Human Activity Recognition using Machine Learning Algorithms

Roya Latifi
School of Economics,
University of California, Los Angeles
Los Angeles, California 90095
royalatifi@g.ucla.edu

Yiran Sun
School of Economics,
University of California, Los Angeles
Los Angeles, California 90095
yiran8sun@g.ucla.edu

Yiping Liu
School of Economics,
University of California, Los Angeles
Los Angeles, California 90095
yipingliu@g.ucla.edu

***Abstract*—This paper analyzes the ability of 7 unique machine learning models to identify the specific movement or action of a person based on sensor data. We use human activity recognition using smartphone data from the UCI machine learning repository. Model performance is evaluated using accuracy. We have implemented Logistic regression with Ridge and Lasso penalization, Linear discriminant analysis (LDA), Support vector Machine (SVM), KNN, Neural Networks and Naive Bayes for our classification problem. We find that before regularization, optimized Support Vector Machine (SVM) performs the best and achieves an accuracy of 98.50% and after PCA regularization, KNN performs the best and archives an accuracy of 95.97%. We found that PCA regularization does not improve our models and does not bring an expected positive influence.**

***Index Terms*—Machine learning, Probability, Classification, Logistic Regression, KNN, Neural Networks, Support vector machines, Random Forest, LDA, Naive Bayes, PCA**

I. Introduction

In daily life, people perform various daily activities with different speeds and directions. Even for the simplest gestures, such as running and walking, the activities can be separated by at least three-axis accelerations and angular velocity. With the technical skills learned in machine learning and big data, we would like to see if we can train our model to make correct judgments on people's different gestures based on the features in people's fundamental activity performance and make the right classification of their activities. We choose seven machine learning algorithms in total to predict human activity (1-Walking, 2-Walking upstairs, 3-Walking downstairs, 4-Sitting, 5-Standing or 6-Laying) by using the smartphone's sensors and apply dimension reduction methods to further compare each models' performance and features. In this project, we developed seven machine learning models such as Logistic Regression with different alpha settings, LDA, SVM, Random Forest, KNN, Neural Networks, and Naive Bayes to solve a multi-classification problem. Our aim is to compare different machine learning algorithms for human physical activity classification. With an accuracy of 98.50%, SVM was able to outperform all the other models that we have implemented.

II. Data

A. Dataset description

We utilized the dataset named "Human Activity Recognition" from the UCI Machine Learning Repository which is a feature-engineered dataset and which was made available in 2012. The "Human Activity Recognition" database is composed of 10,299 observations and 564 variables. The dataset was built from the recordings of 30 study participants aged between 19 and 48 years old performing one of six standard activities while carrying a waist-mounted smartphone with embedded inertial sensors.

The movement data recorded was the x, y, and z accelerometer data (linear acceleration) and gyroscopic data (angular velocity) from the smartphone, specifically a Samsung Galaxy S II[1]. The data was preprocessed and already supplied with 563 engineered features.

Our objective is to classify activities into one of the six activities performed. As for 563 variables in our database, 561 of them are features obtained from the accelerometer and gyroscope and some calculation methods. The remaining is one "subject" column identifying the participants. The dependent variable is named "activity" at last, including six activities we mentioned earlier, which are walking, walking upstairs, walking downstairs, sitting, standing, and laying.

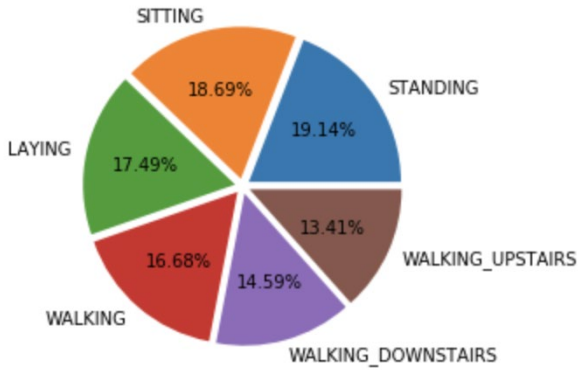


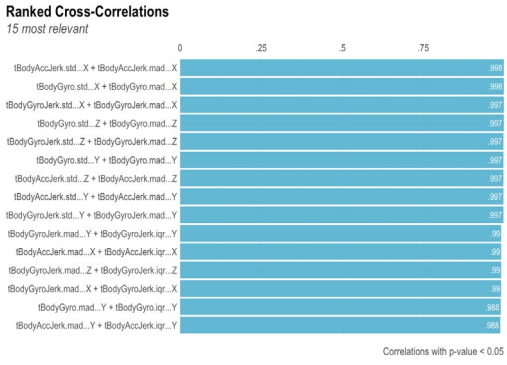
Figure 1

Before further processing the data, we confirm that there is no missing value in our dataset. We also figure out that the original training and testing data from UCI was split based on experiment subjects, which may cause inaccuracy in final activity because of individual differences. So, we add one more step to integrate the training and testing data as a whole and split it by assigning subjects randomly into two sets. Then, we get our data for model establishment. After checking our dataset, we have seen that our data has been split randomly as we can observe from Figure 2.

