. This part was relatively easy to do and I made sure to keep it in a general format so that the same code could be reused to work with a dataset that has a different number of features.

B.3 Data Visualisation

Even though the main focus of this report is the comparison of machine learning algorithms, data visualisation is essential in order to make smart decision regarding what data we are using and if an algorithm is even appropriate or not for a given situation. Because of this it imperative that this analysis is done first, before committing to an algorithm or dataset in order to improve the chances of success.[40]

The first thing I did was to plot out each of the independent variables against the dependent variable to have a better understanding of how the data was related to each other.

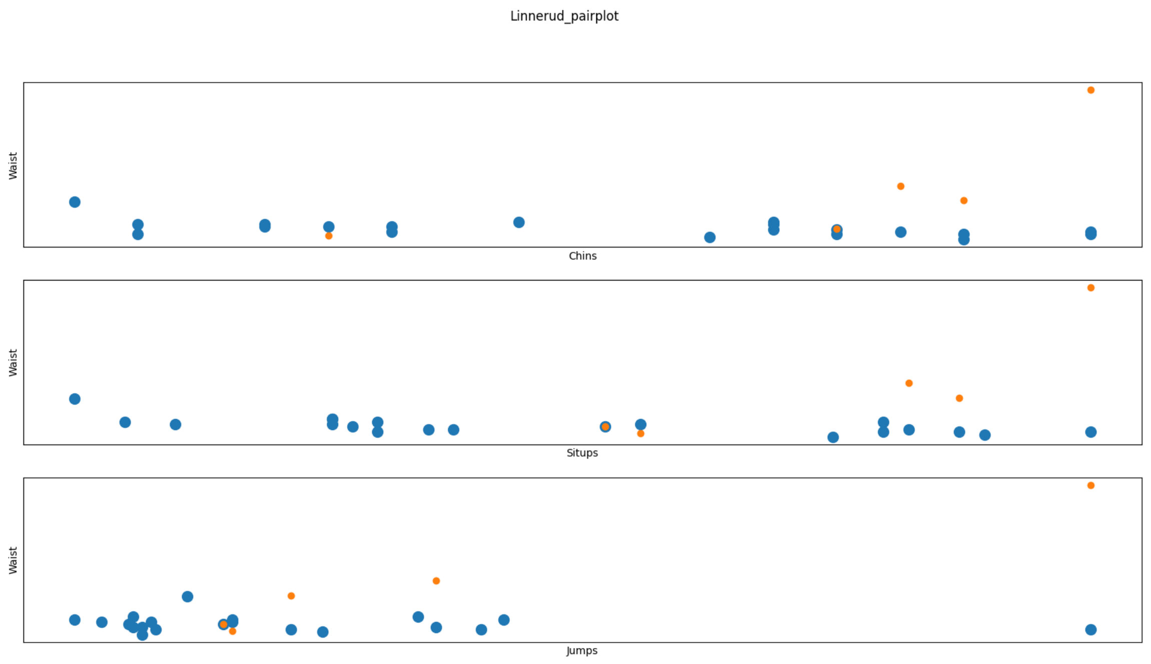


In these 3 graphs we can see that there does seem to be a linear relationship between the features and the dependent variable. However we do also see some outliers in the data which can have a negative affect on the outcome of the model. These can be reduced after applying some preprocessing methods to the data before applying our machine learning algorithms. However, for now this is sufficient information for me too continue the approach of using the Ridge Regression algorithm on this dataset for the proof of concept.

B.4 Running Program

After having written the classifier classes, importing the data, separating it, and visualising it we can finally apply the algorithm to it and see if the results appear accurate. First thing is that the data must be separated into both our training and test sets. The training set will be used to actually train the ridge regression model, i.e., contains the data points that we will fit our line to. The test set is what will be used to test our model and see how accurate its predictions are. It is very important that when training the algorithm no information from the test set is used and vice versa, this is called data snooping and can negatively impact the model.

After the separation has been made the Ridge Regression classifier is instantiated, it is then fitted to the training data and ready to make predictions. The model is then run on the test set where it makes a prediction for each sample in it. I then printed out both the predictions given by the model and the real labels of the test set. Finally, I once again plotted the features against the dependent variable but this time I plotted on top of theses graphs the predictions made by the model to get a better understanding of how the model was performing.



After taking a look at the plotted results I can say that I am reasonably satisfied with the results, especially considering that this was the proof of concept. Almost all of the predictions were reasonably close to their true values, except for the data point that is farthest away and had very poor accuracy. However, I believe that this may have happened due to not having done enough data preprocessing before applying the model. We can see from the graphs that the points tend to grow higher the more towards the right we move. With more preprocessing the data points might be closer together and more evenly spaced which could fix this issue. Also trying different values for the penalty term a might have a positive effect on the predictions as it would flatten the line out more, as it would help with the over prediction closer to the right end of the graphs.

Bibliography

[1] L. R. Weinstock, “Introduction to u.s. economy: Housing market,” *Congressional Research Service*, Jan. 2023.

[2] P. Herman, “The importance of price prediction,” *Future Processing*, Mar. 2023.

