CHAPTER 1

INTRODUCTION

Ability to learn, this is what smallest definition for Machine Learning. Machine Learning is a field of study that have the capability to learn without being explicitly programmed. ML is one of the most exciting technologies that one would have ever come across. As it is evident from the name, it gives the computer that makes it more similar to humans. Machine learning algorithms attempt to draw similar conclusions as humans would by continually analyzing data with a given logical structure.

Machine learning applications are used in industries from Software, marketing, finance. In addition, machine learning is used to analysis and prediction over set of datasets. Our system does the same, it predicts the accident over desire route of user, which helps them to be aware and decrease accidents.

1.1 Purpose of the System

"Prevention is better than cure" is said and appreciated. The scenario behind this is to avoid Accident around the globe. In recent years, Machine learning enabled anomaly detection has emerged as a critical direction towards addressing some challenges. Prediction of road accidents is being achieved by Random Forest Algorithm in Machine Learning. Random Forest classifier is considered to be the best prediction algorithm which is also good in speed and accuracy.

1.2 Scope of the System

In this project, through Machine learning using Random Forest algorithm road accident is predicted. Through this algorithm and using Google API, system use the different type of marker to identify accident zone between user origin and destination and show the probability of accident. Addition to this, prediction also consider weather data, which enhance and increase the efficiency of output. Thus, user and authorized admin can take action accordingly.

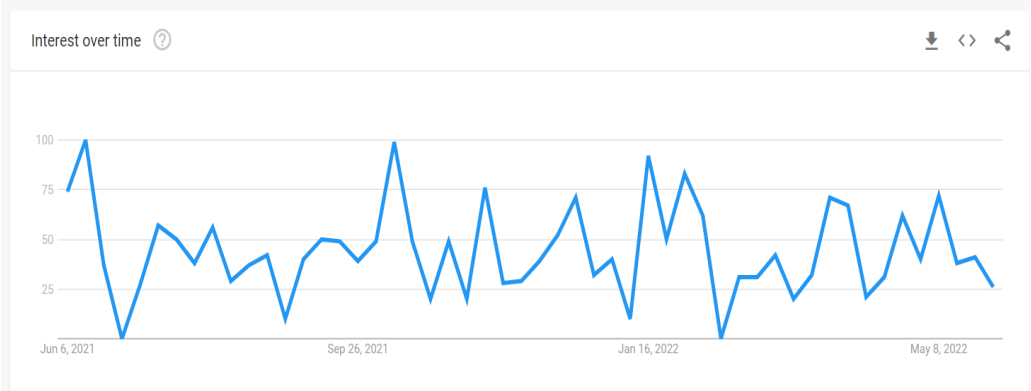


Figure 1.2.1 No. Of Accident in year

Casualty severity	User group	Casualty numbers			Percentage change in 2019 over	
		2005-2009 average	2018	2019	2018	2005-2009 average
Fatal and serious	Bus or coach occupants	277	112	91	-19%	-67% *
	Car occupants	1,773	623	574	-8%	-68% *
	Motorcyclists	1,397	1,080	1,019	-6%	-27% *
	Pedal cyclists	737	782	778	-1%	6%
	Pedestrians	2,021	1,366	1,350	-1%	-33% *
	Other vehicle occupants	197	102	93	-9%	-53% *
	Total	**6,403**	**4,065**	**3,905**	**-4%**	**-39%** *
	Child bus/coach passengers	23	9	5	-44%	-79% *
	Child car passengers	82	19	16	-16%	-80% *
	Child pedal cyclists	63	17	22	29%	-65% *
	Child pedestrians	423	176	157	-11%	-63% *
	Other child casualties	18	20	11	-45%	-39%
	Total	**608**	**241**	**211**	**-12%**	**-65%** *

Table 1.2 Road Accident Death in London

From above Figure, it can easily say that how road accident affected human daily life and death happen due to accident. Road accident, are something which can be avoidable by taking few cares. But still, lots of people engaged in accident. Though we can't do anything special for each people in today rush life. That is why our system going to predict accidental zone with warning sign. So that user and authorized may take action before anything happen.

1.3 Current System

In the past decades, computer vision, machine learning has shown promising results in several daily life problems. Recent improvement in machine learning allows

prediction and analysis more effective. To determine accident zone along with desired road in our system we are using Random Forest Classifier.

Random Forest Classifier predict the accident all over area provided in datasets. But to get access only desired prediction and zone we are using filter. This filter only select those accident which comes into its parameter of 50M along with the destinated road of user.

1.4 Proposed System

To predict the road accident zone in the real-world environment, we tend to apply Random Forest Algorithm to enhance and provide accurate output.

User can easily identify the accident zone, by addressing origin, destination along with date and time in provided place and take action accordingly. Same goes for authorized/admin which can also identify the accident zone. Addition to this, admin can upload real-time accident data into the system. So, that system can provide accurate output.

1.4.1 Why Our Approach Is Better

We have applied the following innovations to address the limitation of current state of the art:

1. Inclusion of more environmental features (more than 20 features from weather and geospatial data) for prediction of road accident. This is an improvement over existing models whose predictors are limited to a handful of speed variance data collected by loop detectors placed at a particular road segment.

2. Our prediction also cover a wider range of road segments with a total of 473 hotspots spread across 32 London boroughs. Existing models for real-time road accident prediction typically only cover a particular segment of a single highway.

3. Our prediction model is able to carry out prediction of road accident probabilities as far as 48 hours in advance (with the help of a 48-hour weather forecast API). Existing models for real-time road accident prediction are limited to carrying out prediction for the next 30 mins to 1 hour.

4. An improvement in accuracy as compared to existing models to predict road accident as depicted in the following table:

Reference	Model	Accuracy
Our project	*Random Forest*	***0.83***
Sun et al [3]	Support Vector Machine (SVM)	0.80
Lv et al [23]	K-nearest neighbours	0.80

Table 1.4.1 Accuracy of different machine learning algorithm

5. The creation of a user-friendly and easily accessible platform for the London public and authorities to utilize our road accident prediction model to inform their driving behaviors.

CHAPTER 2

REQUIREMENT ANALYSIS

2.1 Functional Requirements

The system offers two module User and Admin. User need to provide origin, destination, date and time to get the desired output. System is not collecting any user information. However, Admin need to login by providing username and password.

Addition to this, Admin need to re-login after every 5 minutes. This is given to provide more authentication during uploading of datasets.

2.2 Non-Functional Requirements

2.2.1 User Interface and Human Factors

The objective of this project is to develop computer-based design system for a human factor-based user interface design. The reasons for concentrating on human factors are their increasing importance. In this project we will be providing the user with real-time application. The project is designed in such a way that it is interactive and user friendly. Persons having a basic knowledge of computer can easily operate our proposed system.

2.2.2 Software Requirements

In this project we are using: -

1. Operating System- Windows 8 or above.
2. Front End- HTML, CSS, JAVASCRIPT.
3. Framework- BOOTSTRAP, FLASK.
4. Back End- Python.
5. API- Google Map, DarkSky(Weather data).

2.2.3 Hardware Requirements

This project can run on System: -

1. Currently lowest version of desktop/laptop.
2. Hard Disk- 40 GB
3. RAM – 1 GB

2.2.4 Usability

This project can be used by both the public and the legal authorities. Its usability is very simple, for public they can access the data regarding road accidents in a particular area at a particular time by simply logging into the system through User page. In the user page the user can enter the origin & destination address along with the time & date.

While coming to the usability for any legal authority or traffic police, they can access the system by logging in through the admin page by simply entering the admin username and password. After the successful login they can view and enter the data sets related to accidents happening in the area.

2.2.5 Reliability

The probability of the failure for this project is very low. The cost of maintaining the software is very affordable. As we know that this project is based on real-time application hence it is very reliable to operate. It is designed in such a way that the chances of failure happening is very low, which simply increases the reliability of the system.

2.2.6 Performance

The performance of the system which is provided in this project is very optimized. It provides the user with the probability of how many accidents happening in a particular zone of area at a particular time. We can also enhance the performance of this system by providing every possible data regarding accidents in the particular area in the CSV file format or by manually entering it into the system. To need a better performance user needs to on a stable network for accessing this system.

2.2.7 Supportability

This project is designed in such a way so that it can be supported in mobile devices as well as laptop/desktop. For mobile devices there is no particular limit for supportability but while coming to laptop/desktop it needs to have at least 1 GB

RAM, 40 GB hard disk and currently lowest version of laptop/desktop. It can also be supported in some other devices like Tablets.

2.2.8 Security Requirements

To provide security to the system so that nobody can randomly enter into the admin section we are providing a username and password. Apart from admin no one can enter and see the data present in the system and they cannot make changes also.

2.2.9 Resource Requirements

For designing this project, we need a large amount of time and data. The developer should have a knowledge of Machine learning as well as python programming also. Along with that the developer should be aware of JAVASCRIPT, BOOTSTRAP, CSS and should be friendly with working on API. The basic resources needed in this project is the datasets regarding accidents in excel or CSV format.

