TASK FOR DBSCAN ALGORITHM

In this project so far we are implementing DBSCAN(DENSITY BASED SPATIAL CLUSTERING ALGORITHM TO DETECT NOISE).

So, the provided dataset We had given a large dataset(.csv) where it consists of 5030 data points in which it consists of 5 features or columns, which are Days, Time, Latitude, Longitude, Temperature. Here the work we have to falsified (add noise) to the column of Temperature, in order of 2% 4%, 6%, 8%, 12%, etc. of the actual dataset. Also, we have to set the minPts values for each % of outlier added as in increasing order that is let say m=3, 5, 7, 9, etc. along with the NeareastNeighbour(ns) say, 120, 160, 200, 320, 400 as randomly and run with the DBSCAN algorithm.

So by using that falsified dataset(.csv) we have to detect noises/outliers using DBSCAN algorithm.

Let us know a brief how a DBSCAN algorithm works,

Initially all data points are unvisited. We first go to the first data point and see if it is a core point/border point/outlier point. If it is a “core point”, we mark as visited and then we make a cluster and then we check other point if it is core point/border point/outlier point. If it is outlier, we simply ignore it, we will not include into cluster. If one point is visited and if it is a “Core point” and next point is discovered as a “Core point” and also visited and these two-core points are neighbor of each other and we put both these core point in same cluster. In these ways cluster is propagate. Core point is strong constituent of cluster and also boundary point is always neighbor of core point and also include that boundary point into cluster. And only ignore the noise point, we don’t include into the cluster.

We also calculate the ‘True Positive Rate, False Positive Rate, Recall, Precision and accuracy’ of the outlier detection.

Now we are briefly explaining the process how we execute this algorithm:

- We had added seven times loops for each %of outliers added into dataset and run 5 times inner loop for each minPts and ns value, inside every 5 inner loop of ns and ms we are also doing 2 times loop for every ns and ms values. So there will be 7*5*3= 105 times loops will run. So, in every 5 inner loop iteration we had find the accuracy, TPR, FPR, Recall, Precision. After that we had compute the average accuracy, avg. recall, avg. precision, avg. TPR, avg. FPR for each iteration and stored in (7*5)2D matrix for each % of outlier added.

- Based on it, we had to plot the graph for %of outlier added vs average accuracy w.r.t each minPts values.
- Same way we also had to plot the graph for %of outlier added vs recall w.r.t each minPts values.
- Also for %of outlier added vs precision w.r.t each minPts values.
- And at last for ROC(FPR vs TPR) curve for showing the best MinPts which has less fpr and high tpr of our model.
- So from the graph we will be able to show how well the code is working as when we increase the % of outlier added then the accuracy drops like this.

So, now let us have a brief recap for what are actually the Accuracy, Precision, Recall, TPR,FPR.

TP is how many datapoints are truely detected as outliers from the actual outliers datapoints.