[3] P. Schneider and F. Xhafa, “Chapter 3 - anomaly detection: Concepts and methods,” in *Anomaly Detection and Complex Event Processing over IoT Data Streams* (P. Schneider and F. Xhafa, eds.), pp. 49–66, Academic Press, 2022.

[4] S. Olumide, “Root mean square error (rmse): What you need to know,” *Arize*, Aug 2023.

[5] V. Roman, “Root mean square error (rmse): What you need to know,” *Towards Data Science*, Jan. 2019.

[6] C. Smith, “Housing affordability in england and wales: 2021,” *Census 2021*, Mar. 2022.

[7] “Housing affordability in england and wales: 2021,” *Bank of England*, Mar. 2020.

[8] M. Chatterjeeh, “Data science vs machine learning and artificial intelligence: The difference explained (2024),” *Great Learning*, Nov. 2023.

[9] I. Ake, “Combining machine learning models to predict house prices,” *Solent University*, Sep. 2022.

[10] V. Kanade, “What is machine learning? definition, types, applications, and trends for 2022,” *Spice Works*, Aug. 2022.

[11] V. Vovk, “Chapter 2: Introduction to machine learning and nearest neighbours.” `https://moodle.royalholloway.ac.uk/pluginfile.php/188746/mod_resource/content/27/02_1.pdf`, Sep. 2023.

[12] B. Beers, “What is regression? definition, calculation, and example,” *investopedia*, Mar. 2023.

[13] D. Maulud and A. M. Abdulazeez, “A review on linear regression comprehensive in machine learning,” *Journal of Applied Science and Technology Trends*, vol. 1, pp. 140–147, Dec. 2020.

[14] S. Gupta, “Boston house price prediction based using support vector regressor,” *enjoy algorithms*.

[15] S. Rong and Z. Bao-Wen, “The research of regression model in machine learning field,” in *MATEC Web of Conferences*, vol. 176, p. 01033, EDP Sciences, 2018.

[16] S. Glen, “Error term: Definition and examples,” *Statistics How To*, Nov. 2020.

[17] K. Krzyk, “Cost function of linear regression: Deep learning for beginners,” *Built In*, Jul. 2022.

[18] S.-J. Kim, S.-J. Bae, and M.-W. Jang, “Linear regression machine learning algorithms for estimating reference evapotranspiration using limited climate data,” *Sustainability*, vol. 14, no. 18, p. 11674, 2022.

[19] M. Tranmer and M. Elliot, “Multiple linear regression,” *The Cathie Marsh Centre for Census and Survey Research (CCSR)*, vol. 5, no. 5, pp. 10–11, 2008.

[20] S. Taylor, "Multiple linear regression," *Data Science*, Apr. 2020.

[21] G. C. McDonald, "Ridge regression," *WIREs Computational Statistics*, vol. 1, no. 1, pp. 93–100, 2009.

[22] A. Alin, "Multicollinearity," *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 3, pp. 370–374, 2010.

[23] J. Frost, "Multicollinearity in regression analysis: Problems, detection, and solutions," *Statistics By Jim*, 2017.

[24] S. Wu, "Multicollinearity in regression," *towardsdatascience*, May. 2020.

[25] J. Nagidi, "How to handle overfitting in deep learning models," *Dataaspirant*, Aug. 2020.

[26] P. Gupta, "Regularization in machine learning," *Towards Data Science*, Nov. 2017.

[27] C. Maklin, "Machine learning algorithms part 11: Ridge regression, lasso regression and elastic-net regression," *Medium*, Dec. 2018.

[28] K. Kargin, "Ridge regression fundamentals and modeling in python," *Medium*, Apr. 2021.

[29] L. Demanet, "Numerical analysis lecture notes," *MIT OpenCourseWare*, 2012.

[30] D. Harrison and D. L. Rubinfeld, "Hedonic housing prices and the demand for clean air," *Journal of Environmental Economics and Management*, vol. 5, no. 1, pp. 81–102, 1978.

[31] P. Cortez, "Student Performance." UCI Machine Learning Repository, 2014. DOI: https://doi.org/10.24432/C5TG7T.

[32] M. Redmond, "Communities and Crime Unnormalized." UCI Machine Learning Repository, 2011. DOI: https://doi.org/10.24432/C5PC8X.

[33] P. Cortez and A. M. G. Silva, "Using data mining to predict secondary school student performance," 2008.

[34] "Normalization," *Google Developers*, Jul. 2022.

[35] V. Vovk, "Chapter 6: Data preprocessing, parameter selection, and inductive conformal prediction." `https://moodle.royalholloway.ac.uk/pluginfile.php/195264/mod_resource/content/32/06_1.pdf`, Nov. 2023.

[36] "Regularization," *cscheid*, Nov. 2020.

[37] D. Jain, "R-squared in regression analysis in machine learning," *Geeks for Geeks*, May. 2023.

[38] A. Bajwa, "What is datasets load$_linnerud()insklearn?,'' educative.$

[39] M. Levine, "Statistics 512: Applied linear models," *Purdue University*, 2017.

[40] E. B. Kate Brush, "Data visualization," *Tech Target*, Dec. 2022.