# CHAPTER 3: Feature Scaling and Feature Selection

## Introduction

This chapter is intended to provide a detailed explanation of data transforms, feature scaling, and feature selection methods.

In my machine learning journey, more often than not, I have found that feature pre-processing is a more effective technique In improving my evaluation metric than any other step, like choosing a model algorithm, hyperparameter tuning, etc.

Feature pre-processing is one of the most crucial steps in building a machine learning model. Too few features and your model won't have much to learn from. Too many features and we might be feeding unnecessary information to the model. not only this, but the value is in each of the features need to be considered as well.

We know that there are some set rules for dealing with categorical data, as in, encoding them in different ways. However, a large chunk of the process involves dealing with continuous variables. There are various methods of dealing with continuous variables. Some of them include converting them to a normal distribution or converting them to categorical variables, etc.

There is a couple of go-to techniques I always use regardless of the model I'm using, whether it is a classification task or regression task, or even an unsupervised learning model. these techniques are:

* Feature transformation
* Feature scaling

## Why do we need feature transformation and scaling?

Oftentimes, we have data sets in which different columns have different units - like one column can be in kilograms, while another column can be in centimetres. Furthermore, we can have columns like income which can range from 20,000 to 100,000, and even more, while an age column which can range from 0 to 100 (at the most). Thus, income is about 1000 times larger than age.

But how can we be sure that the modern treats both these variables equally? When we feed these features to the modern as is, there is every chance that the income will influence the result more due to its larger value. but this doesn't necessarily mean it is more important as a predictor. So, to give importance to both age and income, we need feature scaling.

In most examples of machine learning models, you would have observed either the Standard Scaler or MinMax Scaler. However, the powerful sklearn library offers many other future transformations scaling techniques as well, which we can leverage depending on the data we are dealing with. So, what are you waiting for?

Let us explore them one by one with Python code.

We will work with a simple data frame:

A picture containing text

Description automatically generated

Before directly applying any feature transformation or scaling techniques, we need to remember the categorical column: Department and first deal with it. This is because we cannot scale non-numeric values.

For that, we first create a copy of our data frame and store the numerical feature names in a list, and their values as well:

A picture containing chart

Description automatically generated

We will execute this snippet before using a new scaler every time.

### MinMax Scaler

The MinMax scaler is one of the simplest scalers to understand. It just scales all the data between 0 and 1. the formula for calculating the scaled value is –



Thus, a point to note is that it does so for all features are separate. Though (0, 1) Is the default range, we can define our range of max and min values as well. How to implement the MinMax scaler?

1. We will first need to import it



1. Apply it on only the values of the features:



How do the scaled values look like?

Graphical user interface, application

Description automatically generated

You can see how the values were scaled. The minimum value among the columns became 0, and the maximum value was changed to 1, With other values in between. However, suppose we don't want the income or age to have values like 0. let us take the range to be (5, 10)

Text

Description automatically generated

Graphical user interface, application

Description automatically generated

Amazing, right? The MinMax scaler lets you set the range in which you want the variables to be.

### Standard Scaler

Just like the MinMax Scaler, the standard scaler is another popular scaler that is very easy to understand and implement.

For each feature, the Standard Scaler scales the values such that the mean is 0 and the standard deviation is 1 (or the variance).

A picture containing icon

Description automatically generated

However, Standard Scaler assumes that the distribution of the variable is normal. Thus, in case, the variable is not normally distributed, we

1. Either choose a different scaler
2. Of first, convert the variable to a normal distribution and then apply this scaler

Implementing the standard scaler is much simpler to implement a min-max scaler. just like before, we will first import StandardScaler then use it to transform our variable.

Text

Description automatically generated

Graphical user interface, application

Description automatically generated

Let us check the mean and standard deviation of both the columns by performing a describe() function on df\_scaled



Table

Description automatically generated

You will notice that the values are not exactly, but very close to zero (same with standard deviation). this occurs due to the numerical precision of floating-point numbers in Python.

### MaxAbsScaler

In simplest terms, The MaxAbs scaler takes the absolute maximum value of each column and divides each value in the column by the maximum value.