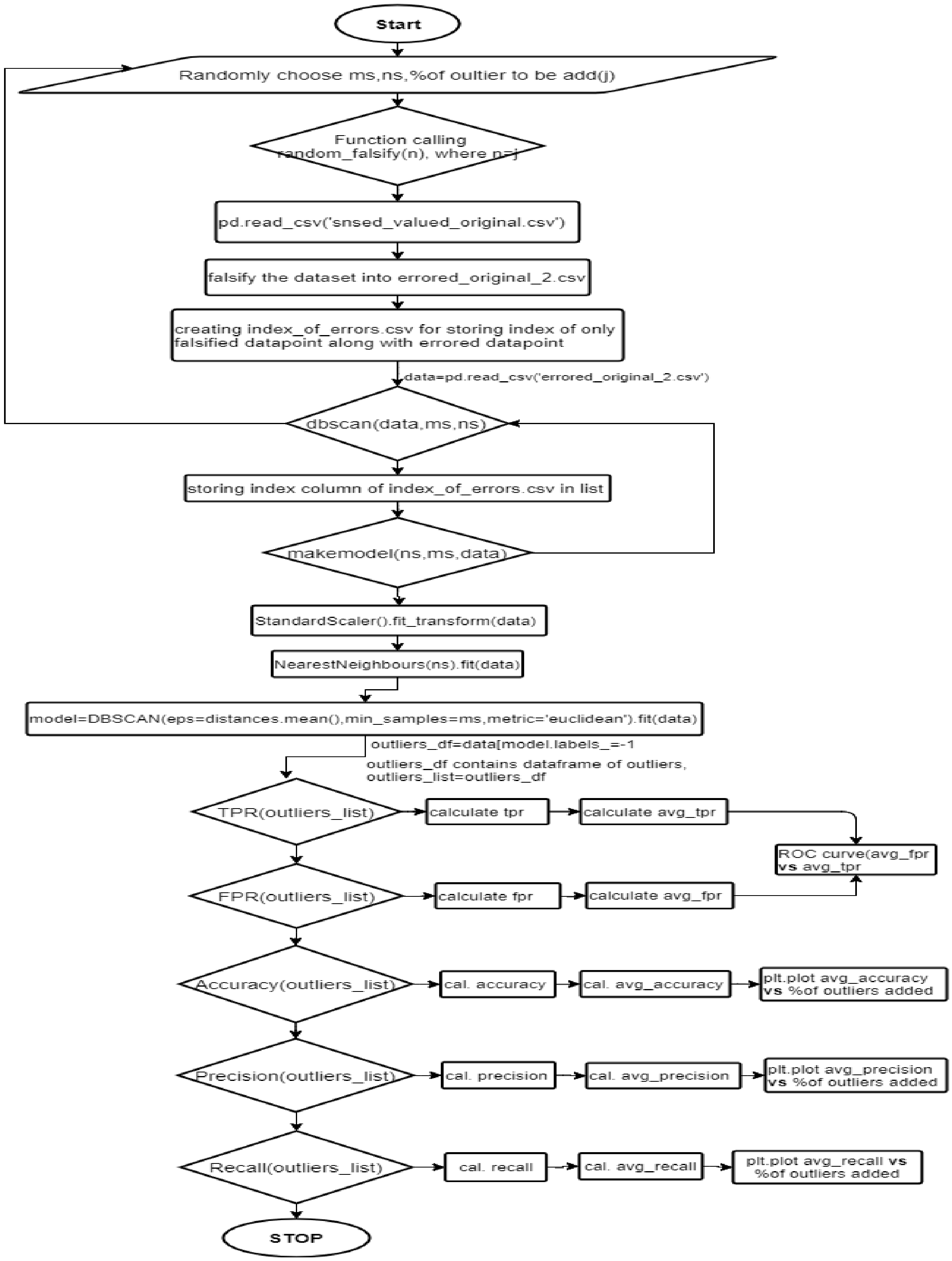
FN is how many actual outlier datapoints are not detected as outliers.

FP is how many datapoint are falsily detected as outliers from the actual outliers datapoints.

TN is how many datapoints are not outliers and also not detected as outliers(good datapoints).

- "TPR"= "TP/(TP+FN)"
- "FPR"= "FN/(TP+FN)"
- "Accuracy"= "(TP+TN)/(TP+TN+FN+FP)*100"
- "Precision"= "TP/(TP+FP)"
- "Recall"= "TP/(TP+FN)"

FLOW CHART FOR DBSCAN WITH PROCESS



- Here ns is the k-nearest neighbour and ms is the MinSamples.
- Where we had given five ms and five ns value where ns is much smaller than ns. Say ns = 120, 160, 200, 320, 400 whereas ms = 3,5,7,9,11 as a randomly picked valued and we have given seven different %of outlier to be falsified such as 2, 4, 6, 8, 10, 12, 14 which is stored in j variable.
- Now we had called random_falsify(n) method, where we are passing j values to the random_falsify(n) method for falsifying the original dataset 'sensed_valed_original.csv' and copying the errored dataset into 'errored_original_2.csv'. In simple word for first iteration we are falsifying 2% of 5030 original datapoint, i.e, 101 datapoint will be falsified and then along with the remaining datapoint is copied into 'errored_original_2.csv'. In this way we will falsify for rest of the j values.
- Now we are creating another 'index_of_errors.csv' for storing index of only falsified datapoint along with errored datapoints.
- Now we assigned the 'errored_original_2.csv' dataset into 'data' dataframe, now we called dbcan(ns, ms and data) function. Say for first iteration ns=120, ms= 3 and data is the errored/falsifed dataset for 2%of all datapoints=101 datapoints. Inside dbscan method we are storing the Index column of 'index_of_errors.csv' in list l.
- And now inside dbscan there is another function called makemodel(ns, ms, data), where we do the StandardScaler().fit_transform(data) and passing ns value to k_neareast neighbour for calculation the distance between each points.
  Here the StandardScaler().fit_transform(data) method is for fit and transform the data(dbscan_data) and StandardScaler for transform dta such that its distribution will have its mean value 0 and standard deviation as 1 for scaling.
- Also we are passing the ns value to NearestNeighbours(ns).fit(data) , where ns is the k-nearest neighbour where it is the how many nearest neighbour we have to consider in term of distance by Euclidean distance and fit the data.
- Now we apply the actual DBSCAN where its parameters are eps = distances.mean(), min_samples = ms and metric = eucildean and fit the data into model dataframe.
- Now model.labels_ will do clusters the datapoints and labels them, the -1 label is consider for outliers, so we print the no. of outliers also we print

