databricksFinal Project For big data

# 1. Locating file in DBFS

```
%fs ls /mnt/data
```

| | path | name | size |
|---|---|---|---|
| 1 | dbfs:/mnt/data/big_data/ | big_data/ | 0 |
| 2 | dbfs:/mnt/data/cab.csv | cab.csv | 4252216 |

Showing all 2 rows.

# 1. Cab trips in the city


Due to high competition in the taxi business with companies like uber dominating the market it is becoming increasingly difficult for normal drivers to make a profit, cab dataset will be analyzed in order to predicted in which areas the drivers will make most money

With the help of random forest method , regression trees we will predict driving in which areas might result in better profits

```
#Importing spark library for R and Dyplr from tidyverse for data manipulation
library(sparklyr)
library(dplyr)

#Starting a spark connection in R using the sparklyr package
sc <- spark_connect(method = "databricks")

#reading file from azure data lake
cab <- spark_read_csv(sc,'/mnt/data/cab.csv')
```

```
#lets take a look at the first five sample rows of the dataset
head(cab,5)
```

```
# Source: spark<?> [?? x 7]
  medallion pickup_datetime     pickup_longitude pickup_latitude
  <chr>     <dttm>                         <dbl>           <dbl>
1 4D24F4D8… 2013-01-13 10:23:00            -73.9            40.8
2 A49C37EB… 2013-01-13 04:52:00            -74.0            40.7
3 1E4B72A8… 2013-01-13 10:47:00            -74.0            40.8
4 F7E4E943… 2013-01-13 11:14:00            -74.0            40.7
5 A9DC75D5… 2013-01-13 11:24:00            -74.0            40.8
# … with 3 more variables: trip_time_in_secs <int>, fare_amount <dbl>,
#   tip_amount <dbl>
```

# 2. Transforming Data

As you can see above, the `cab` dataset has 7 variables , we will focus on variables for location, price and number of trips done by the cab.

In order to work efficiently we will rename existing variable names into smaller ones to make the coding process easier and less time consuming, computing column total

```
cab <- cab %>%
   rename(long = pickup_longitude, lat = pickup_latitude)  %>%  #renaming variables for ease of use
   filter(fare_amount > 0 | tip_amount > 0) %>% #filter NA values not needed in further analysis
   mutate(total = log(fare_amount + tip_amount) ) #computing and adding new variable total
```

# 3. Emphasising on the busiest district in the city

The dataset contains several locations from all over the city however focus will be on the just one disctrict of the city

```
cab <- cab  %>%
    filter(between(lat,  40.70, 40.83) &  #filtering longitude and latitude for mapping
           between(long, -74.025, -73.93))
```

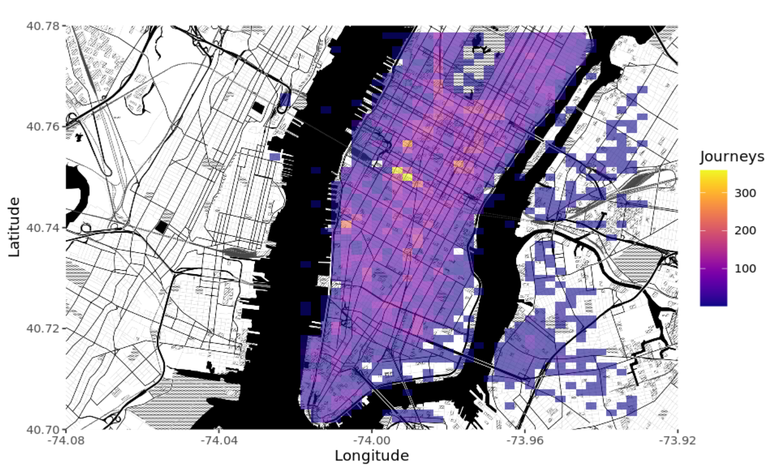
# 4. Plotting trips initiated on the city map

We will plot data on the map using `ggmap` and `ggplot2` to visualize where in district people begin their journeys.

```
#first installing packages essential for plotting maps in databricks because these are not included in the databricks R library
install.packages("ggmap")

install.packages("viridis")
```

Show result

```
#Loading libraries
library(ggmap)
library(viridis)

#Connecting google map api for extracting the map data from google maps
register_google(key = "AIzaSyDdqW6TzpU4YxLgmWgH4DDmOzM-2EThuZY") #this google key will not be valid later because it is private key generated for the project

#creating a variable with coordinates for extracting map from Google maps
city <- c(left = -74.08, bottom = 40.70, right = -73.92, top = 40.78)

#plotting map using ggmaps
ggmap(get_stamenmap(city, zoom = 14, maptype = "toner-background")) +
  scale_fill_viridis(option='plasma') +
  geom_bin2d(data = cab, aes(x = long, y = lat), bins = 60, alpha = 0.6) +
  labs(x = 'Longitude', y = 'Latitude', fill = 'Journeys')

#We can see highest number of trips are started in the uptown area of the district
```




