Contents

[1. Intro and Week overview 2](#_Toc100714723)

[2. End to End ML pipeline/ Workflow 3](#_Toc100714724)

[3. Introduction to Machine Learning Algorithms 5](#_Toc100714725)

[4. Train/Test Split 8](#_Toc100714726)

[5. Performance Metrics 10](#_Toc100714727)

[6. Feature Engineering 16](#_Toc100714728)

[7. MCQ 22](#_Toc100714729)

[Homework 24](#_Toc100714730)

# Intro and Week overview

**What is Machine Learning ?**

Machine learning is a field of science associated with algorithms which allow software applications to continuously learn from data and improve in predicting specific outcomes, without being explicitly programmed to do so. Hence, the main difference between an ML application and a software application is that the former one’s self-learning in a sense which a normal software is not. Rather, a normal software is a simple set of logical instructions involved to follow based on specific conditions, and there is no self-learning involved.

People often are confused about what exactly is ML and Artificial intelligence (AI), and whether these are synonyms. Answer – No they are not synonymous. AI is a more general and vast field of computer science associated with building machines or applications for assisting or replacing manual tasks. Machine learning is a subset of AI which only deals with specific set of mathematical algorithms which solve many problems as explained in below use cases -

* Product recommendation systems for E-commerce websites ( Flipkart, Amazon etc.)
* Media Recommendation systems ( Netflix, Hotstar )
* Chatbots ( Voice/Text ) – Responding or resolving customer problems via automated processes
* Credit Risk Applications – Predicting whether a loan applicant can be a defaulter or not
* Fraud detection – To predict whether a transaction is a fraud or not, used by banks and credit card companies
* Inventory Optimization – Predicting sales of products to manage inventory efficiently
* Healthcare Applications – Predicting whether a Cancer tumor is malignant or benign
* Self-driving cars

And many more…

**Concepts for this week :**

* *End-to-end Machine Learning Pipeline*
* *Feature Engineering –* Preprocessing required before building a ML model
* *Measures of Accuracy* – Performance indicators of how good a ML model is. Each of the below will be discussed in detail in coming sessions -
  + *R^2 and Adjusted R^2*
  + *Confusion Matrix*
  + *Type I and Type II Errors*
  + *Accuracy*
  + *Recall*
  + *Precision*
  + *F Beta*
  + *ROC and AUC*
* *Bias and Variance –* Important concepts related to performance of a model
* *Supervised Machine learning algorithms* 
  + *Regression Techniques*
  + *Classification Techniques*
* *Unsupervised Machine learning algorithms*
* *End to end implementation in a Project*

# End to End ML pipeline/ Workflow

A ML pipeline is the sequence of steps which are typically performed to solve a problem using Machine Learning. It consists for following steps as mentioned in the flowchart below -

1. **Identifying Problem Space –**

Identifying and understanding a problem is the first and very crucial step of an ML project. Without a well defined and proven problem, any project is bound to fail or deliver bad results. But often, this part is not overlooked upon and teams start working on a half baked problem.

For example :

Problem : Prepare a model for better personalized marketing strategy for a shoe retail company.

Above is a very general problem statement and needs more work to get to more precise on outcome.

Better Problem Statement : Prepare a model to predict sales of customers in next 1 month so that marketing spend is allocated efficiently.

This problem clearly states what the stakeholder wants, and why does he want it, hence it can be a good starting point for the workflow.

1. **Data Ingestion :**

Once the problem is well defined and crafted, the next step is to analyze what kind of data can be procured, and from where. In many cases, it can be in house company data that can be used to solve a company’s problem, while in other instances there might be third party data providers which may provide the data on chargeable contract.

Corelogic, Epsilon , IDM are some of the known data providers in US region.

Since the quanta of this data can be huge ( Petabytes of data ), it is usually stored in cloud data warehousing solutions such as Google’s GCP, Amazon’s AWS, Microsoft Azure etc. When working in a corporate project, it is very likely that a data scientist will get to work on these.

1. **EDA :**

EDA (Exploratory data analysis ) has already been explained as practiced as part of previous week. While EDA plays a huge role for data analysis and finding insights from the data, it is also a very important step in a ML workflow.

The data has to be cleaned and treated well before it can be fed to an algorithm otherwise the results might not be accurate.

Furthermore, Univariate and Bivariate analysis, Hypothesis testing etc provide information about useful patterns and relationships in data which can help in feature selection. For example – 1 out of 2 highly correlated variables can be dropped as the addition of other will not add significant value to the model.

