CONCEPTS OF MACHINE LEARNING

ADAPTIVE MODEL FOR LOCATION DETECTION

BY PRANAV CHOPDE, JORDAN UZAN

SUPERVISOR: OMER TZIDKI

TABLE OF CONTENT

INTRODUCTION TO MACHINE LEARNING

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. **Machine learning focuses on the development of computer programs** that can access data and use it learn for themselves.

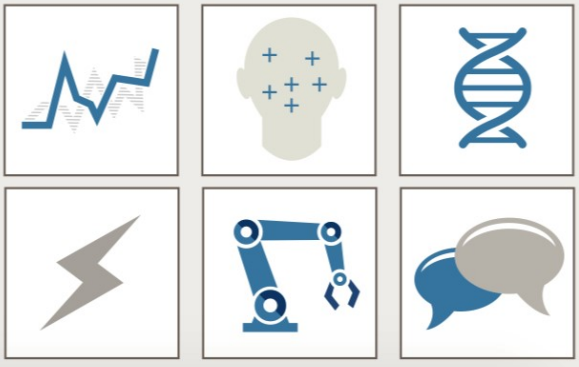
The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. **The primary aim is to allow the computers learn automatically** without human intervention or assistance and adjust actions accordingly.

Machine learning algorithms find natural patterns in data that generate insight and help you make better decisions and predictions. They are used every day to make critical decisions in medical diagnosis, stock trading, energy load forecasting, and more. Media sites rely on machine learning to sift through millions of options to give you song or movie recommendations. Retailers use it to gain insight into their customers' purchasing behaviour.

Real-World Applications

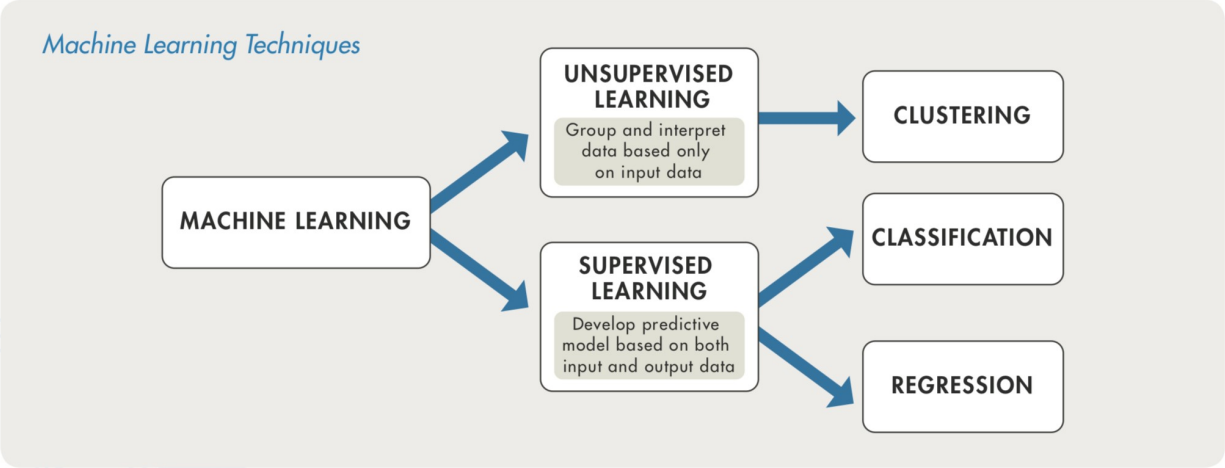
With the rise in big data, machine learning has become particularly important for solving problems in areas like these:

- Computational finance, for credit scoring and algorithmic trading
- Image processing and computer vision, for face recognition, motion detection, and object detection
- Computational biology, for tumor detection, drug discovery, and DNA sequencing
- Energy production, for price and load forecasting
- Automotive, aerospace, and manufacturing, for predictive maintenance
- Natural language processing



TECHNIQUES IN MACHINE LEARNING

Machine learning uses two types of techniques: supervised learning, which trains a model on known input and output data so that it can predict future outputs, and unsupervised learning, which finds hidden patterns or intrinsic structures in input data.

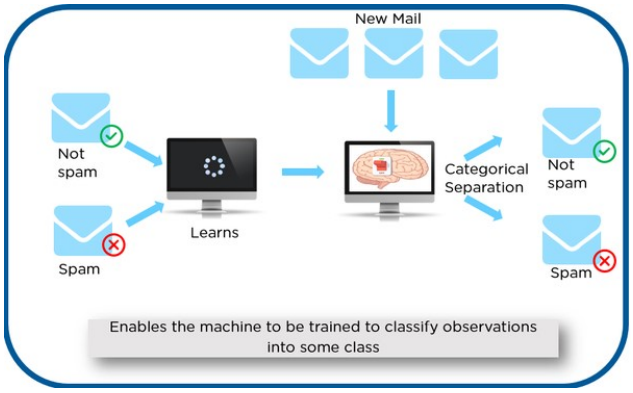


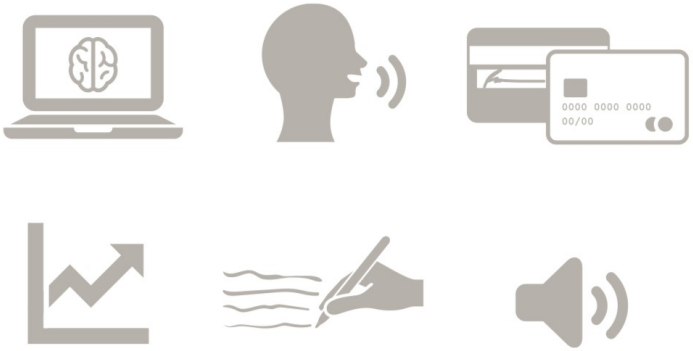
SUPERVISED LEARNING

The aim of supervised machine learning is to build a model that makes predictions based on evidence in the presence of uncertainty. A supervised learning algorithm takes a known set of input data and known responses to the data (output) and trains a model to generate reasonable predictions for the response to new data.

Supervised learning uses classification and regression techniques to develop predictive models.

- **Classification** techniques predict discrete responses—for example, whether an email is genuine or spam, or whether a tumour is cancerous or benign. Classification models classify input data into categories. Typical applications include medical imaging, speech recognition, and credit scoring.
- **Regression** techniques predict continuous responses- for example, changes in temperature or fluctuation in power demand. Typical application includes electricity load forecasting and algorithmic trading.





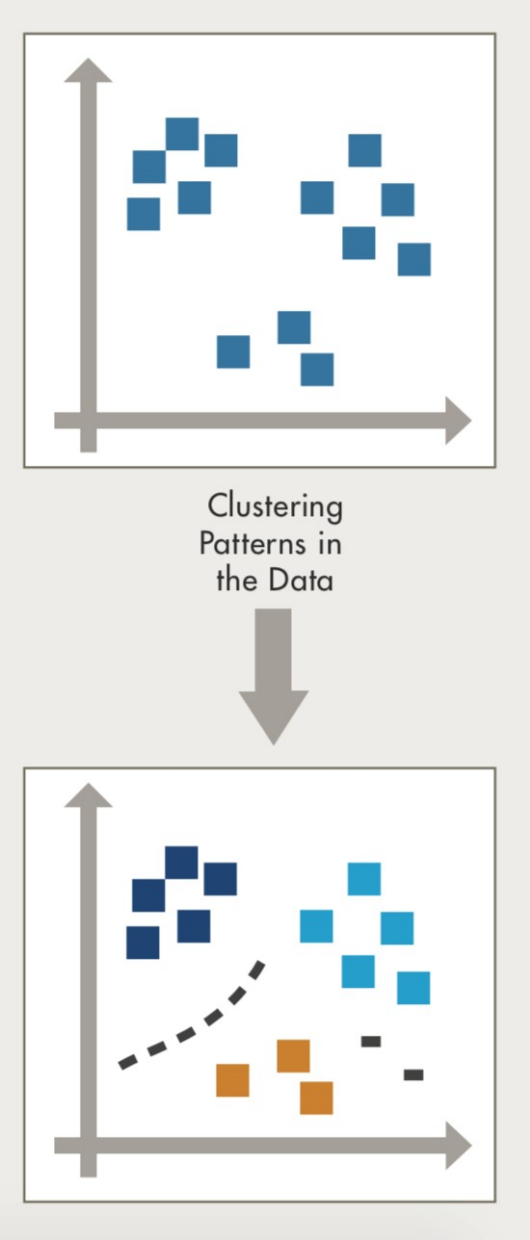
UNSUPERVISED LEARNING

Unsupervised learning finds hidden patterns or intrinsic structures in data. It is used to draw inferences from datasets consisting of input data without labeled responses.

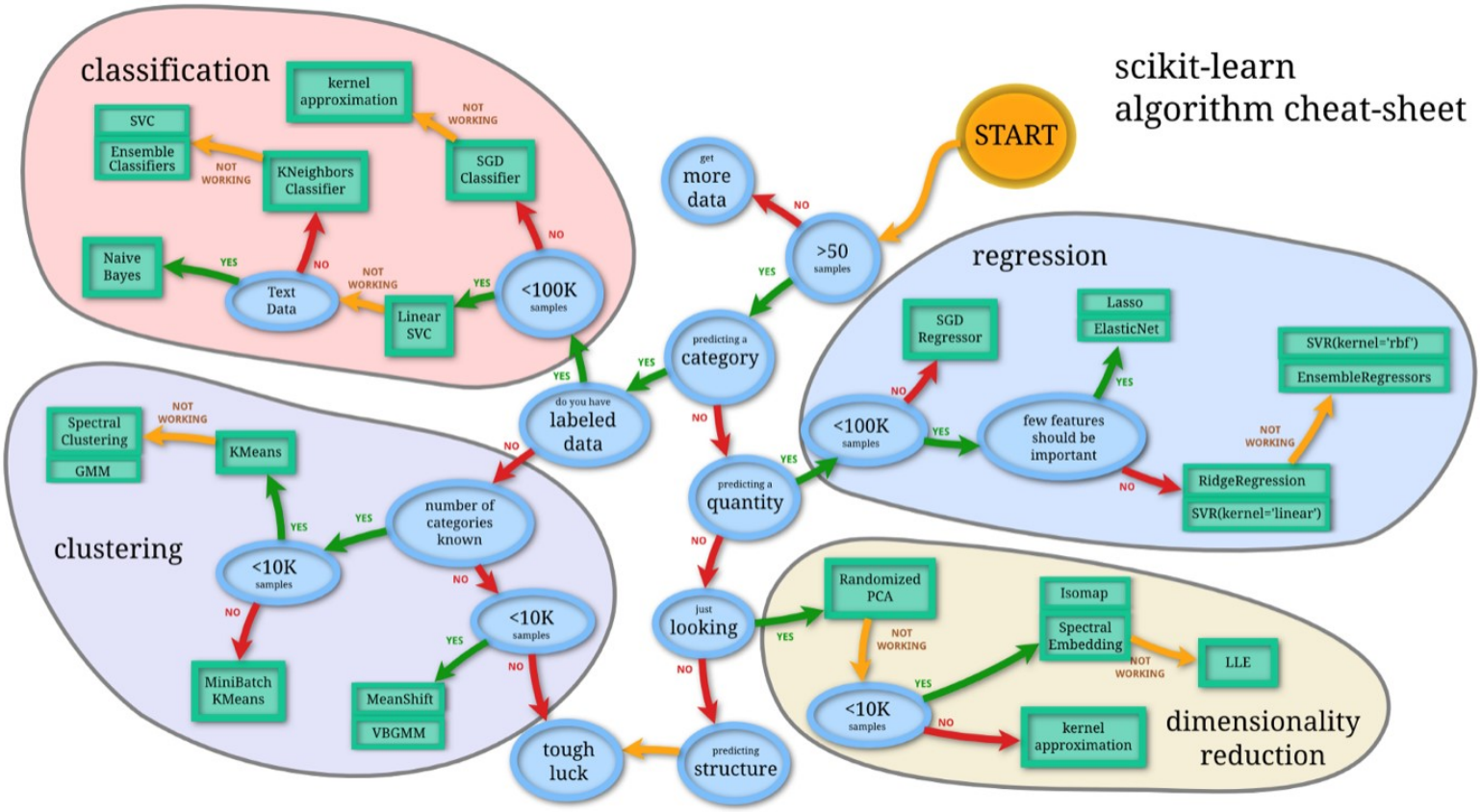
Clustering is the most common unsupervised learning technique. It is used for exploratory data analysis to find hidden patterns or groupings in data.