Thus, it first takes the absolute value of each value in the column and then takes the maximum value out of those. this operation scales the data between the range [-1, 1]. To see how it works, we will add another column called “Balance” which contains negative values.

Text

Description automatically generated with medium confidence

Graphical user interface, application

Description automatically generated

We can confirm that the MaxAbs Scaler works as expected by printing the maximum values of each column before we scaled it:

Graphical user interface, application, Word

Description automatically generated

Thus, we can see that

* each value in the income column is divided by 1200
* each value in the age column is divided by 51
* each value in the balance column is divided by 2000

### Robust Scaler

If you have noticed in the scaler we used so far, each of them was using values like the mean, maximum and minimum values of the columns. all these values are sensitive to outliers. If there are too many outliers in the data, they will influence the mean and the Max value or the min value. Thus, even if we scale this data using the above methods, we cannot guarantee balanced data with a normal distribution

The robust scaler, as the name suggests is not sensitive to the outlier. this scaler –

1. Removes the median from the data
2. Scales the data by the interquartile range (IQR)

Are you familiar with the Inter-Quartile Range? it is nothing but the difference between the first quartile and the third quartile of the variable. the interquartile range can be defined as-

A picture containing logo

Description automatically generated

Thus, the formula would be:

Logo

Description automatically generated

This is the default range, though we can define our own range if you want to. now let us see how we can implement the robust scaler in Python.

Text

Description automatically generated with medium confidence

Graphical user interface, application

Description automatically generated

### Log transform

The Log Transform is one of the most powerful and popular transformation techniques out there. it is primarily used to convert a skewed distribution to a normal distribution/less skewed distribution. In this transform, we take the log of the values in a column and use these values as the column instead.

Why does it work? it is because the log function is equipped to deal with large numbers. here is an example-

Text

Description automatically generated

Thus, in our example, while plotting the histogram of income, it ranges from 0 to 1,20,000:

Chart

Description automatically generated

Let us see what happens when we apply log on this column:

A picture containing text

Description automatically generated

Table

Description automatically generated

Wow! While our income column had extreme values ranging from 1800 to 1,20,000 – the low values are now raining from 7.5 to 11.7! Thus, the log operation had a dual role:

* Reducing the impact of too-low values
* Reducing the impact of too-high values

A small caveat though - if our data has negative values our values ranging from 0 to 1, we cannot apply log transformation directly - since the log of negative values and the numbers between 0 and 1 are undefined, we would get error or NaN values in our data. In such cases, what are values to make them all greater than 1? Then we can apply the log transform.

Let us plot a histogram of the above, using 5 bins:

Chart, bar chart

Description automatically generated

### Power transformer scaler

I often use this feature transformation technique when I am building a linear model. to be more specific, I use it when I'm dealing with heteroskedasticity. Like some other scalers we studied above, the power transformer also changes the distribution of the variable, as in, it makes it more Gaussian(normal). We are familiar with similar power transforms such as square root and cube root transforms, and log transforms.

However, to use them, we need to first study the original distribution, and then make a choice. The Power Transform automates this decision-making by introducing a parameter called lambda. It decides on a generalized power transform by finding the best value of Lambda using either the:

* Box-Cox transform
* The Yeo-Johnson transforms

Why I will not get into too much detail about how each of the above transform’s work, it is helpful to know that Box-Cox works with only positive values, while Yeo-Johnson works with both positive and negative values.

In our case, we will use the Box-Cox transform since all overvalue are positive.

Graphical user interface, text

Description automatically generated

Table

Description automatically generated

### Unit Vector Scaler/Normalizer

Normalization is the process of scaling individual samples to have a unit norm. The most interesting part is that, unlike the other scalers which work on the individual column values, the Normalizer works on the rows! Each row of the data frame with at least one non-zero component is rescaled independently of other samples so that it is the norm (l1, l2, or inf) equals one.

Just like MinMax Scaler, the normalizer also converts the values between 0 and 1, and between -1 to 1 when there are negative values in our data.

However, there is a difference in the way it does so.

* If we are using the L1 norm, the values in each column are converted so that the sum of their absolute values along the row = 1.
* If you are using the L2 norm, the values in each column are first squared and added so that the sum of their absolute values along the row is = 1

Text

Description automatically generated

Table

Description automatically generated

Thus, if you check the first row,



Similarly, you can check for all rows, and try out the above with norm = ‘l1’ as well.

