



MASTER OF COMPUTER and INFORMATION SCIENCES

COMP 809

Data Mining & Machine Learning

ASSIGNMENT 2

Semester 1, 2019

Name: Aditya Panikkar

Student ID: 18007189

Introduction and Motivation

The data provided has been collected from various sensors put on different positions on the body of the person. All this data has been collected from wearable sensors. The devices are also termed as IoT (Internet of Things) devices. The popularity of these wearable devices has increased and there are various companies that manufacture and sell such devices. All these devices produce data. The ability to extract this data, analyse it and draw conclusions from it would result in the better understanding of how the device works and its accuracy at providing the readings. This insight can then be used for either improving the device or marketing it which will result in making the lives of the people who wear them better while also providing monetary benefits to the companies that manufacture them. This is the primary goal of the mining activity, to check whether the readings provided can be pre-processed and fed into a classifier and check the ability of the classifier to accurately predict the activity. Also, linking the accurate readings with an activity could help in the sports domain. If an athlete is performing some activity and the values for the sensor do not fall within the known range, it could mean the athlete is doing something wrong and risk injury. All this data could be provided real time to perform analysis in sports.

In some cases, these could help in monitoring individuals. Old people could wear the devices and the reading could give an idea about their activity and abnormalities can be spotted. Thus, improving the quality of the sensor by understanding if the values provided are consistent and can be used to train a classifier will help in achieving in these tasks. For real time analysis, the model building time and computational cost is essential and keeping those at a minimum will be good.

Data set description

The Sports and Daily Activities data set was chosen. This data set contains data for 19 activities. The activities were:

1. Sitting (A1)
2. Standing (A2)
3. Lying on back and on right side (A3 and A4)
4. Ascending and Descending stairs (A5 and A6)
5. Standing Still in an elevator (A7)
6. Moving around in an elevator (A8)
7. Walking in a parking lot (A9)
8. Walking on a treadmill with a speed of 4 km/h (in flat and 15 deg inclined positions) (A10 and A11)
9. Running on a treadmill with a speed of 8 km/h (A12)
10. Exercising on a stepper (A13)
11. Exercising on a cross trainer (A14)
12. Cycling on an exercise bike in horizontal and vertical positions (A15 and A16)
13. Rowing (A17)
14. Jumping (A18)
15. Playing Basketball (A19)

the data was collected by 5 units placed on the body. Each unit provides 9 readings. The number of activities carried out were 19. Each activity was carried out by 8 people. Each person does the activity for 5 minutes. Readings were captured in 5 second segments

with a frequency of 25 Hertz. Hence, readings for each person have 60 segment files each having 125 rows.

Mining Schemes

Algorithms that have been considered are:

1) K-nearest neighbour

This is a lazy classifier method. As our data set is labelled and we are striving to achieve classification, a supervised learning algorithm is used. This algorithm assumes that all similar things exist in close proximity. It uses distance calculation between the data and its number of (k) neighbours. Usually, the Euclidean distance is computed but other distances such as hamming distance, Manhattan distance, Minkowski distance and Cheby Chev distance can also be calculated. Then it stores the index and distance of this to an ordered collection and sorts in ascending order. The first k entries are then picked and the mode of this is returned for classification. It is important that k is an odd number to avoid ties between distances. This algorithm can be used for text mining, climate forecasting, estimating soil and water parameters. In stock markets, this can be used to uncover market trends.

Strength: The strength of this algorithm is that it is easy to implement and suitable for both regression and classification. The algorithm is robustness to noisy training data. The algorithm is effective even if the training data is large. The algorithm has no training and it can also learn complex models easily.

Weakness: Firstly, we need to do several iterations to come up with an ideal value of k. If we keep it close to 1, the prediction becomes less stable and number of incorrect classifications increase. Increasing the value of k gives more stable and better prediction but this only happens to a certain point. After this, if we increase the value of k, the number of errors again increase. Therefore, selecting the correct value takes time. Also, with increasing the value of k, the computational cost increases as distance for each data sample has to be calculated for all the training samples. High dimensional data requires more storage and computation power.

