**TITLE**

**ABSTRACT**

**ACKNOWLEDGEMENTS**

**TABLE OF CONTENTS**

**LIST OF TABLES AND FIGURES**

**CHAPTER 1: Introduction**

**Section 1:1 Introduction to the problem**

Based on different GPX files uploaded by different users round the globe, we are trying to estimate or build up a model so that it can empower us to predict sports activities in different geo-tracks. Lot of IOT devices like smart phones, smart watches are being regularly used to capture data while performing these kinds of activities. Fast life and high expectancy of life made us serious about getting involved in fitness programs. This project is an attempt to not only predict activities for a certain geo track but also plotting an association between similar tracks. According to specific fitness goal, one can choose similar tracks, available as per his choice of city or country. As we tried to cluster tracks as per the difficulty level, the project will be helpful to choose other tracks to increase the fitness level as per the selected sport activity.

[[implement different clustering methods and find efficiency.

Predictive analysis of different sports activities. (academic paper)]]

**Section 1:2 Project Objectives**

two or three specific objectives

1. Grouping GPX files as per their characteristics and prediction of activity performed to generate the file.
2. Building a web platform to enable users to choose tracks as per their fitness program. For a certain activity, the tracks will be grouped as per different level of competency levels. Users will be able to choose lower or higher or same level of tracks. The selection can be made as per their neighbourhood or city or any country basis.
3. The project may help various government bodies to set their priorities to sanction or complete projects in building tracks suitable for sports like running, hiking, cycling etc. If they can estimate the utilization of a prospective route and predict the kind of activities will be performed on them, will certainly help the organizations to plan for cost estimation and feature build up.

**Section 1:3 Overall Solution**

Not detailed specification of methodology

When we are done with

**Section 1.4 Outline of the thesis**

**CHAPTER 2: Background and Related Work**

**Section 2:1 Related Literature and Other Works**

**Youtube videos: gist**

**Section 2:2 Outline of contributions**

**CHAPTER 3: Design and Solution Overview**

To be completed now

Data Exploration:

Prior to come up with a solution, it is essential to list down what kind of information we are accumulating from the GPX files. Each of the GPX files consist of a series of nodes and each of the node holds four types of data. Timestamp, altitude, longitude & latitude. The combination of latitude and longitude pin points the location whereas the altitude states about the elevation details. The timestamp field provides information related to date and time. The starting node usually signifies the starting point of any journey. Whereas the last point usually lets us know about the end point of the journey. Having said that, we will consider the start and end points based upon time fields available within the file. Only these four fields are not good enough to find out the required clustering among different routes based upon the performed activity. We will make an attempt to derive some more fields from these original fields. The main objective to find out these derived fields is that it will provide us better understandability about the difficulty of a certain track. For an example, if there is a sharp inclination in the way then it is likely that a hiker or runner will face challenges and it will result in decrease of his/her speed. To figure out the change in elevation between successive nodes, we will plot the difference in ‘DeltaElev’ field. One interesting point is that the gadget being used, does not capture on regular intervals. And the speed of the person varies at different point of time as well. So, we would like to calculate the time differences between each two nodes in ‘TimeDiff’ field. From the combination of latitude and longitude fields, we get to realize the ‘Point’ field. From the ‘point’ field, we can compute the distance crossed between each successive node and the result is inserted in ‘GeoPointsDist’ field. One of the most significant things is to know how straight a path is. More bend found in a track, more likely it causes decrease in speed and less distance is covered as well. For activities like cycling, motor-cycling are very much impacted by the change of this field, ‘Angle’. For the start and end node we considered the angle as zero. To calculate the angle of any given point, we collected it’s prior and subsequent node for reference. If it is evident that all the three points are in a straight line, then also we consider its angle as zero.

**[For the time being: Data Manipulation is written here.]**

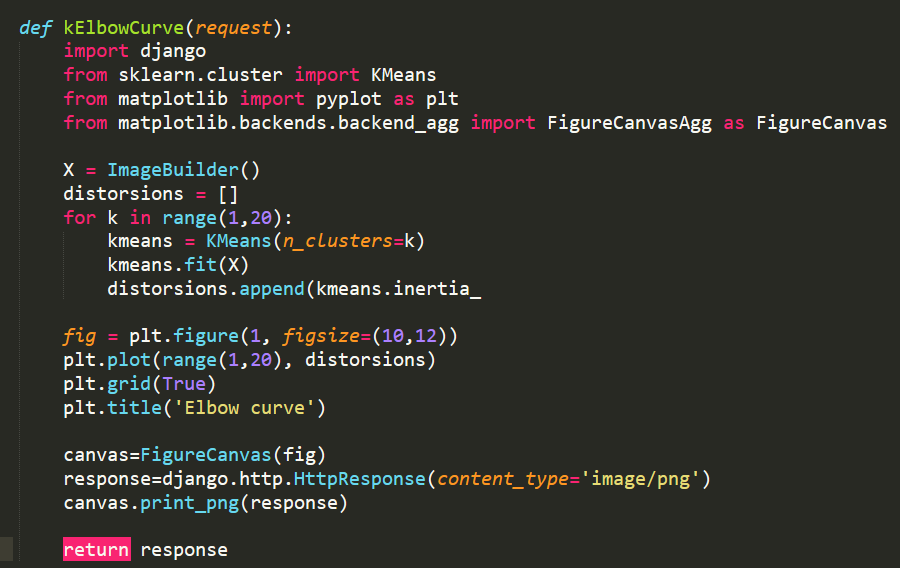
Each of the result set is grouped with the respective filename and summary statistics is prepared for each of the files that are analysed during the process. Summary statistics provides us 6 parameters of data. They are min value, 1st quartile, median, mean, 3rd quartile, and maximum value. As our approach is to automatically form dataset so that it becomes suitable for clustering algorithm, we consider different approaches for different variables. It is to be specified that we will be performing clustering based on six fields. They are altitude, time, speed, timediff, DeltaElev, GeoPointDistance, Angle. We considered the median value for speed, timediff, and Angle field. But for altitude, DeltaElev, and time the different between maximum and minimum are evaluated. GeoPointDistance is measured as the summation of the total distance covered from the starting point to the end.