1. **Feature Engineering**

It involves concepts like Standardization/Normalization of data, Principal Component Analysis etc. These concepts will be covered later in this or subsequent sessions in detail. On a high level, feature engineering is required to make the job easier for algorithm and to reduce computational power required.

It also involves transformation of variables as per use case so that variables can be made usable for a model. For example, for a real-estate data, the year built of a house ( 1956, 1978 etc. ) can hardly be used in any model. But the same can be converted to house age by formula : (Current Year – Year Built), which can be a very useful factor for a given problem.

1. **Model Building**

Once the data is in a polished format for a ML algorithm, multiple models are trained based on relevant algorithms for that problem.

The data is divided into 2 sections : Train data and Test data. ( Let’s say 80%/20% split is done)

The model is trained on trained data and then the performance is checked based on model output on test data. It is like preparing a model for an exam by showing him questions and answers based on 80% data, and then taking a test by asking questions from the rest of the data.

It is done so that the best algorithm can be chosen based on the best performance of the model.

1. **Model Evaluation & Selection**

Finally, out of all the trained models, 1 model is selected based on the best performance.

There are various methods of measuring performance of a model which will be covered later.

1. **Model Deployment –**

The selected model is deployed either on a web based UI or at backend on a cloud data warehouse.

For example, in healthcare apps, chatbots etc., the model may be deployed as a UI application to be used by doctors directly.

While for other use cases such as Credit Risk models, the model is deployed on backed which constantly generates output regarding credit worthiness of an applicant. This data can be further used by a loan manager to decide whether to approve or reject a loan.

# Introduction to Machine Learning Algorithms

Machine Learning Algorithms fall into 2 categories:

1. Supervised Learning
2. Unsupervised Learning
3. **Supervised Learning :**

These are the type of ML algorithms where the model is train data is “tagged” or “mapped/labelled” to an output or “Target” variable. The model gets trained on this train data and then predicts output or “Target” for test data.

Let’s take an example to understand the above in simpler terms :

Let’s say *Predict\_pet* is a ML model which predicts the pet animal. The model is used to predict whether the pet is a *Cat* or *Dog* or *Horse* based on its Height, Weight, Tail Length, Eye Radius, Hair thickness etc.

Now in the train data, the model needs to be provided with “target” values “Cat/Dog/Horse” for it to learn and be able to predict the same for the test data.

Once it learns from the train data, it can classify any pet as one of these 3 animals accurately.

As described earlier, in other ways, Train data is “Tagged” to a target in this case and the model has some sample data to recognize necessary patterns.

Such algorithms are called Supervised Algorithms.

Supervised algorithms are also of 2 types :

1. **Classification**

The target variable is Categorical.

Examples –

* Predicting pet animal type based on animal attributes
* Predicting whether an applicant is a loan defaulter or not
* Predicting whether a customer is loyal customer or not
* Spam detection applications
* Image Classification
* Fraud Detection

1. **Regression**

The target variable is Quantitative.

Examples-

* House Price Prediction
* Employee bonus recommendation
* Sales forecast

**2**. **Unsupervised Learning :**

Contrary to Supervised learning, in Unsupervised learning use cases, the data is not mapped/tagged/labelled to a target variable in the train dataset. It effectively means there is no target variable in such cases.

The model essentially has no benchmark to learn from and hence the name – it is completely unsupervised.

The task or objective of such models is to effectively group/cluster/map data points with respect to each other without any prior input from the data itself.

Unsupervised Learning techniques broadly fall into 2 categories –

1. Clustering Techniques – Where the objective is to group the data points into N number of clusters based on similarities and patterns among data
2. Association Techniques – Where the task is to find specific rules in the data, for example – Those who buy X also tend to buy Y

Some common use cases of Unsupervised learning are as follows -

* Market basket analysis
* Identifying Accident prone areas
* Anomaly detection
* Customer Segmentation
* Song/Films/Shows recommendations

The below chart summaries the breakup of techniques that’ve discussed-

**Differences between Supervised and Unsupervised Learning -**

|  |  |
| --- | --- |
| **Supervised** | **Unsupervised** |
| Train data contains labelled target variable for the model as a reference to train upon | No labelled field |
| Complexity is usually lower than Unsupervised techniques | High complexity |
| Main goal is to predict an output | Main goal is to find hidden patterns and associations in data |
| The results are more accurate in Supervised techniques since there are well defined metrics to measure performance and to improve upon | The results are not as impactful or accurate as in Supervised models |
| It accepts feedback and improves the model accordingly | There is no feedback involved |

# 4. Train/Test Split

Train/test split is the first step of model building ( Only for Supervised learning).

