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 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 experimented 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 561 dimensional feature space representing the daily life physical activities into the categories corresponding to the activities labeled as *walking, walking-upstairs, walking-downstairs, sitting, standing*, and *laying*. To achieve this goal:

1. Dataset was tested if there are outlier data points which would the learning task more challenging,
2. Various feature transformation and selection methods are utilized to find an alternative representation to increase inter-class distinction,
3. Various supervised machine learning models are trained. Their classification and prediction performances are compared. Finally, SVM model has been chosen to carry out other data processing and tuning operations.

My hypothesis is that processes like outlier removal, feature transformation, feature selection and model-parameter tuning will yield better and better classification performance. Contribution of each process will be compared by training a classifier model via SVM.

## Metrics

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

I extracted *precision, recall,* and *f-score* metrics for every single folding. I considered the weighted average of the metrics based on the number of class labels. This way, obtained metrics are more robust against the imbalance between different class labels.

I took the mean of these metrics for every single folding to obtain final metrics regarding the performance of the classification methods.

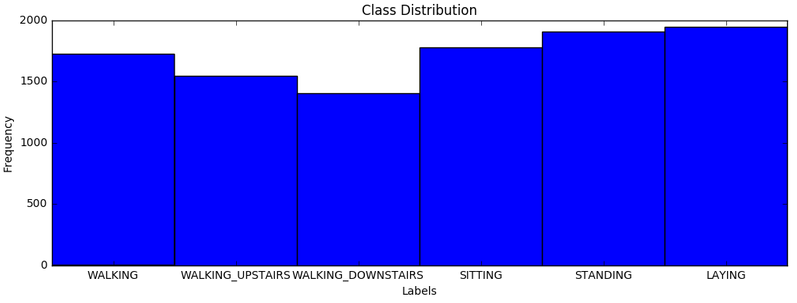
# Analysis

## Data Overview

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 were 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 are 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.

## Classification Overview and Early Analysis

As the ultimate goal of this project is to reach to the highest classification performance, we can start from the very end by choosing the classification method and the metrics that will be helpful in choosing the data-processing methods.

In order to decide on the classification method, I applied various supervised machine learning methods that are available in scikit learn. These are Stochastic Gradient Descent Classifier, Ada-boost Classifier, Decision Tree Classifier, K-Neighbors Classifier, Gaussian Naïve Bayes and Support Vector Machine. Here is the classification performance of these 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

DTR precision: 0.88 recall: 0.87 fscore: 0.87 t\_train: 7.0560sec t\_pred: 0.0027sec

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. I will be using SVM as the main method while discovering different feature processing methods with the default parameters that comes with the SVC model in *scikit learn*:

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)

Now, we have a baseline comparison for the upcoming data-processing methods which is

**Precision (0.94), recall (0.94), fscore (0.93), training time (13.1s), prediction time (3.2s)**.

Benchmarking details will be provided later on in the Benchmark subsection.

## Data Exploration

Checking the dataset for abnormalities like missing values resulted in no NaN values.

Before checking the dataset for outliers, I applied a feature transformation method to help data to become more *normally* distributed and *less skewed*. The reason is because of the fundamental assumption in many predictive models that 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 selection of the feature transformation function will be explained in the Feature Transformation subsection.

### Outlier Removal

For every feature dimension, Tukey’s method was applied to mark potentially outlier data points. In order to make the final decision, I ranked the outlier points based on the number of feature dimensions they were 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-> (start removal from these points)**

Therefore, if I decide to remove outliers, I will start from these highest ranked 2 points marked in 34 feature dimensions. Below, *NUMBER* in <SVM with *NUMBER*>specifies the size of the dataset after the potential outliers are removed starting from the highest rank.

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 classifier performance doesn’t change for the good, I will not remove these points. Therefore, there is no outlier in this dataset.

## Feature Transformation

As it is briefly explained in the Data Exploration section, nonlinear transformation can help detecting the outlier points as it reduces the skewness of the distribution. I will visualize the effect of the nonlinear transformation after I introduce feature selection in the next section.

I utilized different nonlinear functions to make the transformation. These are natural logarithm, square root, and boxcox transformation. Here is the effect of transformation on the classification performance:

SVM noprocced 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**

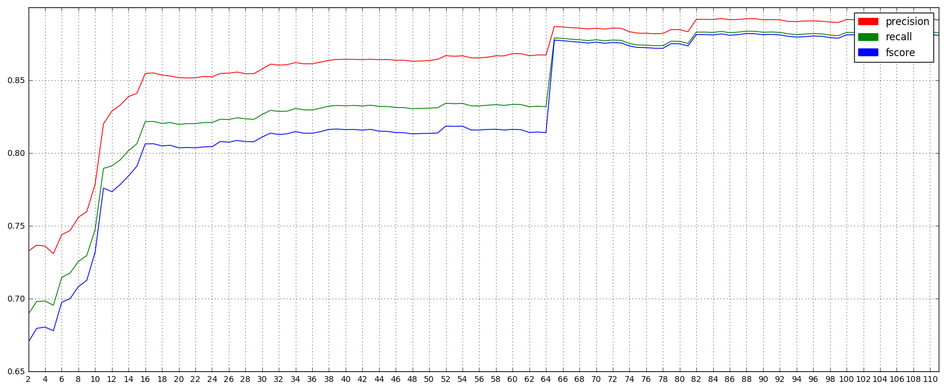
Boxcox transformation improved the fscore just a bit, and improved the training and testing time more than natural logarithm and square rooting.