Applications for clustering include gene sequence analysis, market research, and object recognition.

A central application of unsupervised learning is in the field of density estimation in statistics,[1] though unsupervised learning encompasses many other domains involving summarising and explaining data features. It could be contrasted with supervised learning by saying that whereas supervised learning intends to infer a conditional probability distribution



HOW DO YOU DECIDE WHICH ALGORITHM TO USE?



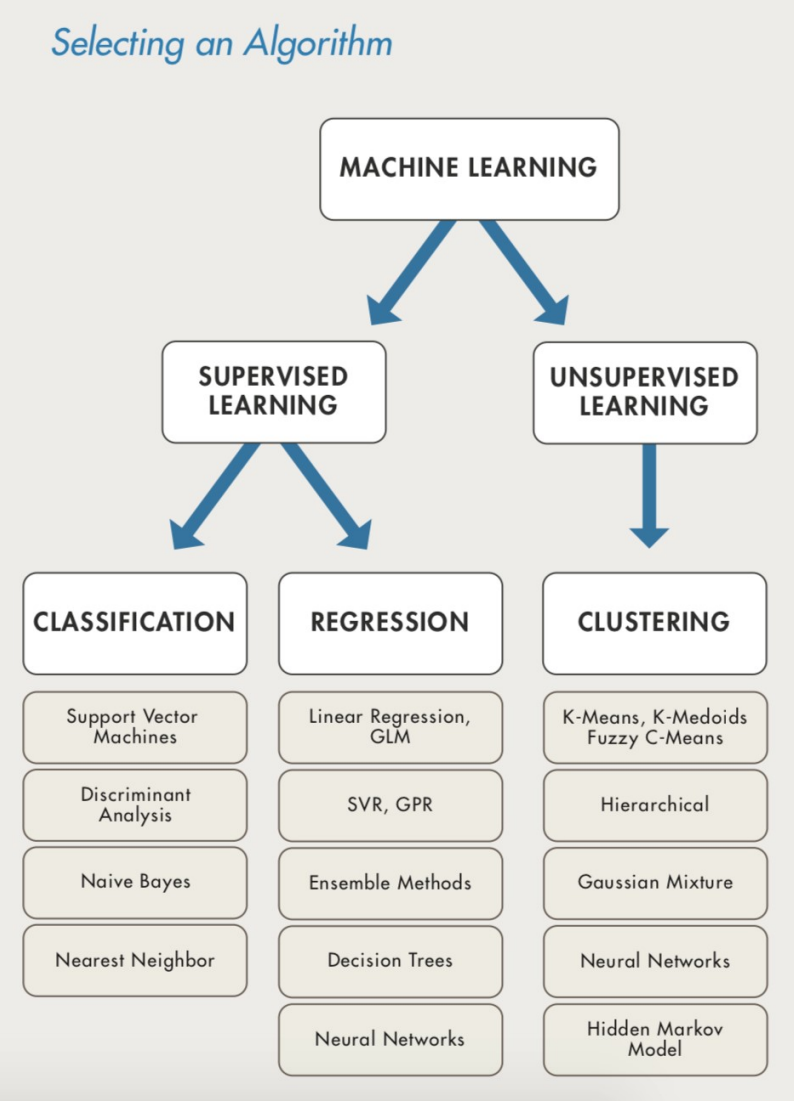
Type of problem: It is obvious that algorithms have been designed to solve specific problems. So, it is important to know what type of problem we are dealing with and what kind of algorithm works best for each type of problem. I don't want to go into much detail but at high level, machine learning algorithms can be classified into Supervised, Unsupervised and Reinforcement learning. Supervised learning by itself can be categorised into Regression, Classification, and Anomaly Detection.

Size of training set: This factor is a big player in our choice of algorithm. For a small training set, high bias/low variance classifiers (e.g., Naive Bayes) have an advantage over low bias/high variance classifiers (e.g., kNN), since the latter will overfit. But low bias/high variance classifiers start to win out as training set grows (they have lower asymptotic error), since high bias classifiers aren't powerful enough to provide accurate models [1].

Accuracy: Depending on the application, the required accuracy will be different. Sometimes an approximation is adequate, which may lead to huge reduction in processing time. In addition, approximate methods are very robust to overfitting.

Training time: Various algorithms have different running time. Training time is normally function of size of dataset and the target accuracy.

Linearity: Lots of machine learning algorithms such as linear regression, logistic regression, and support vector machines make use of linearity. These assumptions aren't bad for some problems, but on others they bring accuracy down. Despite their dangers, linear algorithms are very popular as a first line of attack. They tend to be algorithmically simple and fast to train.



Number of parameters: Parameters affect the algorithm's behaviour, such as error tolerance or number of iterations. Typically, algorithms with large numbers parameters require the most trial and error to find a good combination. Even though having many parameters typically provides greater flexibility, training time and accuracy of the algorithm can sometimes be quite sensitive to getting just the right settings.

Number of features: The number of features in some datasets can be very large compared to the number of data points. This is often the case with genetics or textual data. The large number of features can bog down some learning algorithms, making training time unfeasibly long. Some algorithms such as Support Vector Machines are particularly well suited to this case [2,3]

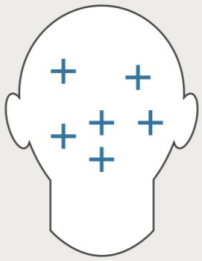
Choosing the right algorithm can seem overwhelming—there are dozens of supervised and unsupervised machine learning algorithms, and each takes a different approach to learning.

There is no best method or one size fits all. Finding the right algorithm is partly just trial and error—even highly experienced data scientists can't tell whether an algorithm will work without trying it out. But algorithm selection also depends on the size and type of data you're working with, the insights you want to get from the data, and how those insights will be used.

WHEN SHOULD YOU USE MACHINE LEARNING?

Consider using machine learning when you have a complex task or problem involving a large amount of data and lots of variables, but no existing formula or equation. For example, machine learning is a good option if you need to handle situations like these:

Hand-written rules and equations are too complex—as in face recognition and speech recognition.



The rules of a task are constantly changing—as in fraud detection from transaction records.

The nature of the data keeps changing, and the program needs to adapt—as in automated trading, energy demand forecasting, and predicting shopping trends.

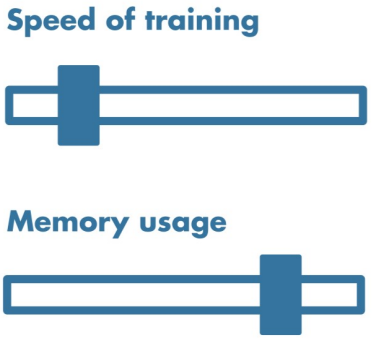
SELECTING THE RIGHT ALGORITHM

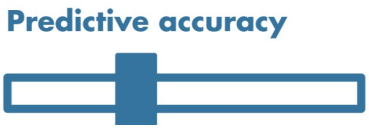
It's also a trade-off between specific characteristics of the algorithms, such as:

- Speed of training
- Memory usage
- Predictive accuracy on new data

• Transparency or interpretability (how easily you can understand the reasons an algorithm makes its predictions)

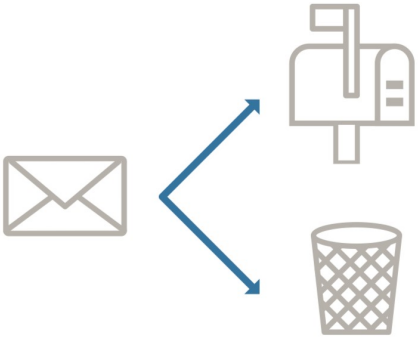
Let's take a closer look at the most commonly used classification and regression algorithms.

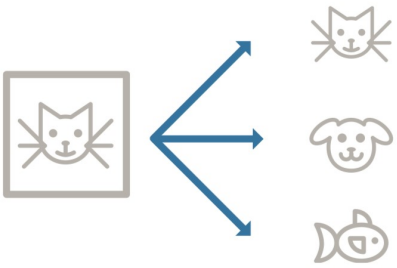




BINARY VS. MULTI CLASS CLASSIFICATION

When you are working on a classification problem, begin by determining whether the problem is binary or multi class. In a binary classification problem, a single training or test item (instance) can only be divided into two classes—for example, if you want to determine whether an email is genuine or spam. In a multi class classification problem, it can be divided into more than two—for example, if you want to train a model to classify an image as a dog, cat, or other animal. Bear in mind that a multi class classification problem is generally more challenging because it requires a more complex model.





MACHINE LEARNING CHALLENGES

Most machine learning challenges relate to handling your data and finding the right model.

Data comes in all shapes and sizes. Real-world datasets can be messy, incomplete, and in a variety of formats. You might just have simple numeric data. But sometimes you are combining several different data types, such as sensor signals, text, and streaming images from a camera.

Preprocessing your data might require specialised knowledge and tools. For example, to select features to train an object detection algorithm requires specialised knowledge of image processing. Different types of data require different approaches to preprocessing.

It takes time to find the best model to fit the data. Choosing the right model is a balancing act. Highly flexible models tend to overfit data by modelings minor variations that could be noise. On the other hand, simple models may assume too much. There are always tradeoffs between model speed, accuracy, and complexity.

Sounds daunting? Don't be discouraged. Remember that trial and error is at the core of machine learning—if one approach or algorithm doesn't work, you simply try another. But a systematic workflow will help you get off to a smooth start.

Statement of the problem:

We have 11 Features and 1 Response variable, The Response quantity is predicted distance between the 2 receiving and 2 transmitting Antenna, Since we know the distance between 2 receiving and 2 transmitting Antenna for some data in advance and we will predict the Response Variable accordingly, that makes the problem regression type in supervised machine learning category.

Now having classified our problem as a regression type in the supervised machine learning type category, lets go on to understand the regression learning and the types of modules in regression learning

REGRESSION LEARNING

Regression is a machine learning algorithm based on **supervised learning**. It performs a regression task. It is mostly used for finding out the relationship between variables and forecasting. I will try to describe it more specifically.

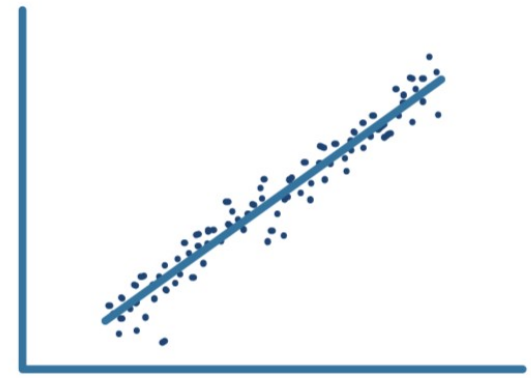
Regression models are used to predict a continuous value. Predicting prices of a house given the features of house like size, price etc is one of the common examples of Regression. It is a supervised technique.

Let's discuss different types of modules in regression learner app. Since we are dealing with regression, we will use all the modules and select the open with the Least RMSE error, since this module will be useful for other model

LINEAR REGRESSION

How it Works:

Linear regression is a statistical modelings technique used to describe a continuous response variable as a linear function of one or more predictor variables. Because linear regression models are simple to interpret and easy to train, they are often the first model to be fitted to a new dataset.



