Feature Selection for Classification in the Human Activity Recognition Project

Ricky Su

September 25, 2017

Contents

1 Introduction

Since 2011, nearly 80% of the world population has had access to a mobile phone [4]. The market is getting closer to reaching virtually everybody, especially with the new generation of smartphones. These relatively new devices, in addition to basic mobile phone capabilities, contain a variety of sensors that can be used to track daily activities. Smartphones have already made their way into our lives, and can potentially assist us in making smarter decisions in our future actions [1].

This gave way to many research projects in the area of potential smartphone applications in assisted living technologies. This was the motivation for the Human Activity Recognition project, a study of 30 volunteer participants in the age bracket of 19-48 performing daily activities: standing, walking, laying, sitting, walking upstairs and walking downstairs. During these activities, a Samsung Galaxy S2 smartphone was attached to the waist to measure inertial body signals, permitting continuous monitoring of numerous physiological signals [1].

From these signals, measured by the raw accelerometer and gyroscope signals in the smartphone, 561 features were recorded describing the movements of the participants. Using these features, researchers have tried to predict what activity each participant was performing, with predictive modeling. The basic idea here is to somehow create a differentiation, or boundary, between each activity within the features. Then, the predictive model can be used in real-time analysis for assisted living technologies. However, this isn't as simple as just putting all 561 features into a model, and checking the result. This method would be computationally heavy, and wouldn't scale in real time. This project aims to propose a number of options that can be used to select a small amount of features, and is still accurate to a threshold.

The goal of this project is to find predictive models that use as few features as possible, but still achieve a consistent accuracy of either 80%, or 90%. Accuracy, in this case, is the percentage of how many times the models can guess the true activity each participant is performing, based on the features, without knowing the activity beforehand. For example, if a participant is walking, we would like the model to predict correctly that the participant is walking. The intuitive method would be to use all of the features for classification. This has proven to be very accurate (>95%), but the cost of generating that number of features in real time, and fitting them in predictive models, uses significant memory and processing power.

2 Data Collection and Preparation

2.1 Data Provenance

Using the embedded accelerometer and gyroscope in the Samsung Galaxy S2, 3-axial linear acceleration and 3-axial angular velocity were captured at a constant rate of 50Hz [2]. These

signals were then preprocessed for noise reduction with a median filter and a 3rd order low-pass Butterworth filter with a 20 Hz cutoff frequency [2]. However, the gravitational and body motion components were contained in the acceleration signal, so another Butterworth low-pass filter was used to separate body-acceleration, and gravity [2]. The activities were labeled manually from video record.

The time signals were then sampled in fixed-width sliding windows of 2.56 sec and 50% overlap (128 data points/window) [2]. This time interval was chosen because Anguita et al. preferred a full walking cycle (two steps) in each window sample, and the average cadence of a person walking is within [90,130] steps/minute, leading to a minimum of 1.5 steps/sec [3]. However, elderly or disabled persons were expected a speed of 50% of an average human.

Each record then consists of the 3-axial acceleration from the accelerometer (total acceleration), the estimated body acceleration, and the 3-axial angular velocity from the gyroscope, resulting in a 561-feature vector with time and frequency domain variables [2]. The participant's ID, along with the activity, were also captured in the data. The entire dataset contains 10,299 records. It is available through the University of California Irvine (UCI) Machine Learning Repository [5].

2.2 Salient Aspects

The features of this dataset came primarily from the accelerometer and gyroscope raw signals ("tAcc-XYZ" and "tGyro-XYZ"). Each feature signal begins with a 't' for time domain, or 'f' for frequency domain. The body linear acceleration and angular velocity were derived in time to obtain Jerk signals, the magnitude which was calculated using the Euclidean norm. Frequency domain signals were obtained from time signals using a Fast Fourier Transform (FFT). A majority of the features result in the X, Y, and Z directions.

Statistical methods were applied to these signals, generating more features through the use of the mean, standard deviation, median absolute deviation, etc. For variables related to angles, additional vectors were obtained by averaging those signals within a window. A complete list of the 561 features can be found in the UCI Machine Learning Repository [5].

