UDACITY

MACHINE LEARNING – NANODEGREE

CLASSIFICATION OF HUMAN ACTIVATION VIA SMARTPHONE SENSOR DATA

By: Mostafa Hamed

1. Definition

1.1. Project Overview

It’s no surprise that wearable devices have become so popular in recent years. The ability to track movements and activities throughout the day personalizes the technology and provides individuals with
the tools to take control of and manage their own habits. However, because these devices are designed to be small and unobtrusive, the onboard hardware often consists of rudimentary sensors that only capture three dimensional accelerations and tilt at a set frequency. As a result machine learning is a common technique to identify patterns to draw further intelligence from this data and correctly classify
.resulting activities from these otherwise meaningless signals

With a few exceptions, up to this point most wearables really only target activity as a single cluster mimicking an advanced pedometer to record differences between if the individual is standing, walking or running. However, activities and fitness have progressed with constantly evolving standards and methods related to the way we keep our bodies healthy. To truly track accurate and categorical progression with these devices it’s important to correctly classify a multitude of specific activities as .each one may impact the overall fitness results for each individual differently

1.2. problem statement

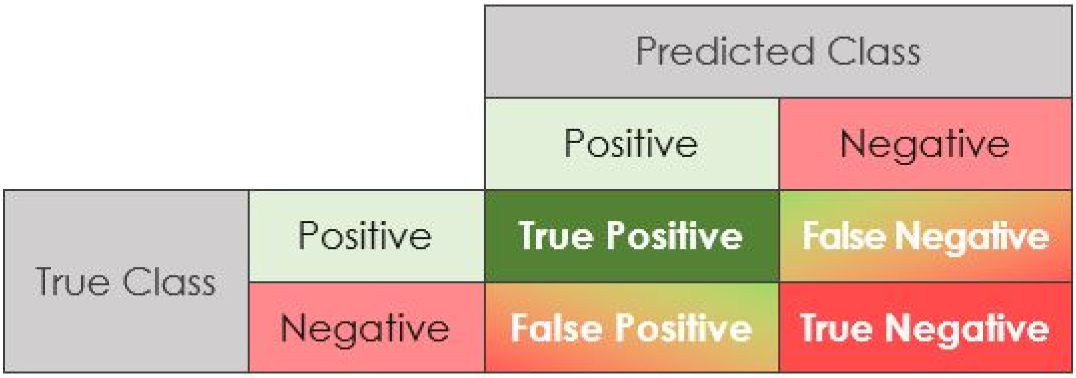
Utilizing machine learning, statistical techniques and a data set of labeled signal attributes this report will explore the applicability of using a classification model to accurately predict and differentiate between a set of unique yet similar activities. If successful this information can be applied to a wide variety of wearable devices to bucket and independently track different activities over the course of a period of time. A solution to this problem can result in tremendous impacts and applications in both health and fitness related tracking for a wide range of individuals.

1.3. Metrics

For multi-label classification problems, like this one, accuracy can be a good starting point and overall metric to determine how well the machine learning algorithm is generalizing the data. Accuracy in this instance is simply the fraction of the correctly classified examples over the total number of examples (Equation 1). The best possible accuracy score is 1.0, indicating the model has correctly classified every example, where the worst score is 0.0 meaning that none of the examples were classified correctly.

accuracy = # of examples correctly classified / Total # of examples (Eq. 1)

Sometimes however, **accuracy is not enough to truly understand the complete behavior of the classification model**. It's also good practice to produce and examine a confusion matrix as this gives further insight into where/how the algorithm is actually making mistakes. In the most basic of uses, it is a graphical indication of the algorithm's precision vs recall for each class within the data set. A binary (yes/no) example of a confusion matrix can be seen in Figure 1.



		Predicted Class	
		Positive	Negative
True Class	Positive	True Positive	False Negative
	Negative	False Positive	True Negative

Figure 1: A standard binary confusion matrix

To apply this to a multi-class classification problem the confusion matrix scales both down and to the right with rows and columns equivalent to the total number of classes. The true positives (accurate predictions) in this case become the diagonal of the matrix, whereas false negatives populate cells in the same row and false positives populate cells in the associated columns. The resulting confusion matrix can then be visualized via a plot type called a heat map (these plots can be seen later in this report).

This type of plot is useful because if a series of "hot" column cells exist outside of the diagonal cell this indicates precision issues for the class belonging to that column. This is because the precision of the model for a particular class is calculated by using Equation 2.

precision = # of true positives / (# of true positives +# of false positives) (Eq. 2)

The precision of a model can best be described as result relevancy, and is a direct indication of a low occurrence of false positives. On the other hand, if a series of "hot" row cells exist outside of the diagonal cell this indicates the associated model has recall issues for the particular class belonging to this row. Recall, which is calculated by using Equation 3, is a measure of if the results returned by the model are truly the predicted class, and is a direct indication of a low occurrence of false negatives.

recall = # of true positives (# of true positives +# of false negatives) (Eq. 3)

These two pieces of information are critical in classification because the cost of making a mistake may be different depending on what type of mistake it actually is. For example, related to this application, if someone is using a device for fitness tracking while doing a set of pushups the cost of a classification mistake may not be as large if the algorithm incorrectly assigns the activity to bicep curls (which still primarily use the arms) as opposed to lunges (which is more related to the legs). In machine learning the model can never be 100% perfect, so it often makes sense to choose the model that makes the “best mistakes” according to the application.

In addition to the metrics mentioned above, training and prediction times were also recorded for the various models. This can be extremely important when deploying the model to a device with hardware limitations or looking to scale it to larger datasets. Depending on the application, if two models perform similarly in terms of accuracy, training and prediction times can become a good differentiator between which one will be the final solution.

2. Analysis

2.1 Data Exploration

The data set for this particular problem was obtained from the UCI Machine Learning Repository. It was collected via gyroscope and accelerometer sensors onboard a waist mounted Smartphone (Samsung Galaxy II) from a group of 30 volunteers between the ages of 19-48. Each individual performed multiple iterations of 3 static postures (sitting, standing, lying) and 3 dynamic activities (walking, walking upstairs, walking downstairs). Additionally, data was also collected on transitional movements between the static postures: sit-to-lie, lie-to-sit, sit-to-stand, stand-to-sit, lie-to-stand and stand-to-lie. Included with this data set are the original raw tri-axial signals from the accelerometer and gyroscope for each participant as well as a preprocessed feature set created through multi-stage filtering and derivation with respect to time.

The preprocessed dataset was already split randomly into training and testing sets with 7761 and 3162 examples, respectively. These values result in approximately 29% of the data reserved for the testing set. Each example is composed of feature vector containing 561 attributes. The 17 main signal variables created from the raw signals, 8 of which have tri-axial values (denoted by –XYZ), can be seen in Table 1.