Best Used...

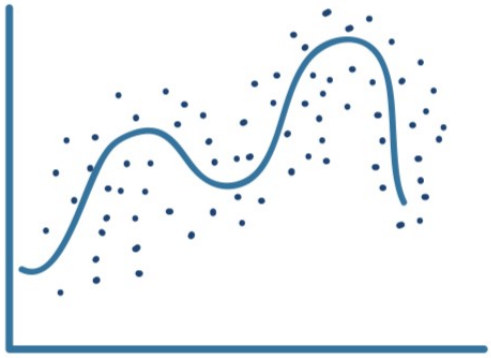
• When you need an algorithm that is easy to interpret and fast to fit

• As a baseline for evaluating other, more complex, regression models

NONLINEAR REGRESSION

How It Works

Nonlinear regression is a statistical modelings technique that helps describe nonlinear relationships in experimental data. Nonlinear regression models are generally assumed to be parametric, where the model is described as a nonlinear equation.



"Nonlinear" refers to a fit function that is a nonlinear function of the parameters. For example, if the fitting parameters

are b0, b1, and b2: the equation y = b0+b1x+b2x2 is a linear function of the fitting parameters, whereas y = (b0xb1)/(x+b2) is a nonlinear function of the fitting parameters.

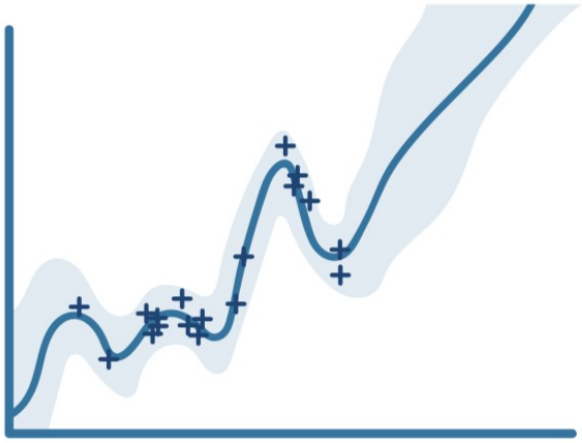
Best Used...

• When data has strong nonlinear trends and cannot be easily transformed into a linear space

- For fitting custom models to data

GAUSSIAN PROCESS REGRESSION MODEL

How it Works



Gaussian process regression (GPR) models are nonparametric models that are used for predicting the value of a continuous response variable. They are widely used in the field of spatial analysis for interpolation in the presence of uncertainty. GPR is also referred to as Kriging.

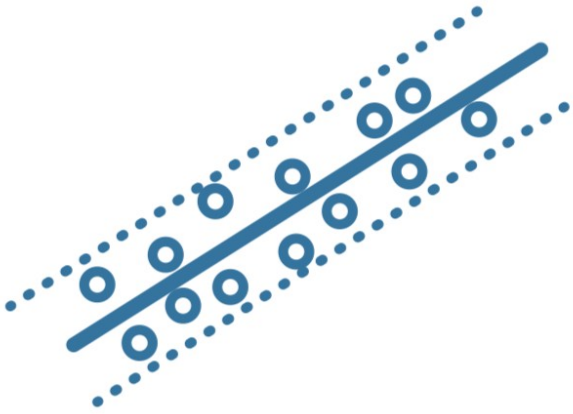
Best Used...

- For interpolating spatial data, such as hydrogeological data for the distribution of ground water

- As a surrogate model to facilitate optimisation of complex designs such as automotive engines

SVM REGRESSION

How It Works



SVM regression algorithms work like SVM classification algorithms, but are modified to be able to predict a continuous response. Instead of finding a hyperplane that separates data, SVM regression algorithms find a model that deviates from the measured data by a value no greater than a small amount, with parameter values that are as small as possible (to minimise sensitivity to error).

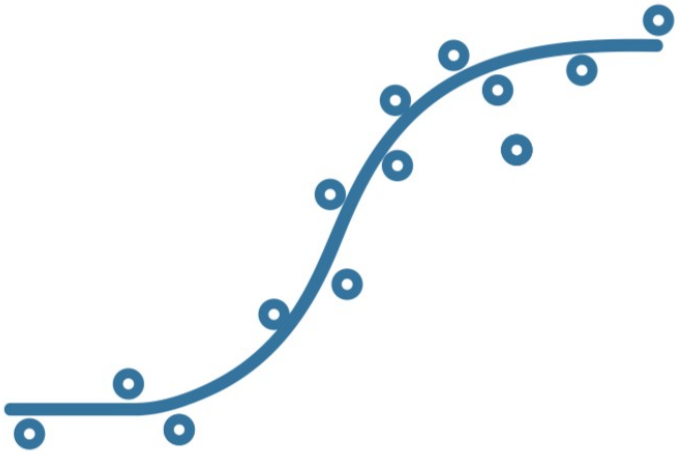
Best Used...

- For high-dimensional data (where there will be a large number of predictor variables)

GENERALISED LINEAR MODEL

How it Works

A generalised linear model is a special case of nonlinear models that uses linear methods. It involves fitting a linear combination of the inputs to a nonlinear function (the link function) of the outputs.



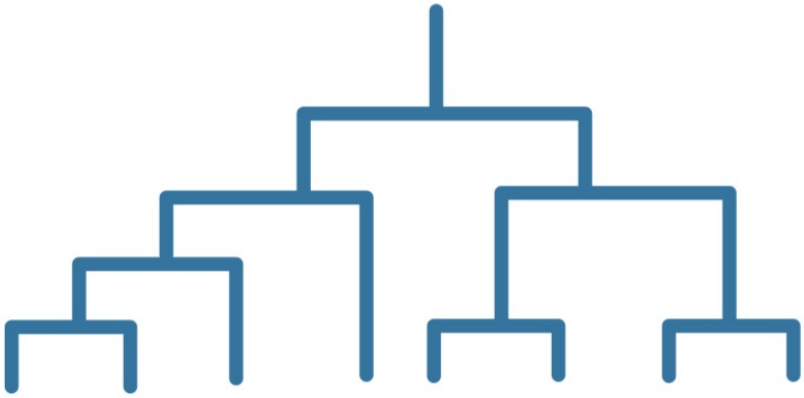
Best Used...

- When the response variables have non normal distributions, such as a response variable that is always expected to be positive

REGRESSION TREE

How It Works

Decision trees for regression are similar to decision trees for classification, but they are modified to be able to predict continuous responses.



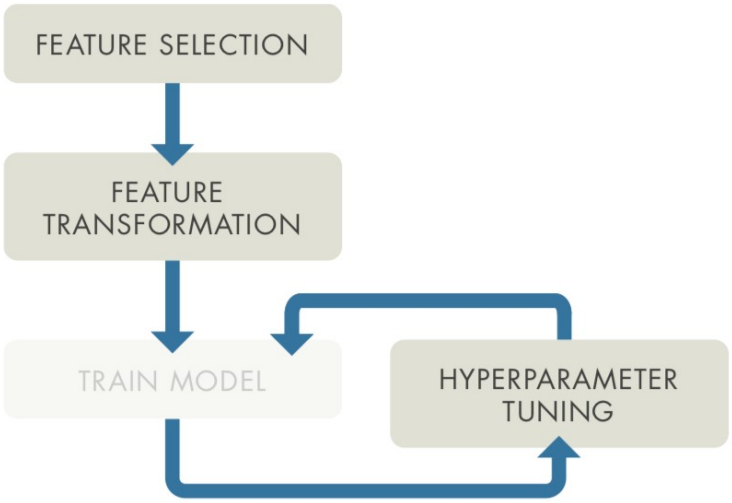
Best Used...

- When predictors are categorical (discrete) or behave nonlinearly

Now we have discussed all the regression learner machine learning module, how to make a machine learning algorithm that satisfies our needs

IMPROVING MODELS

Improving a model means increasing its accuracy and predictive power and preventing overfitting (when the model cannot distinguish between data and noise). Model improvement involves feature engineering (feature selection and transformation) and hyper parameter tuning.



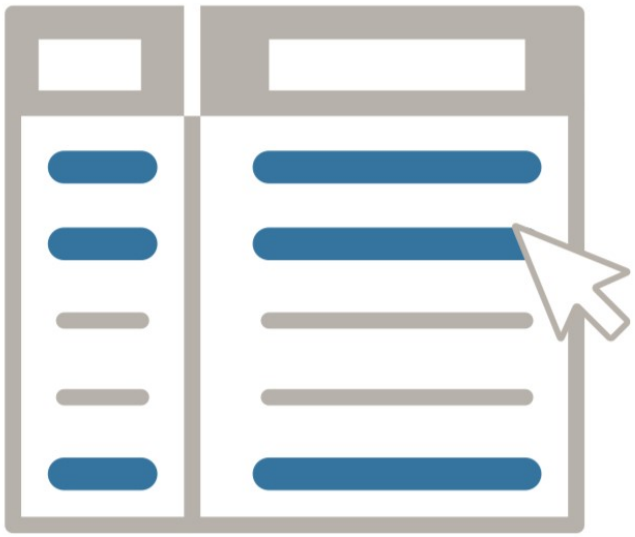
Feature selection: Identifying the most relevant features, or variables, that provide the best predictive power in modelings your data. This could mean adding variables to the model or removing variables that do not improve model performance.

Feature transformation: Turning existing features into new features using techniques such as principal component analysis, nonnegative matrix factorisation, and factor analysis.

Hyper parameter tuning: The process of identifying the set of parameters that provides the best model. Hyper parameters control how a machine learning algorithm fits the model to the data.

FEATURE SELECTION

Feature selection is one of the most important tasks in machine learning. It's especially useful when you're dealing with high- dimensional data or when your dataset contains a large number of features and a limited number of observations. Reducing features also saves storage and computation time and makes your results easier to understand.



Common feature selection techniques include:

Stepwise regression: Sequentially adding or removing features until there is no improvement

in prediction accuracy.

A model is only as good as the features you select to train it.

Sequential feature selection: Iteratively adding or removing predictor variables and evaluating the effect of each change on the performance of the model.

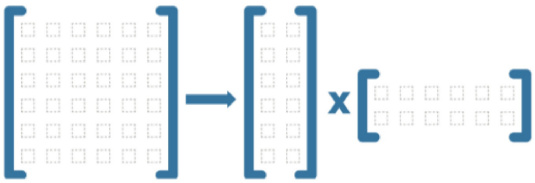
Regularisation: Using shrinkage estimators to remove redundant features by reducing their weights (coefficients) to zero.

Neighbourhood component analysis (NCA): Finding the weight each feature has in predicting the output, so that features with lower weights can be discarded.

FEATURE TRANSFORMATION

Principal component analysis (PCA): Performs a linear transformation on the data so that most of the variance or information in your high-dimensional dataset is captured by the first few principal components. The first principal component will capture the most variance, followed by the second principal component, and so on.

Nonnegative matrix factorisation: Used when model terms must represent nonnegative quantities, such as physical quantities.



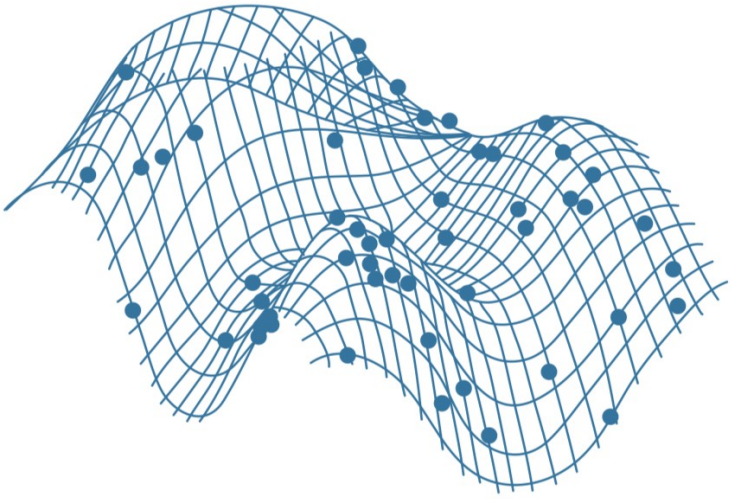
Factor analysis: Identifies underlying correlations between variables in your dataset to provide a representation in terms of a smaller number of unobserved latent factors, or common factors.

