Human Activity Recognition

Introduction:

Human Activity Recognition, or HAR for short, is the problem of predicting what a person is doing based on a trace of their movement using sensors. Human-centered computing is an emerging research field that aims to understand human behavior and integrate users and their social context with computer systems [1]. One of the most recent, challenging and appealing applications in this framework consists in sensing human body motion using smartphones to gather context information about people actions. Sensors are often located on the subject, such as a smartphone or vest, and often record accelerometer data in three dimensions (x, y, z). Although I have tons of collected data from sensors, there is no clear way to analyze the relationship between sensor data to specific actions in a general way. Sensor-based activity recognition seeks the profound high-level knowledge about human activities from multitudes of low-level sensor readings. Conventional pattern recognition approaches have made tremendous progress in the past years. However, those methods often mostly depend on huge hand-crafted feature extraction, which could hinder their generalization performance. Using exploratory analysis and multinomial logistic regression, I build up a predictive model that can classify the human activities using sensor data [2].

Data Collection and Preparation:

The experiments were carried out with a group of 30 volunteers within an age bracket of 19-48 years while wearing a smartphone(Samsung Galaxy) on the waist. Each person performed six activities(WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING, LAYING) wearing a smartphone(Samsung Galaxy) on the waist [3]. The dataset is collected from 30 volunteers(referred as subjects in this dataset), performing different activities. From the experiments, the data was recorded with the help of sensors (accelerometer and Gyroscope) in that smartphone. The total number of records is 10299 and the number of features is 563. The obtained dataset has been randomly partitioned into two sets, where 79.5%(Subject 1-22) of the volunteers was selected for generating the training data and 29.5%(Subject 23-30) the test data.

Exploratory Analysis:

Exploratory analysis was performed by examining tables and plots of the observed data. To check the data imbalance, I plot the number of activities by each subject for training and test data.