CHAPTER 3

MODULES

3.1 Introduction

It includes two basic modules.

1. User
2. Admin

3.1.1 User

System is made for user to get prediction about road accident on their provided data. User need to provide origin, destination, date and time (Under 48 hour).

1. **Home–** User can access prediction page from home page.
2. **Interaction –** This is the section were all magic happen. Predicting road accident between source and destination make sure that user/travellers pay full attention and take full precaution.

3.1.2 Admin

Admin is someone who is authorized by Government to act on behalf of them. Generally, Police is considered as admin. Admin have given full authority and control over system.

1. **Admin Login –** Admin can login into his account by giving his username and password. If both the user's name and password are valid then the admin would be able to access the account.
2. **Dataset –** This section is used to upload and view dataset. In this admin can upload data as a CSV file or a single data set at a time.
3. **Exploration –** This section is for the analysis purpose over uploaded data.
4. **Interaction –** This is the section where the system predict accident. This section help authority to action before any bad happen.
5. **Log out and Home –** Log out allow admin to log out after their work is done and home is something which introduce about system to admin.

CHAPTER 4

SYSTEM ANALYSIS

4.1 Introduction

User-oriented techniques are widely used in software requirement analysis and design. Use cases and usage scenarios facilitate system understanding and provide a common language for communication. This paper presents a scenario-based modelling technique and discusses its applications. In this model, we represent the process and the data flow of the project. We are using use case diagrams as an example and explain the project Road accident acknowledgement with Machine Learning Model.

This Project helps you to identify and indicate the probability of happing accident and locate the location on Google map.

4.2 UML Diagrams

The unified modelling language allows the software engineer to express an analysis model using the modelling notation that is governed by a set of syntactic semantic and pragmatic rules. Our Project has two Uml diagrams to describe the flow.

4.2.1 User Model View

- This view represents the system from the user's perspective.
- In this Model View how a user interact is shown. The user generally enters the source, destination and date/time in the System.

4.2.2 Admin Model View

- This view represents the system from the admin's perspective.
- In this Model View how a admin interact is shown. The admin generally login into the system and then able to update the data and also view the data.

4.3 Use Case Diagram

Use case diagrams are created to visualize the relationships between actors and use cases. A use case is a pattern of behavior the system exhibits. Use case contains actors which are performing the actions. Actors plays a main role and, in our project, we have two actor's admin and the user. The admin performs the actions like login, logout, upload, view data.

4.3.1 Actors

In a use case diagram is any entity that performs a role in one given system. Actors are user and admin who performs the action.

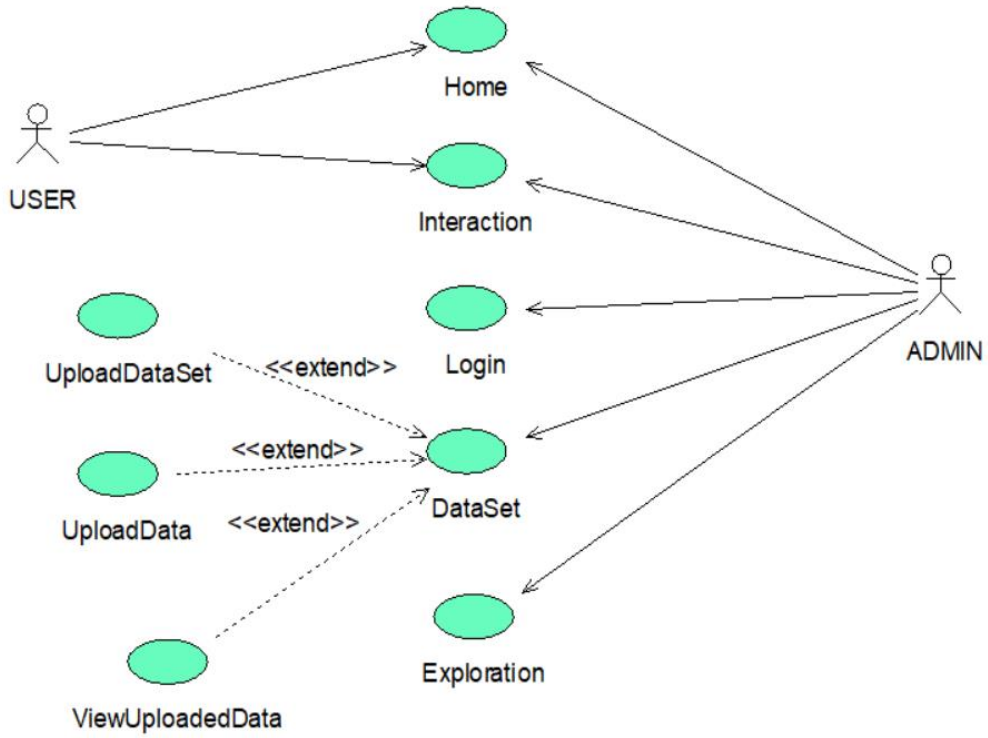


Figure 4.3.1 Use case diagram of Road accident acknowledgement System

The above use case diagram specifies the procedure of the Road accident acknowledgement System with Machine learning Model. In this the user will Interact with the system. The system in which we keep the algorithm named “model” is trained to identify the location where chances of happing accident is more. After detecting the location is marked on the Google Map according to probability of accident happing with different color of marker.

CHAPTER 5

SYSTEM DESIGN

5.1 Introduction

Our proposed system states as follow –

Our system describes one possibility of how to use collected data about traffic accidents to mine frequent patterns and important factors causing different types of accidents and predicting the accident. A user can easily predict accident in his/her journey between source and destination. Which make them aware and they can travel with more precaution. On other hand authority have full access over data and act accordingly. Even they can act before any accident on particular day and may be pre prepared for any kind of tragedy.

5.2 Subsystems

In this project there are two sub-system for user and admin.

5.2.1 Subsystem for admin

Admin is someone who is authorized by Government to act on behalf of them. Generally, Traffic Police is considered as admin. Admin have given full authority and control over system.

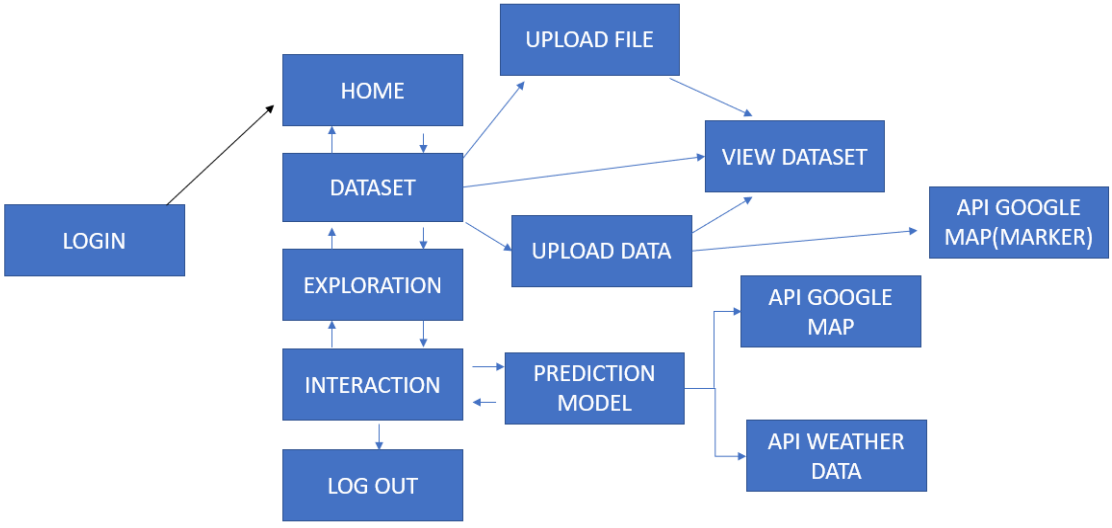


Figure 5.2.1 Block diagram of proposed system for admin

5.2.2 Subsystem for user

System is made for user to get prediction about road accident on their provided data. User need to provide origin, destination, date and time (Under 48 hour).

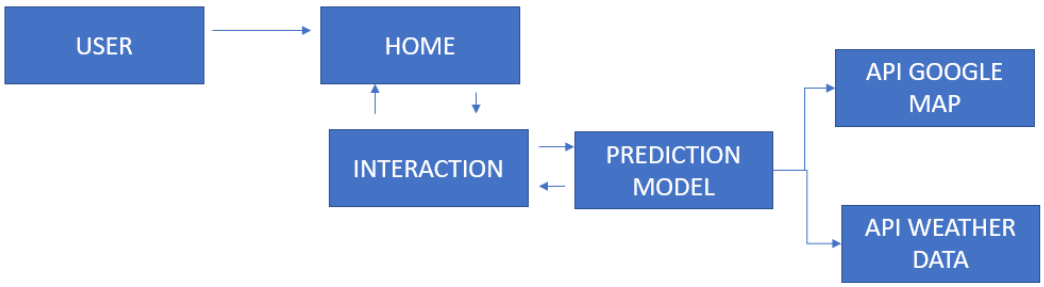


Figure 5.2.2 Block diagram of proposed system for user

5.3 Class Diagram

A Class diagram describe the static structure of a system, or how it is structured rather than how it behaves. Class diagram gives an overview of a system by showing its classes and the relationships among them. In our class diagram we have admin, system and the user are the three main classes in our system. To know the relation between the classes we have Association, Aggregation and generalization.

