Machine Learning Engineer Nanodegree

Capstone Project

Aanvi Goel

April 23rd, 2020

I. Definition

Project Overview

Over the last decade, fitness tracking has become a rising trend due to the convenience of using devices such as smartphones, wrist/body trackers, and smart watches. Such devices collect user data from sensors such as accelerometer, gyroscope, heartbeat monitor for calibrations and measurements.

A key part of fitness tracking is understanding the user's state of motion and autodetection of their activity has become one of the most important features. This will be further explored through this project.

Problem Statement

In this project, I aim to accurately predict the activity of a user based on a smartphone's sensor data – essentially using signals from the accelerometer which measures acceleration and gyroscope sensors which measures angular velocity.

Based on the features extracted from the input signals, 6 different user activities can be identified – 3 static activities (standing, sitting, lying) and 3 dynamic activities (walking, walking up, walking down). This problem can be characterized as a multiclass classification problem. A multiclass classification problem is a classification task which has more than two classes to choose from and needs to assign only one label to each sample.

From the dataset, we are using training examples that can be divided into 6 separate classes. The model learns the patterns specific to each class and uses those patterns to predict the activity label of future data. The task of the machine learning model is to predict which of these classes new data would belong to.

The overall goal of this project is to implement effective machine learning models which would accurately identify the activity label for a user, based on input signals from their smartphone sensors.

Metrics

To evaluate the performance of the models I compared the models based on two main metrics:

1) **Accuracy score**
   Accuracy is the closeness of measurements to a specific value.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

   It can also be calculated as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

where TP – True Positive, TN = True Negatives, FP = False Positives, and FN = False Negatives.

True Positives (TP_i): number of samples of a class_i correctly predicted as samples of the same class_i.

False Negatives (FN_i): number of samples of a class_i incorrectly predicted as a sample of another class_j

False Positives (FP_i): number of samples of other classes_j incorrectly predicted as samples of class_i.

True negatives (TN_i): number of samples belongs to other classes_j not predicted as samples of the actual class_i

2) **Confusion-Matrix**
   A confusion matrix, also known as the error matrix, evaluates the accuracy of a classification. It checks if the model is confused between two different activities and predicting incorrect classification labels.

Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class (or vice versa).

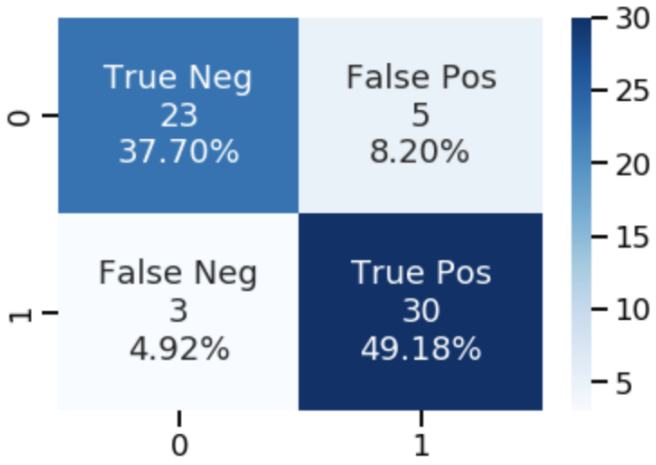


Figure 1: Example of a confusion matrix [2]

II. Analysis

Data Exploration

Datasets and Inputs

Data is downloaded from following source:
https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones

The experiments have been carried out with a group of 30 volunteers who performed the following 6 activities wearing a smartphone on their waist.

Walking -> 1
Walking Up -> 2
Walking Down -> 3
Standing -> 4
Sitting -> 5
Lying -> 6

Using the smartphone’s embedded accelerometer and gyroscope, 3-axial linear acceleration and 3-axial angular velocity are captured at a constant rate of 50Hz. These sensor signals are pre-processed by applying noise filters and then sampled in fixed-width windows (sliding windows) of 2.56 seconds each with 50% overlap. i.e., each window has 128 readings. A 128-size vector is created from each window. From each 128 readings, 561 different features have been engineered by calculating variables from the time and frequency domain. The 561 features are stored in the file "Features.docx" in the “Data” folder.

