HUMAN ACTIVITY RECOGNITION

(MACHINE LEARNING USING PYTHON)

Project Report for the Industrial Training from Webtek Labs. Pvt. Ltd
June – July 2019

Under the Supervision of

MS. MOUSITA DHAR
(Project In-charge, Webtek Labs Pvt. Ltd)

Submitted By

Shashikant Shaw

4th year, 7th Semester, CSE
(Roll: 17600116022)

Of

Hooghly Engineering and Technology College

(Affiliated to Maulana Abul Azad University of Technology)

SUMMER INDUSTRIAL TRAINING
At
WEBTEK LABS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE AWARD OF THE DEGREE OF

B.TECH

(COMPUTER SCIENCE AND ENGINEERING)

Hooghly Engineering and Technology College

(Affiliated to Maulana Abul Azad University of Technology)

Submitted By

Shashikant Shaw

4th year, 7th Semester, CSE

(Roll: 17600116022)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CANDIDATE'S DECLARATION

I hereby declare that I have undertaken an industrial training at "WEBTEK LABS" during June - July 2019 in partial fulfilment of requirements for the award of degree of B.TECH (Computer Science and Engineering) at Hooghly Engineering & Technology College, Hooghly. The work which is being presented in the training report submitted to Department of COMPUTER SCIENCE and ENGINEERING is an authentic record of training work.

_____

Signature of the Student

_____

Signature of the Project Mentor

CERTIFICATE OF APPROVAL

The project "**Human Activity Recognition**" made by **Shashikant Shaw** is hereby approved as a creditable study for the Bachelor of Technology in COMPUTER SCIENCE and ENGINEERING and presented in a manner of satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned this project only for the purpose for which it is submitted.

MS. MOUSITA DHAR

(Project In-charge)

ACKNOWLEDGEMENT

I must begin with trainer, Ms Mousita Dhar who spent a lot of time reviewing my code to make it better result. They helped me a lot by giving new ideas to make my project better.

I would also like to thank my friends and classmates who helped me whenever I faced problems. They also tried their best to solve my problems and helped me with a constant support. I learnt a lot from this project and I will thank again to my project in-charge for giving me such a wonderful project. The project was something different from the book world, it was all about using those knowledge's to implement in the real world.

The experience I received during completing the project was really unfathomable. I thank the University of MAKAUT for giving us a scope for attaining this.

Thank you,

Regards,

Shashikant Shaw

1. INTRODUCTION

1.1 PYTHON

- Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

- PYTHON FEATURES

  - ✓ **Easy** to Learn and Use. Python is **easy** to learn and use
  - ✓ Expressive Language
  - ✓ Interpreted Language
  - ✓ Cross-**platform** Language
  - ✓ Free and Open Source
  - ✓ **Object-Oriented** Language
  - ✓ Extensible
  - ✓ Large Standard Library

- APPLICATIONS OF PYTHON

  - ✓ Web and internet development
  - ✓ Scientific and numeric computing
  - ✓ Data Analysis
  - ✓ Desktop GUIs
  - ✓ Machine Learning
  - ✓ Data visualization
  - ✓ Game Development
  - ✓ Software Development
  - ✓ Business Application

1.2 ANACONDA

Anaconda is a free and open distribution of Python programming languages for data science and machine learning related applications (large-scale data processing, predictive analytics, scientific computing), that aims to simplify package management and deployment. Package versions are managed by the package management system *conda*. Conda is an open

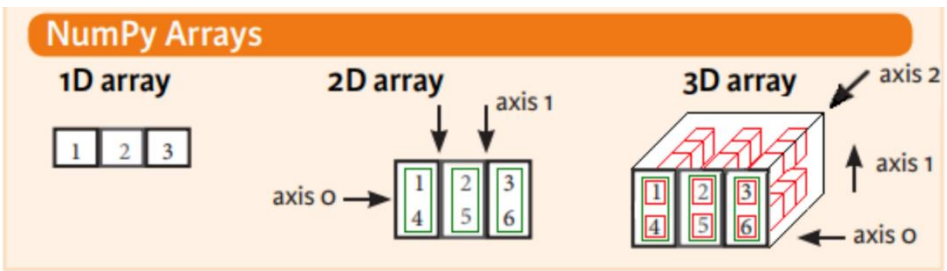
source, cross platform, language-agnostic package manager and environment management system that installs, runs, and updates packages and their dependencies. The Anaconda distribution is used by over 6 million users, and it includes more than 250 popular data science packages suitable for Windows, Linux, and MacOS.