Association -- A relationship between instances of the two classes. There is an association between two classes like admin and the system and also between the system and the user.

Aggregation -- An association in which one class belongs to a collection.

Generalization -- An inheritance link indicating one class is a super class of the other. Class diagrams are widely used to describe the types of objects in a system and their relationships. Classes are composed of three things: a name, attributes, and operations. Below is an example of a class.

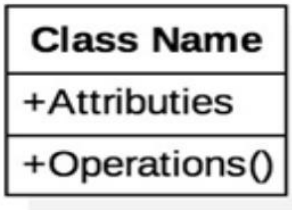


Figure 5.3.1 Class Diagram Icon

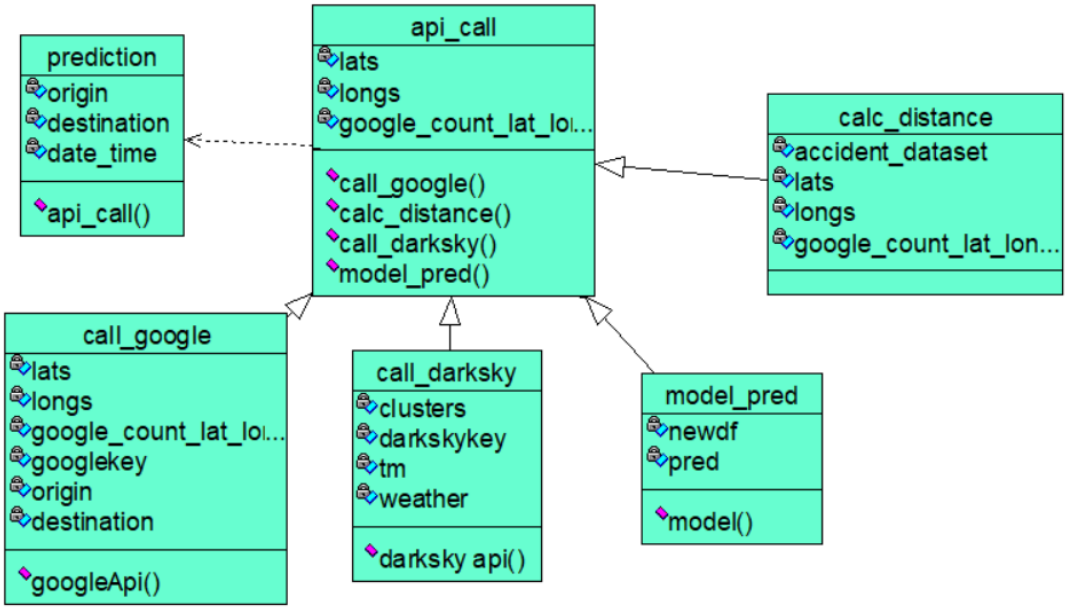


Figure 5.3.2 Class Diagram for Road accident acknowledgement System

5.4 Sequence Diagram

It is a type of interaction diagram, a sequence diagram shows the actors of the object participating in an interaction and the events they generate arranged in a time sequence. In the sequence diagram we describe the flow of the project how it was going on. First the admin login to the page with login id and password. After that we input the dataset into the system later the system performs the algorithm and predict the accident location. Generally, we store a value predefined in order to no need of preprocessing. So, the system checks with that and locate the location with green, yellow and red color.

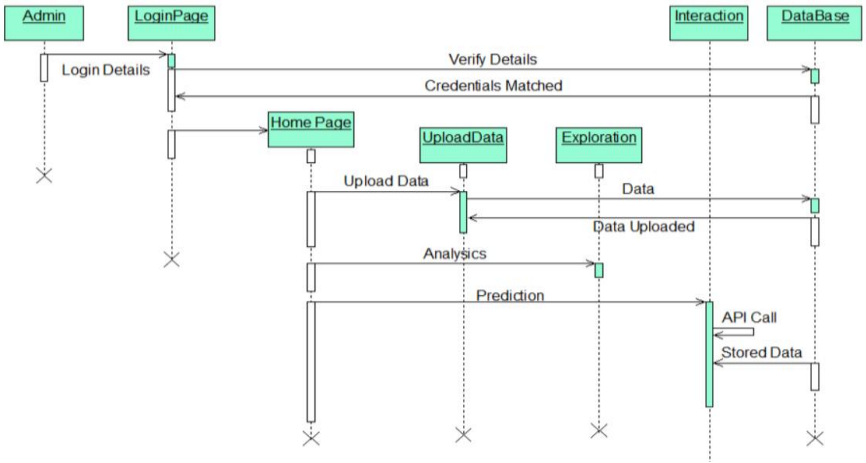


Figure 5.4 Sequence Diagram for Road accident acknowledgement System.

The above sequence diagram specifies the sequential procedure followed by Road accident acknowledgement Model.

5.5 Collaboration Diagram

Collaboration Diagram is same as that of the Sequence diagram where sequence diagram shows the flow and the time line of every process. Collaboration is another of it where it does not show the time line but the line wise process is described here.

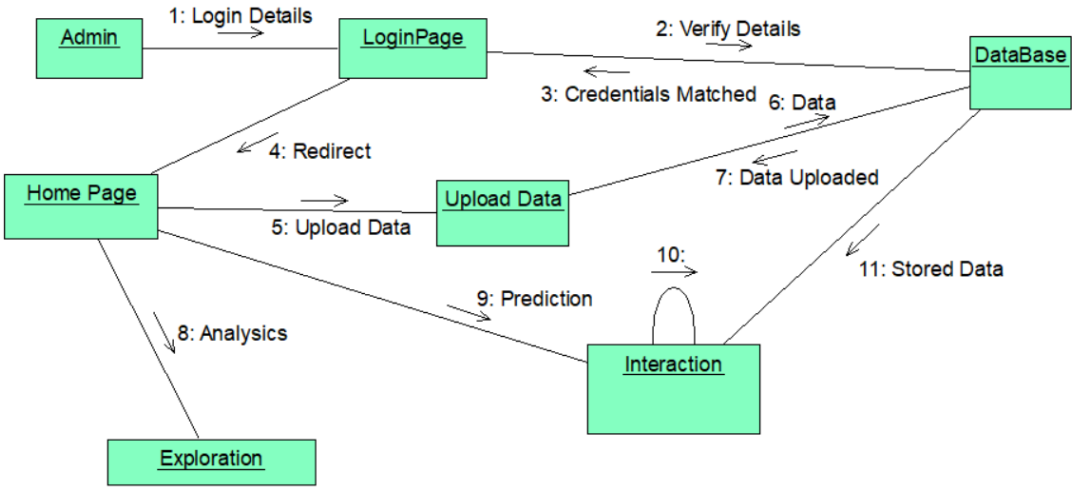
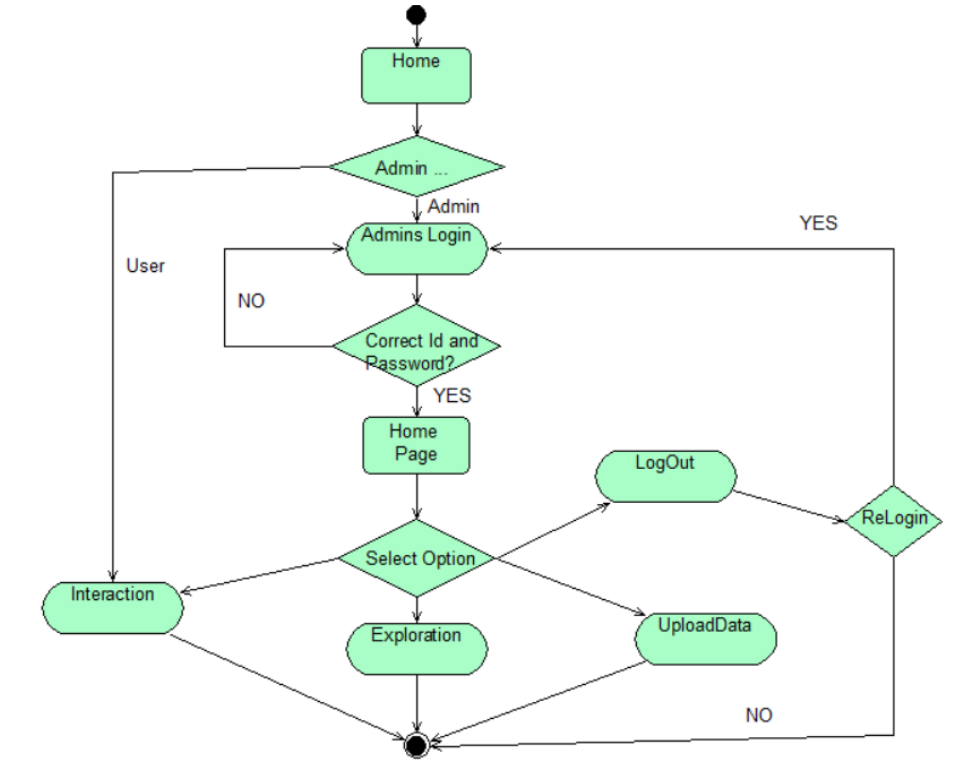


Figure 5.5 Collaboration Diagram for Road accident acknowledgement System

The above collaboration diagram specifies the sequential procedure followed by Road accident acknowledgement System

5.6 Activity Diagram

Activity diagram is another important diagram in UML to describe dynamic aspects of the system Activity diagram, it- is basically a flow chart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. So, the control flow is drawn from one operation to another operation. Activity diagram is the flow chart of the process is done. From the user input to the admin logout process is described here. User enters with in the and the enter the source, destination and date/time.