HYPERPARAMETER TUNING

Like many machine learning tasks, parameter tuning is an iterative process. You begin by setting parameters based on a "best guess" of the outcome. Your goal is to find the "best possible" values—those that yield the best model. As you adjust parameters and model performance begins to improve, you see which parameter settings are effective and which still require tuning.

Three common parameter tuning methods are:

- Bayesian optimisation
- Grid search
- Gradient-based optimisation



A simple algorithm with well-tuned parameters often produces a better model than an inadequately tuned complex algorithm.

REGRESSION LEARNER TO PREDICT DISTANCE BETWEEN ANTENNA

We will be making a regression learner model to predict the distance between a receiving and transmitting antenna, we take you through the complete workflow for developing a real-world machine learning application, from loading data to deploying a trained model. For each training phase, we demonstrate the techniques that are critical to achieving accurate models, and help you master the more challenging training tasks, including selecting algorithms, optimising model parameters, and avoiding overfitting.

In this book you'll also learn how to turn a model into a predictive tool by training it on new data, extracting features , hyper parameter optimisation.

In developing this application, we'll follow these steps:

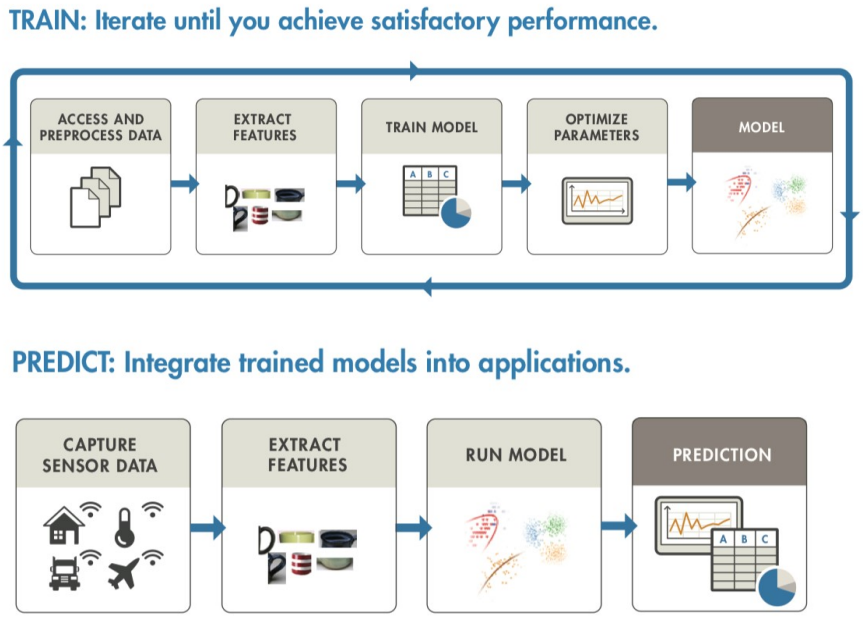
1. Access and explore the data.

2. Preprocess the data and extract features.

3. Develop predictive models.

4. Optimise the model.

5. Deploy analytics to a production system.



STEP 1: ACCESS AND EXPLORE THE DATA

The first step in any machine learning project is understanding what kind of data you are working with. Common ways to explore data include inspecting some examples, creating visualisations, and (more advanced) applying signal processing or clustering techniques to identify patterns.

STEP 2: PREPROCESS THE DATA AND EXTRACT FEATURES

Most datasets require some preprocessing before feature extraction— typical tasks include removing outliers and trends, imputing missing data, and normalising the data. We will now try to understand the features in the code and understand some standard terminology which is predictor and response variable. All other variables which will help us to create a model with the help of above specified modules are know as predictor and the result we want to estimate is know as response variable. We included all the features which includes response variable which is distance in a table

Extracting and selecting features helps improve a machine learning algorithm by focusing on the data that's most likely to produce accurate results.

Extracting Features

Feature extraction is one of the most important parts of machine learning because it turns raw data into information that's suitable for machine learning algorithms. Feature extraction eliminates the redundancy present in many types of measured data, facilitating generalisation during the learning phase. Generalisation is critical to avoiding overfitting the model to specific examples.

Selecting Features

While feature extraction is the first step, we need to avoid using too many features. A model with a larger number of features requires more computational resources during the training stage, and using too many features leads to overfitting.

The challenge is to find the minimum number of features that will capture the essential patterns in the data.

Feature selection is the process of selecting those features that are most relevant for your specific modelling problem and removing unneeded or redundant features. Common approaches to feature selection include stepwise regression, sequential feature selection, and regularisation.

Feature extraction can be time-consuming if you have a large dataset and many features. To speed up this process, you can distribute computation across available cores (or scale to a cluster) using the parfor loop construct in Parallel Computing ToolboxTM.

In the distance predictor example, based on our knowledge of regression, we extract the following types of features:

• phi_initiator, phi_reflector, RSSI_initiator, DQF(Data quality factor), frequency, PMU_diff, PMU_diff_LS, PMU_sum, PMU_sum_ls, y_diff,y_sum

freq_0	PMU_initiator	PMU_reflector	RSSI	PMU_diff	PMU_diff_LS	PMU_sum	PMU_sum_LS	y_diff	y_sum	DQF	Distance
2417	4.587	3.4116	-70	-38.8872	-4617.3	-117.6625	-185.3527	-38.8872	-117.6625	98	201
2420	0.8099	5.4487	-10	-51.9099	-4862.2	-177.2055	-186.6026	-52.0817	-177.2055	90	223
2441	5.2278	5.3260	-43	-62.7337	-5031.3	-159.0922	-192.850	-62.7337	-159.093	100	242
2433	4.3688	0	-94	-481.809	-5065.4	-33.3303	-193.4769	-483.7316	-31.3423	7	580

Partial Features Table

The above is the partial feature table this is how it looks like, these 11 features are the predictors and the 1 response which is the distance. In total, We have 70k such samples in the features table.

Extracting these types of features listed above yields 11 features from the raw data in text files from the dropbox, Some MATLAB programs and

STEP 3: DEVELOP PREDICTIVE MODELS

Developing a predictive model is an iterative process involving these steps:

i. Select the training and validation data.

ii. Select a regression algorithm.

iii. Iteratively train and evaluate regression models.

i. SELECTING THE TRAINING AND VALIDATION DATA

Before training actual classifiers, we need to divide the data into a train- ing and a validation set. The validation set is used to measure accuracy during model development. For large datasets such as the dataset in our case with is more than 70k samples, holding out a certain percentage of the data is appropriate; cross-validation is recommended for smaller datasets because it maxi-mixes how much data is used for model training, and typically results in a model that generalises better.

When you choose "No validation," the model is trained and evaluated on the entire dataset —no data is held out for validation. Retraining the model on the entire dataset can significantly affect its performance, especially if you have a very limited dataset. And knowing how accurately the model performs on the training set gives you guidance on which approaches to take to further improve the model.

II. SELECTING A REGRESSION ALGORITHM

No single machine learning algorithm works for every problem, and identifying the right algorithm is often a process of trial and error. However, being aware of key characteristics of various algorithms empowers you to choose which ones to try first, and to understand the tradeoffs you are making. The following table lists characteristics of popular classification algorithms.

Algorithm	Prediction Speed	Training Speed	Memory Usage	Required Tuning	General Assessment
Logistic Regression (and Linear SVM)	Fast	Fast	Small	Minimal	Good for small problems with linear decision boundaries
Decision Trees	Fast	Fast	Small	Some	Good generalist, but prone to overfitting
(Nonlinear) SVM (and Logistic Regression)	Slow	Slow	Medium	Some	Good for many binary problems, and handles high-dimensional data well
Nearest Neighbor	Moderate	Minimal	Medium	Minimal	Lower accuracy, but easy to use and interpret
Naïve Bayes	Fast	Fast	Medium	Some	Widely used for text, including spam filtering
Ensembles	Moderate	Slow	Varies	Some	High accuracy and good performance for small- to medium-sized datasets
Neural Network	Moderate	Slow	Medium to Large	Lots	Popular for classification, compression, recognition, and forecasting

III. ITERATIVELY TRAINING AND EVALUATING REGRESSION MODELS

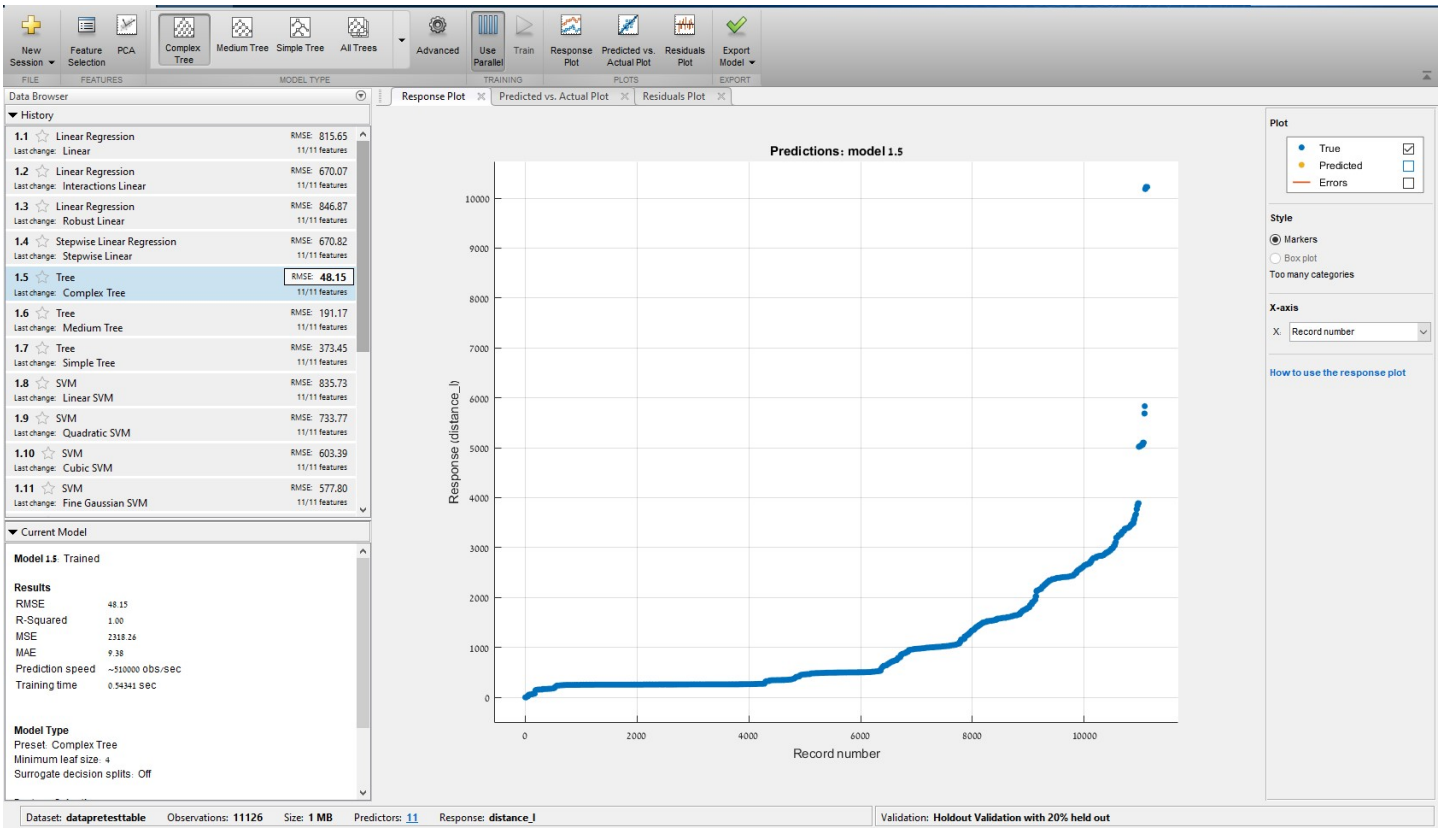
We are now ready to begin the iterative training and evaluation of models. We can either follow the brute force approach and run all the algorithms (this is easy to do with the Regression Learner app), or start with the algorithms that seem to best fit the specific Regression task. You can select an individual classifier or multiple classifiers simultaneously, such as "All Trees." You can then train classifiers in parallel and compare their performance on your data. For each classifier that you train, the accuracy is estimated either on the held-out validation data or using cross validation, depending on which data you selected for validation in the previous step. The initial results suggest that the ("Fine") K-nearest neighbours (KNN) classifier performs well, followed by the quadratic support vector machine (SVM) and (Fine) decision tree.