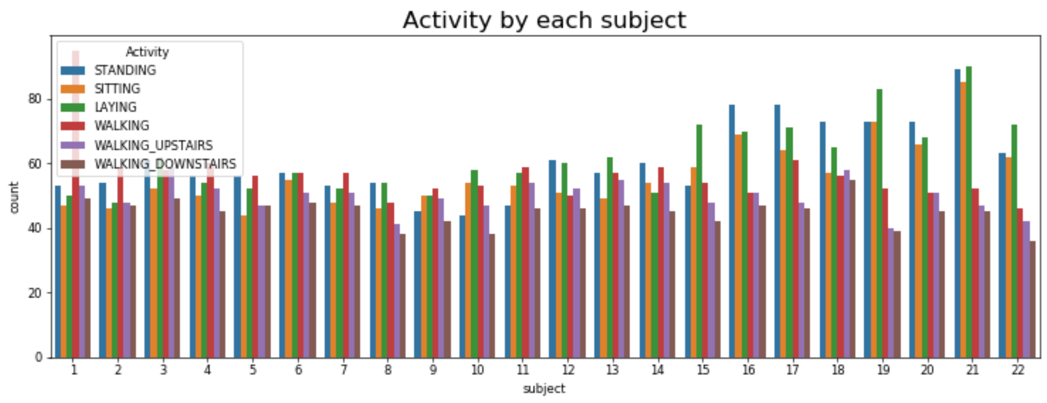


Figure 1. Activity by each subject from training data

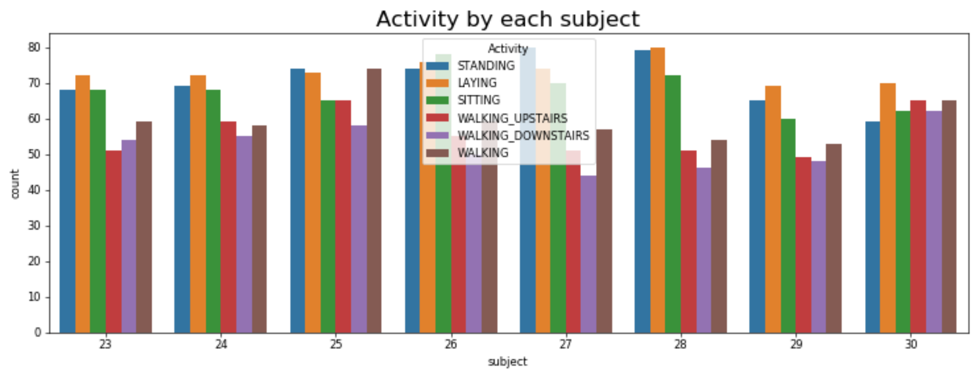


Figure 2. Activity by each subject from test data

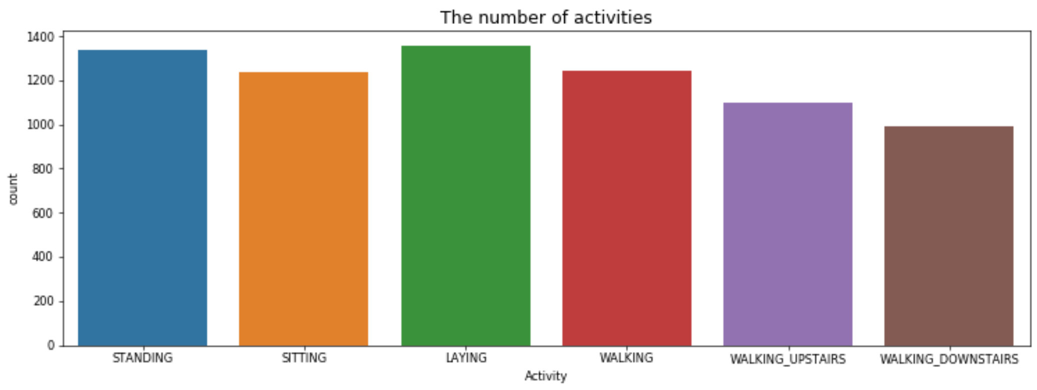


Figure 3. The number of activities

Based on figure 1,2 and 3, activity by each subject and the number of activities is almost well balanced.

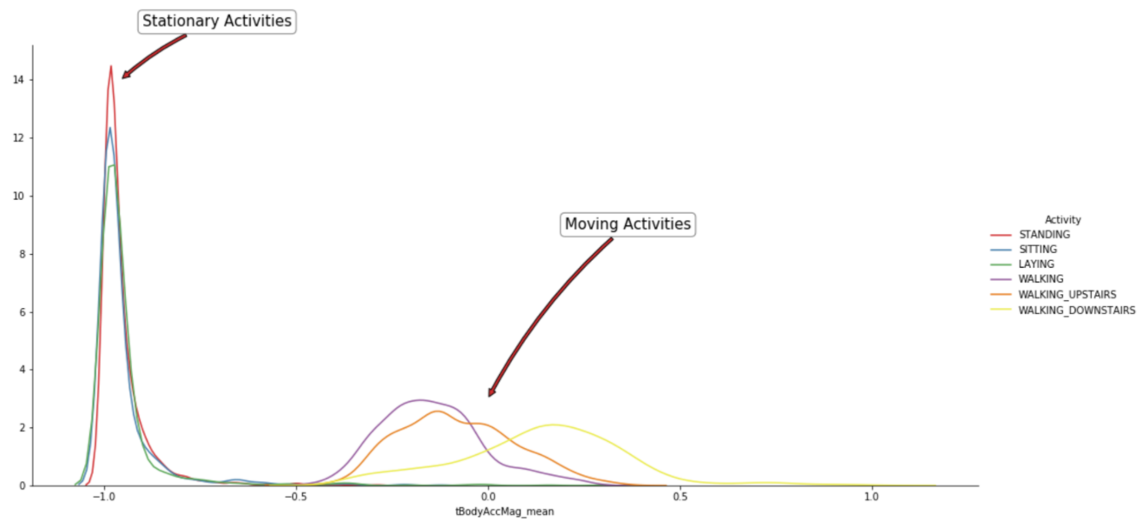


Figure 4. The relationship between stationary and moving activities based on tBodyAccMag-mean

Based on figure 4, it seems that stationary activities(Standing, sitting and laying) are less significant than moving activities(Walking, walking_upstairs and walking_downstairs).