## The Big Question - Normalize or Standardize?

Normalization vs. Standardization is an eternal question among machine learning newcomers. let me elaborate on the answer in this section.

* Normalization is good to use when you know that the distribution of your data does not follow a gaussian distribution. This can be useful in algorithms that do not assume any distribution of the data like K-Nearest Neighbors and Neural Networks.
* Standardization, on the other hand, can be helpful in cases where the data follows a Gaussian distribution. However, this does not have to be necessarily true. Also, unlike normalization, standardization does not have a bounding range. So, even if we have outliers in our data they will not be affected by the standardization.

However, at the end of the day, the choice of using Normalization or Standardization would depend upon your problem and the machine learning algorithm you are using. There is no hard and fast rule to tell you whether to normalize standardize your data. You can always start by fitting your model to raw, normalize, and standardize data and compare the performance for best results.

It is a good practice to fit the scaler on the training data and then uses it to transform the testing data. this would avoid any data leakage during the model testing process. Also, the scaling of target values is generally not required.

## Implementing feature scaling in Python

Now comes the fun part - putting what we have learned into practice. I will be applying features scaling to a few machine learning algorithms on the Big Mart dataset.

I will skip the pre-processing steps since they are out of the scope of this chapter. but you can find them explained in chapter 2.

So, let's first split our data into training and testing sets:

Text

Description automatically generated

Company name

Description automatically generated with low confidence

Graphical user interface, text, application

Description automatically generated

Table

Description automatically generated

We can see that there is a huge difference in the range of values present in our numerical features: Item\_Weight , Item\_Visibility, Item\_MRP, and Outlet\_Establishment.

Let’s try and fix that using feature scaling!

For illustration purposes, I am taking only Item\_weight and Item\_visibility variables.



Table

Description automatically generated

Let’s split the data into train and test sets



I have split the data into train and test in the ratio of 80/20.



We have fitted our MinMax Scaler on the training data first.

Text

Description automatically generated

We are transforming the train and test data separately.

Text, letter

Description automatically generated

Text

Description automatically generated

Text, letter

Description automatically generated

Next, Let’s try to standardize our data.

**Standardization using sklearn**

To standardize your data, you need to import the StandardScalar from the sklearn library and apply it to our dataset. Here’s how you can do it:

Text

Description automatically generated

Let’s create a copy of our dataset so that we can preserve our original data.

Text, letter

Description automatically generated

First, we will create a list containing the numerical features

Chart

Description automatically generated with low confidence

Text

Description automatically generated with medium confidence

You would have noticed that I only applied standardization to my numerical columns and not the other One-Hot Encoded features. Standardizing the One-Hot encoded features would mean assigning a distribution to categorical features. You don’t want to do that!

But why did I not do the same while normalizing the data? Because One-Hot encoded features are already in the range between 0 to 1. So, normalization would not affect their value.

Right, let’s have a look at how standardization has transformed our data:

Table

Description automatically generated

The numerical features are now centered on the mean with a unit standard deviation.

**Comparing unscaled, normalized, and standardized data**

It is always great to visualize your data to understand the distribution present. We can see the comparison between our unscaled and scaled data using boxplots.

Chart, box and whisker chart

Description automatically generated

Chart, box and whisker chart

Description automatically generated

Chart, box and whisker chart

Description automatically generated

You can notice scaling the feature brings everything into perspective. The features are now more comparable and will have a similar effect on the learning models.

## Feature Selection

**Introduction**

Suppose in a particular data set if we have many features, this may increase the threshold value which in turn decreases the accuracy of the model. whenever we give those data to train our model, the model gets confused because it is learning too much data.

To solve this situation what we do is that we do not select all the features from a particular dataset. Instead, we apply various techniques of the feature section.

**Importance of Feature Selection in Machine Learning**

Question learning works on a simple rule - if you put garbage in, you will only get garbage to come out. By garbage here, I mean noise in data.

This becomes even more important when the number of features is very large. you need not use every future at your disposal for creating an algorithm. You can assist your algorithm by feeding in only those features that are really important. I have myself witnessed feature subsets giving better results than a complete set of features for the same algorithm.