2.3 Splitting the Dataset

To assess the performance of our analysis, the dataset was divided into training and testing datasets, sized 70% and 30% respectively. This would be 5,146 training records, and 2,206 test records. We would like to create the best model possible using the training set, and check the performance on the test set. However, using feedback from the test set would invalidate our model, as it would remove the generality of how our model would perform with new data. In this case, we then also split the training set to 70% and 30% also. The new training set is now 5,146 records, and the validation set is 2,206 records. The former set is then used solely for training purposes only, while the latter is used as a validation set.

The validation set allows us to test our models, without using the actual testing dataset.

2.4 Missing and Unusual Data

Reading through the data using Python, it was found that no data was missing. A quick outlier analysis was also applied to check for unusual data. Splitting the data by activity, and considering a criteria of 3 standard deviations of each respective feature, it was found that the number of outliers of each feature, if any, never exceeded 10% of the number of records for each respective activity. We can assume that the entire dataset is valid, and we can use all records for our analysis.

2.5 Feature Engineering

The features in the data describe the activity that each participant is performing. We apply them to the predictive models we create, and they strongly influence the results obtained. As mentioned previously, our goal is to train a classification model that will still achieve over 80%, or 90% accuracy, using the smallest number of features possible. Using too many of the 561 features would be computationally heavy, and would not scale to real-time analysis.

Common methods that are used for feature selection include filter-based, wrapper, and embedded methods. Filter-based methods use some statistical significance, such as variance, to eliminate features that do not meet a certain threshold. Although these methods are intuitive, they do not guarantee strong results. Wrapper methods involve selecting features that best improve the predictive model. Popular wrapper methods are forward selection, backward elimination, and recursive feature elimination. In forward selection, we start with a single feature, and then the best feature is added to the model through each iteration, until negligible improvement, or a threshold is met. In backwards elimination, the model begins with all of the features, and removes the single least significant feature until negligible improvement, or a threshold is met. Recursive feature elimination is a specific type of backwards elimination, which removes a certain number or percentage of the features through each iteration, not just one. A specific type of recursive elimination involves evaluating specific models that output rankings of features, such as a random forest. Then a similar iteration occurs. Filter methods, backwards elimination, and recursive feature elimination were explored in this analysis to attempt to reduce the dimensionality of the dataset, while maintaining high accuracy.

3 Exploratory Analysis

One of the statistical metrics that is used for the filter-based approach is variance. The average variance for each feature is 0.0992. Based on this metric, we can filter features that are either above or below it. Requiring features to be above it would mean that they

have a high variance, possibly leading to differences within the features when separated by activity.

Each feature is normalized between [-1,1]. This would have an effect on the performance of these models, as it would be much quicker to operate on smaller numbers than larger ones. Ultimately, the data has been heavily cleaned and processed before acquiring it for this project.

4 Methods

4.1 Machine Learning Models Used and Model Selection Process

We would like to use a range of different classification models to assess how well we can predict using a limited number of features. In this project, we explored 4 different models, covering simple to complex calculations. These were logistic regression, decision tree, random forest, and support vector machine (SVM). At the basic level, logistic regression calculates the probability of a data point belonging to a class based on an odds-ratio. Support vector machines are algorithms used to find a hyperplane or boundary to separate the data as well as possible by their classes, or in this case, by activity. Decision trees use the most important features to iteratively divide the data into predicted classes. A random forest is a group of decision trees, each with randomized parameters, with the single outcome being the mode of the decision tree outcomes. All models and analyses were implemented using Python's scikit-learn package [6].

4.2 Measures of Uncertainty

Obviously, there can be more analysis done with this. Other models can be explored, and more refining of the parameters of each model can be done. One model that was omitted is the popular Neural Network. Since the models used in this project already achieve high accuracies, using a Neural Network seemed unnecessary, also since my experience with the model is minimal.

As previously mentioned, the training data was split only once, into a new training set and a validation set. However, to push this further, we could have used cross validation. This would be a better way to split the training set, as there would be multiple splits to test the model with, possibly resulting in a better model.

4.3 Feature Selection Process in Models

4.3.1 Filter-based

In the filter-based approaches, various statistical metrics were used to remove features. Correlation within data usually leads to redundant features. A popular filter-based method,

that is implemented here, is to remove highly correlated features. It is intuitive that features moving in the same direction are not beneficial to building separation between classes.