Figure 2

B. Dataset exploration

Based on the total amount of our data, it's important for us to figure out the logic and relationship among variables. We explore the independent variables in the first step. The variables from this dataset were gathered from experiment participant's acceleration and velocity when performing different gestures and processed by some scientific calculation methods. We divide the variables into 4 groups, and then check their correlations in each group. After visualizing the pairs of variables with the highest correlations, we figure out that (1) variables in the same axis are more likely to have a higher relationship; (2) in magnitude perspective, body acceleration has the highest correlation with gravity acceleration; (3) in the fast Fourier transform for angel signals, energy variables in X-axis tend to generate higher correlation.



[1] Deep Learning for Time Series Forecasting. (n.d.). Google Books.

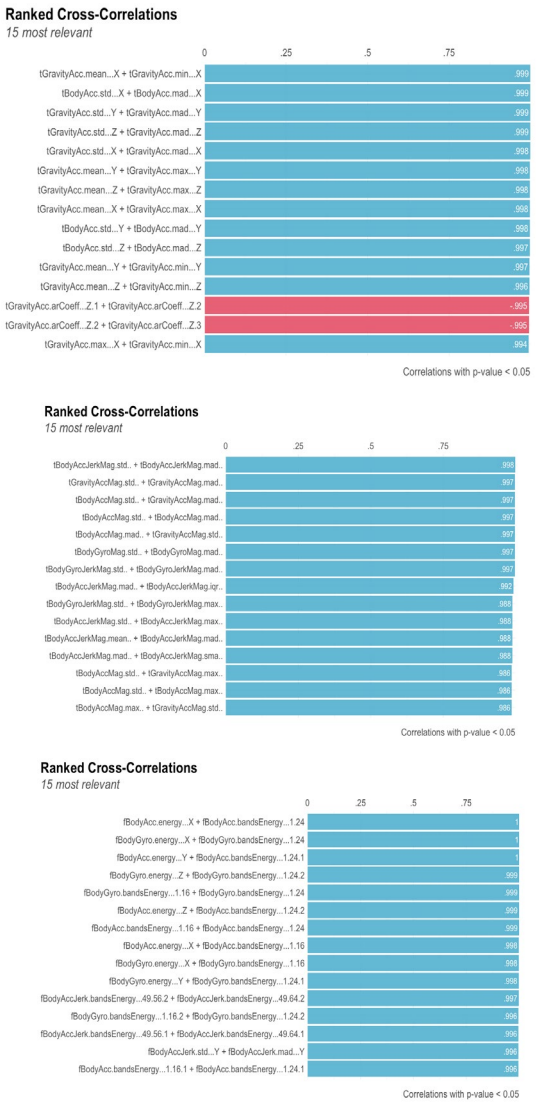


Figure 3

After exploring independent variables, we further checked the predictable variable "Activity" in the dataset. There are 6 types of activities in this column, describing people's daily activities. As shown in the following histogram, the distribution of the predictor variable is quite even for these six activities and they have been assigned to all participants in the experiment.

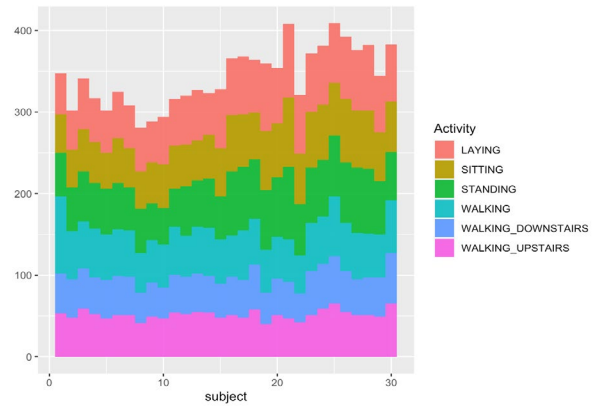


Figure 4

Since our data was feature-engineered already, we did not have to perform normalization on our dataset.

III. Methodology

The predictable variable in this dataset has six features, including walking, walking upstairs, walking downstairs, sitting, standing, and laying. Since it is a multi-classification problem and the predictors have five misclassification potentials, we would like to focus on the model's overall accuracy rather than the recall and precision to compare the models and select the best one.

Besides constructing the models and extracting their accuracy from the basic training and validation dataset, we include the cross-validation procedure for the selection of parameters in our models to get rid of randomness and increase the credibility in our model. With the cross-validation method, we can avoid overfitting in our training data and hence make a generalization of our models. Almost all of our models were implemented using the optimal parameters that we have found through 5-fold cross-validation.

For our multi-classification problem, we have adopted Logistic Regression with different alpha settings, LDA, SVM, Random Forest, Neural Networks, and Naive Bayes. These are supervised machine learning classifier algorithms and can be used to predict physical activity based on the accelerometer signals. Also, we utilized an unsupervised

model KNN. Most of the models that we have selected combined with the usage of cross-validation can avoid the overfitting problem that might be caused by the high dimension nature of our dataset. Finally, since our dataset is high dimensional and has a lot of features, we further implemented the PCA regularization to get rid of the redundant information and to reduce the dimension.