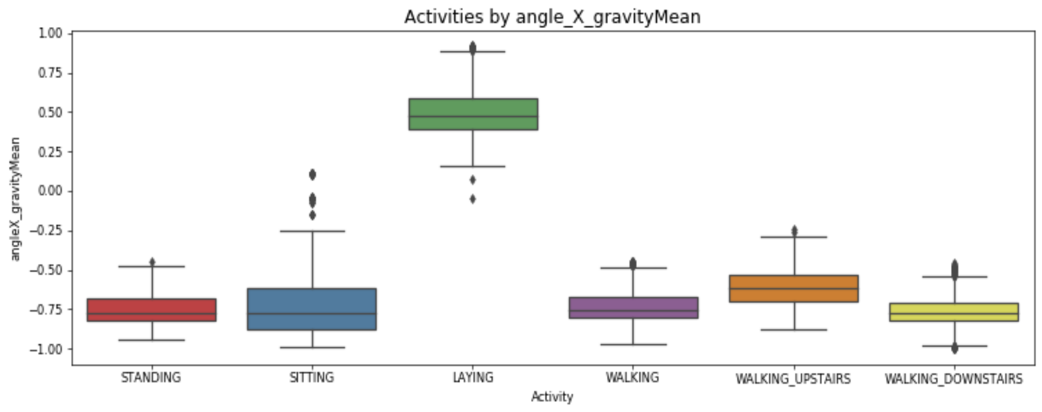


Figure 5. Activities by angle_X_gravityMean

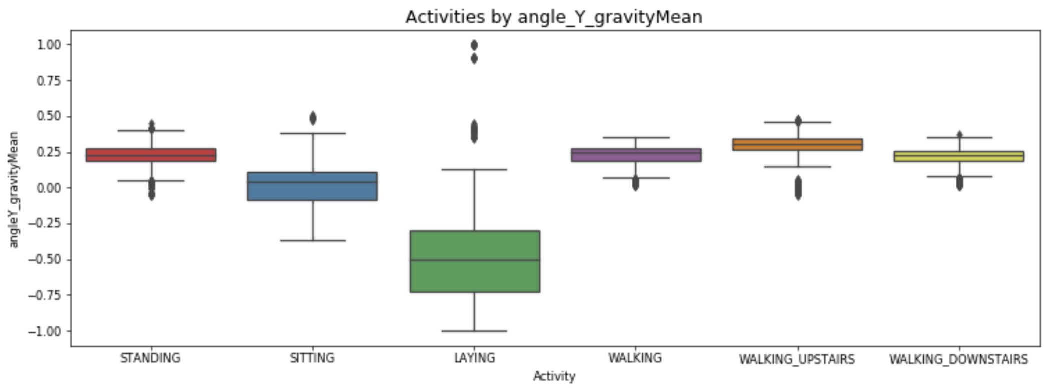


Figure 6. Activities by angle_Y_gravityMean

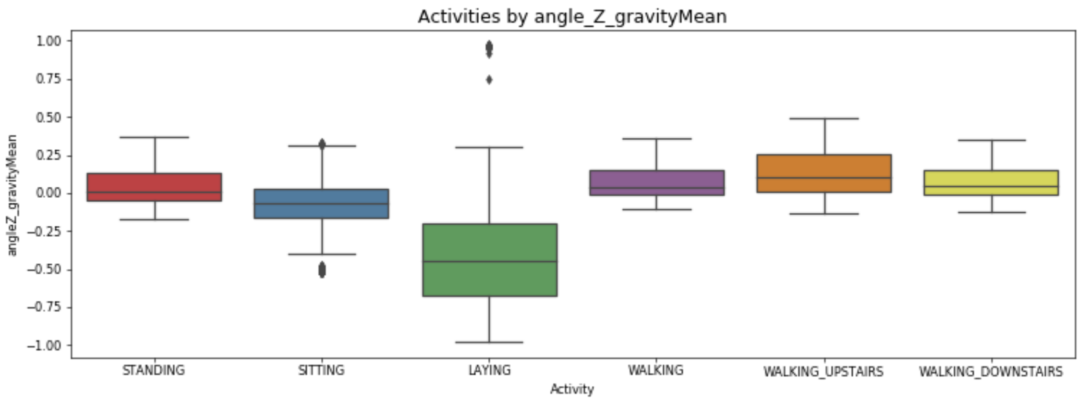


Figure 7. Activities by angle_Z_gravityMean

There is almost no difference among most of the activities, regarding the gravity of angle x,y and z. However only “Laying” is different from the others in three figures.