Variance also plays a large role in filtering. Two methods were implemented using this statistic. The motivation for the first method came intuitively from the information that the variance gives. The mean of each feature is calculated, and grouped by activity. Then taking the variance of each feature's means, we can choose the top N most varied columns. In a basic sense, the features that vary the most between each activity should give large separation between them.

The second method involves removing features that have low variance. The average variance for each feature is 0.0992, so in this case, we only consider features with a variance of 0.25 or greater. This narrows the number of features down to 34. A similar method involves just choosing the top N features with the largest variance. This would just extend the current method just subsetting the 34 features we have.

4.3.2 Backwards Elimination

A popular and effective way to select features is the idea of initially using all of the features, and removing the least significant one through iteration, while evaluating a model. This method iterates until a threshold is met. This method is usually done with a model that places weights on each feature, creating a rank-like structure, so that the least significant feature is apparent. Applying this process on a Logistic Regression showed to achieve promising results.

4.3.3 Recursive Feature Elimination

A special type of backwards elimination is recursive elimination. Features can be recursively selected by removing a number or percentage of them through each iteration. We can then select how many features we want to end up with. This method was applied to each model, as results vary by predictions. As mentioned before, a specific recursive elimination method involves using certain models that produce a feature ranking after it is trained. A common example is using a random forest, which is implemented in this analysis. At the lower level, within each decision tree, the first decision is usually the strongest, and will separate the maximum amount of data. This decision is based on a feature, and through a number of decision trees, we can create a ranking of the importance of each feature. Using this, we can iteratively eliminate the least effective feature until a threshold is met.

4.4 Reproducibility

The code is reproducible through Python. Random states were set in the script, and models with the same accuracies can be produced from scratch.

5 Analysis and Results

5.1 Benchmark

As a baseline to compare our results to, we implement the random forest algorithm using all 561 features. This achieves an accuracy of ≈93%. Computationally, it took close to 18 seconds to train this model, along with predicting. It is clear that this methodology would not scale to real-time analysis. We now use the previously mentioned feature engineering methods to choose the most relevant features.

The results came from partitioning the training set to a training and validation set. Models were finalized on the validation set, without feedback from the test set.

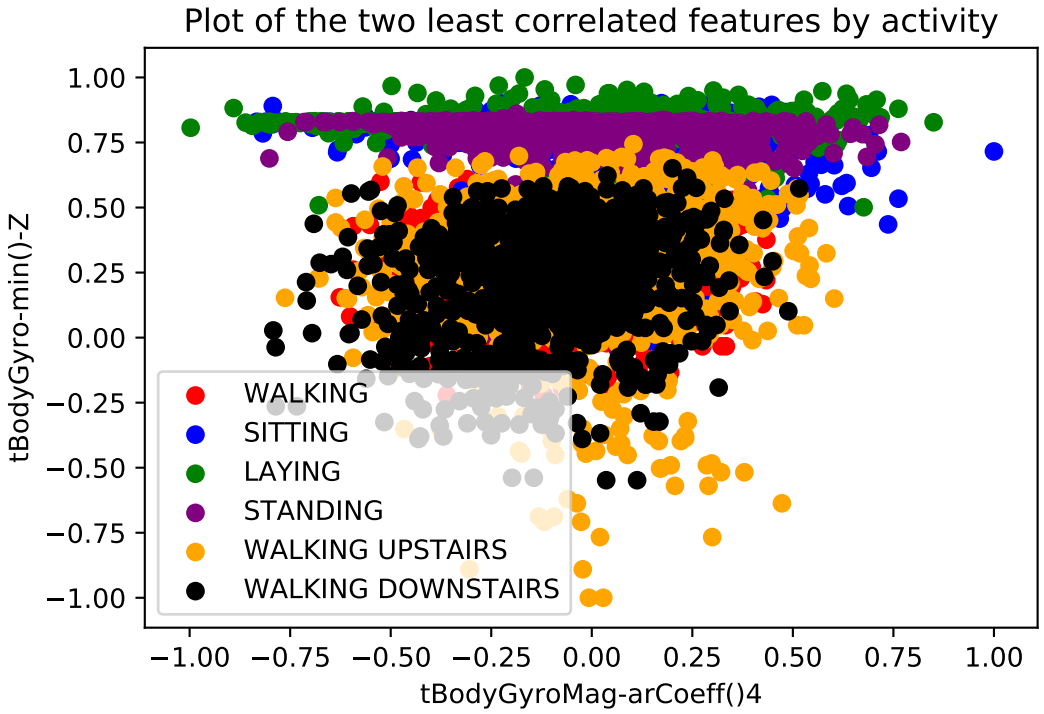
5.2 Methodology

Features were chosen until the models achieved an adequate accuracy on the validation set. It is common that the accuracy will decrease when used to evaluate the test set. The methodology then follows that for the 80% and 90% thresholds. Models were refined on the validation until the threshold was met. The model is then evaluated on the test set.