The top reasons to use features selection are:

* It enables the machine learning algorithms to train faster.
* it reduces the complexity of a model and makes it easier to interpret.
* it improves the accuracy of a model if the right subset is chosen.
* it reduces overfitting.

Next, we will discuss various methodologies and techniques that you can use to subset your features space and help your models perform better and efficiently. So, let's get started.

Features selection methods are broadly classified into three types.

* Filter Methods
* Wrapper Methods
* Embedded Methods

Let's discuss one by one in detail

**Filter Methods**



Filter methods are generally used as a pre-processing step. the selection of features is independent of any machine learning algorithms. Instead, features are selected based on their scores in various statistical tests for their correlation with the outcome variable. The correlation is a subjective term here.

* **Pearson's correlation:** It is used as a measure for qualifying linear dependence between two continuous variables X and Y. its value varies from -1 to +1. Pearson's correlation is given as

Text, letter

Description automatically generated

* **Information Gain – Mutual Information:** Mutual information between two variables is a non-negative value, which measures the dependency between the variables. It is equal to zero if and only if two random variables are independent, and higher values mean higher dependency. In Short, A quantity called mutual information measures the amount of information one can obtain from one random variable given another.
* **Chi-Squared Test:** This is a form of non-parametric test (a test wherein median is an important parameter) in which solution is done with the help of hypothesis testing and P-value. The feature selection is only suited to categorical features or features having discrete data contained within it. The continuous features are not considered and therefore should not be used while performing this test. the formula for this type of test is given as:

**A picture containing text, clipart

Description automatically generated**

If the p-value is less than 0.05 then we reject the null hypothesis and go with the alternate hypothesis in this type of test. Null hypothesis: No association between two variables, Alternate hypothesis: There is an association between two variables.

## Hands-on

Let's do the practical implementation of filter-based feature selection techniques:

**Step1: Importing the required libraries**

A picture containing timeline

Description automatically generated

**Step2: Loading the dataset**

A picture containing graphical user interface

Description automatically generated

Let’s separate the features and target variables



**Step3:** **Split the data into train and test sets**

Text, letter

Description automatically generated

**Step4: Check for correlation**

Now let’s check the correlation between the independent features and we will drop them which are highly correlated to eliminate the multicollinearity issue.



A screenshot of a computer

Description automatically generated with medium confidence

We can visualize the correlation using a heatmap so that we get a clear picture.

A picture containing text

Description automatically generated

A screenshot of a computer

Description automatically generated with low confidence

With the following function we can select highly correlated features it will remove the first feature that is correlated with any another feature.

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Let’s check the features that are highly correlated.

Graphical user interface

Description automatically generated with medium confidence

**Step5: Dropping Correlated Features**

We are dropping does features that are highly correlated.

Text

Description automatically generated

Table

Description automatically generated

**Feature Selection – Information gain**

**Step1: Importing the dataset**

**Data:** [**https://gist.githubusercontent.com/tijptjik/9408623/raw/b237fa5848349a14a14e5d4107dc7897c21951f5/wine.csv**](https://gist.githubusercontent.com/tijptjik/9408623/raw/b237fa5848349a14a14e5d4107dc7897c21951f5/wine.csv)



Graphical user interface

Description automatically generated with low confidence

**Step2: Splitting the dataset**

Text

Description automatically generated

**Step3: Calculating the mutual information**

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Table

Description automatically generated

Let's plot the ordered mutual\_info values per feature

Chart, bar chart

Description automatically generated

**Step3: Selecting K Best features**

We can select the best features using a function called “SelectKBest”



Text

Description automatically generated

**Feature Selection – Chi-squared test**

**Step1: Importing the dataset**

A picture containing text

Description automatically generated

Graphical user interface

Description automatically generated with medium confidence

Let’s take a few categorical variables and perform a chi-squared test on them.