Variable Name	Variable Description
tBodyAcc-XYZ	Time Domain Body Acceleration
tGravityAcc-XYZ	Time Domain Gravitational Acceleration

tBodyAccJerk-XYZ	Linear Acceleration Derived in Time
tBodyGyro-XYZ	Time Domain Body Angular Velocity
tBodyGyroJerk-XYZ	Angular Velocity Derived in Time
tBodyAccMag	Body Acceleration Magnitude using Euclidean Norm
tGravityAccMag	Gravitational Acceleration Magnitude using Euclidean Norm
tBodyAccJerkMag	Linear Acceleration Magnitude using Euclidean Norm
tBodyGyroMag	Body Angular Velocity Magnitude using Euclidean Norm
tBodyGyroJerkMag	Angular Velocity Magnitude using Euclidean Norm
fBodyAcc-XYZ	Frequency Domain FFT of Body Acceleration
fBodyAccJerk-XYZ	Frequency Domain FFT of Linear Acceleration
fBodyGyro-XYZ	Frequency Domain FFT of Body Angular Velocity
fBodyAccMag	Frequency Domain FFT Euclidean Norm of Body Acc.
fBodyAccJerkMag	Frequency Domain FTT of Euclidean Norm of Linear Acc.
fBodyGyroMag	Frequency Domain FFT of Euclidean Norm of Body Ang. Vel.
fBodyGyroJerkMag	Frequency Domain of FFT of Ang. Velocity Derived in Time

Table 1: The 17 main variables and their descriptions

Each of these main signals where then passed through 17 functions to create (9 single component signal variables * 17 functions + 8 three component signal variables * 3 components * 17 functions = 561) the final 561 attribute feature vector for each example. These .17 functions and their descriptions can be seen in Table 2

Function	Function Description
mean	Mean value
std	Standard deviation
mad	Median absolute deviation
max	Largest value in array
min	Smallest value in array
sma	Signal magnitude area
energy	Energy measure. Sum of the squares divided by N

iqr	Interquartile range
entropy	Signal entropy
arCoeff	Auto regression coefficients with Burg order = 4
correlation	correlation coefficient between two signals
maxInds	Largest magnitude frequency component
meanFreq	Frequency Signal Weighted Average
skewness	Skewness of the frequency domain signal
kurtosis	Kurtosis of the frequency domain signal
bands Energy	Energy of a frequency interval
angle	.Angle between two vectors

Table 2: The functions that each signal was passed through to obtain the 561 Attribute Feature Vectors

The insight gained from this initial preprocessed data exploration is that these feature vectors are fairly large considering the application. It was noted that there is definitely a space to explore dimensionality reduction as a lot of these features might overlap resulting in their .redundancy

It was also decided that in addition to the preprocessed data, the original raw data would be studied as an alternative, less complicated option. If a machine learning algorithm is able to draw intelligence from this data a lot of the preprocessing steps could be removed and would ultimately yield a set of smaller and easier to manage feature vectors. There was some processing involved to clean and sort the data into the appropriate labels, experiment and subject ID. This was because multiple iterations of activities were recorded in a single experiment session. Figure 2 was created to verify a correct understanding of how the signals should be processed. Each experiment consists of continuous tri-axial signals as well as an accompanying series of start and stop values for the various .activities indicated in the plot by colored boundaries

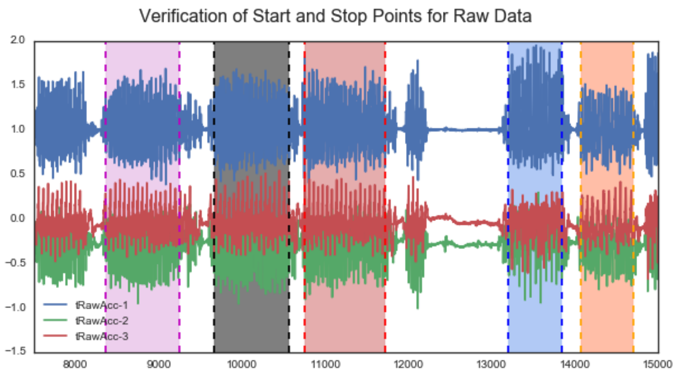


Figure 2: Raw signal data with activity start and stop point verification

Once this was discovered, these signals were split into the associated individual activity slices. From these slices only the minimum, maximum and mean from each sensor and direction was included in the significantly smaller sized (6 signals * 3 functions) 18 attribute final feature vector.

2.1. Exploratory Visualization

One concern when attempting to solve a classification problem (more so with binary classification) is class imbalance. This occurs when there's an overwhelming number of examples for certain classes as opposed to others. For this data set a bar graph was produced to visualize the magnitude of the class imbalance (if it exists) between the various activities, postures and transitions (Figure 3).

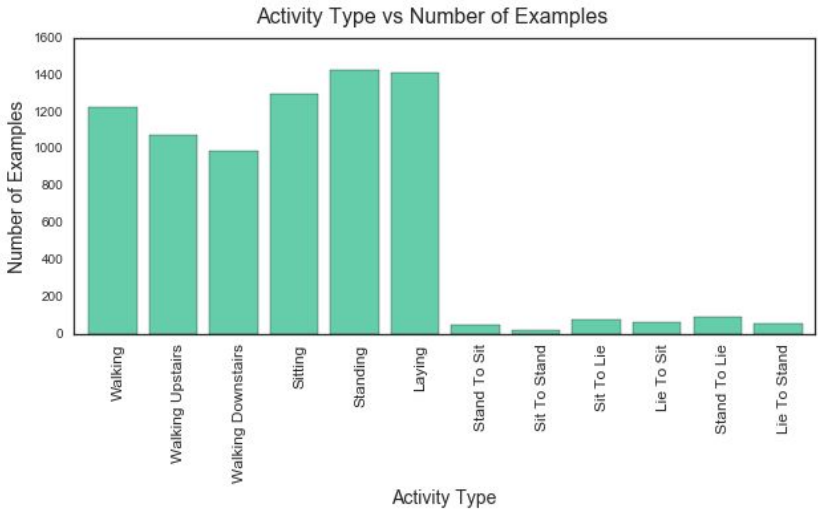


Figure 3: Class imbalance between postures, activities and the associated transitions.

From this graph it's clear that there are significantly less examples for the transition classes. Nothing in particular was done at this point to the dataset to compensate for this class imbalance as it would reduce the size of the dataset too significantly, but it's definitely something to be aware of before proceeding just in case any unexpected results creep up during an analysis phase.

In addition to understanding the classes, several of the features we plotted vs the activity type. This was done to understand how these signal values behave in relation to other values consisting of the same associated classification label (Figure 4). From these plots, it's clear that there are certain distinguishable patterns in these newly formed combination signals for the specific activity categories: dynamic, static and transitional. However, there will be difficulties differentiating between the actual individual activities in those categories. A number of the features were plotted using this technique yielding similar signal profiles. This is indicative that there's likely overlap between multiple features and this data set is a good candidate for dimensionality reduction.