It is not easy to see how these transformation effect the distribution of data. In the next section, I will work on the feature selection. I will visualize the effect of the nonlinear feature transformation on the selected features.

## Exploratory Visualization

I utilized SelectKBest feature selection method to reduce the dimension of the feature space. Best “K” is decided based on the classification performance.

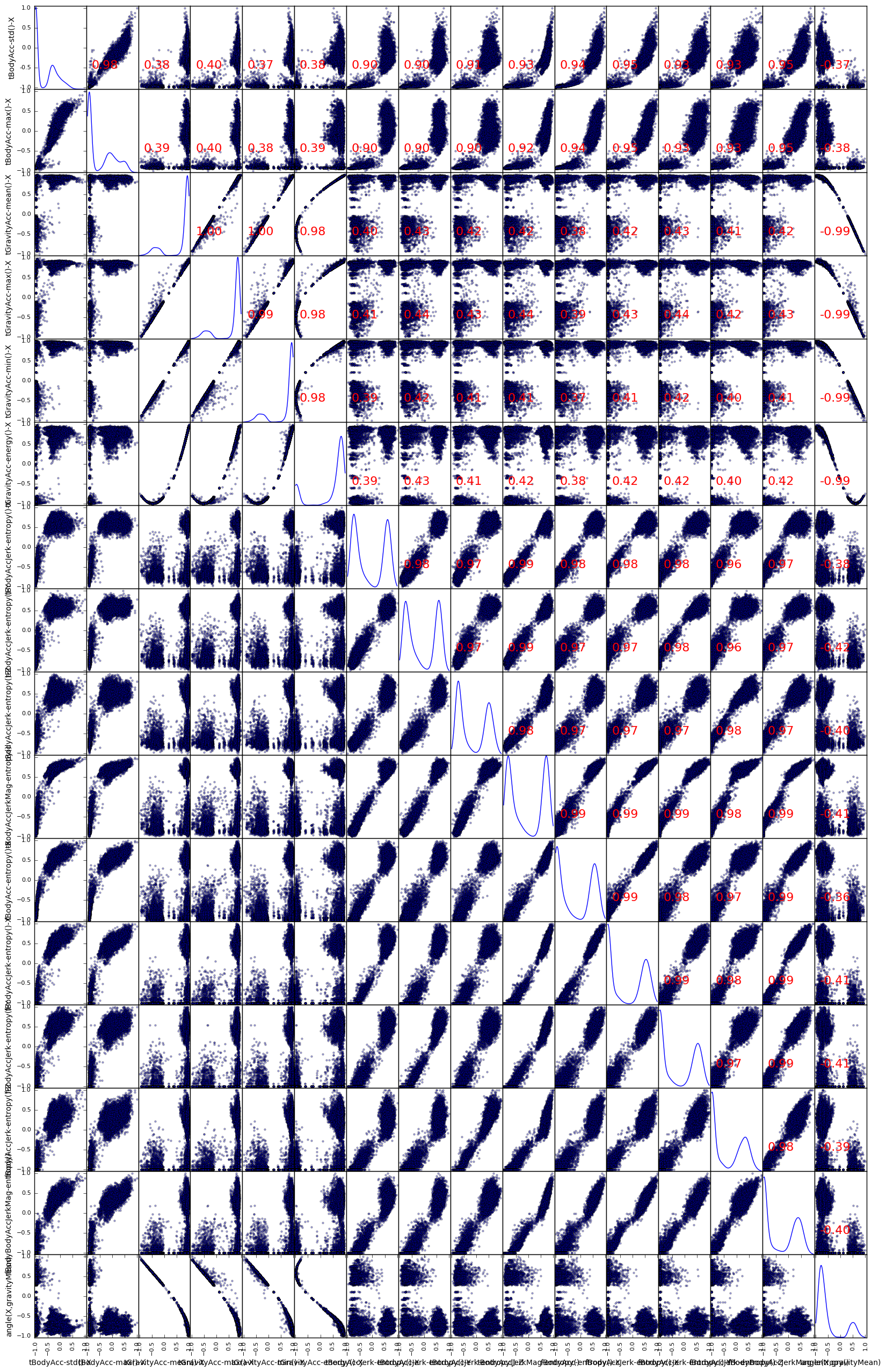
**Fig 2. Precision (Red), Recall (Green), F1-score (blue) vs number of KBest features**