For supervised algorithms, there is need to quantify the performance of the model so that the best model selection can be done. Moreover, it is also useful if one wants to improve model based on some feedback.

For this reason, the data is usually split into 2 sections before building the model :

1. Train data ( X % )
2. Test data ( 100-X%)

Usually the Train data comprises of majority of the data for model to have enough training data.

(80%/20% OR 75%/25% etc.)

Train data ( X%)

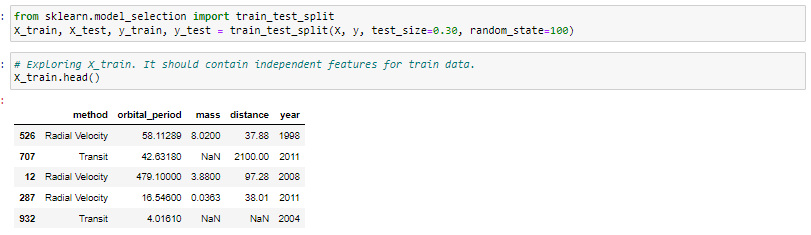
Test Data ( 100-X%)

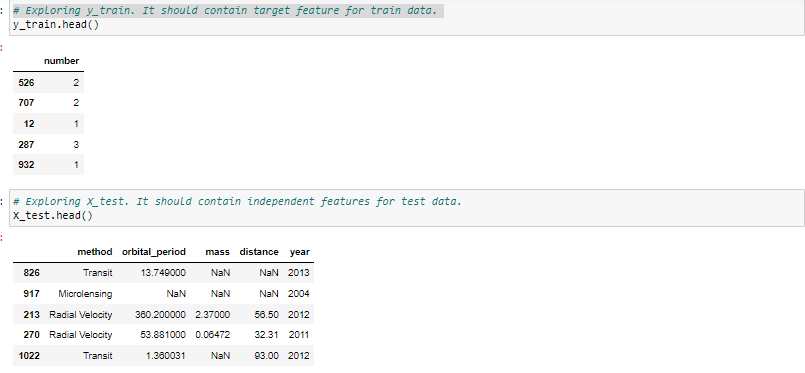
The model is trained on 1 set, called Train dataset. From this part of data, model learns the rules and patterns required to predict the Target variable correctly. It is like schooling the model for an exam later.

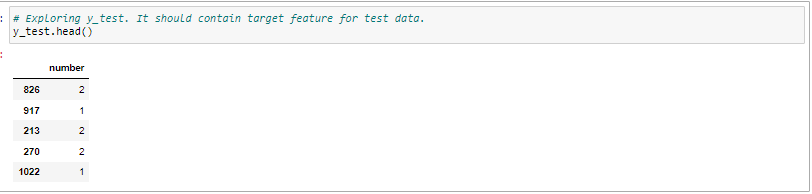
Once done, then the model is used to predict Target for Test data. Please note that we already have the correct/actual target variable for Test dataset. The actual Target data is compared with Predicted Target to measure the performance of the model.

One disadvantage of this method is that performance of the model is extremely dependent on how the split is done. It is guaranteed that model performance will vary a lot with different random splits of train/test data as model will be fed different data to train on each time. This introduces a bias in the model and hampers the performance. We will be learning some advanced train/test splitting techniques later in this week to resolve this issue.

The code for performing train/test split is as below –







# 5. Performance Metrics

Once the train test split is done and model(s) are trained on train dataset, it is important to measure the performance of the model by some Performance indicators or metrics. Based on these metrics, one can take the decision to either chose and deploy a model , or go back and improve the performance by some parameter tuning. Hence, a good understanding of these metrics is extremely important.

Since there are different types of ML techniques as we’ve learnt in this session, the performance measurement also varies accordingly.

**For Classification Models –**

There are following important concepts/measurement metrics for Classification models –

* Confusion Matrix
* Accuracy
* Type I error
* Type 2 error
* Recall
* Precision
* F- Beta
* ROC Curve - Receiver Operating Characteristics Curve
* AUC – Area under the Curve ( ROC Curve)

Classification models are also of 2 types –

* Outcome based on Probability – Binary outcome models such as a model predicting whether it is likely to rain or not on a particular day.

AUC and ROC from the above list are related to such scenarios.

* Outcome as Class Labels – Such as Image classification model predicting the Product name from the image

All other concepts apart from AUC and ROC are applicable in such models.

