MadhavShashi / **Human-Activity-Recognition-Using-Smartphones-Sensor-DataSet** Public

Human activity recognition, is a challenging time series classification task. It involves predicting the movement of a person based on sensor data and traditionally involves deep domain expertise and methods from signal processing to correctly engineer features from the raw data in order to fit a machine learning model.

71 stars 23 forks Branches Tags Activity

Star Notifications

Code Issues Pull requests Actions Projects Security Insights

master 1 Branch 0 Tags Go to file Go to file Code

MadhavShashi HAR README.md 959a0df · 4 years ago

.gitignore	Initial commit	4 years ago
1.HumanActivityRecognition_EDA.ipynb	Project Source Code: First Part	4 years ago
2. Applying Classical ML PREDICTION M...	Project Source Code: Second Part	4 years ago
3. Applying LSTM.ipynb	Project Source Code: Third Part	4 years ago
4. Conclusion.ipynb	Project Source Code: Fourth Part	4 years ago
README.md	HAR README.md	4 years ago

Table of Content

Overview

Smart phones have become a most useful tool in our daily life for communication with advanced technology provided intelligent assistance to the user in their everyday activities. The portable working framework with computing ability and interconnectivity, application programming interfaces for executing outsiders' tools and applications, mobile phones have highlights such as cameras, GPS, web browsers so on., and implanted sensors such as **accelerometers** and **gyroscope** which permits the improvement of applications in view of client's specific area, movement and context.

Activity Recognition (AR) is monitoring the liveliness of a person by using smart phone. Smart phones are used in a wider manner and it becomes one of the ways to identify the human's environmental changes by using the sensors in smart mobiles. *Smart phones are equipped in detecting sensors like gyroscope and accelerometer*. The contraption is demonstrated to examine the state of an individual.

Human Activity Recognition (HAR) framework *collects the raw data from sensors and observes the human movement using different deep learning approach*. Deep learning models are proposed to identify motions of humans with plausible high accuracy by using sensed data.

HAR Dataset from UCI dataset storehouse is utilized. This dataset is collected from 30 persons (referred as subjects in this dataset), performing different activities with a smartphone to their waists. The data is recorded with the help of sensors (*accelerometer and Gyroscope*) in that smartphone. This experiment was video recorded to label the data manually.

This project is to build a model that *predicts the human activities* such as **Walking, Walking_Upstairs, Walking_Downstairs, Sitting, Standing** and **Laying**.

Sources/Useful Links

- **Blog 1** : https://www.ijrte.org/wp-content/uploads/papers/v8i1/A1385058119.pdf
- **Blog 2** : https://machinelearningmastery.com/how-to-model-human-activity-from-smartphone-data/
- **For HAR Data_Set** : https://archive.ics.uci.edu/ml/datasets/Smartphone+Dataset+for+Human+Activity+Recognition+%28HAR%29+in+Ambient+Assisted+Living+%28AAL%29

Problem Statement

Given a new datapoint we have to predict the Human Activity.

Solution

We have a fairly small data set of Human Activity Recognition that has been labeled as "**Walking**", "**Walking Upstairs**", "**Walking Downstairs**", "**Standing**", "**Sitting**" and "**Lying**". We had downloaded HAR Dataset from UCI dataset storehouse and we know that the data set is define in two part first is RAW data set and second is pre-engineered by domain or signal expert engineer. **So first**, we use pre-engineered dataset with classical machine learning (ML) to learn from the data, and predict the human Activity. **Second**, we could then use RAW dataset with Deep learning model to learn from the data, and predict the human Activity.

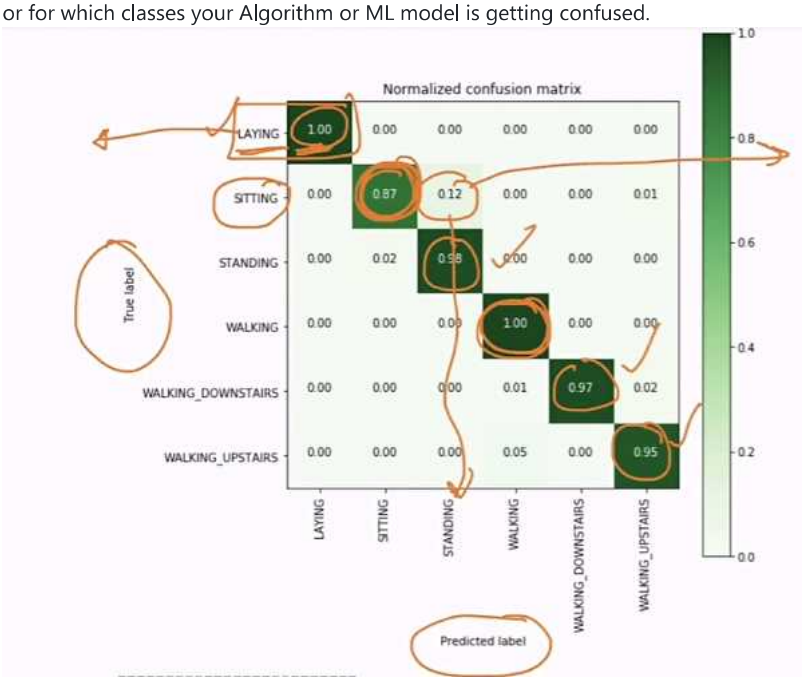
Which type of ML Problem is this?

