Naive Bayes Classification

```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.20, random_state = 0)
```

```
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
# Training the Naive Bayes model on the Training set
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
ac = accuracy_score(y_test,y_pred)
cm = confusion_matrix(y_test, y_pred)
```

```
0.7604166666666666
[[ 86   0   0  11   0   0]
 [  0 388  32  80   0   1]
 [  0  29  96   0   0   6]
 [ 71  38   0 289   0   1]
 [  0   0   0   0  31   2]
 [  0   0  21   0   7  59]]
```

```
[ ]  classifier = BernoulliNB()
     classifier.fit(X_train, y_train)
```

```
     BernoulliNB()
```

```
[ ]  # Predicting the Test set results
     y_pred = classifier.predict(X_test)

     # Making the Confusion Matrix
     from sklearn.metrics import confusion_matrix, accuracy_score
     ac = accuracy_score(y_test,y_pred)
     cm = confusion_matrix(y_test, y_pred)
```

```
[ ]  print(ac)
     print(cm)
```

```
     0.6057692307692307
     [[  0   1   0  96   0   0]
      [  0 304  55 120   0  22]
      [  0  35  29   1   0  66]
      [  0  35   0 364   0   0]
      [  0   2   3   0   0  28]
      [  0   4  24   0   0  59]]
```

Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

Bayes' Theorem

Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes' theorem is stated mathematically as the following equation:

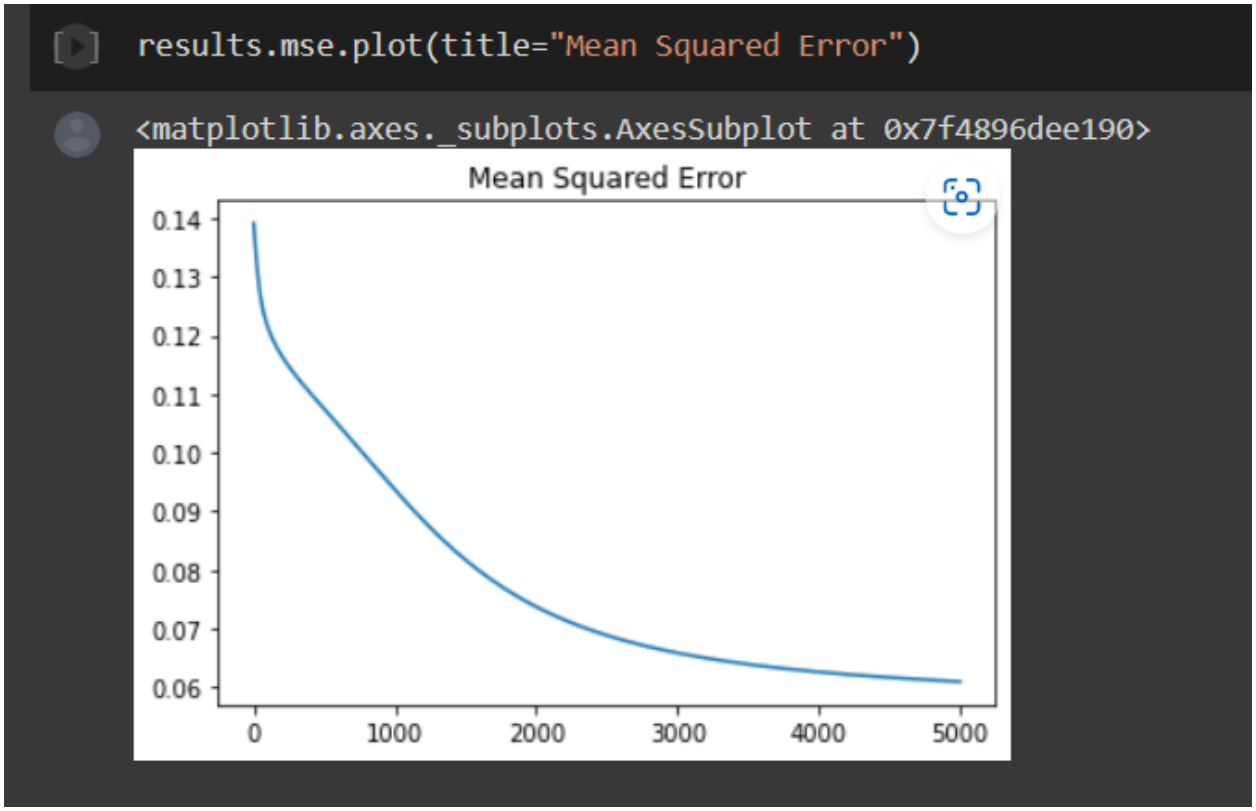
P(A|B) = {P(B|A) P(A)}/{P(B)}

where A and B are events and P(B) ≠ 0.

Basically, we are trying to find probability of event A, given the event B is true. Event B is also termed as evidence.
P(A) is the priori of A (the prior probability, i.e. Probability of event before evidence is seen). The evidence is an attribute value of an unknown instance(here, it is event B).
P(A|B) is a posteriori probability of B, i.e. probability of event after evidence is seen.
Now, with regards to our dataset, we can apply Bayes' theorem in following way:

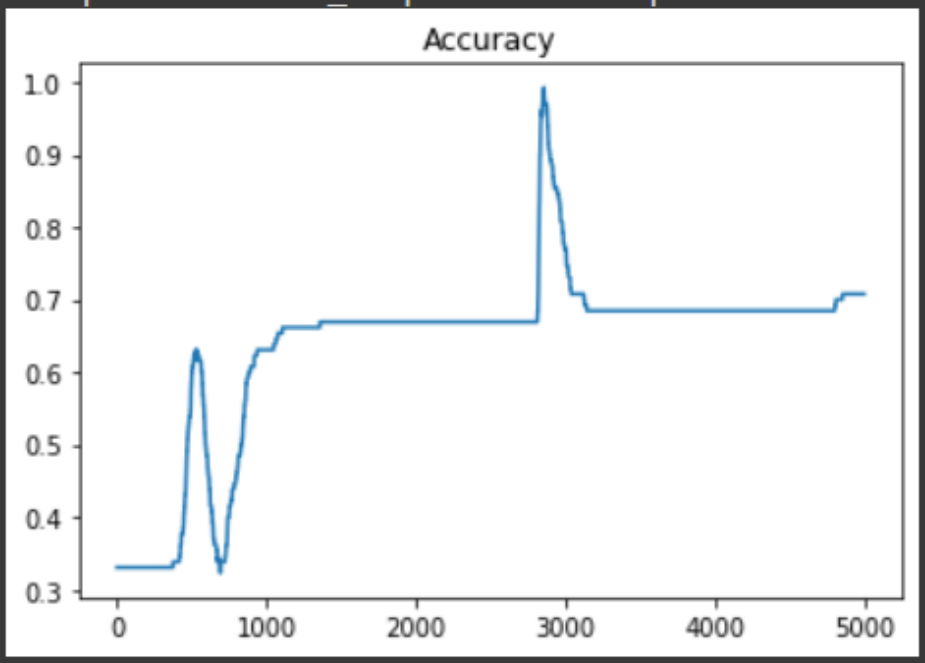
P(y|X) = {P(X|y) P(y)}/{P(X)}

BPN -Back Propagation in ML

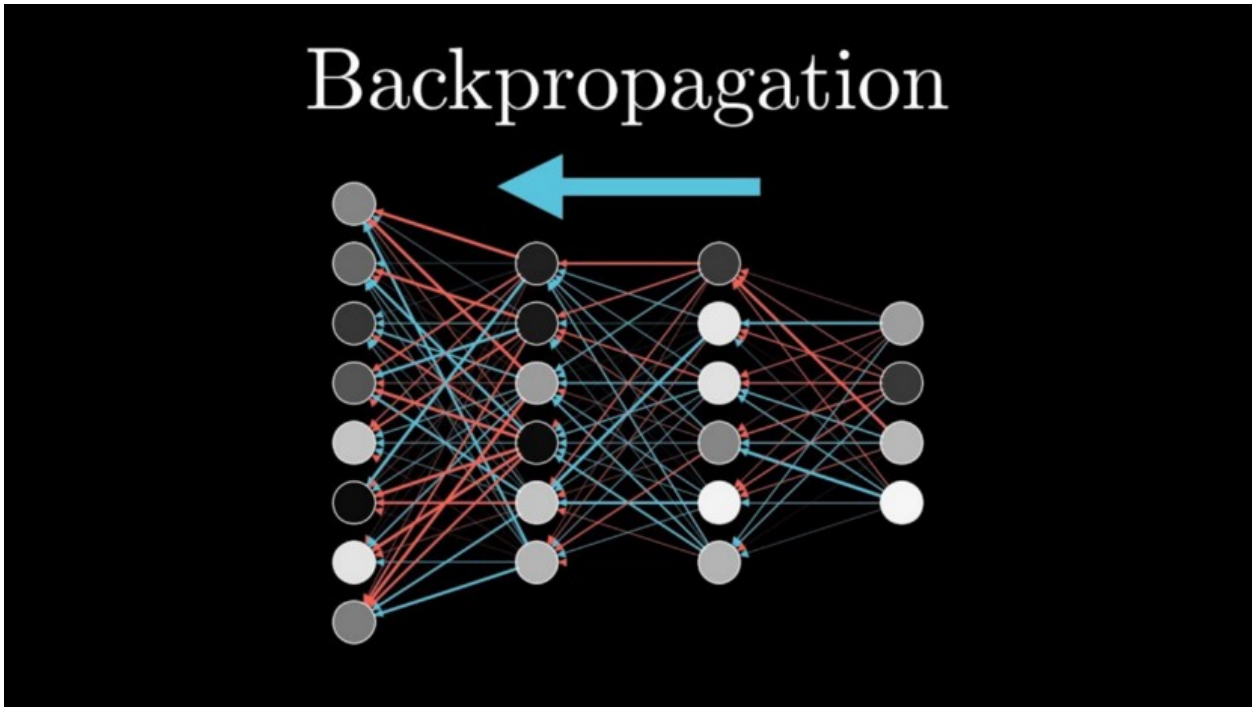


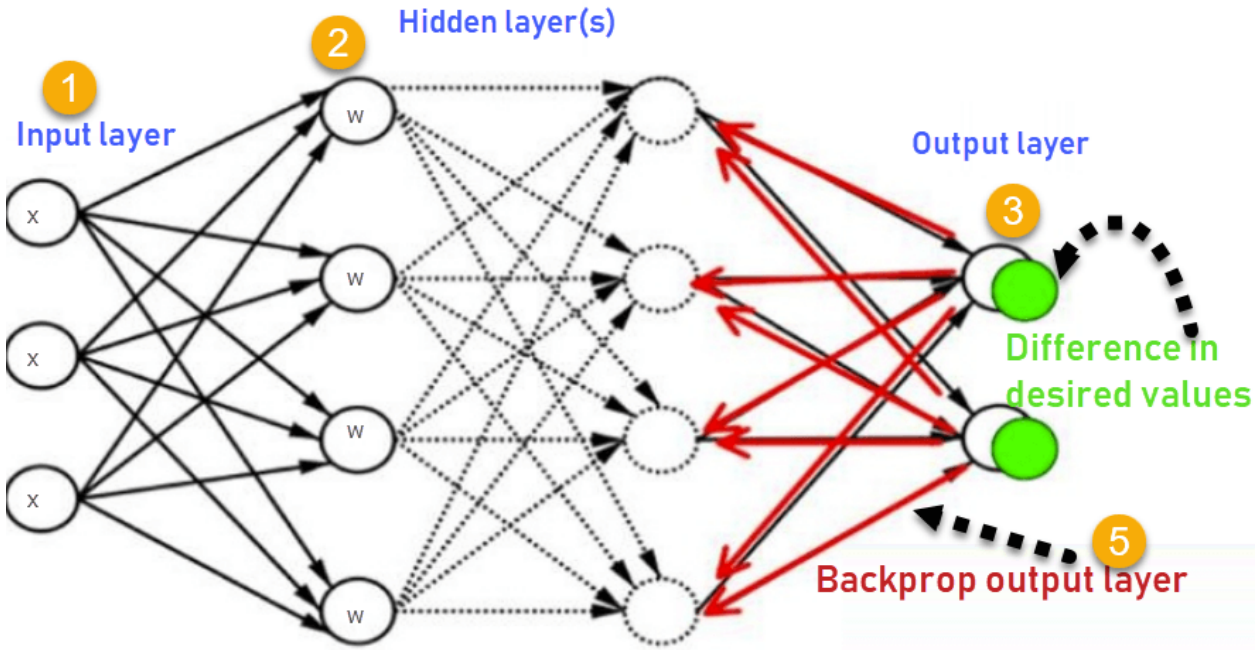
```
[ ]  results.accuracy.plot(title="Accuracy")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4897461390>
```




```
Accuracy: 0.8
```





Backpropagation is the essence of neural network training. It is the method of fine-tuning the weights of a neural network based on the error rate obtained in the previous epoch (i.e., iteration). Proper tuning of the weights allows you to reduce error rates and make the model reliable by increasing its generalization.
Backpropagation in neural network is a short form for “backward propagation of errors.” It is a standard method of training artificial neural networks. This method helps calculate the gradient of a loss function with respect to all the weights in the network.

Types of Backpropagation Networks

Two Types of Backpropagation Networks are:

- Static Back-propagation
- Recurrent Backpropagation

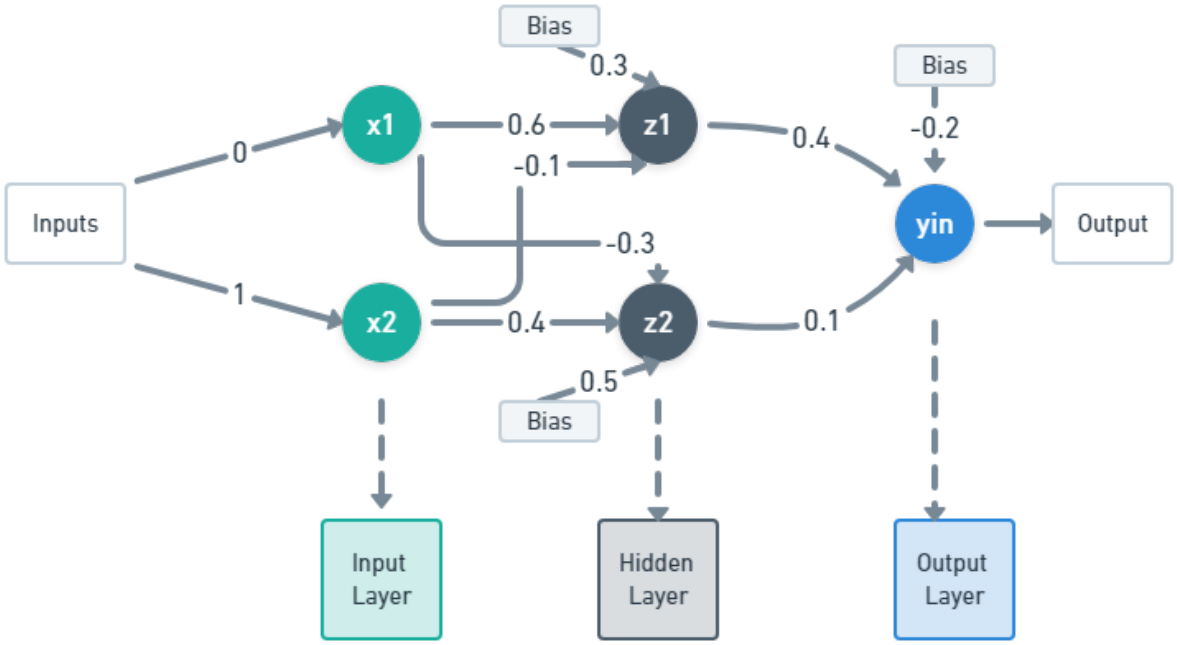
Static back-propagation:

It is one kind of backpropagation network which produces a mapping of a static input for static output. It is useful to solve static classification issues like optical character recognition.

