Description of datasets

Dataset 1 : Power consumption of household

The first dataset involves power consumption of a house based on the various environmental factors around the house with the dataframe having 29 columns. I have dropped date and lights columns, placed the output appliances output value in a dataframe and rest of the columns are in another dataframe. Scaling is required for SVM as there we don't want columns with higher numeric values dominating our experiment and so I have used MinMax scalar to bring all the column values between 0 and 1. If the output value after scaling is greater than 0.4, it is classified as 1 and the rest are put to be 0.

Dataset 2 : Counter strike game

The second dataset involves an 2 team online shooting game – cop team vs terrorists team, with the missions such as bomb defusal, hostage rescue and VIP assassination. The game rated as by game review websites such as GameSpot(8.4) and IGN(8.9) it is played by 20 million people in this world.

Coming to our dataset, it involves 17 columns taken from the various games involving parameters such as – Game Map, Day, Month, Year, Date, Wait Time(s), Match Time(s), Team A , Rounds, Team B, Rounds, Ping, Kills, Assists, Deaths, Mvp's, HS%, Points and the dependent variable Result which states whether a particular team won (1) , lost (0) or tied(2) in the game.

The recent tournament involved US$1,000,000 prize pool which had 24 best teams from all over the world. How will my model be beneficial to them? The counter strike players can have a look at my model to learn how their team is performing and can have a good chance to win a prize in at least any of the million-dollar prize pool amount after improving on the same.

Purpose of random state : All throughout this experiments, I have set the random state to be 50, so that we get to have the same training and validation set every time we run the model.

70% of the dataset is taken as training data and 30% as validation data for all models.

(i) SVM Classification

SVM Classification involves classifying our dataset based on the output value with the separation done using an hyperplane having dimensions equal to the number of columns. I have implemented 3 kernels to perform the separation i.e. Linear, radial basis function and polynomial.

Experiments involve the permutations for all these conditions mentioned below,

- Number of Iterations : 10,100,1000,5000,10000,15000,20000
- Kernels: Linear, Radial basis function kernel and Polynomial
- tolerance : 0.001,0.01,0.1,1

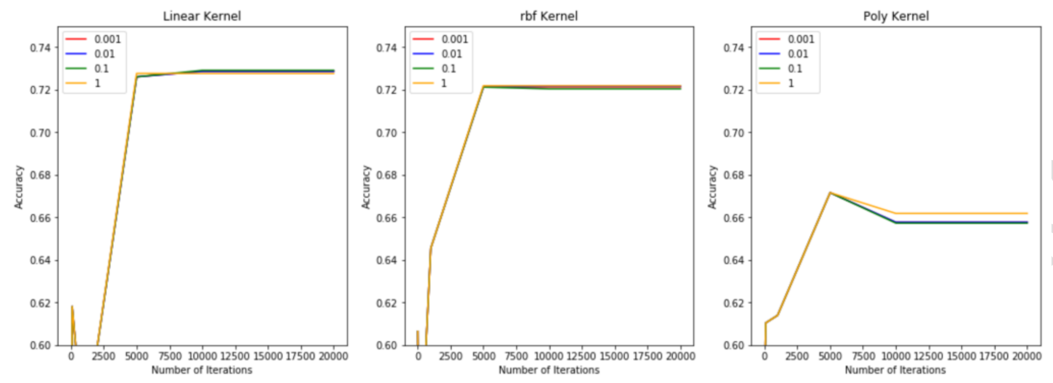
A total of 84 tests were done to compute the resulting accuracy values to determine the best kernel, tolerance and number of iterations. The results of the various experiments were obtained, I have graphically represented those to get a clear idea to choose the best accuracy.