Human activity recognition, **is a challenging time series classification task.** It involves predicting the movement of a person based on sensor data and traditionally involves deep domain expertise and methods from signal processing to correctly engineer features from the raw data in order to fit a machine learning model.

OR in other words you can call, it is a **multiclass classification problem**, for given a new datapoint we have to predict the Human Activity. And *Each datapoint corresponds one of the 6 Activities*.

What is the best performance metric for this Problem?

- **Accuracy** : For any model we have printed the over all accuracy with this simple "Accuracy" metric.
- **Confusion Matrix** : The very important thing that the confusion Matrix had told us what type of errors and what types of confusion are happening.
    - Simply for understanding this metric for this project view, we know that we have 6 class label and often times it could so happen that our Model will be confused between sitting or standing, and walking upstairs or walking downstairs.
    - So, the confusion Matrix is a very-very important way of understanding which class is your Algorithm or ML model is doing very well or for which classes your Algorithm or ML model is getting confused.



We can see clearly in this confusion matrix plot our model is doing very well for class Laying and Walking and good for Standing, Walking_Downstairs and Walking_Upstairs but our model getting confused with Sitting Class.

- **Multi-class log-loss**: We know that Multi-class log-loss is very important metric for multiclass ML problem.

Business Objectives and Constraints

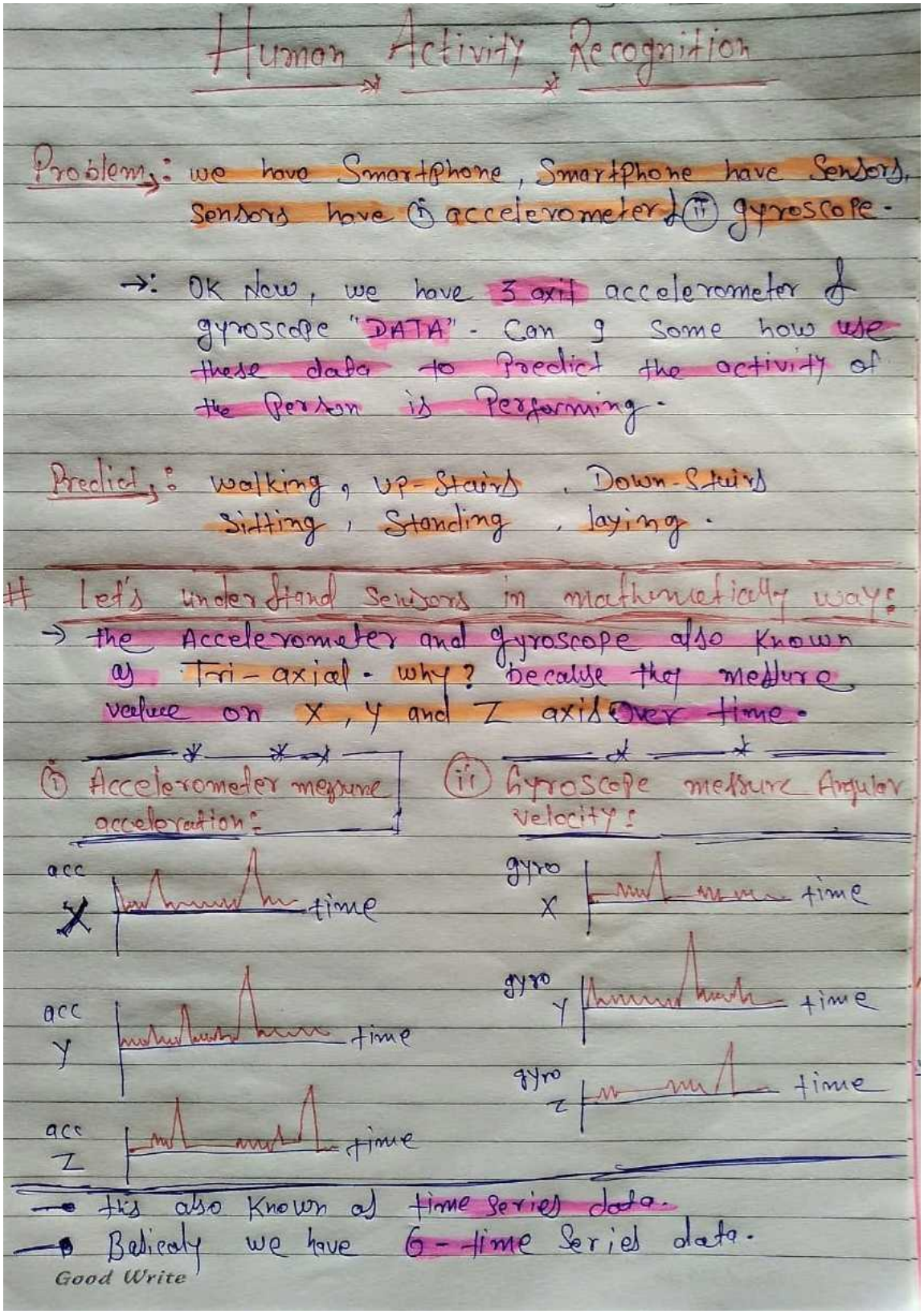
- These days, in addition to Smartphones, we are also using Smart-Watches like Fitbit or Apple-Watch, which help us to track our health. They monitor our each activity throughout the day check how many calories we have burnt. How many hours have we slept. However, in addition to Accelerometer and Gyroscope, they also use Heart-Rate data to monitor our activity. Since, we only have Smartphone data so just by using Accelerometer and Gyroscope data we will monitor the activity of a person. This software can then be converted into an App which can be downloaded in Smartphone. Hence, a person who has Smartphone can monitor his/her health using this App.
- **The cost of a mis-classification can be very high**.
- **No strict latency concerns**.