2) Random Forest

Random Forest is also suitable for both classification and regression tasks. It is a very powerful supervised machine learning algorithm. The algorithm creates a forest based on a number of decision trees. The robustness of the prediction increases with an increase in the number of trees, thus providing a higher accuracy. To classify a new data instance, each tree gives a classification. This tree votes for this same class. The forest then chooses the classification having the greatest number of votes among all the trees in the forest. In the case of regression, the average of all votes is taken. The algorithm can be used in the banking domain to identify loyal customers. It can be used in stock markets to identify the stock behaviour and the expected loss and profit by purchasing the stock. It is also used in computer vision for image classification like Microsoft has used it in Xbox Kinect.

Pseudo Code

If the sample has X cases, then a sample of these X cases is taken at random but with replacement.

If there are Y features, a number $y < Y$ is specified in such a way that at each node of the forest, y features are selected at random out of Y . The best split on these y is used to split the node. This value of y is kept constant as the forest is grown.

Each tree is grown to the largest extent. No pruning is carried out.

Prediction of new data is done by aggregating the predictions of x trees

Strength

The algorithm efficiently manages the missing values and also maintains the accuracy for missing data. Even when there are a large number of trees in the forest, the random classifiers will not overfit the model. It also has the power to handle very large data sets that have high dimensionality.

Weakness

The ability at regression is not as good compared to its ability at classification tasks. It cannot predict beyond the range of training data given to it and the predictions given are not precise and continuous. The algorithm is prone to over fitting the datasets that are noisy. We have no control of what the model does. At best, we can try different parameters to do parameter tuning to change the behaviour of the model to how we like. The real time prediction of the model is also slow.

3) Support Vector Machine (SVM)

This algorithm looks at extremes of the dataset and draws a decision boundary called a hyperplane near the extreme points in the dataset. The extreme data points are the support vectors of the algorithm. So the Support Vector Machine segregates the two classes by a hyperplane. Estimating the optimal value of the hyperplane is very important for the accuracy of predictions. To determine the optimal value, the margin is computed. The distance between the support vector of one class and the hyperplane is called the margin. The ideal hyperplane is the one which has a maximum margin. The algorithm considers that only the support vectors are important for classification whereas the training examples can be ignored. All the data points in the dataset are referred to as vectors but only the extremes are support vectors. This method is also known as linear support vector machine (LSVM). They can be used in pattern recognition for machine fault diagnosis, image interpolation as well as for page ranking algorithms.

Strength

A Support Vector Algorithm is effective in cases where the data is in higher dimensional space. It is also very effective in cases where the number of dimensions are much more than the number of samples. The algorithm uses a subset of data points in the decision functions and hence is memory efficient.

Weakness

The support Vector Machines do not provide probability estimates. They have to be calculated using an expensive five-fold cross validation.

4) Logistic Regression

Logistic regression, despite the name suggests is a good algorithm for classification and not regression. The difference between logistical regression and linear regression is that logistic regression gives a discrete output whereas linear regression gives a continuous outcome. The prediction of the output is transformed using a non-linear function called as the logistic function. Logistic regression can be used for complex

non-linear datasets. The algorithm calculates the relationship between the label we desire to predict and the features in the dataset. It does so by estimating the probabilities using its logistic function. The value of the probabilities have to then be transformed to get values between 0 and 1 but never 0 and 1.

Strengths

Logistical regression does not require many computational resources. The algorithm is easy to implement and is very efficient to train.

Weakness

The algorithm highly relies on the presentation of the data. Since the outcome of the algorithm is only discrete, the algorithm can only predict categorical data. The algorithm is also prone to overfitting.

Experimental Study

The initial plan was to test all the algorithms listed above as they are suitable for the type of data we have. Then a survey of literature for the performance of the algorithms was undertaken. This showed that the main issue with using the k-nearest neighbour algorithm was arriving at the value of k. A smaller value would mean the mis classification chances increases. Increasing k also increases the computational cost as the distance needs to be calculated for each data instance. The better suited algorithms for a classification task are Support Vector Machines and Random Forests in comparison to k nearest neighbour (Thanh Noi & Kappas, 2017; Schilling, 2018). Hence, k-nearest neighbour has been eliminated and we proceed with Random forests and Support Vector Machines.

Data Pre processing

The data set had many different files and the amount of data was huge. This could not be directly fed into a classifier model. Hence some pre-processing was carried out.