COPYRIGHT - SIDDHARTH GOVINDARAJAN

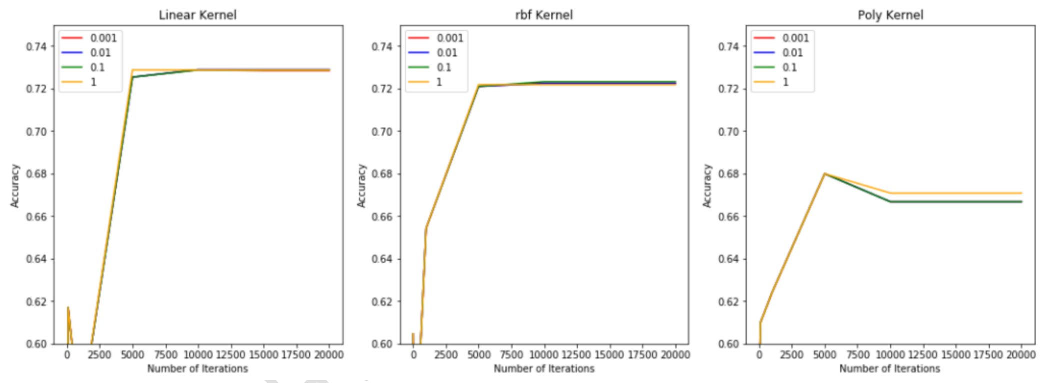
Dataset 1: Power consumption data

Graph set 1 :

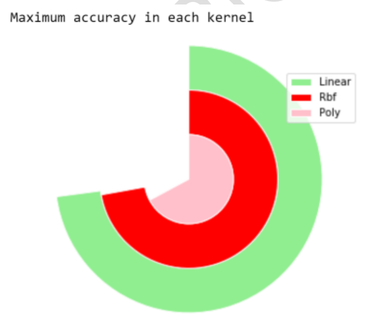
We see that the linear kernel is performing better for training set with number of iterations 5000 and after that there is not much changes.



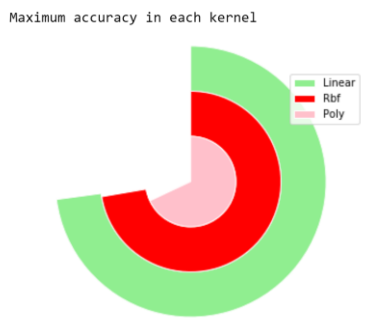
We see that the linear kernel is performing better for training set with number of iterations 5000 and after that there is not much changes.



<u>Training set</u>



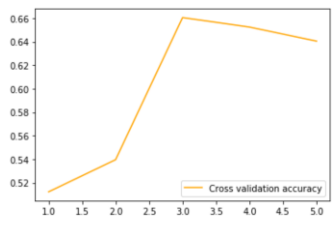
<u>Validation set</u>

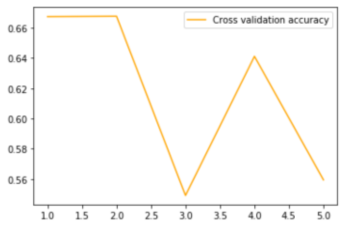


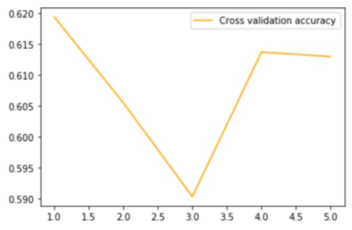
K-Fold Cross Validation

When cross validation is implemented using 5 folds, it gives a good accuracy for the third fold of cross validation for linear.

Linear RBF Poly







It is observed that linear separation performs better in both training and validation set. Performing for 5000 iterations,

Confusion Matrix

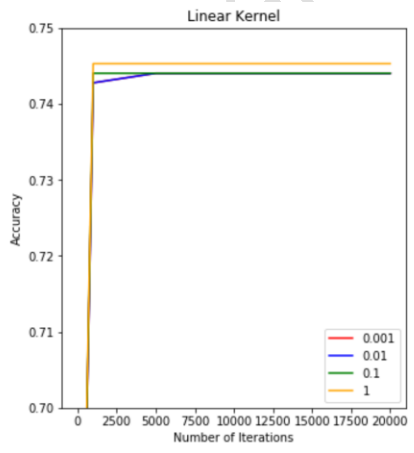
	Predicted 0	Predicted 1
Actual 0	1298 (TN)	913(FP)
Actual 1	693(FN)	3017 (TP)