Data Overview

1. How data was recorded

- 30 participants (*referred as subjects in this dataset*) performed activities of daily living while *carrying a waist-mounted smartphone*. The phone was configured to record two implemented sensors (**accelerometer and gyroscope**). For these time series the directors of the underlying study performed feature generation and generated the dataset by moving a **fixed-width window of 2.56s** over the series. Since the **windows had 50% overlap** the resulting points are **equally spaced (1.28s)**.This experiment was video recorded to label the data manually.
- By using the sensors(**Gyroscope and accelerometer**) in a smartphone, they have captured **'3-axial linear acceleration'**(***tAcc-XYZ***) **from accelerometer and '3-axial angular velocity' (*tGyro-XYZ*) from Gyroscope** with several variations.
    - prefix '**t**' in those metrics denotes time.
    - suffix '**XYZ**' represents 3-axial signals in **X** , **Y**, and **Z** directions.

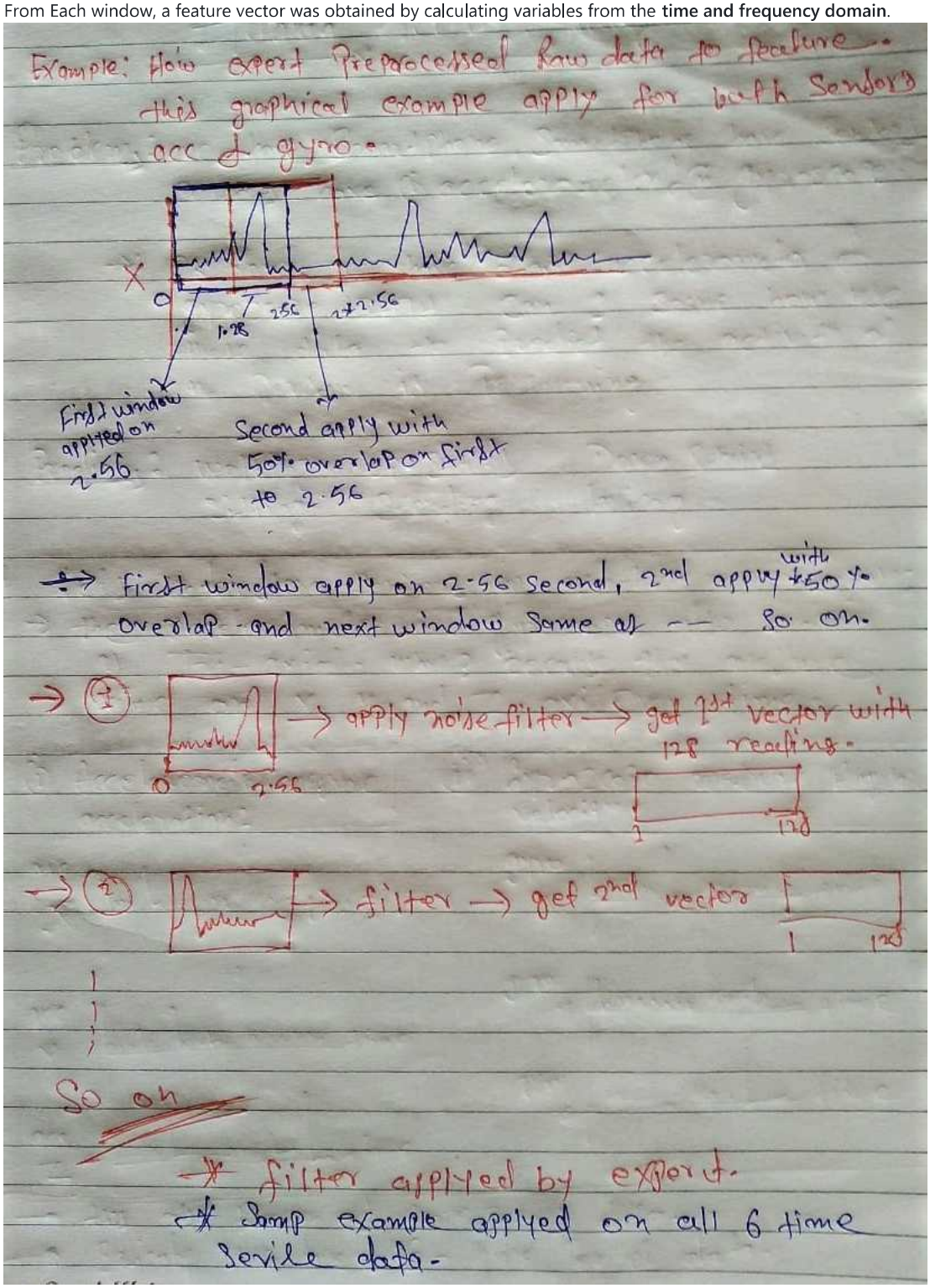
- Let's understand above information in graphical way below:



2. How is the Data preprocessed?

- After getting Raw Sensor Data the Expert (**Domain Expert, Signal Engineer Expert**) are preprocessed this data and make some useful feature. I am not expert but what I understand I explain here how these data are preprocessed.
- These sensor signals are preprocessed by applying **noise filters** and then *sampled in fixed-width windows (sliding windows) of 2.56 seconds each with 50% overlap*. ie., each window has *128* readings.

