**HTML/Javascript**

<html>

<head>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<title>Geocoding service</title>

<style>

html, body, #map-canvas { height: 100%; min-height: 600px; min-width: 700px; margin: 0px; padding: 0px }

#map-canvas { height: 50%; }

#panel { position: absolute; top: 5px; left: 50%; margin-left: -350px; z-index: 5; background-color: #fff; padding: 5px; border: 1px solid #999; }

</style>

<script src="https://maps.googleapis.com/maps/api/js?v=3.exp&sensor=false&libraries=visualization"></script>

</head>

<body>

<div id="panel">

<label>Origin

<input id="origin" type="text" value="">

</label>

<label>Destination

<input id="destination" type="text" value="">

</label>

<input type="button" value="GetDirections" onclick="calcRoute()">

</div>

<div id="control\_panel" style="float:right;width:30%;text-align:left;padding-top:5px">

<div id="routes">

<input id = "r1" type="button" value="Route 1" onclick="displayRoute1()">

<input id = "r2" type="button" value="Route 2" onclick="displayRoute2()">

<input id = "r3" type="button" value="Route 3" onclick="displayRoute3()">

</div>

<div id="total"></div>

</div>

<div id="map-canvas"></div>

<div id="vertex-container">

<label>Points</label>

<ul id="vertex">

</ul>

</div>

<script type="text/javascript" src="R1.js"></script>

<script type="text/javascript">

var directionsDisplay;

var directionsService = new google.maps.DirectionsService();

var map;

var directionsRequest;

var col = ["red","orange","yellow"];

var heatmapDatas,heatmapData= [];

function initialize() {

directionsDisplay = new google.maps.DirectionsRenderer();

var mapOptions = {

zoom: 7,

center: new google.maps.LatLng(48.85727000, 2.35238),

mapTypeId: 'satellite'

};

map = new google.maps.Map(document.getElementById('map-canvas'), mapOptions);

directionsDisplay.setMap(map);

}

function calcRoute() {

var start = document.getElementById('origin').value;

var end = document.getElementById('destination').value;

directionsRequest = {

origin: start,

destination: end,

provideRouteAlternatives : true,

travelMode: google.maps.TravelMode.DRIVING,

unitSystem: google.maps.UnitSystem.METRIC

};

}

function displayRoute1() {

showRoute(0,6,114);

drawHeatMap(heatmapDatas1);

}

function displayRoute2(){

showRoute(1,18,257);

drawHeatMap(heatmapDatas3);

}

function displayRoute3(){

showRoute(2,11,135);

drawHeatMap(heatmapDatas2);

}

function showRoute(i,accpts,accdts){

initialize();

var ul = document.getElementById("vertex");

while (ul.firstChild) {

ul.removeChild(ul.firstChild);

}

directionsService.route(directionsRequest,function (response, status) {

if (status == google.maps.DirectionsStatus.OK) {

new google.maps.DirectionsRenderer({

map: map,

directions: response,

routeIndex: i,

polylineOptions: {strokeColor:col[i]}

});

computeTotalDistance(response,i,accpts,accdts);

var points = response.routes[i].overview\_path;

for (var j = 0; j < points.length; j++) {

var li = document.createElement('li');

li.innerHTML = getLiText(points[j]);

ul.appendChild(li);

}

} else {

//console.log("Directions query failed: " + status);

}

});

}

function drawHeatMap(hmd){

for (var k = 0; k < hmd.length; k++) {

var weightedLoc = {

location: hmd[k],

//weight: Math.pow(2, magnitude[i])

};

heatmapData.push(weightedLoc);

}

heatmap = new google.maps.visualization.HeatmapLayer({

data: heatmapData,

map: map

});

heatmap.setMap(map);

heatmapDatas,heatmapData=[];

}

function getLiText(point) {

var lat = point.lat(),

lng = point.lng();

return lat + "," + lng;

}

function computeTotalDistance(result,h,accpts,accdnts) {

var totalDist = 0;

var totalTime = 0;

var myroute = result.routes[h];

for (i = 0; i < myroute.legs.length; i++) {

totalDist += myroute.legs[i].distance.value;

totalTime += myroute.legs[i].duration.value;

}

totalDist = totalDist / 1000.

document.getElementById("total").innerHTML = "Total Distance is: " + (totalDist/1.6).toFixed(2) + " miles<br>Total time is: " + (totalTime / 60).toFixed(2) + " minutes<br>No. of Accident points: " + accpts + "<br>Total No. of Accidents: " + accdnts;