1.3 PYTHON PACKAGES

Numpy

- ✓ NumPy is the fundamental package for scientific computing with Python. It contains among other things:
- ✓ a powerful N-dimensional array object
- ✓ sophisticated (broadcasting) functions
- ✓ tools for integrating C/C++ and Fortran code
- ✓ useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

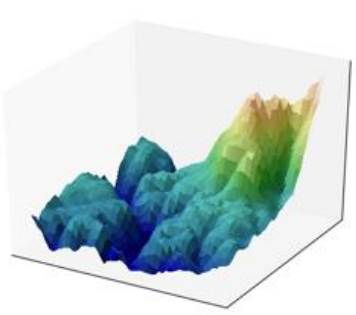


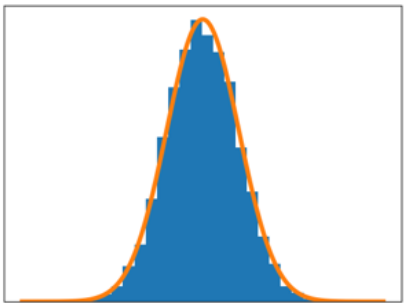
Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shell, the jupyter notebook, web application servers, and four graphical user interface toolkits.

Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatterplots, etc., with just a few lines of code. For simple plotting the pyplot module provides a MATLAB-like interface, particularly when

Combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc., via an object oriented interface or via a set of functions familiar to MATLAB users.





Pandas

Pandas is an open source, BSD-licensed library providing high-performance, easy- to-use data structures and data analysis tools for the *Python* programming language. Pandas library is well suited for data manipulation and analysis using python. In particular, it offers data structures and operations for manipulating numerical tables and time series.

Scikit-learn

Scikit-learn provides machine learning libraries for python some of the features of Scikit- learn includes:

- ✓ Simple and efficient tools for data mining and data analysis
- ✓ Accessible to everybody, and reusable in various contexts
- ✓ Built on NumPy, SciPy, and matplotlib
- ✓ Open source, commercially usable - BSD license

TRAINING WORK UNDERTAKEN

COLLECTING DATA FROM KAGGLE

Kaggle is a platform for predictive modelling and analytics competitions in which statisticians and data miners compete to produce the best models for predicting and describing the datasets uploaded by companies and users. This crowd sourcing approach relies on the fact that there are countless strategies that can be applied to any predictive modelling task and it is impossible to know beforehand which technique or analyst will be most effective.
On 8 March 2017, Google announced that they were acquiring Kaggle. They will join the Google Cloud team and continue to be a distinct brand. In January 2018, Booz Allen and Kaggle launched Data Science Bowl, a machine learning competition to analyse cell images and identify nuclei.

DATA SCIENCE

Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from data in various forms, both structured and unstructured, similar to data mining. Data science is a "concept to unify statistics, data analysis, machine learning and their related methods" in order to "understand and analyse actual phenomena" with data. It employs techniques and theories drawn from many fields within the context of mathematics, statistics, information science, and computer science.
Turing award winner Jim Grey imagined data science as a "fourth paradigm" of science (empirical, theoretical, computational and now data-driven) and asserted that "everything about science is changing because of the impact of information technology" and the data deluge.
When Harvard Business Review called it "The Sexiest Job of the 21st Century “the term became a buzzword, and is now often applied to business analytics, business intelligence, predictive modelling, or any arbitrary use of data, or used as a glamorized term for statistics. In many cases, earlier approaches and solutions are now simply rebranded as