As mentioned before, the models used for this analysis were Logistic Regression, Support Vector Machine, Decision Tree, and Random Forest. The next section is organized into each feature selection method, along with analysis between the 80% and 90% threshold, for ease of comparison. All analysis was done on the validation set first, then on the test set. Top accuracies are bolded.

5.3 Removing Highly Correlated Features

Plotting the two least correlated features, we can see some differences between each activity. This infers that there is possibly a way to create separation between activities. The correlation is -2.4584e-06, which is essentially no correlation at all. This is between tBodyGyroMag-arCoeff()4 and tBodyGyro-min()-Z.



5.3.1 80% Accuracy

Using this filter method, features that have a correlation of over 0.55 or less than -0.55 were removed. This left us with **43 features**. Since this accuracy would most likely decrease on the test set, we do not need to increase our correlation constraint.

Model	Validation Acc.	Test Acc.
SVM	88.62%	76.21%
Decision Tree	76.56%	65.39%
Random Forest	**89.35%**	**81.81%**

5.3.2 90% Accuracy

We had to relieve the correlation criteria a bit to reach the 90% accuracy threshold. Setting new thresholds of correlations amongst features to be within [-0.6, 0.7], **58 features** were used. The validation set reached a top accuracy of 91.57%, and the test set a high of 85.48%, both by a random forest.

Model	Validation Acc.	Test Acc.
SVM	91.48%	81.51%
Decision Tree	81.05%	70.61%
Random Forest	**91.57%**	**85.48%**

5.4 Variance by Activity

5.4.1 80% Accuracy

Another method is filtering by variance. Grouping features by activity, we can choose the top N features with the largest variance. It makes sense that features that change the most between activities will give intuitive separation. Using the top **25 features**, we reach the following results:

Model	Validation Acc.	Test Acc.
SVM	**91.34%**	**87.21%**
Decision Tree	87.72%	79.13%
Random Forest	91.25%	86.16%