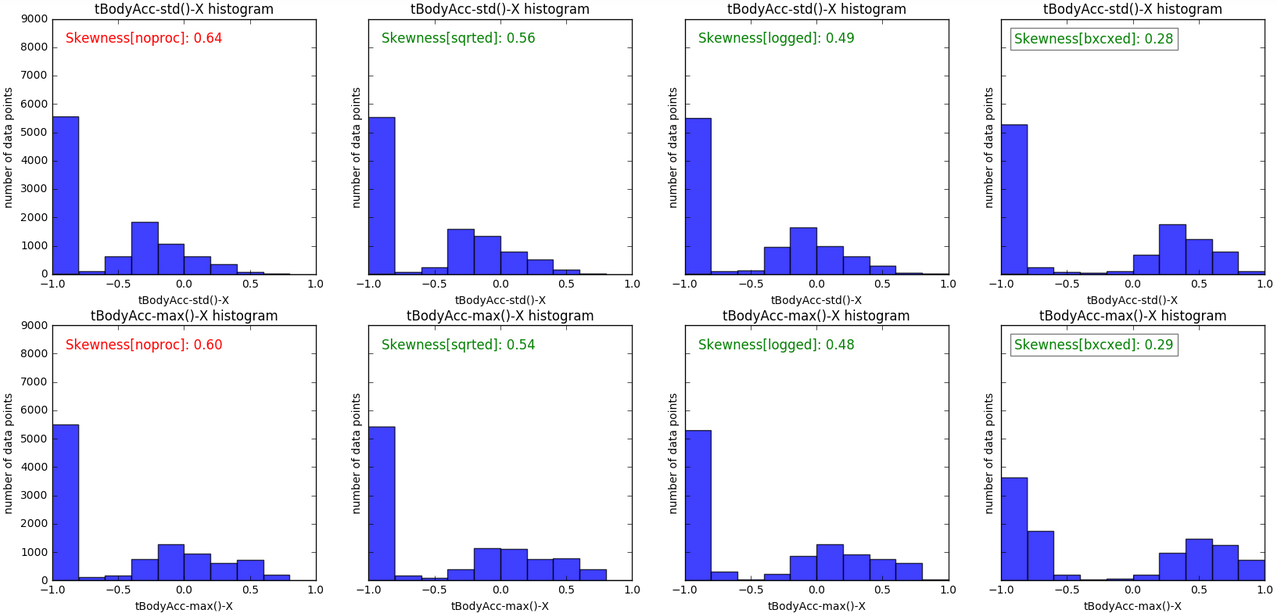
Moreover, redundant features can easily be distinguished by investigating their correlation with other features. If the correlation is high, it may mean that there is high correlation. Correlation matrix will allow us to see the distribution of the feature values individually and also their correlation with others.

According to the graph above, we see that 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 plotted the correlation matrix for the 16-Best features.

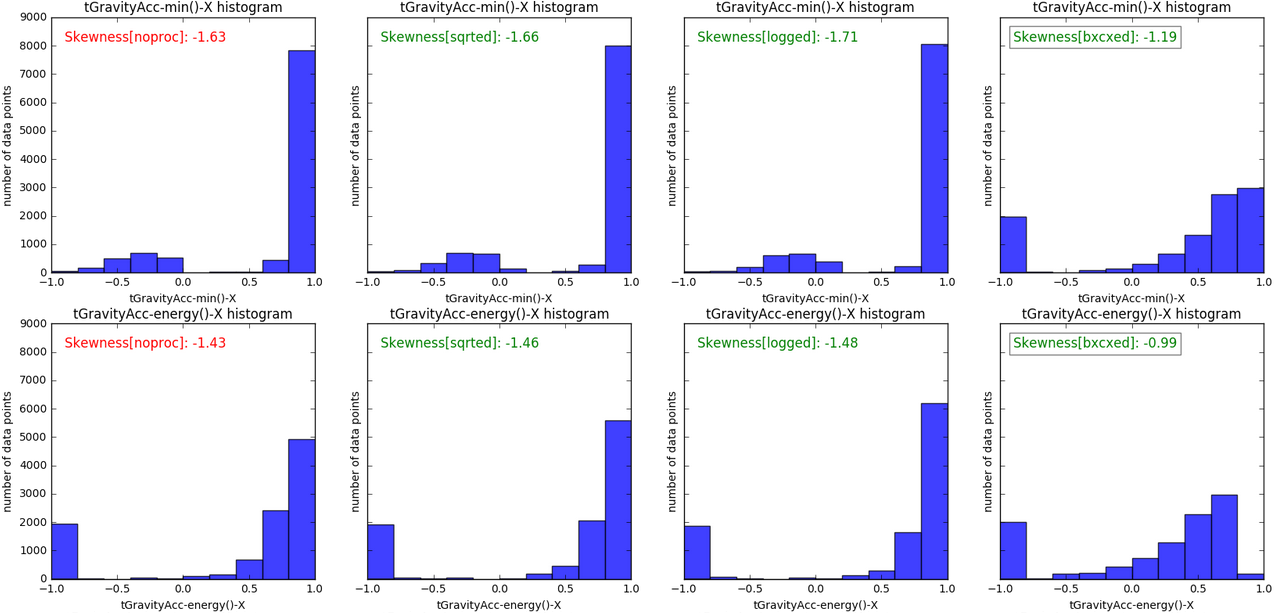
As we can see from the correlation matrix in the next page, these features are quite skewed, even mostly bimodal. Transforming these features with natural logarithm, square root, or boxcoxing may help reduce the skew.