**For Regression Models –**

* R^2
* R^2 Adjusted
* Mean Absolute Error (MAE)
* Mean Squared Error ( MSE)
* Root Mean Squared Error ( RMSE)
* Root Mean Squared Log Error ( RMSLE)
* Mean Absolute Percentage Error ( MAPE )

**For Clustering Models –**

* Silhouette Score
* Rand Index
* Davis Bouldin Index
* Calinski Harabasz Index

Below is a detailed chart summarizing all the concepts/metrics discussed –

Let’s discuss some of the most fundamental ones here in the interest of time so that we are prepared to understand the rest of the advanced metrics when we’ll discuss those in this week later -

* **Confusion Matrix**

If there is a good understanding of train/test split and how the model is trained on train data and then predicts outcome for test data, one would understand that model would be able to predict some of the outcomes correctly and the others incorrectly.

Let’s take an example of Cancer data of 100 patients. For sake of simplicity let’s consider 90 records were in train data, the model gets trained based on those 900. For the rest 10, our model is going to predict whether the patient has a Malignant or Benign tumor.

Now let’s look at below predictions vs Actual results –

(0 represents Benign

1 represents Malignant)

|  |  |  |
| --- | --- | --- |
| **#** | **Actual Target** | **Predicted**  **Target** |
| 1 | 0 | 1 |
| 2 | 1 | 1 |
| 3 | 0 | 1 |
| 4 | 1 | 1 |
| 5 | 1 | 0 |
| 6 | 0 | 1 |
| 7 | 1 | 1 |
| 8 | 0 | 0 |
| 9 | 0 | 0 |
| 10 | 1 | 0 |

From above 10 predictions , one can calculate 4 kinds of results –

* Where 1 is Correctly Predicted (Also called **True Positive**): 3 Instances (# 2,4,7)
* Where 1 is Incorrectly Predicted (Also called **False Positive**): 2 Instances ( #5,10)
* Where 0 is Correctly Predicted (Also called **True Negatives**): 2 Instances (# 8,9)
* Where 0 is Incorrectly Predicted (Also called **False Negatives**): 3 Instances (#1,3,6)

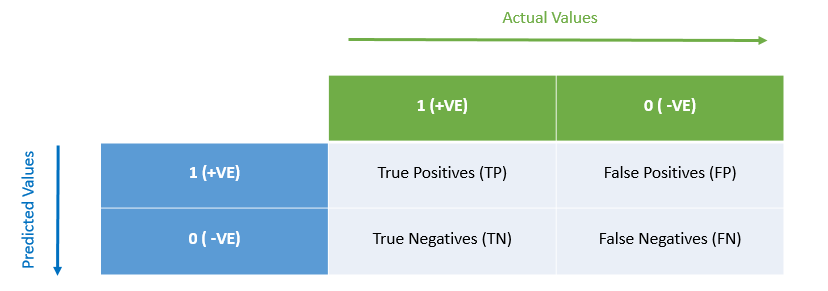
Naming Convention –

Here, True/False corresponds to Correct/Incorrect predictions respectively

Positive/Negative corresponds to Predicted 1/0 respectively

Finally, let’s talk about Confusion Matrix now.

All the above 4 values (TP,TN, FP, FN) can be represented in form of a matrix as below. This matrix is known as confusion matrix-



**But what is the use of Confusion Matrix ?**

Multiple other indicators such as Accuracy, Precision , Recall, Type 1 Error , Type 2 error can be obtained from here.

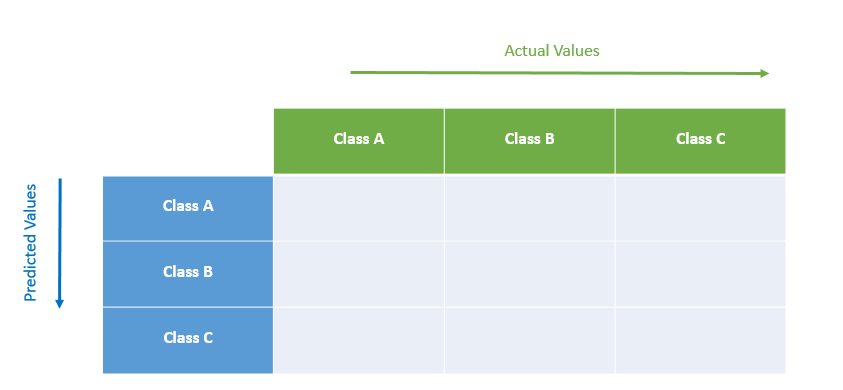
**Confusion Matrix for Multi-class predictor model –**