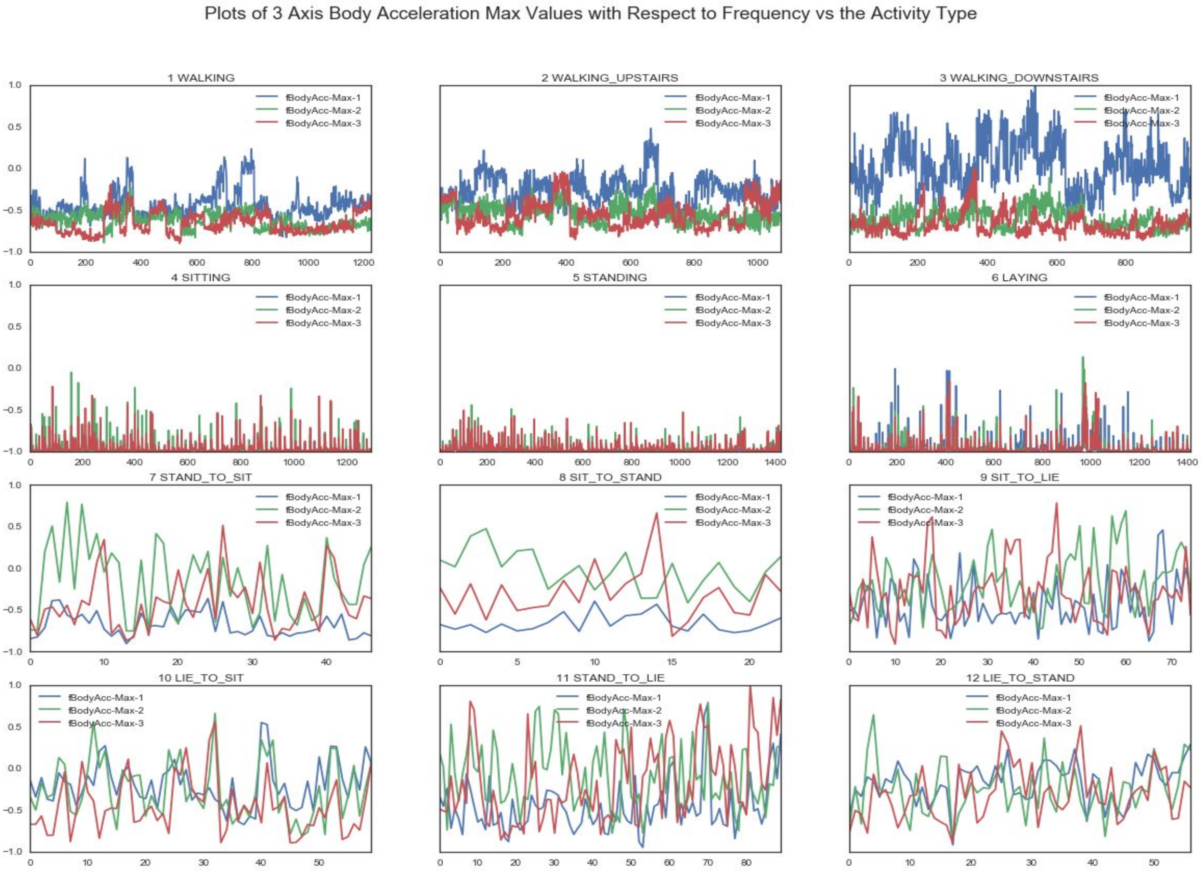


Figure 4: Plots for each activity type and 3 axis body acceleration maximum values in the frequency space.

2.2 Algorithms and Techniques

For dimensionality reduction, principle component analysis (PCA) was used. PCA is an algorithm that generates a new set of features (principle components) that attempt to capture the directions in the data set that contain the most variance. By identifying these “high-variance” directions, conclusions can be made about which features contain the most “information” and which features can be combined into new composite features. Each principle component is created from a linear combination of the original features, and because the resulting vectors are orthogonal to one another they contain no overlapping information. This idea can be best understood when it’s interactions with a given data set are visualized. A simple, yet effective, 2D example of the directions a PCA would uncover can be seen in Figure 5.

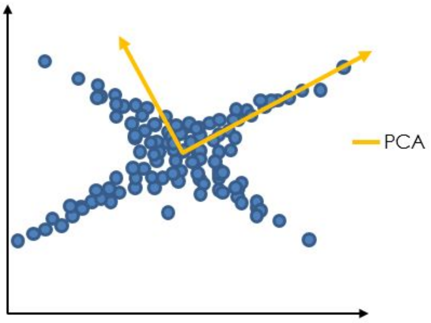


Figure 5: PCA visualization on a 2D example data set.

For the machine learning algorithms, four different types were chosen in the early stages of exploration to see how they responded primarily to the PCA reductions. These four algorithms are Linear Support Vector Classifier, Gaussian Naïve Bayes Classifier, Support Vector Classifier with an ‘rbf’ kernel, and finally a Random Forest Classifier.

Support Vector Classifiers are a set of machine learning algorithms that attempt to draw linear decision boundaries that maximize the margin between the set of classes. This maximization of the margin can best be understood by examining Figure 6. In this figure, there are multiple examples of two different classes either represented by a Blue Circle or a Red Ex. While these classes are clearly separable by a linear line, it’s important to choose the “best” line as there can be an infinite number drawn to simply achieve separation. In this figure, two possible decision boundaries are depicted, both of which successfully complete the task of separating the two classes. However, the one on the right can be deemed as superior due to the maximization of the margins between both the individual classes and the line. Imagine if the white square was actually a Blue Circle needing prediction by one of these two decision boundaries. If margin was not maximized between the two classes, this data point would be misclassified as a depicted in the figure on the left.

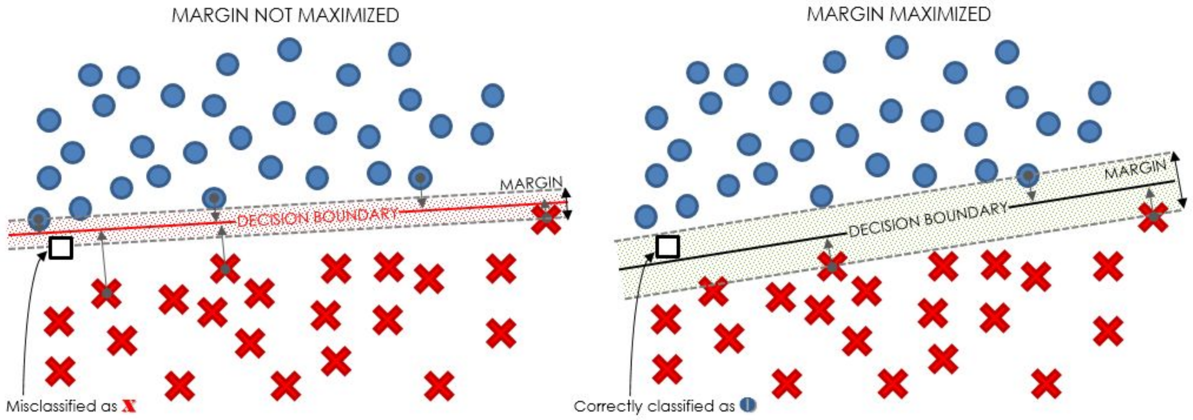


Figure 6: Decision boundary maximization visualized

This method of margin maximization can be extremely powerful in a data set with clear decision boundaries, but it begs the question: what if the data is not so easily linearly separable? This is where Support Vector Classifiers have an ace up their sleeve. Using what is often referred to as the “kernel- Page 8 of 21trick” Support Vector Classifiers can map the original dataset to new spaces of higher dimensionality. A visual representation of this can be seen in Figure 7. In this example, non-linearly separable data in a 2D space is mapped 3 Dimensions. With this mapping the line is now elevated to a plane which can easily (“linearly”) separate the different classes. This “kernel-trick” is the difference between the Linear SVC and SVC with the ‘rbf’ kernel used on this data set. It’s worth mentioning, SVCs can map data to higher dimensions (even infinite), but it becomes increasingly difficult to understand and maybe even harder to visualize. SVCs were chosen due to their successful results in practice because of these principles