The first step was to merge the data. As the data was present in multiple files, it had to be gathered in one single file to be able to carry out any mining activity. Python was used to pre-process the data. A code that would loop through each of the 60 files for one person was written. Simply collecting all this data for one person would yield 125x60 which is 7500 rows for one person. Number of tuples for all the people for one activity would then be 7500x8 which is 60,000 rows. If in the same way, data for all the activities were to be collected, the size of the data set would be 60,000x19 which is 1,140,000 rows. This kind of data would be a lot to process and fetch. Hence, aggregation was performed to reduce the data size. For each segment, another loop was written that did an average of the 125 values while fetching them. So essentially, we clubbed 125 rows into just one for each segment which resulted in just 60 rows per person per activity. The size of the aggregated data now was just 60x8x19 which is 9120 and this was considerably smaller than the original file. This data firstly took a lot less time to fetch in python and secondly, reduced the model building times in WEKA. After this, a function was written which added the 45 attributes as column headers for the aggregated data. This was done to ensure that attribute selection can be carried out in WEKA. The last part of pre-processing was to add the class names to the data.

Samples of the data is given to show how taking the average has affected the data.

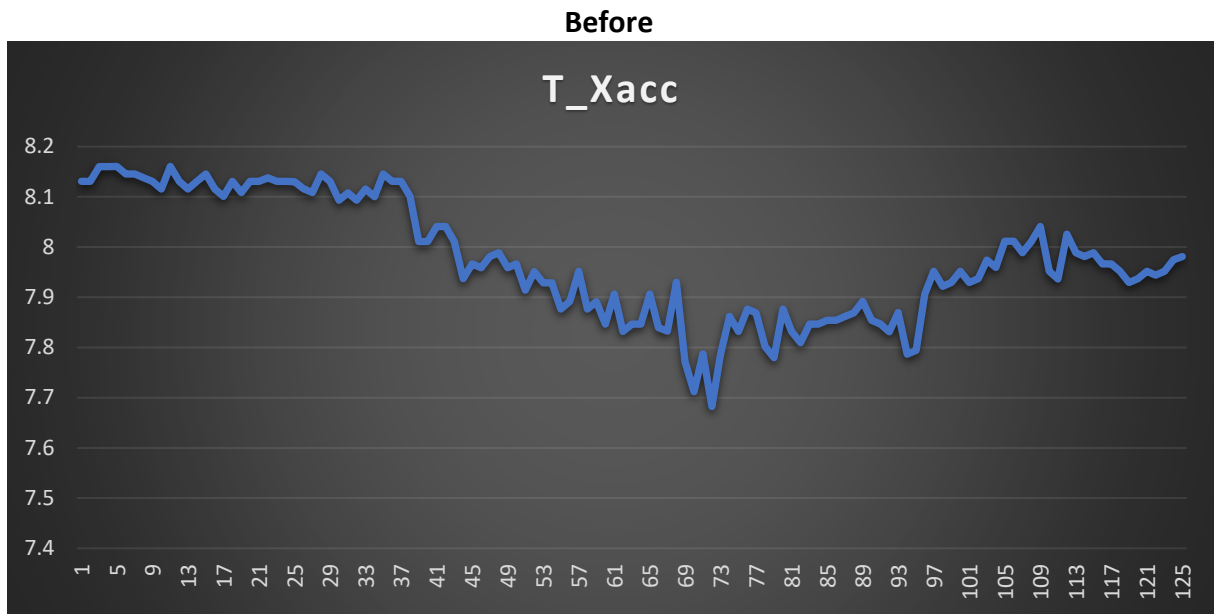


Figure 1: X component of acceleration from the unit on the torso from one segment