Methods:

XGBoost is a decision tree-based ensemble machine learning algorithm that uses a gradient boosting framework. It works similar with random forest method but XGBoost uses optimized gradient boosting algorithm through parallel processing, tree-pruning, handling missing values and regularization to avoid overfitting and bias.

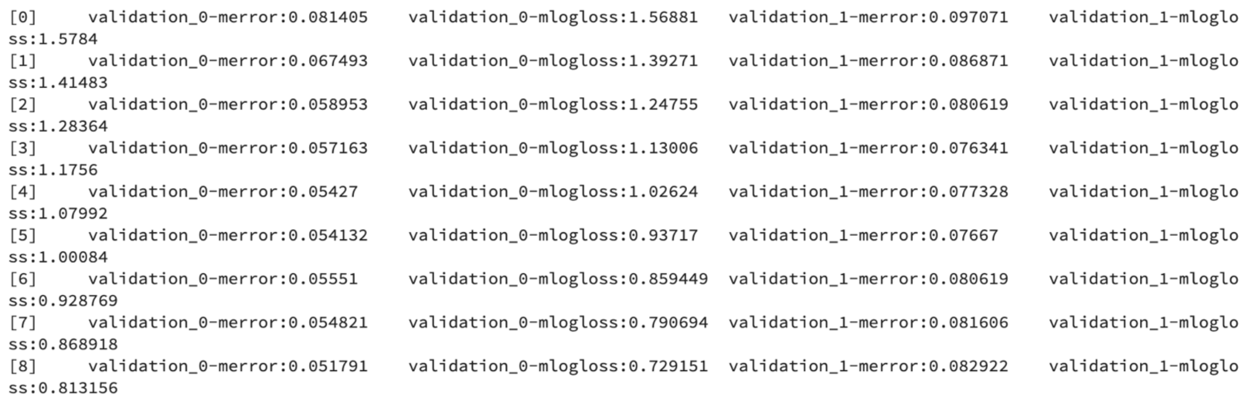
To predict multi-class variable, i used a parameter softprob as objective. Softprob gives the output a vector of ndata * nclass, which can be further reshaped to ndata * nclass matrix. The result contains predicted probability of each data point belonging to each class. In our case, XGBclassifier using softprob gives predicted probability of each data point belonging to each class(Sitting, laying, standing, walking, walking_downstairs, walking_upstairs) and each one of the class is encoded as 0 to 5.

In terms of the feature selection, first I varied the number of features used in XGBlassifier(from 1 to 560 increment by 30). Second, I applied PCA to reduce feature set. The difference between first and second is that PCA extract the important feature from a multivariate data table and to express this feature as a set of few new variables called principal components. These new variables correspond to a linear combination of the originals. The number of principal components is less than or equal to the number of original features. As a result, PCA reduces the dimensionality of a multivariate data to two or three principal components, that can be visualized graphically, with minimal loss of information [6].

Results:

In this project, I used XGBclassifier for the predictive model. When I ran the model with 561 features, the accuracy was 94%. With this, I examined to stop model training before the model overfits the training data using evaluation dataset with X_train,y_train, X_test and y_test. Through this, I can monitor the performance of our model that is being trained on a separate test dataset and stopping the training procedure once the performance on the test dataset has not improved after a fixed number of training iterations [7]. This performance can be measured by the loss function that takes into account the uncertainty of your prediction based on how much it varies from the actual label. It gives us a more nuanced view into the performance of our model.

Using eval_metric that has an array of X and y pairs, I can retrieve the performance of the model on the evaluation dataset and plot it to get insight into how learning unfolded while training.



```
[0]	validation_0-merror:0.081405	validation_0-mlogloss:1.56881	validation_1-merror:0.097071	validation_1-mloglo
ss:1.5784
[1]	validation_0-merror:0.067493	validation_0-mlogloss:1.39271	validation_1-merror:0.086871	validation_1-mloglo
ss:1.41483
[2]	validation_0-merror:0.058953	validation_0-mlogloss:1.24755	validation_1-merror:0.080619	validation_1-mloglo
ss:1.28364
[3]	validation_0-merror:0.057163	validation_0-mlogloss:1.13006	validation_1-merror:0.076341	validation_1-mloglo
ss:1.1756
[4]	validation_0-merror:0.05427	validation_0-mlogloss:1.02624	validation_1-merror:0.077328	validation_1-mloglo
ss:1.07992
[5]	validation_0-merror:0.054132	validation_0-mlogloss:0.93717	validation_1-merror:0.07667	validation_1-mloglo
ss:1.00084
[6]	validation_0-merror:0.05551	validation_0-mlogloss:0.859449	validation_1-merror:0.080619	validation_1-mloglo
ss:0.928769
[7]	validation_0-merror:0.054821	validation_0-mlogloss:0.790694	validation_1-merror:0.081606	validation_1-mloglo
ss:0.868918
[8]	validation_0-merror:0.051791	validation_0-mlogloss:0.729151	validation_1-merror:0.082922	validation_1-mloglo
ss:0.813156
```