5.6 Activity Diagram for Road accident acknowledgement System

The above activity diagram shows the flow of activities of Road accident acknowledgement System. It shows about the start and stop states of the flow along with the decisions and counter statement

CHAPTER 6

IMPLEMENTATION

6.1 Backend Implementation

A web application has been built using the Flask framework. Flask is based on Python as the Server-Side Language and will be used to handle all the server side processes. All the html pages, JavaScript libraries and CSS from front-end are integrated into the web application.

Google Maps API and Google Places API are used for route planning and autocomplete function of places respectively. The first 40,000 calls for Routes API and the first 70,000 characters for the Places API in a month are free. Weather forecasts is obtained by calling Darksky API and it has free 1000 calls per day which is enough for our development use. In addition, a RESTful API module was built to handle users' requests of road accident predictions.

Once a user enters the three inputs, i.e., date and time, traveling origin and destination, a POST request is sent to the backend framework. Google API is then called for route planning. Using the latitudes and longitudes on the route returned by Google, the backend calculates a radius of 50 meters from these points. Any past accident points in this dataset that don't fall within this 50m distance are filtered out.

Next, for each unique cluster in the remaining accident points, Darksky API is called. Each unique cluster will have the same weather forecast. This is a reasonable imputation as each cluster has a 50 meters radius and they should share the same weather. Instead of calling the weather API for many latitudes and longitudes, doing so allows our webpage to return the results to the users faster and reduces lag time.

With the weather data, the final model is now loaded and predictions are made. For those duplicated latitudes and longitudes, i.e., accidents had happened at the same spot multiple times, duplicates are removed. We have not explored other methods of assigning higher probability to these accident-prone locations.

Frontend can now use the predictions to generate visualizations and highlight potential accident sites.

6.1.1 Dataset

We have taken dataset from Kaggle website. In which a comprehensive dataset of road accident in the UK from 2012-2014 (discontinuous) is taken. These data collected and hosted by UK Dept of Transport.

Key Features extracted from dataset are: -

- location: latitude/longitude coordinates
- Week day, hour of day, day of year, year.
- Road condition, speed limit, road number.
- Cluster, local authority.
- Weather data

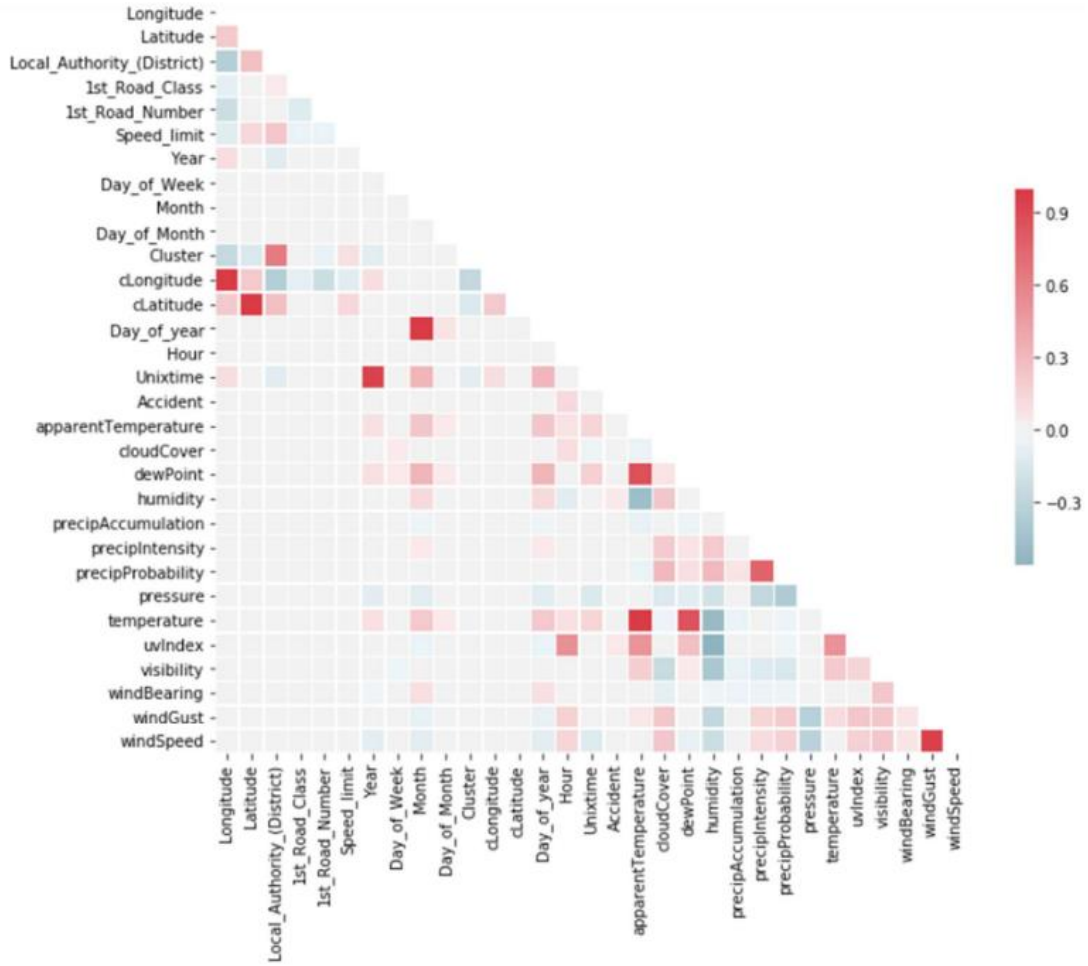


Figure 6.1.1 Features in the UK road accident data set.

Figure 6.1.1 shows a correlation matrix among the numerical features of the dataset. For a set of highly correlated features, it is standard practice to exclude all but one

feature to reduce impact of multicollinearity. However, certain models such as Random Forest, are relatively unaffected by multicollinearity. Therefore, the need to exclude correlated features will depend on the model used. The features and their importance scores for the Random Forest Classifier is shown in Figure 6.1.2.

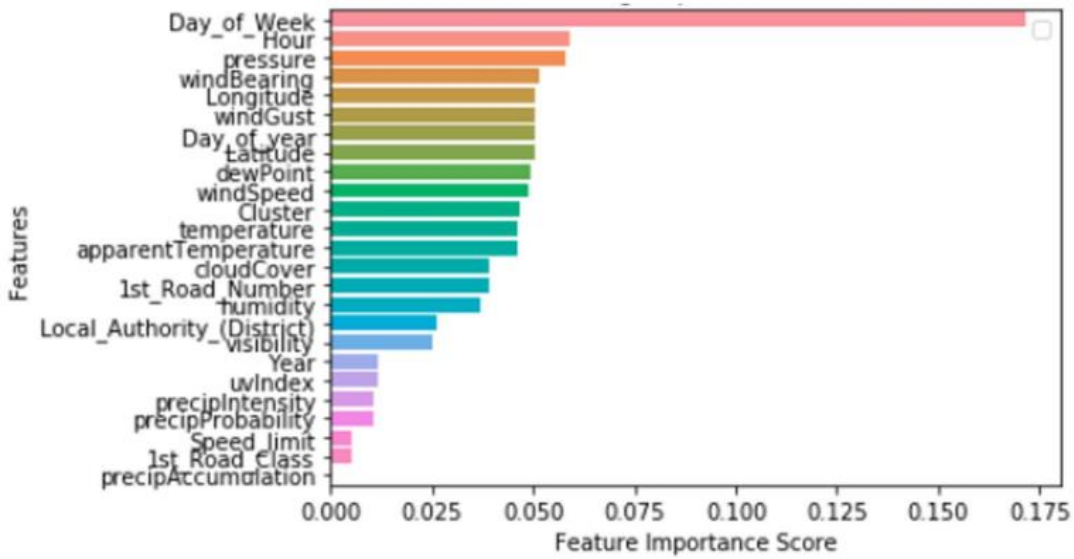


Figure 6.1.2 Feature used in Random Forest Classifier

6.1.2 Libraries Used

1. NumPy is a Python library used for working with arrays. NumPy stands for Numerical Python. It also has functions for working in domain of linear algebra, Fourier transform, and matrices. In Python we have lists that serve the purpose of arrays, but they are slow to process. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.

2. Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

3. Joblib is a set of tools to provide lightweight pipelining in Python. In particular:

1. transparent disk-caching of functions and lazy re-evaluation (memorize pattern)

2. easy simple parallel computing

Joblib is optimized to be fast and robust on large data in particular and has specific optimizations for *NumPy* arrays.

4. Pickle module is used for serializing and de-serializing python object structures.

5. Sklearn has featured the various classification, regression and clustering algorithms including support-vector machines, random forests, gradient boosting, k-means DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

6.2 Frontend Implementation

Engaging and entertaining USER and ADMIN is the main motive behind creating Interactive and attractive UI. HTML, CSS and JAVASCRIPT are the programming language used.

Our project is presented as a website at **http://acetproject.herokuapp.com**. This website contains two main sections: “Dataset” and “Interaction”. The “Dataset” section gives authority to Admin, to add, view and upload whole new file of dataset into the system.

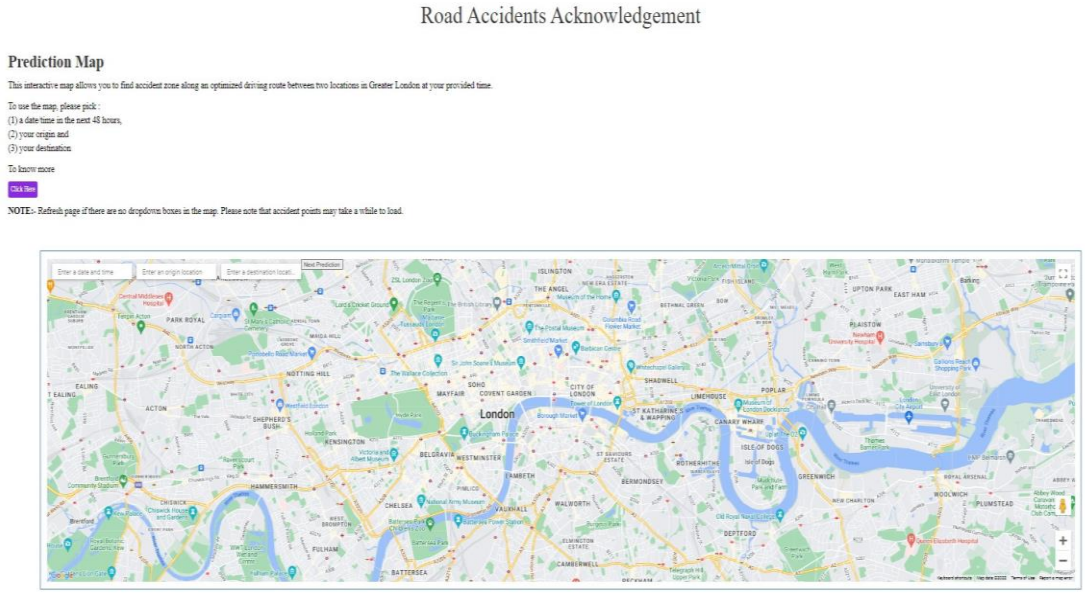


Figure 6.2.1 Interaction (Prediction) Page

The “Interaction” section contains an interactive map which will carry out road accident prediction. This visualization will allow users to input a specific particular date/time. Upon making this selection, the website will fetch weather information that correspond to the chosen date/time. These three inputs (date, time and weather) will be sent to our trained model, which in turn will predict probabilities on accident-prone spots. These spots will then be displayed on the map.

Google Maps APIs will be called to show proposed routes based on user-inputted origin and destination. Users are able to input the origin and destination with suggested options presented by Google Place API. We will collect the latitude-longitude coordinates of various places along the determined route and send these coordinates to our backend platform for model prediction. The returned results from the prediction model will be displayed as colored (Red, Yellow, Green) marker on the route to show probabilities of accidents in the ‘hotspot’ areas**.**

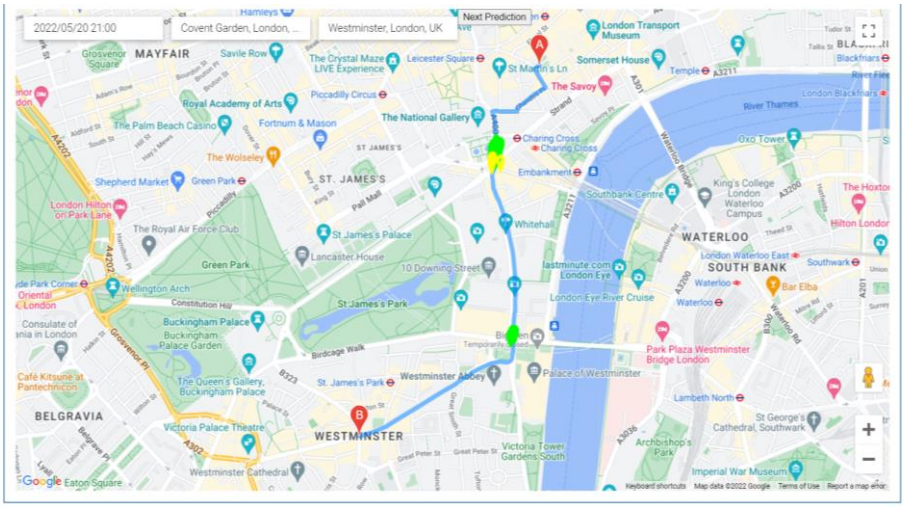


Figure 6.2.2 Fetching data from Google Map

6.3 Random Forest Algorithm

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model. As the name suggests, "Random Forest is a

classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

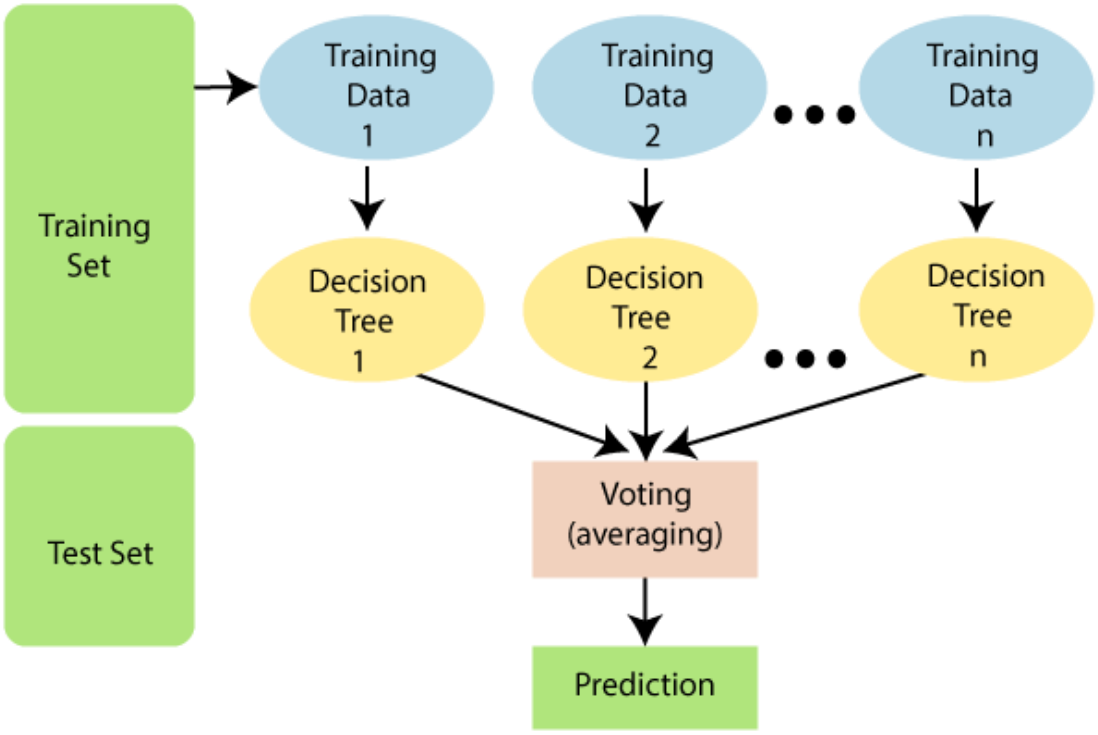


Figure 6.3 Random Forest Algorithm

6.4 Source Code

6.4.1 Random_Forest.iplyb

```
import os
# set wd
#Set your own data files path here
path = "."
os.chdir(path)
```

Import Libraries

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import random
import datetime
```

Import all data