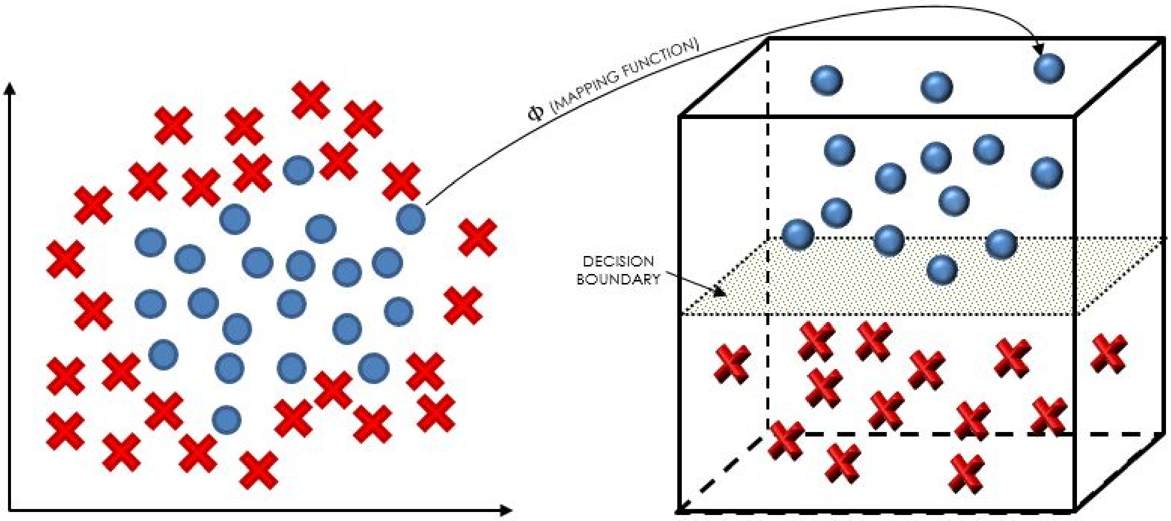


Figure 7: Support Vector Classifier 'kernel-trick' visualized

Naïve Bayes Classifiers are a family of classifiers built on the principles and foundation of the Bayes Theorem. The “naïve” reference stems from the fact they assume the independence between each pair of features. For example, if a Naïve Bayes Classifier considers a vegetable to be a carrot if it is a root, orange and about 15cm long, upon classification, each of these features would contribute independently to the probability that it is classified as a carrot without any regard for the correlations that might exists between the growth environment, color and size of this vegetable. This algorithm was primarily chosen because of its simplicity and the fact that this algorithm scales well because of it.

Finally, a Random Forest Classifier was used which is an ensemble method of decision trees. Decision trees are an algorithm that attempts to build a set of decision paths (branches) that split the data at multiple levels eventually creating buckets (leaves) that minimize classification error. Based on the path a data point traverses down the tree the associated leaf at the end of that path is the classification that is determined (Figure 7).

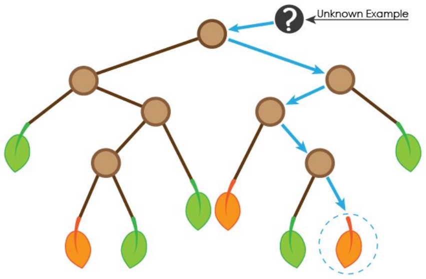


Figure 8: A visual representation of a simple decision tree

However, left to their own devices and the inherent nature in which they operate decision trees can be highly prone of overfitting at larger depths. Random Forests attempt to alleviate this problem bycreating a series (forest) of less complicated decision trees that ultimately predict the final class by themode, mean or an alternate voting method from the individual trees (Figure 8).

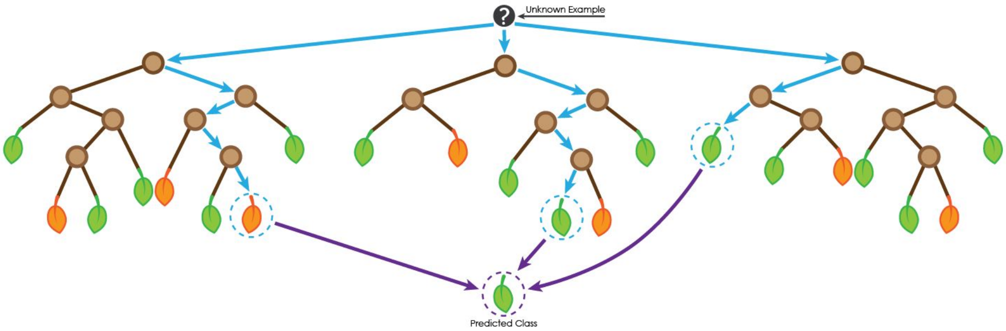


Figure 9: Visual representation of a Random Forest algorithm

2.3 Benchmark

As previously mentioned, the algorithms and solutions will be benchmarked against the following metrics: accuracy, training and prediction times, as well as a visual of the confusion matrix to verify if the algorithm is in fact making mistakes they are not serious (precision vs recall). The seriousness of the mistakes will be determined by if the algorithm classifies a mistake as a similar activity or a non-related activity. For example, even if it is only a few mistakes, it would be undesirable to settle on an algorithm that incorrectly classifies “Standing” as “Walking Upstairs.” A mistake of this nature could seriously compromise the output data if it were being used for fitness or health tracking. It would appear as if the individual had been exercising from time to time throughout the day when maybe they were just standing still for most of it. On the other hand, if an alternate algorithm only made mistakes by classifying various examples of “Standing” as “Sitting,” this might be the better choice even if accuracy is not as high as the others tested.

3 Methodology

3.1 Data Preprocessing

A majority of the preprocessing for the 561 feature attribute dataset had already been conducted and was supplied pre-shuffled and split for training and testing sets. However, as previously mentioned from the data exploration phase, it was determined that there was an opportunity for dimensionality reduction. Utilizing PCA with the same number of original features, the explained variance ratio values were plotted for the resulting composite features (Figure 10).

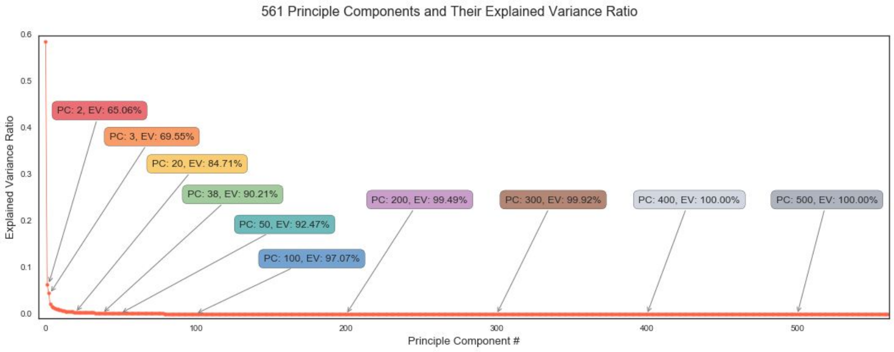


Figure 10: Full Feature Vector PCA and Their Explained Variance Ratio

In this plot, each data point represents the fraction of the total variance that is captured by that particular principle component. With further examination, it becomes obvious that the explained variance contribution stabilizes as the component number increases. This stabilization confirms previous assumptions about feature overlap as the explained variance has negligible change above a PC equal to approximately 200.

