5 classification algorithms - Decision Trees, Boosted Trees, Random Forest, Support Vector Machines and Neural Networks.

Name: Prince Kumar rana

Organization: AITS Tech. System

(www.ai-techsystems.com)

Country: India

Email: ranaprince377@gmail.com

INTRODUCTION

Classification is supervised machine learning algorithms. In supervised machine learning we have label in our data along with features. Classification is process or technique to categorize our data into desired classes.

So, in this section we will discuss classification algorithms.

Keywords- Support vectors, gain, neural, training, testing, testing, entropy, attributes.

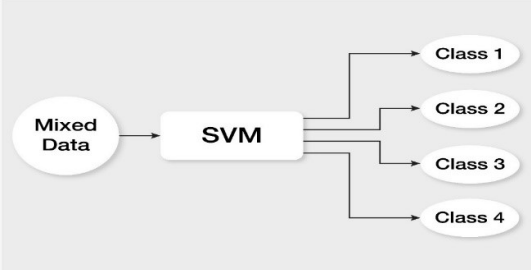
1.SUPPORT VECTOR MACHINE (SVM)

The Support Vector Machine, created by Vladimir Vapnik in the 60s, but pretty much overlooked until the 90s is still one of most popular machine learning classifiers.

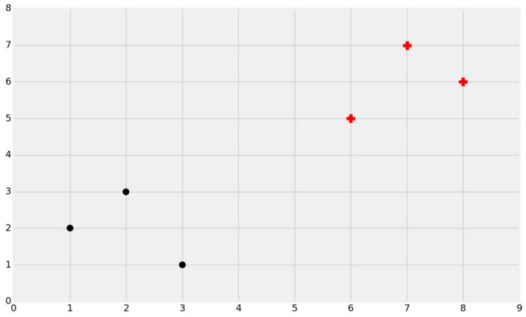
1. SVM or Support Vector Machine is a linear model for classification and regression problems.

2. It can solve linear and non-linear problems and work well for many practical problems.

3. The idea of SVM is simple: The algorithm creates a line or a hyper plane which separates the data into classes. So, the objective of svm is to find best fit line to separate the data. This best fit line is called hyper plane. So, this hyper plane will be the decision boundary.

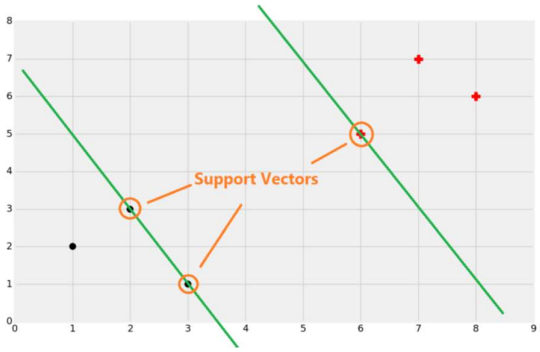


Lets understand it with following data.



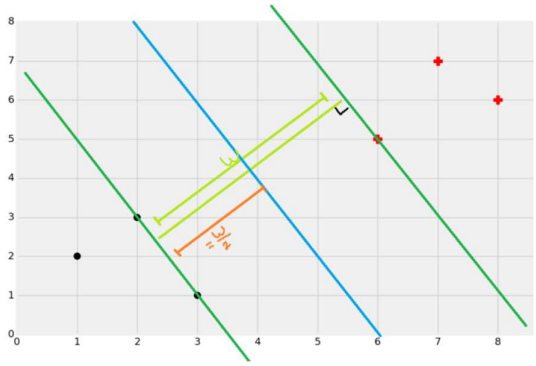
Now with this data SVM find the decision boundary i.s hyperplane or best fit line. Once the bet fit line is discovered, machine can easily classify in which group the new data point will lie whether in red or black group.

