Machine Learning Capstone Project

# Human Activity Recognition with Smartphones

**Kadir Firat Uyanik**

February 12th, 2017

# Definition

Human Activity Recognition (HAR) is a research that aims to develop systems to realize automatic recognition of physical activities to extract information about the user-behavior so that these systems can proactively improve user’s experience and interaction with the computer. This is usually done by utilizing external sensors (e.g. environmental cameras), sensors on the user (e.g. wearables, body-worn sensors), or sensors embedded in the objects that we interact with (e.g. smartphones). HAR systems successfully took place in products like Fitbit, Nintendo Wii for entertainment and Nike+ running shoes for fitness.

One challenge with the activity recognition that doesn’t necessarily exist in object recognition or speech recognition is that HAR offers more degrees of freedom in terms of system design and implementation. Due to its temporal nature, it is not very clear what starts/ends and when. In other words, there is no common clear definition, grammar or structure of human activities that we can use to make a clear and generic problem statement (**Bulling et al., 2014**).

## Project Overview

The focus of this project is to recognize activities of daily living based on the motion related data acquired through a waist-mounted smartphone with embedded motion-sensitive sensors.

Dataset is consisted of the motion related data sampled from the activities of 30 people performing *walking, walking-upstairs, walking-downstairs, sitting, standing*, and *laying* and the activity labels given by the experimenter based on the recorded video of the performed activities.

Source of the sampled data is the embedded sensors streaming data regarding 3-axial linear acceleration and 3-axial angular velocity at a constant rate of 50Hz.

## Problem Statement

In this project, the main goal is to classify the data instances --represented in the 561 dimensional feature space- into the categories of daily life physical activities labeled as *walking, walking-upstairs, walking-downstairs, sitting, standing*, and *laying*. To achieve this goal:

1. **Outlier detection and removal** are tried to see if the classification can be improved,
2. Various **feature transformation and selection** methods are applied to find an alternative representation to increase intra-class similarity and inter-class distinction,
3. Various **supervised machine learning models** are trained. Their classification performances are compared. I chose one model to carry out the data processing mentioned in (i) and (ii).
4. Trained Support Vector Machine (SVM) model in (iii) is then tuned and compared against second best classification method, Stochastic Gradient Descent, based on their classification performance under various dataset size and corrupt data conditions.

Main hypothesis is that the processes like outlier removal, feature transformation, feature selection and model-parameter tuning help improving the classification performance. Contribution of each process will be tested by training an SVM model with the default parameters that comes with the sklearn library. Main goal is to try out different data processing and training methods to obtain the highest classification performance.

## Metrics

I use k-fold cross validation to measure the contribution of the processes mentioned above and to select the main classification method (viz. SVM) among other potential classification methods.

I extract *precision, recall,* and *f-score* metrics for every single folding, and take the mean of the values obtained as a result of k-folds. I consider the weighted average of the metrics based on the number of class labels. This way, obtained metrics are more robust against the class imbalance.

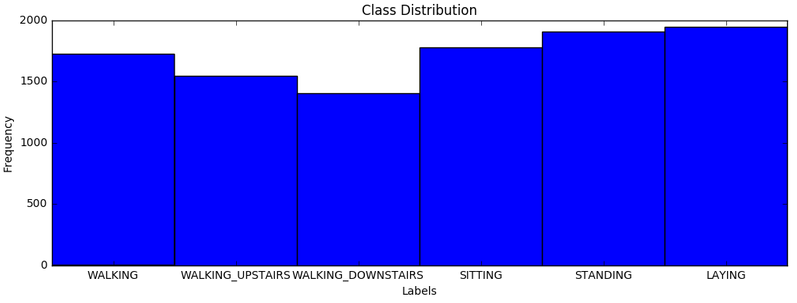
# Analysis

## Data Exploration

Dataset is consisted of 10299 manually labeled samples. Labels are walking, walking-upstairs, walking-downstairs, sitting, standing and laying, and their distribution is as follows:

* 1722 WALKING,
* 1544 WALKING\_UPSTAIRS,
* 1406 WALKING\_DOWNSTAIRS,
* 1777 SITTING,
* 1906 STANDING,
* 1944 LAYING

**Fig 1. Class distribution of the 6 activity classes**



Sampled data was already pre-processed by applying noise filters and re-sampled with fixed-width sliding windows of length 128 (2.56 sec/window) and 50% overlap. In addition, linear acceleration data was filtered from gravitational acceleration as it is almost constant and carries no information regarding the activities. A Butterworth low-pass filter was applied to remove the gravitational acceleration in frequency domain which was assumed to correspond to a frequency of less than 0.3Hz.

Filtered and re-sampled data was then converted into a feature vector of size 561. Features were subtracted from the properties in temporal and frequency domain. These include first order statistical properties like minimum, maximum, mean, standard deviation, and other variability measures like mean absolute deviation, interquartile range, and some other statistical properties like auto-regression coefficients, and correlation.

### Feature Transformation

The fundamental assumption in many predictive models is that the features are assumed to be *normally* distributed. Normal distribution is un-skewed. This means the probability of falling in the right or left side of the mean is equally likely. Skewness greater than zero shows a positively skewed distribution, while lower than zero shows a negatively skewed distribution.

The nonlinear transformation methods can help detecting the outlier points as it reduces the skewness of the distribution, and helps data to become more *normally* distributed.

I utilize different nonlinear functions to transform the data. These are natural logarithm, square root, and boxcox transformation. Before applying these methods, I check the dataset for abnormalities like missing values to make sure that there is no NaN values. In fact, this dataset has no missing values.

I will visualize the effect of the nonlinear feature transformation on the 16 features selected by using SelectKBest method from 561 features as it is much more difficult to visualize the original feature space.

### Outlier Detection

For every feature dimension, I apply Tukey’s method to mark potential outliers in that dimension. In order to make the final decision, I rank the outlier points based on the number of feature dimensions they are marked as outliers. In other words, the higher the rank of the data points, the more likely they are to be outliers. The list below shows the number potential outliers from lowest rank to highest rank:

2 features share 367 potential outliers

3 features share 825 potential outliers

4 features share 825 potential outliers

5 features share 716 potential outliers

6 features share 663 potential outliers

7 features share 893 potential outliers

8 features share 808 potential outliers

9 features share 704 potential outliers

10 features share 576 potential outliers

11 features share 426 potential outliers

12 features share 294 potential outliers

13 features share 156 potential outliers

14 features share 96 potential outliers

15 features share 50 potential outliers

16 features share 27 potential outliers

17 features share 31 potential outliers

18 features share 12 potential outliers

19 features share 15 potential outliers

20 features share 7 potential outliers

21 features share 9 potential outliers

22 features share 7 potential outliers

23 features share 2 potential outliers

24 features share 1 potential outliers

25 features share 8 potential outliers

28 features share 2 potential outliers

33 features share 1 potential outliers

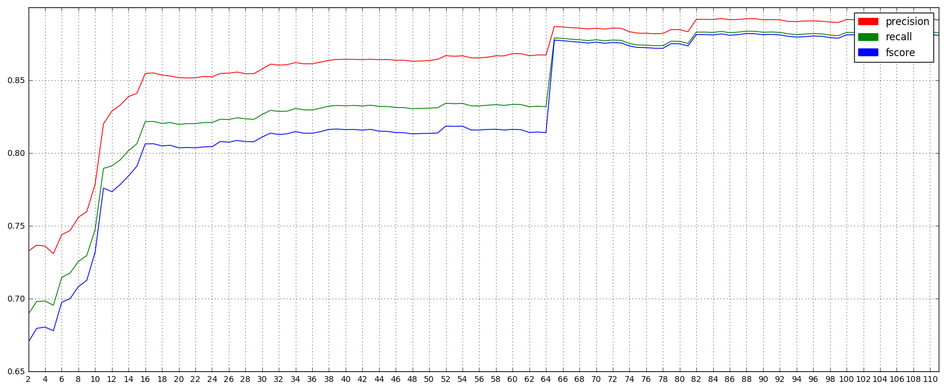
**34 features share 2 potential outliers**

Details about the outlier removal is given in the Data Preprocessing subsection under the Methodology section.

## Exploratory Visualization

I utilize SelectKBest feature selection method to reduce the dimension of the feature space to be able to visualize how features are distributed. Best “K” is decided based on the classification performance of the SVM model as shown in the figure below. Classifier selection is explained in next subsection.