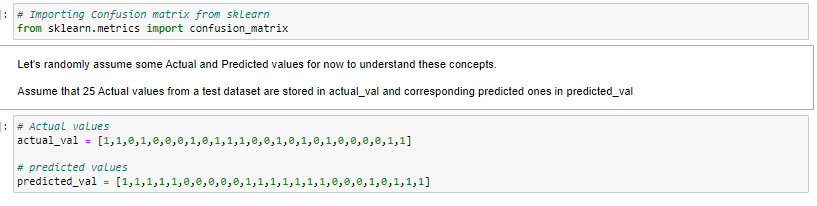
The above discussed example considers a binary classification model to explain the Confusion matrix concept.

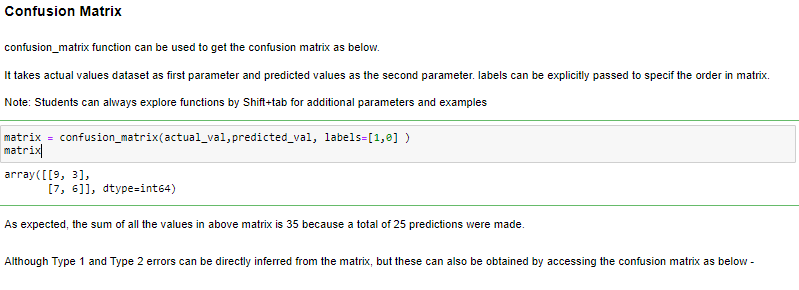
In most of the textbooks and articles, it is common that confusion matrix is only discussed for a binary classification model. This creates confusion among learners regarding how it would look like for a multi class model.

The Confusion Matrix for a multiclass model will not have “Positive”/”Negative” in its terminology and will be an NXN matrix for an N- Class Label predictor.

For example, for a model that predicts target variable from 1 out of 3 labels, the Confusion Matrix would look like –







* **Accuracy**

Accuracy is simply the % of correct predictions. Mathematically, for a binary classification model, it can be given as –

*Accuracy = (TP+TN) / (TP+TN+FP+FN)*

In simpler words –

*Accuracy = (Number of correct predictions) / (Number of total predictions)*

The more the Accuracy, the better predictor a model is. But this is not a universal rule and the reason will be discussed shortly. But generally, a model with 90% accuracy is better than the one which predicts 60% times successfully.

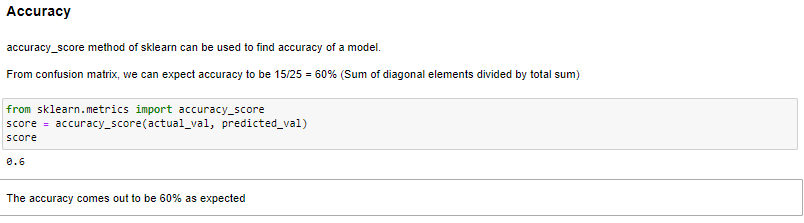
For an N-Label predictor model, the accuracy can be obtained by summing up the diagonal values in confusion matrix and dividing it by the size of test dataset. (The Diagonal of a confusion matrix always represent correctly predicted outcomes )

Disadvantage :

Accuracy can only be a good indicator for a balanced dataset. Balanced datasets have fairly equal distribution of 0s and 1s as Targets in training data for the model to be unbiased.

For example if training data had 1000 rows, and out of 1000, 900 belonged were labelled as 0s, then the trained model will get biased because it was better trained to predict 0s, as compared to 1s.

There are ways to deal with this disadvantage cases, which will be discussed later in classification session.



* **Type 1 & Type 2 Errors**
* **Type 1 Error**

When the Actual Negative class ( 0 ) is incorrectly predicted , this is known as Type-I error.

Essentially, **False Positives** are Type-I errors.

Examples :

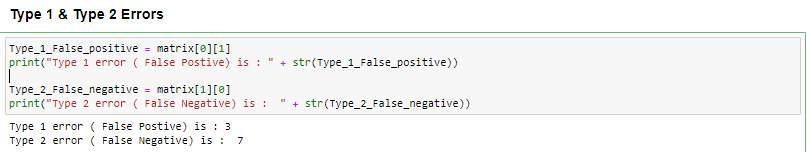
* Predicting a Student to be eligible (1) for admission in Harvard Business School when in reality he was not eligible (0) due to poor grades.
* Predicting a mail as SPAM (1) when it is not a SPAM (0).
* **Type 2 Error**

When the Actual Positive class ( 1 ) is incorrectly predicted , this is known as Type-II error.