- From Each window, a feature vector was obtained by calculating variables from the **time and frequency domain**.



- The accelertion signal was saperated into Body and Gravity acceleration signals(**tBodyAcc-XYZ** and **tGravityAcc-XYZ**) using some low pass filter with corner frequecy of 0.3Hz.

- After that, the body linear acceleration and angular velocity were derived in time to obtian *jerk signals* (**tBodyAccJerk-XYZ** and **tBodyGyroJerk-XYZ**).

- The magnitude of these 3-dimensional signals were calculated using the Euclidian norm. This magnitudes are represented as features with names like **tBodyAccMag_**, ***tGravityAccMag***, ***tBodyAccJerkMag***, **_tBodyGyroMag and tBodyGyroJerkMag.**

- Finally, We've got frequency domain signals from some of the available signals by applying a FFT (**Fast Fourier Transform**). These signals obtained were labeled with **prefix 'f'** just like original signals with **prefix 't'**. These signals are labeled as **fBodyAcc-XYZ**, **fBodyGyroMag** etc.,.

- These are the signals that we got so far.

  - tBodyAcc-XYZ
  - tGravityAcc-XYZ

- tBodyAccJerk-XYZ
- tBodyGyro-XYZ
- tBodyGyroJerk-XYZ
- tBodyAccMag
- tGravityAccMag
- tBodyAccJerkMag
- tBodyGyroMag
- tBodyGyroJerkMag
- fBodyAcc-XYZ
- fBodyAccJerk-XYZ
- fBodyGyro-XYZ
- fBodyAccMag
- fBodyAccJerkMag
- fBodyGyroMag
- fBodyGyroJerkMag

- We can esitmate some set of variables from the above signals. ie., We will estimate the following properties on each and every signal that we recoreded so far.

- For better remember : we can see above image, EXPERTS apply some filter on each window and get 1st vector, 2nd vector and…….. so on. On top of these vector they computed below listed function.

  - **mean()**: Mean value
  - **std()**: Standard deviation
  - **mad()**: Median absolute deviation
  - **max()**: Largest value in array
  - **min()**: Smallest value in array
  - **sma()**: Signal magnitude area
  - **energy()**: Energy measure. Sum of the squares divided by the number of values.
  - **iqr()**: Interquartile range
  - **entropy()**: Signal entropy
  - **arCoeff()**: Autorregresion coefficients with Burg order equal to 4
  - **correlation()**: correlation coefficient between two signals
  - **maxInds()**: index of the frequency component with largest magnitude
  - **meanFreq()**: Weighted average of the frequency components to obtain a mean frequency
  - **skewness()**: skewness of the frequency domain signal
  - **kurtosis()**: kurtosis of the frequency domain signal
  - **bandsEnergy()**: Energy of a frequency interval within the 64 bins of the FFT of each window.
  - **angle()**: Angle between to vectors.

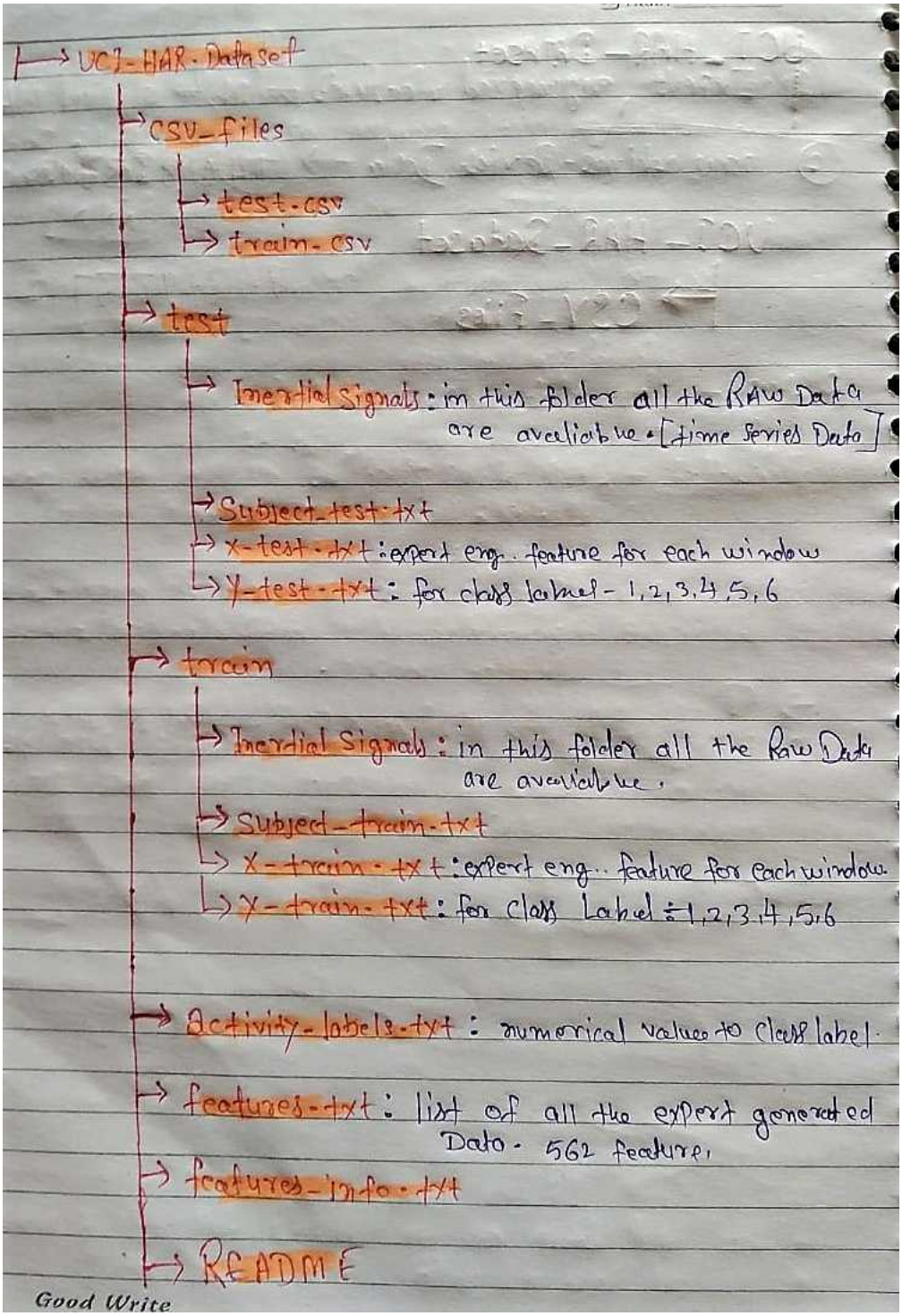
- We can obtain some other vectors by taking the average of signals in a single window sample. These are used on the angle() variable' `