Recurrent Backpropagation:

Recurrent Back propagation in data mining is fed forward until a fixed value is achieved. After that, the error is computed and propagated backward.
The main difference between both of these methods is: that the mapping is rapid in static back-propagation while it is nonstatic in recurrent backpropagation.

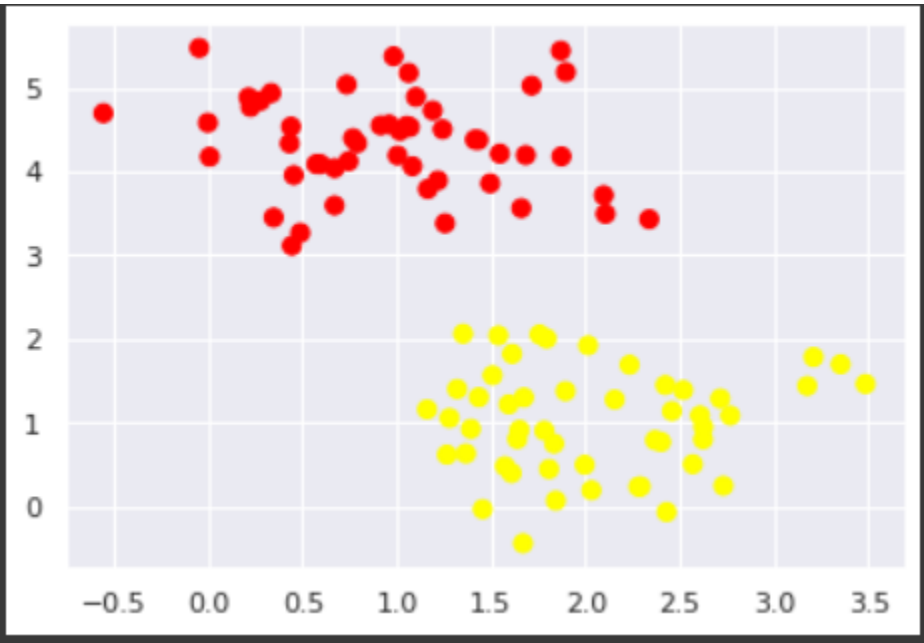
Multi layer feed forward neural network




```
Iteration: 500. Loss: 0.6242267489433289. Accuracy: 86.23999786376953
Iteration: 1000. Loss: 0.42309409379959106. Accuracy: 89.55999755859375
Iteration: 1500. Loss: 0.25730085372924805. Accuracy: 90.61000061035156
Iteration: 2000. Loss: 0.3684542179107666. Accuracy: 91.30000305175781
Iteration: 2500. Loss: 0.37141624093055725. Accuracy: 91.68000030517578
Iteration: 3000. Loss: 0.2659453749656677. Accuracy: 92.02999877929688
```

SVM

```
[ ]  from sklearn.datasets import make_blobs
     X,y= make_blobs(n_samples=100,centers=2,random_state=0,cluster_std=0.60)
     plt.scatter(X[:,0],X[:,1],c=y,s=50,cmap="autumn");
     print(X,y)
```

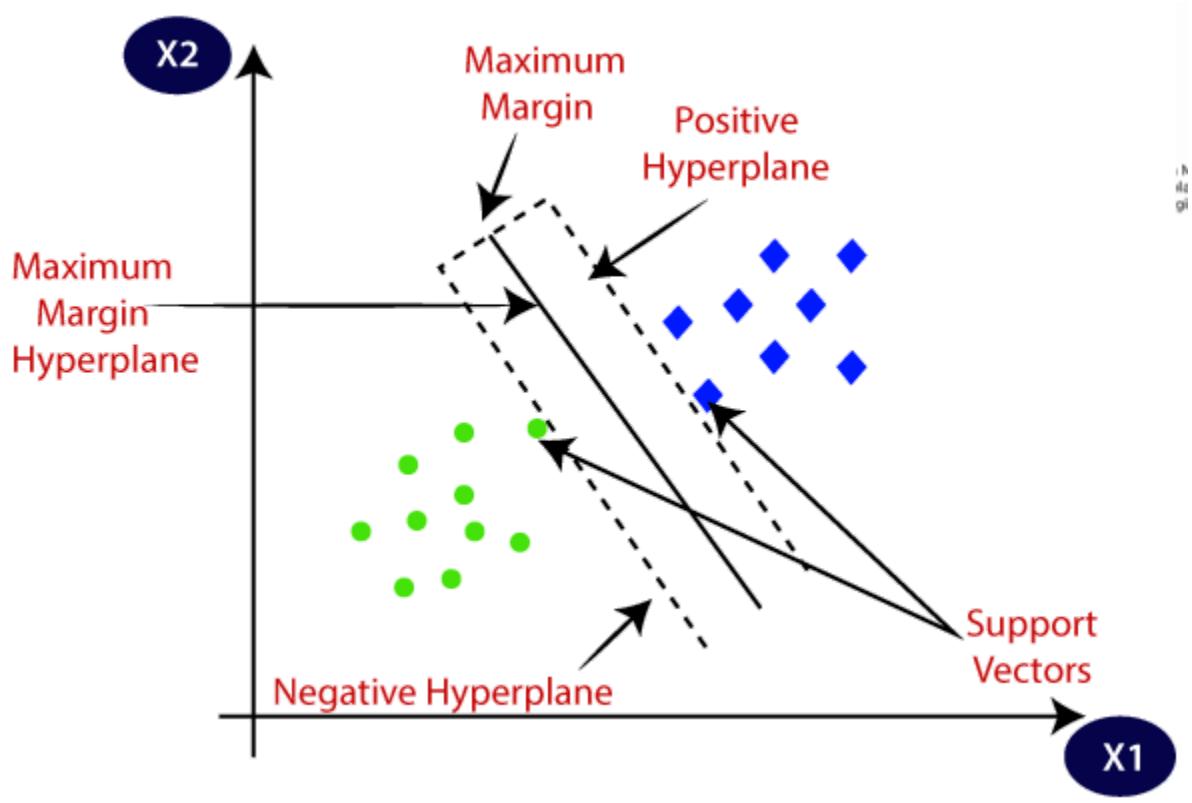



```
[[11  0]
 [ 0  9]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      1.00      1.00         9

    accuracy                           1.00        20
   macro avg       1.00      1.00      1.00        20
weighted avg       1.00      1.00      1.00        20
```