Figure 8. Log loss and classification error for training and test data

Based on figure 8, our model reports log loss on both the train and test datasets each epoch. With this, I created two plots for log loss and classification error.

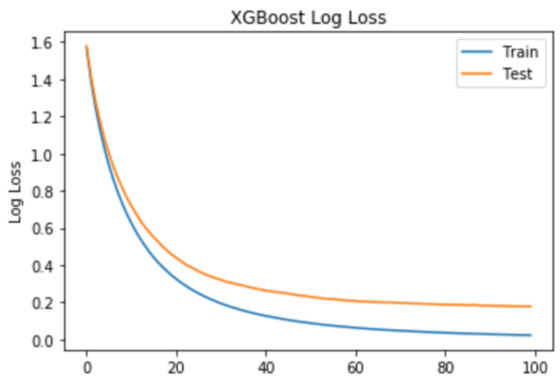


Figure 9. Log loss learning curve

In the log loss graph, as the epoch approaches 100, log loss slowly decreases. So, I do not need to stop the learning early. I can see that the model stopped training at epoch 100 and that the model with the best loss was observed at epoch 100 in log loss. Based on this log loss learning curve, it is unlikely that it might have overfitting during the training our dataset.

Using feature importance in XGBclassifier, I can see that importance feature that indicates how useful or valuable each feature was in the construction of the boosted decision trees within the model. The more an attribute is used to make key decisions with constructing decision tree, the higher its relative importance.

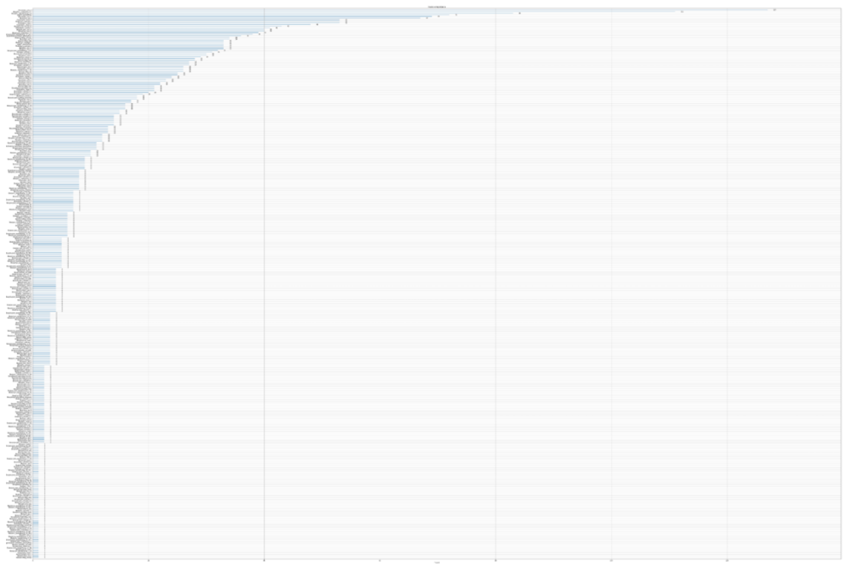
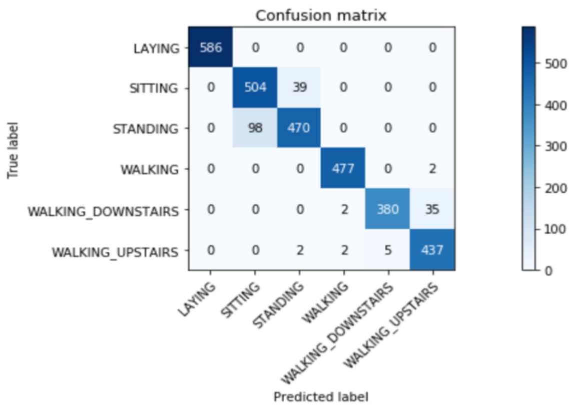