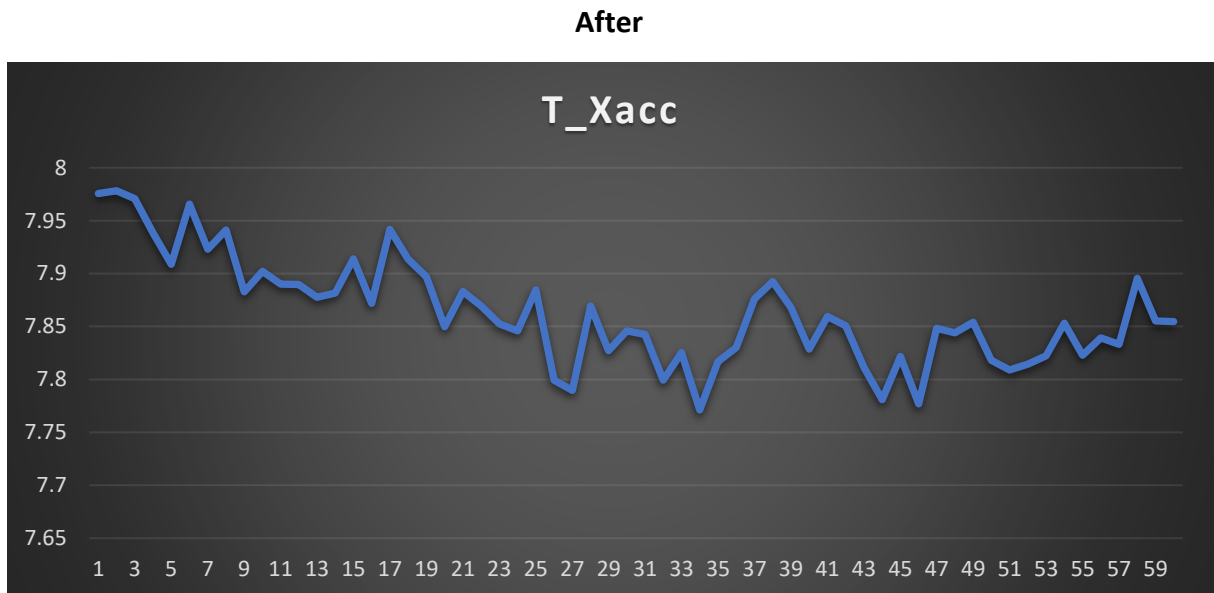


Figure 2: Aggregation of X component of acceleration from the unit on torso for all segments of a person

The first image above shows data from just 1 segment. Each segment has 125 features corresponding to the X component of acceleration. We have 60 such segments for each person for each activity. Reducing the 125 features to one per segment by taking the average has reduced the size of the data considerably, yet has not affected the nature or the pattern of the original data. The second image confirms this. The image contains one feature per segment and represents the graph of the data from each of the 60 different segments. The analysis which we perform for the transformed data set will also hold true for the original data set. There is no loss of patterns.

All the experiments have been performed using a 10-fold cross validation. The experimental plan is to have a model with high accuracy and a lower training time. These are performance measures taken to benchmark the selected algorithms. The experimental plan is given below:

- i. First step was to pre-process and reduce the size of the data to make processing faster and easier.
- ii. Each classifier was taken and ideal filters and parameters were iteratively determined to have a preliminary comparison of build times and accuracies among the chosen classifiers.
- iii. The parameters obtained in the preliminary step were then used to carry out further analysis and determine the best algorithm among the 3 chosen ones.
- iv. The number of experiments were controlled by:
 - a. First the pre-processed data was used with each classifier to establish a baseline for performance evaluation. A step by step filter and attribute selection was made and at any step, if the accuracy dropped or build time increased below the first in comparison to the previous iteration, no further tuning was done.
 - b. Only after identifying the ideal number of features and filter type and identifying a winner case, further parameter tuning was done to this specific case to further increase the performance.

The first step was to run each of the selected classifiers on the pre-processed data and check the model building times and accuracies for each. Weights were assigned to both these performance evaluators. As we want have identified the application domain as real time analytics, the model build times are crucial. Any model yielding a build time of more than 10 seconds has not been considered. For others, the time has been normalized to a scale of 10. The actual build time of the system will be deducted from 10 to yield the weight for time. This will ensure that a model taking more time gives a lesser value than a model having lesser build time. The accuracy has been assigned a weight of 1. The model with the highest weighted sum will win. The final weight calculation is:

$$(10 - \text{time in seconds}) + (\text{Accuracy} * 1)$$

Next, feature selection was carried out and various combinations were tested to see how it affects the performance measures that have been set.

1) Analysis of Support Vector Machine.

The pre-processed data was fed directly into the classifier. The model building time and accuracy were noted. After this, the correlational attribute eval filter was used to rank all the 45 attributes and the activity was repeated. As this yielded a better accuracy and build time, iterations were carried out to reduce the number of attributes to see if it yielded better performance. The first iteration with 32 parameters was done which reduced the build time with a relative minor drop in accuracy, hence the next iteration with 25 was carried out which reduced the model build time to 0.85 seconds but the accuracy by 2%. Parameter tuning was then done by changing the filter type to standardized training data. This provided the best accuracy and build time. Further iterations were not carried out as this method did not achieve lesser build times without compromising accuracies. The next filter used was the cfc subset eval and both the search methods were used. None of these gave a better accuracy than the correlation attribute eval. Cfs works by eliminating the attributes that have high co-relation between them but this caused the number of attributes to drop decreasing the accuracy of the model. Finally, AdaBoostM1 was

used with Correlational attribute eval which yielded the best accuracy but a build time of 12.77 seconds. The iterations have been summarized in the table below.