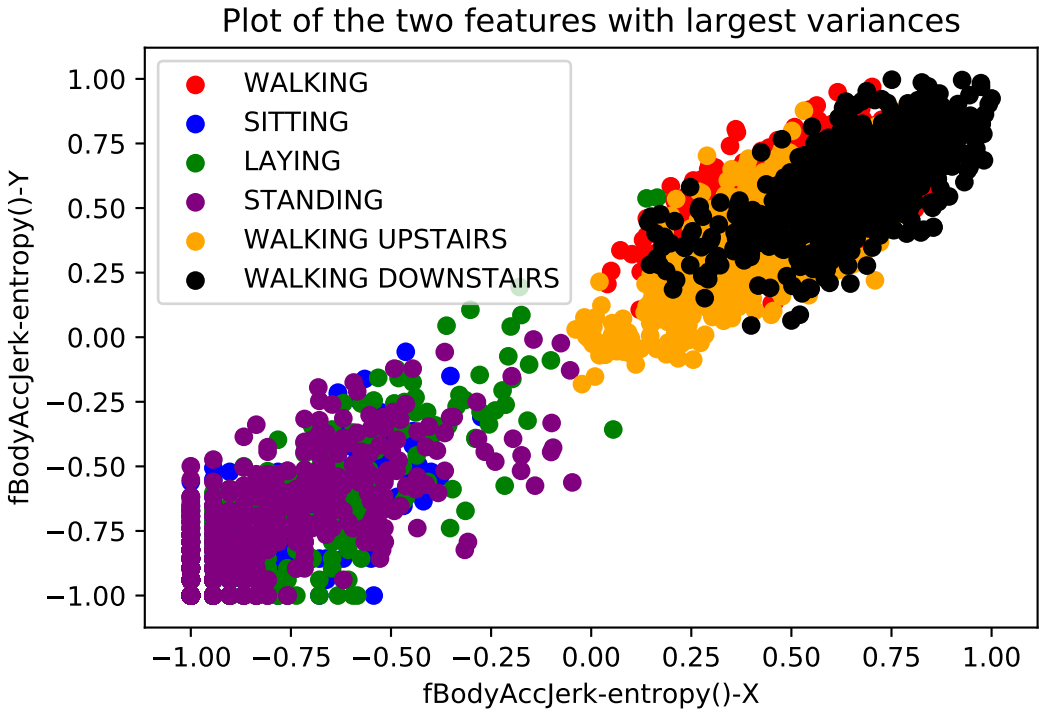
5.4.2 90% Accuracy

With 25 features, the accuracy on the validation set is not high enough for the test set. Increasing to **50 features** instead, we reach the threshold.

Model	Validation Acc.	Test Acc.
SVM	**93.56%**	**91.96%**
Decision Tree	89.66%	80.76%
Random Forest	92.70%	86.66%

5.5 High Variance

Plotting the two features with the highest variance, we see sort of an even better separation between activities. These were fBodyAccJerk-entropy()-X with a variance of 0.56502 and fBodyAccJerk-entropy()-Y at 0.54243.



5.5.1 80% Accuracy

Without grouping by activity, the average variance for each feature is ≈0.0992, with the max as 0.56502 and min as 0.001665. We select all features who have a higher variance than 0.4, resulting in **17 features**. Random forest performs the best here.

Model	Validation Acc.	Test Acc.
SVM	87.72%	83.03%
Decision Tree	81.60%	75.06%
Random Forest	**87.85%**	**83.44%**

5.5.2 90% Accuracy

If we lower the variance threshold, to select more features, we can reach the 90% threshold. With a new threshold of 0.25, we end up with **34 features**, reaching high accuracies with a SVM.

Model	Validation Acc.	Test Acc.
SVM	**94.02%**	**91.18%**
Decision Tree	88.44%	79.30%
Random Forest	93.16%	87.00%

5.6 Backwards Elimination

5.6.1 80% Accuracy

Using this expensive method, we remove the least significant feature per iteration. This wrapper method was used with a logistic regression, and the features were then evaluated on each of the 4 previously described models. **7 features** were chosen, and each validation accuracy was over 90%, with the highest being 94.47%, coming from a random forest. Since we are using such few features, the model may do even worse than normal on the test set. Even though the validation accuracy was high, we would like to make sure the test set accuracy will still be over 80%. However, we still reached a high accuracy of 86.90% using logistic regression or a SVM.

Model	Validation Acc.	Test Acc.
Logistic Regression	89.98%	**86.90%**
SVM	91.21%	**86.90%**
Decision Tree	92.75%	80.35%
Random Forest	**94.47%**	84.49%

The features are as follows:

Feature
tGravityAcc-mean()-X
tGravityAcc-mean()-Y
tGravityAcc-energy()-X
tBodyAccJerk-entropy()-X
tBodyAccJerkMag-iqr()
fBodyAcc-std()-X
fBodyAccMag-mad()

5.6.2 90% Accuracy

Increasing to **10 features**, even though the validation accuracy was high, the threshold for the test set did not meet the threshold. Perhaps the top 15 would work better.

Model	Validation Acc.	Test Acc.
Logistic Regression	92.25%	87.38%
SVM	93.70%	**88.80%**
Decision Tree	92.93%	81.71%
Random Forest	**95.42%**	84.32%

The features are as follows:

Feature
tGravityAcc-mean()-X
tGravityAcc-mean()-Y
tGravityAcc-max()-Y
tGravityAcc-energy()-X
tBodyAccJerk-entropy()-X
tBodyGyroJerk-entropy()-X
tBodyAccJerkMag-iqr()
fBodyAcc-std()-X
fBodyGyro-bandsEnergy()-25,32
fBodyAccMag-mad()

5.7 Recursive Feature Elimination

5.7.1 80% Accuracy

Using RFE, we can still choose the number of features we'd like to end up with, while removing a certain number, or percentage, of features per iteration. Applying this method to each model, we retrieve specific feature selections. In this case, we chose only **5 features**,

removing 10% of the features through each iteration. The RFE method used here will have selected features based on each respective model. Surprisingly, a decision tree reached the highest accuracy on the validation set, while a random forest attains the highest on the test set.