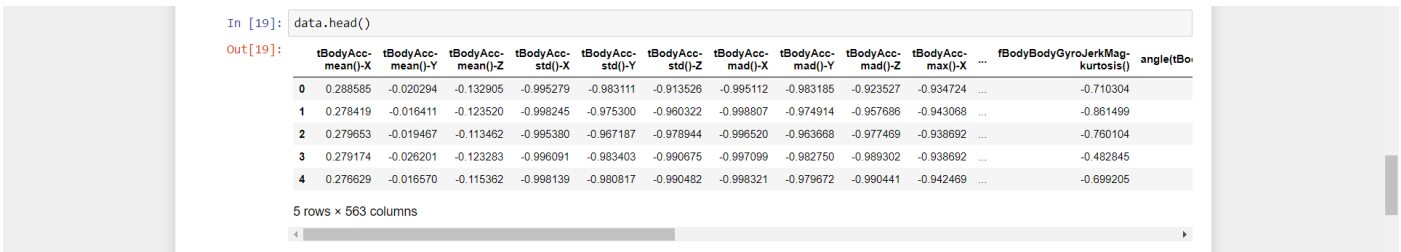
"Data science" to be more attractive, which can cause the term to become "dilute[d] beyond usefulness." While many university programs now offer a data science degree, there exists no consensus on a definition or suitable curriculum contents. Because of the current popularity
Of this term, there are many "advocacy efforts" surrounding the field. To its discredit, however, many data science and big data projects fail to deliver useful results, often as a result of poor management and utilization of resources.

PROJECT IN BRIEF:

The project is about predicting the human activity from the dataset collected from kaggle, the dataset has data recorded from the smartphone sensors and human activity performed respectively. This project involves in building a Machine Learning model which will be capable of predicting the human activity on the basis of the recorded data from the sensors. Human activity involves, laying, sitting, standing, walking, walking upstairs, and walking downstairs.

DATASET

The dataset contains 7352 rows and 563 columns.



```
In [19]: data.head()
```

Out[19]:

	tBodyAcc-mean()-X	tBodyAcc-mean()-Y	tBodyAcc-mean()-Z	tBodyAcc-std()-X	tBodyAcc-std()-Y	tBodyAcc-std()-Z	tBodyAcc-mad()-X	tBodyAcc-mad()-Y	tBodyAcc-mad()-Z	tBodyAcc-max()-X	...	fBodyBodyGyroJerkMag-kurtosis()	angle(tBo
0	0.288585	-0.020294	-0.132905	-0.995279	-0.983111	-0.913526	-0.995112	-0.983185	-0.923527	-0.934724	...	-0.710304	
1	0.278419	-0.016411	-0.123520	-0.998245	-0.975300	-0.960322	-0.998807	-0.974914	-0.957686	-0.943068	...	-0.861499	
2	0.279653	-0.019467	-0.113462	-0.995380	-0.967187	-0.978944	-0.996520	-0.963668	-0.977469	-0.938692	...	-0.760104	
3	0.279174	-0.026201	-0.123283	-0.996091	-0.983403	-0.990675	-0.997099	-0.982750	-0.989302	-0.938692	...	-0.482845	
4	0.276629	-0.016570	-0.115362	-0.998139	-0.980817	-0.990482	-0.998321	-0.979672	-0.990441	-0.942469	...	-0.699205	

5 rows × 563 columns

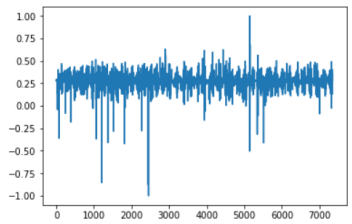
Steps Involved:

1. Import required modules.
2. Load the dataset in a data frame
3. Visualize the data.
4. Shape the data so that it fits in the algorithm.
5. Separate the data into test and train.
6. Apply the algorithm.
7. Apply the test data to check the result.
8. Check for accuracy.

Data Visualization

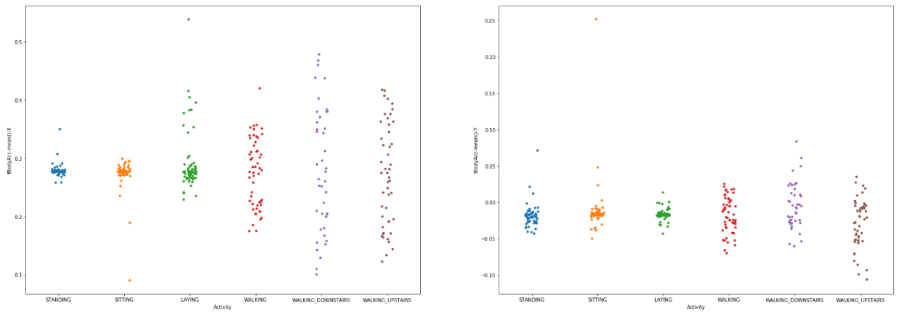
Visualizing all the sensors data recorded in the dataset by plotting the graph.

```
In [4]: sensor=tt_df.iloc[:,0]
        time=np.arange(0,len(sensor),1)
        # import matplotlib.pyplot as plt

        plt.plot(time, sensor)

Out[4]: [<matplotlib.lines.Line2D at 0x7fa519830550>]
```



