Human Activity Recognition

Nageswara Rao Gurram

September 24, 2020

1 Introduction

These days we can see many fitness tracking devices in the market that make use of sensors embedded in them to determine not only the person's posture such as sitting, standing and laying down etc.. but also motion like walking, running, hiking etc.. Due to this project, I tried to predict the person's motion and posture from the smartphone device measurements. The data is obtained from the standard UCI machine learning repository. For more details on how this data is captured and prepared, one can refer this link [1].

As the hard part of the any data science project i.e cleaning and pre-processing is carried out already in the original dataset, analysis and modeling is carried out as part of the project. Although, we could add more features from the raw data that's given in the repository [1], I decided its not necessary as the accuracy is already reached to 99% with the given features itself.

1.1 Project Objective

Problem type The data consists in format of features and corresponding labels, so it can be treated as supervised learning. The objective is to predict the correct motion posture from the given sensor (processed) data. The variables in the the data correspond to different statistics in the time and frequency domain of the sensor data.

Analysis Before diving into the modeling part, it makes sense to do some analysis on the data. It helps to decide on what kind of data we're dealing with. In the Section. 2, I've done some analysis to know the balance and visualizations for linearity checks.

Models In the next section, I've applied couple of ensemble models that works best on classification problems. The results showed both models predicted with 98% accuracy on the test set.

Feature selection and Dimensionality reduction Later, some more analysis is done to see the important features and the relevance of them in predicting the labels. Its observed, we only need not more than 10 features to get 90% accuracy. Likewise, I've performed the PCA to remove the redundant information and showed that the top-10 eigen vectors can explain enough variance in the data to get 90% accuracy.

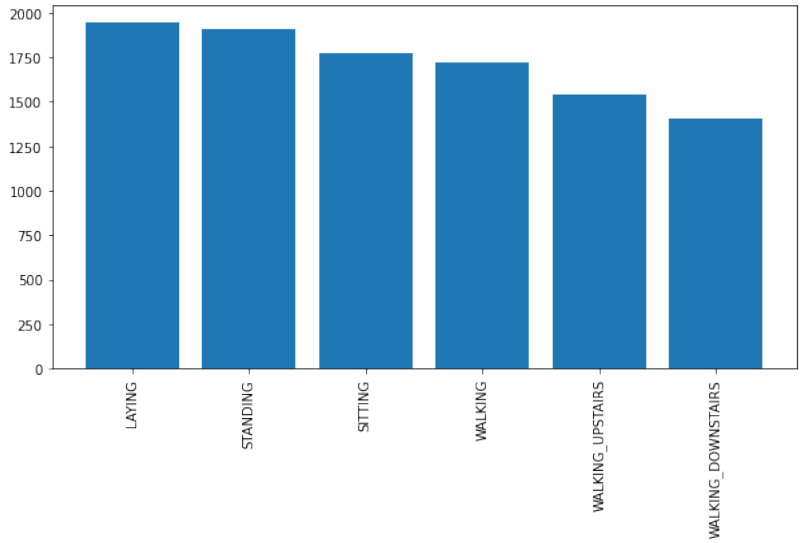

Figure 1: label distribution in the data set

2 Data Analysis

As a first step in understanding the data, we do exploratory analysis on the distribution of features and labels. Any model built on the skewed data can go horribly wrong on real world data as model is trained to get better accuracy on the frequently appearing labels. For this reason, one needs to visualize the distribution of the labels in the data set. In this problem also, I've plotted the label distribution as an histogram in Fig. 1.

Although there is little difference in the frequencies of the labels, by looking at the overall data set we can say that its not skewed and all the labels have enough data.

As the next step, to understand more about the data distribution visual plots can be handy. Many times the data can be in hundreds to thousands of dimensions size, so visualizing might be a very difficult task. For this reason, generally we reduce the dimensionality into 2 or 3 by using techniques such PCA or t-SNE. In this project too, I've visualized the data using these techniques.

2.1 PCA

Principal Component Analysis (PCA) is one of the widely used techniques to reduce the features dimensions. PCA tries to find the new basis for the feature set where it tries to keep variance maximum. There are many other interpretations for the PCA such as it can also be interpreted as minimization of projection errors onto the new basis. After reducing the dimensions of the original features set size from 561 to 2 using PCA, it can be visualized as in Fig. 2.

From the Fig.2, we can see that all the points when a person is walking is oriented to one side and for sitting, laying and standing (motion less) postures data is to the other side. This is true because of accelerometer calibration shows different from when person is moving to staying idle.