Model	Validation Acc.	Test Acc.
Logistic Regression	79.42%	79.84%
SVM	86.13%	79.40%
Decision Tree	**93.34%**	80.35%
Random Forest	87.99%	**80.72%**

The features are as follows:

Logistic Regression	SVM	Decision Tree	Random Forest
tGravityAcc-energy()-X	tBodyAcc-std()-X	tGravityAcc-mean()-Y	tBodyAcc-max()-X
tGravityAcc-arCoeff()-X,1	tGravityAcc-mean()-Y	tGravityAcc-min()-X'	tGravityAcc-min()-X
tBodyAccJerk-entropy()-X	tGravityAcc-arCoeff()-X,1	tGravityAcc-arCoeff()-Z,1	tBodyAccMag-std()
tBodyAccJerkMag-iqr()	tBodyAccJerk-entropy()-X	fBodyAccJerk-bandsEnergy()-1,8	fBodyAccMag-mad()
fBodyAccMag-mad()	fBodyAccJerk-mean()-Z	fBodyAccMag-mad()	angle(X,gravityMean)

5.7.2 90% Accuracy

Aiming for 90%, we increase to **10 features**. However, even though the validation accuracies seem promising, the test set accuracies slightly missed the threshold.

Model	Validation Acc.	Test Acc.
Logistic Regression	90.25%	85.27%
SVM	94.24%	**88.87%**
Decision Tree	93.61%	83.20%
Random Forest	**95.24%**	81.95%

The features are as follows:

Logistic Regression	SVM	Decision Tree	Random Forest
tGravityAcc-mean()-Y	tBodyAcc-std()-X	tBodyAcc-correlation()-X,Y	tBodyAcc-max()-X
tGravityAcc-max()-Y	tBodyAcc-correlation()-X,Y	tGravityAcc-mean()-Y	tGravityAcc-mean()-X
tGravityAcc-min()-Y	tGravityAcc-mean()-Y	tGravityAcc-max()-X	tGravityAcc-mean()-Y
tGravityAcc-energy()-X	tGravityAcc-max()-Y	tGravityAcc-min()-X	tGravityAcc-max()-Y
tGravityAcc-energy()-Y	tGravityAcc-energy()-X	tGravityAcc-energy()-Y	tGravityAcc-min()-X
tGravityAcc-arCoeff()-X,1	tGravityAcc-arCoeff()-X,1	tGravityAcc-arCoeff()-X,1	tGravityAcc-energy()-X
tBodyAccJerk-entropy()-X	tBodyAccJerk-entropy()-X	tGravityAcc-arCoeff()-Z,1	tGravityAcc-arCoeff()-Z,2
tBodyAccJerkMag-iqr()	tBodyAccJerk-entropy()-Z	tBodyGyro-correlation()-Y,Z	tBodyAccMag-std()
fBodyGyro-bandsEnergy()-33,40.1	tBodyGyroJerk-entropy()-X	fBodyAccJerk-bandsEnergy()-1,8	fBodyAccMag-mad()
fBodyAccMag-mad()	fBodyAccJerk-mean()-Z	fBodyAccMag-mad()	angle(X,gravityMean)

5.8 Recursive Feature Elimination - Feature Ranking

5.8.1 80% Accuracy

From the random forest algorithm with all 561 features, we can select features based on their respective ranking that the model used. Selecting the top 20 most important features, a random forest achieves a high accuracy on the validation set. However, since we are only using 20 features, it is most likely that the accuracy will decrease heavily on the test set. Hence, the threshold was not met here. It's interesting that the random forest algorithm using the **20 features** achieves a worse accuracy, since the ranking came from a random forest.

Model	Validation Acc.	Test Acc.
SVM	88.53%	**75.64%**
Decision Tree	91.57%	70.95%
Random Forest	**93.29%**	72.99%

5.8.2 90% Accuracy

Here, we require more features. Evaluating the top **30 features**, we reach a top validation accuracy of 97.28% with a random forest. Increasing that number to **40 features** does not boast a much larger benefit, as the top validation accuracy only increases to 97.82%, also by a random forest. The results are as follows, with a SVM at the top of both options for the test set.