Figure 10. Feature Importance in XGBclassifier

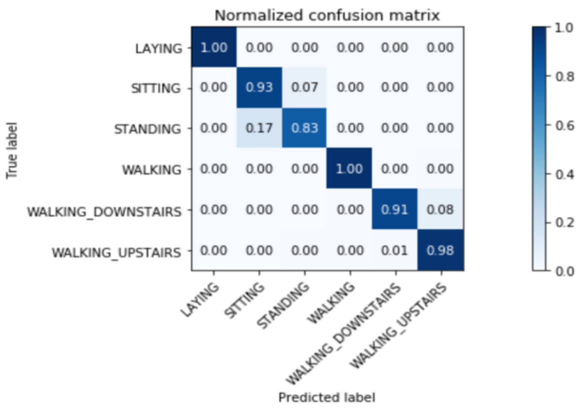
Figure 10 shows the order of 561 features ranked by the score indicating the extent to which each individual feature is valuable for the construction of the boosted decision tree. The best 5 important features are tGravityAcc-min()-X, tBodyAcc-correlation()-X,Y, fBodyAcc-bandsEnergy()-1,8, angle(X,gravityMean) and tGravityAcc-min()-Y. As a result, these 5 features are more important than other features to establish the decision tree using XGBclassifier.

In terms of confusion matrix, I created normalized and not normalized one.



True label \ Predicted label	LAYING	SITTING	STANDING	WALKING	WALKING_DOWNSTAIRS	WALKING_UPSTAIRS
LAYING	586	0	0	0	0	0
SITTING	0	504	39	0	0	0
STANDING	0	98	470	0	0	0
WALKING	0	0	0	477	0	2
WALKING_DOWNSTAIRS	0	0	0	2	380	35
WALKING_UPSTAIRS	0	0	2	2	5	437

Figure 11. Confusion matrix for XGB classifier



True label \ Predicted label	LAYING	SITTING	STANDING	WALKING	WALKING_DOWNSTAIRS	WALKING_UPSTAIRS
LAYING	1.00	0.00	0.00	0.00	0.00	0.00
SITTING	0.00	0.93	0.07	0.00	0.00	0.00
STANDING	0.00	0.17	0.83	0.00	0.00	0.00
WALKING	0.00	0.00	0.00	1.00	0.00	0.00
WALKING_DOWNSTAIRS	0.00	0.00	0.00	0.00	0.91	0.08
WALKING_UPSTAIRS	0.00	0.00	0.00	0.00	0.01	0.98

Figure 12. Normalized Confusion matrix for XGB classifier

The number of misclassification “sitting” is 98 given that the true label was “standing”. And the number of misclassification “standing” is 39 given that the true label was “sitting”. With this, the number of misclassification “walking_upstairs” is 2 given that the true label was “walking”. The number of mis classification “walking” is 2 given that the true label was “walking_downstairs”. The number of misclassification “walking_upstairs” is 35 given that the true label was “walking_downstairs”. Also, the number of misclassification “standing” is 2 given that the true label was “walking_upstairs”. The number of misclassification “walking” is 2 given that the true label was “walking_upstairs”. The number of misclassification “walking_downstairs” given that the true label was “walking_upstairs”. Based on the confusion matrix, I know that misclassification in terms of moving activities is more likely to occur among moving activities(walking, walking_upstairs and walking_downstairs). With this, the misclassification in terms of stationary activities is more likely to occur among stationary activities(laying, sitting and standing).

In terms of the number of features, I varied the number of features from 1 to 560 increment by 30. Basically, as the number of features increase the accuracy also increases. Although accuracy was a little bit fluctuated between 220 to 240 features, it was enhanced. With this, when I used 16 features, the accuracy was about 80%. The accuracy was about 90% with 96 features.