Additionally, a series of PCs have been labeled with the variance that would be captured if a PCA was used to reduce the feature space to this dimension. This explained variance (EV) percentage is calculated by summing the associated value for each lesser PC up to and including the PC in question. Some notable mentions are that 2, 3 and 38 component PCAs capture 65.06%, 69.55%, and 90.21% of the variance for this dataset, respectively. It can be argued that the 2 and 3 component PCAs do not yield a high enough percentage of data variance explanation however, because these dimension are in a space that can be easily understood and visualized it's worth looking at how data points behave in these reduced feature spaces. The first visualization for a 2 component PCA reduction can be seen in Figure 11.

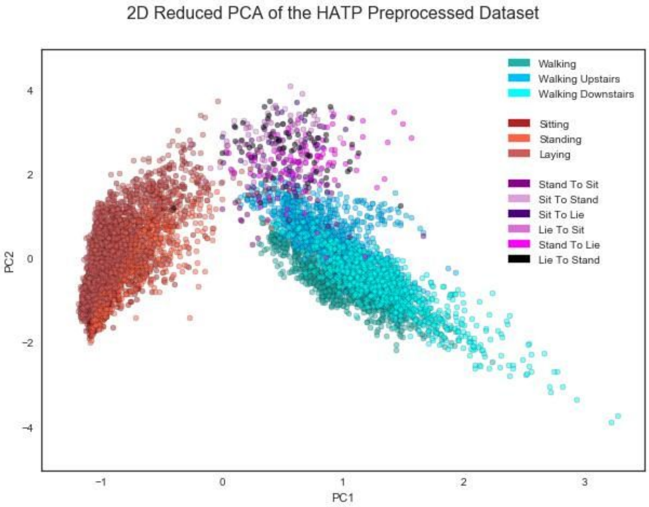


Figure 11: The 2 Components PCA Dimensionality Reduction Visualization

As previously expected from the data exploration phase and verified by this PCA plot, there is indeed some clustering happening with the specific activity categories (dynamic, static, transitional). Even with a massive feature reduction from 561 to 2 by using PCA, those definite signal differences seen earlier are still apparent. If the only interest was in classifying the activity category, this 2 component PCA might be satisfactory. However, given the amount of overlap internally inside each activity-type cluster any machine learning algorithm will likely have trouble picking out the key individual activities.

It is a somewhat similar story with a 3 component PCA however, there is definitely more class separation with the added dimension (Figure 12).

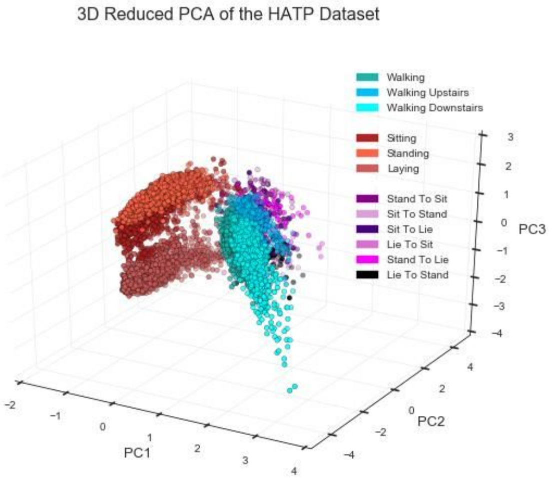


Figure 12: The 3 Components PCA Dimensionality Reduction Visualization

3.1 Implementation

For baselining the algorithms the data was reduced using a 3 component PCA. While this reduction only captures about 70% of the variance in the data it highlights a few previous assumptions as well as reveals some interesting insights on how the different algorithms behave on this particular data set. The individual testing and training accuracies for each tested algorithm can be seen in Figure 13.

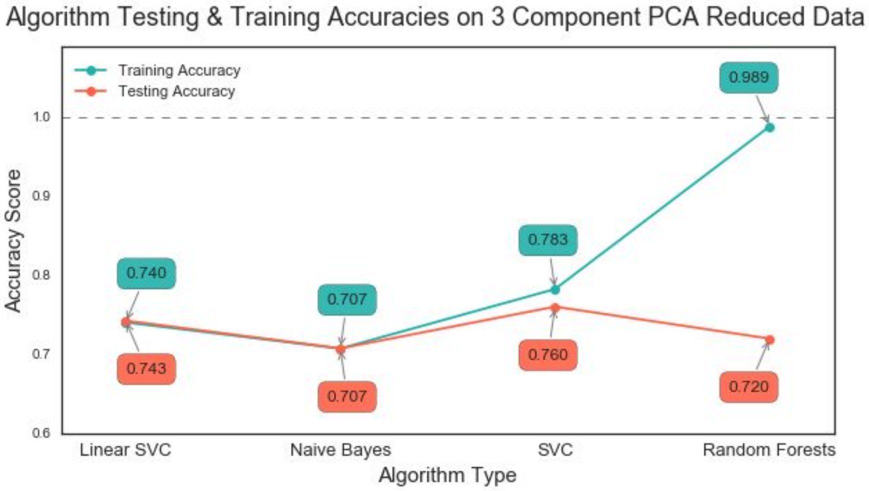


Figure 13: Testing and training accuracies for a 3 component PCA reduced data set

The default settings for each algorithm were used for this comparison. The most obvious conclusion from this plot is the overfitting that’s happening with the Random Forest Classifier.

However, most of the testing accuracies are all within the same general range. This is where the associated confusion matrices, seen in Figure 14, allow for a deeper understanding of how each algorithm is behaving.