```
#Random Forest
```

```
pima = pd.read_csv("alldatapoints_w_weather.csv")
```

```
pima.head()
```

	Unnamed: 0	Longitude	Latitude	Day_of_Week	Local_Authority_(District)	1st_Road_Class	1st_Road_Number	Speed_limit	Year	Date	...	precipProbability	precipType	pressure
0	0	-0.185496	51.483253	5	12	3	308	30	2012	19/1/2012	...	0.0	none	1022.61
1	1	-0.185496	51.483253	0	12	3	308	30	2012	16/7/2012	...	0.0	none	1018.66
2	2	-0.185496	51.483253	5	12	3	308	30	2012	5/5/2012	...	0.0	none	1010.30
3	3	-0.185496	51.483253	2	12	3	308	30	2012	2/5/2012	...	0.0	none	1020.73
4	4	-0.160418	51.501567	5	1	3	3216	30	2012	12/1/2012	...	0.0	none	1027.15

5 rows × 34 columns

```
# Import train_test_split function
from sklearn.model_selection import train_test_split

X=pima[['Longitude',
    'Latitude',
    'Cluster',
    'Day_of_Week',
    'Hour',
    'Day_of_year',
    'Local_Authority_(District)',
    '1st_Road_Class',
    '1st_Road_Number',
    'Speed_limit',
    'Year',
    'apparentTemperature',
    'cloudCover',
    'dewPoint',
    'humidity',
    'precipAccumulation',
    'precipIntensity',
    'precipProbability',
    'pressure',
    'temperature','uvIndex',
    'visibility',
    'windBearing',
    'windGust',
    'windSpeed']]  # Features
y=pima['Accident']  # Labels

# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
# 70% training and 30% test
```

```
#Import Random Forest Model
from sklearn.ensemble import RandomForestClassifier

#Create a Gaussian Classifier
clf=RandomForestClassifier(n_estimators=100)

#Train the model using the training sets y_pred=clf.predict(X_test)
clf.fit(X_train,y_train)

y_pred=clf.predict(X_test)
```

```
from sklearn.ensemble import RandomForestClassifier

#Create a Gaussian Classifier
clf=RandomForestClassifier(n_estimators=100)

#Train the model using the training sets y_pred=clf.predict(X_test)
clf.fit(X_train,y_train)
```

```
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=2,
            min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

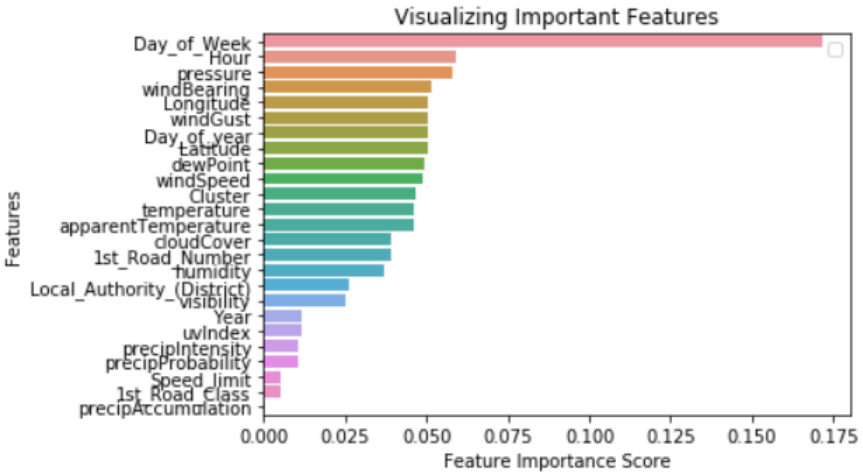
```
feature_cols = [
    'Longitude',
    'Latitude',
    'Cluster',
    'Day_of_Week',
    'Hour',
    'Day_of_year',
    'Local_Authority_(District)',
    '1st_Road_Class',
    '1st_Road_Number',
    'Speed_limit',
    'Year',
    'apparentTemperature',
    'cloudCover',
    'dewPoint',
    'humidity',
    'precipAccumulation',
    'precipIntensity',
    'precipProbability',
    'pressure',
    'temperature','uvIndex',
    'visibility',
    'windBearing',
    'windGust',
    'windSpeed'
]
```

```
import pandas as pd
feature_imp = pd.Series(clf.feature_importances_,index=feature_cols)
.sort_values(ascending=False)
feature_imp
```

```
Day_of_Week                  0.171669
Hour                         0.059017
pressure                     0.057898
windBearing                  0.051640
Longitude                    0.050573
windGust                     0.050467
Day_of_year                  0.050284
Latitude                     0.050244
dewPoint                     0.049295
windSpeed                    0.048615
Cluster                      0.046737
temperature                  0.046185
apparentTemperature          0.046116
cloudCover                   0.038969
1st_Road_Number              0.038923
humidity                     0.036827
Local_Authority_(District)   0.026118
visibility                   0.025149
Year                         0.011769
uvIndex                      0.011454
precipIntensity              0.010738
precipProbability            0.010439
Speed_limit                  0.005375
1st_Road_Class               0.005253
precipAccumulation           0.000246
dtype: float64
```

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
# Creating a bar plot
sns.barplot(x=feature_imp, y=feature_imp.index)
# Add labels to your graph
plt.xlabel('Feature Importance Score')
plt.ylabel('Features')
plt.title("Visualizing Important Features")
plt.legend()
plt.show()
```

```
No handles with labels found to put in legend.
```



6.4.2 Api_Call_Pred.py

api_call_pred.py

app > api_call_pred.py > ...

You, 2 weeks ago | 1 author (You)

```python
from xml.dom.minidom import TypeInfo
import requests
import pandas as pd
import datetime
import numpy as np
import itertools
import os
from sklearn.externals import joblib
from app.config import Config


darkskykey = Config['darkskykey']
googlekey = Config['googlekey']


script_path = os.path.dirname(os.path.abspath( __file__ + "/../"))
accident_dataset = pd.read_csv("./app/UPLOAD_FOLDER/only_accident_points.csv")

# Load  model nad model columns
model = joblib.load("./model/model.pkl")
model_columns = joblib.load("./model/model_columns.pkl")
      You, 4 weeks ago • 'github'

def call_google(origin, destination, googlekey):
    PARAMS = {'origin': origin, 'destination': destination, 'key': googlekey, }
    URL = "https://maps.googleapis.com/maps/api/directions/json"
    res = requests.get(url=URL, params=PARAMS)
    data = res.json()
    # parse json to retrieve all lat-lng and  array of DirectionsLeg objects
    waypoints = data['routes'][0]['legs']

    lats = []
    longs = []

    google_count_lat_long = 0

    # find cluster of intenest from google api route
    for leg (variable) step: Any
        for step in leg['steps']:
            start_loc = step['start_location']
            # print("lat: " + str(start_loc['lat']) + ", lng: " + str(start_loc['lng']))
            lats.append(start_loc['lat'])
            longs.append(start_loc['lng'])
            google_count_lat_long += 1

    lats = tuple(lats)
    longs = tuple(longs)
    print("total waypoints: " + str(google_count_lat_long))

    return lats, longs, google_count_lat_long


```

```
def calc_distance(accident_dataset, lats, longs, google_count_lat_long):
    # load all cluster accident waypoints to check against proximity
    accident_point_counts = len(accident_dataset.index)

    # approximate radius of earth in km
    R = 6373.0
    new = accident_dataset.append([accident_dataset] * (google_count_lat_long - 1), ignore_index=True)
    # repeat data frame (9746*waypoints_count) times
    lats_r = list(
        itertools.chain.from_iterable(itertools.repeat(x, accident_point_counts) for x in lats))  # repeat 9746 times
    longs_r = list(itertools.chain.from_iterable(itertools.repeat(x, accident_point_counts) for x in longs))

    # append
    new['lat2'] = np.radians(lats_r)#convert degrees to radians and store in array
    new['long2'] = np.radians(longs_r)

    # cal radiun50m
    new['lat1'] = np.radians(new['Latitude'])
    new['long1'] = np.radians(new['Longitude'])

    new['dlon'] = new['long2'] - new['long1']
    new['dlat'] = new['lat2'] - new['lat1']
    #formula to calculate distance between lat and long:sin²(Δϕ/2) + cos ϕ1 · cos ϕ2 · sin²(Δλ/2)
    new['a'] = np.sin(new['dlat'] / 2) ** 2 + np.cos(new['lat1']) * np.cos(new['lat2']) * np.sin(new['dlon'] / 2) ** 2
    new['distance'] = R * (2 * np.arctan2(np.sqrt(new['a']), np.sqrt(1 - new['a'])))


    return new


    You, 4 weeks ago • 'github' …
def call_darksky(clusters, darkskykey, tm):
    # weather api call
    weather = pd.DataFrame()
    print(tm) #2019-04-11T23:36

