Smartphone-Based Recognition of Human Activities and Postural Transitions Data Set

Pierrick Pujol

ESILV

Dataset Introduction

Activity recognition data set built from the recordings of 30 subjects performing basic activities and postural transitions while carrying a smartphone with embedded inertial sensors.

https://archive.ics.uci.edu/ml/datasets/Smartphone-Based+Recognition+of+Human+Activities+and+Postural+Transitions

Dataset context

The experiment was carried out on 30 participants (19 to 48 years)

They followed an activity protocol composed of 6 basic activities:

- 3 static postures : standing, sitting, lying
- 3 dynamic activities : walking, walking downstairs and walking upstairs

Dataset context

The experiment also included postural transitions that occurred between the static postures :

- Stand to sit
- Sit to stand
- Sit to lie
- Lie to sit
- Stand to lie
- Lie to stand

Data collection

All the participants were wearing a Samsung Galaxy S II

Using the accelerometer and gyroscope of the device they captured :

- 3-axial linear acceleration
- 3-axial angular velocity

The experiments were video-recorded to label the data manually

The obtained dataset was randomly partitioned into 2 sets :

- 70% of the volunteers were selected for generating the training data
- 30% the test data

Data processing

The data from the accelerometer and gyroscope was pre-processed by applying noise filters and then sampled in fixed-width sliding windows

From each window, a vector of 561 features was obtained by calculating variables from the time and frequency domain.

Goal : classification

Be able to predict the activity of participants, among the following 12 activities :

- WALKING
- WALKING_UPSTAIRS
- WALKING_DOWNSTAIRS
- SITTING
- STANDING
- LAYING
- STAND_TO_SIT
- SIT_TO_STAND
- SIT_TO_LIE
- LIE_TO_SIT
- STAND_TO_LIE
- LIE_TO_STAND

Data is difficult to use

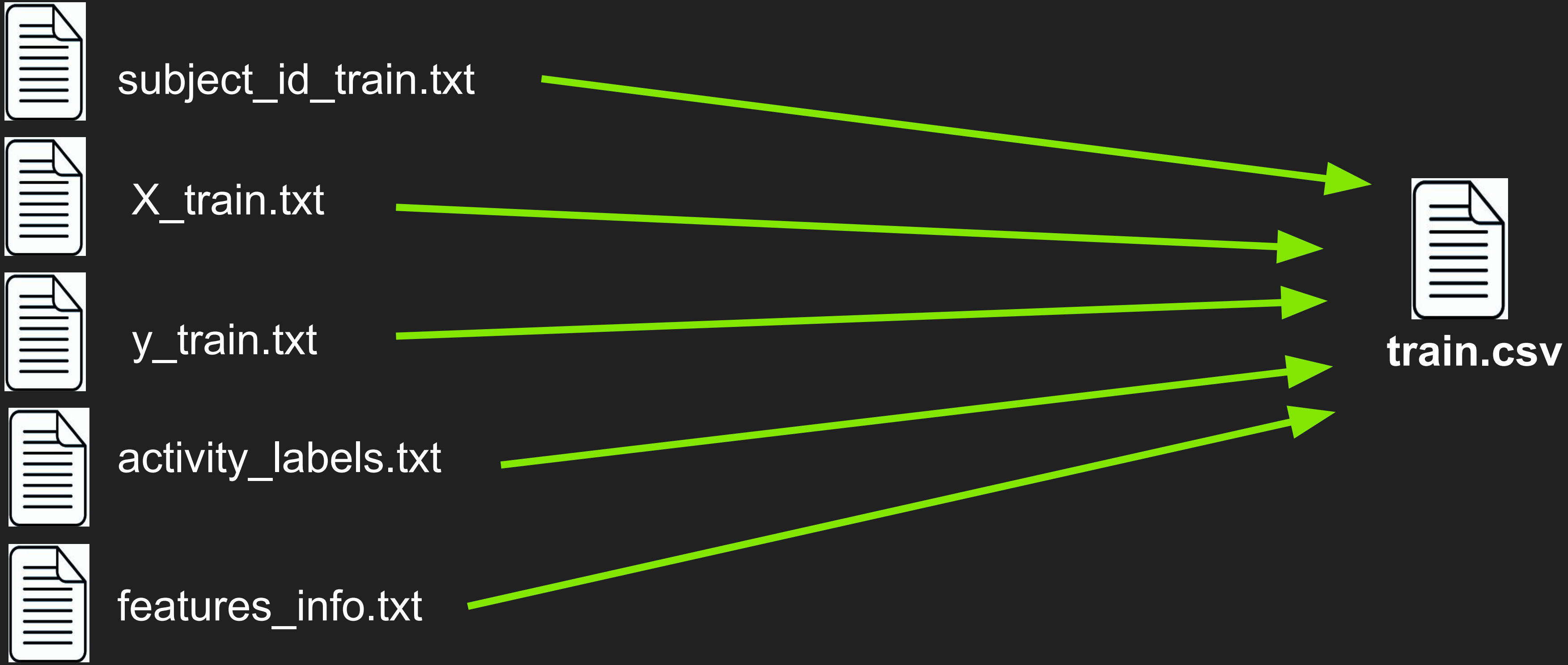
The data is distributed in several files.