IV. Model Implementation

A. Evaluation metrics

We split our data into ⅘ and ⅕ for a training set and testing set, respectively. For each model, we optimized hyper-parameters by using the cross-validation method and after finding the optimal parameters, we trained our models on the randomized training sample to avoid the classifier getting biased to a particular pattern in time series. Once each model's hyper-parameters were optimized and each model was trained on a training sample, we evaluated each model's performance over the testing set and reported the results. Accuracy on the testing sample is used to evaluate and compare the models.

B. Models

1. Logistic Regression

Logistic Regression is one of the most common models used when the dependent variable is categorical. In logistic regression, there are two types of penalties, L1 and L2. L1 penalty, which is also named as Lasso regression, adds a penalty on the sum of non-zero coefficient absolute values. Lasso introduces a new hyperparameter, alpha, to penalize weights. Ridge takes a step further and penalizes the model for the sum of the squared value of the weights. Thus, the weights not only tend to have smaller absolute values but also really tend to penalize the extremes of the weights, resulting in a group of weights that are more evenly distributed.

In this dataset, the output of this model returns the probability of the type of activities that the experiment participants are performing. We built both regression models by changing the value of alpha, which is 1 for lasso and 0 for the ridge. The optimal value of lambda selected by the 5-fold cross-validation is 0.01 for both lasso and ridge models. Overall, ridge regression generates better performance than lasso. The accuracy of training data for the L1 penalty is 96.24%, compared to 98.19% for the L2 penalty. And in testing data, the ridge has an accuracy of 97.43%, compared with lasso at 95.97%. As shown in the confusion matrix, the precision of sitting and standing is lower than the other activities in both penalty methods. And these two are more likely to be confused between each other, which is probably caused by the similarity of the features when people are performing these two activities.

Logistic regression Confusion matrix:

	Reference					
Prediction	LAYING	SITTING	STANDING	WALKING	WALKING_DOWNSTAIRS	WALKING_UPSTAIRS
LAYING	1941	8	0	0	0	0
SITTING	0	1596	139	0	0	0
STANDING	0	164	1761	0	0	0
WALKING	0	0	0	1695	25	20
WALKING_DOWNSTAIRS	3	0	0	2	1326	31
WALKING_UPSTAIRS	0	9	6	25	55	1493

Lasso Regression

	Reference					
Prediction	LAYING	SITTING	STANDING	WALKING	WALKING_DOWNSTAIRS	WALKING_UPSTAIRS
LAYING	378	4	0	0	0	0
SITTING	0	343	21	0	0	0
STANDING	0	24	360	0	0	0
WALKING	0	0	0	336	0	2
WALKING_DOWNSTAIRS	0	0	0	0	286	1
WALKING_UPSTAIRS	0	0	0	1	0	304

Ridge Regression

2. LDA

Compared with logistic regression, Linear Discriminant Analysis (LDA) solves the problem that the parameter of the logistic regression model may not be stable and returns the

predicted classes based on the given observations by involving the determination of linear equations. The discriminant function is built by the sum of multiplication between discriminant coefficients and variables. By choosing the optimal coefficients, LDA maximizes the distance between means of multiple categorical variables. Unlike PCA, LDA aims to discriminate between the classes like SVM, rather than finding linear combinations of features that capture the data's variance. The model separated the groups with Bayes decision boundaries. After running the model, its performance meets our expectations since in both training and testing dataset, LDA generates a higher accuracy than logistic regression. The accuracy of training data is 98.43%, and increases to 97.57% for the testing set.

Linear Discriminant Analysis Confusion Matrix:

	Reference					
Prediction	LAYING	SITTING	STANDING	WALKING	WALKING_DOWNSTAIRS	WALKING_UPSTAIRS
LAYING	378	5	0	0	0	0
SITTING	0	345	23	0	0	0
STANDING	0	21	358	0	0	0
WALKING	0	0	0	337	0	0
WALKING_DOWNSTAIRS	0	0	0	0	285	0
WALKING_UPSTAIRS	0	0	0	0	1	307

3. Random Forest

Random Forest algorithm is a supervised classification algorithm. It consists of a large number of individual decision trees and it finds the optimal values by choosing the majority among multiple decision trees. Random forest is very attractive in our dataset since we have high dimensions in our data and since random forest can do nonparametric estimation and predictors selection. We have used 5 fold cross-validation to find the optimal mtry value which is the number of sample variables selected for each tree and the number of trees. Mtry = 33 and number of trees = 500 combinations provided the highest accuracy and lowest OOB estimate of error rate. With the hyperparameters that we have obtained using cross-validation, we have trained our training sample and achieved an accuracy of 97.86% in the testing sample. The random forest became our third best model in terms of accuracy.