  - gravityMean
  - tBodyAccMean
  - tBodyAccJerkMean
  - tBodyGyroMean
  - tBodyGyroJerkMean

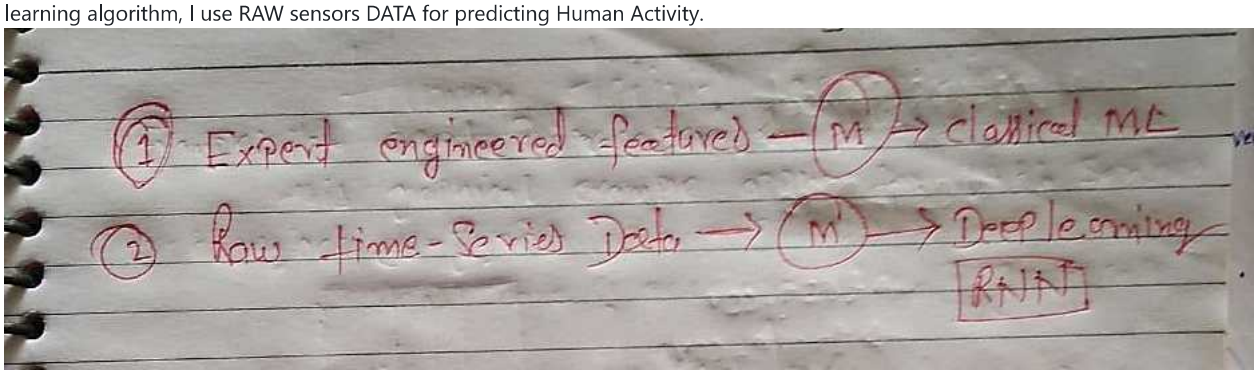
3. Y_Labels(Encoded)

- In the dataset, Y_labels are represented as numbers from 1 to 6 as their identifiers.
  - WALKING as **1**
  - WALKING_UPSTAIRS as **2**
  - WALKING_DOWNSTAIRS as **3**
  - SITTING as **4**
  - STANDING as **5**
  - LAYING as **6**

4. Data Directory



- **Important Note**: When I am applying Machine learning algorithm, I use these experts created feature data. When we are applying Deep learning algorithm, I use RAW sensors DATA for predicting Human Activity.



- The data is provided as a single zip file that is about **58 megabytes** in size. The direct link for this download is: [UCI HAR Dataset.zip]

Train and Test ratio

30 subjects(*volunteers*) data is randomly split to **70%** of the volunteers were taken as **training data** and remaining **30%** subjects' recordings were taken for **test data**. *e.g.* 21 subjects for train and nine for test.

Agenda

1. Analyzing the Data (EDA)

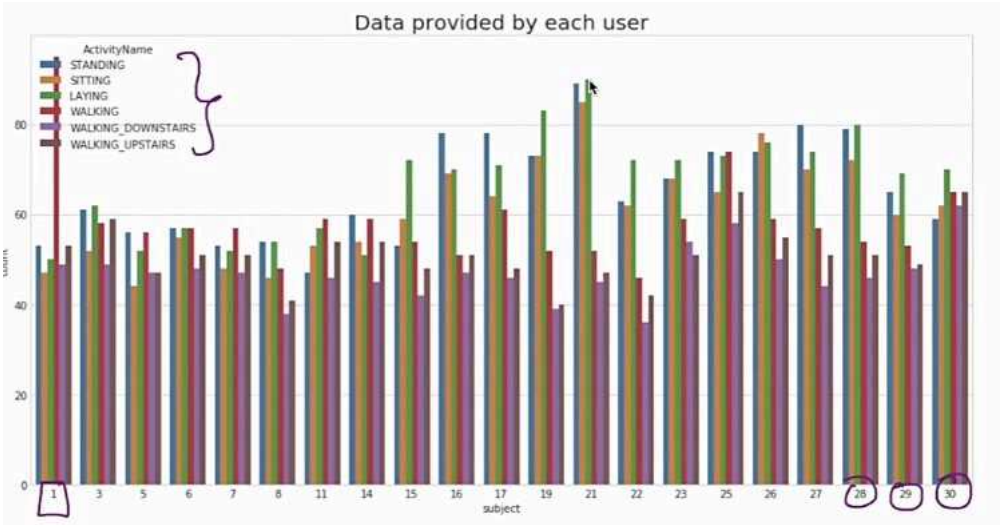
- Some Analysis on Data Set below:
- Here, first I perform EDA on Expert generated Data set. We try to understand the data then create some machine Learning model on top of it.
- Start with loading the feature.txt file then train data and test data and analysis these data.
- Total Data point and feature count in train and test data:

```
train = pd.read_csv('UCI_HAR_dataset/csv_files/train.csv')
test = pd.read_csv('UCI_HAR_dataset/csv_files/test.csv')
print(train.shape, test.shape)
```

```
Output: (7352, 564) (2947, 564)
```