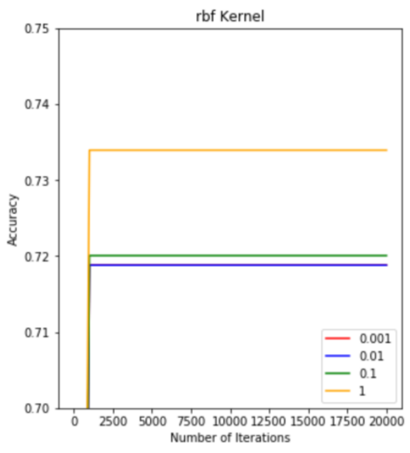
Parameters	Formula	Computation
Accuracy	(True positive + True negative) / (True positive + True negative + False Positive + False Negative)	0.7288
Sensitivity	True positive/(True positive + False Negative)	0.8132
Specificity	True Negative/(True Negative + False Positive)	0.5871
Precision	True Positive / (True positive + False Positive)	0.7677

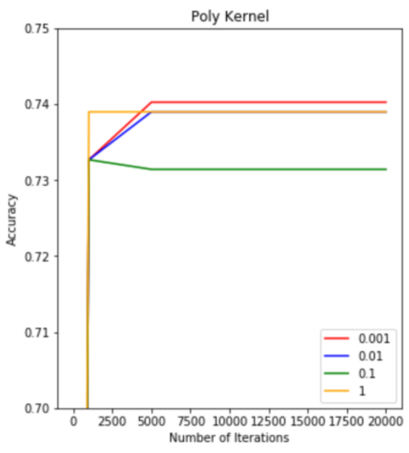
SIDHARTH GOVINDARAJAN

Dataset 2: Counter Strike game

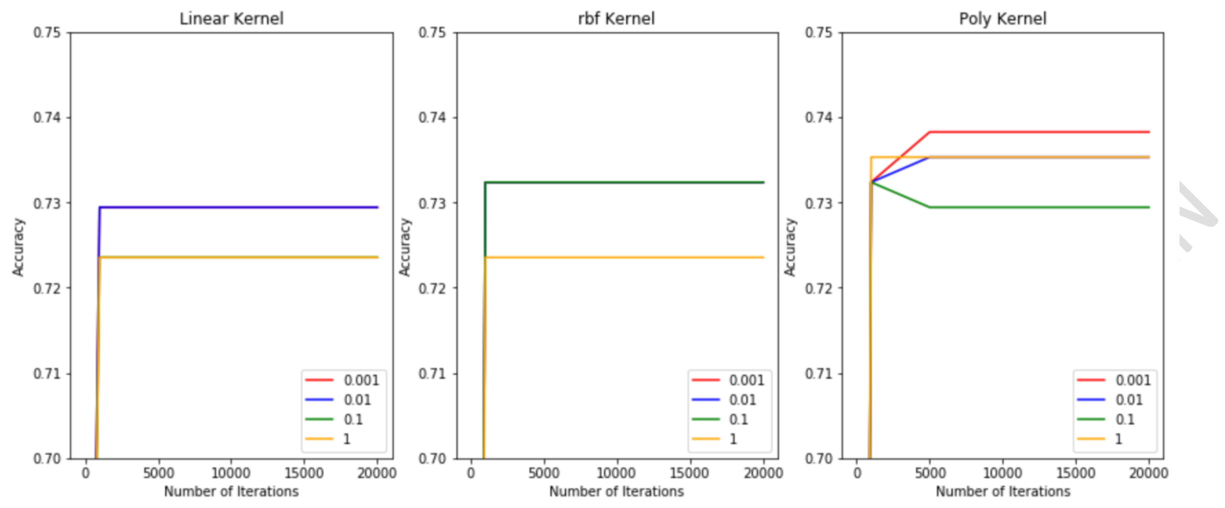
Graph set 1 :