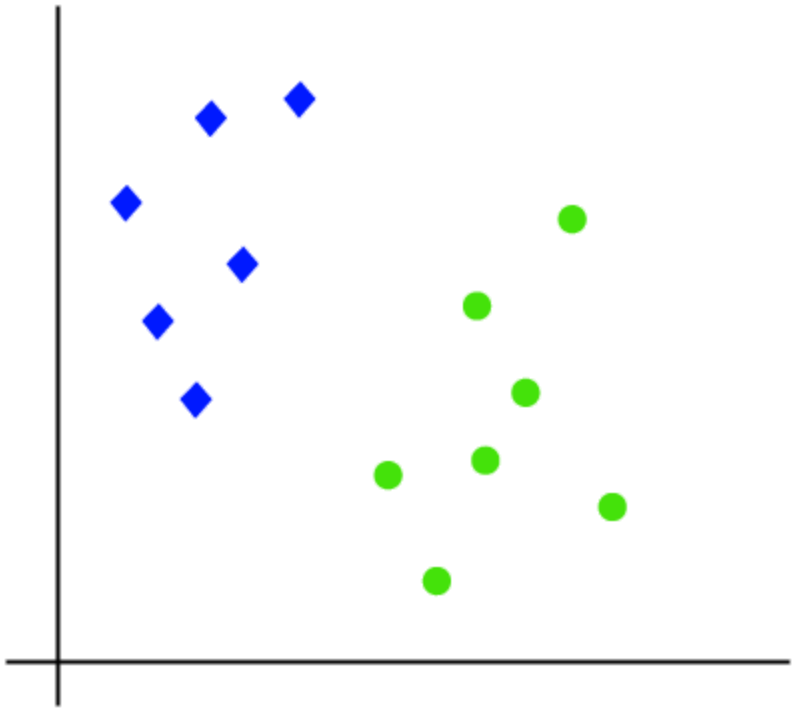
HYPER PARAMETERS

Hyperparameters Hyperparameters are parameters whose values control the learning process and determine the values of model parameters that a learning algorithm ends up learning. The prefix 'hyper_' suggests that they are 'top-level' parameters that control the learning process and the model parameters that result from it.

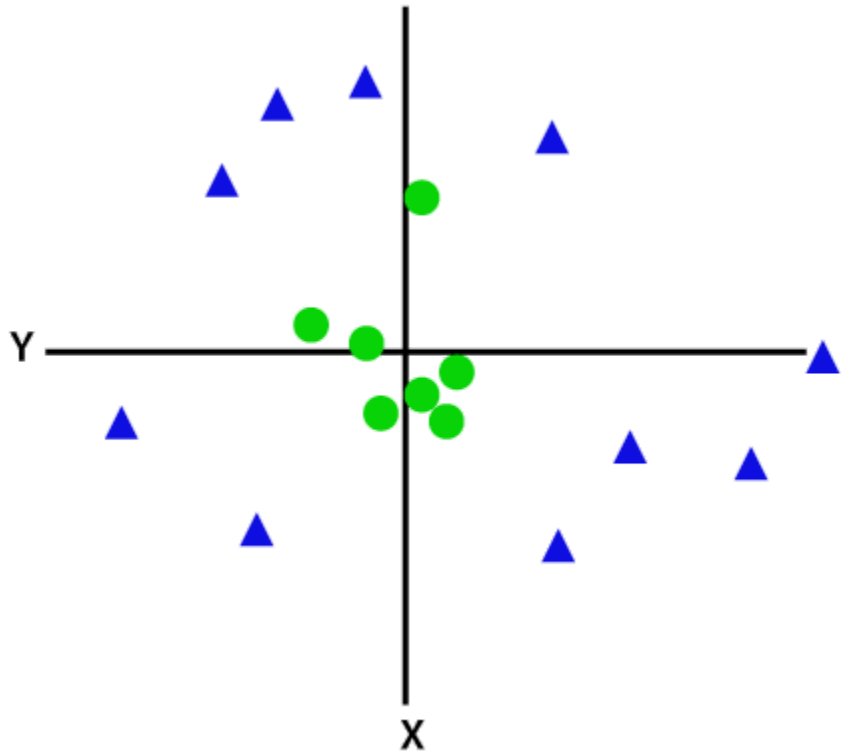
As a machine learning engineer designing a model, you choose and set hyperparameter values that your learning algorithm will use before the training of the model even begins. In this light, hyperparameters are said to be external to the model because the model cannot change its values during learning/training.



Linera svm



Non linear svm



Logistic Regression

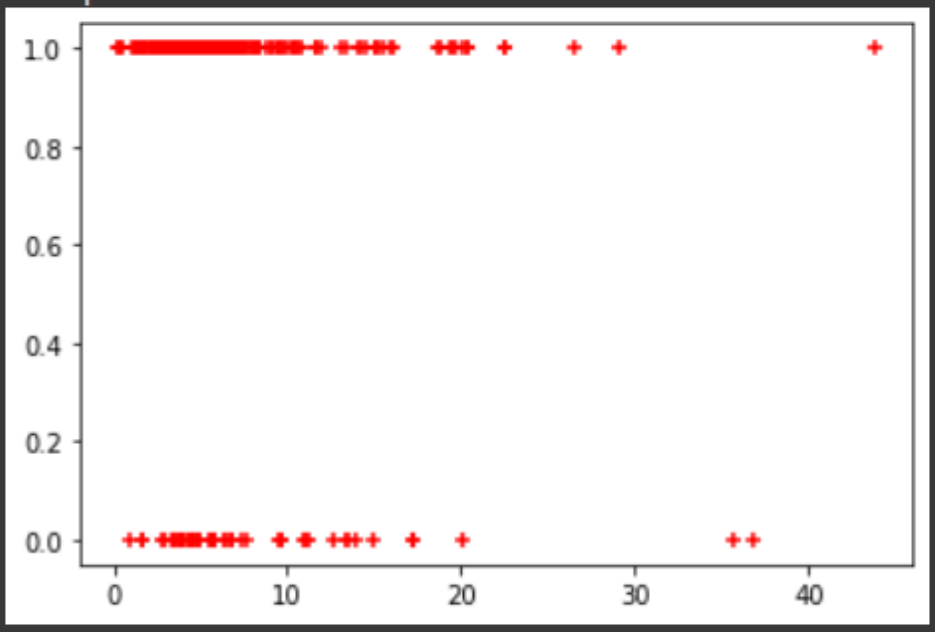
This type of statistical model (also known as *logit model*) is often used for classification and predictive analytics. Logistic regression estimates the probability of an event occurring, such as voted or didn't vote, based on a given dataset of independent variables. Since the outcome is a probability, the dependent variable is bounded between 0 and 1. In logistic regression, a logit transformation is applied on the odds—that is, the probability of success divided by the probability of failure. This is also commonly known as the log odds, or the natural logarithm of odds, and this logistic function is represented by the following formulas:

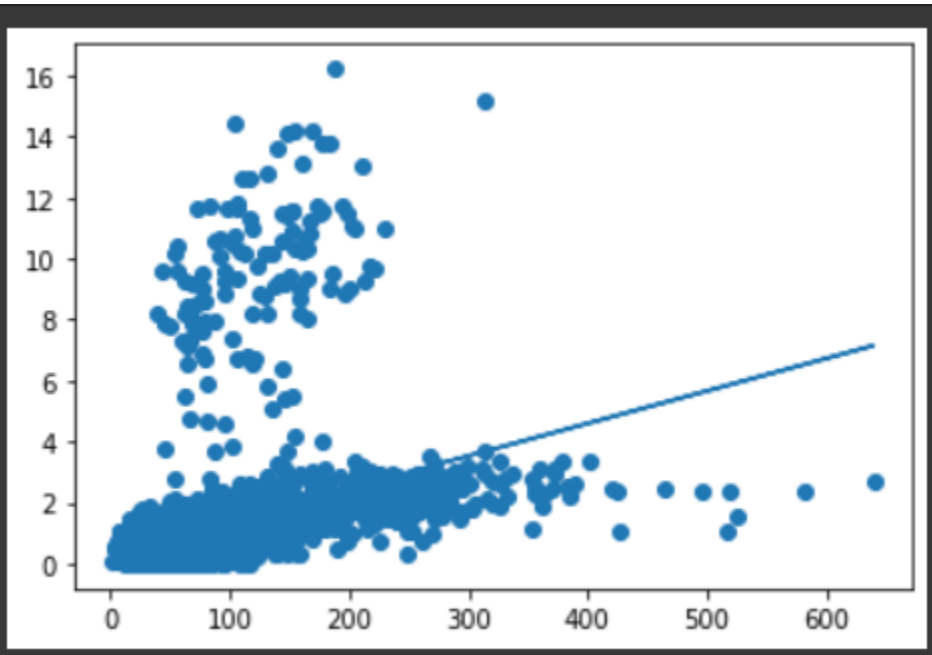
Logit(pi) = 1/(1+ exp(-pi))