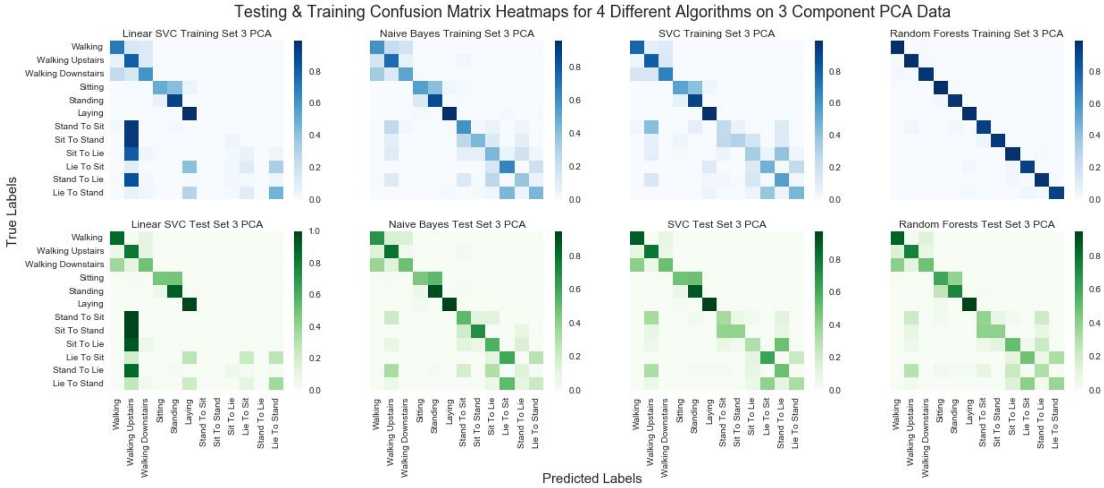


Figure 14: Confusion matrix visual comparison between the 4 classifier types

The training confusion matrices are denoted in “Blue” where the testing confusion matrices are displayed in “Green.” Annotations for the number of mistakes have been removed and the “heat- magnitudes” have been normalized per row to account for the class imbalance in the dataset. The predicted labels are along x-axis while the true labels are along the y-axis. A perfectly accurate algorithm depicted as a normalized confusion matrix would be the visual representation of the identity matrix (solid color along the top-left to bottom-right diagonal). The closest example to this can be seen in the training CM from the Random Forest Classifier. This visually reiterates the fact that it is overfit to the training data as the test set CM directly below it differs quite drastically. One additional key takeaway from this plot is that while the Linear SVC had the second highest testing accuracy it’s actually making more detrimental mistakes. Because of the way the activities are structured it’s desired the algorithms avoid incorrect classification in the bottom left and top right corners. This indicates the mistakes are happening on the more dissimilar activity types. For example, the LinearSVC algorithm is classifying almost every single transitional activity as “Walking Upstairs,” highlighted by the ‘hot’ cells outside of the diagonal cell in its column. As previously mentioned this is an indication of poor model precision for this class. The other algorithms also make these mistakes, but these areas in their associated plots are a lot less saturated indicating a fewer number of them.

Finally, before choosing how best to proceed, the training and prediction times were also plotted to see which model performed this best in this regard (Figure 15). As expected due to its simplicity, the Naïve Bayes Classifier was the best overall performer using this metric.

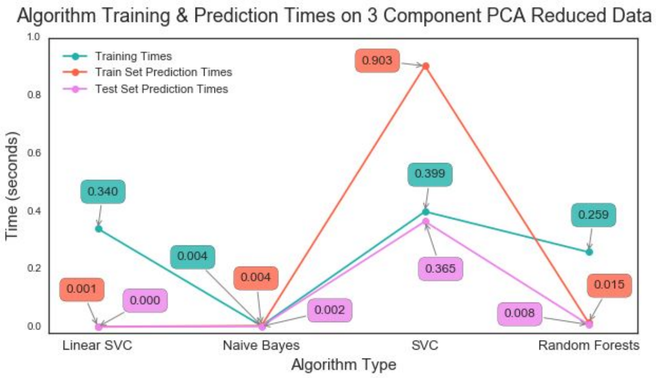


Figure 15: Training & prediction times (train set & test set) for the 4 different classifiers

After compiling the conclusions from Figures 12, 13 & 14, it was decided to proceed with both the most accurate algorithm as well as the fastest algorithm as this would be a true battle between speed and precision. The two algorithm types that were selected were Naïve Bayes (for its speed) and SVC with ‘rbf’ kernel (for its accuracy).

3.2 Refinement

The first step in the refinement process was to reevaluate the amount of PCA feature space reduction. Since the 3 component PCA only captured approximately 70% of the data variance, it was determined that a 38 component PCA that captures 90% of the variance, while still limiting the feature vector size, would be chosen for the next round of testing for the two selected algorithm types. Both the testing and training accuracies for Naïve Bayes and SVC on the 38 component PCA transformed data set can be seen in Figure 16. It’s important to note that while the SVC still scored higher against this metric, both algorithms increased in performance by approximately 20% over their original value which is roughly equivalent to the same increment in which the data variance capture increased.

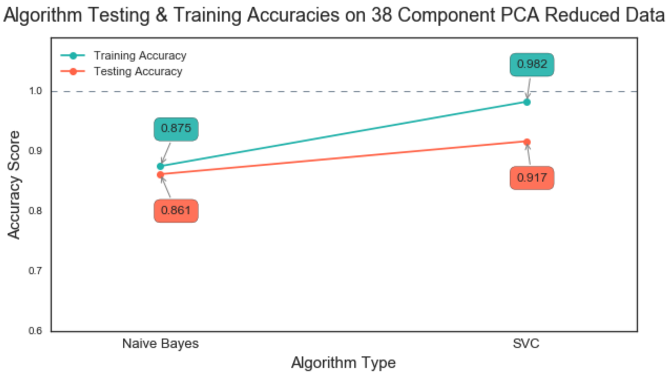


Figure 16: Testing and training accuracies for a 38 component PCA reduced data

To test this correlation between the explained variance captured by the number of principle components and the accuracy of the individual models, iterations of both train and test accuracies were recorded for an incremental list of a select number of principle components. The resulting plot for Naïve Bayes and SVC can be seen in Figure 17.

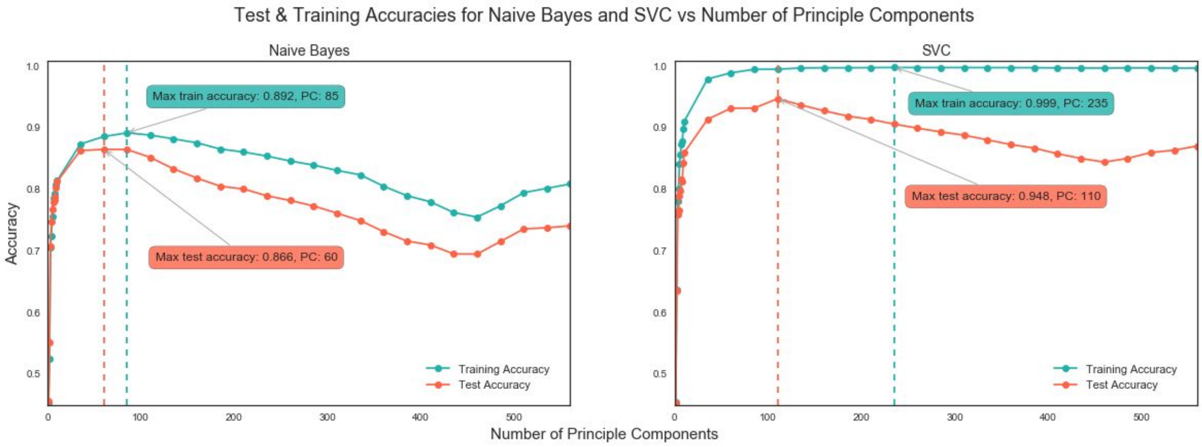


Figure 17: Naive Bayes and SVC test and training accuracies vs number of principle components

While both testing curves follow more or less the same profile, just at different magnitudes, the training curve for Naïve Bayes drops off with this profile. This is due to the previously mentioned feature independence that Naïve Bayes assumes. With the number of features in this dataset generated from only 6 distinct signals there are likely similarities and correlations between them, and Naïve Bayes is not equipped to identify them. On the other had SVC experiences some overfitting in the higher order of principle components indicated by a stable training curve and a declining testing curve. From these plots it was decided that accuracy would be the clear winner and the SVC would be tuned for the best performance and the final model. This was implemented by transforming the data using a 110 component PCA (the highest testing data point recorded) for a feature space reduction of approximately 80%. Then a Cross Validation Grid Search was conducted over several C and gamma values for the SVC algorithm. This iterative algorithm cycles through a set of dictionary parameters in attempts to find the optimal parameter combination that returns the best cross-validation estimator score from the supplied metric (accuracy score in this case).