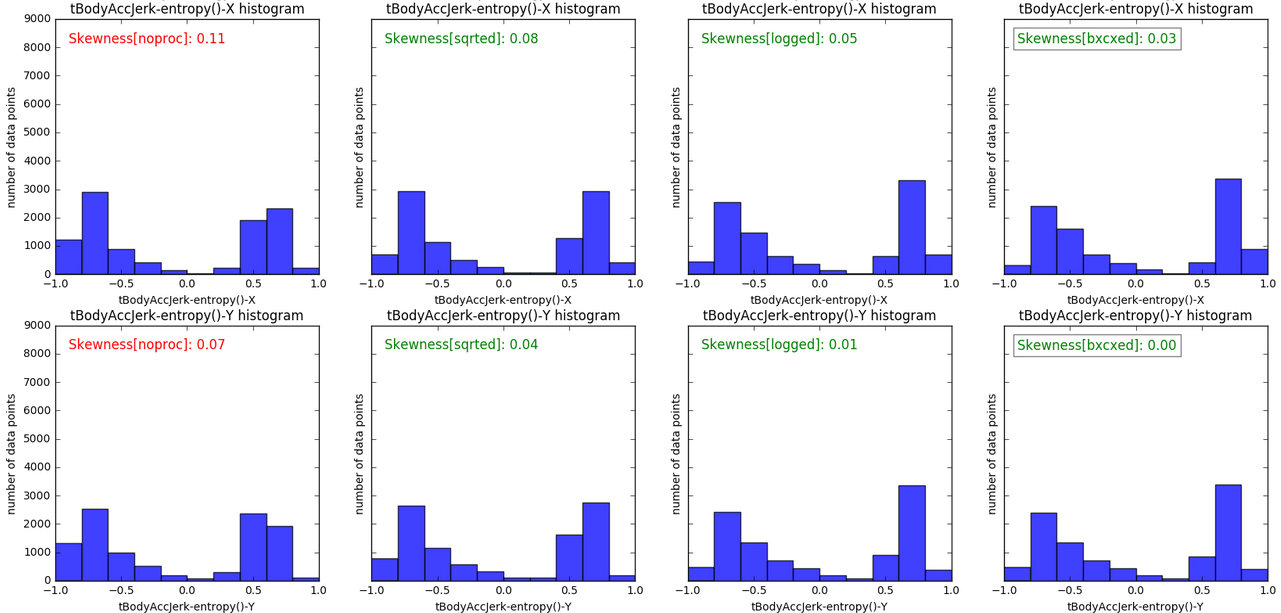


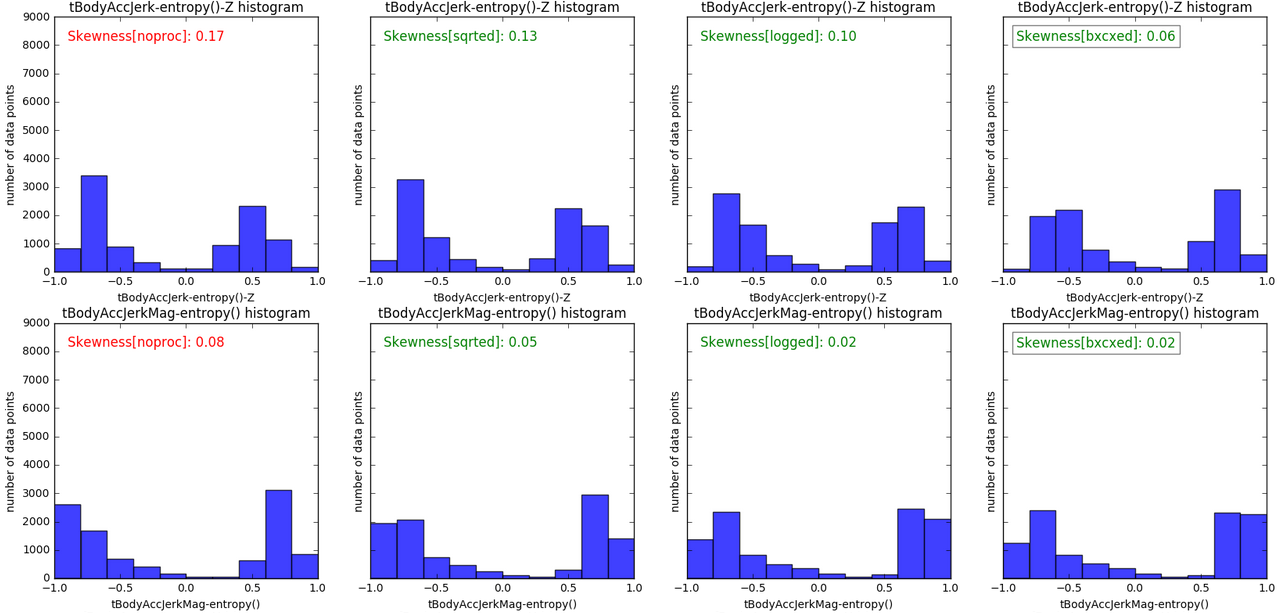
**Fig 3. Correlation Matrix of the 16-Best features**