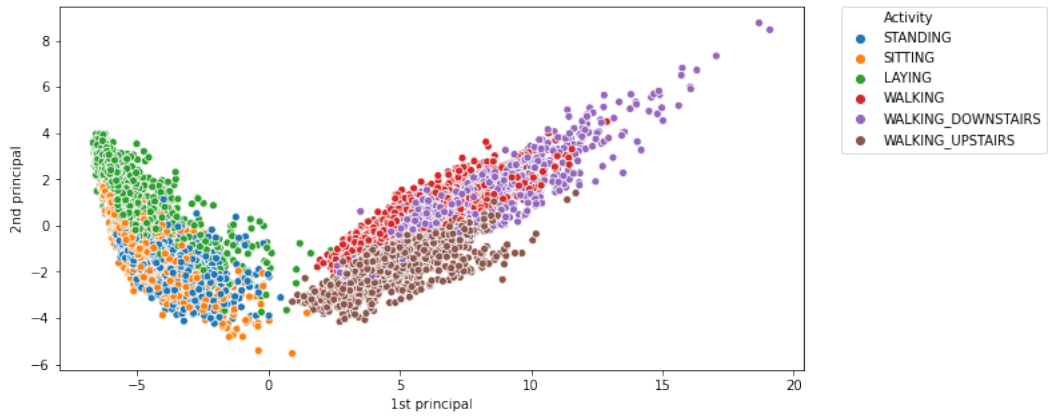

Figure 2: Visualization of the data in 2D using PCA

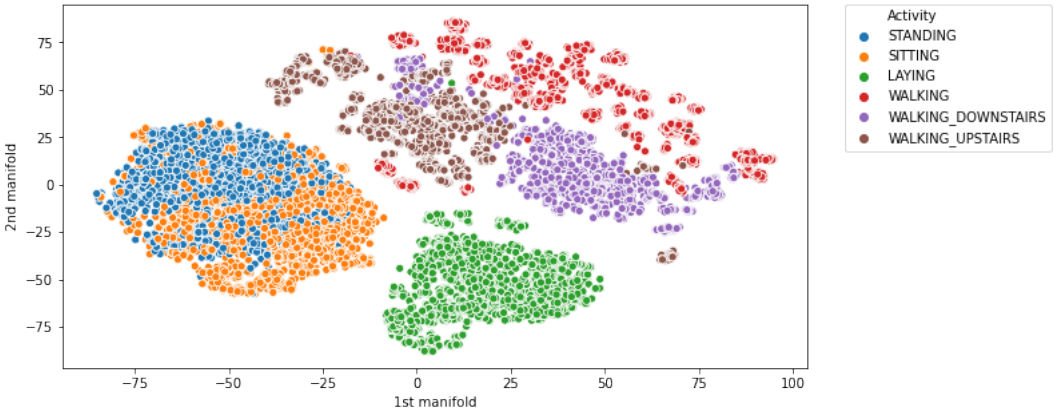

Figure 3: Visualization of the data in 2D using t-SNE

From this we can say that accelerometer plays an important role in determining the posture of the person.

2.2 t-SNE

t-Stochastic Neighborhood Embedding is another dimensionality reduction which tries to learn non linear manifolds in the data where PCA fails to do. It converts similarities between data points to joint probabilities and tries to minimize the Kullback-Leibler divergence between the joint probabilities of the low-dimensional embedding and the high-dimensional data.

Specifically, it models each high-dimensional object by a two- or three-dimensional point in such a way that similar objects are modeled by nearby points and dissimilar objects are modeled by distant points with high probability.

Note: As the t-SNE on whole dataset may run very slow, I've reduced the feature dimension first by running PCA and then did t-SNE on sub sample of the original data.

From the Fig. 3, we can observe t-SNE is able to learn better manifolds that separates the data more clearly than PCA.

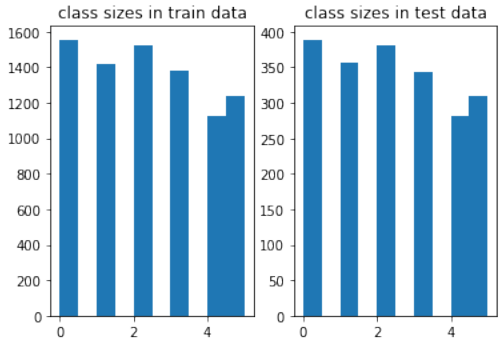

Figure 4: Stratified Sampling

3 Data Preprocessing

For any kind of dataset, preprocessing step arises before submitting to model. Although the dataset is already cleaned and curated perfectly here, we need to do some steps that are required to train any model.