	30 Features		40 Features	
Model	Validation Acc.	Test Acc.	Validation Acc.	Test Acc.
SVM	92.97%	**85.61%**	94.70%	**89.85%**
Decision Tree	95.15%	77.64%	95.19%	83.68%
Random Forest	**97.28%**	79.67%	**97.82%**	85.44%

6 Conclusion

The goal of this project was to explore feature selection through a real world example. Data sets often have too many features to choose for analysis, and from a classification analysis point of view, that can lead to many problems, such as overfitting. Overfitting is when a model is trained too harshly on a data set, and so it cannot work well generally on new data. This usually happens when using too many training points, or features. We aimed to find a predictive model that uses as few features as possible to achieve an accuracy of at least 80%, and also at least 90%.

The data comes from the Human Activity Recognition Project. There are 561 features, measuring the movements of a person performing six different daily activities: walking, standing, sitting, laying, walking upstairs, and walking downstairs [2]. The goal of the Human Activity Recognition Project was to implement a real time analysis for assisted living technologies [2].

Through the analysis, it seems that the best methods were the most exhaustive ones. These were backwards elimination, and recursive feature elimination. This is probably because it takes account of the features and models iteratively, adjusting based on a moving metric. Comparing this to the filter based methods, where the statistical metric chosen as the threshold does not constantly change as number of features change, it does much better. The filter methods implemented in our analysis, using correlation, and variance require the most features to reach the accuracy thresholds. The better of these two metrics is discriminating by variance.

Obviously, there are some potential problems to this analysis. First off, all possible choices for feature selection were not exhausted in this analysis. There can be more future work done, which may definitely achieve better results, such as creating a combination of the most selected features of this analysis. Perhaps there are certain features that work best together. The parameters for each model were not heavily adjusted. If more time permitted, a grid search of each model would definitely help the analysis, checking multiple ranges of parameters for the best accuracy possible. Also, we do not know how well the process of scaling this to a real-time analysis would be. Since the training set has become much smaller, we could have also used cross-validation to refine our models, using the 70% to 30% split. Another potential problem may be that the data was already normalized to -1 and 1. Perhaps we could have reached better results without normalizing the data.

Comparing results to the random forest baseline, there is definitely no need to use all 561 features. In some cases, we even begin to see overfitting, as some models which use less features achieve even better results. Overall, it looks like using around 6-8 features will result in a consistent model with accuracies of over 80%, and around 15 features for 90%.

There can be much more future work done on this project, with much more analysis. Ultimately, we found that backwards and recursive feature elimination worked best,

achieving high accuracy with small numbers of features. As mentioned before, it would be interesting to combine features from different feature selection methods. Overall, it seems support vector machines and a random forests do the best in this analysis. A final table recording the best results follows, with a comparison to the original baseline.

LR	Logistic Regression
SVM	Support Vector Machine
RF	Random Forest

	80% Threshold			90% Threshold		
Method	No. Features	Model	Test Accuracy	No. Features	Model	Test Accuracy
Baseline Random Forest	561	RF	92.67%	561	RF	92.67%
Remove High Correlation	43	RF	81.81%	58	RF	85.48%
Variance by Activity	25	SVM	87.21%	50	SVM	91.96%
High Variance	17	RF	83.44%	34	SVM	91.18%
Backwards Elimination	7	SVM/LR	86.90%	10	SVM	88.80%
Recursive Elimination	5	RF	80.72%	10	SVM	88.87%
Recursive Ranking Elimination	20	SVM	75.64%	40	SVM	89.85%

References

[1] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge L Reyes-Ortiz. Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In *International workshop on ambient assisted living*, pages 216–223. Springer, 2012.

[2] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2013.

[3] Chiraz BenAbdelkader, Ross Cutler, and Larry Davis. Stride and cadence as a biometric in automatic person identification and verification. In *Automatic Face and Gesture Recognition, 2002. Proceedings. Fifth IEEE International Conference on*, pages 372–377. IEEE, 2002.

[4] Jessica Ekholm and Sylvain Fabre. Mobile data traffic and revenue, worldwide, 2010-2015. *Gartner Mobile Communications Worldwide*, 2011.

[5] M. Lichman. UCI machine learning repository, 2013.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.