But here the main challenge is how to find best fit line. For, this svm first discovers the support vectors as shown in figure below



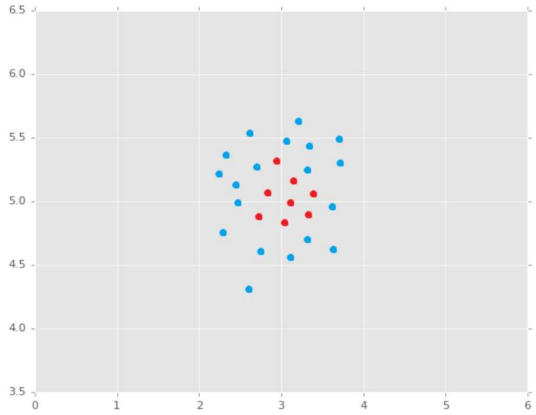
So now the decision boundary may line somewhere between these two support vectors. For, this svm calculate the maximum separation between two

support vectors **'W'**. And now the decision boundary becomes at half of the maximum width of two support vectors i.s **'W/2'.**

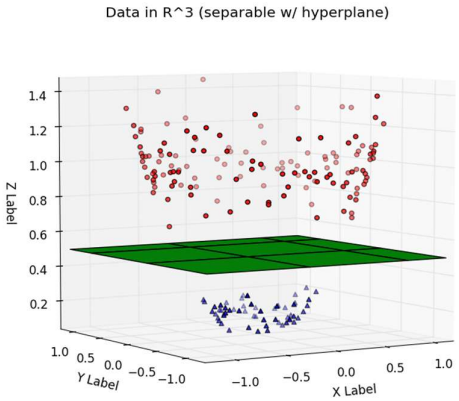


And we have got the decision boundary.

This is easy to find decision boundary with this linear data. But, what the data is non liner something like as shown in figure below.



To find hyperplane for this data, first we have to convert the data into the 3D as shown in figure.



Now it can be easily understood how we got decision boundary for this type of non linear data.

2.NEURAL NETWORK

Artificial neural networks are computing systems that are inspired by the biological neural networks that constitute animal brains, but it is not same or identical to brain. The objective of the Neural network was to solve problems in the same way that a human brain would. To learn such network, we provide many examples to it. For example image recognition. It can identify the give input image is cat or not. This learns by making the feature itself like tail, size etc and label it as cat and not cat after told by us.

Neural Networks consist of the following components

- An input layer, x
- An arbitrary amount of hidden layers
- An output layer, ŷ
- A set of weights and biases between each layer, *W and b*
- A choice of activation function for each hidden layer, *σ*. we'll use a Sigmoid activation function.

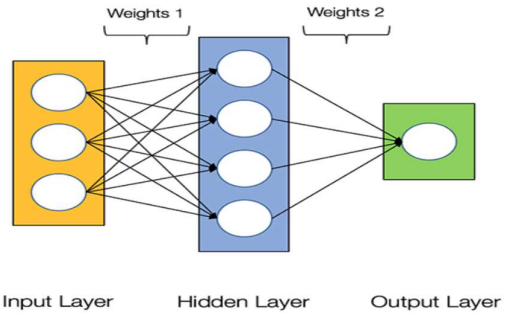


Figure: Architecture of a 2-layer Neural Network

A. Training the Neural Network

The output $\hat{y}$ of a simple 2-layer Neural Network is:

$$\hat{y} = \sigma(W_2\sigma(W_1 x + b_1) + b_2)$$

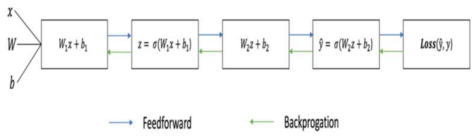
You might notice that in the equation above, the weights ***W*** and the biases ***b***are the only variables that affects the output $\hat{y}$.

Naturally, the right values for the weights and biases determines the strength of the predictions.

The process of fine-tuning the weights and biases from the input data is known as training the Neural Network.

Each iteration of the training process consists of the following steps:

- Calculating the predicted output $\hat{y}$, known as feedforward
- Updating the weights and biases, known as backpropagation



B. *Feedforward*

As we've seen in the sequential graph above, feedforward is just simple calculus and for a basic 2-layer neural network, the output of the Neural Network is:

$$\hat{y} = \sigma(W_2\sigma(W_1 x + b_1) + b_2)$$

However, we still need a way to evaluate the "goodness" of our predictions (i.e. how far off are our predictions), The Loss Function allows us to do exactly that.