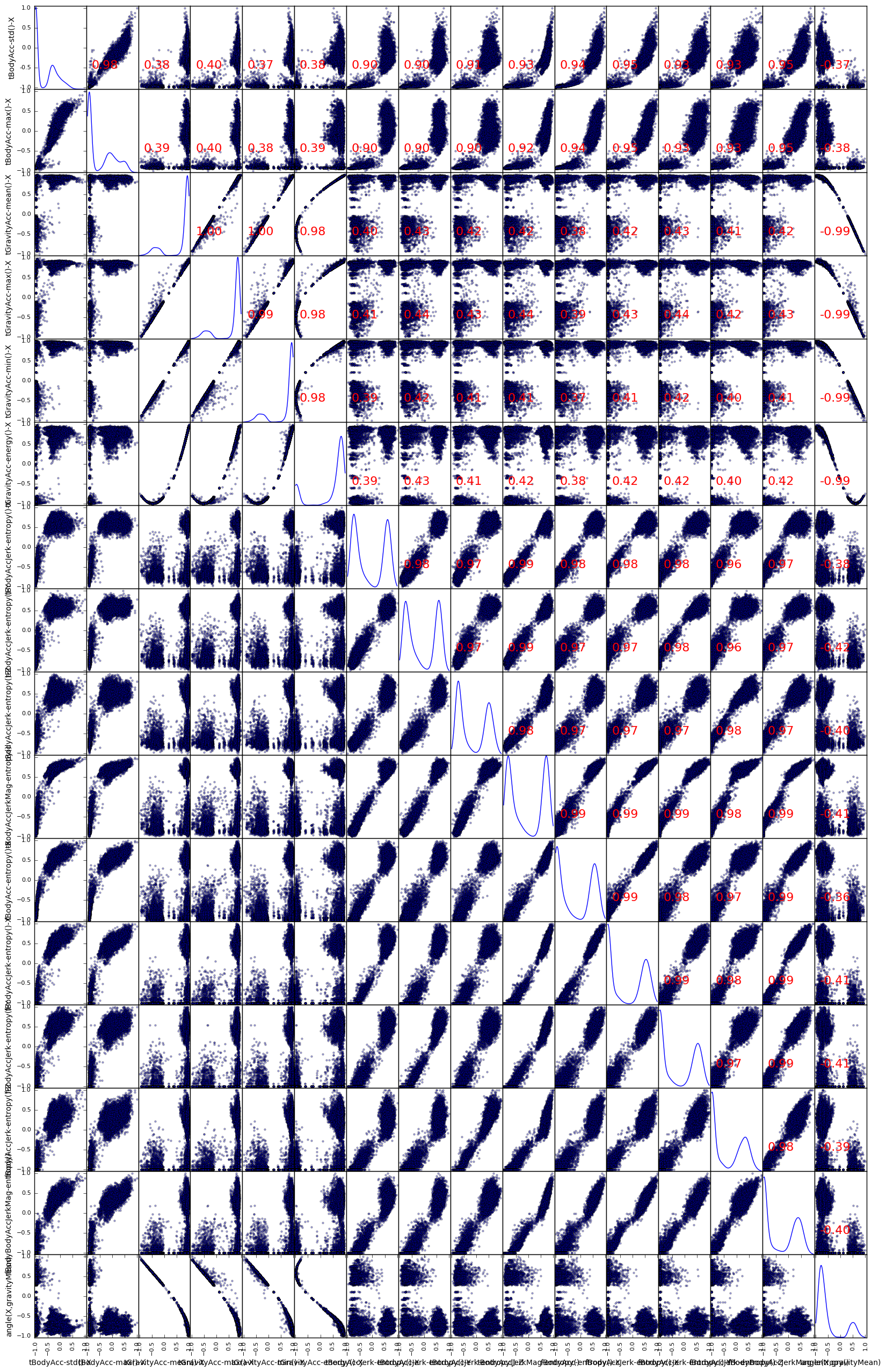
**Fig 2. Precision (Red), Recall (Green), F1-score (blue) vs number of KBest features**



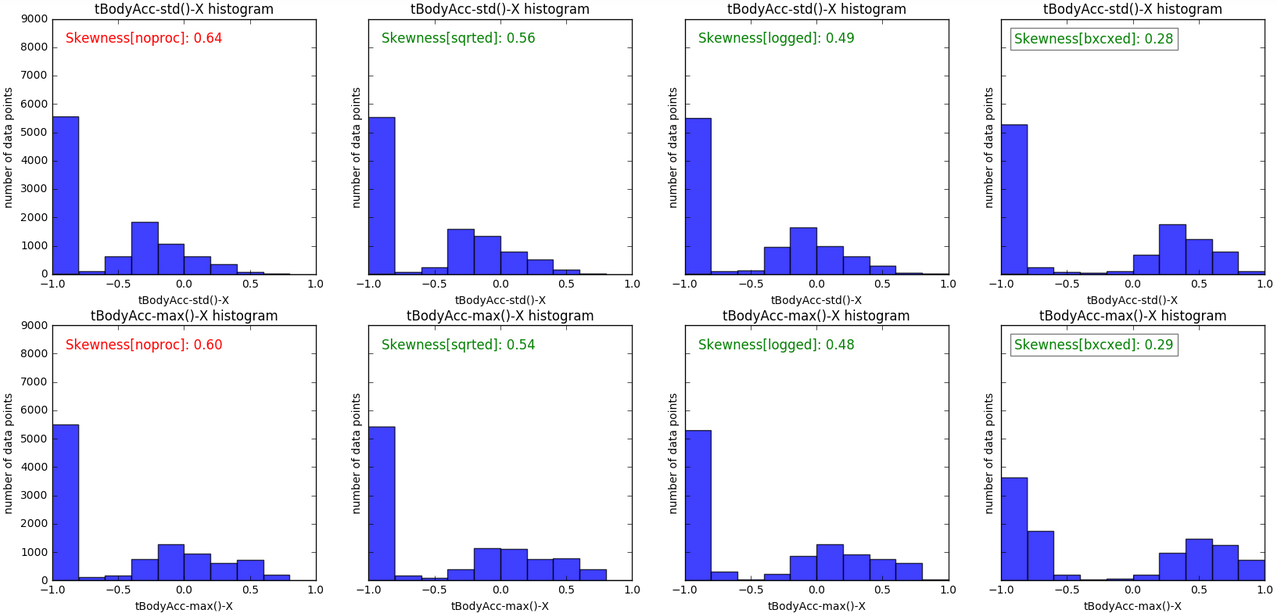
According to the graph above, SVM reaches the first peak performance (precision: 0.85, recall: 0.82, f-score: 0.81) when “K=16”. To further investigate how the features are distributed and correlated, I take these 16-Best features and plot the correlation matrix for them.

As we can see from the correlation matrix in the next page, redundant features can easily be distinguished by investigating their correlation with other features. For instance, third, fourth and fifth features are almost perfectly linearly proportional. Therefore, two of these features can be dropped. It also allows us to see the distribution of the feature values individually. Distribution of the most of the features are quite skewed, even mostly bimodal (especially the ones after the 7th feature). Transforming these features with natural logarithm, square root, or boxcoxing may help reduce the skew.

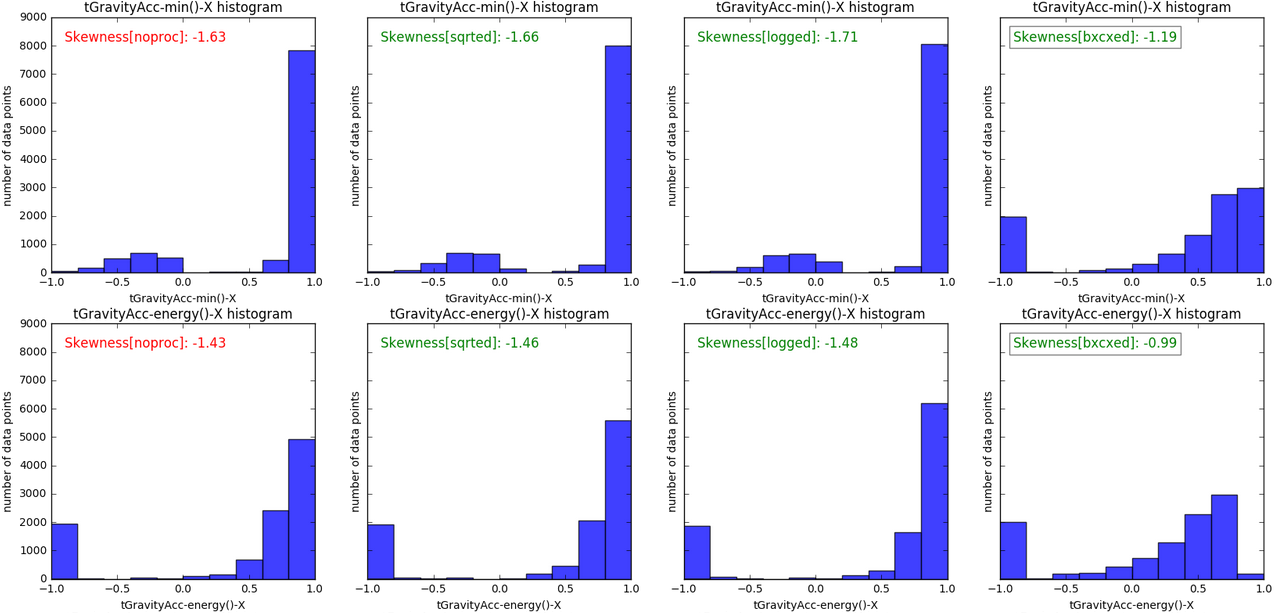
In the Feature Selection subsection of the Methodology section, I compare the SelectKBest method with Principal Component Analysis (PCA) to finally decide on the simplest feature space description that has most distinct features. I chose PCA because it helps testing the effect of feature transformation and SelectKBest method all together as PCA is executed on the raw data.