	No of Attributes	Performance Parameters		(10-Time)+Accuracy*1	Feature Selection		Parameter Tuning
		Model Building Time	Accuracy	Weighted sum	Filter Type	Search Method	
Support Vector Machine	45	1.73 seconds	96.98%	105.25	-	-	-
	45	1.23 seconds	97%	105.77	Correlational Attribute Eval	Ranker	-
	32	1.16 seconds	96.69%	105.53	Correlational Attribute Eval	Ranker = 32	
	25	0.85 seconds	95.03%	104.15	Correlational Attribute Eval	Ranker = 25	
	25	1.23 seconds	97.75%	106.52	Correlational Attribute Eval	Ranker = 25	Filter type = Standardized training data
	26	0.92 seconds	95.23%	104.31	Cfs Subset Eval	Best First	-
	35	0.9 seconds	94.84%	103.94	Cfs Subset Eval	Greedy Stepwise	-
	46	12.77 seconds	98.46%	95.69	Correlational Attribute Eval	Ranker	AdaBoostM1

Figure 3: Performance Table for Support Vector Machine

2) Analysis of Logistic Regression

Similar iterations were carried out with Logistic Regression as the classifier. The Optimum no of attributes and parameters are given below.

	No of Attributes	Performance Parameters		(10-Time)+Accuracy*1	Feature Selection		Parameter Tuning
		Model Building Time	Accuracy	Weighted sum	Filter Type	Search Method	
Logistic Regression	45	40.39 seconds	98.62%	N/A	-	-	-
	45	41.6 seconds	98.62%	N/A	Correlational Attribute Eval	Ranker	-
	35	34.04 seconds	97.31%	N/A	Cfs Subset Eval	Best First	-
	35	29.97 seconds	97.37%	N/A	Cfs Subset Eval	Greedy Stepwise	-
	45	11.18 seconds	97.89%	N/A	Correlational Attribute Eval	Ranker	MaxBoostingIterations =50
	45	3.64 seconds	97.88%	104.24	Correlational Attribute Eval	Ranker	MaxBoostingIterations =50, AIC true
	32	2.77 seconds	97.51%	104.74	Correlational Attribute Eval	Ranker = 32	MaxBoostingIteration =50, AIC true

Figure 4: Performance Table for Logistic Regression

3) Analysis of Random Forests

The same was carried out for Random Forest and the best parameters have been highlighted below.

	No of Attributes	Performance Parameters		(10-Time)+Accuracy*1	Feature Selection		Parameter Tuning
		Model Building Time	Accuracy	Weighted sum	Filter Type	Search Method	
Random Forest	45	5.56 seconds	99.58%	104.02	-	-	-
	45	5.15 seconds	99.61%	104.46	Correlational Attribute Eval	Ranker	-
	32	4.88 seconds	99.61%	104.73	Correlational Attribute Eval	Ranker = 32	
	35	5.05 seconds	99.46%	104.41	Cfs Subset Eval	Best First	-
	34	5.16 seconds	99.41%	104.25	Cfs Subset Eval	Greedy Stepwise	-
	32	5.19 seconds	99.63%	104.44	Correlational Attribute Eval	Ranker = 32	No of seeds = 5
	32	1.58 seconds	99.56%	107.98	Correlational Attribute Eval	Ranker = 32	Num iterations = 30, No of seeds = 5
	32	1.57 seconds	99.58%	108.01	Correlational Attribute Eval	Ranker = 32	Num iterations = 30, No of seeds = 9
	32	3.65 seconds	99.63%	105.98	Correlational Attribute Eval	Ranker = 32	Num iterations = 70, No of seeds = 9

Figure 5: Performance Table for Support Random Forest

In the preliminary step, different filter types were used to balance the performance parameters. Random Forest gave the highest accuracy and the least model building time. The best parameters that were identified for each of the three algorithms that gave the best accuracy and build time parameters were used in stage 2 with Principal Components Analysis (PCA). This was chosen as PCA reduces the number of dimensions in the data set while ensuring that the highest amount of information is retained. The results after this are given below.