Random Forest Confusion Matrix:

	Reference					
Prediction	LAYING	SITTING	STANDING	WALKING	WALKING_DOWNSTAIRS	WALKING_UPSTAIRS
LAYING	365	0	0	0	0	0
SITTING	0	359	10	0	0	0
STANDING	0	17	395	0	0	0
WALKING	0	0	0	326	0	0
WALKING_DOWNSTAIRS	0	0	0	7	267	3
WALKING_UPSTAIRS	0	0	0	1	6	304

4. Support Vector Machine

In multi-class classification, SVM takes data points and tries to find a line that maximizes the separation between classes. It can produce large accuracy with less computing power. There are two approaches that SVM can take in multi-class classification. One is using one-vs-one and breaking down the problem to multiple binary classification cases, so reducing it to a multiple binary classification problem, and the other one is using the One-vs-rest approach which divides the data points as class x and rest.[2]

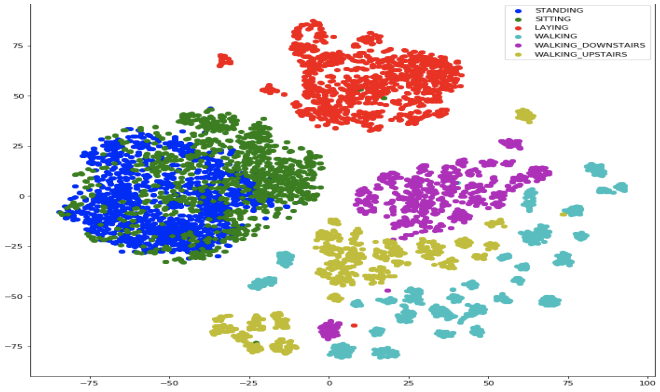


Figure 5

In Figure 5, we can see the distribution of the classes in our data. The plot shows that the classes are very distinguishable for almost all classes but also shows it is a bit

[2] https://machinelearningmastery.com/one-vs-rest-and-one-vs-one-for-multi-class-classification/

harder to differentiate between sitting and standing. With this kind of distribution, our expectation was that the SVM model would classify most of the activities correctly since drawing a line and separating between classes seems achievable.

We have tuned in our parameters using 5-fold Cross-Validation and found the optimal number of parameter C, which trades off correct classification of training examples against maximization of the decision function's margin[3], as 0.5. The optimal model that was trained on the training sample achieved an accuracy of 98.50% on the testing sample. This high accuracy proved our hypothesis that SVM would classify most of the activities correctly. When we look at the confusion matrix below, we can see that all activities except sitting and standing were classified accurately. The sitting and standing classes were less separable as we have seen from Figure 5, so it is understandable that SVM struggled classifying some data points belonging to those classes. Nevertheless, SVM achieved a very high accuracy and became the best classification model.

Support Vector Machine Confusion Matrix:

	Reference					
Prediction	LAYING	SITTING	STANDING	WALKING	WALKING_DOWNSTAIRS	WALKING_UPSTAIRS
LAYING	365	0	0	0	0	0
SITTING	0	363	18	0	0	0
STANDING	0	13	387	0	0	0
WALKING	0	0	0	334	0	0
WALKING_DOWNSTAIRS	0	0	0	0	273	0
WALKING_UPSTAIRS	0	0	0	0	0	307

5. K-Nearest Neighbors

The K-nearest neighbors algorithm (KNN) is an unsupervised machine learning algorithm and also a non-parametric approach. It assumes that similar things exist in close proximity[4] and 'feature similarity' to predict the values of any new data points and it does not embrace a learning approach. It looks into training data for the K points that are closest to a new observation. After finding the K closest points (through Euclidean distance), KNN will make a prediction about the new observation's class by looking at the most common class in the k-nearest neighbors. KNN normally requires data normalization to perform better, but since our dataset was already feature engineered and normalization was applied, we did not implement a normalization process.

Since KNN is a distance-based algorithm, in higher dimensional space like our dataset, the cost to calculate distance becomes expensive and time consuming. Thus, to determine the k value, the number of neighbors, we have only tried k values equal to 2, 5, 10, 15, 20 in our 5 fold cross-validation method to get the optimal k value. With k =15, KNN achieved an accuracy of 96.02% on the testing sample. The KNN performs very well but it is not as good as our previous models. The reason is that KNN does not tell us which predictors are important and as we have mentioned previously, since the cost to calculate distance becomes expensive, it also affects the performance of the model. Also, as we have seen from Figure 5, some activity classes like standing and sitting were very close. So KNN, being a lazy learner, cannot make the right classification for those data points since it relies on the assumption that similar things exist in close proximity. The confusion matrices imply that the uncertainty between walking downstairs-walking and sitting-standing activity pairs is the cause of the predictive errors. But overall, the accuracy is still very high.