- **Encoding**: There are six labels in the data, each of which specifies the either posture or motion state of the person. Any model expects a labels to be encoded while supplying to it. So, the labels are encoded as following:
  - 'LAYING', - 0
  - 'SITTING', - 1
  - 'STANDING' - 2
  - 'WALKING' - 3
  - 'WALKING_DOWNSTAIRS' - 4
  - 'WALKING_UPSTAIRS' - 5
- **Train/Test Split:** As a general rule, the data set is divided in 0.8:0.2 split for train and test the model respectively. Few things to remember while splitting the data in multi-class classification problem setting are:
  - Shuffle: As most of the models work internally stochastically, we need to shuffle the data before splitting.
  - Stratified Sampling: When there are multiple classes in the data set, splitting into train and test datasets should receive equal proportions of all the classes present which is also called 'Stratified Sampling'.

4 Methods

After analysing the data, we can say that predictive models that can learn non-linear decision boundaries can classify well. As part of this project, I've tried two ensemble models for

	precision	recall	f1-score	support
LAYING	1.00	1.00	1.00	389
SITTING	0.98	1.00	0.99	356
STANDING	1.00	0.98	0.99	381
WALKING	1.00	0.99	0.99	344
WALKING_DOWNSTAIRS	0.99	1.00	0.99	281
WALKING_UPSTAIRS	0.99	1.00	1.00	309
accuracy			0.99	2060
macro avg	0.99	0.99	0.99	2060
weighted avg	0.99	0.99	0.99	2060

Figure 5: Gradient Boosting model classification report

classification on this dataset. Ensemble models consist of multiple weak predictors to predict a single hypothesis as a whole. Empirically when these models are very diverse they tend to reduce the over-fitting and so the variance in the predictions. In this project, I've applied two common methods in ensemble learning i.e bagging and boosting.

4.1 Gradient Boosting

Boosting involves incrementally building an ensemble by training each new model instance to emphasize the training instances that previous models mis-classified. In Boosting, an equal weight (uniform probability distribution) is given to the sample training data (say D1) at the very starting round. This data (D1) is then given to a base learner (say L1). The mis-classified instances by L1 are assigned a weight higher than the correctly classified instances, but keeping in mind that the total probability distribution will be equal to 1. This boosted data (say D2) is then given to second base learner (say L2) and so on. The results are then combined in the form of voting. [2]

Gradient boosting is one kind of boosting technique that builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage a number of regression trees are fit on the negative gradient of the binomial or multinomial deviance loss function. Here I've used 'HistGradientBoostingClassifier' which is based on LightGBM. [3]

Results The results are almost perfect accurate on the test dataset as shown in the Fig. 6 and Fig. 5. The gradient boosting model achieved **99.3%** accuracy on the test dataset. Due to the near perfect result, I didn't explore on hyper-parameter tuning and picking the best model by using validation data sets kind of strategies.

4.2 Random Forest

In the second experiment, I've used Random Forest which is another popular ensemble technique. Random forests are one kind of bagging techniques, where the weak learners are early pruned decision trees (plays the role of weak learners) that are trained on sub-sampled data. The randomness is in the model is by taking a stochastic sub-sample of the data set

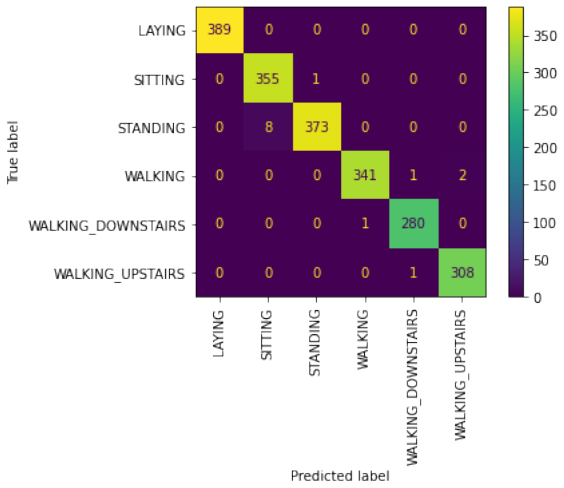

Figure 6: Gradient Boosting model confusion matrix

	precision	recall	f1-score	support
LAYING	1.00	1.00	1.00	389
SITTING	0.97	0.97	0.97	356
STANDING	0.97	0.97	0.97	381
WALKING	1.00	0.98	0.99	344
WALKING_DOWNSTAIRS	0.98	0.98	0.98	281
WALKING_UPSTAIRS	0.97	0.99	0.98	309
accuracy			0.98	2060
macro avg	0.98	0.98	0.98	2060
weighted avg	0.98	0.98	0.98	2060

Figure 7: Random forest model classification report

and also the feature selection at decision tree build time. At the end, the label is chosen based on majority agreement of the trees.