By using a low pass filter with corner frequency of 0.3Hz, the acceleration signal is separated into Body and Gravity acceleration signals (tBodyAcc-XYZ and tGravityAcc-XYZ). The body linear acceleration and angular velocity are derived in time to obtain jerk signals. Time domain signals are converted to frequency domain by applying a FFT (Fast Fourier Transform). These signals obtained are labelled with prefix 'f' just like original signals with prefix 't'. For each window, the mean, max, mad, sma, arcoefficient, energy-bands, entropy etc., are also calculated for base signals.

The obtained dataset is randomly partitioned into two sets, where 70% for training data and 30% for the test data. The dataset is balanced between the multiple classes as can be seen from the image below.

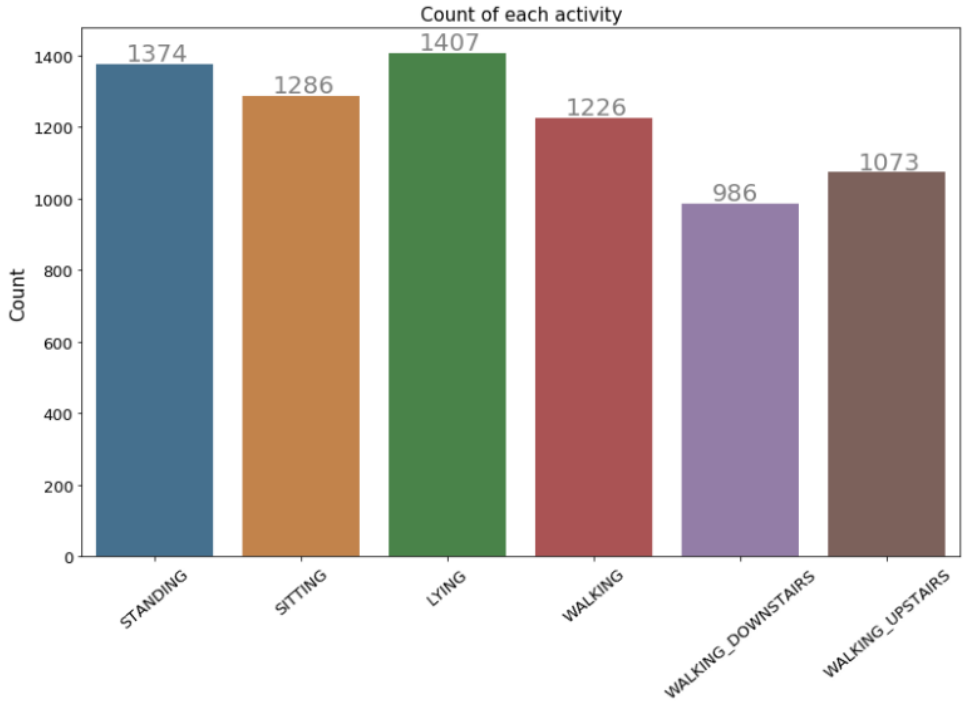


Figure 2: Distribution of classes in dataset

Algorithms and Techniques

To solve this multiclassification problem, I used the following learning models:

1. Supervised Learning: Random Forest Classifier
2. Supervised Learning: Linear SVM
3. Deep Learning: LSTM

Supervised Learning:

Supervised learning is a method by which you can use labeled training data to train a function that you can then generalize for new examples. The training involves a critic that can indicate when the function is correct or not, and then alter the function to produce the correct result.

In supervised learning, you create a function (or model) by using labeled training data that consists of input data and a wanted output. The supervision comes in the form of the wanted output, which in turn lets you adjust the function based on the actual output it produces. When trained, you can apply this function to new observations to produce an output (prediction or classification) that ideally responds correctly.