    #time format for darksky API, eg 2019-04-11T23:00:00
    datetime_str = datetime.datetime.strptime(tm, '%Y-%m-%dT%H:%M').strftime('%Y-%m-%dT%H:%M')
    tm2 = datetime_str[0:10] + "T" + datetime_str[11:13] + ":00:00"

    for index, row in clusters.iterrows():
        lat = row["Latitude"]
        long = row["Longitude"]
        weather_url = "https://api.darksky.net/forecast/" + darkskykey + "/" + str(lat) + "," + str(long) + "," + tm2 + \
                      "?exclude=[currently,minutely,daily,flags]"
        w_response = requests.get(weather_url)
        w_data = w_response.json()

        # put json into a dataframe
        datetime_object = datetime.datetime.strptime(tm, '%Y-%m-%dT%H:%M')
        iweather = pd.DataFrame(w_data["hourly"]["data"][datetime_object.hour], index=[0])
        iweather["Cluster"] = row['Cluster']
        iweather['precipAccumulation'] = 0

        weather = weather.append(iweather)

    return weather

def model_pred(new_df):

      # do prediction for current datetime for all
      prob = pd.DataFrame(model.predict_proba(new_df), columns=['No', 'probability'])
      prob = prob[['probability']]
      #merge with long lat
      output = prob.merge(new_df[['Latitude', 'Longitude']], how='outer', left_index=True, right_index=True)

      #drop duplicates of same lat long (multiple accidents)
      output["Latitude"] = round(output["Latitude"], 5)
      output["Longitude"] = round(output["Longitude"], 5)
      output = output.drop_duplicates(subset=['Longitude', 'Latitude'], keep="last")

```

```
    # to json
    processed_results = []
    for index, row in output.iterrows():
        lat = float(row['Latitude'])
        long = float(row['Longitude'])
        prob = float(row['probability'])

        result = {'lat': lat, 'lng': long, 'probability': prob}
        processed_results.append(result)

    print("total accident count:", len(output))

    return processed_results



def api_call(origin, destination, tm):

    #creating date  time object strptime
    datetime_object = datetime.datetime.strptime(tm, '%Y-%m-%dT%H:%M')
```

```
def api_call(origin, destination, tm):

    #creating date  time object strptime
    datetime_object = datetime.datetime.strptime(tm, '%Y-%m-%dT%H:%M')
      You, 4 weeks ago • 'github' …
    # get route planning
    lats, longs, google_count_lat_long = call_google(origin, destination, googlekey)

    # calculate distance between past accident points and route
    dist = calc_distance(accident_dataset, lats, longs, google_count_lat_long)

    # filter for past accident points with distance <50m - route cluster
    dat = dist[dist['distance'] < 0.050][['Longitude','Latitude','Day_of_Week','Local_Authority_(District)',
                                          '1st_Road_Class','1st_Road_Number','Speed_limit', 'Year','Cluster',
                                          'Day_of_year', 'Hour']]

    #if no cluster, exit
    if len(dat) == 0:
        return print(" Hooray! No accidents predicted in your route.")

    else:
        # filter for accident points in route cluster
        dat = dat.drop(columns=['Hour', 'Day_of_year', 'Day_of_Week', 'Year'], axis=0)
        dat['Hour'] = datetime_object.hour
        day_of_year = (datetime_object - datetime.datetime(datetime_object.year, 1, 1)).days + 1
        dat['Day_of_year'] = day_of_year
        day_of_week = datetime_object.date().weekday() + 1
        dat['Day_of_Week'] = day_of_week
        dat['Year'] = datetime_object.year


        #get weather prediction for unique cluster in past accident dataset
        ucluster = list(dat['Cluster'].unique())

        clusters = dat[dat['Cluster'].isin(ucluster)].drop_duplicates(subset='Cluster', keep='first')
        weather = call_darksky(clusters, darkskykey, tm)

        # merge with accident data - df with latlong and weather
        final_df = pd.merge(dat, weather, how='left', on=['Cluster'])
        # final_df.to_csv('./filename.csv', index = False)
        # print(final_df)
        final_df = final_df.drop(columns=['time', 'summary', 'icon', 'ozone'], axis=0)
        final_df = final_df[model_columns]

        #run model predicition
        processed_results = model_pred(final_df)