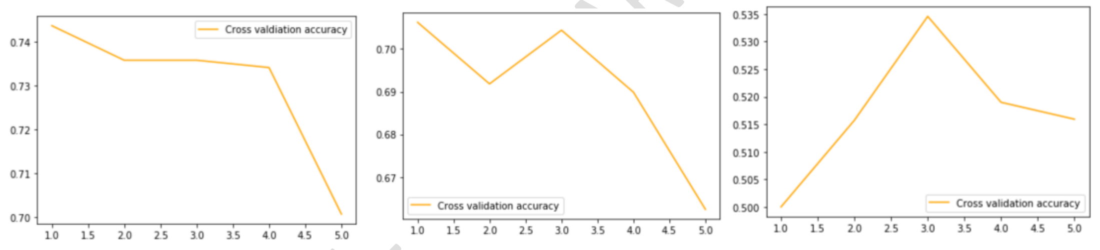
We see that the linear kernel is performing better for training set with number of iterations 1000 and after that there is not much changes.



We see that the poly kernel is performing better for validation set with number of iterations 5000 and after that there is not much changes.

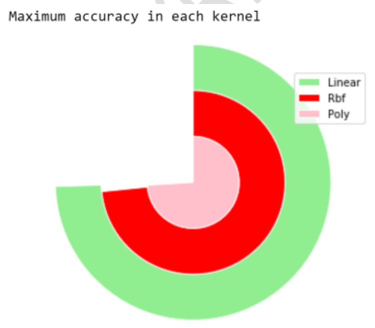
<u>K-Fold Cross Validation</u>

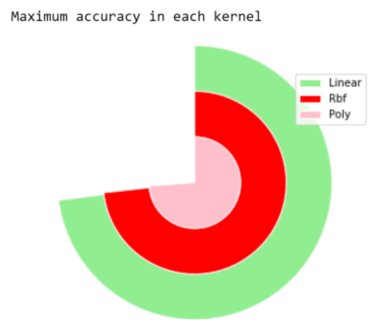
Linear RBF Poly



When cross validation is implemented using 5 folds, it gives a good accuracy for the first fold of cross validation for linear.

<u>Training</u> <u>Validation</u>





It is observed in the polar chart that the poly is having better accuracy than the other 2 kernels. Though we get the linear as a best for training set, we will go with the one best shown in validation data – poly.

	Predicted lose (0)	Predicted win (1)	Predicted Tie (2)
Actual Lose (0)	145	32	0
Actual Win (1)	36	103	0
Actual Tie (2)	15	9	0

Computing for only win and lose, neglecting tie,

Parameters	Formula	Computation
Accuracy	(True positive + True negative) / (True positive + True negative + False Positive + False Negative)	0.7848
Sensitivity	True positive/(True positive + False Negative)	0.7410
Specificity	True Negative/(True Negative + False Positive)	0.8192
Precision	True Positive / (True positive + False Positive)	0.7630

(ii) Decision tree

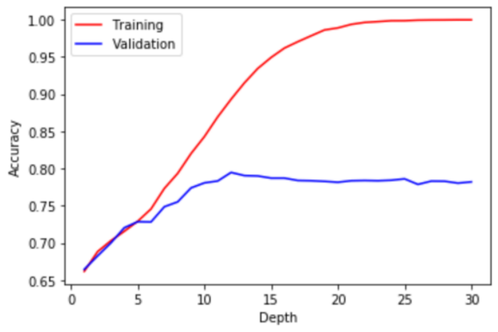
Decision tree is one of the most interesting supervised learning technique and in this report it has been used as classification in both the datasets. The structure of decision tree involves rules built from top to bottom like a flow chart and is easy to understand for everyone. The top most node in the tree is root node and each node is split into two or more sub-nodes and the node which doesn't split is called as a leaf node.

The accuracy of a decision tree increases with depth and so the question arises why not to increase the depth to maximum accuracy. The answer to this is quite simple, maximum best accuracy can be attained using training data but it causes overfitting in validation data and so we need to restrict the tree after a certain depth. That is determined by plotting line graph for training and validation, the point at which the accuracy of validation set decreases, that's the point where we need to stop training our model.

There are 2 criterians to build the decision tree – Gini and Entropy and I have chosen Gini to build the tree.