To use train/test data :

- subject_id_train.txt : id of each subject (1 to 30)
- X_train.txt : data from accelerometer and gyroscope
- y_train.txt : the activity for each line of X_train.txt
- activity_labels.txt : the correspondence between number and activity name
- features_info.txt : the list of features

Data preprocessing : group data

Create a python script that will allow you to group the data into a single csv file



Data preprocessing

Steps completed :

- Checking that there is no lack of value
- Checking that there is no duplicate value
- Checking the datatypes of the 2 files (train.csv et test.csv)
- Checking if we need to scale the dataset

	tBodyAcc-Mean-1	tBodyAcc-Mean-2	tBodyAcc-Mean-3	tBodyAcc-STD-1	tBodyAcc-STD-2	tBodyAcc-STD-3	tBodyAcc-Mad-1	tBodyAcc-Mad-2	tBodyAcc-Mad-3	tBodyAcc-Max-1	...	fBodyGyroJer Skewr
count	7767.000000	7767.000000	7767.000000	7767.000000	7767.000000	7767.000000	7767.000000	7767.000000	7767.000000	7767.000000	...	7767.0
mean	0.038759	-0.000647	-0.018155	-0.599017	-0.634424	-0.691270	-0.623886	-0.657884	-0.740154	-0.360200	...	-0.3
std	0.101996	0.099974	0.089927	0.441481	0.367558	0.321641	0.418113	0.348005	0.272619	0.499259	...	0.3
min	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	...	-1.0
25%	0.032037	-0.011209	-0.028448	-0.992140	-0.983570	-0.984661	-0.992902	-0.984131	-0.986661	-0.795613	...	-0.5
50%	0.038975	-0.002921	-0.019602	-0.914202	-0.827970	-0.827696	-0.924421	-0.838559	-0.852735	-0.717007	...	-0.3
75%	0.044000	0.004303	-0.011676	-0.246026	-0.313069	-0.450478	-0.294903	-0.362671	-0.540521	0.054178	...	-0.1
max	1.000000	1.000000	1.000000	1.000000	0.945956	1.000000	1.000000	0.960341	1.000000	1.000000	...	0.9

8 rows × 562 columns

In general the fact of scaling a dataset makes it possible to improve the accuracy of the models.

We can see that the values are between -1 and +1, therefore it is not necessary to scale.

Data Visualisation

What are the main features ?

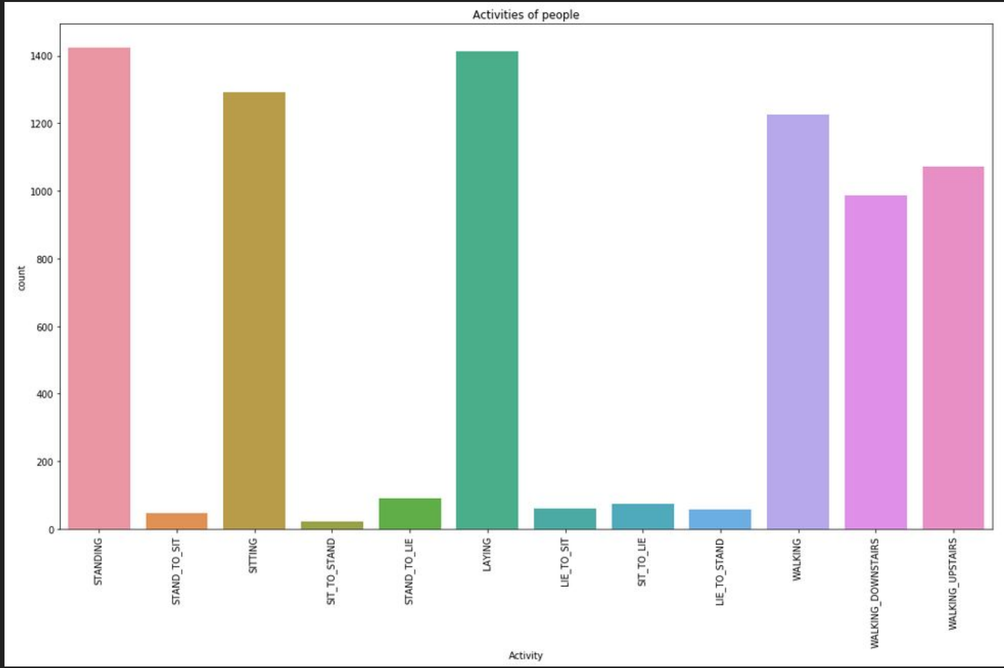
	count
fBodyAccJerk	79
fBodyGyro	79
fBodyAcc	79
tBodyGyroJerk	41
tBodyAcc	41
tBodyGyro	41
tBodyAccJerk	41
tGravityAcc	40