Essentially, **False Negatives** are Type-II errors.

Examples :

* Predicting a Student to be NOT eligible (0) for admission in Harvard Business School when in reality he was the topper of the country and was eligible (1)
* Predicting a cancer tumor as benign (0) when in real it was deadly (1).



# 6. Feature Engineering

Feature Engineering is a pre-processing step before training an ML model. Feature engineering serves following objectives –

* **Reduce the computational power required to train the model**. While we usually overlook this factor in ML class based mock projects, but in real world it is a huge factor since the size of data can be as high as Pentabytes or more. A model can take anywhere between few hours to weeks to train. Moreover, the systems running these computations are usually paid cloud services, so any extra computation adds up to the costs. For these reasons, it becomes extremely crucial to pre-process the data in a way such that the training requires less power.
* **Improving the accuracy.** More often than not, treating your data in the right way will not only make the job easier for computer, but it also improves the accuracy. For example, for regression models scaling of the data is must otherwise they produce bad models.
* **Get more value out of data.** Extra information can be extracted out of available data which in turn might be very useful for the model. For example, for a model which predicts house prices, the field “Year built (Contains year in which the house was built , eg 1987, 1913, 1999 etc)” cannot add much value. But the same can be converted to “Age of the house” by transformation as “Current Year – Year Built” , and that can be an extremely important variable for the model.

Now that we have discussed the goals of Feature engineering, let’s see what all techniques it actually entails –

1. Data Cleaning
   1. Null Value Treatment
   2. Outlier Treatment
   3. Treating incorrect labels (Merge “Male”, “M”, “Men” all to one category -> “M”)
   4. Date Values Treatment (Extracting Week, month, day from complete date)
   5. Feature Extraction (Extracting *House Age* from *Year Built*)
2. Exploratory Data Analysis
3. One-hot Coding
4. Feature Scaling
5. Dimensionality Reduction

Data Cleaning and EDA have been extensively discussed and practiced already in previous week. EDA can also be useful in feature selection as features having relationships among themselves can be dropped from training data. The reason behind this would be discussed later in this week.

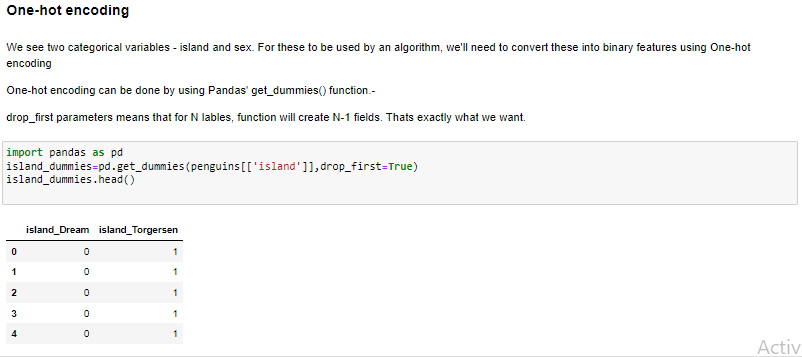
Let’s discuss rest of the concepts –

1. **One-hot encoding**

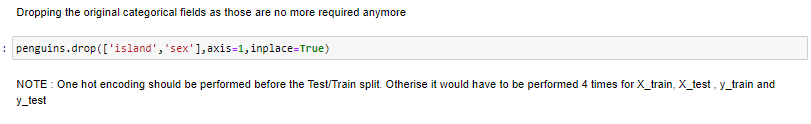
Computerised algorithms don’t understand categorical data. Hence, it is a common practice to encode Categorical variable to binary fields for the model.

A categorical variable having N labels ( distinct values) can be represented by N-1 binary columns (having 0s and 1s). This is because if all the N-1 fields are 0, then that is enough information for the model to deduce that leftover field must be 1.

It can be implemented in Python as follows –







1. **Feature Scaling**

Scaling refers to transforming the numerical fields onto a common scale.

A dataset might have many different type of fields which might have different units, measured in different ways, have very different ranges. It might be unfair to train a model on such data since there is no benchmark for the model to prioritise the fields for training. There may be a scenario that a field gets high weightage or importance in predicting an outcome just because it had extremely high values as compared to other fields, but in reality that field might not have impact on the target, or that the high values were resulting due to choice of units.