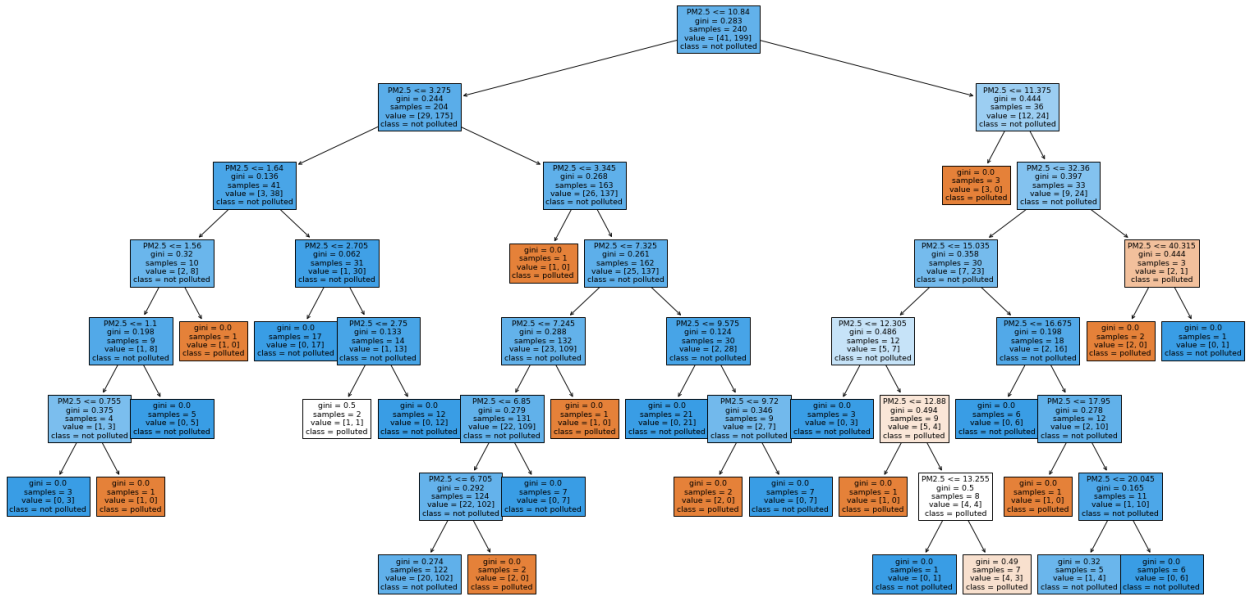
ln(pi/(1-pi)) = Beta_0 + Beta_1*X_1 + … + B_k*K_k

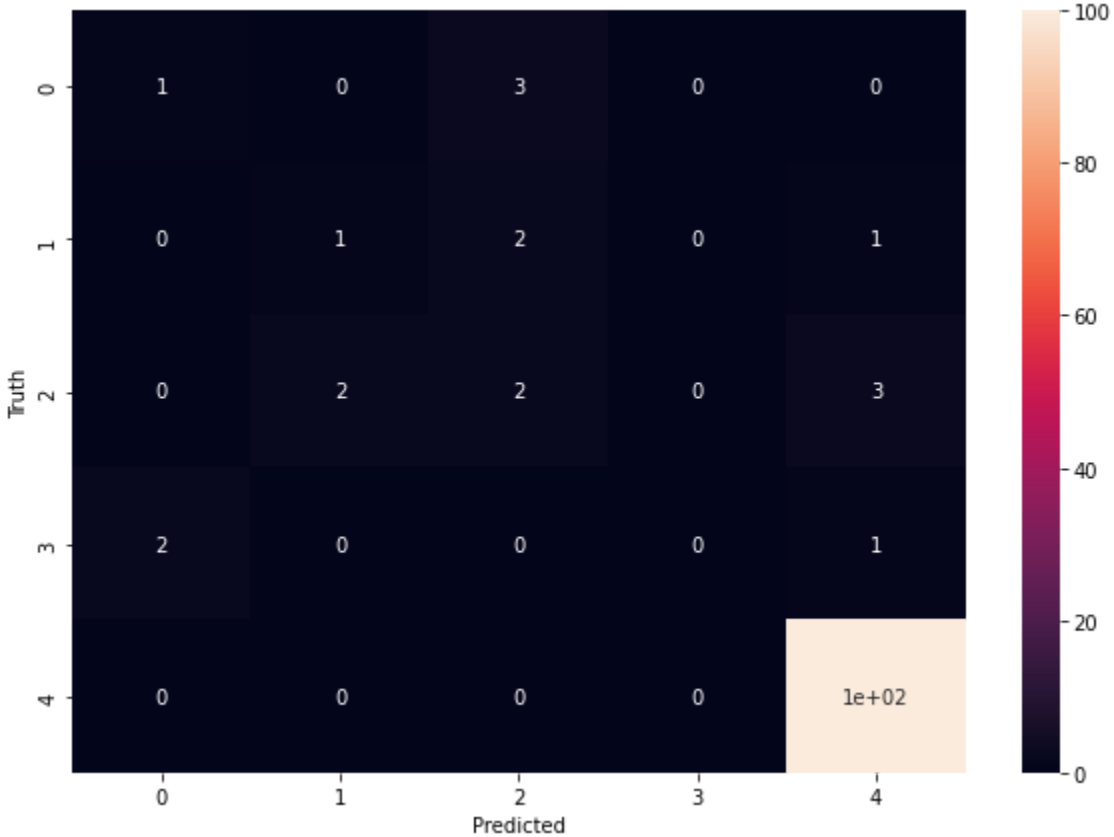
In this logistic regression equation, logit(pi) is the dependent or response variable and x is the independent variable. The beta parameter, or coefficient, in this model is commonly estimated via maximum likelihood estimation (MLE). This method tests different values of beta through multiple iterations to optimize for the best fit of log odds. All of these iterations produce the log likelihood function, and logistic regression seeks to maximize this function to find the best parameter estimate. Once the optimal coefficient (or coefficients if there is more than one independent variable) is found, the conditional probabilities for each observation can be calculated, logged, and summed together to yield a predicted probability. For binary classification, a probability less than .5 will predict 0 while a probability greater than 0 will predict 1. After the model has been computed, it's best practice to evaluate the how well the model predicts the dependent variable, which is called goodness of fit. The Hosmer–Lemeshow test is a popular method to assess model fit.

```
<matplotlib.collections.PathCollection at 0x7fa8447f5c10>
```