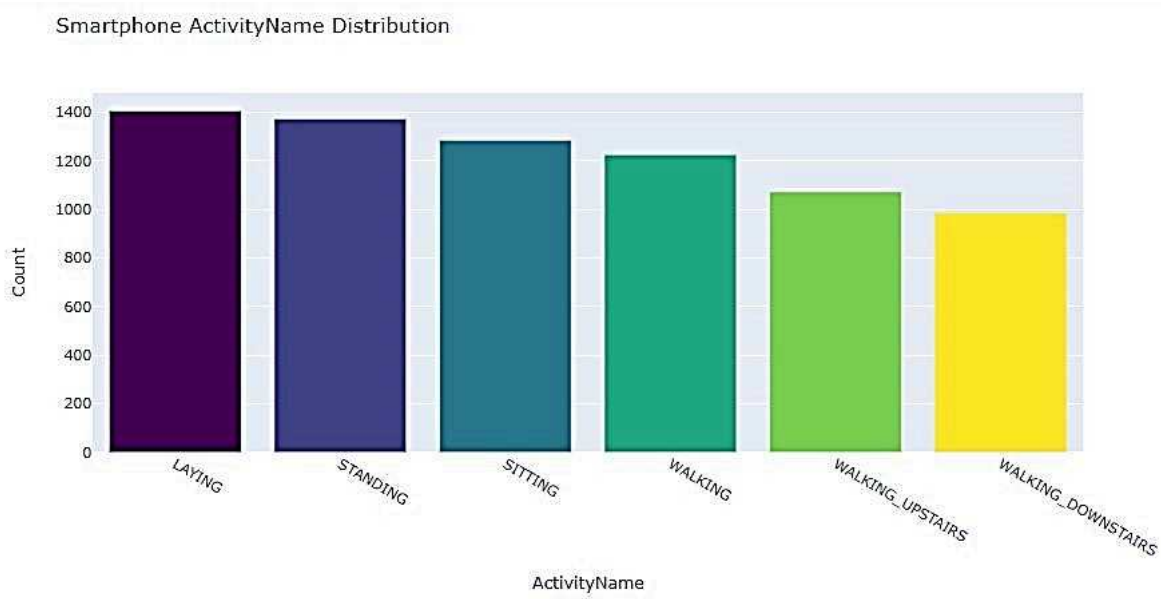
- **investigate participants activity durations**: Since the dataset has been created in a scientific environment nearly equal preconditions for the participants can be assumed. Let us investigate their activity durations.

```
sns.set_style('whitegrid')
plt.rcParams['font.family'] = 'Dejavu Sans'
plt.figure(figsize=(16,8))
plt.title('Data provided by each user', fontsize=20)
sns.countplot(x='subject',hue='ActivityName', data = train)
plt.show()
```



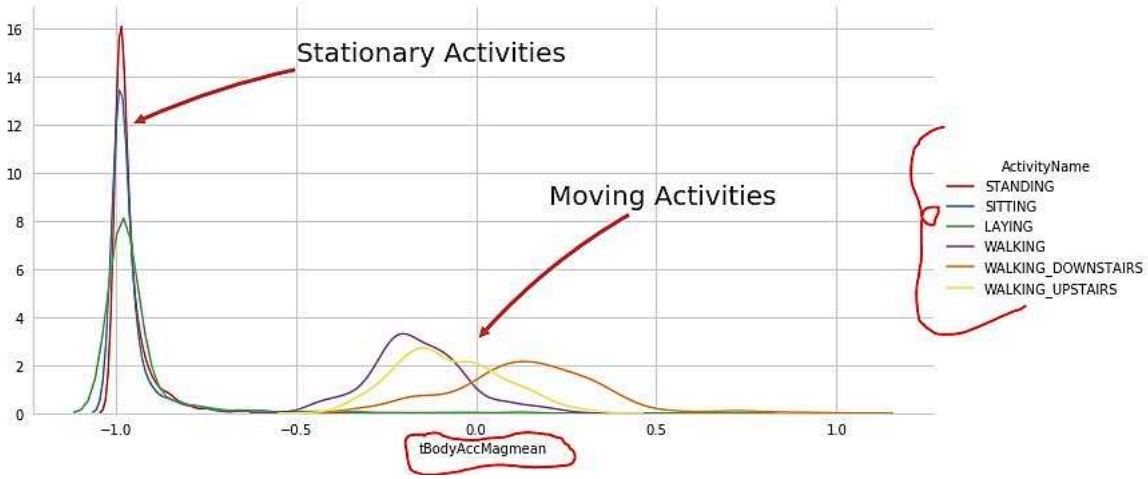
- Nearly all participants have more data for walking upstairs than downstairs. Assuming an equal number of up- and down-walks the participants need longer walking upstairs.
- We know that we have six class classification so, we have the big problem is to know or check is there any imbalanced in the data. And after plotting above graph we can say data is balanced.
- We have got almost same number of reading from all the subject.