C. *Loss Function:-*

$$Sum - of - Squares\ Error = \sum_{i=1}^{n}(y - \hat{y})^2$$

That is, the sum-of-squares error is simply the sum of the difference between each predicted value and the actual value. The difference is squared so that we measure the absolute value of the difference.

** Our goal in training is to find the best set of weights and biases that minimizes the loss function.*

D. *Backpropagation :*

Now that we've measured the error of our prediction (loss), we need to find a way to propagate the error back, and to update our weights and biases.

In order to know the appropriate amount to adjust the weights and biases by, we need to know the derivative of the loss function with respect to the weights and biases.

However, we can't directly calculate the derivative of the loss function with respect to the weights and biases because the equation of the loss function does not contain the weights and biases. Therefore, we need the chain rule to help us calculate it.

$$Loss(y,\hat{y}) = \sum_{i=1}^{n}(y - \hat{y})^2$$

$$\frac{\partial\, Loss(y,\hat{y})}{\partial W} = \frac{\partial Loss(y,\hat{y})}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial z} * \frac{\partial z}{\partial W} \qquad \text{where } z = Wx + b$$

$$= 2(y - \hat{y}) * \text{derivative of sigmoid function} * x$$

$$= 2(y - \hat{y}) * z(1\text{-}z) * x$$

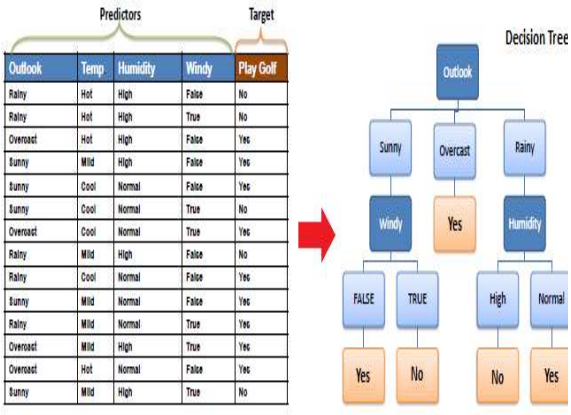
3.DECISION TREE

Decision tree is a classification algorithm which classify the given input data in tree like structure on the basis of some condition. Let's understand it with an example with the following data.

Table 4.6 The weather data with identification codes.

ID code	Outlook	Temperature	Humidity	Windy	Play
a	sunny	hot	high	false	no
b	sunny	hot	high	true	no
c	overcast	hot	high	false	yes
d	rainy	mild	high	false	yes
e	rainy	cool	normal	false	yes
f	rainy	cool	normal	true	no
g	overcast	cool	normal	true	yes
h	sunny	mild	high	false	no
i	sunny	cool	normal	false	yes
j	rainy	mild	normal	false	yes
k	sunny	mild	normal	true	yes
l	overcast	mild	high	true	yes
m	overcast	hot	normal	false	yes
n	rainy	mild	high	true	no

Decision tree classify the given data down the tree from root node to leaf node. The decision tree representation for the above data is



Data is classified by starting from root node by testing the attribute

specified by that node. After testing attribute we move down the tree

which corresponds to one value of attributes. This process repeats at every node of tree till leaf of tree.

The challenge here is -

Which attribute is best classifier?Here the challenge is the selecting of which attribute to test at the node. We select the attribute which is most useful to classify data, for this, we use the information gain that measures how well the attribute separates the training data.

The ***information gain*** is simply the expected reduction in the entropy caused by partitioning the training data according to attribute. So, the attribute which have more gain will be chosen for the classification of data.

ENTROPY:

$$Entropy(S) \equiv \sum_{i=1}^{c} -p_i \log_2 p_i$$

Common terms used with Decision trees:

1. *Root Node:* It represents entire population or sample, and this further gets divided into two or more homogeneous sets.

2. *Splitting:* It is a process of dividing a node into two or more sub-nodes on the basis of attribute.