We have split the 70k samples in the features table into two tables, training set of 10k samples and testing set of 60k samples.

We are going to train 10k samples with hold-out validation of 20%. We are going to select the results whose RMSE value for the validation is minimum.

So we are going to feed the features table in the Regression Learner App in MATLAB 2019A.

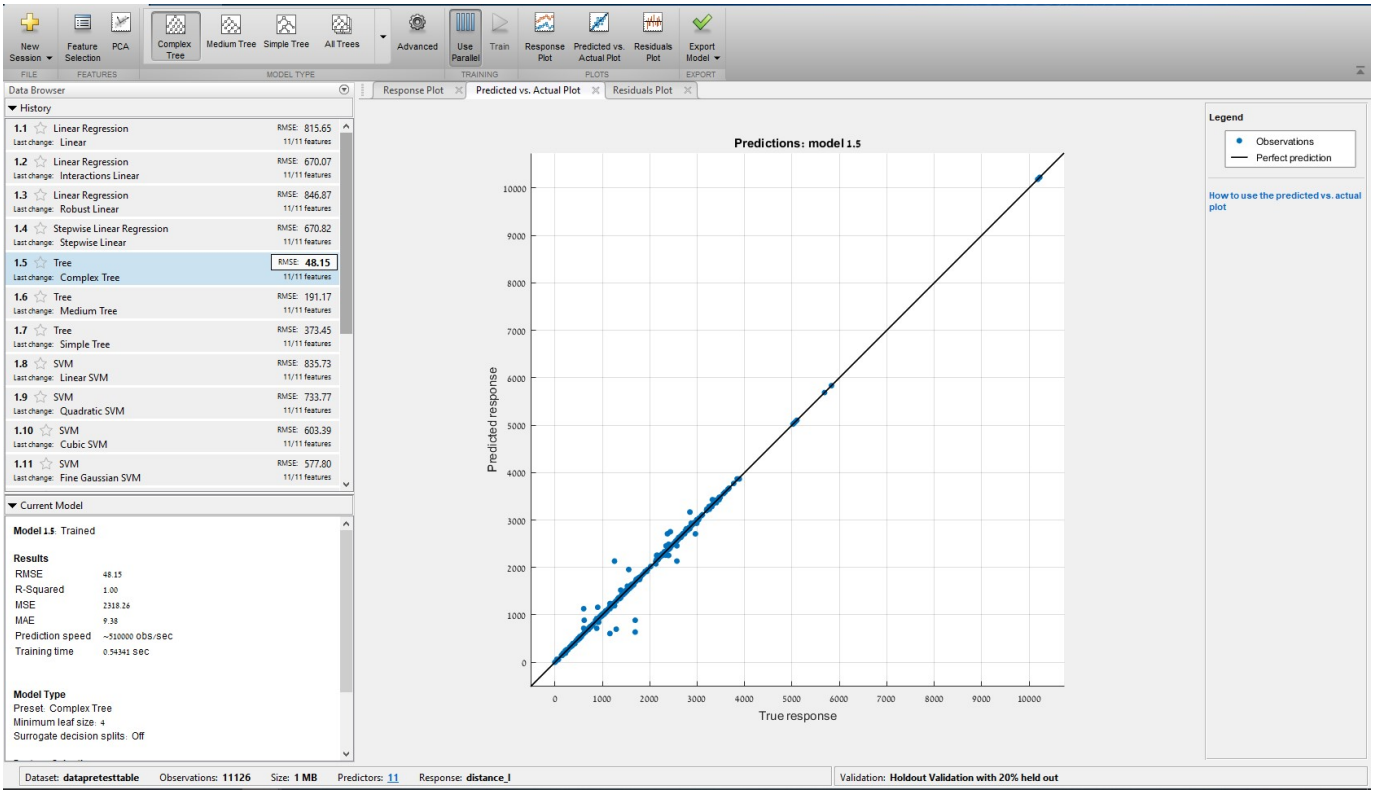
Now we train the model to understand how machine predicts the distance when the inputs are the features of the model



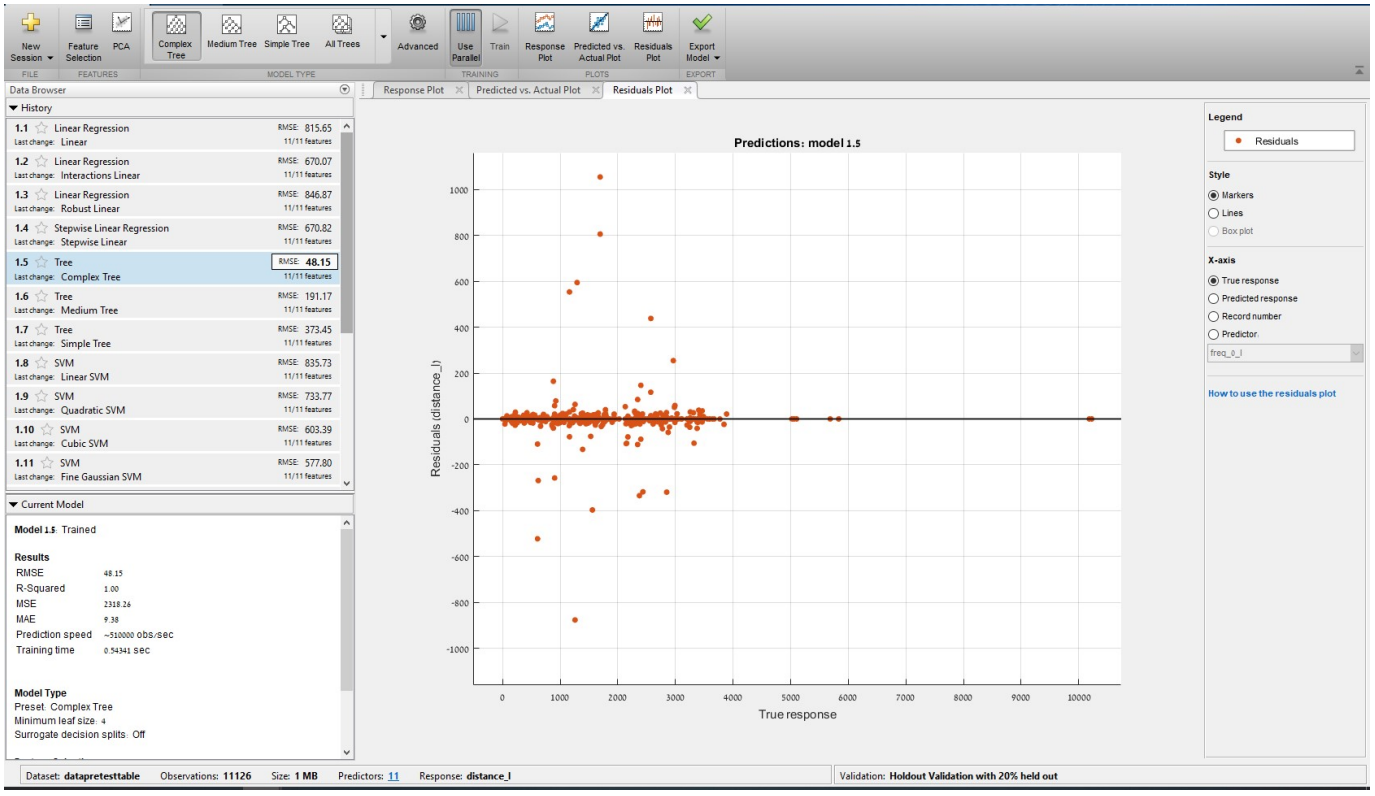
The Training Dataset

This is the distance vs the number of samples feed to the machine with 20% validation

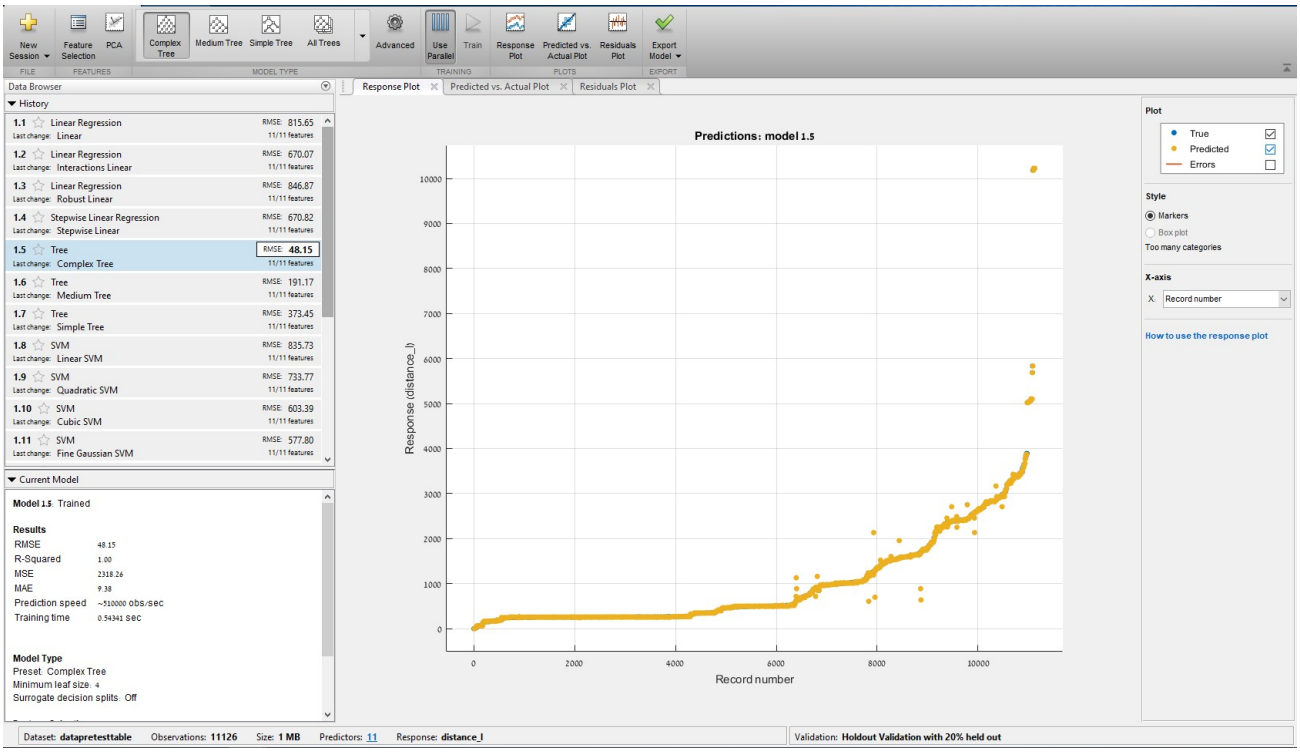
All the plot presented in this section is done of the validation data set.