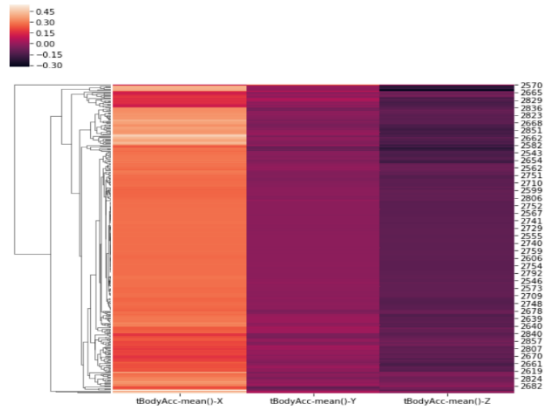
It is good that the data is almost evenly distributed for all the activities among all the subjects. Let's pick subject 15 and compare the activities with the first three variables - mean body acceleration in 3 spatial dimensions.

```
In [9]: sub15 = data.loc[data['subject']==15]
        fig = plt.figure(figsize=(32,24))
        ax1 = fig.add_subplot(221)
        ax1 = sns.stripplot(x='Activity', y=sub15.iloc[:,0], data=sub15, jitter=True)
        ax2 = fig.add_subplot(222)
        ax2 = sns.stripplot(x='Activity', y=sub15.iloc[:,1], data=sub15, jitter=True)
        plt.show()
```



So, the mean body acceleration is more variable for walking activities than for passive ones especially in the X direction. Let's create a dendrogram and see if we can discover any structure with mean body acceleration.

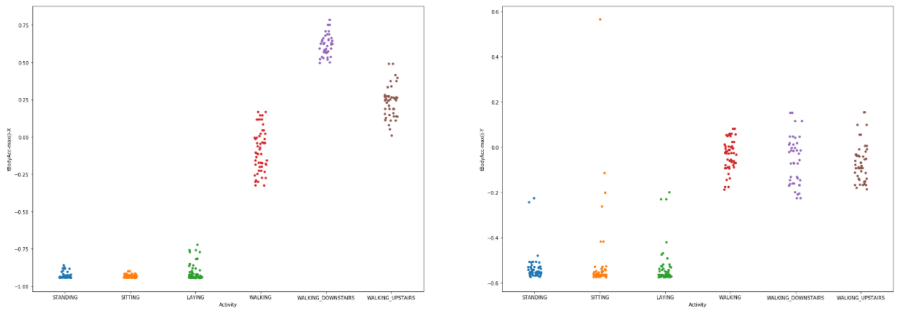
```
In [11]: sns.clustermap(sub15.iloc[:,[0,1,2]], col_cluster=False)

Out[11]: <seaborn.matrix.ClusterGrid at 0x7fa516247f28>
```



Even though we see some dark spots in the X and Z directions (possibly from the walking activities), the bulk of the map is pretty homogenous and does not help much. Perhaps other attributes like maximum or minimum acceleration might give us a better insight than the average.

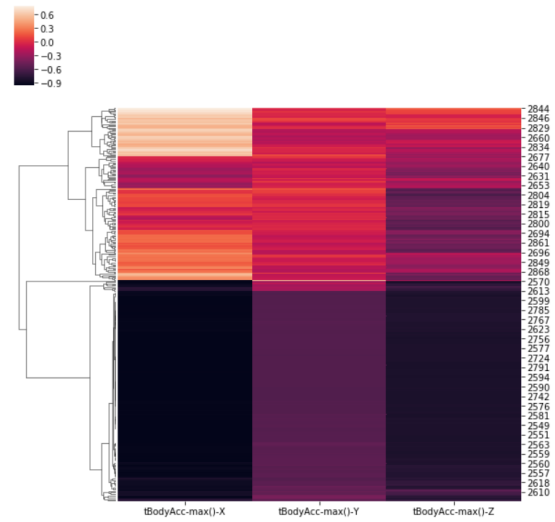
Plotting maximum acceleration with activity.

```
In [12]: fig = plt.figure(figsize=(32,24))
         ax1 = fig.add_subplot(221)
         ax1 = sns.stripplot(x='Activity', y='tBodyAcc-max()-X', data=sub15, jitter=True)
         ax2 = fig.add_subplot(222)
         ax2 = sns.stripplot(x='Activity', y='tBodyAcc-max()-Y', data=sub15, jitter=True)
         plt.show()
```



Passive activities fall mostly below the active ones. It actually makes sense that maximum acceleration is higher during the walking activities. Let's again plot the cluster map but this time with maximum acceleration. Notice the walk down activity is above all others in the X-direction recording values between 0.5 and 0.8.