The relevant code snippet is attached below:

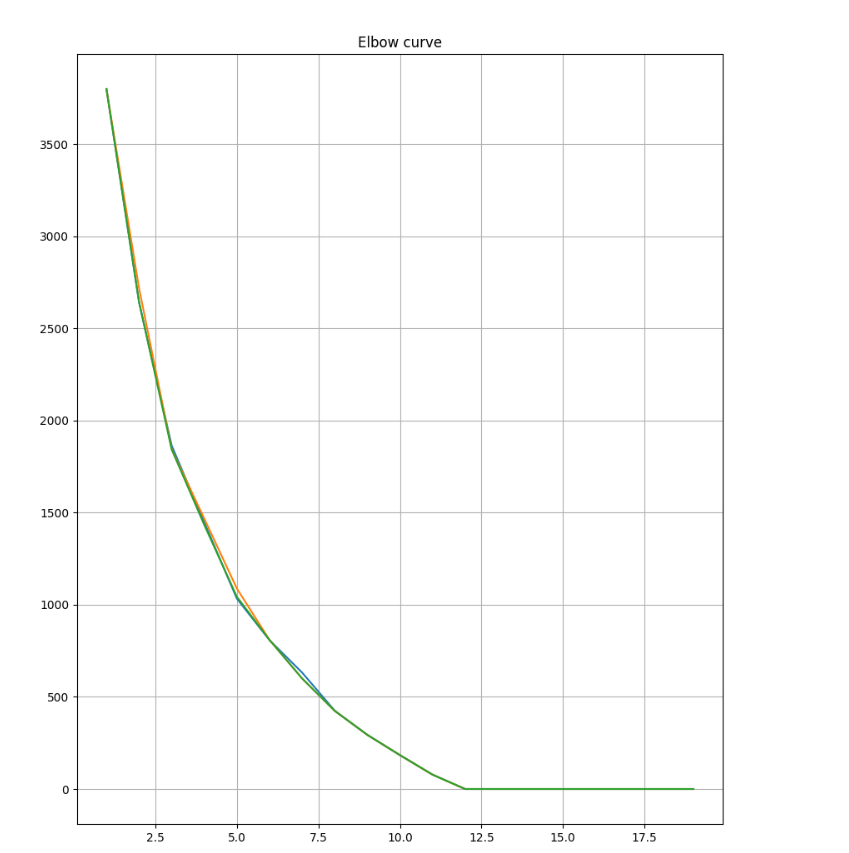


Based upon the mentioned fields value a new dataset is created, which will be used in different machine learning algorithm. In our project, we used KMeans algorithm to find out the similarity within different routes and form clusters with more closely related paths.

In Kmeans algorithm, it is very important to find out how many clusters we need to form based upon a given dataset. As per the unsupervised machine learning guideline, we created ‘Elbow’ curve in python programming to find out that significant number. The code snippet is given as:

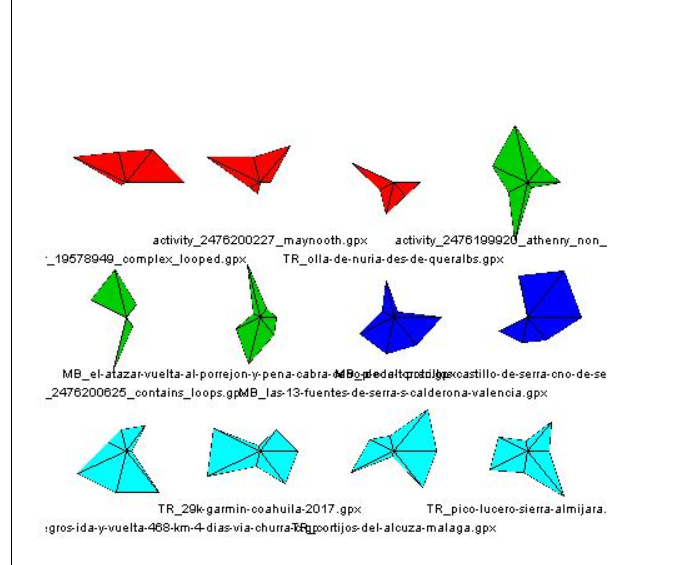


A sample output that got generated from the test dataset is given here:



As per the graph plotted, after 4th directional change (consider the movement towards the +ve x-axis) in the line, it became parallel to x-axis. So, in this instance maximum four clusters can be organized, not more than that.