The Response Plot



Residue Plot

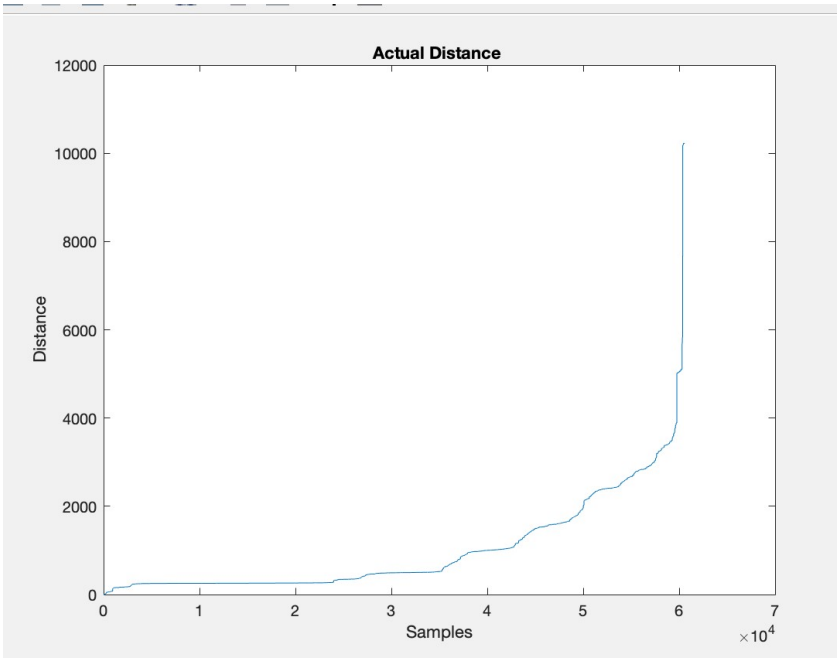


Response Plot

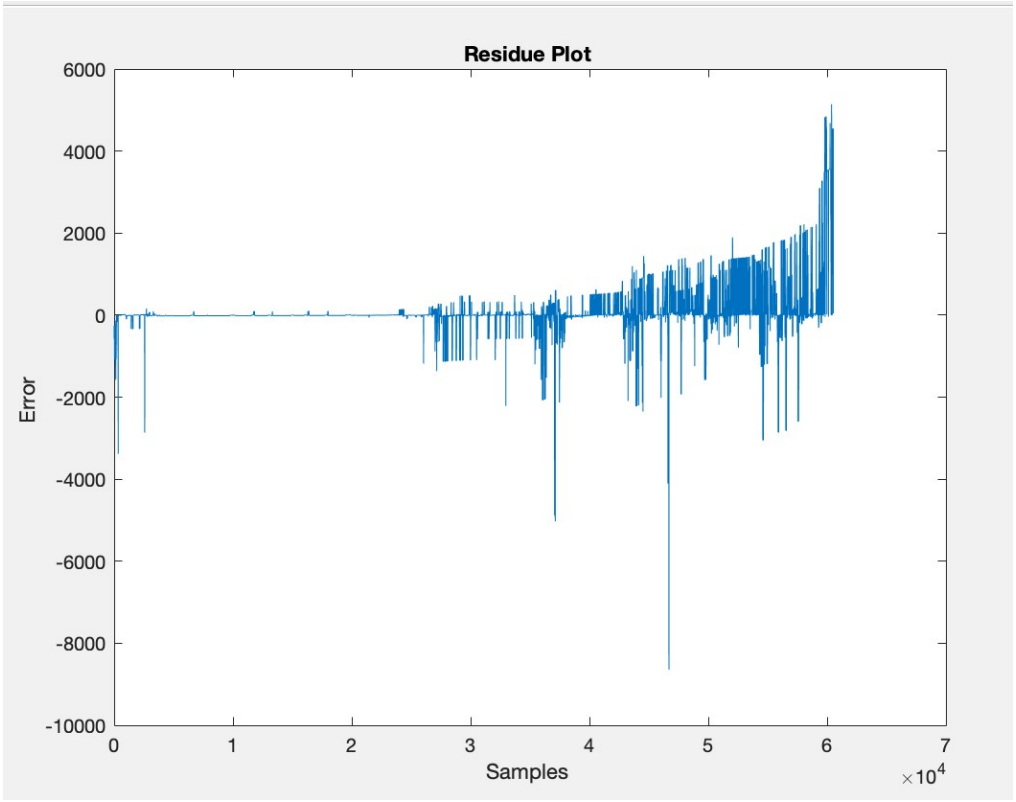
Then we exported the trained model back to the MATLAB and then tested it on the 60k testing data the results of the testing are as follows

Validation RMSE: 48.15 cm	**Validation Success to predict the distance : 99.8%**
Testing RMSE : 671.11 cm	**Testing Success to predict the distance : 80.11%**