To avoid such bias in the model, Feature scaling is done. While it is a must for some algorithms such as regression, it might be optional for some others like Decision trees. This difference is because of the underlying mechanism of these algorithms, which we’ll understand as we’ll discuss those concepts later.

It is advised that all the irrelevant columns are dropped before applying scaling techniques, so as to preserve computing power ( example – ID column)

Feature Scaling mainly is of 2 types –

* Standardisation
* Normalisation

**Standardisation**

The formula for Standardisation is given by –

Where

= Scaled value

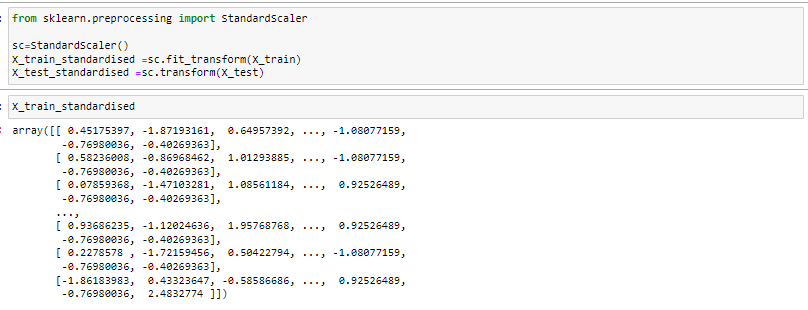
x(i) = unscaled value

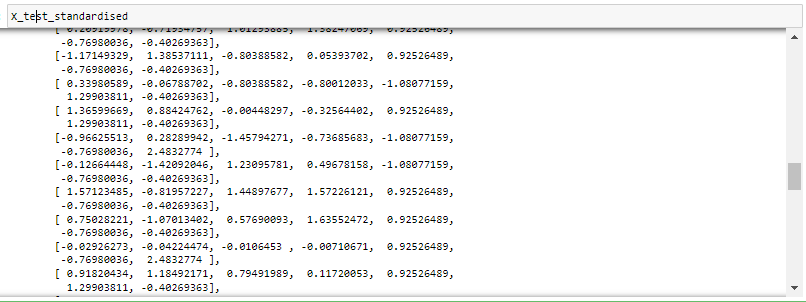
= Mean value of the column

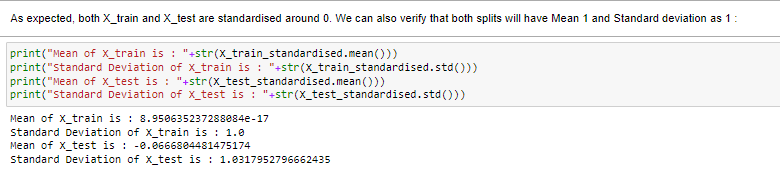
The resulting scaled value will have Mean as 0 and Standard deviation as 1.

Please note that Standardised Scaled columns will not have a defined range, but these will follow a Gaussian distribution

Standardisation can be implemented in Python as following –







**Normalisation**

It is also known as MinMax scaling.

This type of scaling will transform a numerical value to a range between 0 to 1.