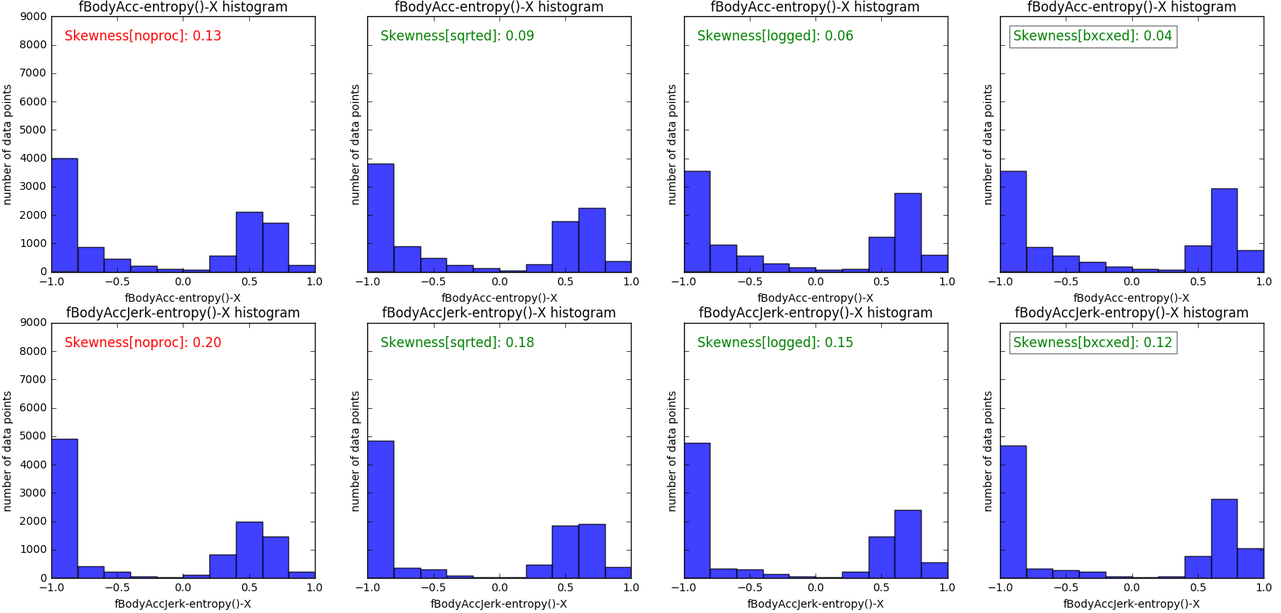


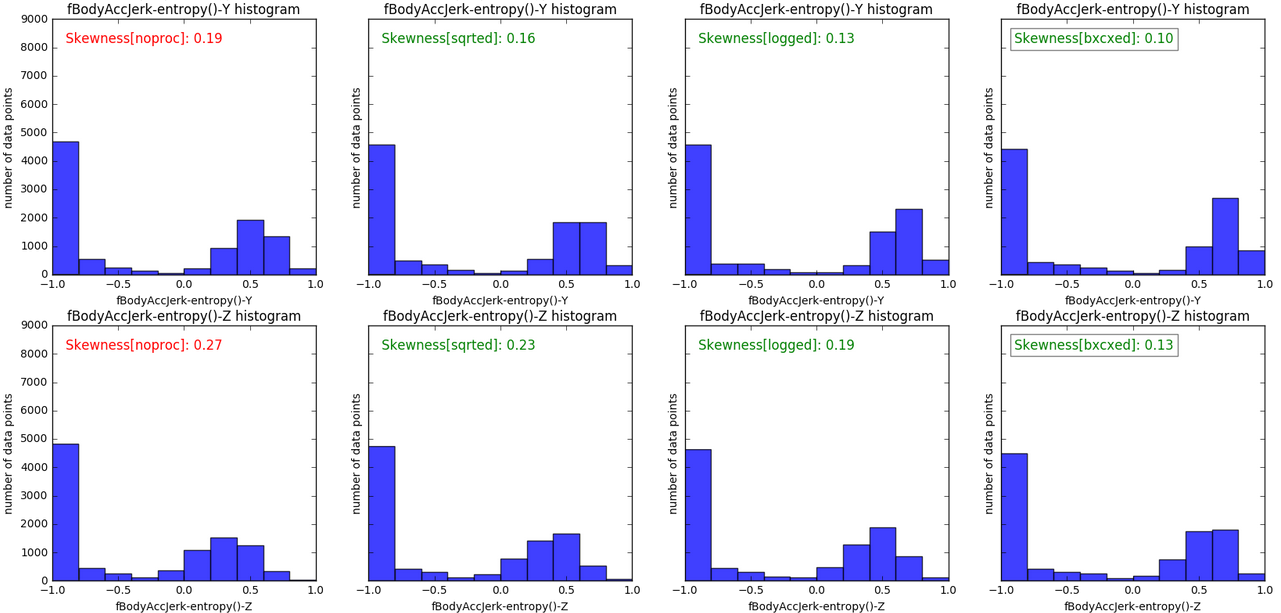


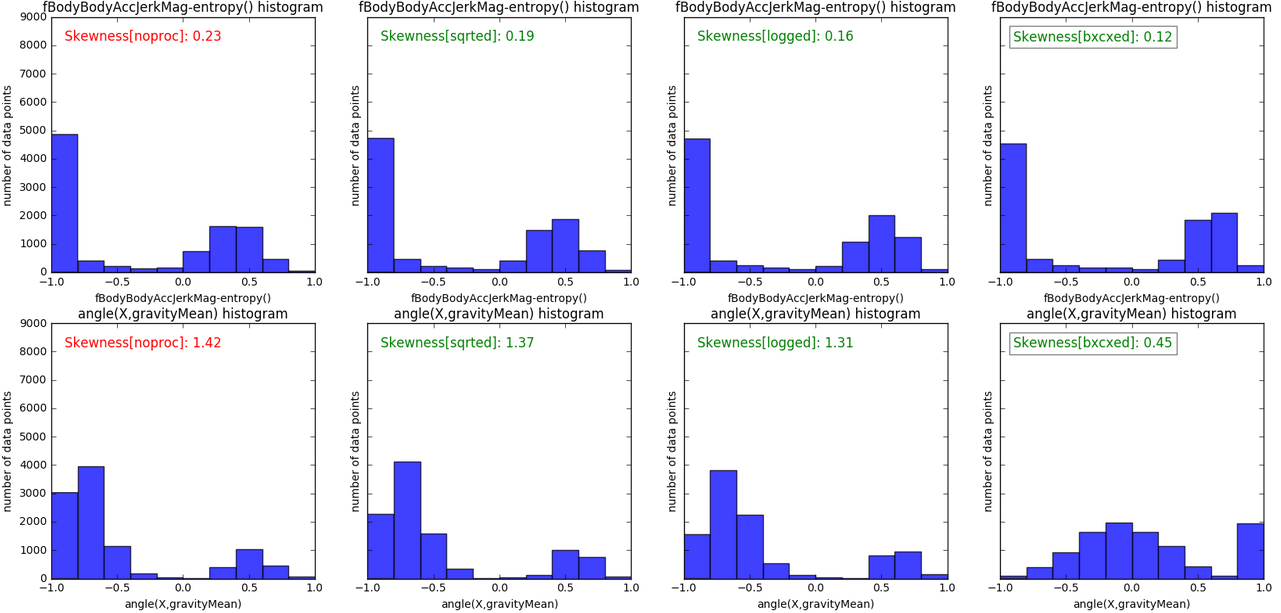












Half of the 16-Best features are among the top 20% of the 561 features sorted by their skewness from low to high, and the other half are among the top 20%-65%. Therefore, it seems KBest feature selection consider skewness when it selects the most representative features.

# Algorithms and Techniques

## Feature Selection

Below are two classification performances, (i) baseline SVM classification