Dataset 1 : **Power Consumption**

In this dataset, I have plotted for 30 depths, training and validation dataset and it is observed that the decision tree starts to overfit after level 12 and so we need to prune our decision tree to have our max depth to 12.



The confusion matrix:

COPYRIGHT – SIDDHARTH GOVINDARAJAN

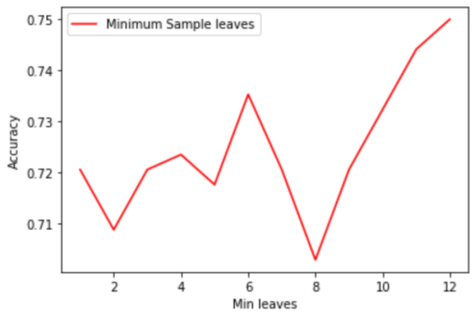
	Predicted 0	Predicted 1
Actual 0	1685 (TN)	526(FP)
Actual 1	689 (FN)	3021 (TP)

Parameters	Formula	Computation
Accuracy	(True positive + True negative) / (True positive + True negative + False Positive + False Negative)	0.7948
Sensitivity	True positive/(True positive + False Negative)	0.8143
Specificity	True Negative/(True Negative + False Positive)	0.7621
Precision	True Positive / (True positive + False Positive)	0.8517

COPYRIGHT - SIDDHARTH GOVINDARAJAN

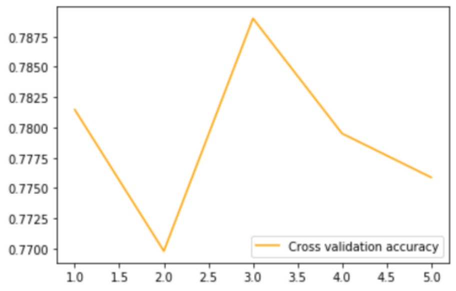
Performing pruning on a fully grown tree

Here I have taken the depth to be 10 and plotted the accuracy for various minimum sample leaves. It is seen that the accuracy of close to 0.75 is obtained when we have the minimum sample leaves as 12.



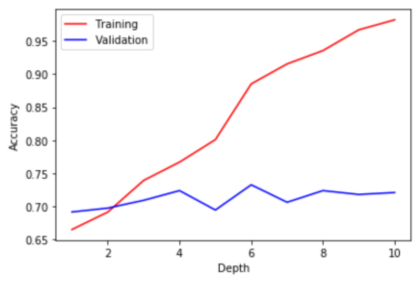
K-Cross validation

When the cross validation is done for the max-depth of 12 with 5 cross-validation, we see that the accuracy is great for 3rd fold.



Dataset 2 : Counter strike game

In this dataset, I have plotted for 30 depths, training and validation dataset and it is observed that the decision tree starts to overfit after level 4 and so we need to prune our decision tree to have our max depth to 4.



Confusion Matrix :

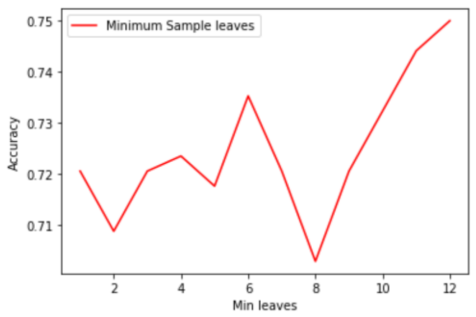
	Predicted lose (0)	Predicted win (1)	Predicted Tie (2)
Actual Lose (0)	150	26	1
Actual Win (1)	43	94	2
Actual Tie (2)	20	2	2

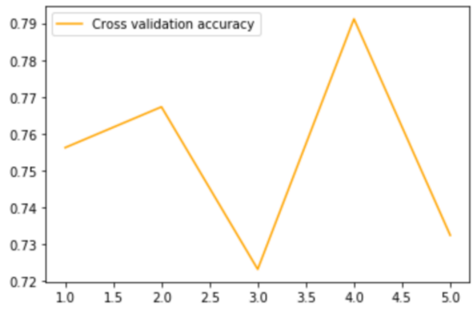
Computing for only win and lose, neglecting tie,