Because wday and month sparklyr variants are not available in Spark SQL we will import the data to compute locally

```
#importing data for computing locally in the R notebook
cab1 <- sdf_collect(cab)
head(cab1,5)
```

```
# A tibble: 5 x 8
  medallion pickup_datetime      long   lat trip_time_in_se… fare_amount
  <chr>     <dttm>              <dbl> <dbl>            <int>       <dbl>
1 4D24F4D8… 2013-01-13 10:23:00 -73.9  40.8              600         8
2 A49C37EB… 2013-01-13 04:52:00 -74.0  40.7              840        18
3 1E4B72A8… 2013-01-13 10:47:00 -74.0  40.8               60         3.5
4 F7E4E943… 2013-01-13 11:14:00 -74.0  40.7              720        11.5
5 A9DC75D5… 2013-01-13 11:24:00 -74.0  40.8              240         6.5
# … with 2 more variables: tip_amount <dbl>, total <dbl>
```

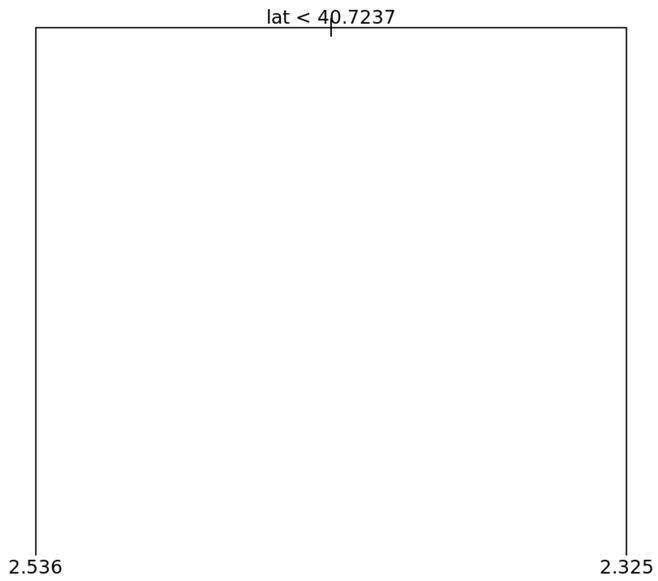
## 5. Predicting cab charges using a tree

Regression tree will be used in prder to find cutpoints in the predictors i.e longitude and latitude, in order to enhance the predictive capability

```
#tree package has to be installed in databricks R notebook it is not preloaded in the databricks R package library
install.packages("tree")
```

Show result

```
library(tree)

# regression tree
fitted_tree <- tree(total ~ lat + long, data = cab1)

# structure of regression tree
plot(fitted_tree)
text(fitted_tree)

summary(fitted_tree)
```




## 6. Making dates easy to work with & tidying data for another tree

We will mutate our time variables into integers so that we can apply them into our model

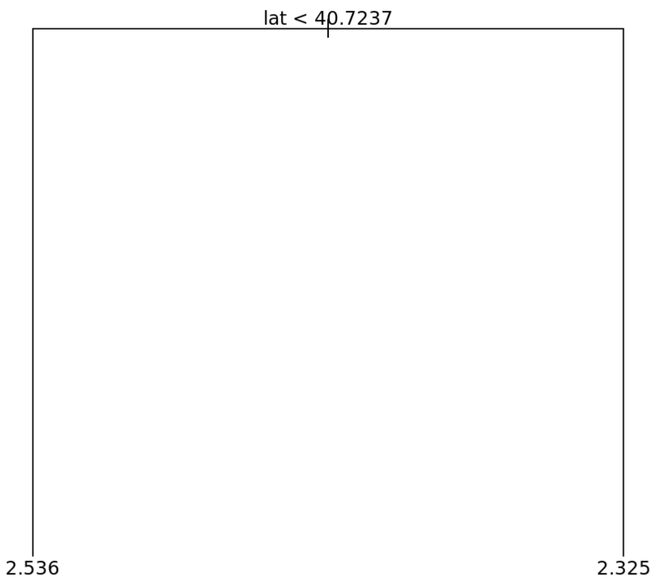
Before we only included one split where it took only `lat` into account and it precidited that where `lat < 40.7237` the rides are more expensive, which is the city downtown. But to make our prediction more precise we will take into consideration time variables

```
#This package is used to turn dates into integers
library(lubridate)

# tidying variables for ease of computing
cab1 <- cab1 %>%
    mutate(hour = hour(pickup_datetime),
           wday = wday(pickup_datetime, label = TRUE),
           month = month(pickup_datetime, label = TRUE))
```

```
Attaching package: ‘lubridate’

The following objects are masked from ‘package:dplyr’:

    intersect, setdiff, union

The following objects are masked from ‘package:base’:

    date, intersect, setdiff, union
```