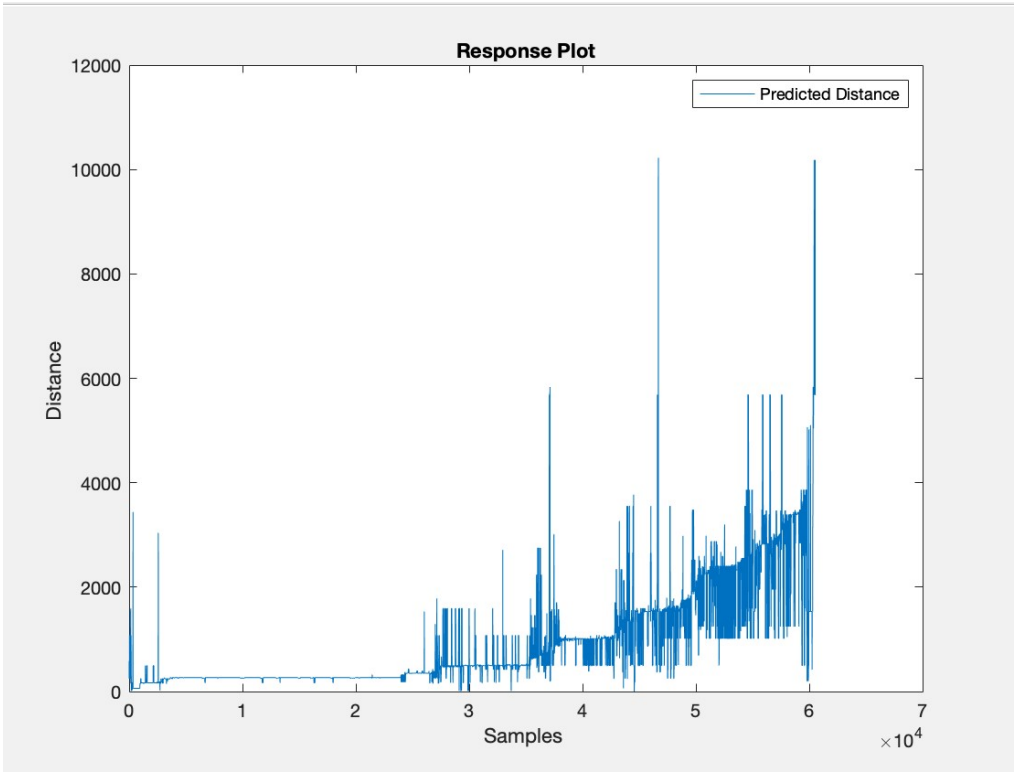
Testing Data Graphs are as follows



Actual Distance



Residue Plot

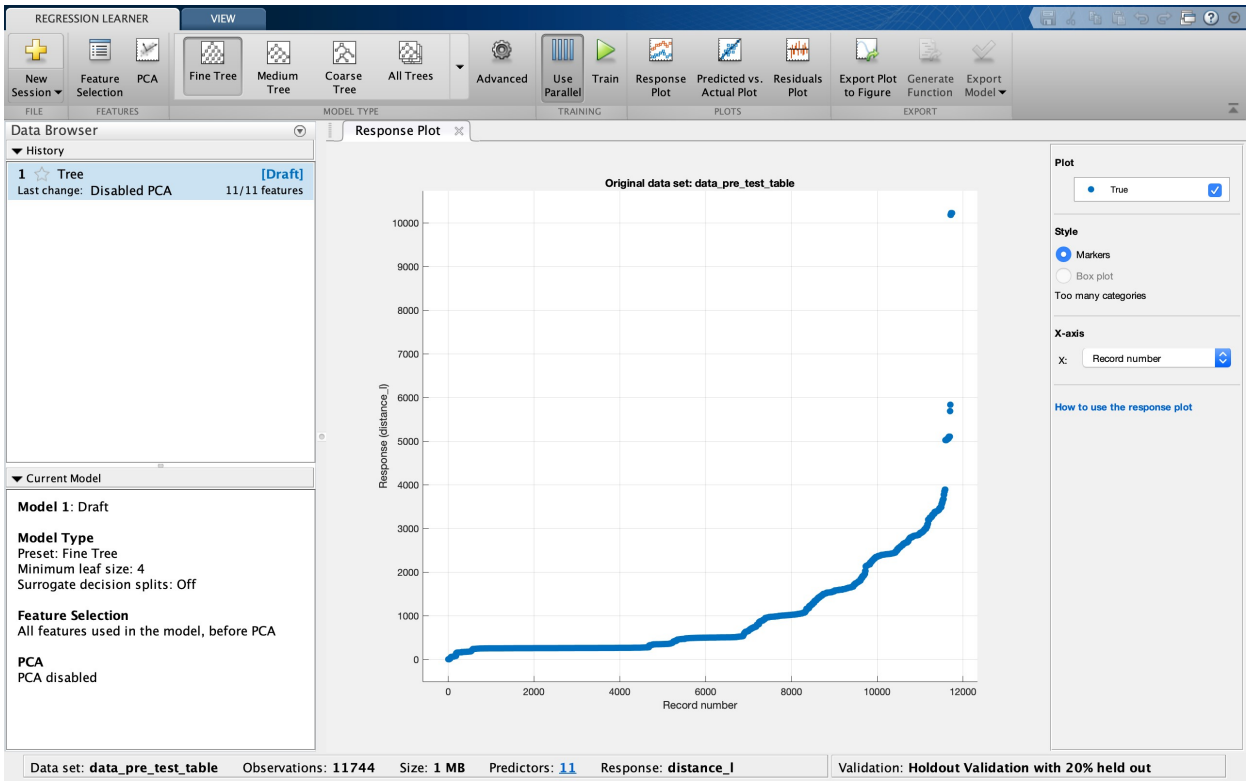


Response Plot

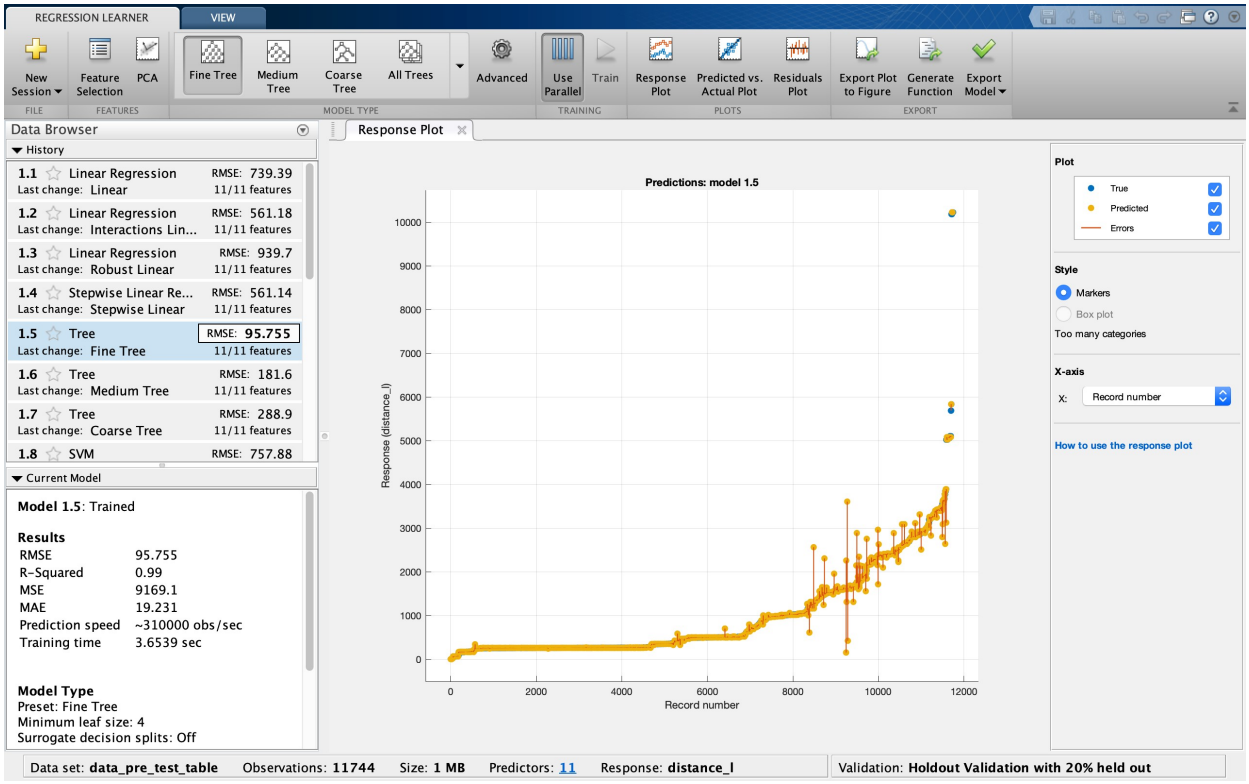
Now Since we found out fine trees are regression model that gives the least RMSE and so to improve the model's accuracy to predict the distance more accurately than the last mode, we decide to split the 70k data set in such a way that we train every 6th samples, i.e 1,7,13,19 and so on…

This 6th sample will be used as training set and rest will be used as testing set for the new model.

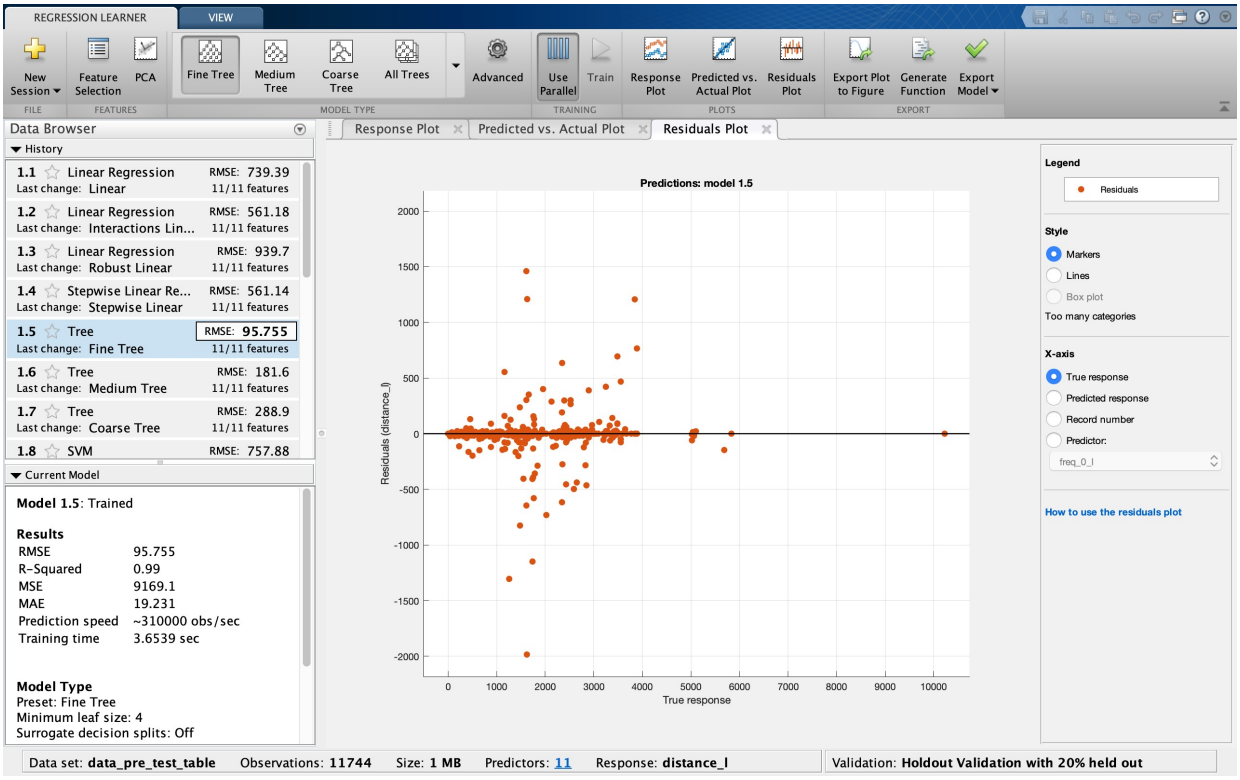
So we feed this new training set to the Regression Learner and the results we as follows:



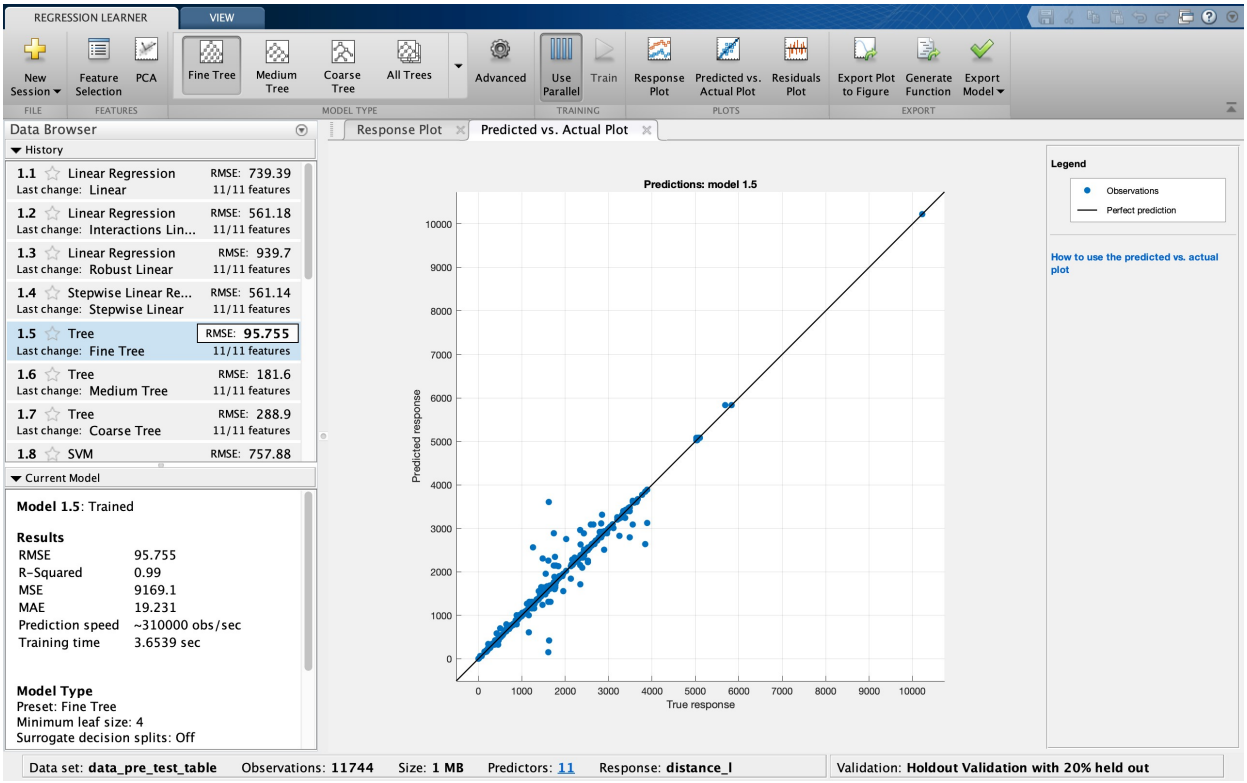
Training Dataset



Response Plot



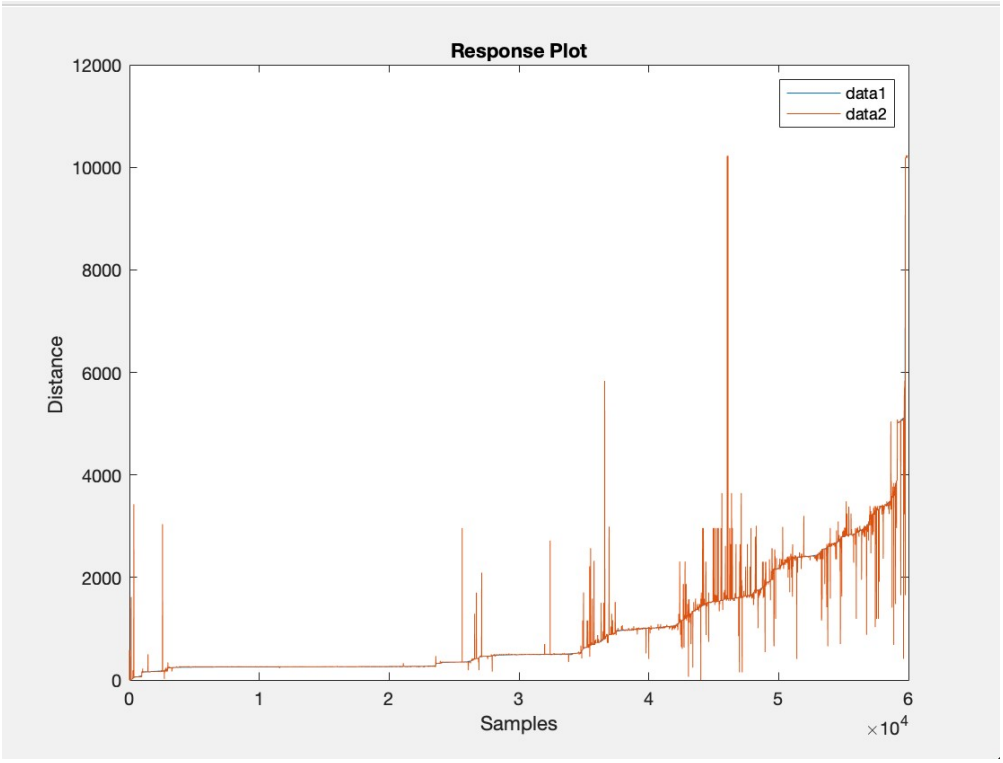
Residue Plot



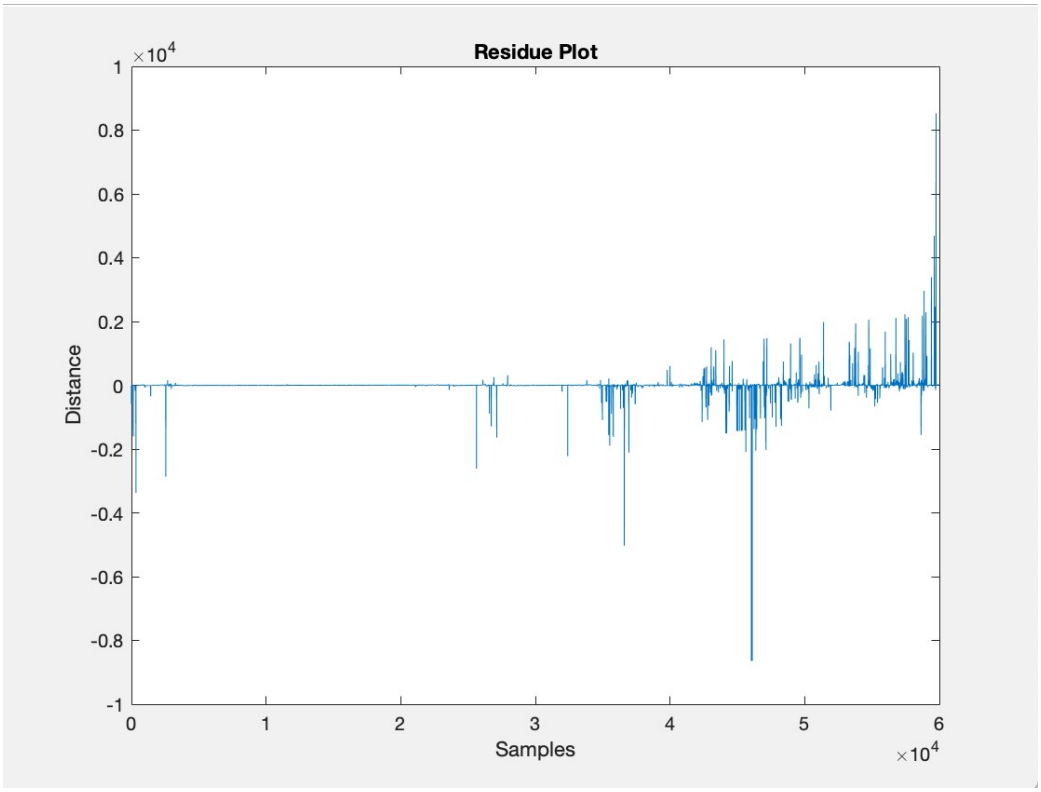
Actual vs Predicted Graph

We then export the model back into the MATLAB work space and then tested it on the 60k testing dataset and the results were shocking but expected to us.

The Testing Graphs are as follows:



Response Plot

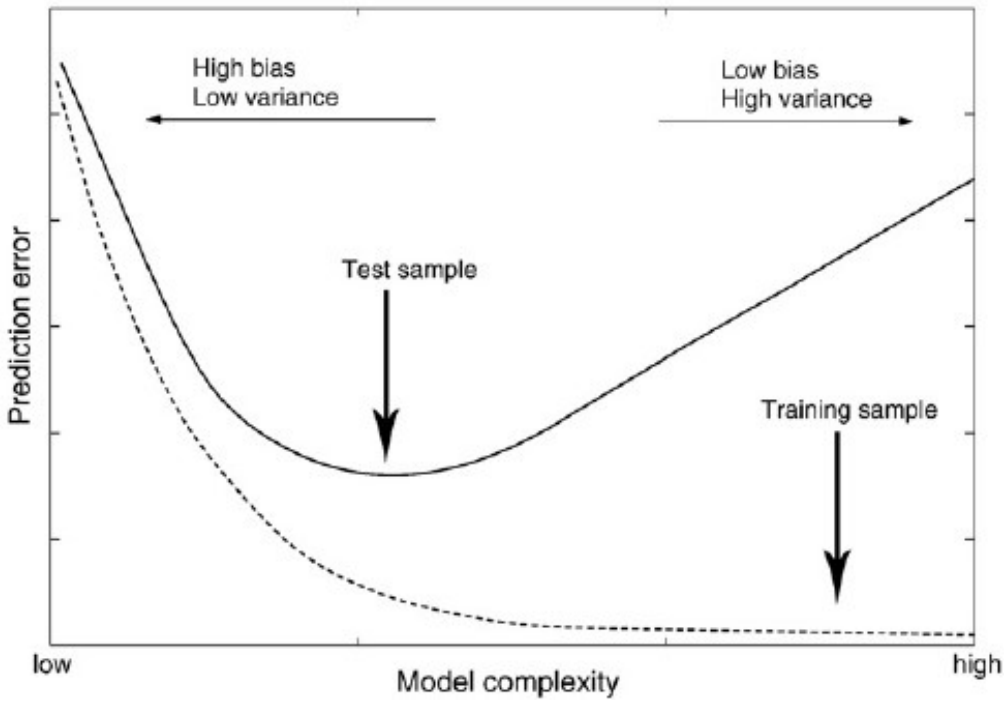


Residue Plot

Validation RMSE: 95.755 cm	**Validation Success to predict the distance : 99.4%**
Testing RMSE : 111.94 cm	**Testing Success to predict the distance : 98.03%**

We likely faced little bit of overfitting in the data that lead to less testing data success,

We just increase the validation error little bit to decrease the testing data error.

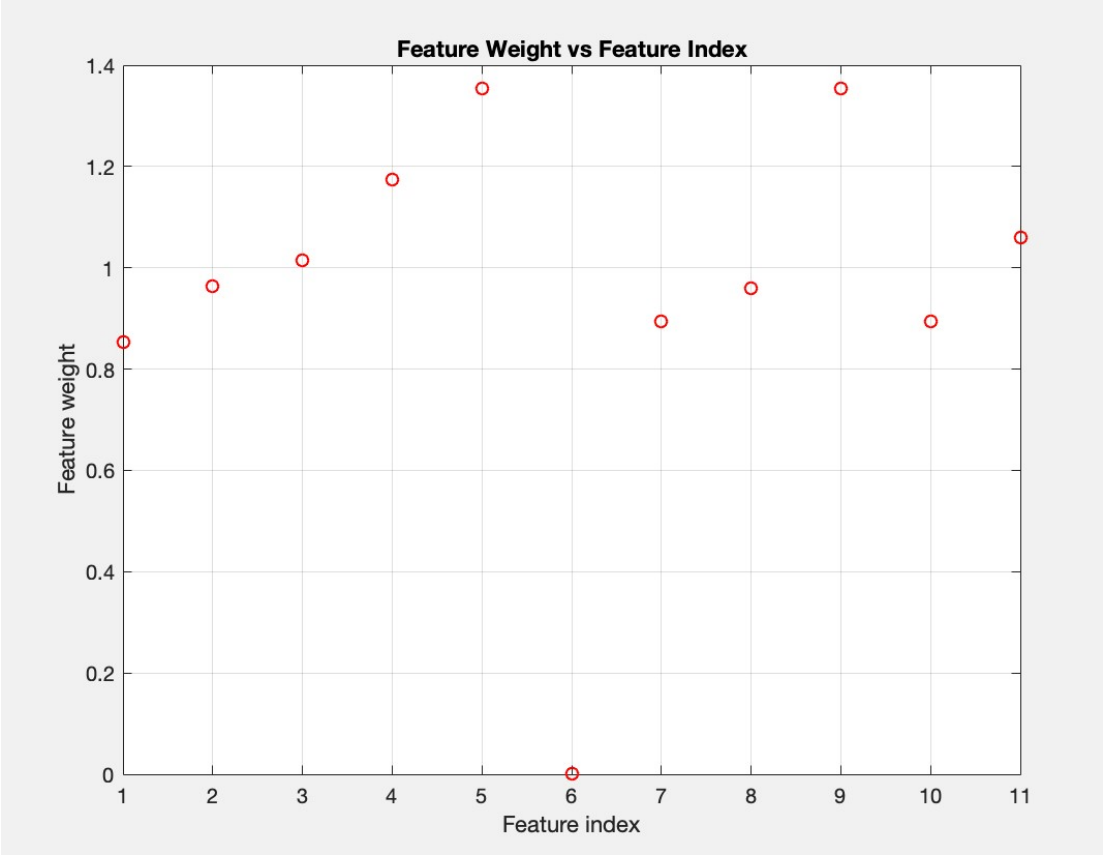


STEP 4: OPTIMISE THE MODEL

i. Using Feature Selection to Correct Misclassification and Overfitting

So far, we've used all 11 features that we extracted when we trained the model. A final approach for optimising performance entails feature selection: removing features that are either redundant or not carrying useful information. This step reduces computational cost and storage requirements, and it results in a simpler model that is less likely to overfit. Reducing the feature set is important for the heart sounds application because it reduces the size of the model, making it easier to deploy on an embedded device.

Feature selection approaches include systematically evaluating feature subsets (which is computationally very expensive) and incorporating feature selection into the model construction process by applying weights to each feature (using fewer features helps minimise the objective function used during training).

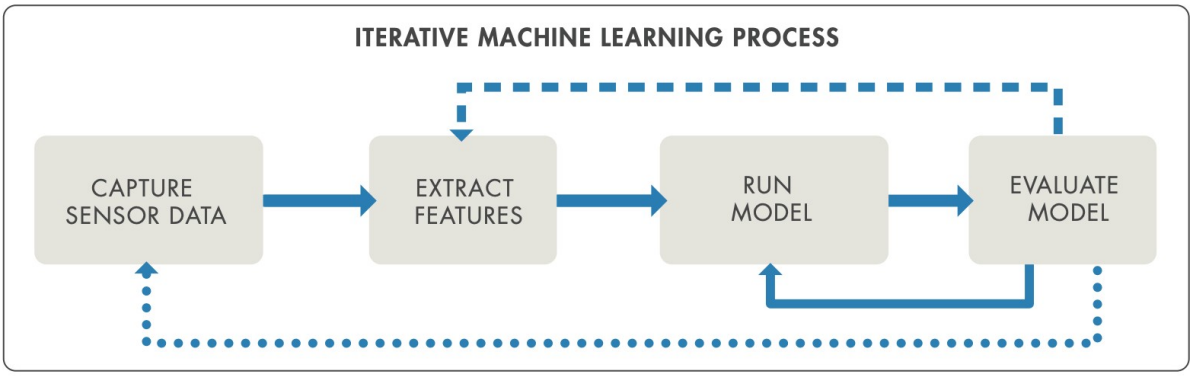


Feature Weight vs Feature Index

For our distance estimation regression model, we apply neighbourhood component analysis (NCA), a powerful feature selection technique that can handle very high-dimensional datasets. NCA reveals that about half the features do not contribute significantly to the model. We can therefore reduce the number of features from 11 to 10.

ii. ITERATING TO TRY OTHER ALGORITHM

To further improve the model, we could try the same series of optimisation steps with different algorithms, since how much improvement the various optimisation techniques yields varies with the algorithm. You can repeat the iterative process described in the previous section—for example, revisiting the Fine tree algorithm, which performed well initially. You might even go back to the feature extraction phase and look for additional features. To identify the best model, it is invariably necessary to iterate between the different phases of the machine learning workflow. Since we have control on the training set we can manipulate the training set in our favour. Thats what we did in step 3



STEP 5: ESSENTIAL TOOLS FOR MACHINE LEARNING

With MATLAB and specialized toolboxes, you can develop, tune, and deploy predictive analytics without extensive knowledge of machine learning or data science. MATLAB and related products offer all the tools you need to build and deploy your analytics:

- Data processing capabilities that simplify time-consuming tasks, including handling missing data or outliers, removing noise, and time-aligning data with different sample rates

- Specialised machine learning apps to speed up your workflow, letting you quickly compare and select algorithms
- Programmatic workflows for removing unnecessary features and fine-tuning model parameters to achieve robust performance
- Tools for scaling the machine learning workflow to big data and compute clusters
- Automatic code generation tools for rapidly deploying your analytics to embedded targets

CONCLUSION

- Well- tune regression algorithm performs excellent in predicting the distance between the transmitting and receiving antenna
- Choosing the dataset for training to increase validation error in order to decrease the testing error was done quite successfully
- This project helped us to explore the opportunity to learn Machine Learning in MATLAB. The project helped us improve our skills in Matlab environment, working with new directories like signal processing toolbox, regression learner app and classification learning app.
- Lastly I would like to thank Omer Tzidki to provide us a wonderful opportunity and helped us in every step of the project and everyone of our friends and family member to stand beside us and morally support us.

THANK YOU!