the total no. of clusters as mobel.labels_ by substracting -1 label as it is outliers.

- After that we will call TPR(outliers_list) function where outliers_list is the outliers_df and it contains the dataframe of outliers by assigning model.labels_=-1. Here we calculate TPR and then calculate the average TPR for each ms values epoch. Also we call FPR(outliers_list) function and calculate avg. fpr and avg. tpr and then plot a graph called ROC for determining the best ms value for our model.

  ROC curve stands for Receiver Operating Characterstic where graph is plot between FPR vs TPR. The good ms value can be determing by higher TPR and lower FPR region in the curve.

- Similarly we call Accuracy(outliers_list) fun. And cal. Avg. accuracy and plot graph between avg. accuracy and %of outliers added.
- Similarly we call precision(outliers_list) and recall(outliers_list) function and we cal. the avg. precision and avg. recall and plot avg. recision vs %of outlier to add also for avg. recall vs %of outliers to add.
- So finally, Process execution STOP.

RESULT AND SCREENSHOT

- **RESULT OF ITERATION**

ITERATION=1

PERCENTAGE OF OUTLIERS WHICH IS FALSIFIED=2%

Total number of falsified outlier added =101

ns=120 , eps=0.9918815011352472 , min_samples=3
Number of clusters=11
Number of outliers detected=94
TPR = 0.8217821782178217
FPR = 0.1782178217821782
Precison = 0.8829787234042553
Recall = 0.8217821782178217
Accuracy = 99.42345924453281
TP+TN+FP+FN = 5030

#for this ns=120, eps=0.9918815011352472, min_samples=3
it will run for more 2 times.

ns=160 , eps=1.0771535055632713 , min_samples=5
Number of clusters=1
Number of outliers detected=98
TPR = 0.8316831683168316
FPR = 0.16831683168316833
Precison = 0.8571428571428571
Recall = 0.8316831683168316
Accuracy = 99.38369781312127
TP+TN+FP+FN = 5030

#also for this ns=160, eps=1.0771535055632713, min_samples=5
it will run for more 2 times.

#Now similarly, for rest of each ns= 200, 320, 400 and
ms= 7, 9, 11 will run for 3 times.
Hence, this will be 1 iteration, inside it will run 3*7=35 times.

#Now for ITERATION=2, we will falsify percentage
of outliers which is falsified=4% and do the same task.

#similarly upto iteration 7=14% to be falsified, we will do.
so, lets see for Iteration =7,

ITERATION=7

PERCENTAGE OF OUTLIERS WHICH IS FALSIFIED=14%

Total number of falsified outlier added =705
ns=120 , eps=1.0090919997487906 , min_samples=3
Number of clusters=18
Number of outliers detected=111
TPR = 0.14042553191489363
FPR = 0.8595744680851064
Precison = 0.8918918918918919
Recall = 0.14042553191489363
Accuracy = 87.71371769383698
TP+TN+FP+FN = 5030

#for this ns=120, eps=1.0090919997487906, min_samples=3
it will run for more 2 times.
#Similarly it will do for ns=160, 200, 320, 400 w.r.t
ms= 5, 7, 9, 11 and run for each 3 times.

Runtime for executing the program: 102.58745813369751 seconds.