In R code, when we run the Kmeans algorithm the generated output turned to be something similar like the one attached below:

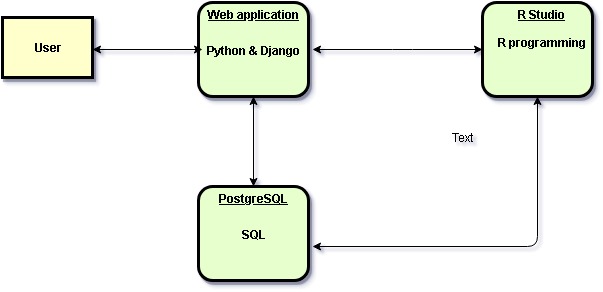


Star diagrams are plotted from the multivariate dataset and each of the colour signifies the close match between given files. File names are mentioned below the stars and it is helpful to figure out which files are closely related from the output picture.

**Application architecture:**

Our prime focus was to create a web application, so that users can visualize their GPX file characteristics, specially the route in a form of a map and choose similar routes as per their fitness program. A basic option is provided to upload all training data from the local directory to the application domain. The main 3 platforms, getting used in the application are 1) python programming with Django web framework, 2) PostgreSQL, and 3) R programming. Having said that, some other technologies or conventions like REST-API, Ajax, Javascript, Jquery library, Bootstrap library etc played a significant role in making our project a success. They will also be discussed as per the need.

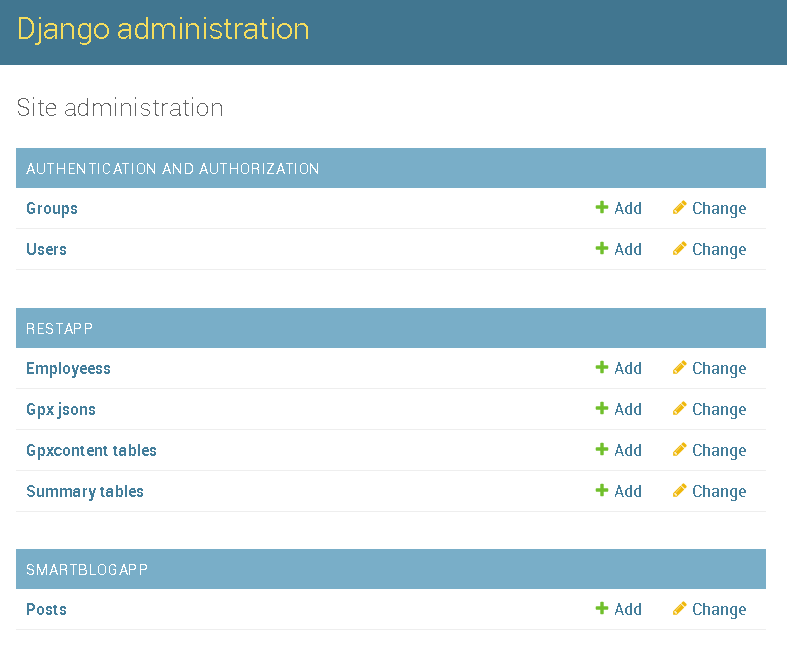
Figure:



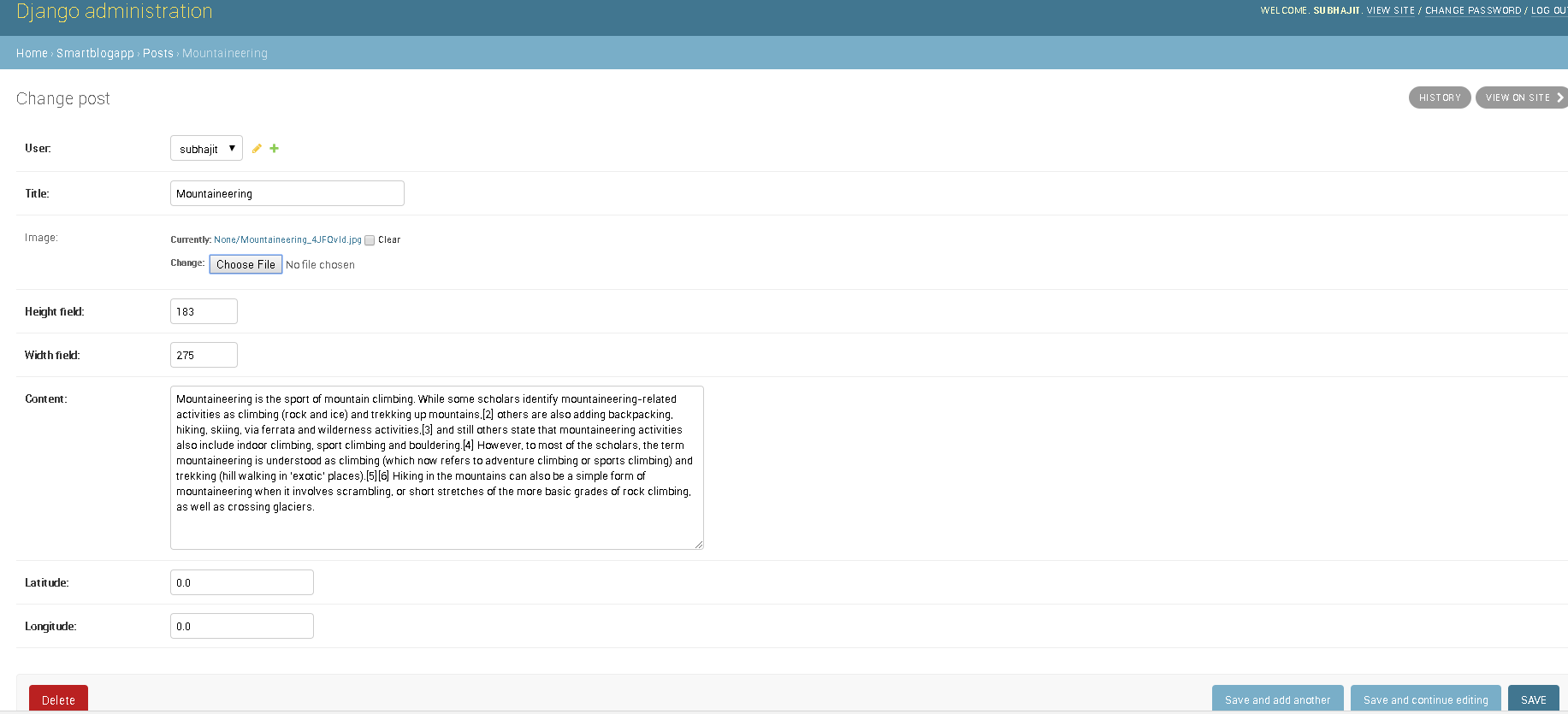
A web application is created with python programming and Django web framework. A user interacts with the application through a user interface. The web application follows client server architecture and as per need interacts with database (PostgreSQL) and executes R script. The database is maintained mainly for data store and retrieval purpose though some calculation part occurred in PostgreSQL as well (it will be detailed in the PostgreSQL section). Whereas the created R script, is triggered (for the time being the triggering is manual) from the Django framework when all the GPX files content is loaded in the PostgreSQL tables and there is a need to prepare dataset for machine learning algorithm execution. After R script execution, user will be able to load PostgreSQL respective tables, from which results will be shown up in the UI (user interface).

Web Application:

We created a blog application, where users will be able to upload their GPX files as per their performed activity and will be able to find out association between different tracts as per different activities. For the time being, we considered that the files get generated from one person. Later point of time data model can be customized to store records for different individuals. For the sake of application management and maintaining application model hierarchy, we created 3 apps within the same project. They are a) admin, b) smartblog-app and c) rest-app. The admin section is good enough to create a user and to provide the basic authentication and authorization for different users.



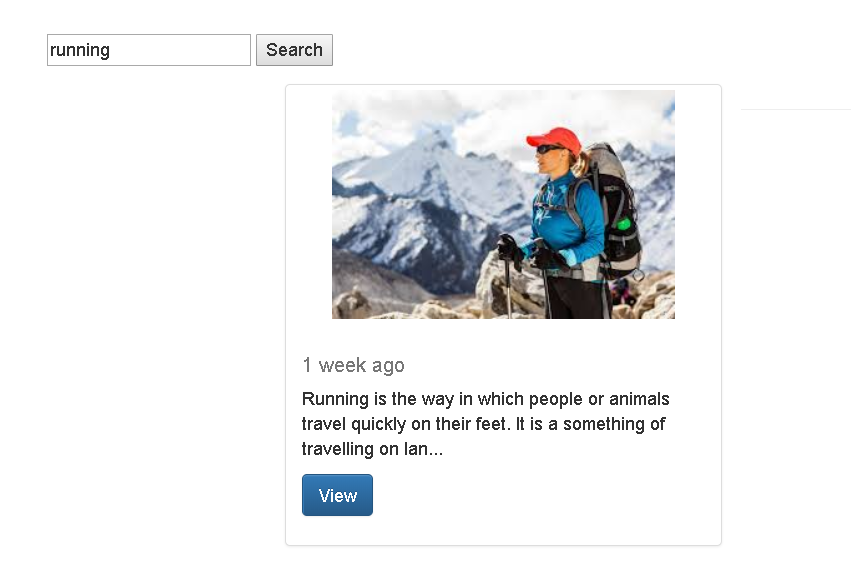
From the admin section, it will always be possible to control the entities listed within the mention applications. For an example running related information can be customized or created from the admin section itself.