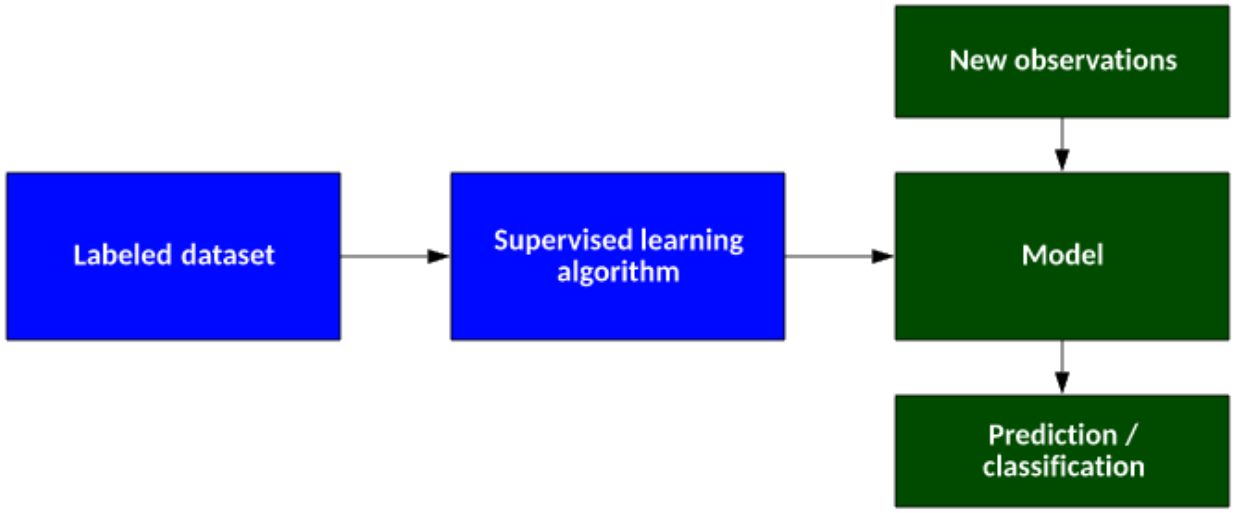


Figure 3: A typical supervised learning algorithm

Random Forest Classifier:

It is an ensemble tree-based learning algorithm. The Random Forest Classifier creates a set of decision trees from randomly selected subset of training set. It aggregates the votes from different decision trees to decide the final class of the test object.

Suppose training set is given as: [X1, X2, X3, X4] with corresponding labels as [L1, L2, L3, L4], random forest may create three decision trees taking input of subset for example,

1. [X1, X2, X3]
2. [X1, X2, X4]
3. [X2, X3, X4]

So finally, it predicts based on the majority of votes from each of the decision trees made.

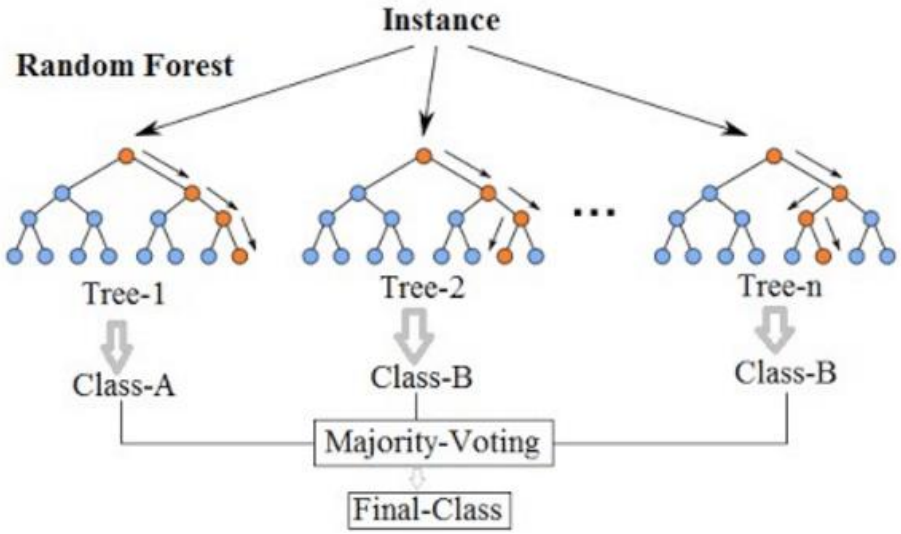


Figure 4: Structure of a Random Forest Classification

Linear SVM:

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. In two-dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side.

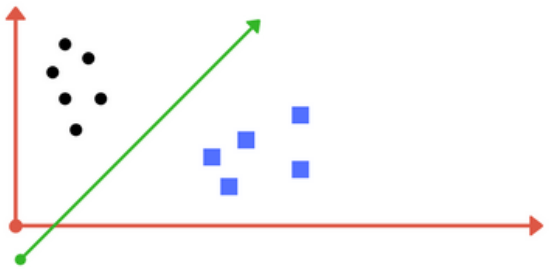


Figure 5: Example of hyperplane dividing dataset into two classes

LSTM:

Long Short Term Memory networks – usually just called "LSTMs" – are a special kind of Recurrent Neural Networks (RNN), capable of learning long-term dependencies. LSTMs are explicitly designed to avoid the long-term dependency problem.