Results Like Gradient boosting, Random forest also predicted to near perfect accuracy score on the test data set as shown in Fig. and Fig. This model's average accuracy on the test data set is **98.15%**.

Although both models resulted near perfect accuracy scores, random forests made some errors between the classes 'SITTING' and 'STANDING'. This can be due to the proximity between the features of corresponding class labels. Random forests tries to make decision boundaries by making piece-wise linear boundaries between different class labels. This may fail sometimes when the two different classes data points are overlapped and linear decision boundaries fail to separate them.

5 Feature Selection

Tree based models like Random forests can also give the the feature importance scores while training the model. These feature importance metrics are calculated based on the impurity

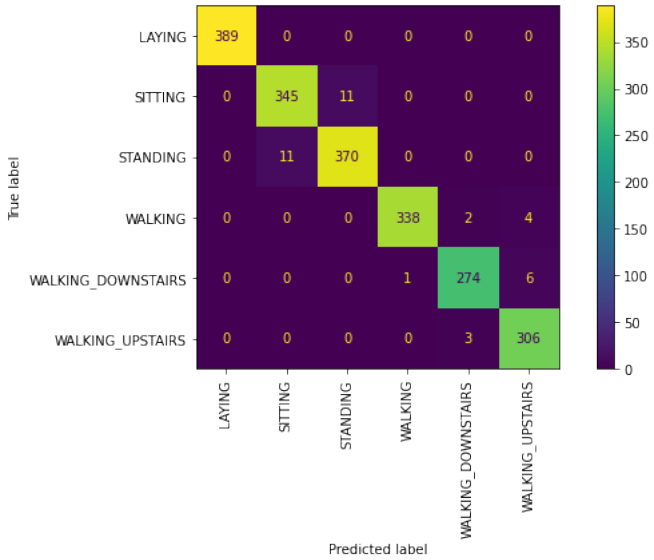

Figure 8: Random forest model confusion matrix

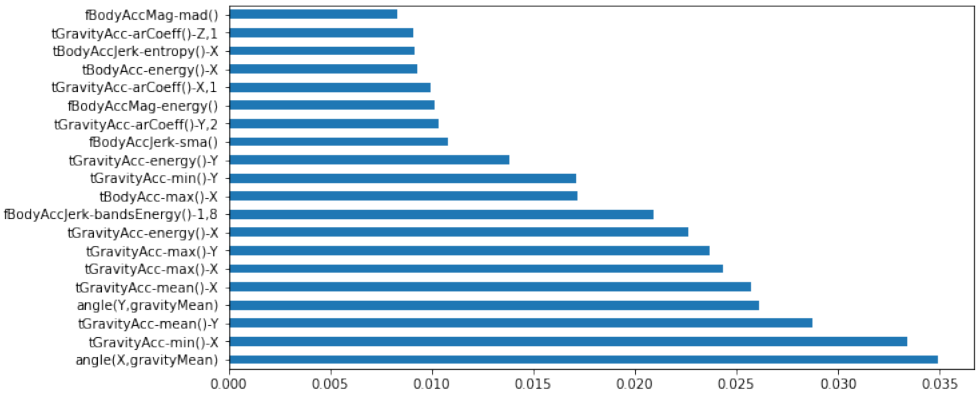

Figure 9: Feature importance scores in Random forests

criterion (either entropy or gini index) reduction when that corresponding feature is chosen to spilt the data set at a node. The feature importance scores in this dataset are shown in Fig. 9.

From the plot, we can say features that correspond to 'body acceleration' and 'gravity acceleration' are important contributors in the predictions.

In the next step, I've analyzed the minimum number of features needed to get atleast 80% and 90% accuracy. For this, we can take the top features from above plot and train the model with a spliced data (spliced with only top features). Finally, I plotted the accuracy vs the number of features needed in Fig. 10. From the plot, we can see that only 2 features are needed for getting 80% accuracy and 9 features are required out of 561 to get 90% accuracy.