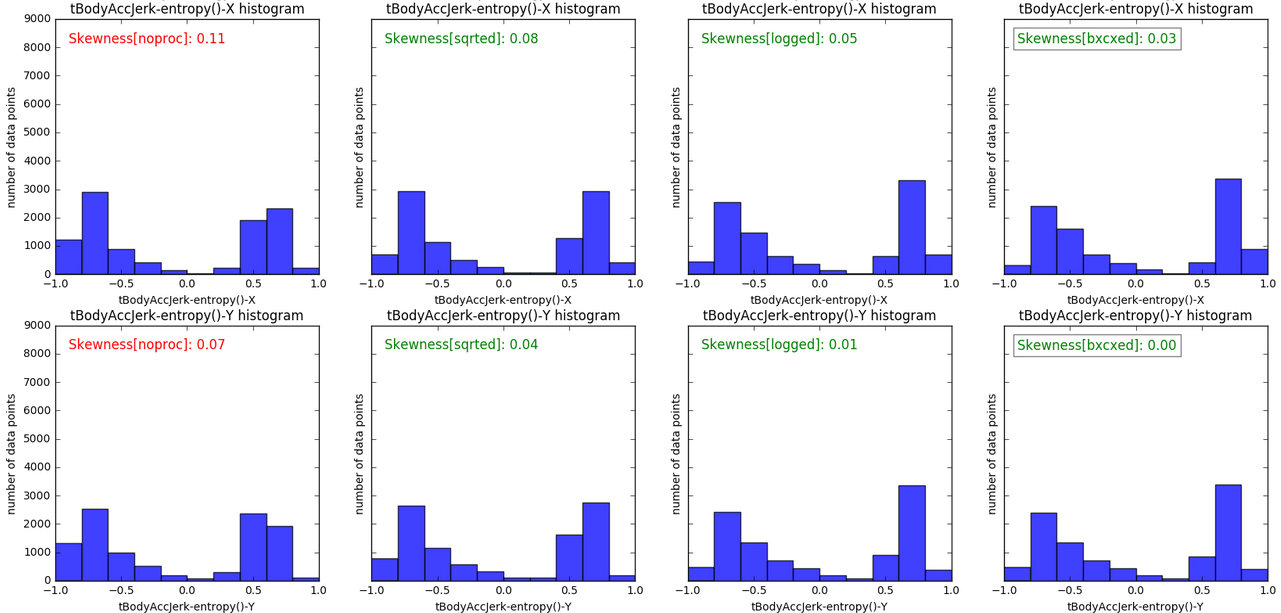


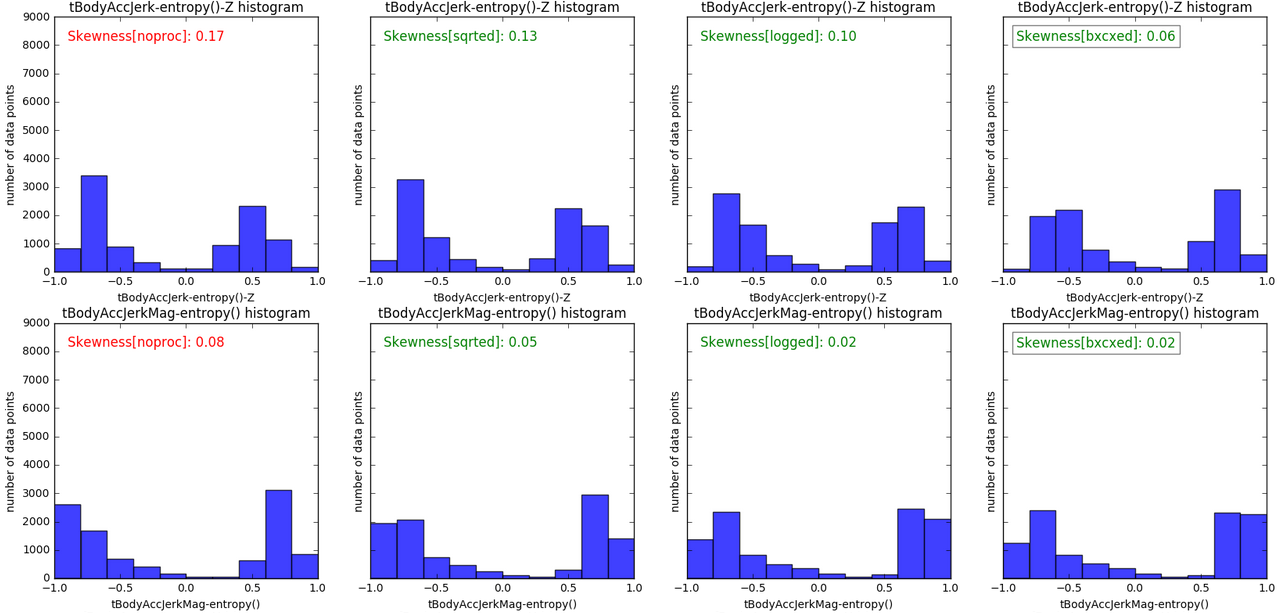
**Fig 3. Correlation Matrix of the 16-Best features**