- **SCREENSHOT FOR AVERAGE DATAFRAMES(7 X 5) MATRIX:**

SCREENSHOT #1 : Average FPR

avg_fpr

	m=3	m=5	m=7	m=9	m=11
2%	99.423459	99.383698	99.125249	99.522863	99.642147
4%	98.568588	98.051690	97.892644	98.290258	98.091451
6%	96.322068	96.421471	96.481113	96.461233	96.441352
8%	94.174950	95.009940	94.691849	94.333996	94.592445
10%	91.769384	92.405567	92.544732	91.988072	91.829026
12%	89.980119	90.715706	90.815109	89.483101	89.622266
14%	87.713718	88.170974	87.634195	87.614314	86.978131

SCREENSHOT #2 : Average TPR

avg_tpr

	m=3	m=5	m=7	m=9	m=11
2%	0.821782	0.831683	0.722772	0.801980	0.821782
4%	0.698020	0.599010	0.574257	0.603960	0.524752
6%	0.427152	0.447020	0.466887	0.413907	0.407285
8%	0.307692	0.409429	0.367246	0.295285	0.325062
10%	0.206759	0.266402	0.278330	0.202783	0.182903
12%	0.187086	0.251656	0.250000	0.125828	0.135762
14%	0.140426	0.181560	0.130496	0.117730	0.070922

SCREENSHOT #3 : Average Precision

avg_precision

	m=3	m=5	m=7	m=9	m=11
2%	0.882979	0.857143	0.820225	0.952941	1.0
4%	0.927632	0.876812	0.852941	0.953125	1.0
6%	0.914894	0.912162	0.898089	0.992063	1.0
8%	0.898551	0.926966	0.925000	0.991667	1.0
10%	0.873950	0.911565	0.921053	0.980769	1.0
12%	0.896825	0.910180	0.943750	0.987013	1.0
14%	0.891892	0.876712	0.910891	0.988095	1.0

SCREENSHOT #4 : Average Recall

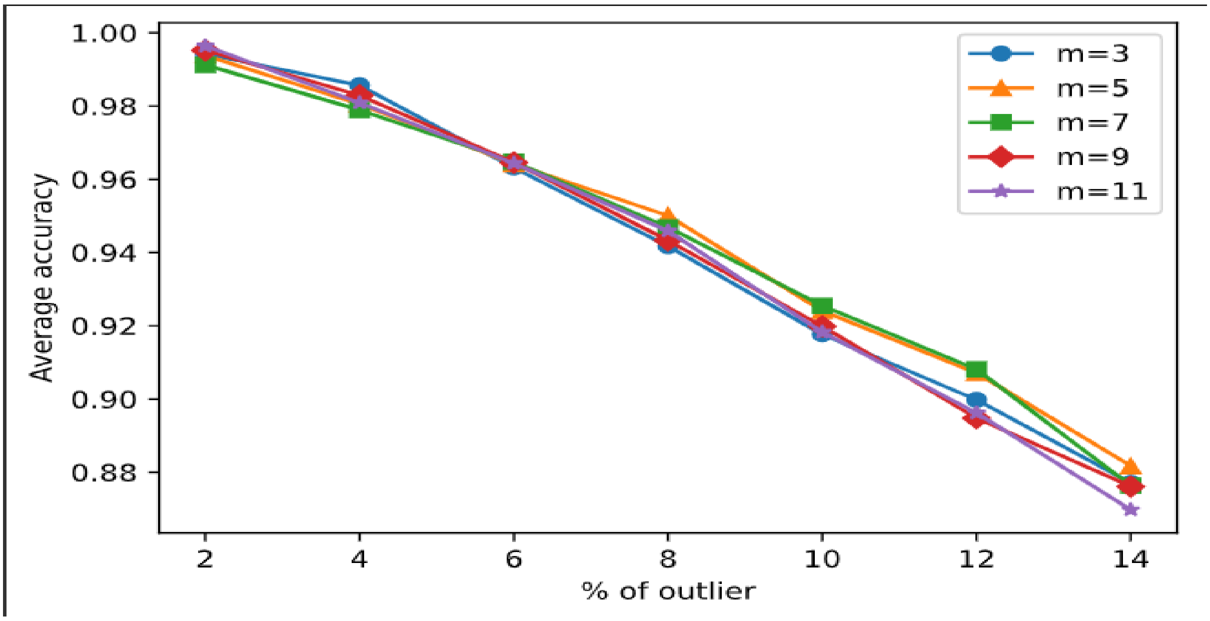
avg_recall

	m=3	m=5	m=7	m=9	m=11
2%	0.821782	0.831683	0.722772	0.801980	0.821782
4%	0.698020	0.599010	0.574257	0.603960	0.524752
6%	0.427152	0.447020	0.466887	0.413907	0.407285
8%	0.307692	0.409429	0.367246	0.295285	0.325062
10%	0.206759	0.266402	0.278330	0.202783	0.182903
12%	0.187086	0.251656	0.250000	0.125828	0.135762
14%	0.140426	0.181560	0.130496	0.117730	0.070922