}

google.maps.event.addDomListener(window, 'load', initialize);

</script>

</body>

</html>

**R Code**

#Reading the Accident Dataset, Accident points and the Traffic Volume

setwd("C:/Users/Raghu/Documents/Predictive Analytics")

accds<-read.csv('NYPD\_Motor\_Vehicle\_Collisions.csv',header=T,sep=',')

ds<-read.csv('Points\_R1.txt',header=T,sep=',')

vol<-read.csv('Traffic\_Volume.csv',header=T,sep=',')

library(plyr)

vol<-rename(vol, c("From"="Start", "To"="End"))

#Retrieving the Streetnames from the traffic volume dataset

q<-"select distinct(Streetname) from vol order by Streetname"

res3<- sqldf(q)

#Calculating the mean of the traffic for each hour for each street

k=0

for(i in 8: ncol(vol)){

res4<-aggregate(vol[,i] ~ STREETNAME, vol, mean)

res4[,2]<-round(res4[,2],digits=2)

res4<- res4[order(res4[,1]),]

res3<-cbind(res3, res4[,2])

names(res3)[ncol(res3)]<-paste(k,":00",sep="")

k=k+1

}

#Exploring the Accident Coordinates and rounding them to 4 decimals

names(ds[,1])<-"Latitude"

names(ds[,2])<-"Longitude"

ds$Location <- "x"

ds[,1]<-round(ds[,1],digits=4)

ds[,2]<-round(ds[,2],digits=4)

ds[,3]<-paste(ds[,1],",",ds[,2],sep="")

names(ds[,3])<-"Location"

ds <- ds[order(ds[,3]),]

#DOing the same for the location coordinates in the Accident Dataset

accds1<-accds[,-c(20:23,26:29)]

accds1<-accds1[!is.na(accds1$LONGITUDE),]

accds1[,5]<-round(accds1[,5],digits=4)

accds1[,6]<-round(accds1[,6],digits=4)

accds1[,7]<-paste(accds1[,5],",",accds1[,6],sep="")

accds1 <- accds1[order(accds1$LOCATION),]

#Subsetting the accident dataset based on accident coordinates

library(sqldf)

q<-"Select a.\* from accds1 a join ds b on a.Location = b.Location"

res1<- sqldf(q)

#Calculating the count of accidents per each accident location

q<-"Select Location, count(Location) from res1 group by Location"

res2<- sqldf(q)

#Transforming the Subset Dataset to our needs

out <- strsplit(as.character(res1$TIME),':')

out<-do.call(rbind, out)

out<-data.frame(out)

res1<-cbind(res1,out[,1])

names(res1)[ncol(res1)]<-"Hour"

names(res1)[8]<-"Street\_Name"

#Extracting the Street names and the time

q<-"Select \* from res1 Order by Street\_Name,Hour"

res1<-sqldf(q)

#Selecting the data based on the streetnames in the earlier dataset

q<-"Select \* from res3 where STREETNAME in (Select distinct Street\_Name from res1) "

streets<-sqldf(q)

#Transposing the streetnames

temp<-streets[,-1]

temp<-t(temp)

temp<-data.frame(temp)

col\_temp=ncol(temp)

for(i in 1:24){

temp[i,col\_temp+1] <-i-1

}

names(temp)[ncol(temp)]<-"Hour"

#Selecting the Streetnames from the resultant dataset

q<-"Select STREETNAME from res3 where STREETNAME in (Select distinct Street\_Name from res1) "

streetnames<-sqldf(q)

temp\_base<-data.frame()

z=1

for(x in 1:nrow(streetnames)){

for(y in 1:nrow(temp) ){

temp\_base[z,1]<-streetnames[x,1]

temp\_base[z,2]<-temp[y,ncol(temp)]

temp\_base[z,3]<-temp[y,x]

z=z+1

}

}

#Writing the resultant dataset to a temporary file

write.csv(res1,"temp.csv")

####################################################

#Reading the temporary file

res1<-read.csv('temp.csv',header=T,sep=',')

res1<-res1[,-c(1:7)]

#Merging the datasets based on Streetnames

q<-"Select a.\*,b.V3 from res1 a inner join temp\_base b on a.Street\_name=b.V1 and a.Hour=b.V2 "

res8<-sqldf(q)

#Calculating the total injured

res8[,18]<-res8[,5]+res8[,7]+res8[,9]+res8[,11]

names(res8)[18]<-"Total Injured"