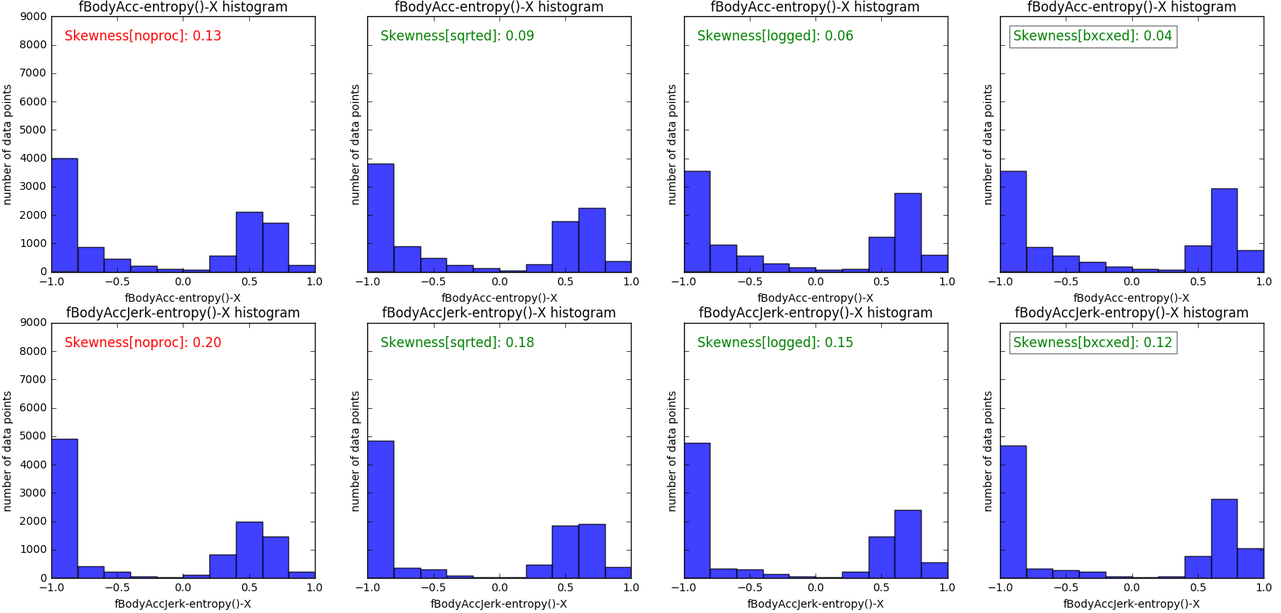


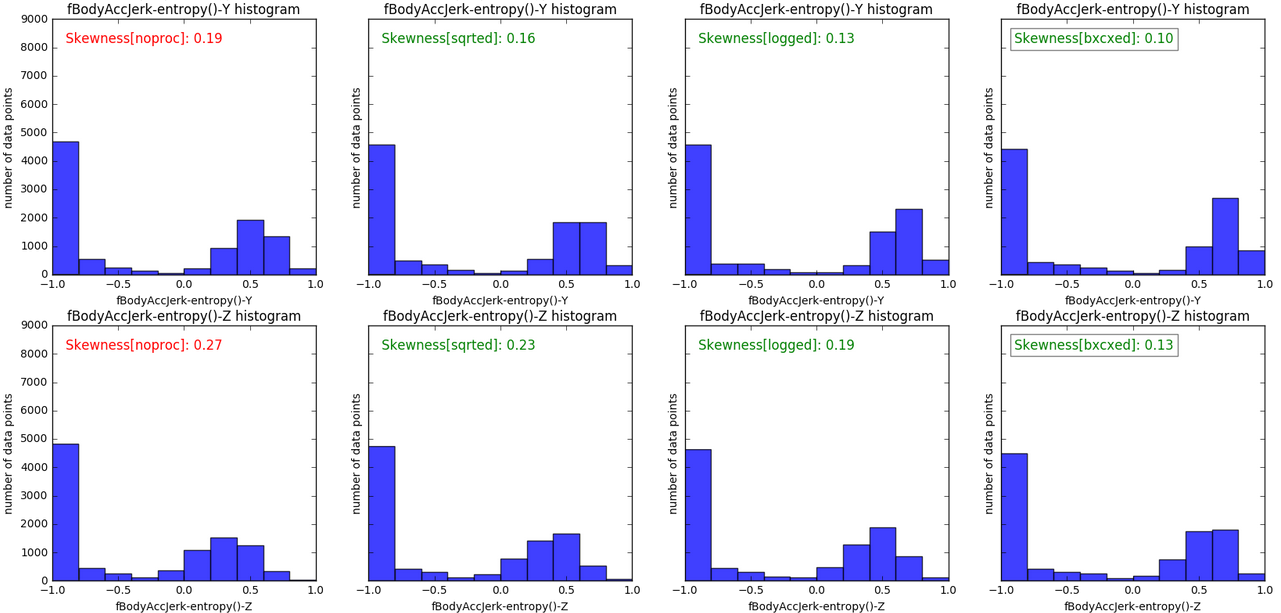


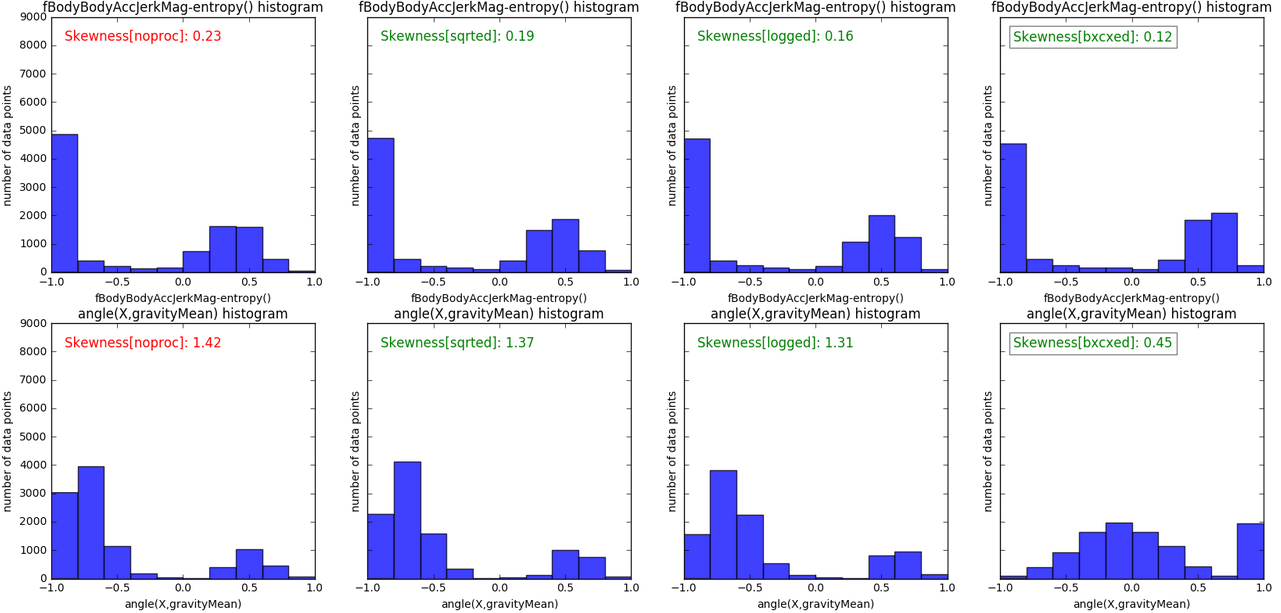












## Algorithms and Techniques

Ultimate goal of this project is to achieve the highest classification performance. Data preprocessing steps like outlier detection and feature transformation prepare dataset for the next step; feature selection and model training. Details about the effect of outlier removal and feature transformation is given in the Implementation section.

Feature selection and model training are two intertwined processes, both eventually depend on the model selection and evaluation.

As the dataset includes the class labels, we will use supervised machine learning methods to recognize the physical activities of the users. Among the supervised machine learning methods that are available in sklearn, *Stochastic Gradient Descent Classifier*, *Adaboost Classifier*, *K-Neighbors Classifier*, *Gaussian Naïve Bayes* and *Support Vector Machine* are the ones that provide class prediction probabilities. This is the most important criterion for this particular study. The reason is that the developed application is intended to ask for supervision of the user when the trained model doesn’t have a clear prediction about a test instance. This will in turn will help the model re-train and update itself based on the novel test instances it encounters. Novelty of a test instance is decided based on the class prediction probabilities or in other words confidence of the prediction. If the class prediction probabilities of a test instance close to each other, then this is not a confident prediction, therefore this means a novel case which require user supervision.

Final decision on the classifier method is made based on the benchmarking criteria explained in the Benchmark subsection. Feature Selection was guided by the results of the benchmarking to make decision upon two feature selection methods; *SelectKBest* and *Principal Component Analysis*.

## Benchmark

This project is a preliminary study towards building a machine learning application which will be able to learn from the user when a novel activity sequence is obtained. In other words, this application will ask for user’s supervision when it is not quite sure about the prediction result of an activity instance. In other words, it will re-train when a novel data instance is encountered[[1]](#footnote-1). This will require this application to (i) be mostly accurate in its predictions, (ii) be able to make quick predictions, (iii) be able to measure the confidence of the predictions, (iv) be able to learn and re-learn based on the given user input to be able to learn incrementally from small datasets to larger datasets having more than 10000 data points.

Therefore, following metrics are used to benchmark various methods in feature selection and model training.

1. Classification performance (precision, recall and fscore),
2. Testing time,
3. Availability of class prediction probabilities,
4. Training time.

# Methodology

## Data Preprocessing

### Feature Transformation

Here is the effect of the nonlinear transformations on the classification performance; no-transformation, square root, natural logarithm and boxcox respectively.

SVM notransform precision: 0.94 recall: 0.94 fscore: 0.93 t\_train: 10.06 t\_test: 2.44

SVM with sqrted precision: 0.94 recall: 0.94 fscore: 0.94 t\_train: 9.72 t\_test: 2.36

SVM with logged precision: 0.94 recall: 0.94 fscore: 0.94 t\_train: 9.61 t\_test: 2.34

**SVM with bxcxed precision: 0.94 recall: 0.94 fscore: 0.94 t\_train: 8.46 t\_test: 2.12**

All of the nonlinear transformations improved the training and testing time of the classification. Boxcox transformation improved the fscore, and improved the training and testing time more than the other nonlinear transformations. Therefore, in feature selection, Boxcox transformed dataset will be used.

### Outlier Removal

Potential outliers are ranked based on the number of feature dimensions they are marked as outliers. Highest ranked outliers have more priority in removal process. Outlier removal process is guided by the classification results. If removal of potential outliers don’t improve the classification performance, these data points are not removed from the dataset as they carry relevant information for the learning task.

Below, *NUMBER* in <SVM with *NUMBER*>specifies the size of the dataset after the potential outliers are removed starting from the highest rank (as it is explained in Outlier Detection subsection). Dataset includes 10299 data points and it reduces as the outlier points are removed as follows:

SVM with 10297 precision: 0.94 recall: 0.94 fscore: 0.94 t\_train: 8.44 t\_test: 2.13

SVM with 10296 precision: 0.94 recall: 0.94 fscore: 0.94 t\_train: 8.42 t\_test: 2.11

SVM with 10294 precision: 0.94 recall: 0.94 fscore: 0.94 t\_train: 8.44 t\_test: 2.12

SVM with 10286 precision: 0.94 recall: 0.94 fscore: 0.94 t\_train: 8.39 t\_test: 2.12

SVM with 10285 precision: 0.94 recall: 0.94 fscore: 0.94 t\_train: 8.40 t\_test: 2.12

SVM with 10283 precision: 0.94 recall: 0.94 fscore: 0.94 t\_train: 8.40 t\_test: 2.11

SVM with 10276 precision: 0.94 recall: 0.94 fscore: 0.94 t\_train: 8.38 t\_test: 2.11

SVM with 10267 precision: 0.94 recall: 0.94 fscore: 0.94 t\_train: 8.34 t\_test: 2.10

SVM with 10260 precision: 0.94 recall: 0.94 fscore: 0.94 t\_train: 8.32 t\_test: 2.09