- fBodyAcc* = acceleration feature
- fBodyGyro* = gyroscope feature
- tGravity* = gravity feature

We can see that there are mainly gyroscope and acceleration features.

Data Visualisation

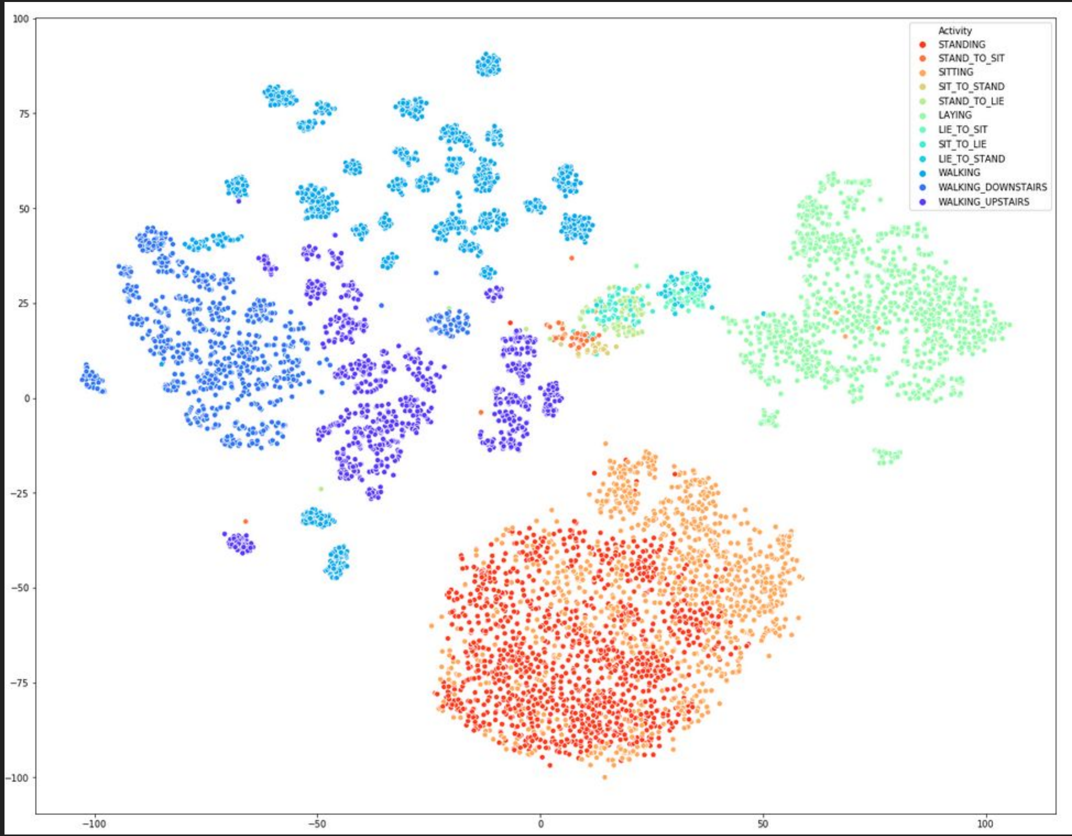
Checking for class imbalance



The data of postural transitions represent a minority of the dataset

The activities are not unbalanced, there are roughly the same number of samples for each static activity. It's the same for the postural transitions, there are roughly the same number of samples for each type of postural transitions