Graphical user interface, table

Description automatically generated with medium confidence

Let’s perform label encoding on those columns

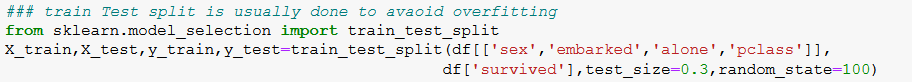
Graphical user interface, text

Description automatically generated with medium confidence

A picture containing table

Description automatically generated

**Step2: Splitting data into train and test**



Performing chi-squared test using chi2 function from sklearn library. The function returns two values F-score and P-value.

Graphical user interface, text, application, email

Description automatically generated

Let’s subset the p-values from the results and print the results.

Text

Description automatically generated

Text

Description automatically generated

The p-values for all the variables are less than 0.05 which means the independent variables are associated with the target variable.

Sorting the results in decreasing order to find out the most important feature among all the features.

Text

Description automatically generated

**Observation**

The sex column is the most important column when compared to the output feature Survived.

**Wrapper Methods**

Diagram

Description automatically generated

In wrapper methods, we try to use a subset of features and train a model using them. Based on the inferences that we draw from the previous model; we decide to add or remove features from your subset. the problem is essentially reducing a search problem. These methods are usually computationally very expensive.

Some common examples of wrapper methods are forward feature selection, backward feature elimination, recursive feature elimination, etc.

* **Forward selection:** forward selection is an alternative method that we start with having no feature in the model. In each iteration, we keep adding the feature which best improves our model till an addition of a new variable does not improve the performance of the model.
* **Backward elimination:** In backward elimination, we start with all the features and remove the least significant feature at each iteration which improves the performance of the model. we repeat this until no improvement is observed on the removal of features.
* **Recursive feature elimination:** It is a greedy optimization algorithm that aims to find the best performing feature subset. It repeatedly creates models and keeps aside the best or the worst performing feature at each iteration. It constructs the next model with the left features until all the features are exhausted. It then ranks the features based on the order of their elimination.

One of the best ways for implementing feature selection with wrapper methods is to use the Boruta package that finds the importance of a feature by creating shadow features.

It works in the following steps:

1. Firstly, it adds randomness to the given data set by creating shuffled copies of all features (which are called shadow features).
2. Then, it rains a random forest classifier on the extended data set and applies a feature importance measure (the default is Mean Decrease Accuracy) to evaluate the importance of each feature where higher means more important.
3. At every iteration, it checks whether a real feature has higher importance than the best of its shadow features ( i.e., whether the feature has a higher Z-score than the maximum Z-score of its shadow features) and constantly removes features that are deemed highly unimportant.
4. Finally, the algorithm stops either when all features get confirmed or rejected or it reaches a specific limit of random forest runs.

### Implementation of Wrapper methods in python

Too much theory so far. now let us discuss wrapper methods with an example of the bottom house prices state asset available in Sklearn. the dataset contains 506 observations of 14 different features. the real estate can be imported using the load\_boston() function available in Sklearn.datasets module.

**Step1: Importing the dataset**

Text

Description automatically generated

Let's convert this raw data into a data frame including target variable and actual data along with the feature names.

Text

Description automatically generated

Text, table

Description automatically generated

Here, the target variable is price. We will be fitting a regression model to predict Price by selecting optimal features through wrapper methods.

**Step2: Forward Selection**

Mlxtend library contains built-in implementation for most of the wrapper methods-based feature selection techniques. SequentialFeatureSelector() function comes with various combinations of feature selection techniques.

Text

Description automatically generated

Text

Description automatically generated

SequentialFeatureSelector() function accepts the following major arguments:

* LinearRegression() is an estimator for the entire process. Similarly, it can be any classification-based algorithm.
* K\_features indicates the number of features to be selected. It can be any random value, but the optimal value can be found by analyzing and visualizing the scores for different features.
* forward and floating argument are different flavors of wrapper methods, here, forward = True and floating = False are for the Forwarding Selection technique.
* The scoring argument specifies the evaluation criteria to be used. for regression problems, there is only an r2 score in default implementation. similarly, for classification, it can be accuracy, precision, recall, f1-score, etc.
* cv argument is for k-fold cross-validation.

Now let's fit the above-defined feature selector on the Boston house price prediction dataset.

Graphical user interface, text, application, email

Description automatically generated

**Step3: Backward elimination**

The same SequentialFeatureSelector() function can be used to perform backward elimination by disabling the forward argument.