        final = {}
        final["accidents"] = processed_results

        return final
```

CHAPTER 7

TESTING

SYSTEM TEST

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

7.1 TYPES OF TESTS

7.1.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system conFigureuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

7.1.2 Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

7.1.3 Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input: identified classes of valid input must be accepted.

Invalid Input: identified classes of invalid input must be rejected.

Functions: identified functions must be exercised.

Output: identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

7.1.4 System Test

System testing ensures that the entire integrated software system meets requirements. It tests a conFigureuration to ensure known and predictable results. An example of system testing is the conFigureuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

7.1.5 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

7.1.6 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box.you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

7.2 Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

7.2.1 Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

7.2.2 **Features to be tested**

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

7.2.3 Test Results:

All the test cases mentioned above passed successfully. No defects encountered.

7.2.4 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

CHAPTER 8

OUTPUT SCREENS

8.1 USER

1. This is home page for the user, from where user can easily direct to prediction page.

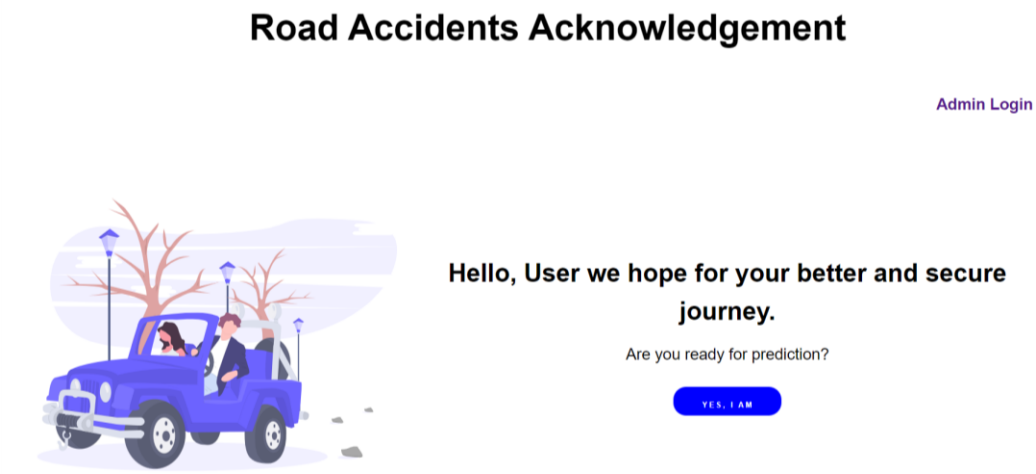


Figure 8.1.1 User Home Page

2. This is prediction page used by user.

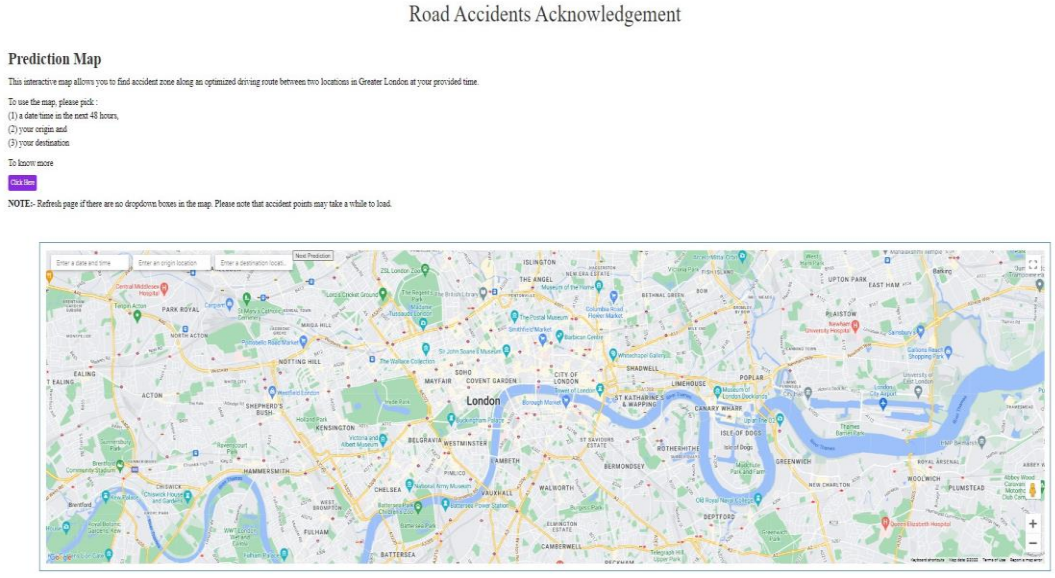


Figure 8.1.2 User Prediction Page

3. After providing required data from the user for making prediction, below Figure will be shown in the browser.

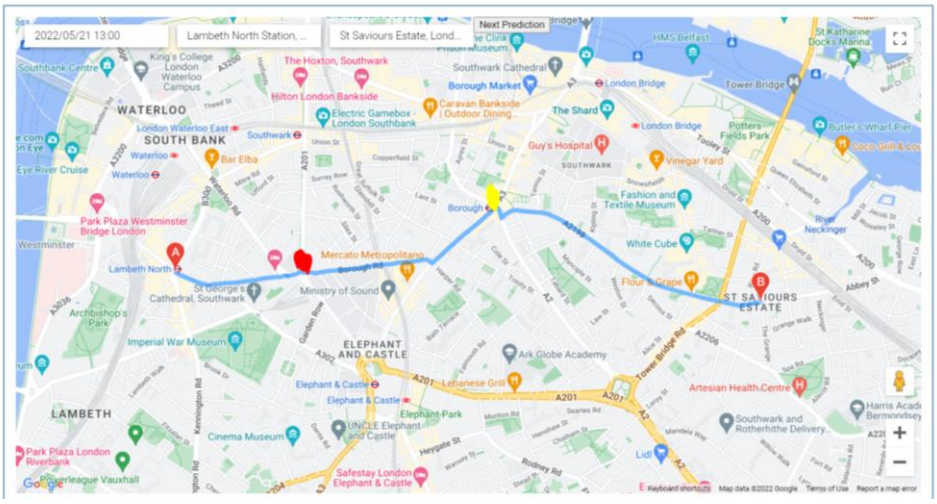


Figure 8.1.3 Prediction page (Output)

8.2 ADMIN

1. This is Login page for the admin for the authentication purpose.

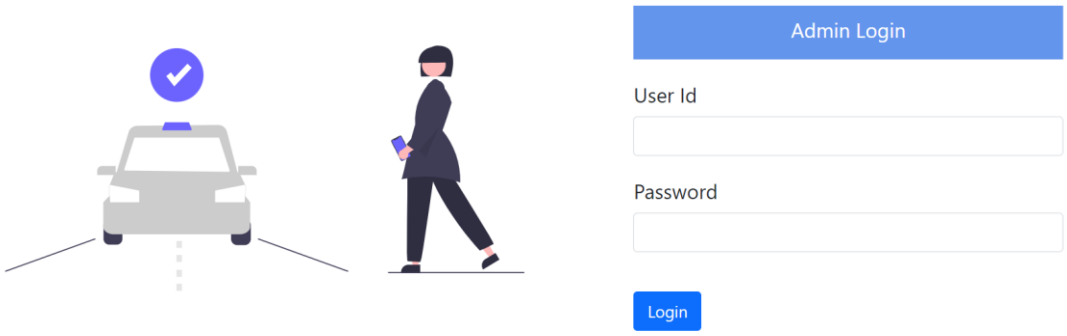


Figure 8.2.1 Admin Login page

2. This page provide option to an admin to upload a whole CSV file or to upload single set of data. Addition to this, its also allow to view uploaded dataset.

Figure 8.2.2 Admin dataset

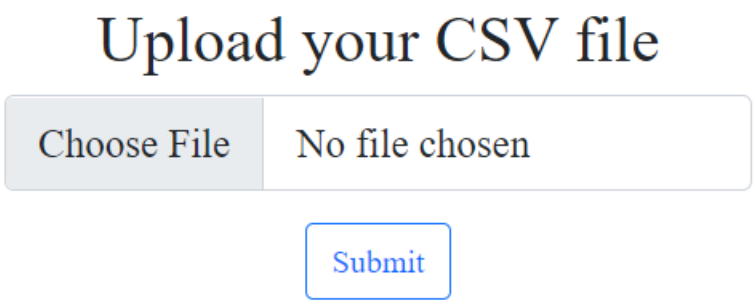


Figure 8.2.3 To upload CSV file

4. Admin can upload single data set from this page, by using google map admin can easily track longitude and latitude of accident.

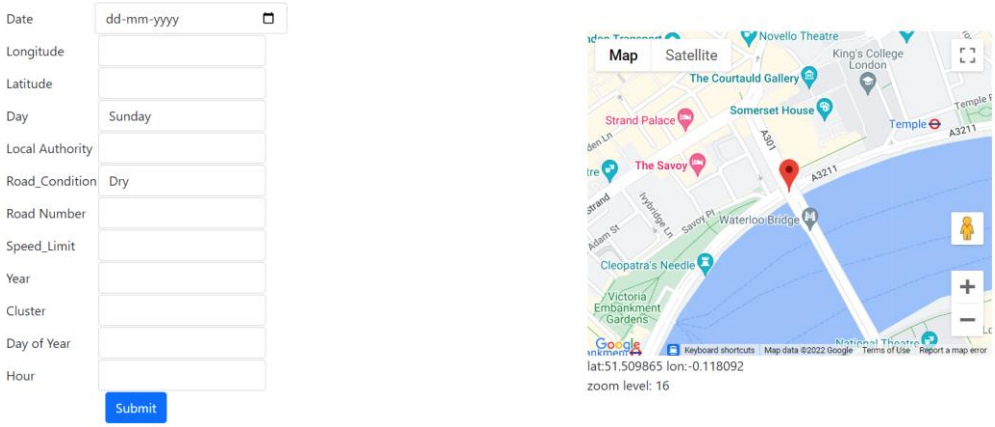


Figure 8.2.4 Upload single data set

5. Admin can view data from this page.

Uploaded data

	Longitude	Latitude	Day_of_Week	Local_Authority_(District)	1st_Road_Class	1st_Road_Number	Speed_limit	Year	Cluster	Day_of_year	Hour
0	-0.185496	51.483253	5	12	3	308	30	2012	1	19	7
1	-0.160418	51.501567	5	1	3	3216	30	2012	2	12	14
2	-0.197303	51.508884	4	12	3	402	30	2012	3	4	8
3	-0.173448	51.481986	7	12	3	3220	30	2012	4	7	23
4	-0.209678	51.516717	7	12	4	450	30	2012	13	35	10
5	-0.195132	51.494552	6	12	3	4	30	2012	5	41	20
6	-0.155861	51.489535	3	12	3	3216	30	2012	28	38	19
7	-0.195423	51.494467	7	12	3	4	30	2012	5	35	22

Figure 8.2.5 View Dataset

6. Login Again option is provided to the admin after the admin logout.

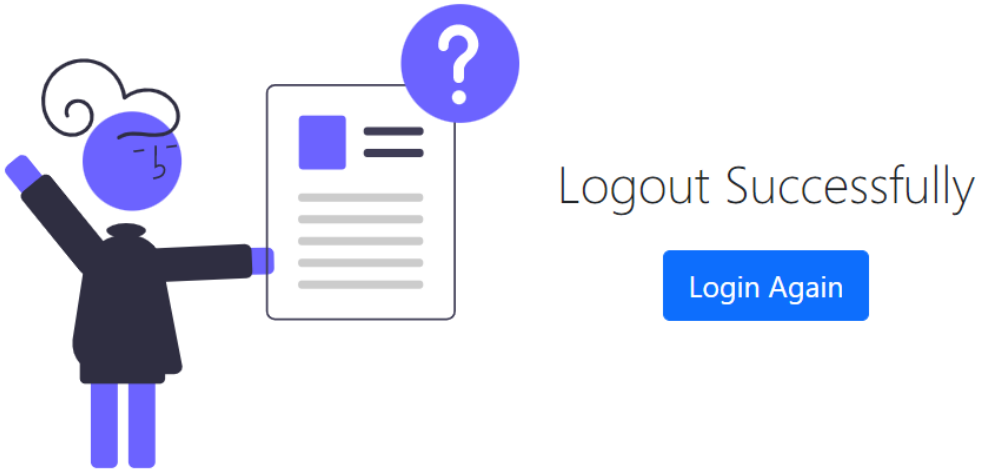


Figure 8.2.6 Logout success page

CONCLUSION

Conclusion

The emerging trends and the availability of intelligent technologies make us to develop new models that help to satisfy the needs of emerging world. So, we have developed a road accident prediction which can possibly contribute to public healthcare. The model proposes an efficient real-time machine learning based framework to automate the process of road accident via random forest classifier using datasets.

Our proposed system provides real-time precaution to the people from accident in their way from origin to destination. This system proposes Google map API to show the route and accidental zone with colored marker.

Addition to all, Admin can be more attentive and can take care of accidental zone if any things happen.

Since this method is highly sensitive to the datasets, admin need to take care of data before uploading into the system. This system works very effectively and efficiently in predicting the road accident between the origin and destination of user and generate the number of accidents happened and accidental zone.

BIBLIOGRAPHY

References

1. https://www.javatpoint.com/machine-learning-random-forest-algorithm
2. https://stackoverflow.com/
3. https://developers.google.com/apis-explorer
4. https://www.kaggle.com/daveianhickey/2000-16-traffic-flow-england-scotland-wales/version/8
5. https://darksky.net/dev
6. https://flask.palletsprojects.com/en/2.1.x/
7. https://mode.com/pythontutorial/libraries/pandas/#:~:text=Pandas%20is%20a%20Python%20library,extremely%20active%20community%20of%20contributors.