The ***'C' parameter*** controls the tradeoff between misclassified examples and the complexity of the decision surfaces. The supplied values to the Grid Search were: 1, 10, 100 and 1000. A lower number attempts to make the decision surfaces smooth, where a high C value always attempt to classify every example correctly resulting in complex and often overfit decision surfaces.

The ***'gamma' parameter*** can best be described as the inverse magnitude of the radius of influence that

a single training example invokes on the created the decision surfaces. A large gamma value means that the influence of a single data point is low, where if gamma is small each training example's influence is high. The supplied values for gamma to the Grid Search were: 0.001, 0.01, 0.1, 1 and 10.

4 Results

4.1 Model Evaluation and Validation

The final model for the preprocessed, 110 component PCA data settled on a C value equal to 10 and a gamma value equal to 0.001. The final training and testing accuracies are 98.8% and 93.9%, respectively. In addition, the resulting training and testing set confusion matrices can be visualized by the heat map plots in Figure 18.

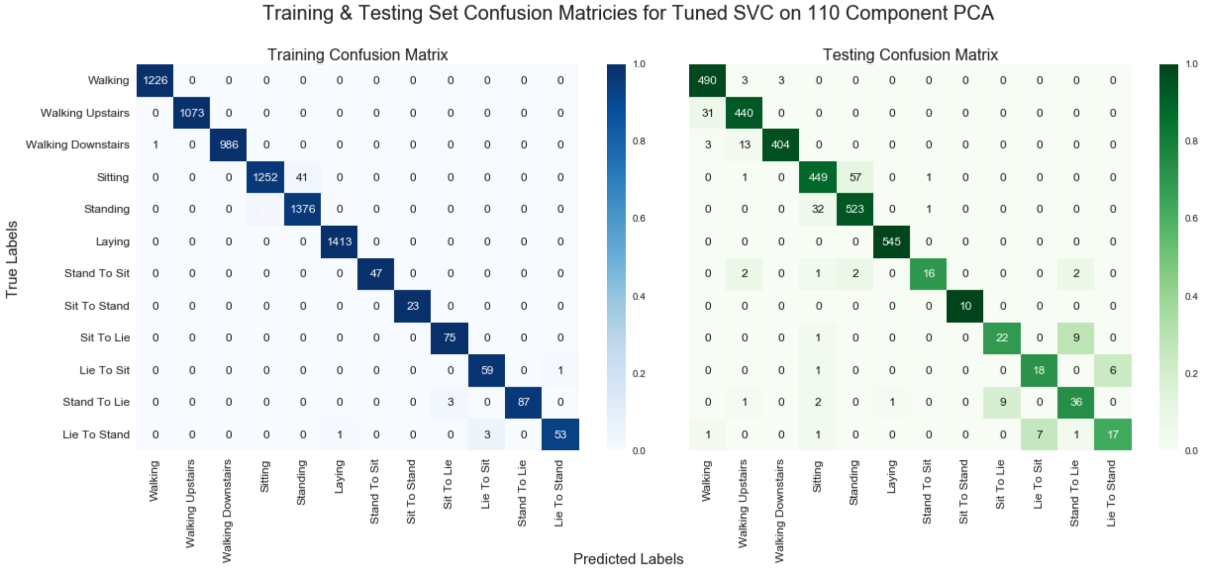
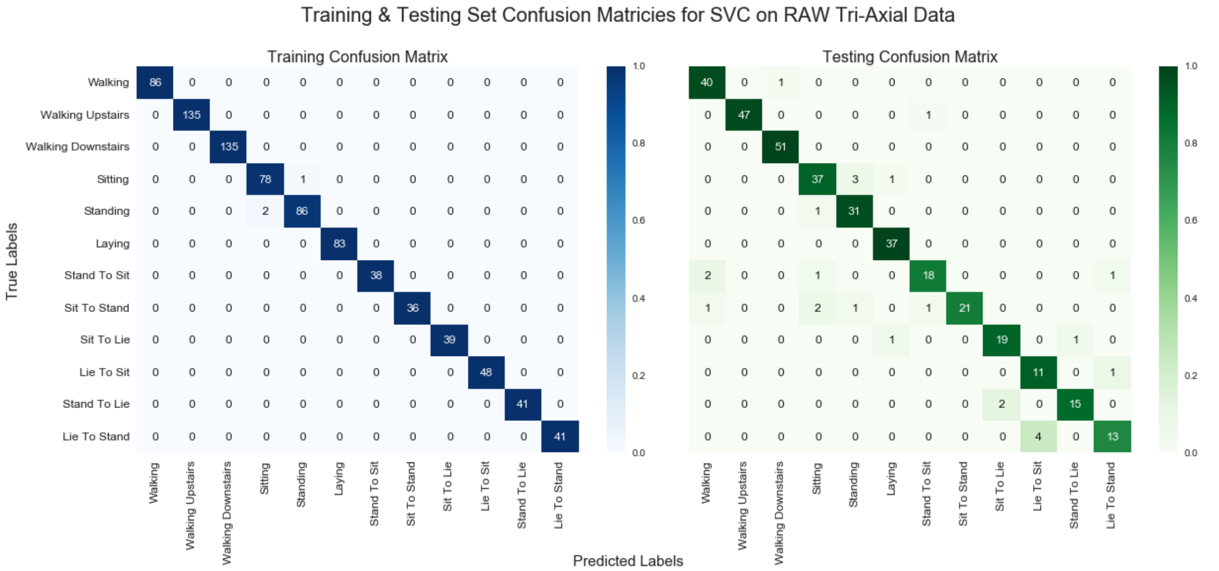


Figure 18: Tuned SVC train and test confusion matrices for a 110 component PCA reduction

From this plot, it's apparent the final model fits the data very well on both training and testing sets. Also, a lot of the initial misclassification problems have been alleviated. Where there are mistakes, these misclassifications are similar activities in the same larger activity categories minimizing their impact.

Finally, the same Grid Search algorithm was used to find the best C and gamma values for an SVC on the raw tri-axial data. For this data the best C parameter was equal to 100 while the gamma value was 0.1. The tuned accuracies for the testing and training sets are 99.6% and 93.1%, respectively. The resulting confusion matrices can be seen in Figure 19.