SVM with 10245 precision: 0.94 recall: 0.94 fscore: 0.94 t\_train: 8.29 t\_test: 2.08

SVM with 10233 precision: 0.94 recall: 0.94 fscore: 0.94 t\_train: 8.30 t\_test: 2.09

SVM with 10202 precision: 0.94 recall: 0.94 fscore: 0.93 t\_train: 8.24 t\_test: 2.08

SVM with 10175 precision: 0.94 recall: 0.94 fscore: 0.93 t\_train: 8.19 t\_test: 2.07

SVM with 10125 precision: 0.94 recall: 0.94 fscore: 0.93 t\_train: 8.12 t\_test: 2.06

SVM with 10029 precision: 0.94 recall: 0.94 fscore: 0.93 t\_train: 8.05 t\_test: 2.04

SVM with 9873 precision: 0.94 recall: 0.93 fscore: 0.93 t\_train: 7.86 t\_test: 1.98

SVM with 9579 precision: 0.94 recall: 0.93 fscore: 0.93 t\_train: 7.55 t\_test: 1.90

SVM with 9153 precision: 0.94 recall: 0.93 fscore: 0.93 t\_train: 7.13 t\_test: 1.78

SVM with 8577 precision: 0.94 recall: 0.93 fscore: 0.93 t\_train: 6.56 t\_test: 1.60

SVM with 7873 precision: 0.93 recall: 0.93 fscore: 0.93 t\_train: 5.91 t\_test: 1.39

SVM with 7065 precision: 0.93 recall: 0.93 fscore: 0.92 t\_train: 5.23 t\_test: 1.17

SVM with 6172 precision: 0.93 recall: 0.92 fscore: 0.92 t\_train: 4.67 t\_test: 0.96

SVM with 5509 precision: 0.92 recall: 0.91 fscore: 0.91 t\_train: 4.26 t\_test: 0.79

SVM with 4793 precision: 0.92 recall: 0.91 fscore: 0.91 t\_train: 3.84 t\_test: 0.63

SVM with 3968 precision: 0.89 recall: 0.89 fscore: 0.89 t\_train: 3.35 t\_test: 0.47

SVM with 3143 precision: 0.88 recall: 0.88 fscore: 0.87 t\_train: 2.51 t\_test: 0.28

SVM with 2776 precision: 0.89 recall: 0.88 fscore: 0.88 t\_train: 2.13 t\_test: 0.22

As the outlier removal doesn’t improve the classification performance, no data is removed from the dataset.

## Feature Selection

Half of the 16-Best features are among the top 20% of the 561 features sorted by their skewness from low to high, and the other half are among the top 20%-65%. Therefore, it seems KBest feature selection considers skewness when it selects the most representative features and still it might be a good method to reduce the complexity of the learning space and improve classification performance.

Below are classification performances of (i) baseline SVM classification performance with 561 features with no feature transformation, (ii) SVM classification performance with 561 boxcoxed features and (iii) SVM classification performance with 16-Best boxcoxed features:

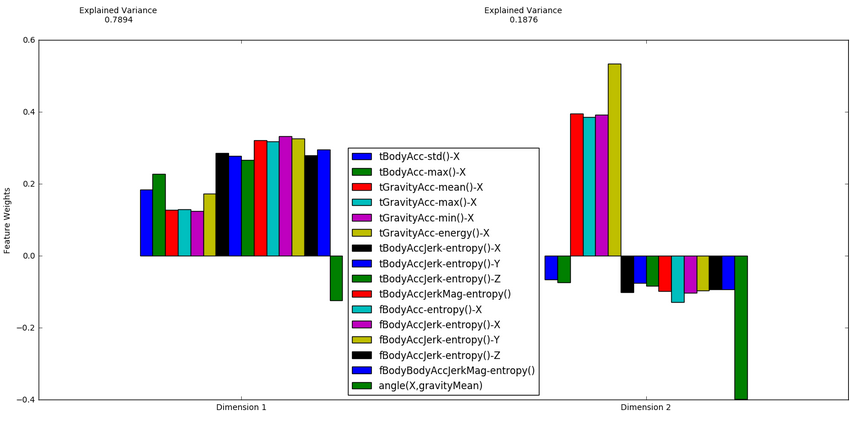
**SVM notransform precision: 0.94 recall: 0.94 fscore: 0.93 t\_train: 10.06 t\_test: 2.44**

**SVM with bxcxed precision: 0.94 recall: 0.94 fscore: 0.94 t\_train: 8.46 t\_test: 2.12**

**SVM KBest= 16 precision: 0.85 recall: 0.82 fscore: 0.81 t\_train: 0.88 t\_test: 0.20**

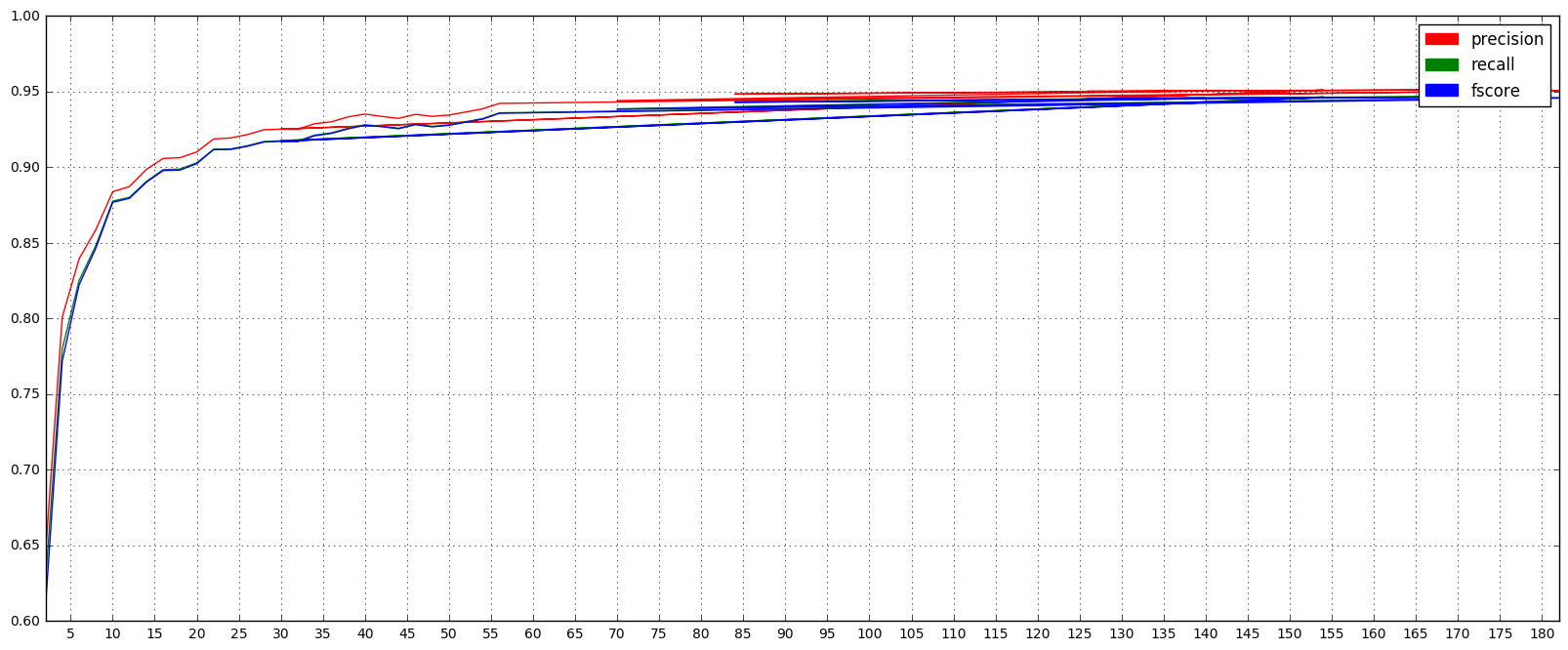
Although KBest feature selection helps reduce the time required for training and testing by 10 times, it sacrifices more than 10% of the precision, recall and fscore scores. As some of these 16-Best features were found to be correlated -- as it was shown in the correlation matrix before-, another feature selection method can be useful to test how expressive these features are. For this purpose, I checked the correlation of 16-Best features with the two principal components of the PCA.

**Fig 4. Major and minor principal components and their relationship with 16-Best features**



This figure shows the level of alignment of each feature with the principal components. Major principal component has a power of 0.79 to explain the variance of the 561 dimensional feature space. It seems like none of the 16-Best features are substantially aligned with the major principal component. Therefore, this principal component can actually improve the classification performance if it is added to the 16-Best features. To further investigate the importance of the components selected by PCA, I did an exhaustive search and visualized the classification performance according to the number of principal components. PCA is done on the original dataset, not on the boxcox-ed dataset because transformed feature-set yielded inferior classification performance.