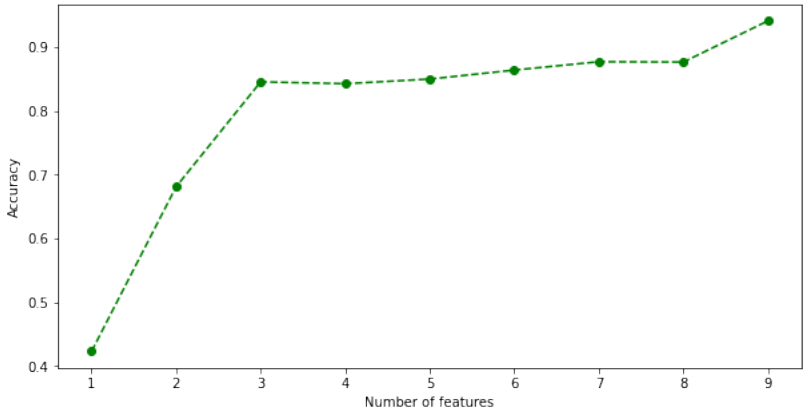

Figure 10: Accuracy vs Number of features

6 Dimensionality Reduction

The core difference between 'Feature Selection' and 'Dimensionality Reduction' is the ability to change feature space into another basis. In the above strategy, we are permitted to take subset of the given features to train the model but not in anyway alter them. However, when we reduce the feature dimensions with techniques like PCA, the meaning of the original features is lost. The new features represent either some linear or non-linear combination of the original features and interpretability is only abstract.

In the data analysis section, the PCA is already applied on the original dataset for visualizing in 2D plane. Here, I've used the same algorithm to reduce the input dimensions to get the desired accuracy. For 'Dimensionality Reduction', we apply PCA on the original dataset and take the top-k principal components which explains the maximum variance in the data. Later, each data point is projected on to this new axis (basis) to get the new (reduced size) data point.

The transformation $\mathbf{T} = \mathbf{XW}$ maps a data vector x_i from an original space of p variables to a new space of p variables which are uncorrelated over the dataset. However, not all the principal components need to be kept. Keeping only the first L principal components, produced by using only the first L eigenvectors, gives the truncated transformation

$\mathbf{T_L} = \mathbf{XW_L}$

where the matrix $\mathbf{T_L}$ now has n rows but only L columns. In other words, PCA learns a linear transformation $t = \mathbf{W}^T x, x \in R^p, t \in R^L$, where the columns of $p * L$ matrix $\mathbf{W}$ form an orthogonal basis for the L features that are decorrelated. By construction, of all the transformed data matrices with only L columns, this score matrix maximises the variance in the original data that has been preserved, while minimising the total squared reconstruction error $||\mathbf{X} - \mathbf{X_L}||_\mathbf{2}^\mathbf{2}$. [4]

The plot Fig.11 shows the accuracy vs the reduced dimensions to get that on Random forest classifier. It can be observed that like above only top 10 principal components are enough to get the accuracy of 90%.

Note: Although, we only needed top-10 features or principal components to predict 90%, it doesn't mean that those two sets represents the same data. Principal components are the top eigen vectors (based on eigenvalues) that explain most of the variance where as top-k

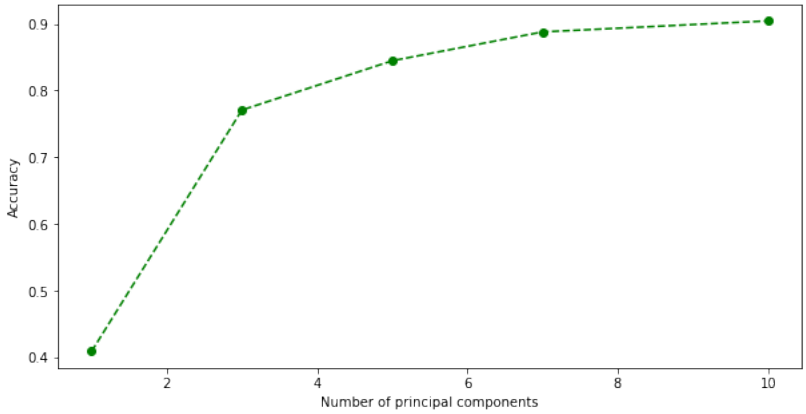

Figure 11: Accuracy vs Number of principal components

features are the subset of the original set.

References

[1] *Human Activity Recognition*, `https://archive.ics.uci.edu/ml/datasets/human+activity+recog`

[2] *Boosting*,
`https://en.wikipedia.org/wiki/Ensemble_learning`.

[3] *Light BGM*
`https://github.com/Microsoft/LightGBM`

[4] *Dimensionality reduction with PCA*
`https://en.wikipedia.org/wiki/Principal_component_analysis#Dimensionality_reduction`