KNN Confusion Matrix:

	Reference					
Prediction	LAYING	SITTING	STANDING	WALKING	WALKING_DOWNSTAIRS	WALKING_UPSTAIRS
LAYING	364	1	0	0	0	0
SITTING	1	321	18	0	0	0
STANDING	0	54	387	0	0	0
WALKING	0	0	0	334	6	1
WALKING_DOWNSTAIRS	0	0	0	0	266	0
WALKING_UPSTAIRS	0	0	0	0	1	306

[3] https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html

[4] https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761

6. Neural Network

Neural networks are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns.[5] This algorithm that works like our brain is composed of three parts. The first part is the input layer which takes the data into the algorithm. In the second part, hidden layers, that are made of several units called neurons, optimize the weights of the inputs and process the inputs through the activation functions to something that the output layer can use and the third part gives the output out. The attractive part about the neural network is, it can learn from the past on its own unlike KNN and they can generate more accurate output.

We have adapted a feedforward neural network for our dataset and we have tried different combinations of hidden layers and neurons to find the best model in terms of accuracy. Since our dataset has high dimensions and since our computers are not strong enough for computing more complicated combinations of hyperparameters, we have tried simpler neural network models with low number of neurons and hidden layers. In all of the models we tried, we used the logistic activation function. Table 1 shows the neuron and hidden layer combinations we have applied and the accuracy we got on the testing samples.

With one hidden layer and 6 neurons in that hidden layer, we were able to achieve an accuracy of 97.03%. The accuracy that neural networks achieved is very good but it is lower than some of the models that we have implemented previously, like SVM. Neural networks requires much more data than traditional algorithms and lower dimensions in order to learn from the data better. However, our dataset is a relatively small dataset for this algorithm and also our dimension is relatively high. Therefore, the denser model did not become the best model in our implementations.

Hidden Configuration	Training Accuracy	Testing Accuracy
(2)	**99.37 %**	**95.43 %**
(6,3)	**99.04 %**	**95.53 %**
(6)	**99.30 %**	**97.03 %**
(10)	**99.73 %**	**96.94 %**
(6,6)	**99.19 %**	**96.26 %**

Table 1

Neural Networks Confusion Matrix:

Prediction \ Reference	0	1	2	3	4	5
0	365	0	0	0	0	0
1	0	359	25	0	0	0
2	0	16	379	0	0	1
3	0	0	1	332	2	5
4	0	1	0	2	267	4
5	0	0	0	0	4	297

7. Naïve Bayes

Naive Bayes Classifier utilizes Bayesian theory, a simple probabilistic classifier that calculates the posterior probability with independent assumptions based on likelihood and prior probability. And after we get the conditional distribution for each class, we follow the basic decision rule: maximize the posterior probability using prior probability. The Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, it is known to outperform even highly sophisticated classification methods.

In our case, Naive Bayes Classifier was trained through 5-fold cross-validation while keeping tuning parameters ‘fL’ and ‘adjust’ constant. The accuracy on the training set is 0.7763 and the following table is the confusion matrix of the model prediction on the testing data. We can observe that ‘SITTING’ and ‘STANDING’ are easily misclassified by the model. The overall model is giving 0.767 accuracy on the testing data.

[5] https://wiki.pathmind.com/neural-network

Naïve Bayes Confusion Matrix:

```
Confusion Matrix and Statistics

                    Reference
Prediction           LAYING SITTING STANDING WALKING WALKING_DOWNSTAIRS
  LAYING                361      41       10       0                  0
  SITTING                 1     287      191       0                  0
  STANDING                1      47      201       0                  0
  WALKING                 0       0        0     252                 10
  WALKING_DOWNSTAIRS      0       0        0      40                214
  WALKING_UPSTAIRS        2       1        3      42                 49
```

C. PCA Dimension Reduction and Model Fitting

Principal Component Analysis (PCA) is a dimensionality reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set. Reducing the number of variables of a data set naturally comes at the expense of accuracy, but the trick in dimensionality reduction is to trade a little accuracy for simplicity.

The main feature of our dataset is that it contains more than 500 variables, which is quite large compared to our sample size. Recalling the curse of dimensionality problem, we utilized Principal Component Analysis (PCA) to reduce its dimension. From the components proportion plot, we observed that almost 95% of the variance can be explained by the first 100 variables. Decreasing from 560 variables to 100 components, is a great improvement we made and we decided to go with these components for our next step, which is applying the above models again with the new dataset we just built, in order to find out if the regularization improves the model performance.

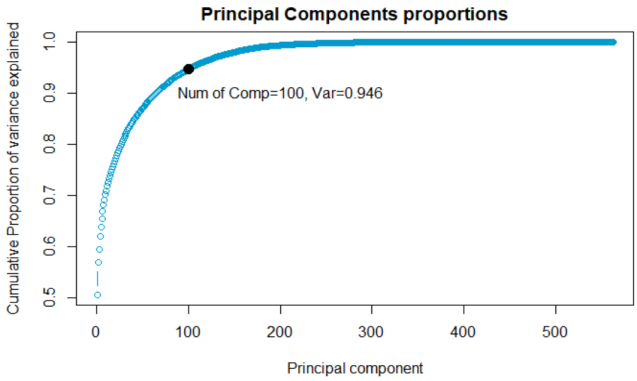


Figure 7

V. Results

Once each model's hyperparameters had been optimized and PCA regularization applied, we combined all the results in one table to compare the performance of each model as in table 1.