3. *Leaf/ Terminal Node:* Nodes do not split is called Leaf or Terminal node.

4. *Parent and Child Node:* A node, which is divided into sub-nodes is called parent node of sub-nodes whereas sub-nodes are the child of parent node.

4. RANDOM DECISION FOREST

Random forest is the supervised classification machine learning algorithm. Random forest is a collection of decision tree. We have learnt about decision tree in which we take a root node classify the training data set given to root node by selecting best attribute and go down to left and right to corresponding value of attribute.

In random forest we create different decision tree by selecting different attribute and combine all the created decision tree. In a random forest, more number of tree means more accuracy, so accuracy of random forest is higher than decision tree.

ADVANTAGE:

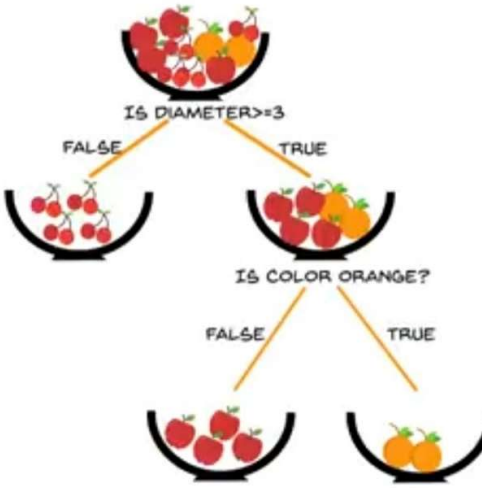
1. Use of multiple trees reduce the risk of overfitting.
2. Training time is less.
3. Runs effectively on large database.
4. For large data it produces highly accurate predictions.
5. Random forest can maintain accuracy when a large proportion of data is missing.

Now, lets focus how random forest algorithm work.

Let we have built three trees.

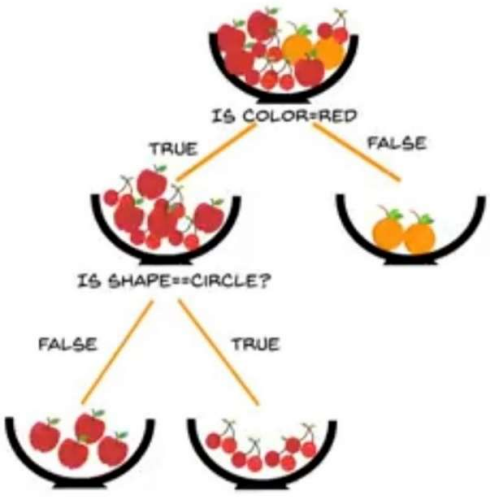
First decision tree1 by taking attributes *Is diameter >=3 and *is colour orange?

Representation of tree1 as :-



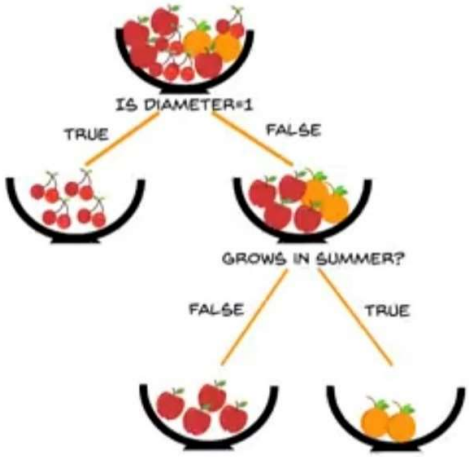
And then second tree is created by considering the attribute *Is colour = red and *Is shape == circle?

Representation as :-

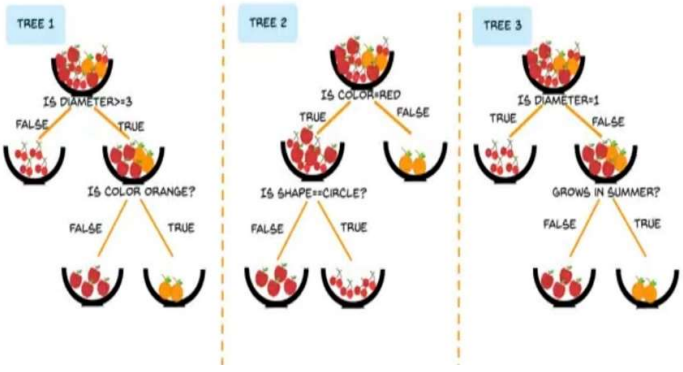


And third tree is created with Attributes *Is diameter=1 and *Grows in summer

The representation is:-



We have created three trees. Now, to build a forest combine all the trees as represented in the figure below.