## 7. Aplpying the new variables to another tree

```
# applying model again
fitted_tree <- tree(total ~ lat + long + hour + wday + month, data = cab1)

# structure of the tree
plot(fitted_tree)
text(fitted_tree)
```

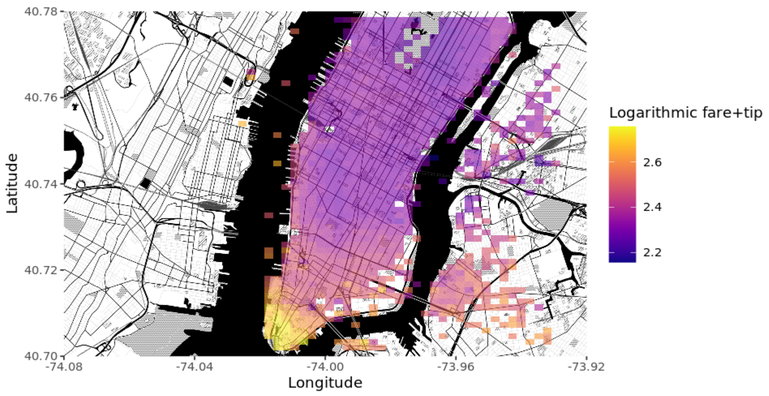



## 9. Plotting the predicted fare

```
library(randomForest)

# applying randomForest model
fitted_forest <- randomForest(total ~ lat + long + hour + wday + month,
    data=cab1, ntree=80, sampsize=10000)

# Printing the fitted_forest object
fitted_forest
```

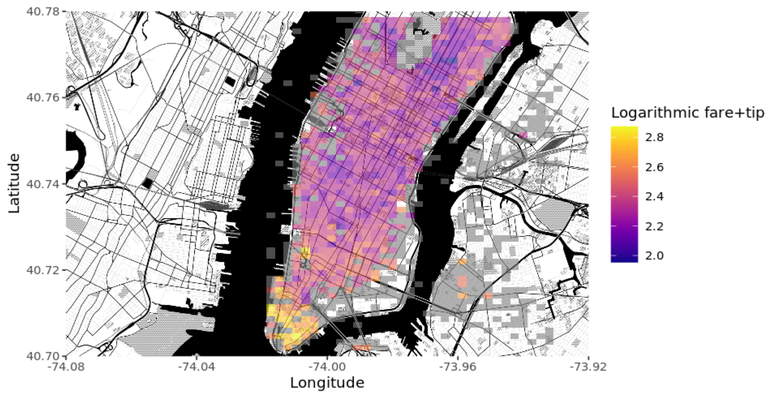
Show result

```
# Extracting the prediction from forest_fit
cab1$pred_total <- fitted_forest$predicted

# Plotting the predicted mean trip prices from according to the random forest
ggmap(get_stamenmap(city, zoom = 14, maptype = "toner-background")) +
    scale_fill_viridis(option = 'plasma') +
    stat_summary_2d(data=cab1, aes(x = long, y = lat, z = pred_total),
                    fun = mean, alpha = 0.6, bins = 60) +
    labs(x = 'Longitude', y = 'Latitude', fill = 'Logarithmic fare+tip')
```




## 10. Displaying fare from raw dataset with real fare prices and comparing with the predicted model

```
# Function that returns the mean *if* there are 15 or more datapoints
mean_if_enough_data <- function(x) {
    ifelse( length(x) >= 15, mean(x), NA)
}

# Plotting the mean trip prices from the data
ggmap(get_stamenmap(city, zoom = 14, maptype = "toner-background")) +
    stat_summary_2d(data=cab, aes(x = long, y = lat, z = total),
                    fun = mean_if_enough_data,
                    alpha = 0.6, bins = 60) +
  scale_fill_viridis(option = 'plasma') +
  labs(x = 'Longitude', y = 'Latitude', fill = 'Logarithmic fare+tip')
```




Our model predicted that fares are higher in the downtown area and in midtown it will be average, eventhough the actual fare data is different for midtown area than our predicted model, the prediction that downtown has high fares correlates very closely to the actual fares prices

## 11. Highest number of trips taken by the public are in the downtown area