#Calculating the total killed

res8[,19]<-res8[,6]+res8[,8]+res8[,10]+res8[,12]

names(res8)[19]<-"Total Killed"

#Writing it to a temporary file

write.csv(res8,"temp1.csv")

#Calculating Overall mean volume, Total Accidents and no of accident points

res9<-res8[,c(13,15:19)]

Mean\_Vol<-mean(res9$V3)

Tot\_Acc<-nrow(res9)

colnames(res9)<-c("Cont\_Vehicle","Veh\_Type","Hour","Mean\_Vol","Total\_Injured","Total\_Killed")

Acc\_points\_T<-sqldf("Select count(distinct(LOCATION)) from res8 where hour = 19")

#Subsetting the dataset at time 7 PM and calculating the above parameters

res9<-sqldf("Select \* from res9 where hour = 19")

Mean\_Vol\_at\_T<-mean(res9[,4])

Tot\_Acc\_at\_T<-nrow(res9)

#Determining the Accident density

Acc\_density\_at\_T<-round((Tot\_Acc\_at\_T \* Mean\_Vol)/(Mean\_Vol\_at\_T \* Tot\_Acc ),digits=2)

names(res1)[13]<-"Cont\_Vehicle"

names(res1)[15]<-"Veh\_Type"

#Calculating the overall and at 7 PM , the mean frequency of 'Contributing Factor'

Veh1<-sqldf("Select Cont\_Vehicle,count(Cont\_Vehicle) from res9 group by Cont\_Vehicle order by Cont\_Vehicle")

Veh2<-sqldf("Select Cont\_Vehicle,count(Cont\_Vehicle) from res1 group by Cont\_Vehicle order by Cont\_Vehicle")

Veh3<-sqldf("Select \* from Veh1 a inner join Veh2 b on a.Cont\_Vehicle=b.Cont\_Vehicle")

Veh3<-Veh3[,-3]

Veh3[,4]<-round((Veh3[,2]/Veh3[,3]),digits=2)

Veh3<-Veh3[,-c(2,3)]

names(Veh3)[2]<-"Cont\_Vehicle\_Frequency"

res9<-res9[order(res9$Cont\_Vehicle),]

res9<-sqldf("Select a.\* ,b.Cont\_Vehicle\_Frequency from res9 a inner join Veh3 b where a.Cont\_Vehicle=b.Cont\_Vehicle")

#Calculating the overall and at 7 PM , the mean frequency of 'Vehicle Type'

Veh1<-sqldf("Select Veh\_Type,count(Veh\_Type) from res9 group by Veh\_Type order by Veh\_Type")

Veh2<-sqldf("Select Veh\_Type,count(Veh\_Type) from res1 group by Veh\_Type order by Veh\_Type")

Veh3<-sqldf("Select \* from Veh1 a inner join Veh2 b on a.Veh\_Type=b.Veh\_Type")

Veh3<-Veh3[,-3]

Veh3[,4]<-round((Veh3[,2]/Veh3[,3]),digits=2)

Veh3<-Veh3[,-c(2,3)]

names(Veh3)[2]<-"Veh\_Type\_Frequency"

res9<-res9[order(res9$Veh\_Type),]

res9<-sqldf("Select a.\* ,b.Veh\_Type\_Frequency from res9 a inner join Veh3 b where a.Veh\_Type=b.Veh\_Type")

#Removing Unwanted Variables

rm(Veh1,Veh2,Veh3)

res9<-res9[,-c(1:3,5:6)]

#Normalizing the variables

for(i in 1:nrow(res9)){

res9[i, 1] <- (res9[i,1]-min(res9[,1:3]))/ max(res9[,1:3]) - (min(res9[,1:3]))

}

#Prinicipal Component Analysis

pca\_list<-vector("list",nrow(res9))

pca\_list[[i]]<-princomp(res9[, 1:3])

loadings\_df <- pca\_list[[length(pca\_list)]]$loadings[,]

loadings\_df[,]<-abs(loadings\_df[,])

#Calculating the RSI's for each prinicpal component

RSI<-data.frame()

for(i in 1:3){

x<-as.matrix(loadings\_df[,i])

for(j in 1: nrow(res9)){

y<-as.matrix(res9[j,])

z<-y %\*% x

RSI[j,i]<-Acc\_density\_at\_T + z[1,1]

}

}

#The mean RSI for the route

RSI<-c(mean(RSI[,1]),mean(RSI[,2]),mean(RSI[,3]))