Parameters	Formula	Computation
Accuracy	(True positive + True negative) / (True positive + True negative + False Positive + False Negative)	0.7235
Sensitivity	True positive/(True positive + False Negative)	0.6861
Specificity	True Negative/(True Negative + False Positive)	0.8523
Precision	True Positive / (True positive + False Positive)	0.7833

Performing pruning on a fully grown tree

Here I have taken the depth to be 7 and plotted the accuracy for various minimum sample leaves. It is seen that the accuracy of more than 0.75 is obtained when we have the minimum sample leaves as 4. (Left figure)





K-Cross validation

When the cross validation is done for the max-depth of 4 with 5 cross-validation, we see that the accuracy is highest for 4rd fold. (Right figure)

(iii) Boosting – Decision tree

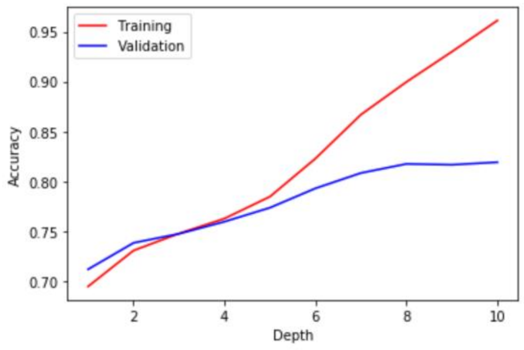
It has been found that when weak learning algorithms are put together namely Ensemble, they give us a better accuracy. During the first iteration every dataset has equal chance of getting picked for

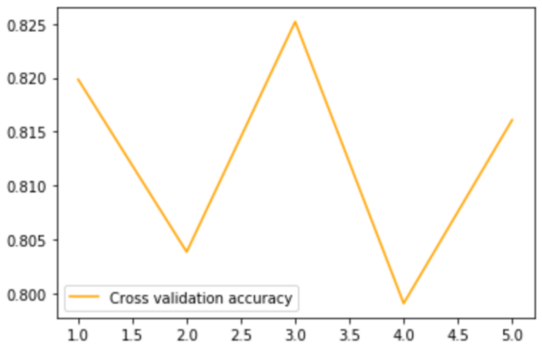
SIDDHARTH GOVINDARAJAN

implementation and with series of iterations, those dataset which get misclassified get a better chance to be picked for model implementation. I have implemented GradientBoostingClassifier to perform this process and there has been a good accuracy when compared to the models implemented without ensemble method.

Dataset 1 : Power Consumption

The accuracy for validation dataset sees no dip until level 8 and so having the depth of 8 would avoid us overfitting problem. (Left figure below)





K-Cross validation

When the cross validation is done for the max-depth of 10 with 5 cross-validation, we see that the accuracy is highest for 3rd fold. (Right figure below)

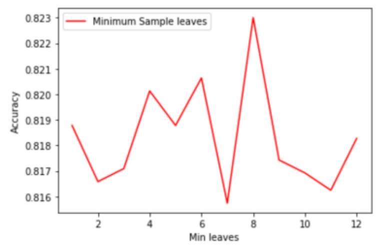
Confusion Matrix :

	Predicted 0	Predicted 1
Actual 0	1667(TN)	544(FP)
Actual 1	525(FN)	3185(TP)

Parameters	Formula	Computation
Accuracy	(True positive + True negative) / (True positive + True negative + False Positive + False Negative)	0.8195
Sensitivity	True positive/(True positive + False Negative)	0.8585
Specificity	True Negative/(True Negative + False Positive)	0.7540
Precision	True Positive / (True positive + False Positive)	0.8541

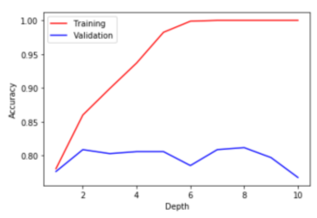
Performing pruning on a fully grown tree : Here I have taken the depth to be 10 and plotted the accuracy for various minimum sample leaves. It is seen that the accuracy of more than 0.82 is obtained when we have the minimum sample leaves as 8.