All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure. LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.

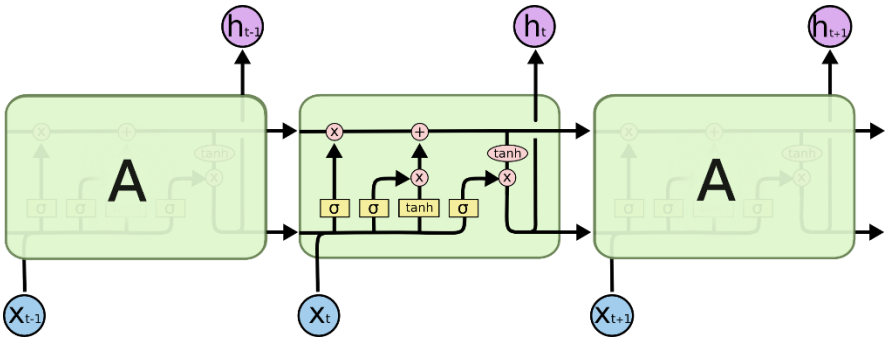


Figure 6: Repeating module of an LSTM

Benchmark

A Decision Tree is a simple representation for classifying examples. It is a Supervised Machine Learning where the data is continuously split according to a certain parameter.

Decision Tree consists of:

Nodes: Test for the value of a certain attribute.
Edges/ Branch: Correspond to the outcome of a test and connect to the next node or leaf.
Leaf nodes: Terminal nodes that predict the outcome (represent class labels or class distribution).

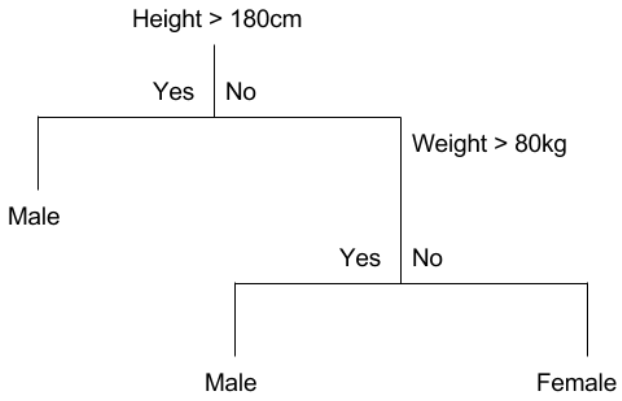


Figure 7: Example of a simple classification tree

Decision tree is a simple classification model which is fast, easy to interpret and fairly accurate for simple data sets.

III. Methodology

Data Preprocessing

Data Preparation:

Pandas dataframes are created from training and test data csv files. Each column of the dataframe represents the feature value for the given subject(user) and feature column. Additional columns for subject ID, activity and label are added. Any empty or NAN records are dropped using the dropna() function.

	1 tBodyAcc- mean()-X	2 tBodyAcc- mean()-Y	3 tBodyAcc- mean()-Z	4 tBodyAcc- std()-X	5 tBodyAcc- std()-Y	6 tBodyAcc- std()-Z	7 tBodyAcc- mad()-X	8 tBodyAcc- mad()-Y
0	0.288585	-0.020294	-0.132905	-0.995279	-0.983111	-0.913526	-0.995112	-0.983185
1	0.278419	-0.016411	-0.123520	-0.998245	-0.975300	-0.960322	-0.998807	-0.974914
2	0.279653	-0.019467	-0.113462	-0.995380	-0.967187	-0.978944	-0.996520	-0.963668
3	0.279174	-0.026201	-0.123283	-0.996091	-0.983403	-0.990675	-0.997099	-0.982750
4	0.276629	-0.016570	-0.115362	-0.998139	-0.980817	-0.990482	-0.998321	-0.979672

Feature Selection:

My dataset has 561 different features- which can be a lot of features for training a model. Not all features contribute to the classification decision. We also run the risk of over-fitting our model by training it on a feature-rich dataset.

To reduce the number of features, I chose to analyze the most important features by using the feature importance functionality in sklearn library. Feature importance is a class of techniques that assign a score to input features based on how useful they are at predicting a target variable.