PM2.5 <= 10.84
gini = 0.283
samples = 240
value = [41, 199]
class = not polluted
PM2.5 <= 3.275
gini = 0.244
samples = 204
value = [29, 175]
class = not polluted
PM2.5 <= 11.375
gini = 0.444
samples = 36
value = [12, 24]
class = not polluted
PM2.5 <= 1.64
gini = 0.136
samples = 41
value = [3, 38]
class = not polluted
PM2.5 <= 3.345
gini = 0.268
samples = 163
value = [26, 137]
class = not polluted
gini = 0.0
samples = 3
value = [3, 0]
class = polluted
PM2.5 <= 32.36
gini = 0.397
samples = 33
value = [9, 24]
class = not polluted
PM2.5 <= 1.56
gini = 0.32
samples = 10
value = [2, 8]
class = not polluted
PM2.5 <= 2.705
gini = 0.062
samples = 31
value = [1, 30]
class = not polluted
gini = 0.0
samples = 1
value = [1, 0]
class = polluted
PM2.5 <= 7.325
gini = 0.261
samples = 162
value = [25, 137]
class = not polluted
PM2.5 <= 15.035
gini = 0.358
samples = 30
value = [7, 23]
class = not polluted
PM2.5 <= 40.315
gini = 0.444
samples = 3
value = [2, 1]
class = polluted
PM2.5 <= 1.1
gini = 0.198
samples = 9
value = [1, 8]
class = not polluted
gini = 0.0
samples = 1
value = [1, 0]
class = polluted
gini = 0.0
samples = 17
value = [0, 17]
class = not polluted
PM2.5 <= 2.75
gini = 0.133
samples = 14
value = [1, 13]
class = not polluted
PM2.5 <= 7.245
gini = 0.288
samples = 132
value = [23, 109]
class = not polluted
PM2.5 <= 9.575
gini = 0.124
samples = 30
value = [2, 28]
class = not polluted
PM2.5 <= 12.305
gini = 0.486
samples = 12
value = [5, 7]
class = not polluted
PM2.5 <= 16.675
gini = 0.198
samples = 18
value = [2, 16]
class = not polluted
gini = 0.0
samples = 2
value = [2, 0]
class = polluted
gini = 0.0
samples = 1
value = [0, 1]
class = not polluted
PM2.5 <= 0.755
gini = 0.375
samples = 4
value = [1, 3]
class = not polluted
gini = 0.0
samples = 5
value = [0, 5]
class = not polluted
gini = 0.5
samples = 2
value = [1, 1]
class = polluted
gini = 0.0
samples = 12
value = [0, 12]
class = not polluted
PM2.5 <= 6.85
gini = 0.279
samples = 131
value = [22, 109]
class = not polluted
gini = 0.0
samples = 1
value = [1, 0]
class = polluted
gini = 0.0
samples = 21
value = [0, 21]
class = not polluted
PM2.5 <= 9.72
gini = 0.346
samples = 9
value = [2, 7]
class = not polluted
gini = 0.0
samples = 3
value = [0, 3]
class = not polluted
PM2.5 <= 12.88
gini = 0.494
samples = 9
value = [5, 4]
class = polluted
gini = 0.0
samples = 6
value = [0, 6]
class = not polluted
PM2.5 <= 17.95
gini = 0.278
samples = 12
value = [2, 10]
class = not polluted
gini = 0.0
samples = 3
value = [0, 3]
class = not polluted
gini = 0.0
samples = 1
value = [1, 0]
class = polluted
PM2.5 <= 6.705
gini = 0.292
samples = 124
value = [22, 102]
class = not polluted
gini = 0.0
samples = 7
value = [0, 7]
class = not polluted
gini = 0.0
samples = 2
value = [2, 0]
class = polluted
gini = 0.0
samples = 7
value = [0, 7]
class = not polluted
gini = 0.0
samples = 1
value = [1, 0]
class = polluted
PM2.5 <= 13.255
gini = 0.5
samples = 8
value = [4, 4]
class = polluted
gini = 0.0
samples = 1
value = [1, 0]
class = polluted
PM2.5 <= 20.045
gini = 0.165
samples = 11
value = [1, 10]
class = not polluted
gini = 0.274
samples = 122
value = [20, 102]
class = not polluted
gini = 0.0
samples = 2
value = [2, 0]
class = polluted
gini = 0.0
samples = 1
value = [0, 1]
class = not polluted
gini = 0.49
samples = 7
value = [4, 3]
class = polluted
gini = 0.32
samples = 5
value = [1, 4]
class = not polluted
gini = 0.0
samples = 6
value = [0, 6]
class = not polluted


Truth
Predicted
0
1
2
3
4
1
0
3
0
0
0
1
2
0
1
0
2
2
0
3
2
0
0
0
1
0
0
0
0
1e+02
100
80
60
40
20
0

Decision Tree

A tree has many analogies in real life, and turns out that it has influenced a wide area of machine learning, covering both classification and regression. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions. Though a commonly used tool in data mining for deriving a strategy to reach a particular goal, its also widely used in machine learning, which will be the main focus of this article.

How can an algorithm be represented as a tree?

For this let's consider a very basic example that uses titanic data set for predicting whether a passenger will survive or not. Below model uses 3 features/attributes/columns from the data set, namely sex, age and sibsp (number of spouses or children along).

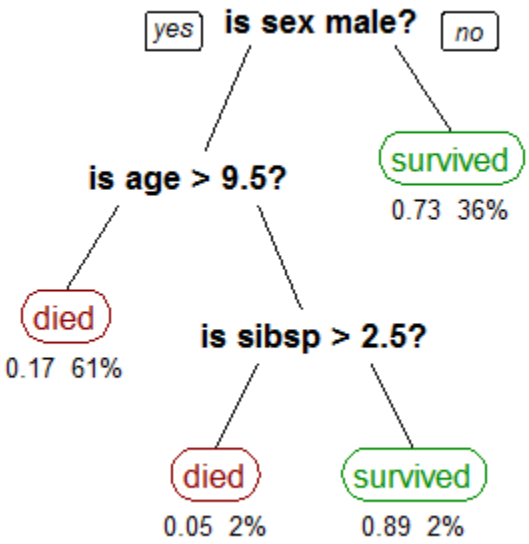
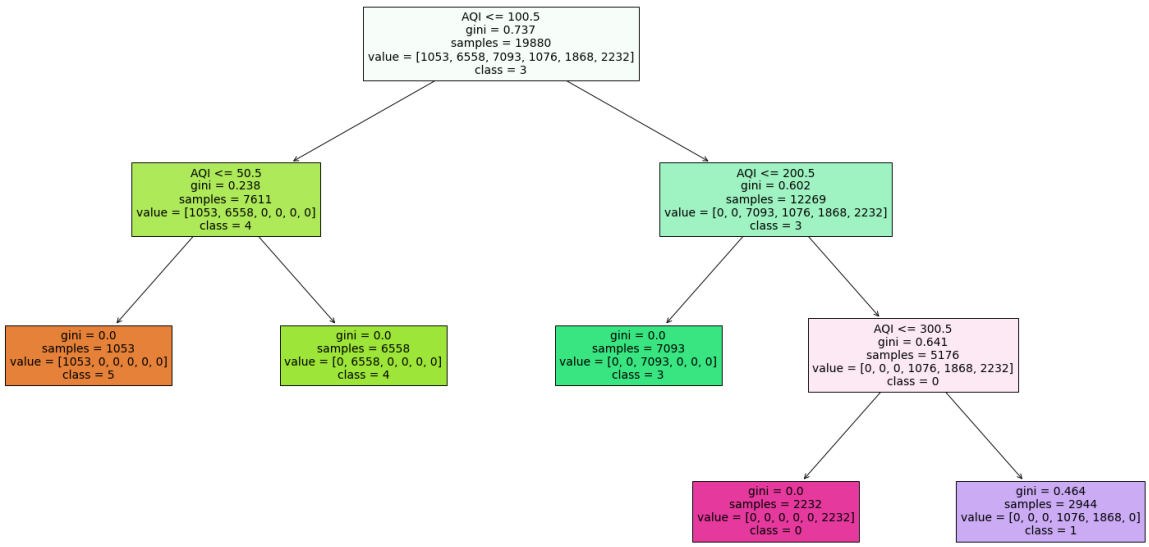


Image taken from wikipedia

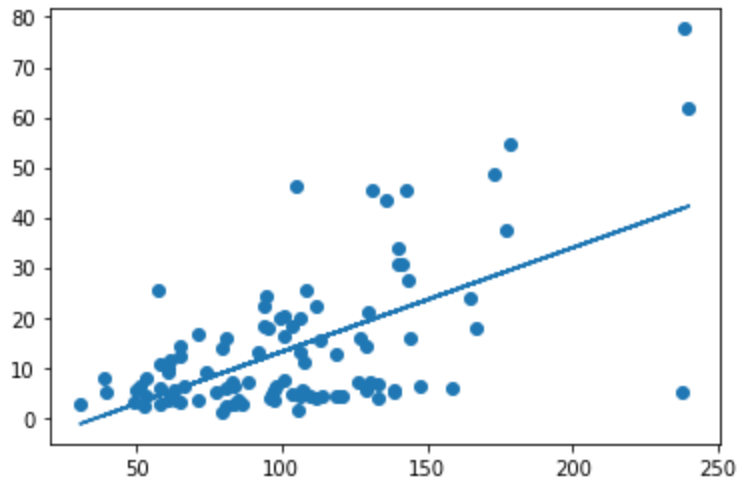
A decision tree is drawn upside down with its root at the top. In the image on the left, the bold text in black represents a condition/internal node, based on which the tree splits into branches/ edges. The end of the branch that doesn't split anymore is the decision/leaf, in this case, whether the passenger died or survived, represented as red and green text respectively.