Training Confusion Matrix

	Walking	Walking Upstairs	Walking Downstairs	Sitting	Standing	Laying	Stand To Sit	Sit To Stand	Sit To Lie	Lie To Sit	Stand To Lie	Lie To Stand
Walking	86	0	0	0	0	0	0	0	0	0	0	0
Walking Upstairs	0	135	0	0	0	0	0	0	0	0	0	0
Walking Downstairs	0	0	135	0	0	0	0	0	0	0	0	0
Sitting	0	0	0	78	1	0	0	0	0	0	0	0
Standing	0	0	0	2	86	0	0	0	0	0	0	0
Laying	0	0	0	0	0	83	0	0	0	0	0	0
Stand To Sit	0	0	0	0	0	0	38	0	0	0	0	0
Sit To Stand	0	0	0	0	0	0	0	36	0	0	0	0
Sit To Lie	0	0	0	0	0	0	0	0	39	0	0	0
Lie To Sit	0	0	0	0	0	0	0	0	0	48	0	0
Stand To Lie	0	0	0	0	0	0	0	0	0	0	41	0
Lie To Stand	0	0	0	0	0	0	0	0	0	0	0	41

Testing Confusion Matrix

	Walking	Walking Upstairs	Walking Downstairs	Sitting	Standing	Laying	Stand To Sit	Sit To Stand	Sit To Lie	Lie To Sit	Stand To Lie	Lie To Stand
Walking	40	0	1	0	0	0	0	0	0	0	0	0
Walking Upstairs	0	47	0	0	0	0	1	0	0	0	0	0
Walking Downstairs	0	0	51	0	0	0	0	0	0	0	0	0
Sitting	0	0	0	37	3	1	0	0	0	0	0	0
Standing	0	0	0	1	31	0	0	0	0	0	0	0
Laying	0	0	0	0	0	37	0	0	0	0	0	0
Stand To Sit	2	0	0	1	0	0	18	0	0	0	0	1
Sit To Stand	1	0	0	2	1	0	1	21	0	0	0	0
Sit To Lie	0	0	0	0	0	1	0	0	19	0	1	0
Lie To Sit	0	0	0	0	0	0	0	0	0	11	0	1
Stand To Lie	0	0	0	0	0	0	0	0	2	0	15	0
Lie To Stand	0	0	0	0	0	0	0	0	0	4	0	13

Figure 19: Training and testing confusion matrix heatmap plots for raw tri-axial data

What's interesting in this case is that the extremely simplified tri-axial data model performs on par with the more complex model fitted to the 110 component PCA reduction of the preprocessed data. Granted there are far less examples in this dataset when compared to the preprocessed version, but the confusion matrices reveal that both are generalizing the data in the same way and making very few mistakes in the same areas. The most common mistakes both these model make are between the transitional classes Lie-to-Stand and Lie-to-Sit, as well as the static classes sitting and standing.

4.2 Justification

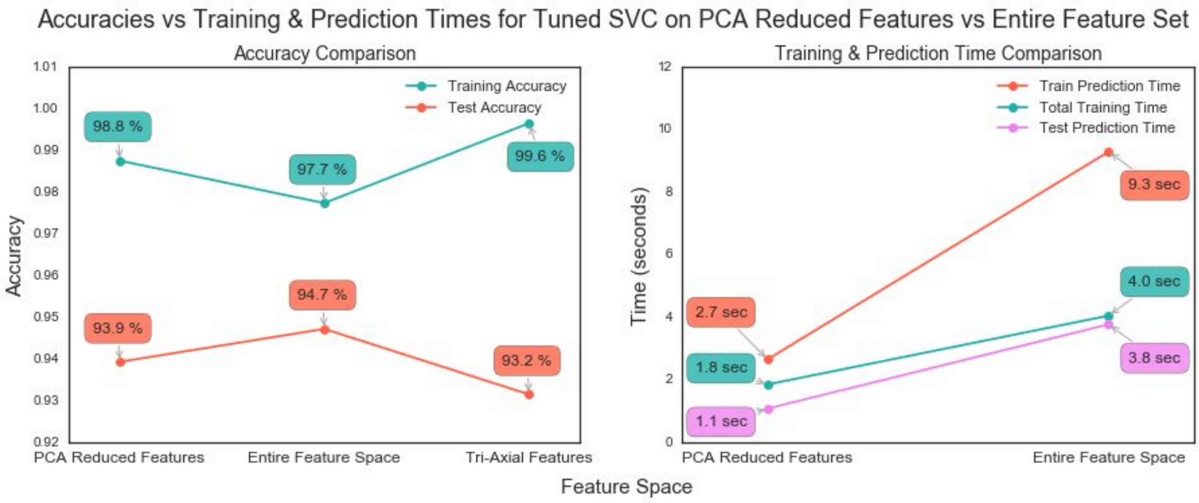
Both models trained and tuned on the preprocessed and tri-axial raw data are viable solutions to this classification problem. From the baselining on the 3 and 38 component PCA reduction, the accuracy score increased by 23.6% and 3.3%, respectively. While it could be argued that the extra 3.3% increase in accuracy is not worth extending feature vectors out to 110 components that accuracy increase mostly comes from the fact that the model does a better job of handling the class imbalance. For some of the models, especially on the reduced data, there are just not enough examples of the transitional activities contributing to the variance directions to properly classify them against the other activities. Therefore, for this particular use case on the preprocessed data set the more complex 110 feature PCA reduced model is the better choice.

What was highly unexpected however, was for the raw signal data to perform so well. With a feature vector size of only 18 this ends up being an 83.6% reduction over the feature vectors used on the preprocessed data reduced with PCA, whereas the accuracy only suffers a loss of performance of 0.85%. It's subjective to say this is a 'better' model however, due to the differences in the number of training examples between the two data sets.

5 Conclusions

5.1 Free-Form Visualization

For the final free-from visualization, it was decided to train an SVC on the entire feature space with the same tuning parameters as the model that was used to fit the 110 component PCA reduced data. This was done primarily to study the differences in terms of accuracy and training and prediction times for each model. The associated plots for comparing these two models can be seen in Figure 20 . Additionally, the accuracy data point from the model created using the raw tri-axial feature set has been included in the plot on the left. However, the corresponding data for this associated model has been removed from the time comparison plot to avoid confusion since the number of examples was significantly less.



The plot on the left shows some slight overfitting on the PCA reduced and tri-axial models, whereas the entire features model performs better in this regard. However, examining the plot on the right hand side, it becomes apparent that while the total training time and test prediction timelines increase at roughly the same slope, the training prediction timeline rises at a significantly higher rate. This indicates, with a larger feature space, classification of a higher number of examples is not a linear relationship. Therefore there are performance benefits with the reduced feature space models with minimal impact to the actual accuracy

5.2 Reflection

Overall, the final model performs very well for classification of these specific activities from this data set. The initial hypothesis about the applicability for PCA reductions held throughout the course of implementation. The most interesting result however was the fact that the feature set produced from the raw tri-axial data performed so comparably.

For machine learning in general, this project was really eye opening to the artistic aspects of tuning parameters, selecting the models and appropriately balancing tradeoffs. The method of attack really is application dependent and up to the engineer to decide how best to proceed and what tradeoffs are most important to balance. Ultimately for this problem accuracy was a main driver, but examining the confusion matrices really gives more insight into the model that should be selected.

5.3 Improvement

Related directly to the final models, a more narrowly focused grid search could be done to possibly squeeze a few more percentage points out of the final testing accuracies. Recall that for the grid search the values of the C and gamma were scaled by a factor of 10 across the range. After the initial tuned parameters were revealed a secondary grid search could be performed to narrow this range in the newly discovered window and find the true optimal values.

Additionally, in order to really develop a solution for the greater applications of this particular problem there would definitely need to be more activity classes and data gathered to support the signals for these classes. It would also be interesting to have the associated age and biometric information (height, weight, BMI, % body fat, etc.) for each experiment to see if there are differences between them or errors in predictions depending on how these groups perform the individual activities. Another additional piece of information that would be useful is more data captured with the onboard sensors from different cell phone manufacturing to see if these same assumptions hold across devices.