**Fig 5. Precision (Red), Recall (Green), F1-score (blue) vs number of components of PCA**



Peak classification performance is obtained with 117 components. Compared to the previous results, SVM with 117 components performs the best in terms of precision, recall and fscore metrics. However, training and testing times are 2 times worse than the 16-Best features.

**SVM(561) notransform precision: 0.94 recall: 0.94 fscore: 0.93 t\_train: 10.06 t\_test: 2.44**

**SVM(561) with bxcxed precision: 0.94 recall: 0.94 fscore: 0.94 t\_train: 8.46 t\_test: 2.12**

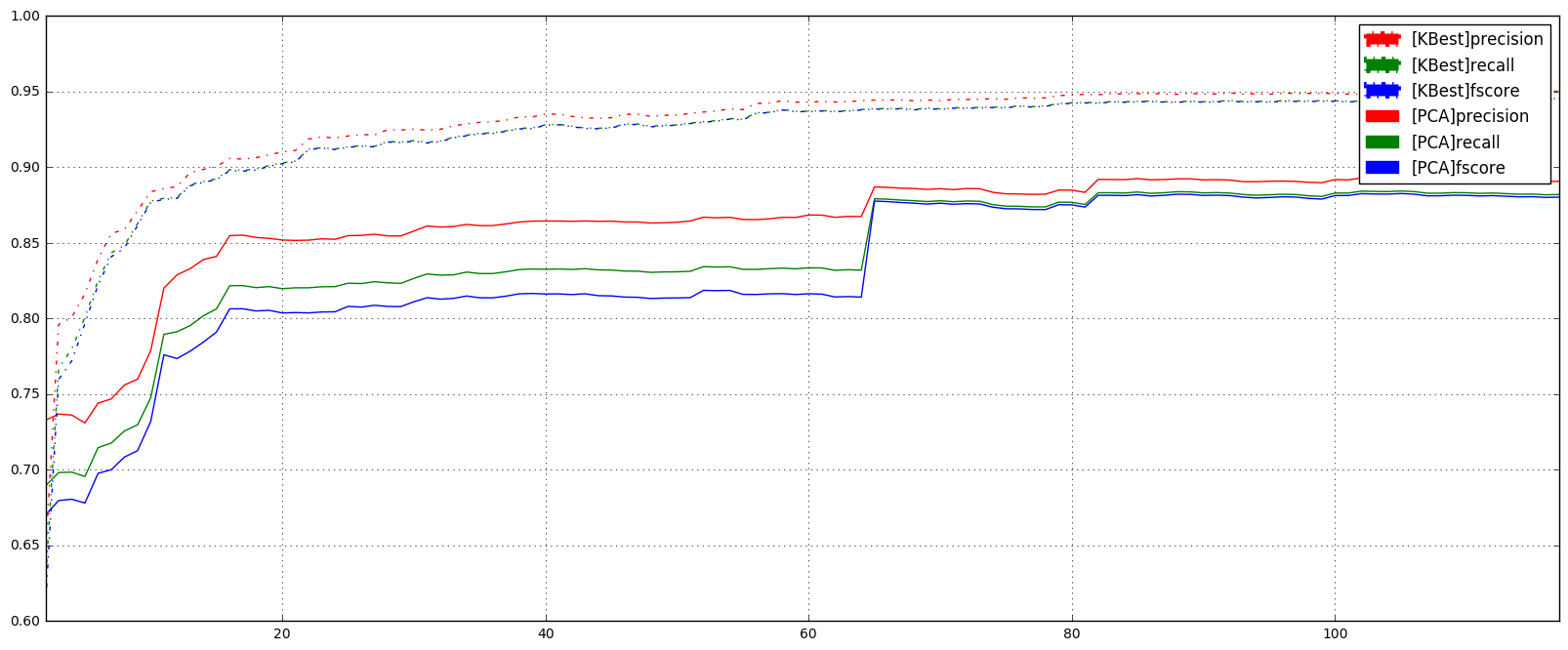
**SVM( 16) KBest= 16 precision: 0.85 recall: 0.82 fscore: 0.81 t\_train: 0.88 t\_test: 0.20**

**SVM(117 ) KBest=117 precision:0.89 recall:0.88 fscore:0.88 t\_train:3.57 t\_test:0.91**

**SVM(117) npca =117 precision: 0.95 recall: 0.95 fscore: 0.95 t\_train: 1.52 t\_test: 0.36**

As shown in figure 6, PCA’s components are better for the classification performance. In other words, KBest features don’t add additional expression power. Therefore, there is no need to use the features obtained by SelectKBest when we can use the PCA’s components as features.

**Fig 6. Classification performance of SVM with PCA and KBest selection methods vs Feature Vector Size**



## Implementation

Feature transformation, feature selection/projection, outlier detection, classifier selection, classifier tuning; these are five main tasks that needed special attention to improve the classification performance or the activity recognition system designed with this work.

So far, I used SVM to make the final decision on the feature transformation method, feature selection method and the outlier removal.

For feature transformation, I applied natural logarithm, square root and boxcox method on the dataset to obtain less skewed dataset. Although boxcox improved the performance of the classifier more than other nonlinear feature transformation, PCA on the boxcoxed transformed data yielded a worse result compared to the PCA applied on the original data.

For feature selection, I applied KBest and PCA methods. Features obtained by KBest yielded worse classification performance compared to the PCA components. Merging the features obtained by KBest and components obtained by PCA didn’t yield a better performance either. I decided to use PCA for feature selection and this made the feature transformation irrelevant as PCA with transformed features yielded worse results compared to the PCA with original feature set.

For outlier removal, I utilized Tukey’s method to mark the outliers. I sorted the outliers based on their rank. Outliers were removed from dataset starting from the highest ranked outliers. However, outlier removal didn’t yield a better classification performance. These results suggest that outlier removal is not necessary.

No feature transformation or outlier removal yielded a better classification performance than the classification done with the 117 dimensional principal components.

SVM was used as the primary classification method and I decided to challenge SVM with SGD as they both provide comparable classification performances better than the other supervised machine learning methods:

GNB precision: 0.80 recall: 0.73 fscore: 0.72 t\_train: 0.2316sec t\_pred: 0.0533sec

**SVM precision: 0.94 recall: 0.94 fscore: 0.93 t\_train: 13.0880sec t\_pred: 3.1674sec**

**SGD precision: 0.95 recall: 0.94 fscore: 0.94 t\_train: 0.4460sec t\_pred: 0.0050sec**

Ada precision: 0.37 recall: 0.54 fscore: 0.41 t\_train: 41.1263sec t\_pred: 0.0405sec

KNC precision: 0.91 recall: 0.91 fscore: 0.91 t\_train: 0.9203sec t\_pred: 11.7007sec

SVM and SGD have the highest precision, recall and f1-score. SGD is the quickest in prediction and second quickest in training. SVM model had the following parameters that comes with the SVC model in *scikit learn by default*:

SVC(C=1.0, cache\_size=200, class\_weight=None, coef0=0.0, decision\_function\_shape=None, degree=3, gamma='auto', kernel='rbf', max\_iter=-1, probability=False, random\_state=None, shrinking=True, tol=0.001, verbose=False)

For classifier selection, I used the metrics like precision, recall and fscore. These metrics are measured by using 10-fold cross-validation. Results obtained at each fold are then mean-ed to get the final results.

## Refinement

To further increase the classification performance of the SVM model, I applied cross-validated grid search to find the best parameters for the trained model with the following parameter settings:

{'C': [1, 10, 100, 1000], 'gamma': ['auto'], 'kernel': ['linear']}

{'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001], 'kernel': ['rbf']}

SVM model with parameters -- 'kernel': 'rbf', 'C': 100, 'gamma': 0.001-- revealed the best result. Tuned classifier trained and classification performance was measure by using 10-fold cross-validation.