- **Next question is how many data points do I have per class level.**



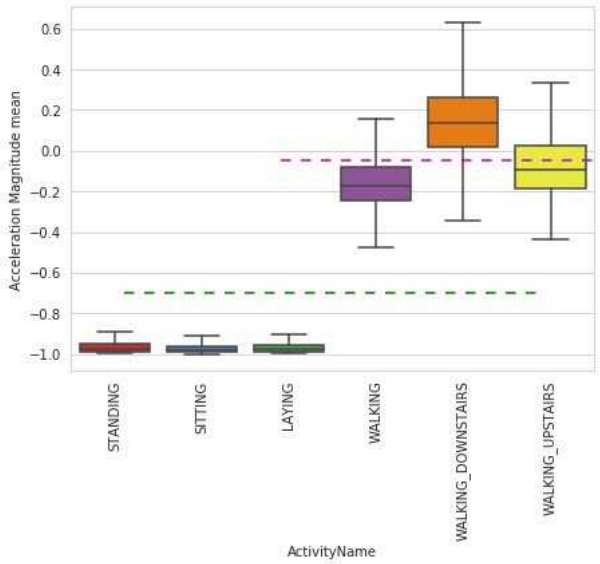
  - Data is almost balanced.
  - Although there are fluctuations in the label counts, the labels are quite equally distributed.
  - Assuming the participants had to walk the same number of stairs upwards as well as downwards and knowing the smartphones had a constant sampling rate, there should be the same amount of datapoints for walking upstairs and downstairs.
  - Disregarding the possibility of flawed data, the participants seem to walk roughly 10% faster downwards.

- Now time is to know **Static and Dynamic Activities of Human**:
  - In static activities (sit, stand, lie down) motion information will not be very useful.
  - In the dynamic activities (**Walking, WalkingUpstairs,WalkingDownstairs**) motion info will be significant.
  - Here we are using "**tBodyAccMagmean**" (*tBody acceleration magnitude feature mean value*) function to plot the graph for better understanding of *Static and Dynamic Activities of Human*.



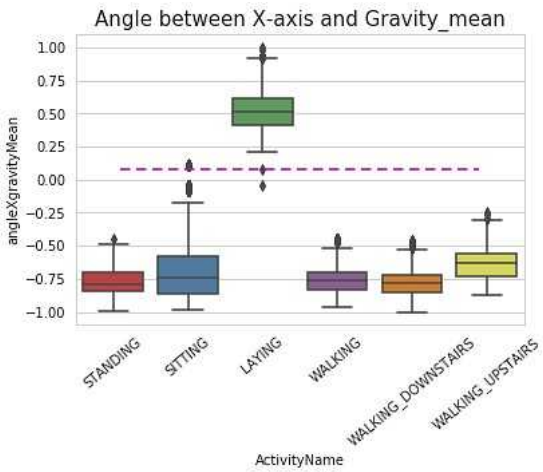
  - We can see **tbodyAccMagmean** feature separate very well the *Static and Dynamic Activities of Human*.

- Now we plot the **Static and Dynamic Activities of Human on Box plot** for understanding:



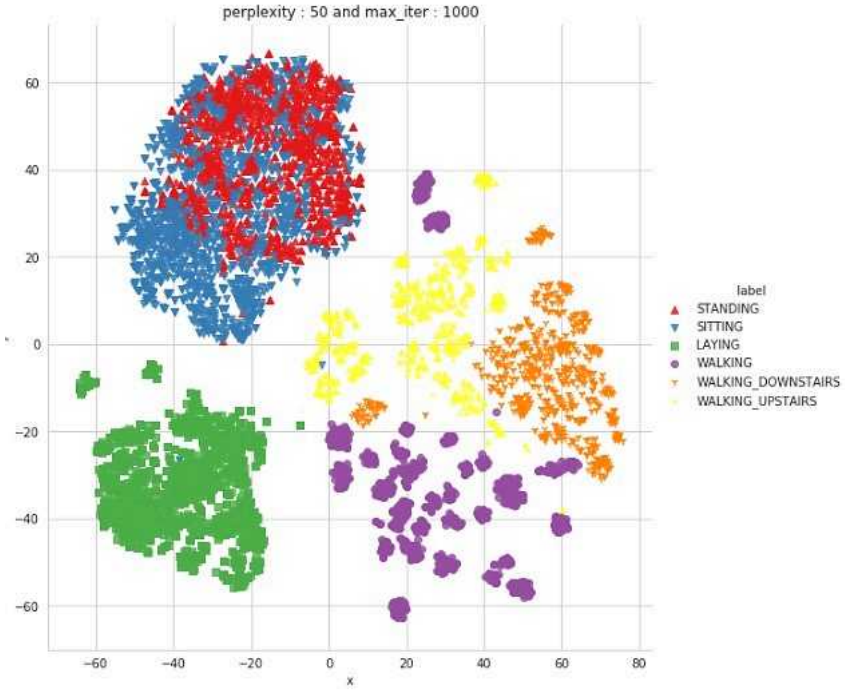
  - If **tAccMean** is < **-0.8** then the Activities are either *Standing* or *Sitting* or *Laying*.
  - If **tAccMean** is > **-0.6** then the Activities are either *Walking* or *WalkingDownstairs* or *WalkingUpstairs*.
  - If **tAccMean** > **0.0** then the Activity is **WalkingDownstairs**.
  - We can classify **75%** the Acitivity labels with some errors.