	No of Attributes	Performance Parameters		(10-Time)+Accuracy*1	Feature Selection		Parameter Tuning
		Model Building Time	Accuracy	Weighted sum	Filter Type	Search Method	
Support Vector Machine	25	3.13 seconds	96%	102.87	Principal Componants	Ranker	Filter type = Standardized training data
Logistic Regression	25	2.32 seconds	91.64%	99.32	Principal Componants	Ranker	MaxBoostingliteration =50, AIC true
Random Forest	25	1.5 seconds	97.97%	106.47	Principal Componants	Ranker	Num iterations = 30, No of seeds = 9

Figure 6: Performance Table after stage 2 for all selected algorithms

Chart for change in weighted sum is given below which shows the decrease is the least for Random Forest.

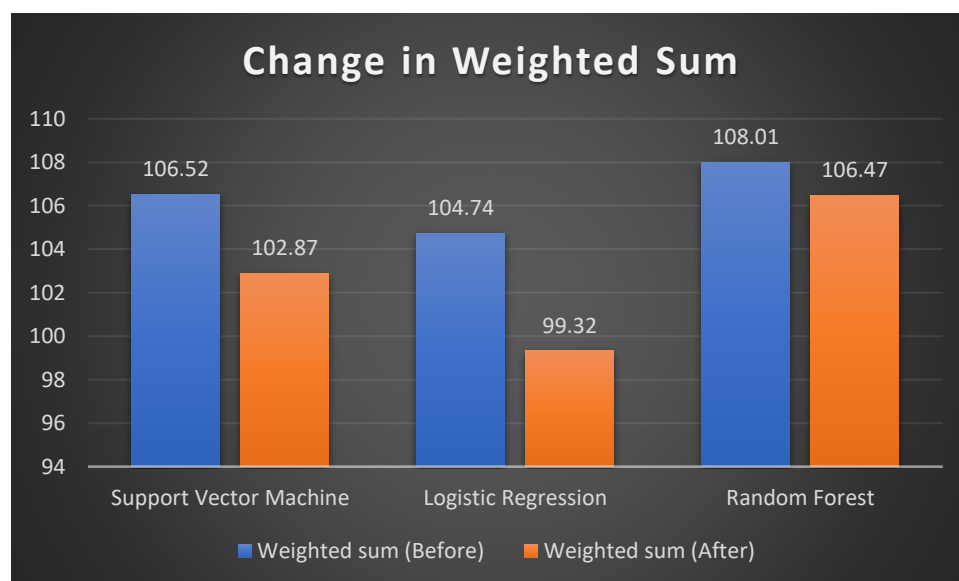


Figure 7: Change in weighted sum for the Algorithms after PCA.

The weighted sum for Random Forest after using PCA is the highest. The model building time for SVM after PCA was applied and attributes were reduced increased from 1.23 seconds to 3.13 seconds. For logistic regression, though the build time did not increase substantially, the accuracy dropped from 97.51% to 91.64%. Random Forest had a reduced build time with the least impact in accuracy. Therefore, the winning algorithm is Random Forest.

The parameter tuning in each case has increased the performance of the algorithms. In the case of Support Vectors, changing the filter type to Standardized data both increased Accuracy and decreased build time. In the case of logistic regression, the build time decreased to the least and the accuracy dropped only slightly. The weighted sum of this case was still the highest compared to the others. This was achieved successively, first by changing the MaxBoostIterations to 50, and then setting AIC to true. For Random forest, Reducing the Number of Iterations to 30 and increasing the seeds to 9 gave the least build time among all iterations with the accuracy slightly dropping by 0.03%. The sum of this case still was higher compared to any other cases. Therefore, we can conclude that in each case, the parameter tuning has helped us increase the weighted sum.

Even after PCA, the first algorithm as per the weighted sum is still Random Forest, runner-up is Support Vector and the third is Logistic Regression. The comparison Graphs for Model Building time, Accuracy and weighted sum are given below.