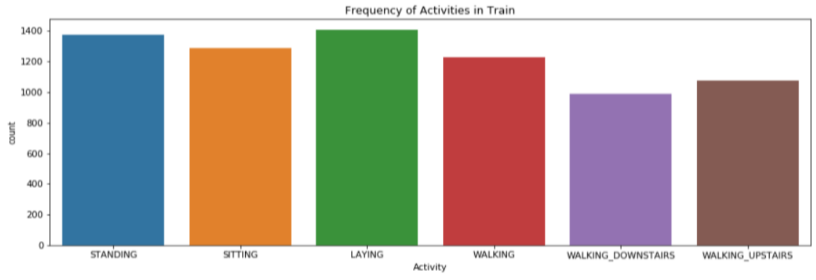
```
In [14]: sns.clustermap(sub15[['tBodyAcc-max()-X', 'tBodyAcc-max()-Y', 'tBodyAcc-max()-Z']], col_cluster=False)
Out[14]: <seaborn.matrix.ClusterGrid at 0x7fa50c6ded68>
```



We can now see the difference in the distribution between the active and passive activities with the walk down activity (values between 0.5 and 0.8) clearly distinct from all others especially in the X-direction. The passive activities are indistinguishable and present no clear pattern in any direction (X, Y, and Z).

Plotting the frequencies of the Activities

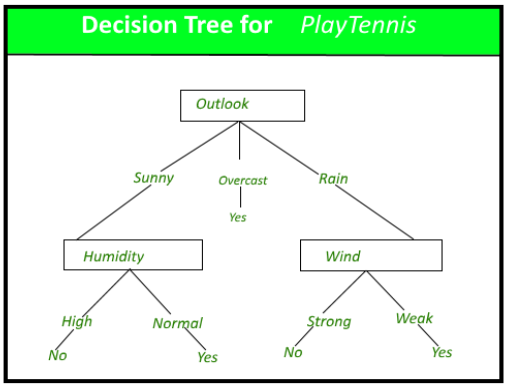
```
In [19]: plt.figure(figsize = (15,5))
         sns.countplot(data.Activity)
         plt.xlabel('Activity')
         plt.ylabel('count')
         plt.title('Frequency of Activities in Train')
         plt.show()
```



Algorithms used:

Decision Tree Classifier

Decision Tree: Decision tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.



A decision Tree concept for playing Tennis

Construction of Decision Tree:

A tree can be *"learned"* by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called *recursive partitioning*. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions. The construction of decision tree classifier does not

require any domain knowledge or parameter setting, and therefore is appropriate for exploratory knowledge discovery. Decision trees can handle high dimensional data. In general decision tree classifier has good accuracy. Decision tree induction is a typical inductive approach to learn knowledge on classification.

Decision Tree Representation:

Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, and then moving down the tree branch corresponding to the value of the attribute as shown in the above figure. This process is then repeated for the sub tree rooted at the new nodes.

Machine Learning Model with an accuracy score:

```
In [25]: #Applying the DecisionTreeClassifier Algorithm in the dataset.

In [26]: l=[]
X_train, X_test, y_train, y_test = train_test_split( X, Y, test_size = 0.1, random_state = 95)
clf_entropy = DecisionTreeClassifier(criterion="entropy",min_samples_split=.05)
clf_entropy.fit(X_train, y_train)
y_pred_en = clf_entropy.predict(X_test)
#accuracy_score(y_test,y_pred_en)
l.append(accuracy_score(y_test,y_pred_en))

In [27]: #printing the accuracy score
import math
print(math.floor(l[0]*100))

92
```

Confusion Matrix and Classification Report:

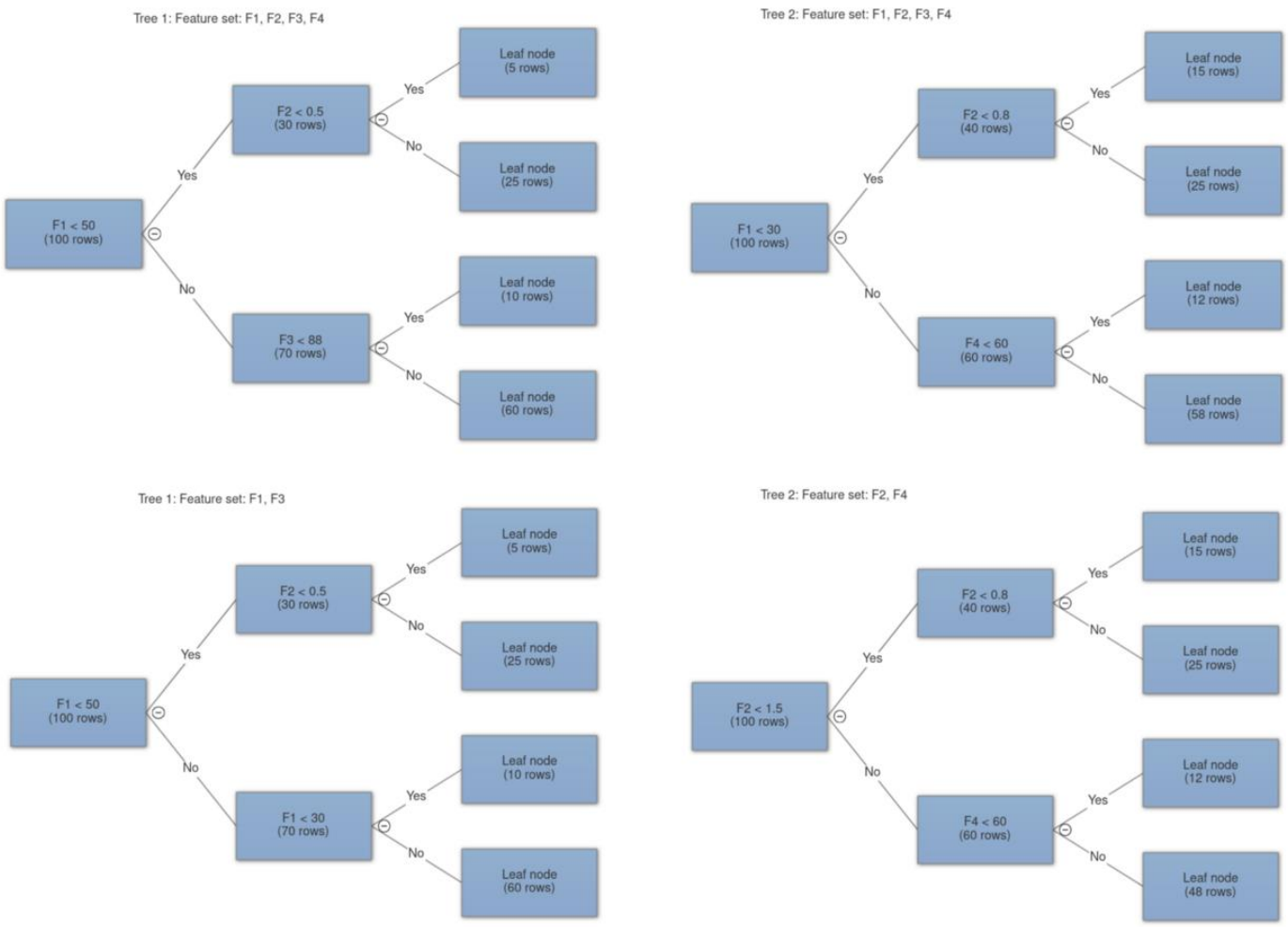
```
In [33]: from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

results = confusion_matrix(y_test, y_pred_en)
print (results)

[[141   0   0   0   0   0]
 [  0 125  18   0   0   0]
 [  0   2 123   0   0   0]
 [  0   0   0  98   3   6]
 [  0   0   0   8  83   6]
 [  0   0   1   7   3 112]]

In [31]: print (classification_report(y_test, y_pred_en))

                    precision    recall  f1-score   support

             LAYING       1.00      1.00      1.00       141
           SITTING       0.98      0.87      0.93       143
          STANDING       0.87      0.98      0.92       125
           WALKING       0.87      0.92      0.89       107
WALKING_DOWNSTAIRS       0.93      0.86      0.89        97
  WALKING_UPSTAIRS       0.90      0.91      0.91       123

         micro avg       0.93      0.93      0.93       736
         macro avg       0.93      0.92      0.92       736
      weighted avg       0.93      0.93      0.93       736
```

Random Forest Classifier

Random forest is the prime example of ensemble machine learning method. In simple words, an ensemble method is a way to aggregate less predictive base models to produce a better predictive model. Random forests, as one could intuitively guess, ensembles various decision trees to produce a more generalized model by reducing the notorious over-fitting tendency of decision trees. Both decision trees and random forests can be used for regression as well as classification problems.