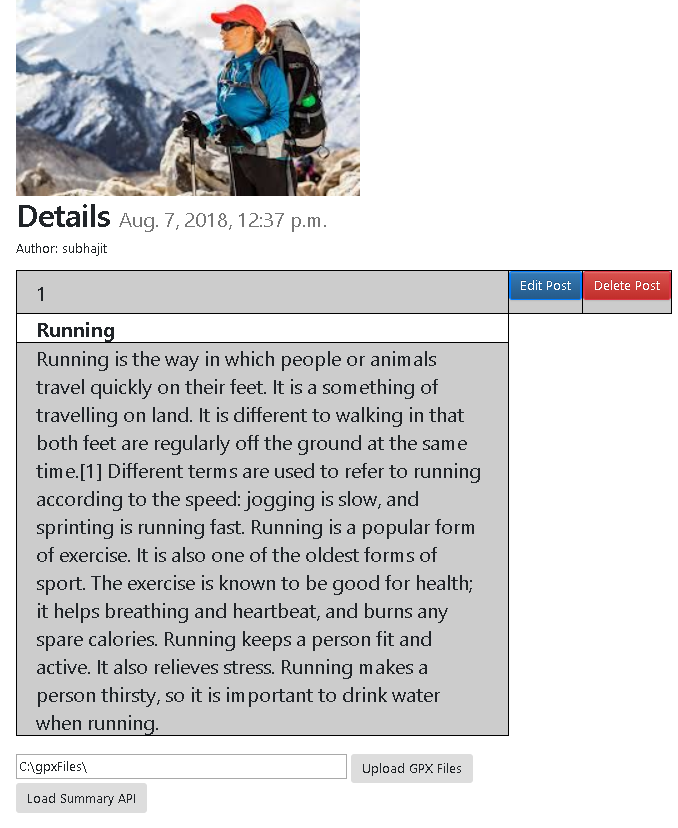
Similarly, the entities created in the rest-app can be altered from the admin section by the same process. But for each of the cases creation and alteration are manual.

This first section will also help the administrator control the main application and the endpoints that are created for REST-APIs.

The second section is the main application which holds the activity name, its description, and the creation time details. User can upload a picture relevant to the performed activity as well. The landing page looks similar to this:

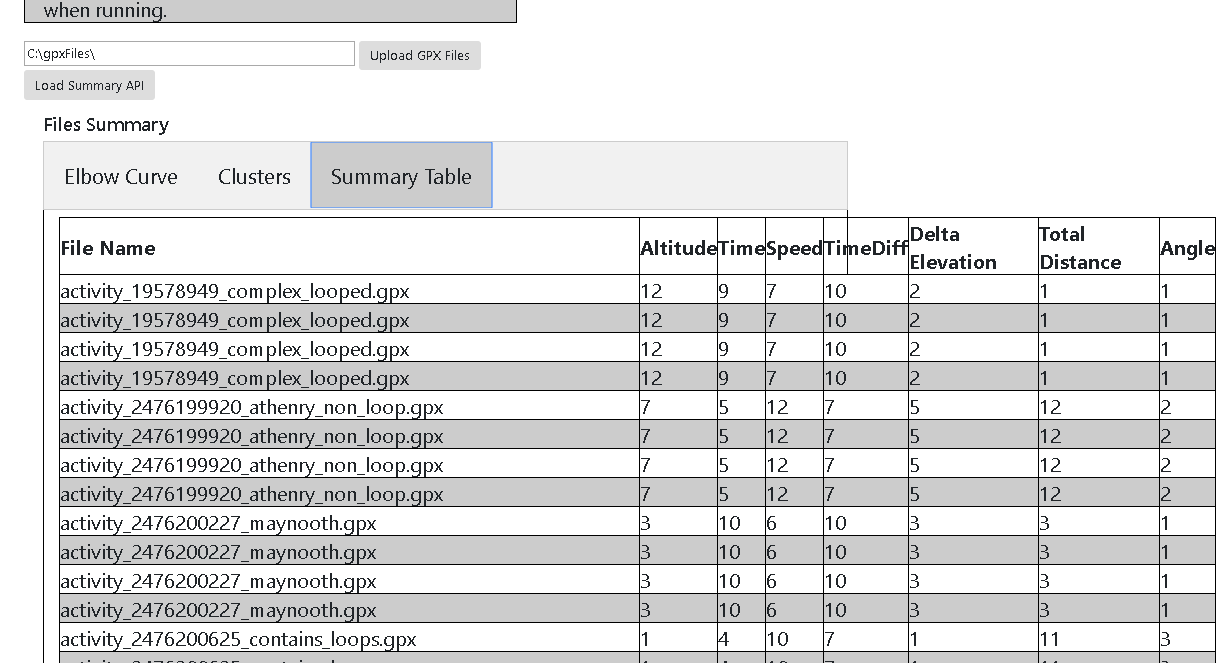


As per the entries made to the system, we can search an activity and navigate to its details page by pressing view button. The detail page will look like this:

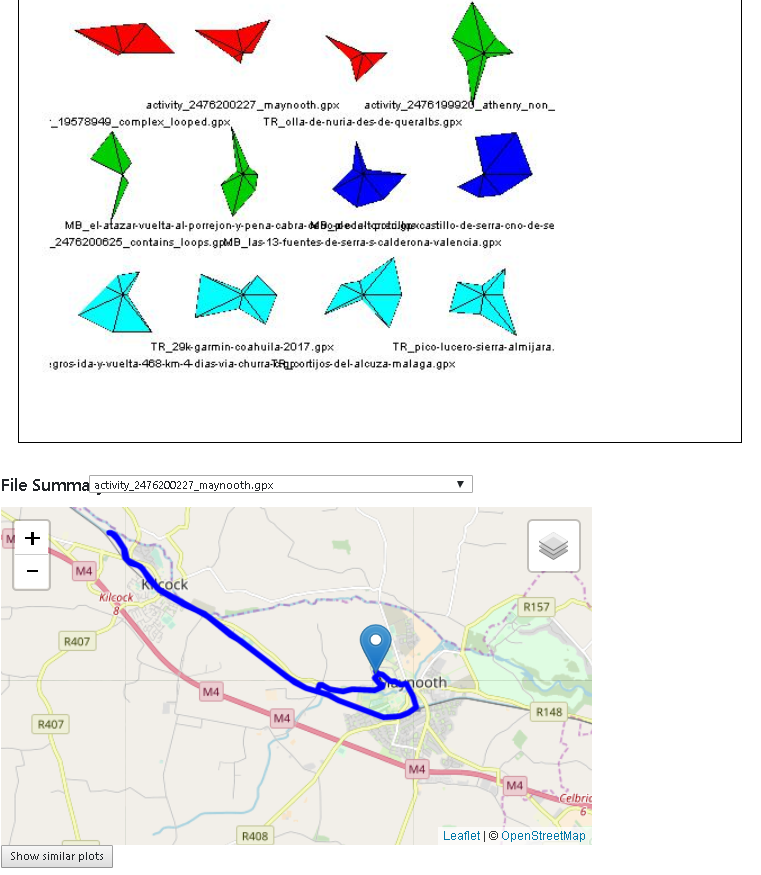


Along with edit and delete options, user will be asked to upload gpx files through ‘Upload GPX Files’ button. Please note the files should be placed under ‘c:\gpxfiles’ directory. Actually this button uploads all the content of the gpx files to gpxcontentable in PostgreSQL for further process. Now when the upload will be completed, the system will ask the user to manually run the designed R script (this step could have been automated, but faced challenges, which is described in section). After R script execution user is required to enable the all configured REST-APIs within the application to populate data as per R script output. The ‘Load summary API’ button will exactly do the same. Based upon the output generated from the R script, results are shown in different tabs.

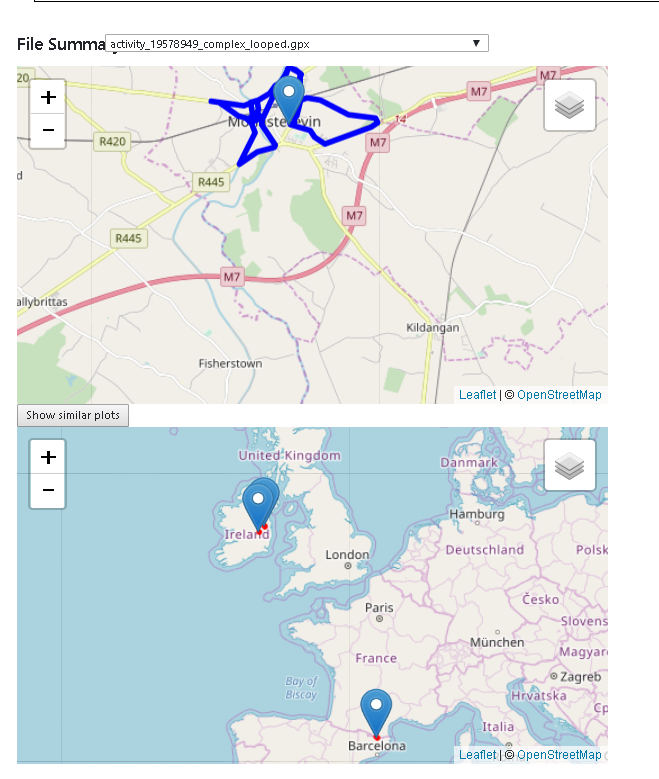




User will be able to visualize characteristics of different files find out similar tracks by colour matching star figures. All the file options will be made available to the user in a dropdown control. On selection of a track user will be able to find out similar tracks by triggering ‘show similar plots’ button.



For an example, if user selects the file ‘activity\_2476200227\_maynooth.gpx’, two other files (marked in red) should be displayed as in total there are 3 red marked files present in the star plot.



As per the test data, only 3 similar files are shown in the similar plot. In this way the application helps the user to find out same kind of tracks found across different geographic location to achieve their fitness goal.