- Position of **GravityAccelerationComponants** also matters:



  - If angleX, **gravityMean** > **0** then Activity is *Laying*.
  - We can classify all datapoints belonging to Laying activity with just a single if else statement.

- Apply **t-sne** on the data to know **how much the Activities are Separable?** We know that we have 561-Dimension expert engineered feature now apply TSNE on these features to see how much these features are helpful.



  - We can clearly see the *TSNE cluster*, All the Activity are clean separate **except "Standing" and "Sitting"**.

2. Machine Learning Models:

- **Important Note as we discussed previous**: I used the **561 expert engineered features** and we will *apply classical Machine Learning Model* on top of it.
- The Machine Learning Model which I applied are:

a. Logistic Regression

- Logistic regression is a linear model for classification. In this model, the probabilities describing the possible outcomes of a single trial are modeled using a logistic function. The logistic function is a sigmoid function, which takes any real input and outputs a value between 0 and 1, and hence is ideal for classification.

  When a model learns the training data too closely, it fails to fit new data or predict unseen observations reliably. This condition is called overfitting and is countered, in one of many ways, with ridge (L2) regularization. Ridge regularization penalizes model predictors if they are too big, thus enforcing them to be small. This reduces model variance and avoids overfitting.

- **Hyperparameter Tuning**: Cross-validation is a good technique to tune model parameters like regularization factor and the tolerance for stopping criteria (for determining when to stop training). Here, a validation set is held out from the training data for each run (called fold) while the model is trained on the remaining training data and then evaluated on the validation set. This is repeated for the total number of folds (say five or 10) and the parameters from the fold with the best evaluation score are used as the optimum parameters for the model.

b. Linear SVC

- The objective of a **Linear SVC** (Support Vector Classifier) is to fit to the data you provide, returning a "best fit" hyperplane that divides, or categorizes, your data. From there, after getting the hyperplane, you can then feed some features to your classifier to see what the "predicted" class is.

c. Kernal SVM

- **SVM** algorithms use a set of mathematical functions that are defined as the **kernel**. The function of **kernel** is to take data as input and transform it into the required form. Different SVM algorithms use different types of kernel functions. These functions can be different types.

d. Decision Tree

- Decision trees is a hierarchical model also known as classification and regression trees. They have the property of predicting response from data. The attributes of the decision trees are mapped into nodes. The edges of the tree represent the possible output values. Each branch of the tree represents a classification rule, from the root to the leaf node.
- This method has been used for several tasks in the field of pattern recognition and machine learning as a predictive model. The main goal is to predict the next value given several input variable.

e. Random Forest Classifier

- Random Forest is an outfit of unpruned demand or descends like bootstrapping algorithm with various decision trees. Each tree depends upon the estimations of the vector picked unpredictably and independently. Random Forest reliably gives an immense improvement than the single tree classifier. Each tree is fabricated using the algorithm.

f. Gradient Boosted

- Gradient boosting is an AI method for relapse and order issues, which creates an expectation model as a group of powerless forecast models, normally choice trees. The goal of any directed learning algorithm is to characterize a misfortune work and limit it. Gradient boosting machines are in light of a ensemble of choice trees where numerous weak learner trees are utilized in mix as a group to give preferred forecasts over singular trees. Boost has unrivalled regularization and better treatment of missing qualities and also much improved proficiency.
- **NOTE**: I am trying to run the "GradientBoostingClassifier()" with "GridSearchCV", but my system is not supported this pice of code.

3. Deep Learning Models:

- Now, I created **LSTM based Deep learning Model** on the **Raw time series Data**.
- HAR is one of the time series classification problem. In this project various machine learning and deep learning models have been worked out to get the best final result. In the same sequence, we can use LSTM (long short-term memory) model of the Recurrent Neural Network (RNN) to recognize various activities of humans like standing, climbing upstairs and downstairs etc.
- **LSTM model** is a type of recurrent neural network capable of learning order dependence in sequence prediction problems. This model is used as this helps in remembering values over arbitrary intervals.
- I applied LSTM as follows:
  - **1-Layer of LSTM**
  - **2-Layer of LSTM with more hyperparameter tuning**

4. Results & Conclusion

- For below table we are comparing all the **ML model Accuracy score**.

Model Name	Features	Hyperparameter Tuning	Accuracy Score
Logistic Regression	`Expert generated Feature`	Done	**95.83%**
Linear SVC	`Expert generated Feature`	Done	**96.47%**
RBF SVM classifier	`Expert generated Feature`	Done	**96.27%**
Decision Tree	`Expert generated Feature`	Done	**86.46%**
Random Forest	`Expert generated Feature`	Done	**92.06%**

- We can choose **Linear SVC** or **rbf SVM classifier** or **Logistic Regression** as our best model while applying ML Classical Model.
- For Below table we are comparing **Deep Learning LSTM Model**.

Model Name	Features	Hyperparameter Tuning	crossentropy	Accuracy Value
LSTM With 1 La…	…	Done	0.47…	0.90%

Releases

No releases published

Packages

No packages published

Languages

- Jupyter Notebook 100.0%