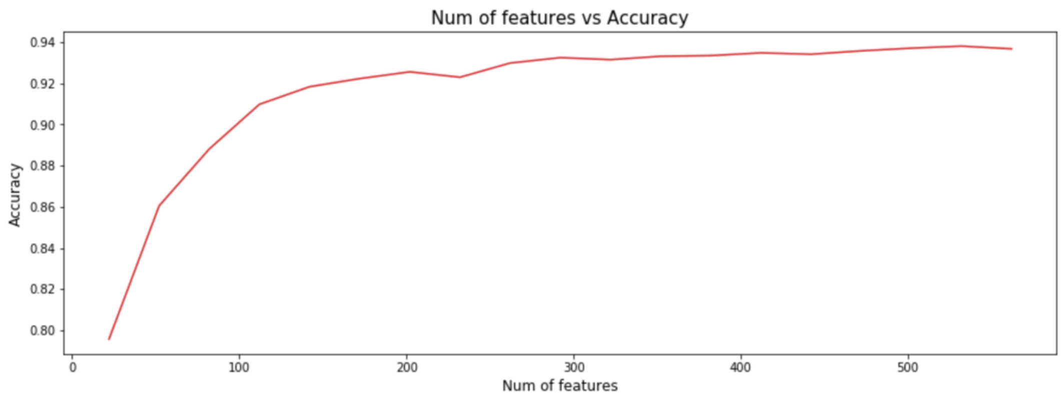


Figure 13. Number of features vs Accuracy of XGBclaassfier

For dimensionality reduction, I used Principal Component Analysis(PCA). PCA enables us to create and use a reduced set of variables which are called principal component. PCA aims at reducing a large set of variables to a small set that still contains most of the information in the large set [8].

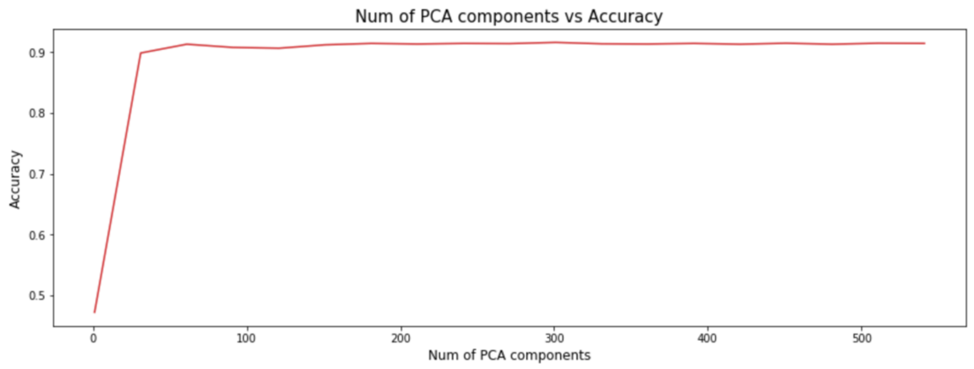


Figure 14. Number of PCA components vs Accuracy of XGBclassifier

Although accuracy was a little bit fluctuated, it increased. It seems that the accuracy leveled off after about 30 components. Minimum number of PCA components required to obtain about 90% accuracy is 31. Finally, this analysis shows that how the number of features affect the accuracy of the predictive model. Furthermore, we see that the difference between the number of features and the number of PCA components in terms of the accuracy. In this project, while I didn't show the measure of uncertainty, we can see how XGBclassifier works well with our data using log loss learning curve. Although I did not compare the accuracy of XGBclassifier with the accuracy of other predictive models, overall XGBclassifier performed well with our dataset.

References

1. How to Model Human Activity from Smartphone Data. URL: https://machinelearningmastery.com/how-to-model-human-activity-from-smartphone-data/

2. A public domain dataset for human activity recognition using smartphones. URL: https://upcommons.upc.edu/handle/2117/20897

3. Human Activity Recognition Using Smartphones Data Set. URL: https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones

4. PCA-Principal Components Analysis Essentials. URL: http://www.sthda.com/english/articles/31-principal-component-methods-in-r-practical-guide/112-pca-principal-component-analysis-essentials/

5. Avoid Overfitting by Early Stopping with XGBoost in Python. URL: https://machinelearningmastery.com/avoid-overfitting-by-early-stopping-with-xgboost-in-python/

6. Principal Components. URL: https://www.itl.nist.gov/div898/handbook/pmc/section5/pmc55.htm