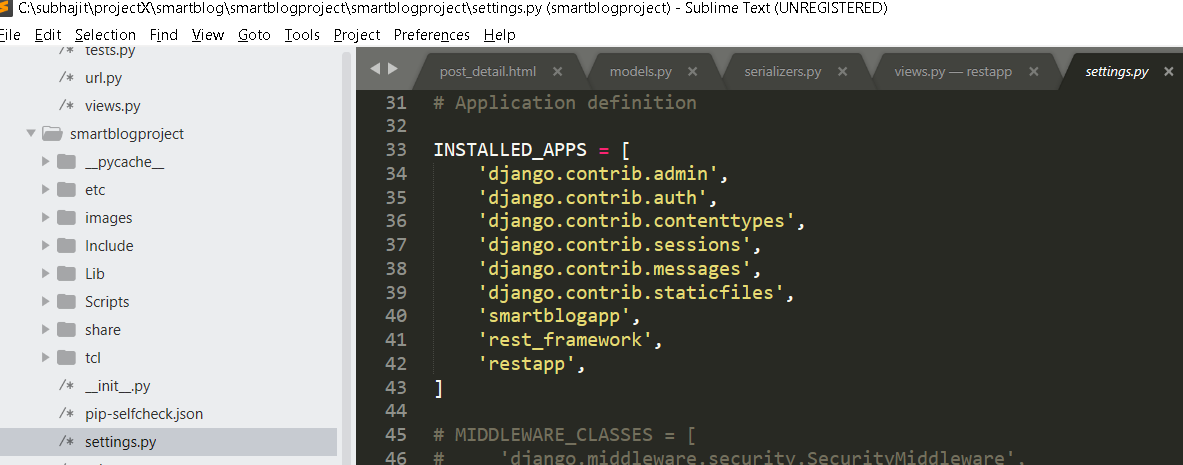
User will be able to store the relevant data and execute machine learning algorithm as per different activities. The above example was regarding ‘running’ activity and the relevant files were process.

The last part of the web application is REST-APP. This app acts as a service and hosts couple of endpoints which follow REST protocol and whichever application consume this service will fetch response in JSON format. The REST-APP fetches data from couple of tables in PostgreSQL and expose the data as in json response. In the main application ‘smartblogapp’, javascript code is written to make some AJAX calls from the client side and to consume the json response from the REST-API created.

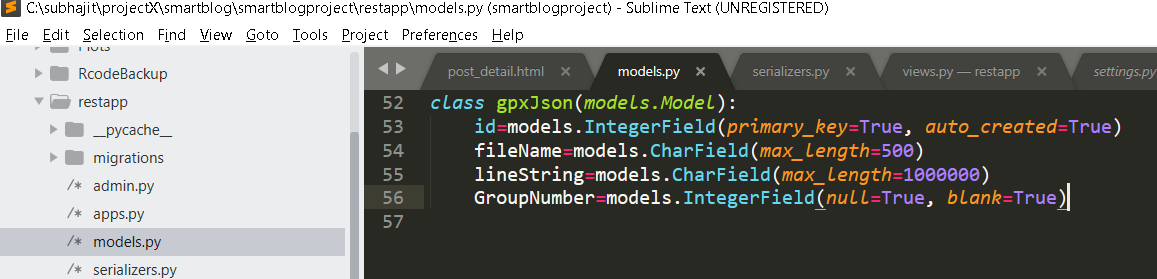
Creating a REST-API:

There are couple of steps required to create a REST-API in Django framework. They are:

1. Djangorestframework should be installed using command prompt and ‘rest\_framework’ should be referred as installed app in the settings file.

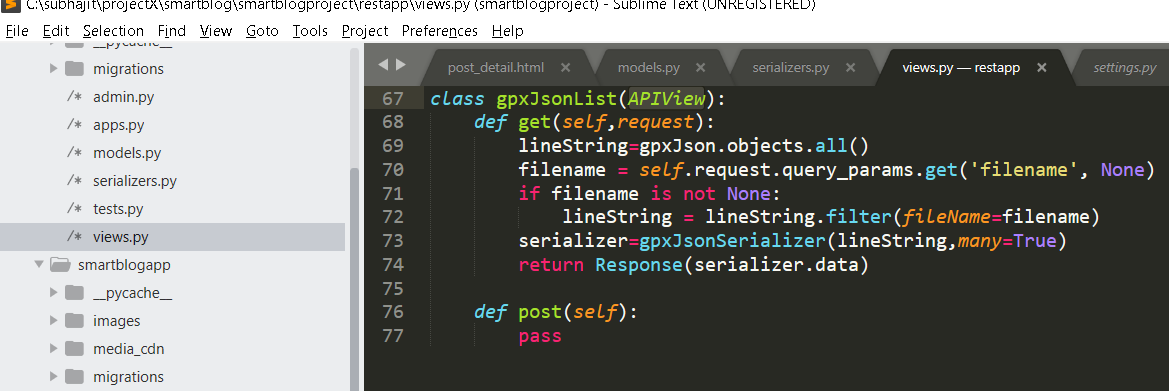


1. Create separate model class for each endpoint. Please note that the id field is reserved to make relationship between database entity and model object. Except id field, other fields are declared within the model class to map against the columns of the database table(entity). For an example while creating gpxjson class in model fields were declared like this.