Graphical user interface, text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**Note:**

Here we are directly using the optimal value of k\_featues argument in both forward selection and backward elimination. in order to find out the optimal number of significant features, we can use the hit and trial method for different values of k\_features and make the final decision by plotting it against the model performance.

Text, letter

Description automatically generated

The same visualization can be achieved through plot\_sequential\_feature\_selection() function available in mlxtend.plotting module.

Text

Description automatically generated

Chart, line chart

Description automatically generated

Here, on the y-axis, the performance label indicates the R-Squared values for the different numbers of features.

**Step4: Recursive Feature Elimination**

Below, you will see an example of RFE using the above Random Forest Regressor model:

Graphical user interface, text

Description automatically generated

Train and Test Split

Text

Description automatically generated with medium confidence

Let’s fit the model to the training set using RFE



After fitting the estimator, it has a “.support\_” attribute that gives a Boolean mask with False values for discarded features. We can use it to subset our data:

Columns before subsetting

Graphical user interface, text, application, Word

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Using the above Boolean mask, we can select the features.



Table

Description automatically generated

Let’s check the score of the model.

Graphical user interface, text

Description automatically generated

Even after dropping 3 features, we got a very good score of 86.75% which is very impressive!

**Embedded Methods**

Diagram

Description automatically generated

Embedded methods combine the qualities of filter and wrapper methods. It’s implemented by algorithms that have their own built-in feature selection methods.

Some of the most popular examples of these methods are LASSO and RIDGE regression which have inbuilt penalization functions to reduce overfitting.

* Lasso regression performs L1 regularization which adds penalty equivalent to the absolute value of the magnitude of coefficients.
* Ridge regression performs L2 regularization which adds penalty equivalent to the square of the magnitude of coefficients.

### Implementing Lasso and Ridge

Importing necessary libraries

Text, letter

Description automatically generated

**Step1: Importing dataset**

A picture containing chart

Description automatically generated

The dataset contains the scores of different exams and tries to predict the chance of getting admission to the university.

Table

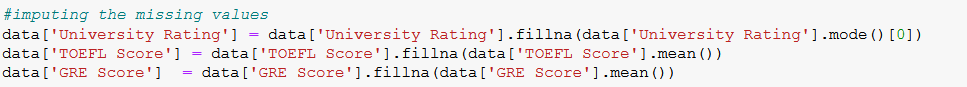
Description automatically generated

Text, table

Description automatically generated

There are a few missing values in the dataset. Let’s impute them

**Step2: Imputing the missing values**



Table

Description automatically generated

Now the data looks good and there are no missing values. Also, the first column is just serial numbers, so we don’t need that column. Let’s drop it from data and make it cleaner.

Table

Description automatically generated

Separating the features and target variables.

Chart

Description automatically generated with low confidence

Let’s standardize the data to make them independent of units

Text

Description automatically generated with medium confidence

Splitting the train and test sets



### Lasso Regularization



Graphical user interface, text, application

Description automatically generated

Fitting the lasso model

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**Step3: Lasso**

Text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

**Step4: Ridge**

Fitting the Ridge model

Graphical user interface, text

Description automatically generated

Graphical user interface, text, application, Word

Description automatically generated

Lasso and Ridge algorithms have penalized the model coefficients and helped model from overfitting

## Principal Component Analysis:

The principal component analysis is an unsupervised machine learning algorithm used for feature selection using dimensionality reduction techniques. As the name suggests, it finds out the principal components from the data. PCA transforms and fits the data from a higher-dimensional space to a new, lower-dimensional subspace This results in an entirely new coordinate system of the points where the first axis corresponds to the first principal component that explains the most variance in the data.

**What are the principal components?** Principal components are the derived features that explain the maximum variance in the data. The first principal component explains the most variance, the 2nd a bit less, and so on. Each of the new dimensions found using PCA is a linear combination of the old features.

Let's take the following example where the data is distributed like the diagram on the left:

Chart, scatter chart

Description automatically generated

In the diagram above, we are considering 3 orthogonal (*C3 is in the third dimension*) axes to show the distribution of data. If you notice the diagram on the right, the first two axes **C1** and **C2** successfully explain the maximum variation in the data whereas the axes **C3** only consist of a fewer number of points. Hence, while considering the principal components C1 and C2 will be our choices.