Default parameter settings of the SVM doesn’t include class imbalance that was mentioned in the Data Overview subsection. Setting the *class\_weight* parameter to ‘*balanced*’ did not change the classification performance and increased the training and testing time by 20%:

**SVM(117) tuned precision: 0.96 recall: 0.95 fscore: 0.95 t\_train: 0.87 t\_test:0.17 [tuned, PCAed, no class balancing]**

**SVM(117) PCAed precision:0.95 recall:0.95 fscore:0.95 t\_train:1.19 t\_test:0.23 [tuned, PCAed, with class balancing]**

Finally, I checked other classifiers with the PCAed dataset (viz.dataset w/ the 117 principal components):

**DTR precision: 0.79 recall: 0.79 fscore: 0.79 t\_train: 2.09 t\_test: 0.00**

**SGD precision: 0.95 recall: 0.95 fscore: 0.95 t\_train: 0.14 t\_test: 0.00**

**GNB precision: 0.81 recall: 0.79 fscore: 0.79 t\_train: 0.05 t\_test: 0.02**

**Ada precision: 0.33 recall: 0.42 fscore: 0.30 t\_train: 11.87 t\_test: 0.02**

**SVM precision: 0.96 recall: 0.95 fscore: 0.95 t\_train: 0.87 t\_test: 0.17**

**KNC precision: 0.91 recall: 0.91 fscore: 0.91 t\_train: 0.13 t\_test: 1.66**

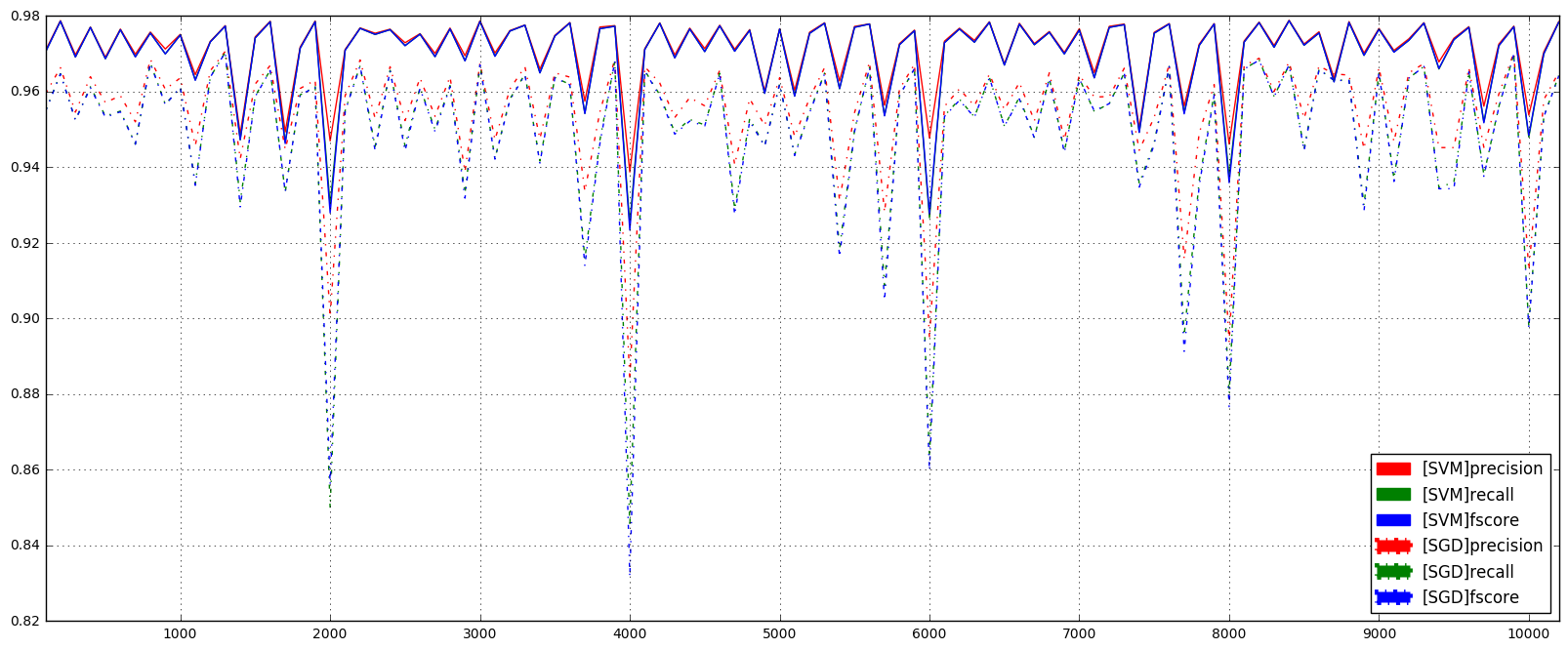
These results revealed that SGD and SVM are two best alternatives for this task. Final evaluation is done based on the robustness of these models against missing data. Second evaluation will be based on their classification performance with respect to the dataset size.

# Results

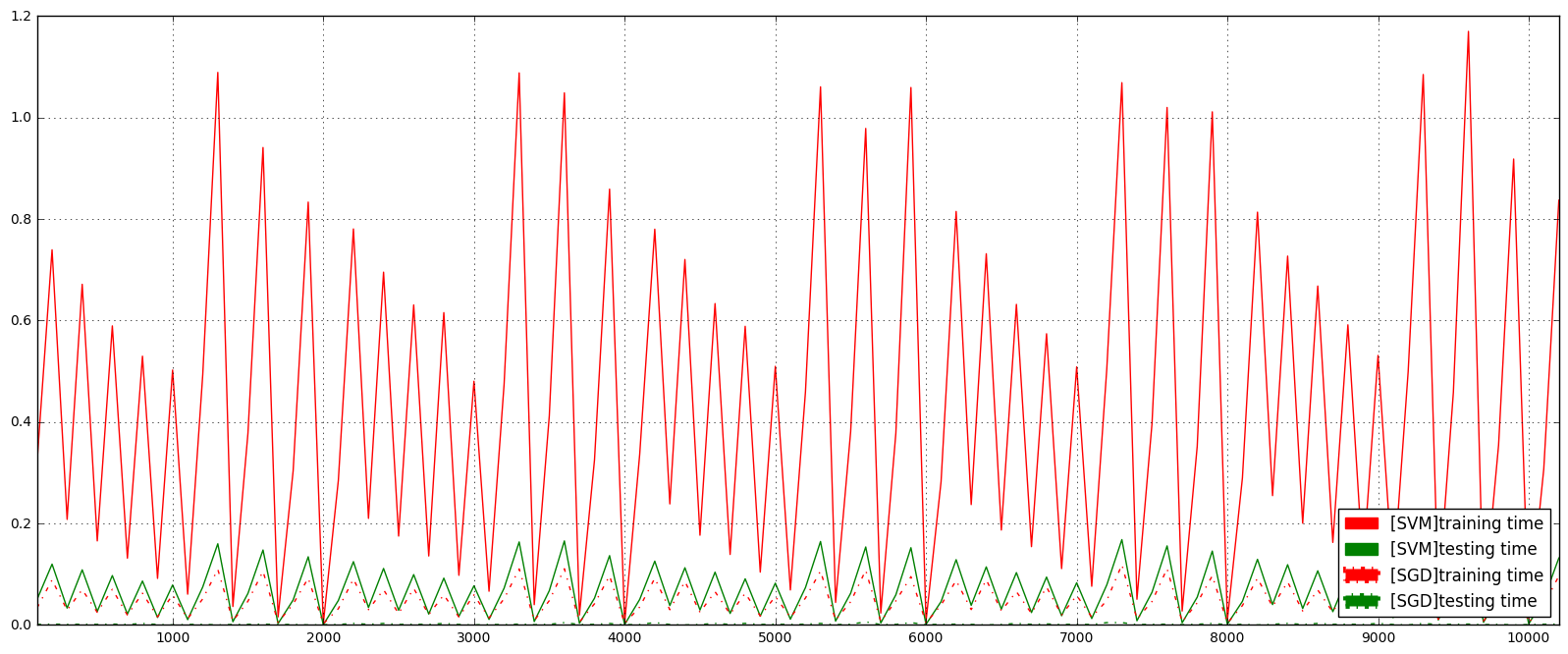
## Model Evaluation and Validation

Classification performance is measured against the dataset size to see the behavior of the trained model in a real life scenario where the data is slowly gathered and the model is trained with the available dataset. Below is the classification performance of the SVM and SGD when the dataset is being collected slowly and trained 100 points at a time and reaches up to more than 10000 points. We can see that SVM performs better than SGD all the time. SGD is better in training and testing times compared to SVM, but the difference is less than a second. Therefore, SVM is still acceptable as the classifier of choice.