Data Visualisation



Some data seems more difficult to separate than others, for example standing and sitting data seem rather mixed, there is also this problem with the activities walking and walking upstairs

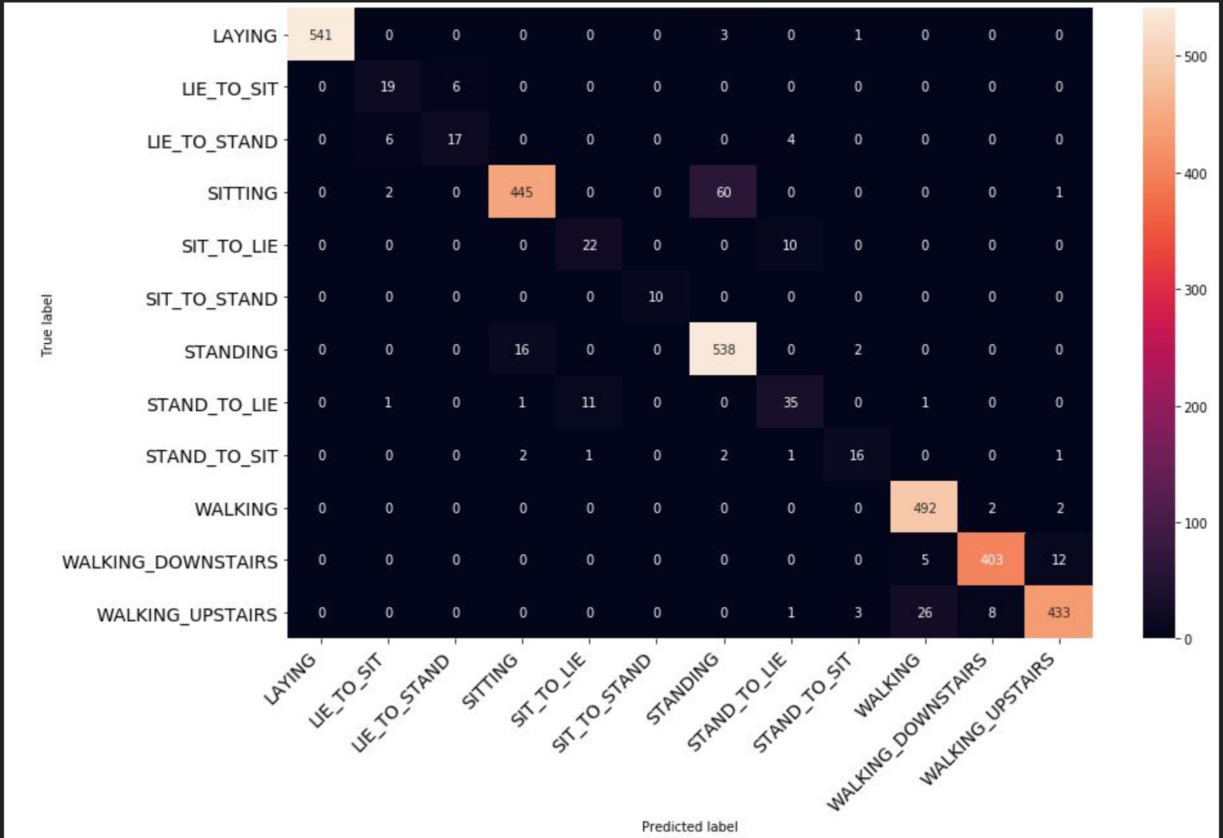
Data Modelling

The algorithms used :