**Python Implementation**

**Step1: Importing required libraries**

A picture containing graphical user interface

Description automatically generated

**Step2: Importing the dataset**

Graphical user interface

Description automatically generated with low confidence

Text

Description automatically generated

Let’s check the null values in the dataset.

**Step3: Dropping NA values**

A picture containing table

Description automatically generated



Index and Class variables are dropped. Now we can standardize this data as all the data is on a different scale.

Text

Description automatically generated with low confidence

Table

Description automatically generated

**Step4: Fitting PCA model**

Let’s fit the PCA model and check the variance explained by the number of variables.

Text

Description automatically generated

Chart

Description automatically generated

From the diagram above, it can be seen that 4 principal components explain almost 90% of the variance in data and 5 principal components explain around 95% of the variance in data.

So, instead of giving all the columns as input, we’d only feed these 4 principal components of the data to the machine learning algorithm and we’d obtain a similar result.

Text

Description automatically generated

Table

Description automatically generated

Here, we see that earlier we had 9 columns in the data earlier. Now with the help of Screen plot and PCA, we have reduced the number of features to be used for model building to 4. This is the advantage of PCA. *It drastically reduces the number of features, thereby considerably reducing the* training *time for the model.*

**SUMMARY**

In this chapter you discovered how you can prepare your data for machine learning in python, you learned why do we need to scale the data and different techniques to scale the data. Then, you learned 3 different feature selection techniques and understood how to implement them practically. Finally, we have seen what PCA is and how to implement it to reduce the dimensionality of the data.

## Assessment

**Choose the appropriate option**

1. What do you mean by feature engineering?
   1. Creating a new feature
   2. Transforming a feature
   3. Encoding a feature
   4. All the above
2. We can handle missing values using the following methods?
   1. Ignoring such rows or dropping such records.
   2. Fill values with mean, mode, and median
   3. Fill with a constant value
   4. Fill values manually
   5. All the above
3. Which of the below technique does not belong to Wrapper Methods
   1. Forward Selection
   2. Ridge
   3. Backward Elimination
   4. RFE
4. The chi-Squared test is used to compare?
   1. Categorical features
   2. Continuous features
   3. Both Continuous and Categorical features
   4. None of the above
5. Lasso regression is also called as.
   1. L2
   2. L1
   3. Elastic Net
   4. None of the above

**Fill in the spaces with appropriate answers**

1. MinMax normalization performs a liner transformation on the original data and converts it into a given \_\_\_\_\_\_\_\_\_\_\_\_\_ and \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ range.
2. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ is good to use when you know that the distribution of your data does not follow a Gaussian distribution.
3. To perform standardization, data needs to follow \_\_\_\_\_\_\_\_\_\_\_\_\_ distribution.
4. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ is a greedy optimization algorithm which aims to find the best performing feature subset.
5. L2 regularization adds penalty equivalent to \_\_\_\_\_\_\_\_\_\_\_\_ of the magnitude of coefficients.

**True or False**

1. Standardization transforms the data into the range of 0 and 1
   1. True
   2. False
2. Feature selection is used to filter a subset of input variables on which the attention should focus.
   1. True
   2. False
3. Feature engineering involves the transformation of features so that we can extract features to improve the process of feature selection
   1. Ture
   2. False
4. Starting from the full set and evaluating the metric for the set without each feature. At each stage, the set is shrunk by the feature is called forward selection.
   1. True
   2. False
5. PCA is a feature scaling technique.
   1. True
   2. False

## Programming Assessment

By using the dataset from below link. Perform the following tasks.

1. Import the dataset
2. Separate the target and independent variables
3. Split the dataset into train and test
4. Scale the data using MinMaxScaling
5. Select the best features using SelectKBest, Correlation and Wrapper Methods.

## Assessment Solutions

**Choose the appropriate option**

1. D
2. E
3. B
4. A
5. B

**Fill in the spaces with appropriate answers**

1. Minimum and Maximum
2. Normalization
3. Gaussian
4. Recursive Feature Elimination
5. Square

**True or False**

1. False
2. True
3. True
4. False
5. False