Models	Accuracy (Training Set)	Accuracy (Testing Set)	PCA Accuracy
Logistic Regression	(L1 Penalty) 96.24%	95.97%	95.39%
	(L2 Penalty) 98.19%	97.43%	95.58%
LDA	97.91%	98.20%	93.46%
Random Forest	100%	97.86%	94.32%
Naive Bayes	77.63%	76.70%	85.34%
KNN	96.46%	96.02%	95.97%
SVM	99.47%	98.50%	95.73%
Neural Networks	99.30%	97.04%	94.51%

Table 2

As previously noted, all models have a high accuracy score except Naive Bayes and we would make some reasonable assumptions why this happened. We would also elaborate more on the findings after we reduced the dataset dimension and discuss them in detail.

Focusing on the original dataset first. Before we applied PCA regularization, Support Vector Machine (SVM) is giving the highest accuracy on the testing sample of 0.9850, slightly higher than Linear Discriminant Analysis (LDA)'s 0.982. Following them, the Random Forest model, the Logistic Regression with Ridge penalty, and the Neural Networks model also had relatively high accuracy, all higher than 0.97. As things went on, the K-Nearest Neighbors and

the Logistic Regression with Lasso penalty had a close outcome which gives about 0.96 accuracy. Up to now, we could actually say all these models performed quite well, generating above 95% prediction accuracy.

However, the worst performance comes from Naive Bayes which only predicted 76.7% of data correctly in the testing sample. While Naive Bayes Classifier is famous for its good working with large dataset and simplicity, we think the problem here is our large dimension and the relatively small sample size. Without enough samples to compute the distribution of each class, Naive Bayes model would be unable to make precise predictions on each case.

PCA was helping us handle the dimension problem and now comparing the model results after PCA regularization with their performance. First of all, we noticed that the performance of Naive Bayes was improved a lot after dimension reduction, accuracy increasing from 0.767 to 0.8534, which testifies our assumption that a large number of variables was worsening its fitting capability. Nevertheless, the regularization didn't bring other models the expected positive effects. Except for KNN model and Logistic Regression with Lasso penalty which was not hugely affected, the rest models' performance was all impaired, outputting lower accuracy than they used to do with the original dataset. PCA generally did not improve the accuracy for most models. This is to be expected, as PCA does not transform the data with the intention of discriminating between classes.

All things considered, we would like to believe it is because of the nature of the PCA method, which is an unsupervised algorithm that helps reduce dimension. After PCA pre-processing, only part of the information is included in the new dataset while we are not sure if it is the information we need or not. Although we tried to choose the components which could explain as much variance as possible, the nature of unsupervised algorithms makes us unable to link our decision rule with the most important thing in our research, the dependent variable. And it would lead to some important information loss just because it is not taking a large part of the variance. So that's why all those supervised algorithms were impaired. Fortunately, the KNN model, which is also an unsupervised algorithm, survived. As for the Logistic Regression with Lasso penalty, its own variable selection function is playing an important role.

VI. Conclusion and Future Work

In this paper, we have developed seven different machine learning algorithms for predicting human activities. We have implemented Logistic regression with Ridge and Lasso penalization, Linear discriminant analysis (LDA), Support vector Machine (SVM), KNN, Neural Networks and Naive Bayes for our classification problem. The models were trained on a training sample of 8239 human activities after finding the optimal hyperparameters using 5-fold cross validation and tested on a testing sample of 2060 human activities. We have used the testing sample accuracy to compare the performance of the models. The SVM algorithm performed the best in terms of accuracy and achieved an accuracy of 98.50% before regularization and the KNN model performed the best after PCA regularization achieving an accuracy of 95.97%. We have found that PCA regularization does not improve the models except the Naive Bayes algorithm. And the KNN model becomes the best model after regularization because it doesn't rely on learning procedure so it is not affected as much as other supervised models when there is loss in information. Our finding indicates that using PCA results in losing important information which causes sophisticated and supervised algorithms to perform worse. Therefore, our conclusion is that for our dataset that has distinguishable classes, the best model is the SVM algorithm.

For future work, more complicated combinations of FNN could be tried. Since our computers were not powerful enough to perform denser FNN models, we weren't able to try different combinations of activation functions, learning rates and more dense structure for our neural networks algorithm. There are countless more possible combinations

of FNN models that can be tried and variations of neural networks such as Recurrent Neural Networks, Long Short-Term Memory, Convolutional Neural Networks can be applied as well. Another strategy to employ could be Gradient Boosting Machine (Adaboost) or different machine learning algorithms. But overall, we are very happy with the machine learning algorithms that we have tested and with their results.

VII. References

[1] A Beginner's Guide to Neural Networks and Deep Learning. Pathmind. (n.d.). https://wiki.pathmind.com/neural-network.