performance with 561 boxcoxed features and (ii) SVM classification performance with 16-Best boxcoxed features:

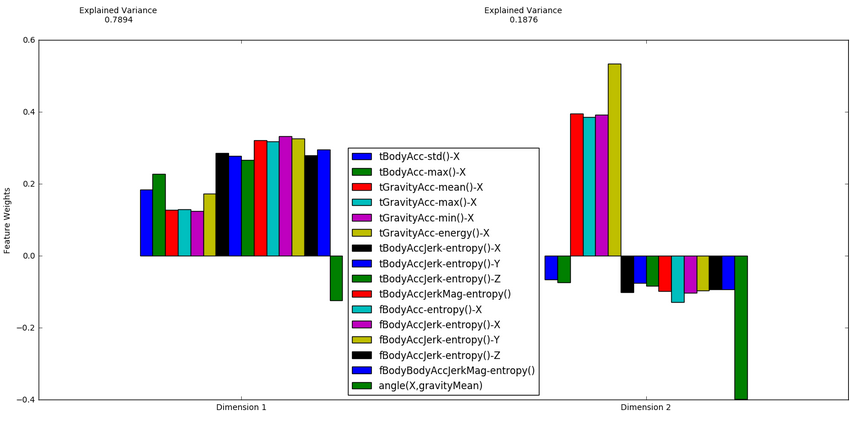
**SVM with bxcxed precision: 0.94 recall: 0.94 fscore: 0.94 t\_train: 8.46 t\_test: 2.12**

**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. Below I will try to augment these features with the principal component analysis.

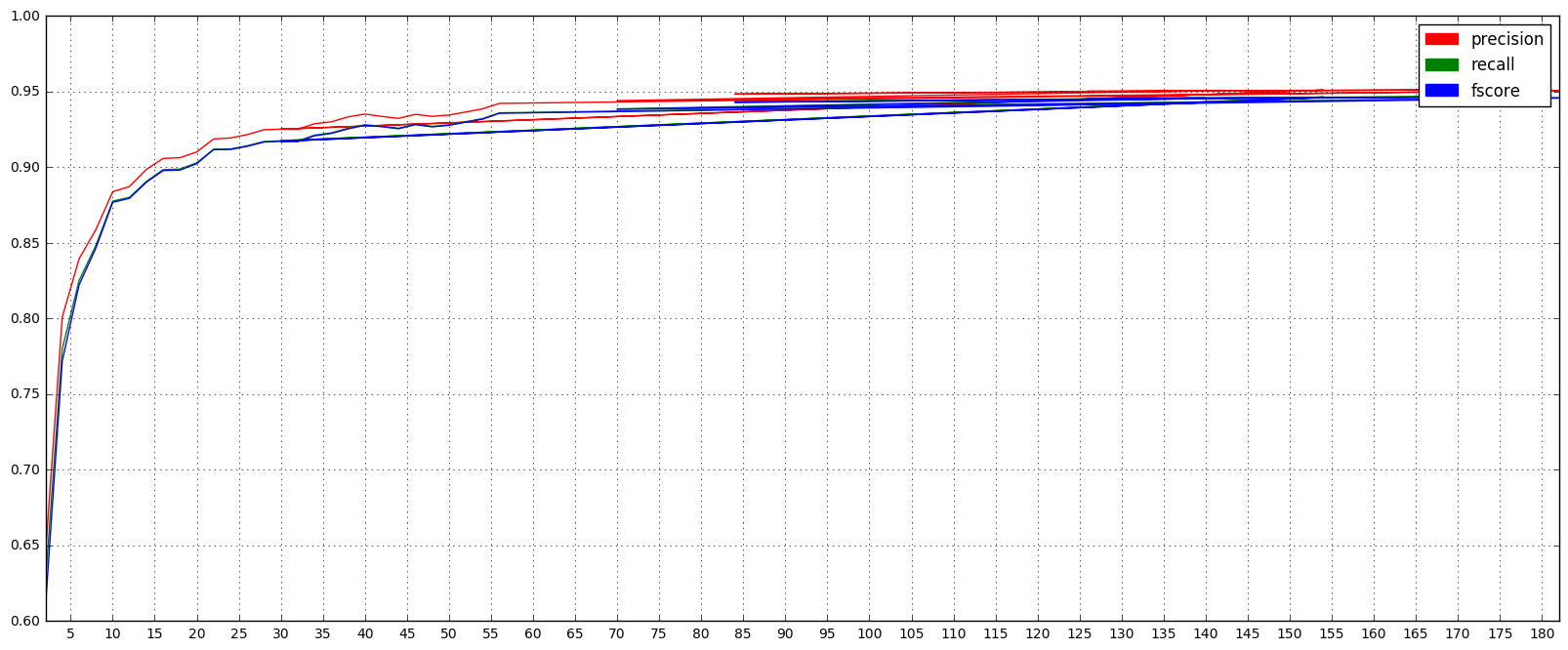
First, to see how expressive these 16-Best features are, I checked their correlation with principal components of the PCA.