The formula for this type of scaling is given by –

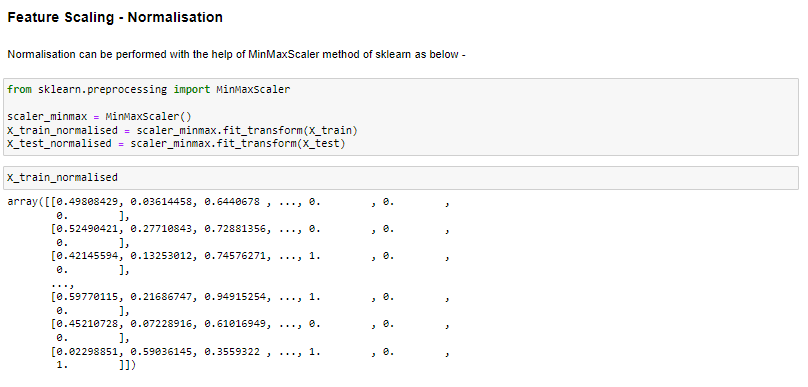
Where

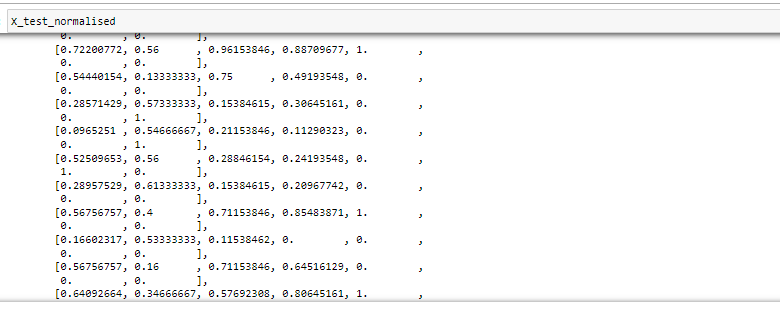
= Scaled value

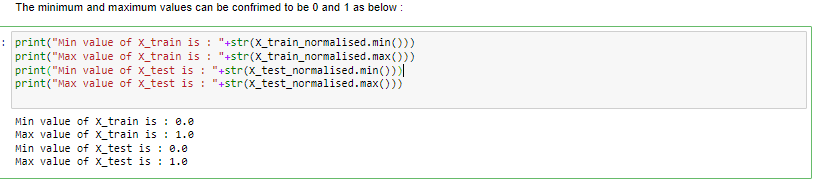
x(i) = unscaled value

= Minimum value of the column

It can be implemented in Python as –







**Which one to choose - Standardisation or Normalisation?**

This is an age old dilemma and there is no right answer here.

Although it has been observed that if the original distribution of the field follows a normal distribution then Normalisation can be preferred for better results. In other cases, Standardisation might work well.

But still, if it is possible given the cost and time constraints, the best approach is to try all 3 data – Raw, Normalised and Standardised, and compare the performance. The scaling technique with best performance can be used.

1. **Dimensionality Reduction**

In real word production scenario, there are cases when a model has 1000s of variables available to be trained upon.

In such scenarios, it becomes almost impossible to manually select and drop the features, and check collinearities.

To overcome this problem, dimensionality reduction techniques are used which transform multiple features into a really low count.

One such technique which is very common is called Principal Component Analysis.

We’ll touch base upon the same in more detail later this week as it is an advanced concept and requires some context.

This ends our fundamentals theory session on ML for this week. A lot of these concepts have set the right foundations which are going to make the whole ML journey a lot easier and exciting.

# 7. MCQ

1. In which of the following labelled data is used –
   1. Reinforcement learning
   2. Supervised learning
   3. Clustering
   4. Unsupervised learning
2. Identify the algorithm type : Using IPL data to predict which bowler is most like to take up wicket of a particular batsman –
   1. Regression
   2. Clustering
   3. PCA
   4. Classification
3. K-Mean is a technique associated with –
   1. Regression
   2. Classification
   3. Clustering
   4. None of these
4. Which of the following is an Unsupervised learning technique :
   1. Decision Trees
   2. Random Forest
   3. Lasso Regression
   4. Apriori Algorithm
5. Market basket analysis is an example of –
   1. Clustering
   2. Ridge Regression
   3. Linear Regression
   4. Supervised Learning
6. Confusion matrix cannot be used in case of –
   1. Decision Trees
   2. Logistic Regression
   3. Ridge Regression
   4. SVM
7. Accuracy is calculated as –
   1. TP / TN
   2. (TP+TN ) / (TP+TN+FP+FN)
   3. (TP+TN)/ (FP+FN)
   4. (TP+FP)/ (TN+FN)
8. A model trained on 98% data and tested on 2% data will have –
   1. High Bias & High Variance
   2. High Variance & Low Bias
   3. Low Bias and Low Variance
   4. Low Variance and High Bias
9. An under-fitted model will have –
   1. High Bias & High Variance
   2. High Variance & Low Bias
   3. Low Bias and Low Variance
   4. Low Variance and High Bias
10. Wrong prediction of a cancer tumor to be malignant when it actually is benign is –
    1. Type 1 error
    2. Type 2 error
    3. No error

Homework

You are provided with Flight data of various airlines across routes with prices. The level of the data is on 1 flight, listing all the details of the flight such as carrier, date, route, duration etc. The goal is to build a Regression model so as to predict prices of the flights if all these attributes for future flights are provided as input.

While that is the larger objective ( Prediction of price), but since we have not yet covered regression models in our sessions, students are asked to do following tasks –

* Check if data has missing values, if yes then do the appropriate imputation
* Transform datetime variables for better usability. For example – Extract date, month, year, time separately wherever possible
* Convert all the features into proper usable format. For example stops can be a numerical feature if “Stops” is removed and data type is changed to int. Similarly duration can be transformed.
* Make dummies for usable categorical variables
* Do Standardization or Normalization of applicable features