- **SCREENSHOT FOR GRAPHS**

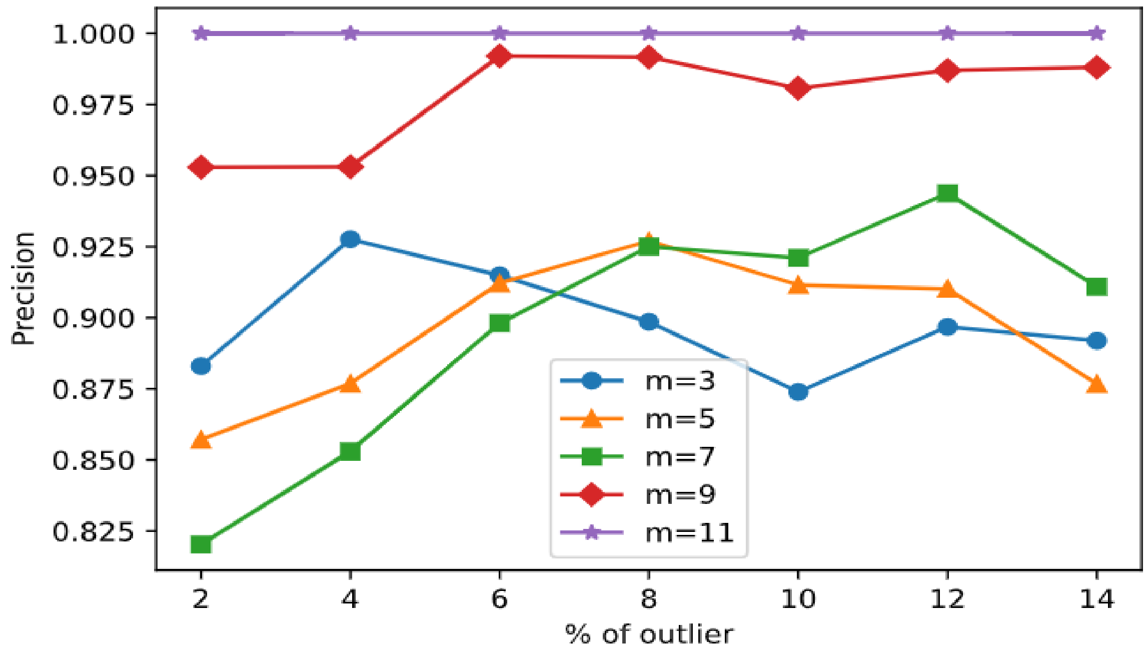
SCREENSHOT #1 : Average Accuracy vs %of outliers added

As we can see that as the m(it is ms) value increases the Accuracy for correctly detecting the outliers for each '%of outliers to be added' is decreases. Here so m=5 is detecting better from other assumption m values.



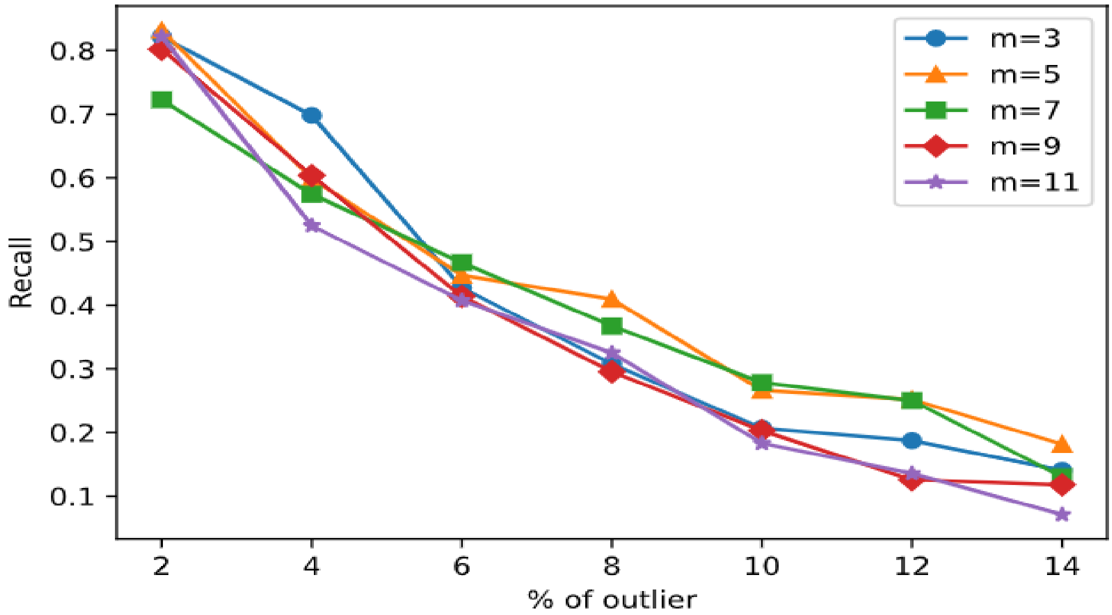
SCREENSHOT #2 : Average Precision vs %of outliers added