PREDICTION:

When a new data is given for the prediction all the decision trees give some output. But these are the

outputs of all the decision tree, then how the random forest predict the output? The answer to this question is that it counts the similar outputs of trees, and the output which has more number of votes will be the output of Random Forest.

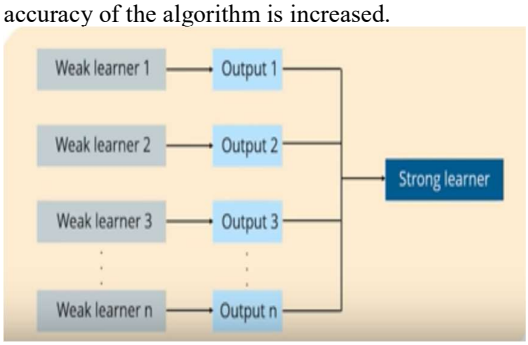
6.BOOSTING TREE ALGORITHM

Boosting: Boosting is the process or method that uses set of machine learning algorithms to combine weak learners to form strong learners in order to increase the accuracy of the model. Boosting is an iterative procedure.

Let's begin with working of boosting algorithm. Let we have N learners. Initially we assign the equal weight for all records of original training data. Now we provide training to the 'learner1'(tree1), and this learner classify the data.

If the accuracy of the 'learner1' is 100% then there is no problem and no need for further iteration. But if it classifies some data points incorrectly, then we increase the weight for those data points which was classified incorrectly. Now we provide training to the next 'learner2' (or tree2). Learner2 classifies the data, again learner 2 can classify some datapoint incorrectly. We again increase the weight for incorrectly classified datapoints and perform the next iteration. We repeat the iteration till the learner N.

Till now we have got N different outputs. These all N learners are the weak learners. Their outputs are combined to make a strong learner. So, the accuracy of the algorithm is increased.



CONCLUSION

We have discussed different classification algorithms, We can chose any algorithms for classification. If The accuracy is a concern, then one can test different Algorithms and cross validate them. If the taining dataset is small, one should use model that have low variance and high bias. If the the training data is large one should the algorithm that have high variance and low bias.

In [31]:
```
#Accuracy score
accuracy1 = dtree.score(X_test,y_test)          #DecisionTree
accuracy2 = sv.score(X_test,y_test)             #SVM
accuracy3 = RFC.score(X_test,y_test)            #RandomForest
print("DecisionTree = ", accuracy1)
print("SVM = ", accuracy2)
print("RandomForest = ", accuracy3)
```

```
DecisionTree =  1.0
SVM =  1.0
RandomForest =  0.9666666666666667
```

In [32]:
```
#Prediction
pred1 = dtree.predict(X_test)
pred2 = sv.predict(X_test)
pred3 = RFC.predict(X_test)
print(pred1,"#DecisionTree")
print(pred2, "#SVM")
print(pred3, "#RandomForest")
```

```
[2 1 1 0 0 2 2 0 1 2 0 1 0 2 1 1 0 2 2 1 2 0 1 0 0 0 0 1 0 2] #DecisionTree
[2 1 1 0 0 2 2 0 1 2 0 1 0 2 1 1 0 2 2 1 2 0 1 0 0 0 0 1 0 2] #SVM
[2 1 1 0 0 2 2 0 1 2 0 1 0 2 1 1 0 2 1 1 2 0 1 0 0 0 0 1 0 2] #RandomForest
```

REFERENCES

[1] https://www.youtube.com/watch?v=eM4uJ6XGnSM

[2] https://www.youtube.com/watch?v=kho6oANGu_A&t=26s