- Logistic regression
- LGBM
- Linear SVC
- Decision tree
- Random Forest

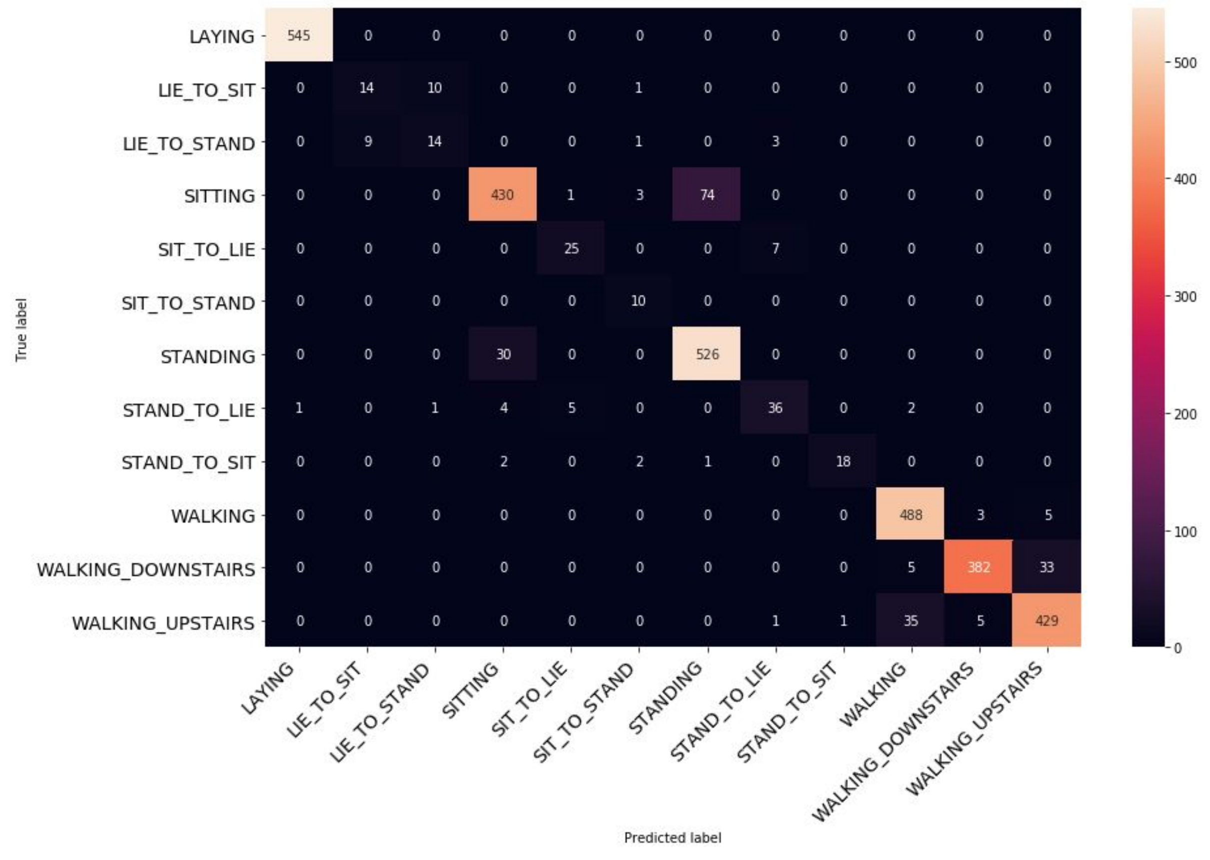
Hyperparameter tuning : RandomizedSearchCV() from sklearn