It is clearly seen that as the balanced curve is for m=5 also 3,7 as zig_zag and imbalanced curve is 9,11



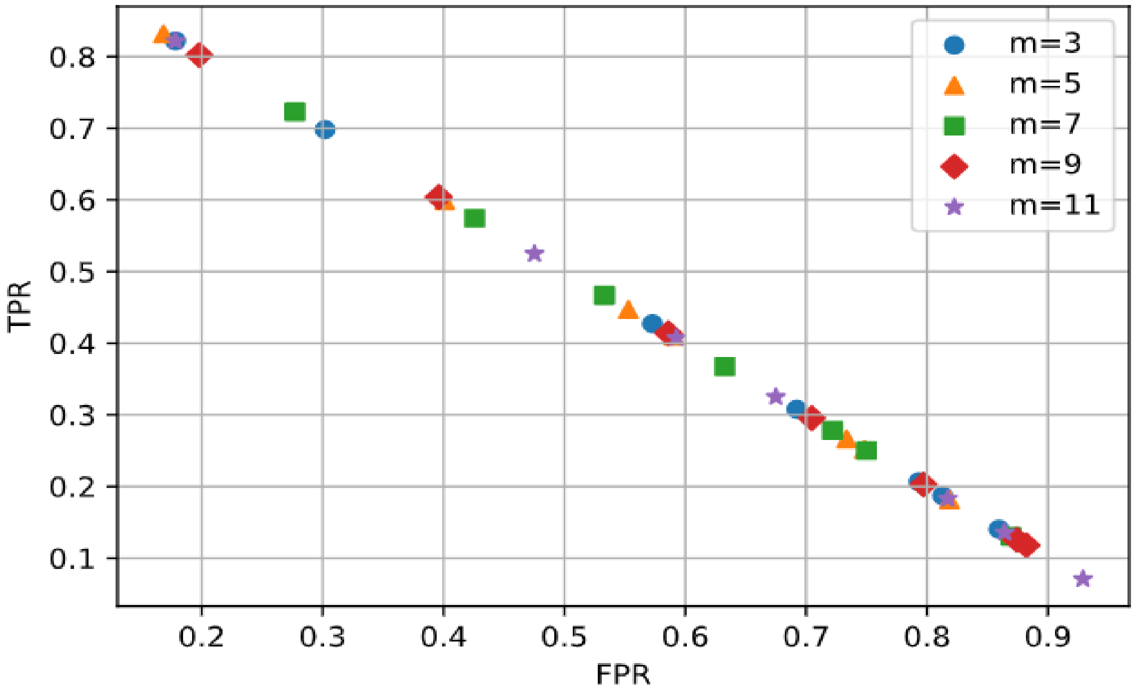
SCREENSHOT #3 : Average Recall vs %of outliers added

As we can see that as the m decreases the recall is better. As we see that m=5 also 3,7 are somehow better values compared to m=9,11 as the %of outliers added increases. As Recall is inversely proportional to FN, so recall should be 1 or nearby.



SCREENSHOT #3 : ROC Curve(FPR vs TPR)

This curve is used for determining the best m value with high tpr and low fpr. As we can see that m(MinSample)=5 also as well as m=3 has the best value for our model out of the rest assumption values of m.



CONCLUSION

- After going through various research papers, web articles and blogs, we have found out there are multiple ways to improve the accuracy, precision, recall, TPR. But we have successfully implemented the DBSCAN algorithm till now.
- We had plotted a graph successfully between % of outlier added vs average accuracy, precision, recall.
- Also, we had implemented ROC curve for determining the best value for min sample which has high tpr and low fpr.
- Here we can see that when % of outlier is increasing for each different minPts values the accuracy drops.
- As a Conclusion the DBSCAN algorithm is far better as compared to k-means and other clustering algorithm because it can nicely handle the outliers.