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



This figure shows the level of alignment each feature has 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 that yields inferior classification performance.

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



Peak classification performance was 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 time is 2 times worse than the 16-Best features.

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

**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**

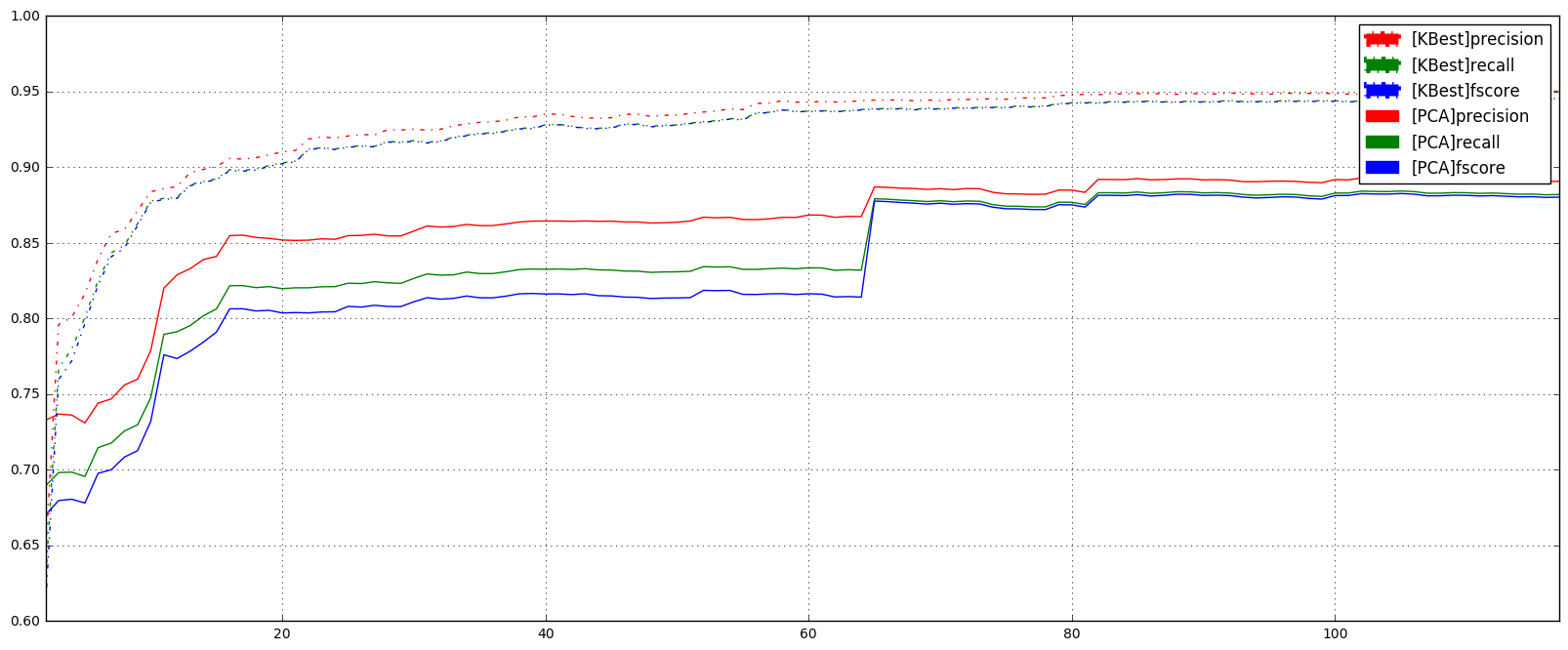
**SVM(111) KBest=111 precision: 0.89 recall: 0.88 fscore: 0.88 t\_train: 2.52 t\_test: 0.64**

SVM yielded in better classification performance and worse training and testing time. Therefore, there might be a combination (components and KBest features) that might result in the best results. After training the SVM model with the different combinations, I got the following best result:

**SVM[kbest 2 npca117] precision:0.95 recall:0.95 fscore:0.95 t\_train:1.54 t\_test:0.36**

This has no additional improvement over the SVM with pca=117. Therefore, for the best classification performance we can’t go above the 0.95 for precision, recall and fscore, not with the feature transformation nor selection. According to the Figure 2 and Figure 5, PCA is a better choice than KBest for feature selection anytime. Therefore, KBest doesn’t add additional expression power next to the PCA.