Feature bagging: **b**ootstrap **agg**regating or bagging is a method of selecting a random number of samples from the original set with replacement. In feature bagging the original feature set is randomly sampled and passed onto different trees (without replacement since having redundant features makes no sense). This is done to decrease the correlation among trees. A feature with unmatched great importance will cause every decision tree to choose it for the first and possible consequent splits, this will make all the trees behave similarly and ultimately more correlated which is undesirable.

Aggregation: The core concept that makes random forests better than decision trees is aggregating uncorrelated trees. The idea is to create several crappy model trees (low depth) and average them out to create a better random forest. Mean of some random errors is zero hence we can expect generalized predictive results from our forest. In case of regression we can average out the prediction of each tree (mean) while in case of classification problems we can simply take the majority of the class voted by each tree (mode).



Machine Learning Model with an accuracy score:

```
In [36]: #applying the RandomForestClassifier Algorithm in the dataset.

In [42]: from sklearn.ensemble import RandomForestClassifier

In [43]: rf=RandomForestClassifier(n_estimators=20,random_state=31,min_samples_split=.05)

In [44]: rf.fit(X_train,y_train)

Out[44]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                     max_depth=None, max_features='auto', max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=0.05,
                     min_weight_fraction_leaf=0.0, n_estimators=20, n_jobs=None,
                     oob_score=False, random_state=31, verbose=0, warm_start=False)

In [45]: y_pred=rf.predict(X_test)

In [46]: print(accuracy_score(y_test,y_pred))
         0.9442934782608695
```

Confusion Matrix and Classification Report:

```
In [38]: from sklearn.metrics import confusion_matrix
         from sklearn.metrics import accuracy_score
         from sklearn.metrics import classification_report

         results = confusion_matrix(y_test, y_pred)
         print (results)

         [[141   0   0   0   0   0]
          [  0 133  10   0   0   0]
          [  0  13 112   0   0   0]
          [  0   0   0 103   1   3]
          [  0   0   0   4  89   4]
          [  0   0   0   3   3 117]]

In [40]: print (classification_report(y_test, y_pred))

                             precision    recall  f1-score   support

                     LAYING       1.00      1.00      1.00       141
                    SITTING       0.91      0.93      0.92       143
                   STANDING       0.92      0.90      0.91       125
                    WALKING       0.94      0.96      0.95       107
         WALKING_DOWNSTAIRS       0.96      0.92      0.94        97
           WALKING_UPSTAIRS       0.94      0.95      0.95       123

                  micro avg       0.94      0.94      0.94       736
                  macro avg       0.94      0.94      0.94       736
               weighted avg       0.94      0.94      0.94       736
```

RESULTS AND DISCUSSION

➢ Result

- Machine learning model gives an accuracy score of 92% approximately when applying the Decision Tree Classifier algorithm.
- Machine learning model gives an accuracy score of 94% approximately when applying the Random Forest Classifier algorithm.

➢ Discussion

Machine Learning has evolved in almost every sectors, starting from IT Industries to Medical, to provide new features and allow humans to put less effort. Machine learning models built are giving exceptional results. My dataset consisted of the sensors data recorded by the smartphone and the user activity. I built the model which is able to predict the human activity by providing the input data (recorded by smartphone) with an accuracy of 94%.

CONCLUSION

Based on the dataset, I applied Decision Tree Classifier and Random Forest on the dataset. Both the algorithms worked approximately similar with an absolute difference of 2% in the accuracy score. Decision Tree predicted the test data with an accuracy score 92% whereas Random Forest predicted the test data with an accuracy score 94%.

REFERENCES

- https://www.python.org/
- https://anaconda.org/anaconda/python
- http://www.numpy.org/
- https://matplotlib.org/
- http://scikit-learn.org/
- https://pandas.pydata.org/
- https://pandas.pydata.org/
- https://ipython.org/