**Fig 7. SVM and SGD vs dataset size**

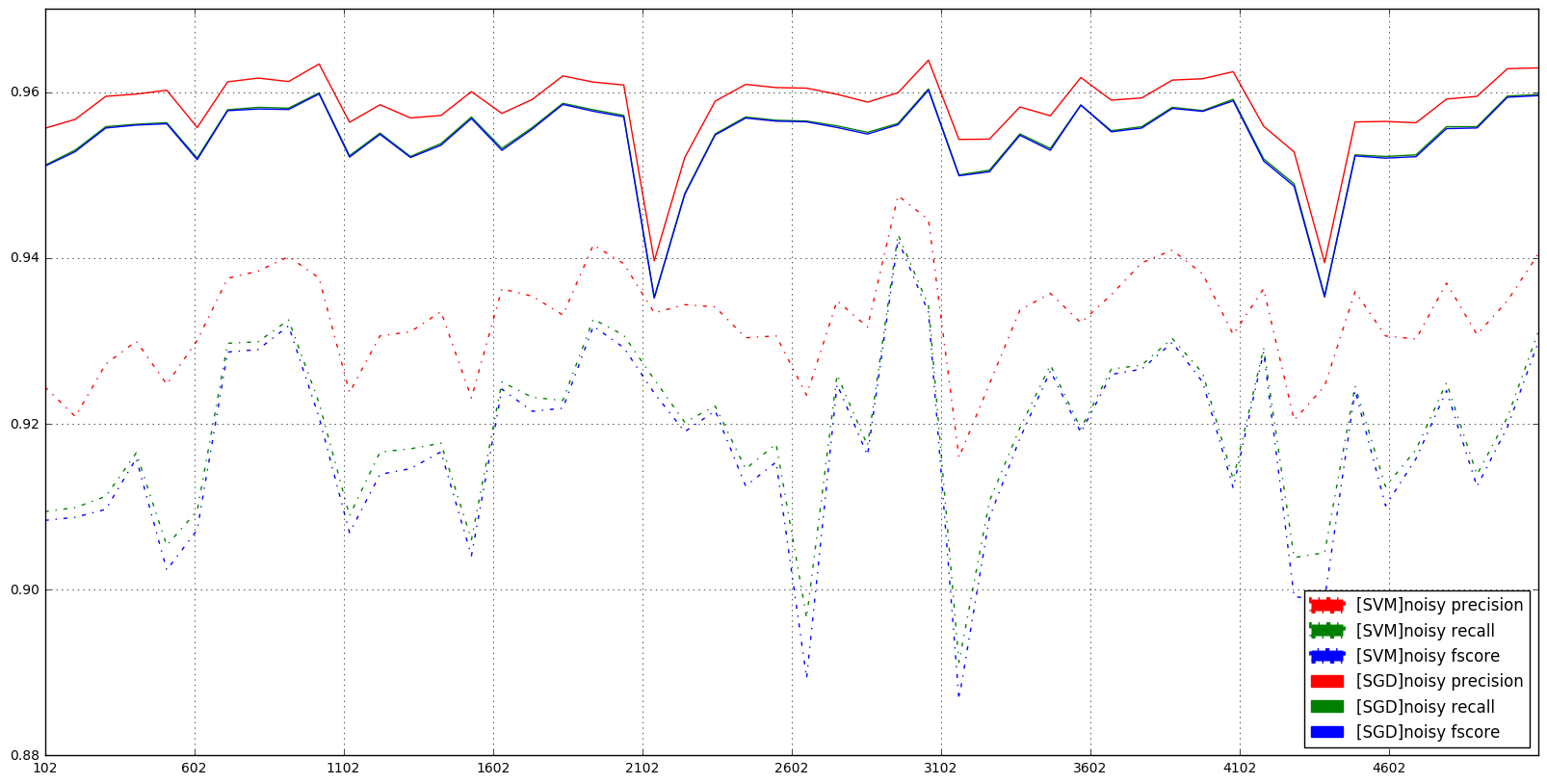


**Fig 8. SVM and SGD training and testing times vs dataset size**

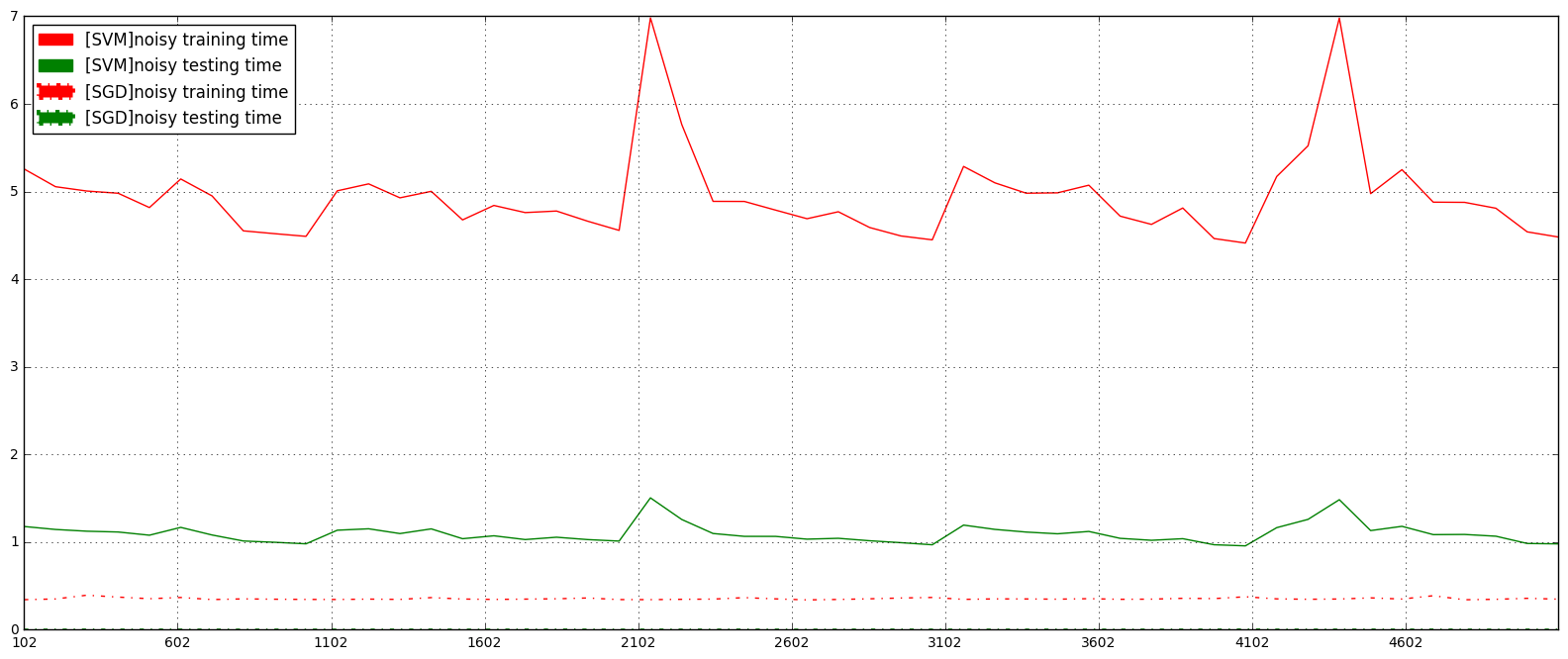
****

During data collection sometimes data might be corrupted, dropped or very noisy. Therefore some of the values for the features might be missing or useless. Below is the simulation of this scenario where I introduce missing data values and fill them with the mean value of the corresponding feature in the whole dataset. In other words, I approximate the missing value of a feature with the mean value of the corresponding feature of the remaining data. For this, I implemented a basic impurity test to see the performance of the SVM classifier against the missing data. I randomly replaced the **half of the feature values** of the data instances with the NaN values. I checked the classification performance of the SVM against number of data instance which had half of the feature values im-purified as shown in the following graph. Only visible change was on the classification time and testing time, classification performance was not affected by this test. SVM is more robust against impurities compared to SGD and performs better, but slower in training and testing times more than before.

**Fig 9. Tuned SVM and SGD classification performance versus missing data points**



**Fig 10. Tuned SVM and SGD training and testing times versus missing points**



# Conclusion

This study shows that SVM is a reasonable method to classify physical activities that are represented by the motion-related sensor data obtained from body-worn smartphones. SGD performs the classification task quite fast but is not as accurate as SVM. Nonlinear transformation methods and outlier removal methods do not always improve the classification results therefore data processing methods should always be checked a classification method to see their contribution in the classification performance. SelectKBest feature selection method is good for reducing the feature space dimensions but PCA can perform better. Therefore, feature selection should be accompanied by the classification to decide on the final method and the size of the reduced/transformed feature space.

Here is a summary of the obtained results; classification with default data, classification with boxcoxed data, classification with 16-Best features, classification with 117-Best features, classification with 117 PCA components, and classification with 117 PCA components and tuned classifier.

SVM(561) noprocced precision: 0.94 recall: 0.94 fscore: 0.93 t\_train: 10.06 t\_test: 2.44

SVM(561) bxcxed precision: 0.94 recall: 0.94 fscore: 0.94 t\_train: 8.46 t\_test: 2.12

SVM( 16) KBested precision: 0.85 recall: 0.82 fscore: 0.81 t\_train: 0.88 t\_test: 0.20

SVM(117) KBested precision:0.89 recall:0.88 fscore:0.88 t\_train:3.57 t\_test:0.91

**SVM(117) PCAed precision: 0.95 recall: 0.95 fscore: 0.95 t\_train: 1.52 t\_test: 0.36**

**SVM(117) tuned precision: 0.96 recall: 0.95 fscore: 0.95 t\_train: 0.87 t\_test:0.17**

Dataset used in this study was already quite clean and very well labeled. To see the real life classification performance, we need to run these methods with a raw dataset and with semi-supervision. These are left for further studies later on.

# References

Bulling, Andreas, Ulf Blanke, and Bernt Schiele. "A tutorial on human activity recognition using body-worn inertial sensors." *ACM Computing Surveys (CSUR)* 46.3 (2014): 33.

1. This is usually called concept drift or incremental learning but for this project I will take a naïve approach and focus on offline learning. [↑](#footnote-ref-1)