[2] M.Ranjbaran, P. (2020, December 15). Machine Learning Basics with the K-Nearest Neighbors Algorithm. Data Mining Knowledge. https://r9knowledge.wordpress.com/2020/12/15/machine-learning-basics-with-the-k-nearest-neighbors-algorithm/.

[3] Learning, U. C. I. M. (2019, November 13). Human Activity Recognition with Smartphones. Kaggle. https://www.kaggle.com/uciml/human-activity-recognition-with-smartphones.

[4] UCI Machine Learning Repository: Human Activity Recognition Using Smartphones Data Set. (n.d.). https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones.

[5] Wikimedia Foundation. (2020, December 31). Support vector machine. Wikipedia. https://en.wikipedia.org/wiki/Support_vector_machine.

Appendix:

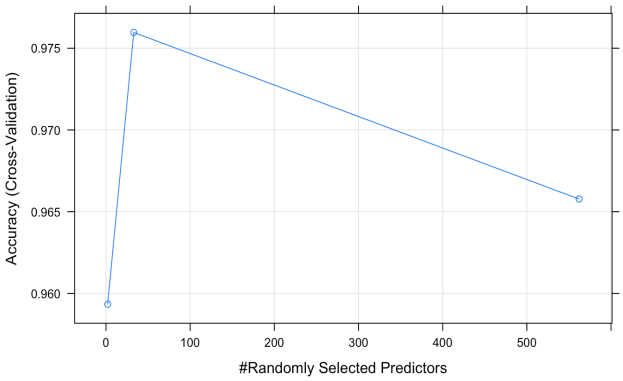
(See our source code for more information)

Random Forest:

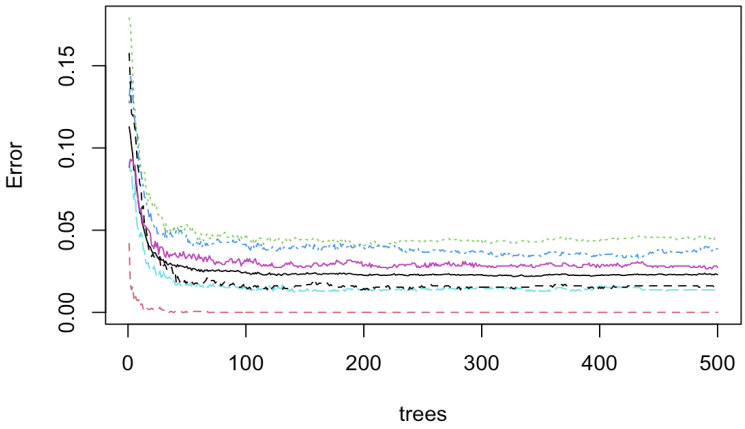
Statistics by Class:

	Class: LAYING	Class: SITTING	Class: STANDING	Class: WALKING	Class: WALKING_DOWNSTAIRS	Class: WALKING_UPSTAIRS
Sensitivity	1.0000	0.9548	0.9753	0.9760	0.9780	0.9902
Specificity	1.0000	0.9941	0.9897	1.0000	0.9944	0.9960
Pos Pred Value	1.0000	0.9729	0.9587	1.0000	0.9639	0.9775
Neg Pred Value	1.0000	0.9899	0.9939	0.9954	0.9966	0.9983
Precision	1.0000	0.9729	0.9587	1.0000	0.9639	0.9775
Recall	1.0000	0.9548	0.9753	0.9760	0.9780	0.9902
F1	1.0000	0.9638	0.9670	0.9879	0.9709	0.9838
Prevalence	0.1772	0.1825	0.1966	0.1621	0.1325	0.1490
Detection Rate	0.1772	0.1743	0.1917	0.1583	0.1296	0.1476
Detection Prevalence	0.1772	0.1791	0.2000	0.1583	0.1345	0.1510
Balanced Accuracy	1.0000	0.9744	0.9825	0.9880	0.9862	0.9931

Cross Validation Results (5 Fold):



Random Forest Cross Validation



SVM:

	Class: LAYING	Class: SITTING	Class: STANDING	Class: WALKING	Class: WALKING_DOWNSTAIRS	Class: WALKING_UPSTAIRS
Sensitivity	1.0000	0.9654	0.9556	1.0000	1.0000	1.000
Specificity	1.0000	0.9893	0.9921	1.0000	1.0000	1.000
Pos Pred Value	1.0000	0.9528	0.9675	1.0000	1.0000	1.000
Neg Pred Value	1.0000	0.9923	0.9892	1.0000	1.0000	1.000
Precision	1.0000	0.9528	0.9675	1.0000	1.0000	1.000
Recall	1.0000	0.9654	0.9556	1.0000	1.0000	1.000
F1	1.0000	0.9590	0.9615	1.0000	1.0000	1.000
Prevalence	0.1772	0.1825	0.1966	0.1621	0.1325	0.149
Detection Rate	0.1772	0.1762	0.1879	0.1621	0.1325	0.149
Detection Prevalence	0.1772	0.1850	0.1942	0.1621	0.1325	0.149
Balanced Accuracy	1.0000	0.9774	0.9739	1.0000	1.0000	1.000