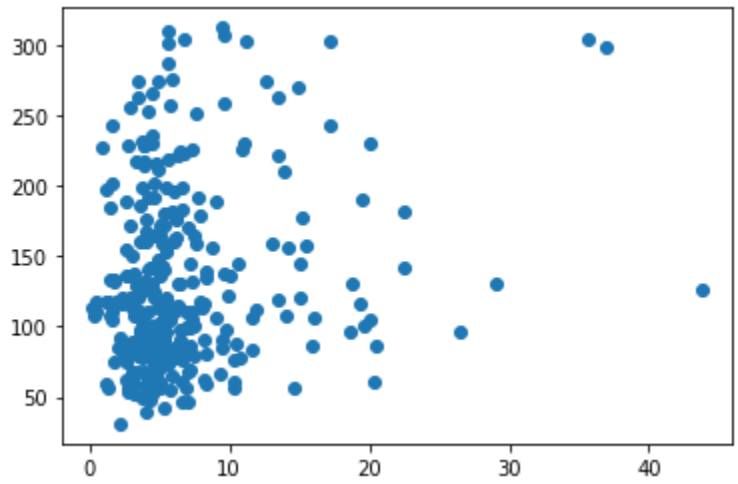
Linear Regression

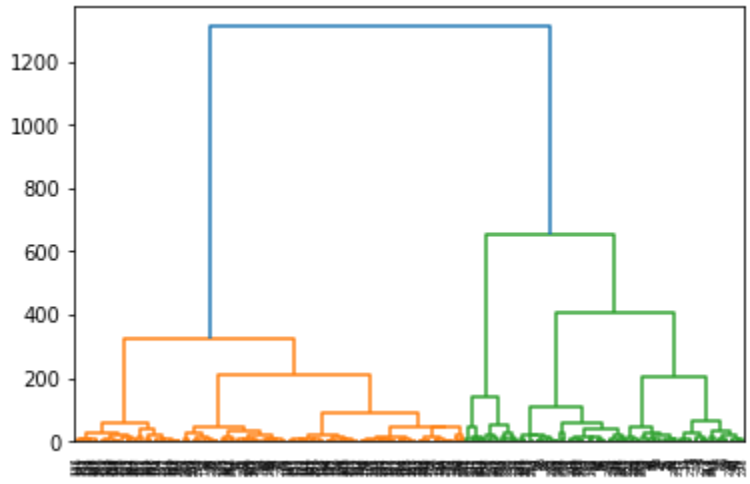
Linear regression analysis is used to predict the value of a variable based on the value of another variable. The variable you want to predict is called the dependent variable. The variable you are using to predict the other variable's value is called the independent variable.

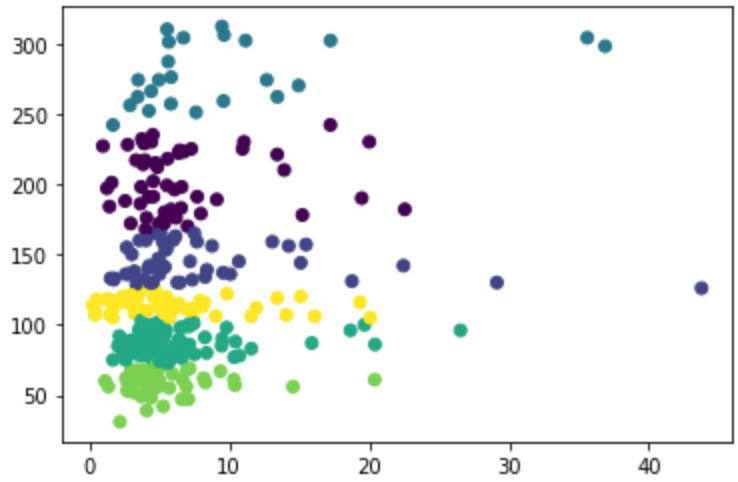
This form of analysis estimates the coefficients of the linear equation, involving one or more independent variables that best predict the value of the dependent variable. Linear regression fits a straight line or surface that minimizes the discrepancies between predicted and actual output values. There are simple linear regression calculators that use a "least squares" method to discover the best-fit line for a set of paired data. You then estimate the value of X (dependent variable) from Y (independent variable).



Hierarchical clustering algorithm







HC VS KMEANS

K-Means is that it needs us to pre-enter the number of clusters (K) but Hierarchical clustering has no such requirements. The algorithm on itself deduces the optimum number of cluster and displays it form of dendrogram.

Performance of K-Means on spherical data is better than that of HCA Hierarchical clustering is a purely agglomerative approach and goes on to build one giant cluster.

K-Means algorithm in all its iterations has same number of clusters.

K-Means need circular data, while Hierarchical clustering has no such requirement.

K-Means uses median or mean to compute centroid for representing cluster while HCA has various linkage method that may or may not employ the centroid.

With introduction of mini batches K-Means can work with very large datasets but HCA lacks in this regard.

Hierarchical methods are suited for cases which require arrangement of the clusters into a natural hierarchy. In K-means all clusters are on same level i.e. similar WCSS or cohesiveness.

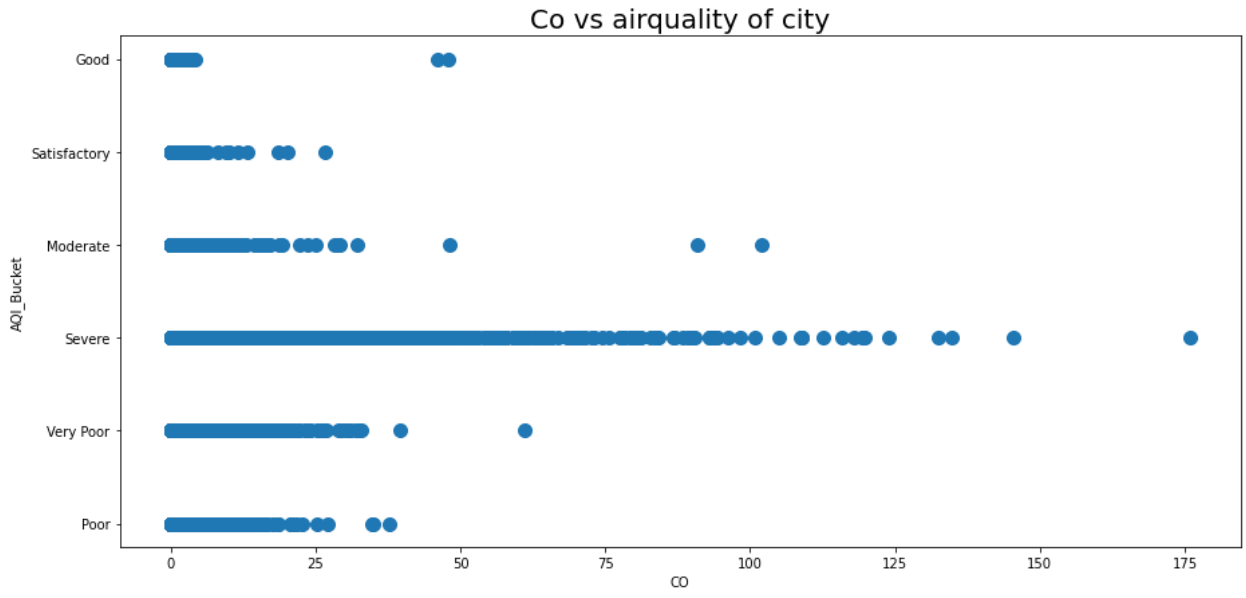
HCA can produce reproducible results while older versions of K-Means can't

K-Means simply divides data into mutually exclusive subsets while HCA arranges it into a tree format.