Using the importance scores obtained, I decided to select only the highest scoring features, i.e. the top 125 features.

```
53 tGravityAcc-min()-X\n                 0.240265
394 fBodyAccJerk-bandsEnergy()-1,24\n    0.202710
560 angle(Y,gravityMean)\n               0.132955
509 fBodyAccMag-energy()\n               0.109366
75 tGravityAcc-arCoeff()-Z,2\n           0.096493
449 fBodyGyro-maxInds-X\n                0.022702
160 tBodyGyro-correlation()-Y,Z\n        0.015639
58 tGravityAcc-energy()-Y\n              0.015564
210 tBodyAccMag-arCoeff()1\n             0.014908
130 tBodyGyro-max()-X\n                  0.008665
276 fBodyAcc-max()-Y\n                   0.008357
54 tGravityAcc-min()-Y\n                 0.007554
38 tBodyAcc-correlation()-X,Y\n          0.007002
51 tGravityAcc-max()-Y\n                 0.005957
133 tBodyGyro-min()-X\n                  0.005184
433 fBodyGyro-max()-X\n                  0.004969
118 tBodyAccJerk-correlation()-X,Y\n     0.003757
170 tBodyGyroJerk-max()-X\n              0.003287
72 tGravityAcc-arCoeff()-Y,3\n           0.002864
52 tGravityAcc-max()-Z\n                 0.002829
dtype: float64
```

Figure 8: Snapshot of top 20 features with scores

Implementation

1. Decision Trees Classifier is implemented as a benchmark model.
2. Using the benchmark model, the feature importance scores are calculated, and top 125 features selected for the final learning model implementations. Low scoring features are dropped from the dataframe.
3. First supervised learning algorithm implemented is a Random Forest Classifier. The model is run with 100 estimators.
4. The second model Linear Support Vector Classifier is implemented.

5. For the deep learning model – RNN with LSTM, the signals are directly considered as inputs for the model. DataFrames are created with the input signal for x_train and x_test and the activity labels for y_train and y_test.
6. A 2-layer LSTM model is implemented with an ‘Adam’ optimizer.
7. For each model, the following metrics are recorded for comparison later:
    a. Accuracy
    b. Precision
    c. Recall
    d. F1-score
    e. Confusion matrix are recorded

IV. Results

Model Evaluation and Validation

The two supervised learning models outperformed the benchmark model. The accuracy jumped from 86% to 90-96%.

	Model	Accuracy
0	Decision Tree Classifier	0.862572
1	Random Forest Classifier	0.908381
2	Linear SVC	0.937903
3	LSTM	0.604004

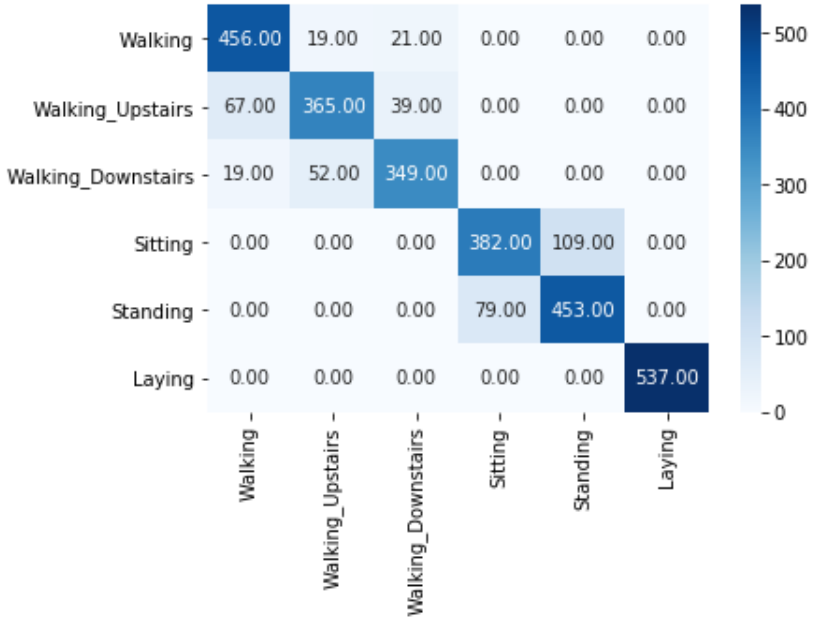
Justification

Benchmark:

	precision	recall	f1-score	support
1	0.84	0.92	0.88	496
2	0.84	0.77	0.80	471
3	0.85	0.83	0.84	420
4	0.83	0.78	0.80	491
5	0.81	0.85	0.83	532
6	1.00	1.00	1.00	537
accuracy			0.86	2947
macro avg	0.86	0.86	0.86	2947
weighted avg	0.86	0.86	0.86	2947