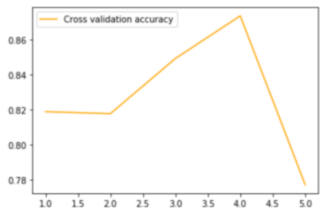
COPYRIGHT - S[illegible]HARTH G[illegible]AJAN



Dataset 2 : Counter Strike gaming

The accuracy for validation dataset sees no dip until level 2 and so having the depth of 2 would avoid us overfitting problem. (Left side figure)





<u>K-Cross validation</u>

When the cross validation is done for the max-depth of 10 with 5 cross-validation, we see that the accuracy is highest for 4th fold. (Right side figure)

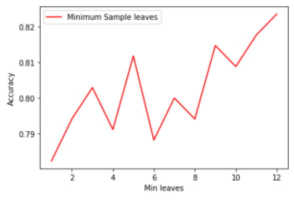
<u>Confusion matrix</u>

	Predicted lose (0)	Predicted win (1)	Predicted Tie (2)
Actual Lose (0)	141	36	0
Actual Win (1)	26	113	0
Actual Tie (2)	0	0	24

Since the tie counts are less, we will determine the confusion matrix parameters for the winning and losing as marked.Metrics only for winning and losing,

Parameters	Formula	Computation
Accuracy	(True positive + True negative) / (True positive + True negative + False Positive + False Negative)	0.8038
Sensitivity	True positive/(True positive + False Negative)	0.8130
Specificity	True Negative/(True Negative + False Positive)	0.7966
Precision	True Positive / (True positive + False Positive)	0.7584

<u>Performing pruning on a fully grown</u> tree : Here I have taken the depth to be 10 and plotted the accuracy for various minimum sample leaves. It is seen that the accuracy of more than 0.82 is obtained when we have the minimum sample leaves as 12.



COPYRIGHT - SIDDHARTH GOVINDARAJAN

Comparison between 3 algorithms : Power Consumption dataset

Parameters	SVM	Decision Tree	Boosting
Accuracy	0.7288	0.7948	0.8195
Sensitivity / True positive rate	0.8132	0.8143	0.8585
Specificity	0.5871	0.7621	0.7540
Precision	0.7677	0.8517	0.8541

When comparing the 4 parameters that we have obtained for each of the 3 models, boosting is performing better than the other 2 models.

Comparison between 3 algorithms : Counter Strike dataset – win or lose

Parameters	SVM	Decision Tree	Boosting
Accuracy	0.7848	0.7235	0.8038
Sensitivity / True positive rate	0.7410	0.6861	0.8130
Specificity	0.8192	0.8523	0.7966
Precision	0.7630	0.7833	0.7584

When comparing the 4 parameters that we have obtained for each of the 3 models, Decision tree and boosting are performing better than SVM. Since accuracy is better for Boosting algorithm, we can use boosting to build our model.

Comparing the best models of 2 datasets:

The accuracy, sensitivity, specificity and precision was higher in the dataset which had more rows (Power consumption) when compared with the dataset which had many rows (Counter strike). The decision tree model was implemented, the power consumption dataset said that the living room temperature was the most important factor determining the power consumption. The decision tree implemented for counter strike said that the top root which is the number of deaths determine the victory of a team and so a team members should improve upon staying alive in the game period.

References:

1. https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47
2. http://ijarcsse.com/Before_August_2017/docs/papers/Volume_4/10_October2014/V4I10-0254.pdf
3. https://medium.com/@mtterribile/understanding-cross-validations-purpose-53490faf6a86
4. https://matplotlib.org/
5. https://en.wikipedia.org/wiki/Counter-Strike:_Global_Offensive_Major_Championships
6. https://en.wikipedia.org/wiki/Counter-Strike
7. https://medium.com/greyatom/decision-trees-a-simple-way-to-visualize-a-decision-dc506a403aeb
8. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

COPYRIGHT - SIDDHARTH GOVINDARAJAN