K-Means

K-means clustering is one of the simplest and popular unsupervised machine learning algorithms.

Typically, unsupervised algorithms make inferences from datasets using only input vectors without referring to known, or labelled, outcomes.

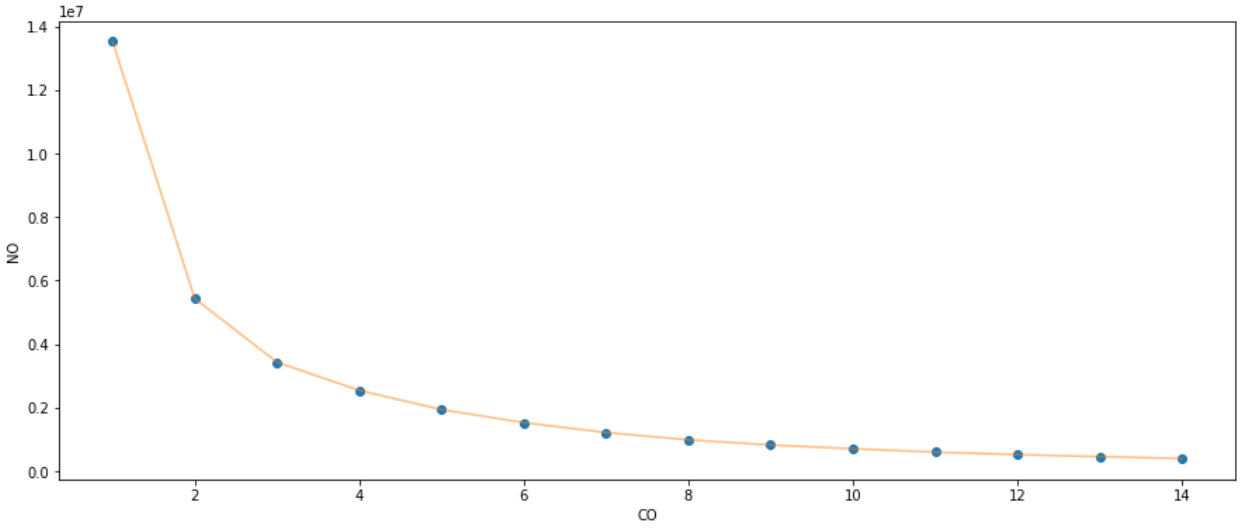


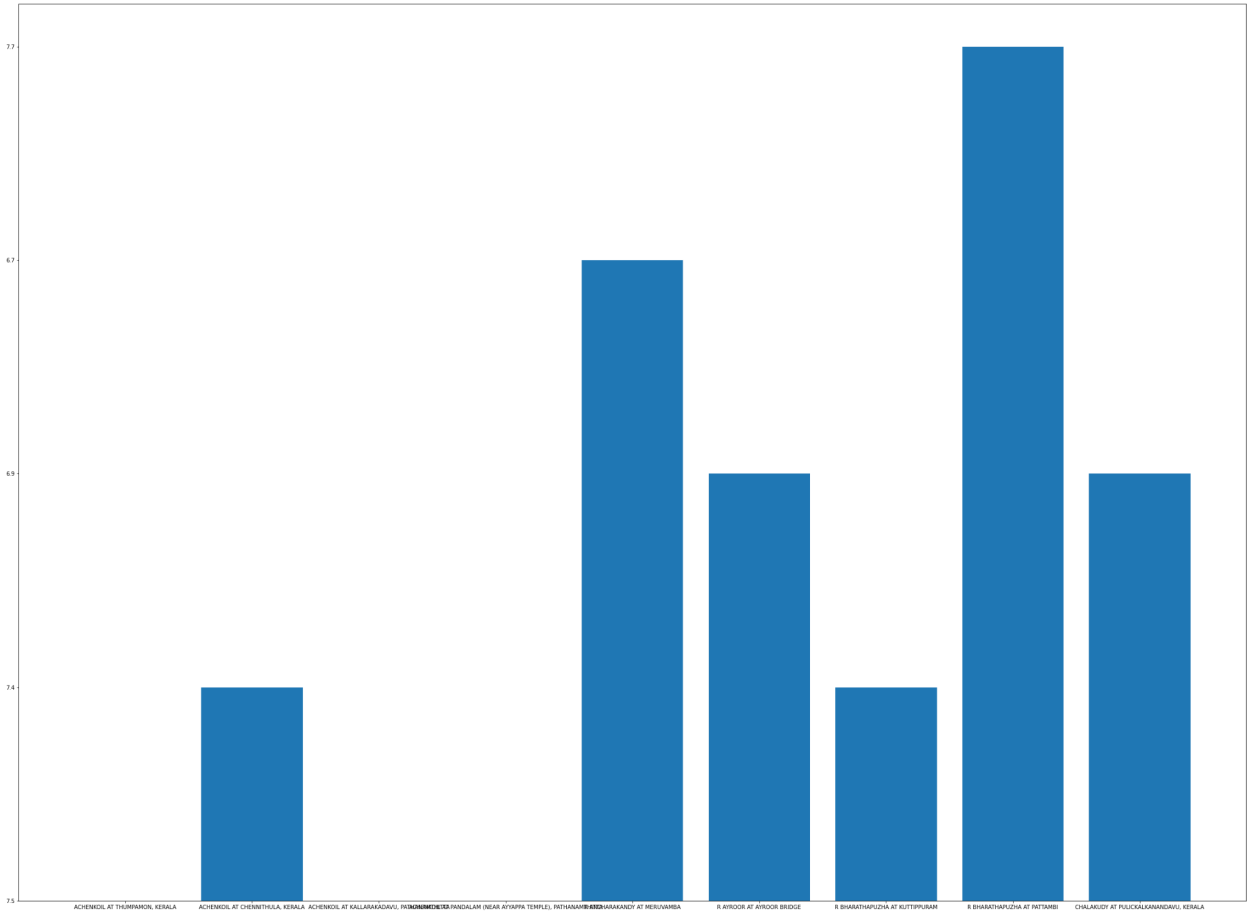
A cluster refers to a collection of data points aggregated together because of certain similarities.

You'll define a target number *k*, which refers to the number of centroids you need in the dataset. A centroid is the imaginary or real location representing the center of the cluster.

Every data point is allocated to each of the clusters through reducing the in-cluster sum of squares.

In other words, the K-means algorithm identifies *k* number of centroids, and then allocates every data point to the nearest cluster, while keeping the centroids as small as possible.




7.7
6.7
6.9
7.4
7.5
ACHENKOIL AT THUMPAMON, KERALA
ACHENKOIL AT CHENNITHULA, KERALA
ACHENKOIL AT KALLARAKADAVU, PATHANAMTHITTA
R AYROOR AT AYROOR BRIDGE
R BHARATHAPUZHA AT KUTTIPPURAM
R BHARATHAPUZHA AT PATTAMBI
CHALAKUDY AT PULICKALKANANDAVU, KERALA