Cross Validation Results (5 Fold):

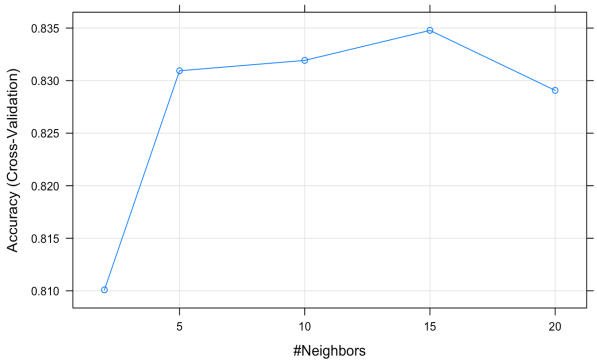
cost	Accuracy	Kappa
0.25	0.9832522	0.9798545
0.50	0.9833736	0.9800006
1.00	0.9814318	0.9776648

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cost = 0.5.

KNN:

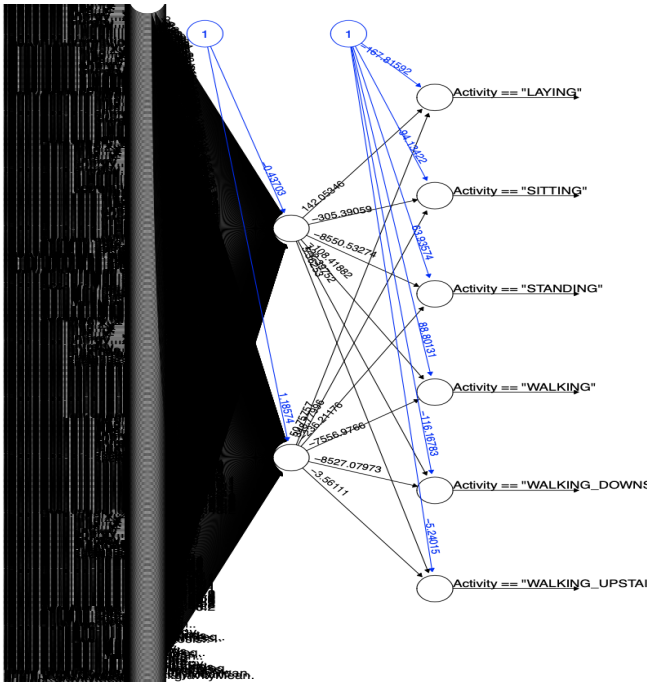
Cross Validation Results (5 Fold):

	Class: LAYING	Class: SITTING	Class: STANDING	Class: WALKING	Class: WALKING_DOWNSTAIRS	Class: WALKING_UPSTAIRS
Sensitivity	0.9973	0.8537	0.9556	1.0000	0.9744	0.9967
Specificity	0.9994	0.9887	0.9674	0.9959	1.0000	0.9994
Pos Pred Value	0.9973	0.9441	0.8776	0.9795	1.0000	0.9967
Neg Pred Value	0.9994	0.9680	0.9889	1.0000	0.9961	0.9994
Precision	0.9973	0.9441	0.8776	0.9795	1.0000	0.9967
Recall	0.9973	0.8537	0.9556	1.0000	0.9744	0.9967
F1	0.9973	0.8966	0.9149	0.9896	0.9870	0.9967
Prevalence	0.1772	0.1825	0.1966	0.1621	0.1325	0.1490
Detection Rate	0.1767	0.1558	0.1879	0.1621	0.1291	0.1485
Detection Prevalence	0.1772	0.1650	0.2141	0.1655	0.1291	0.1490
Balanced Accuracy	0.9983	0.9212	0.9615	0.9980	0.9872	0.9981



k	Accuracy	Kappa
2	0.8100933	0.7713316
5	0.8309390	0.7963753
10	0.8319244	0.7975212
15	0.8347804	0.8009513
20	0.8290731	0.7940577

Neural Networks:




```
          Accuracy : 0.9704
            95% CI : (0.9621, 0.9773)
No Information Rate : 0.1966
P-Value [Acc > NIR] : < 2.2e-16

             Kappa : 0.9644

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0 Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
Sensitivity            1.0000   0.9548   0.9358   0.9940   0.9780   0.9674
Specificity            1.0000   0.9852   0.9897   0.9954   0.9961   0.9977
Pos Pred Value         1.0000   0.9349   0.9571   0.9765   0.9745   0.9867
Neg Pred Value         1.0000   0.9899   0.9844   0.9988   0.9966   0.9943
Prevalence             0.1772   0.1825   0.1966   0.1621   0.1325   0.1490
Detection Rate         0.1772   0.1743   0.1840   0.1612   0.1296   0.1442
Detection Prevalence   0.1772   0.1864   0.1922   0.1650   0.1330   0.1461
Balanced Accuracy      1.0000   0.9700   0.9628   0.9947   0.9871   0.9826
```