Logistic regression model with hyperparameter tuning



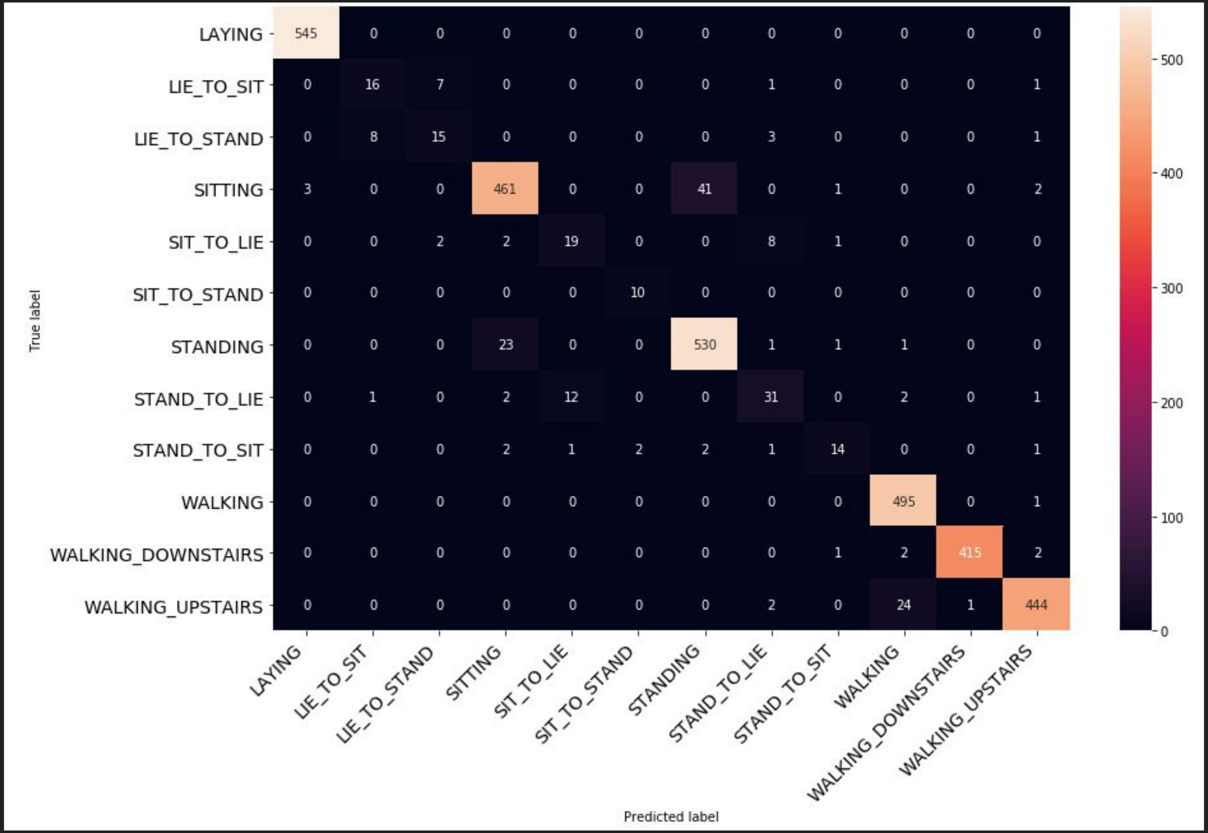
Accuracy :
0.939595192915876

LGBM classifier



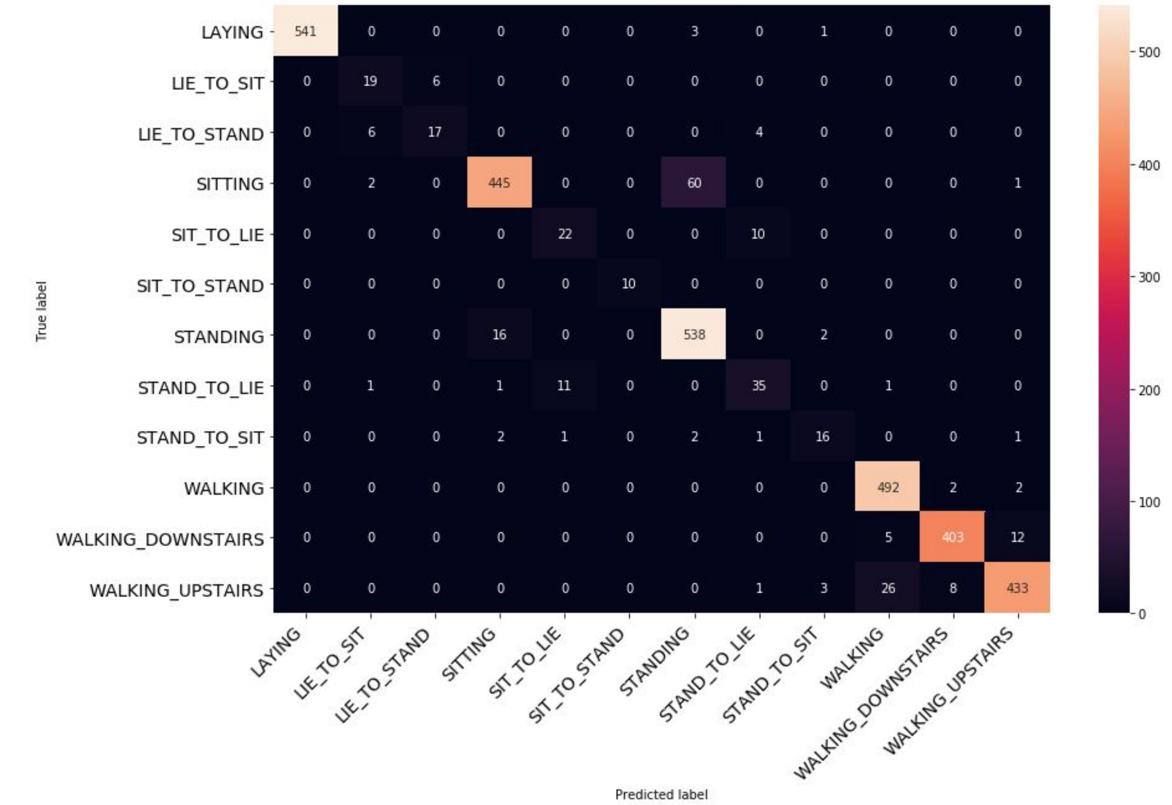
Accuracy :
0.9225173940543959

Linear SVC model with hyperparameter Tuning



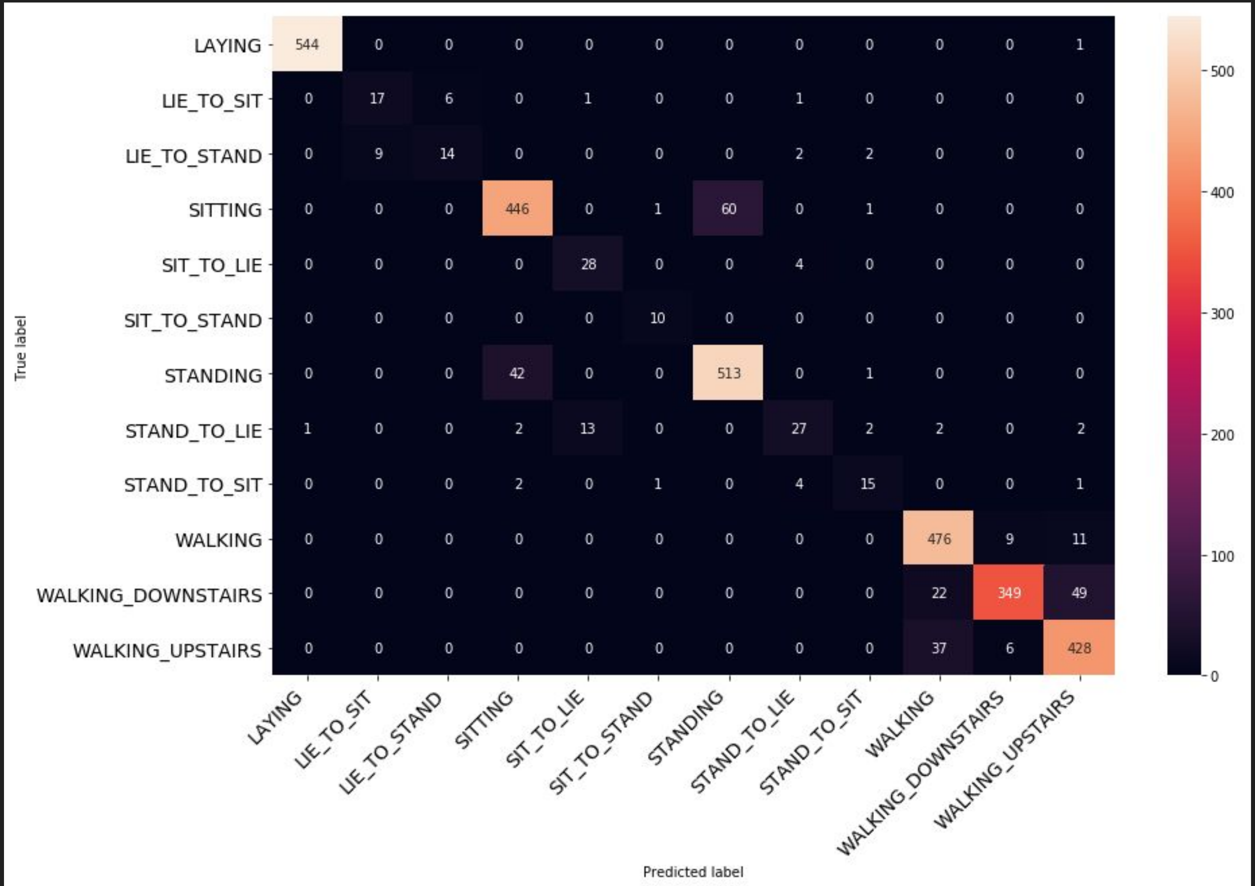
Accuracy :
0.9471853257432005

Decision tree classifier with hyperparameter Tuning



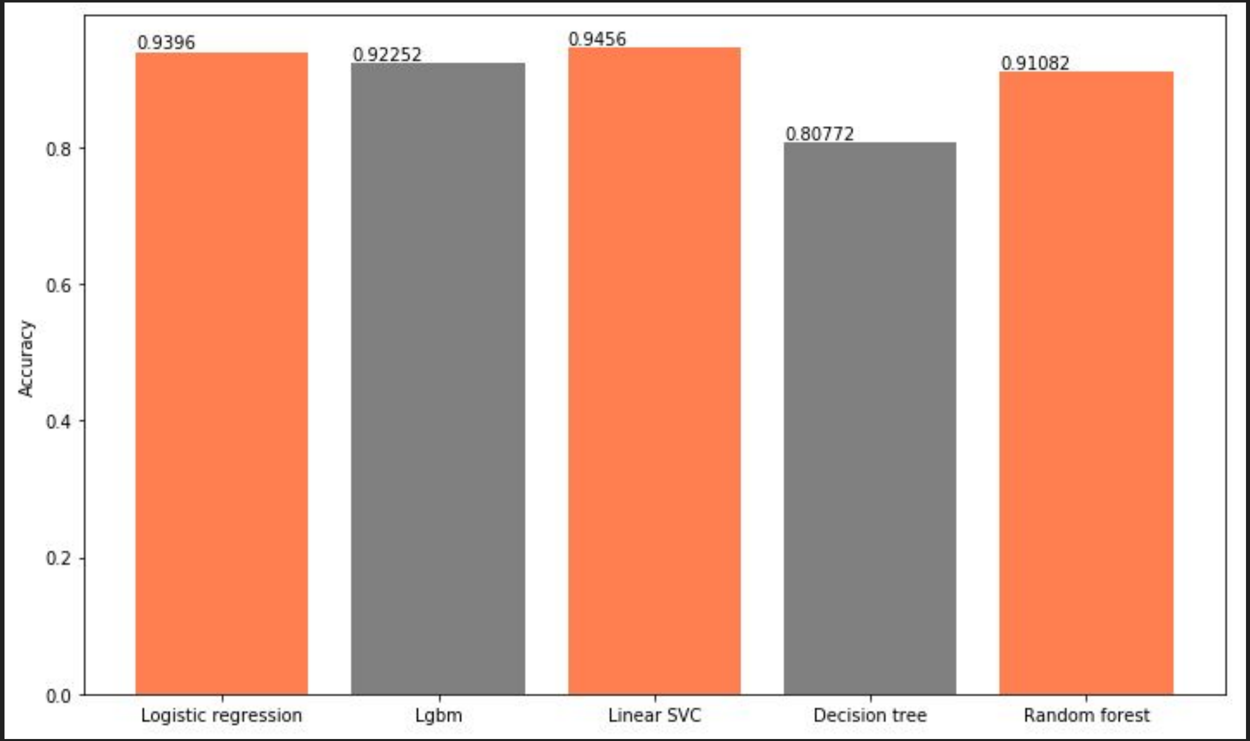
Accuracy :
0.8102466793168881

Random Forest classifier with hyperparameter Tuning



Accuracy :
0.9067046173308033

Comparison of models



The model that gives me the best accuracy is the linear SVC model, we can predict the activity (static or transition) of the person with an accuracy close to 95% (0.94719).

Serialising our Prediction Model

Serialisation is a method of converting python in-memory objects to a storage format that allows recovery of the python object's original structure from the stored format.

```
dump(linear_SVCm_rs, 'HumanActivitiesRecognition.joblib')
```

Joblib's dump method makes it easy to serialise a model

Test loading the model and using it

Joblib reconstructs a Python object from a file persisted

```
loaded_classifier = load('HumanActivitiesRecognition.joblib')

y_pred = loaded_classifier.predict(X_test)

loaded_classifier_accuracy = accuracy_score(y_true = y_test, y_pred = y_pred)

print("Accuracy using linear SVC : ",loaded_classifier_accuracy)
```

```
Accuracy using linear SVC :  0.9449715370018975
```

Above we can see that I can load and use the linear SVC model previously serialised

REST API with Python Django and Django REST Framework

I created a django API that uses my serialised model

To test the API :

- Go into the django directory (apirest/apirestFolder)
- Start the server with the command : **python3 manage.py runserver**
- Start the example script : **curl_predict_example.sh**

The script will send the json data to predict to the django API, this data corresponds to the activity of a participant. The API will return a json which contains the result of the prediction.