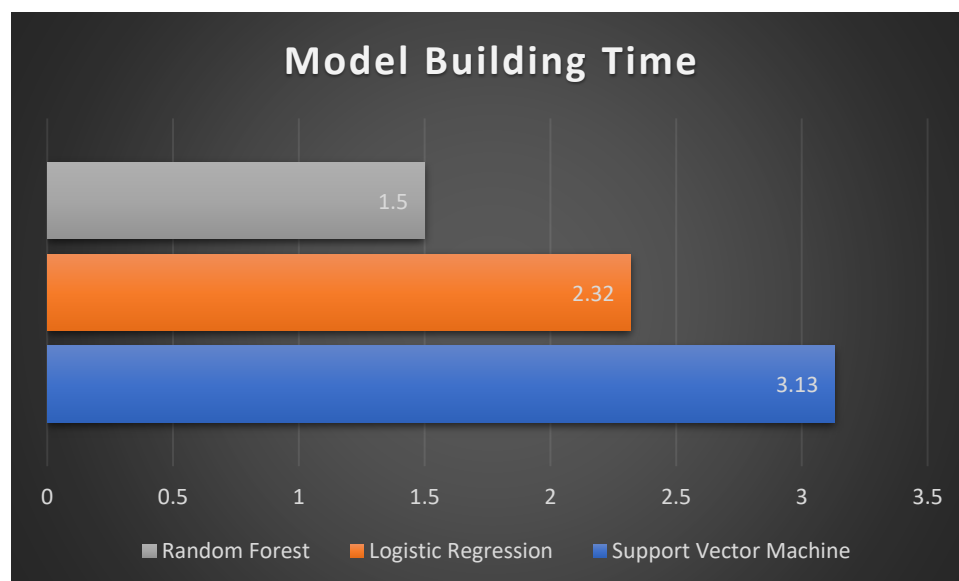


Figure 8: Model Build Times for the Algorithms.

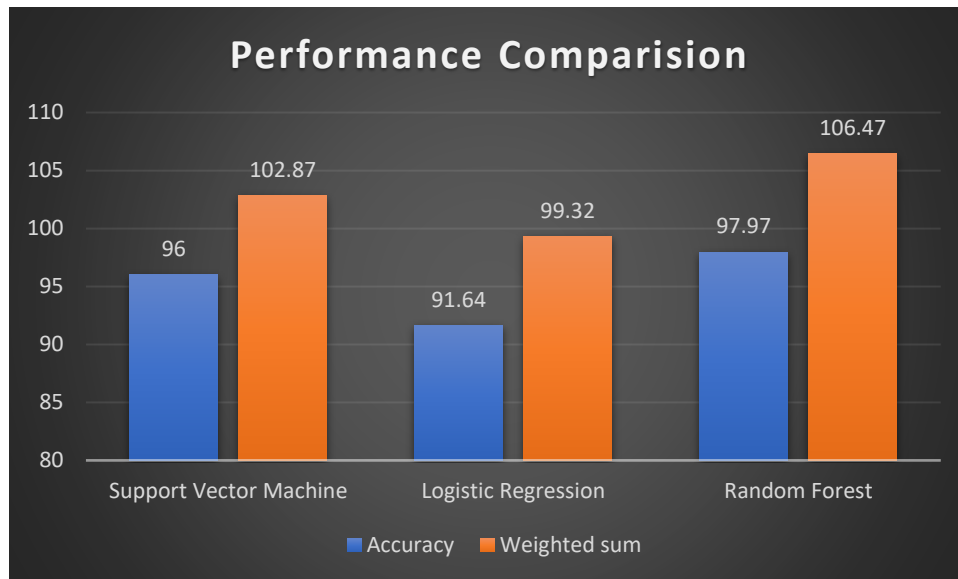



Figure 9: Model Build Times for the Algorithms.

Therefore, looking at the change in weighted sum, the Model Build time and the Accuracy, we can conclude that Random Forest is the best performing algorithm from the rest. The final Ranking is:

- 1) Random Forest
- 2) Support Vector
- 3) Logistic Regression

References

Yukse, M., & Barshan, B. (2011). Human Activity Classification with Miniature Inertial and Magnetic Sensor Signals. Retrieved from http://kilyos.ee.bilkent.edu.tr/~billur/publ_list/eusipco11_1.pdf

Schilling, M. (2018). Comparison of machine learning techniques in email spam detection - DEV Community . Retrieved from <https://dev.to/matchilling/comparison-of-machine-learning-techniques-in-email-spam-detection--2p0h>

Thanh Noi, P., & Kappas, M. (2017). Comparison of Random Forest, k-Nearest Neighbor, and Support Vector Machine Classifiers for Land Cover Classification Using Sentinel-2 Imagery. *Sensors*, 18(2), 18. doi: 10.3390/s18010018