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



# Model Training

SVM was chosen among many other supervised machine learning methods as it was explained in Classification Overview and Early Analysis. Below are classification performance of the SVM classifier, (i) with the 561 dimensional feature vector with no processing, (ii) with the 561 dimensional feature vector after the feature transformation with Boxcox method, (iii) with the 117 dimensional feature vector that is obtained by Principal Component Analysis.

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

To further increase the performance of the classifier, 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:

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

## Benchmark

Classification performance was measured in terms of precision, recall and fscore metrics as well as training time and testing time.

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.

90% classification accuracy based on precision, recall and f1-score is quite acceptable. As a Machine Learning Application Developer, we can create an interface which can help user to improve the classifier if the confidence level of a prediction is lower than a certain threshold. This way the classifier will be able to get supervision from the user when it is not confident about the prediction and be able to learn from that.

# Methodology

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.

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, Principal Component Analysis (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 performance compared to the components obtained by PCA. Merging the features obtained by KBest and components obtained by PCA didn’t yield a better performance compared to only PCA either.

For outlier detection and removal, I utilized Tukey’s method but detected potential outliers removed the performance of the classifier when removed from the dataset.

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.

For the classifier tuning, I used a primitive bread-first parameter search to check all the parameter combinations. Tuned classifier trained and classification performance was measure by using 10-fold cross-validation.

Here is a summary of the obtained results.

**SVM noprocced 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(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**

Finally, I checked other classifiers with the final dataset configuration to give the final decision on the classifier.

**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**

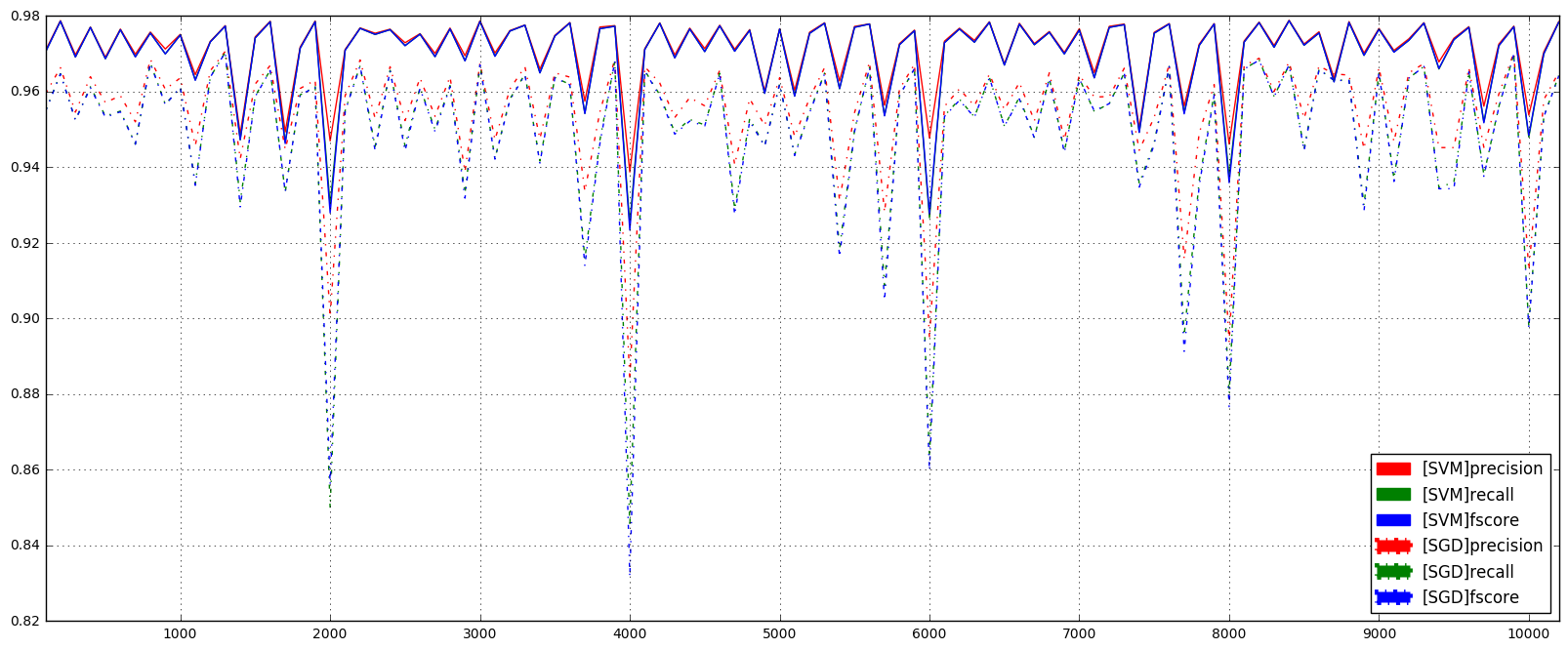
These results revealed that SGD and SVM are two best alternatives for this task. Final evaluation will be done based on the robustness of these models against missing data, and their classification performance with respect to the datasize.

# Results

## Model Evaluation and Validation

Below is the classification performance of the SVM and SGD when the dataset is being collected and trained at the same time. Dataset size starts from 100 and grows slowly and reaches up to 10000. SVM performs better than SGD all the time.

**Fig 7. SVM and SGD 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. 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 available data.

**TBA**

# 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.