Accuracy: 0.8625721072276892

Confusion Matrix:

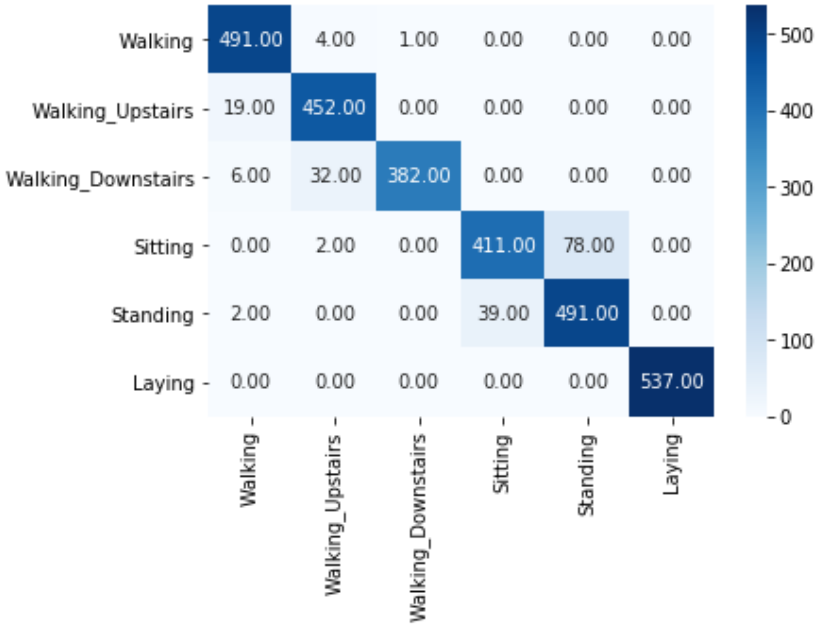


Random Forest Classifier:

	precision	recall	f1-score	support
1	0.95	0.99	0.97	496
2	0.92	0.96	0.94	471
3	1.00	0.91	0.95	420
4	0.91	0.84	0.87	491
5	0.86	0.92	0.89	532
6	1.00	1.00	1.00	537
accuracy			0.94	2947
macro avg	0.94	0.94	0.94	2947
weighted avg	0.94	0.94	0.94	2947

Accuracy: 0.9379029521547336

Confusion Matrix:

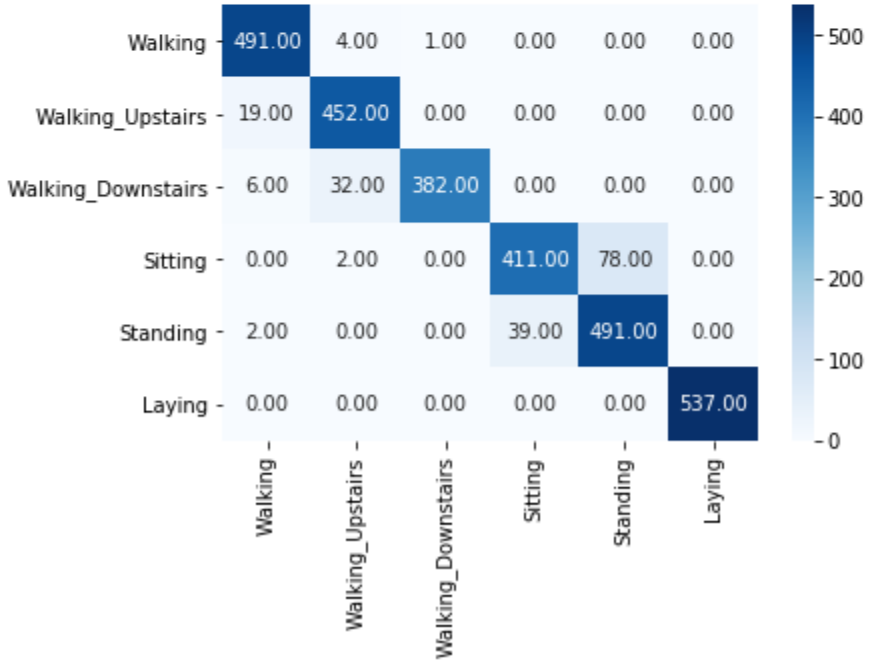


Linear SVM:

	precision	recall	f1-score	support
1	0.95	0.99	0.97	496
2	0.92	0.96	0.94	471
3	1.00	0.91	0.95	420
4	0.91	0.84	0.87	491
5	0.86	0.92	0.89	532
6	1.00	1.00	1.00	537
accuracy			0.94	2947
macro avg	0.94	0.94	0.94	2947
weighted avg	0.94	0.94	0.94	2947

Accuracy: 0.9379029521547336

Confusion Matrix:

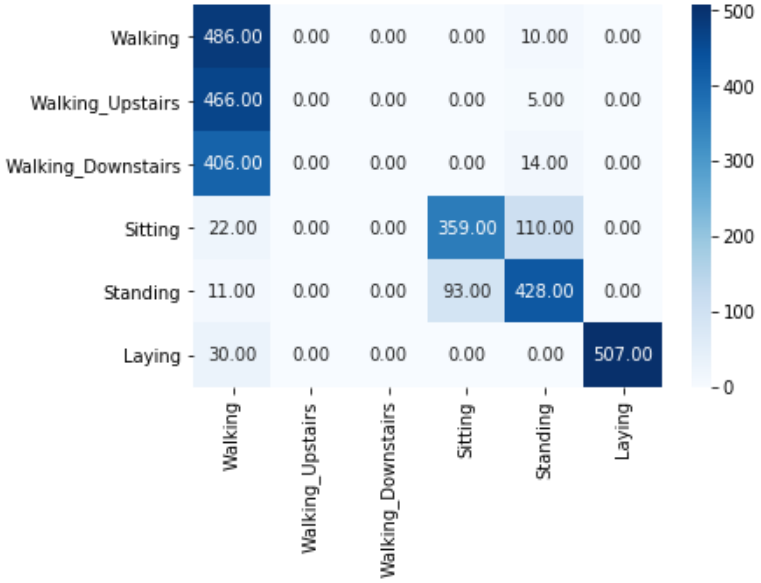


LSTM:

	precision	recall	f1-score	support
1	0.34	0.98	0.51	496
2	0.00	0.00	0.00	471
3	0.00	0.00	0.00	420
4	0.79	0.73	0.76	491
5	0.75	0.80	0.78	532
6	1.00	0.94	0.97	537
accuracy			0.60	2947
macro avg	0.48	0.58	0.50	2947
weighted avg	0.51	0.60	0.53	2947

Accuracy: 0.6040040719375637

Confusion Matrix:



V. Conclusion

Reflection

Both my supervised learning models performed well with good accuracy and marginal class confusion. The feature engineering, already a part of the dataset, played a major role is the performance of the models.

The LSTM model did not perform too well- with only 60% accuracy. This was expected since the model was implemented without preprocessing the input data. The implemented model is

also a simple 2-layer LSTM. By tuning the hyperparameters of the model, we can expect better performance. The advantage of the deep learning model is that the model can be directly applied to the input signals without having to engineer the features or preprocess the data with feature selection.

Improvement

1. The LSTM model implementation can be improvised using hyperparameter tuning to get better accuracy.
2. The predictions can be expanded to include transition labels as well, such as sit-to-stand, sit-lying etc.
3. The dataset can be expanded to include other activities such as running/jogging, exercising etc. This would help us get a more holistic understanding of real-life user activity.
4. The current data is collected using a smartphone placed at the user's waist. It would be interesting to explore other data points from other devices such as fitness trackers and smartwatches. We can see is a similar model can be applied to sensor data from those devices as well.

VI. References

1) *Powers, David M W (2011).* "Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation" (PDF). *Journal of Machine Learning Technologies. 2 (1): 37–63.*
2) https://medium.com/@dtuk81/confusion-matrix-visualization-fc31e3f30fea
3) https://developer.ibm.com/technologies/artificial-intelligence/articles/cc-supervised-learning-models/
4) https://towardsdatascience.com/random-forest-classification-and-its-implementation-d5d840dbead0
5) https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72
6) https://colah.github.io/posts/2015-08-Understanding-LSTMs/
7) https://machinelearningmastery.com/calculate-feature-importance-with-python/