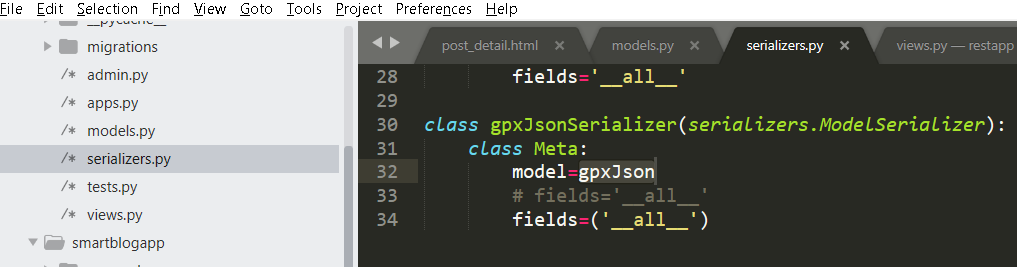
The motive was to pull line-string (that will help to draw the path) and group number (which cluster the file belongs to) with respect to filenames.

1. Create a view class for each model class created. The response of the view class should be a APIView (type) which derived from ‘rest\_framework.views’.



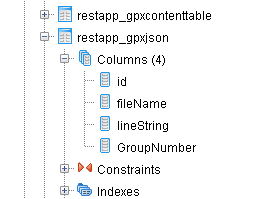
Here ‘gpxJson’ is a model object. By using query-string all the objects are mapped to lineString variable. The serialized response from the view actually exposed through an endpoint.

1. Creating serializers fills up the bridge between view and model entities. Serializers help to convert model objects into stream and vice versa.



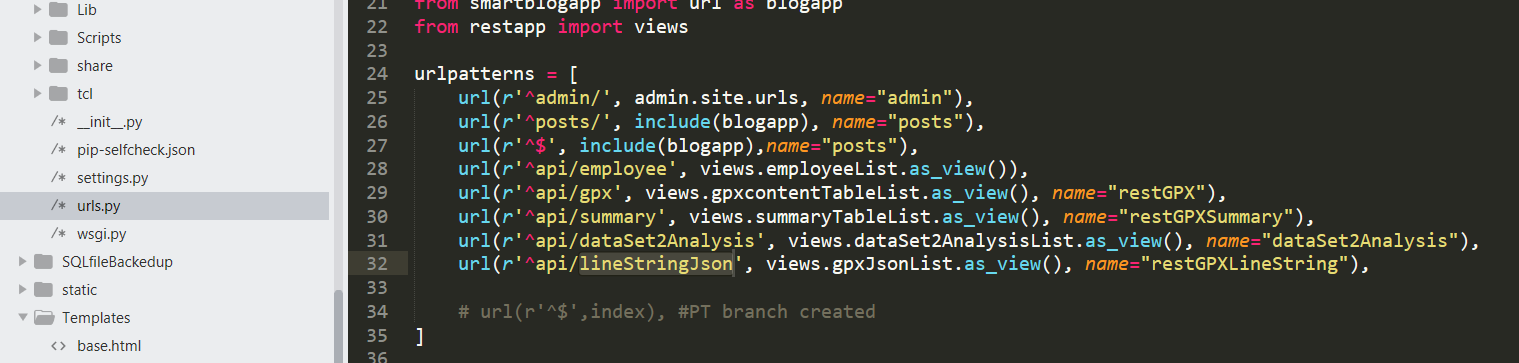
For serialization of gpxJson model entity, gpxJsonSerializer class is created for both way parsing process. Note: here all the fields are parsed with no filter restriction. By specifying only the allowed fields could have imposed filter restriction.

1. Model migration: when we are done with model changes, we can migrate changes to the database and can alter or create schema of the database entities. We are following code first approach and migration with the help of manage.py file helped us to create table in PostgreSQL like:



Note: gpxjson is the model class name and restapp is the app name. SO, by convention restapp\_gpxjson table is formed.

1. The last step is to expose the endpoint through a URI. In our case we defined that in url.py file like:



Sample response:

**Within the web application (admin, blog app, restapp)**

**R programming**

**PostgreSQL**

R(data load, extraction, derived fields, functions, Kmean and other algos from site)

PostgreSQL (tables used, relation between them)

Django (Model-view-template;)

Python

REST-API

AJAX

Javascript, Jquery

Bootstrap

Local- Heroku AWS S3

{**verify the variables here as they all are not required.}**

**CHAPTER 4: Analysis and Evaluation**

**CHAPTER 5: Conclusions and Future Work.**

**Section 5:1 Summary of Thesis:**

**Section 5:2 Overall Evaluation**

**Section 5:3 Future Work**

**REFERENCES/BIBLIOGRAPHY**

**APPENDICES**

**Summarising academic papers**