In [1]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
from datetime import datetime
import matplotlib.pyplot as plt
from scipy.stats.mstats import mode
```

In [2]:
```
driver_id_path = 'driver_ids.csv'
ride_id_path = 'ride_ids.csv'
timestamps_path = 'ride_timestamps.csv'

driver_ids = pd.read_csv(driver_id_path)
ride_ids = pd.read_csv(ride_id_path)
timestamps = pd.read_csv(timestamps_path)
```

General Statistics on the Driver and Rides

In [3]:
```
sep = ride_ids[['driver_id', 'ride_id']]

print("Number of unique drives", len(np.unique(sep['driver_id'])))
print('total number of rides', sep.shape[0])
grouped = sep.groupby('driver_id').count()
print("max number of rides by a single driver", max(grouped['ride_id']))
grouped.describe()
```

```
Number of unique drives 937
total number of rides 193502
max number of rides by a single driver 919
```

Out[3]:

	ride_id
count	937.000000
mean	206.512273
std	173.254063
min	3.000000
25%	47.000000
50%	200.000000
75%	316.000000
max	919.000000

Driver Prime Time

In [4]:
```
driver_prime = ride_ids[["driver_id", "ride_prime_time"]]
driver_prime_count = driver_prime.groupby('ride_prime_time').count()
driver_prime_count.rename(columns={"driver_id": "number_of_rides"}, inpl
ace=True)
driver_prime_count.reset_index(level=0, inplace=True)
print("Number of Rides done in each prime time")
driver_prime_count
```

```
Number of Rides done in each prime time
```

Out[4]:

	ride_prime_time	number_of_rides
0	0	125412
1	25	33677
2	50	17712
3	75	8208
4	100	6216
5	150	1686
6	200	432
7	250	101
8	300	31
9	350	15
10	400	11
11	500	1

Calculating Driver Rentention

In [5]:
```
# Get only one event for each ride
driv_ride = ride_ids[['driver_id', 'ride_id']]
uniq_rides = timestamps[timestamps['event'] == 'accepted_at'][['ride_id'
, 'timestamp']]
# Matches rides with driver id
driv_ride_time = pd.merge(uniq_rides, driv_ride, on='ride_id')
print("Number of rides is", driv_ride_time.shape[0])
driv_ride_time.head()
```

```
Number of rides is 184819
```

Out[5]:

	ride_id	timestamp	driver_id
0	00003037a262d9ee40e61b5c0718f7f0	2016-06-13 09:39:51	d967f5296732fa55266b5f1314e7447b
1	00005eae40882760d675da5effb89ae3	2016-05-14 05:23:25	0656192a402808805282e60761bda088
2	000061d42cf29f73b591041d9a1b2973	2016-05-16 15:43:14	c468a648519cd42da75e6aa9dadf733e
3	00006efeb0d5e3ccad7d921ddeee9900	2016-05-11 19:29:43	689bdf87fb2de49f98bf4946cfaa5068
4	00012759befd5d34a0609800f6a1ee59	2016-05-31 15:45:05	eece82fe623b4bb335a9b9e20eb0ca54

In [6]:
```
# gets each driver's latest ride
latest_ride = driv_ride_time.groupby('driver_id').max()
latest_ride.rename(columns={'timestamp':'latest_ride'}, inplace=True)
latest_ride.drop('ride_id', axis=1, inplace=True)

# gets each driver's oldest ride
oldest_ride = driv_ride_time.groupby('driver_id').min()
oldest_ride.rename(columns={'timestamp':'oldest_ride'}, inplace=True)
oldest_ride.drop('ride_id', axis=1, inplace=True)

# Dataframe of driver id, oldest ride time, and latest ride time
retention_raw = pd.merge(oldest_ride, latest_ride, on='driver_id')
retention_raw.head()
```

Out[6]:

driver_id	oldest_ride	latest_ride
002be0ffdc997bd5c50703158b7c2491	2016-03-29 18:47:01	2016-06-23 10:06:30
007f0389f9c7b03ef97098422f902e62	2016-03-29 22:28:34	2016-06-22 13:17:44
011e5c5dfc5c2c92501b8b24d47509bc	2016-04-05 10:55:00	2016-06-12 20:22:27
0152a2f305e71d26cc964f8d4411add9	2016-04-25 15:59:35	2016-06-26 10:16:39
01674381af7edd264113d4e6ed55ecda	2016-04-29 07:50:47	2016-06-24 13:03:42

In [7]:
```
# Converts Oldest and Latest ride times to datatime objects for easier c
omparison
to_date_obj = lambda x: datetime.strptime(x, '%Y-%m-%d %H:%M:%S').date()
oldest_date_obj = retention_raw['oldest_ride'].apply(to_date_obj)
latest_date_obj = retention_raw['latest_ride'].apply(to_date_obj)
retention = pd.concat([oldest_date_obj,latest_date_obj], axis=1)

retention['retention_period (in days)'] = (retention['latest_ride'] - re
tention['oldest_ride']).dt.days
```

In [8]:
```
# Creates DataFrame with only the Driver Id and the Retention Period
driver_v_retention = retention.filter(['driver_id','retention_period (in
days)'], axis=1)
driver_v_retention.reset_index(level=0, inplace=True)

print("There are " + str(driver_v_retention.shape[0])
      + " rows in the DataFrame and " + str(len(np.unique(driver_v_reten
tion['driver_id'])))
      + " unique drivers")
driver_v_retention.head()
```

```
There are 844 rows in the DataFrame and 844 unique drivers
```

Out[8]:

	driver_id	retention_period (in days)
0	002be0ffdc997bd5c50703158b7c2491	86
1	007f0389f9c7b03ef97098422f902e62	85
2	011e5c5dfc5c2c92501b8b24d47509bc	68
3	0152a2f305e71d26cc964f8d4411add9	62
4	01674381af7edd264113d4e6ed55ecda	56

In [9]:
```
#Groups drivers by their retention period
driver_retention_count = driver_v_retention.groupby('retention_period (i
n days)').count()
driver_retention_count.reset_index(level=0, inplace=True)
```

In [10]:
```
# driver_retention_count.to_csv('driver_ren.csv', index=False)
```

Driver Statistics

In [11]:
```
# The statistics for every single driver
mean_stats_by_driver = ride_ids.groupby("driver_id").mean()
mean_stats_by_driver.reset_index(level=0, inplace=True)
mean_stats_by_driver.rename(columns=
                            {"ride_distance": "avg_ride_distance",
                             "ride_duration": "avg_ride_duration",
                             "ride_prime_time": "avg_ride_prime_time"},
inplace=True)
mean_stats_by_driver.head()
```

Out[11]:

	driver_id	avg_ride_distance	avg_ride_duration	avg_ride_prime_time
0	002be0ffdc997bd5c50703158b7c2491	6282.624549	798.693141	19.404332
1	007f0389f9c7b03ef97098422f902e62	3791.322581	661.193548	20.161290
2	011e5c5dfc5c2c92501b8b24d47509bc	7930.970588	858.970588	19.852941
3	0152a2f305e71d26cc964f8d4411add9	7702.821990	913.722513	10.732984
4	01674381af7edd264113d4e6ed55ecda	8329.717333	953.181333	12.533333

Calculating the Formula

Prime Time Percentile

In [12]:
```
prime_time = ride_ids[['driver_id', 'ride_prime_time']]
prime_time_sum = prime_time.groupby("driver_id").sum().sort_values(by="ride_prime_time", ascending=False)
highest_prime_t = prime_time_sum.iloc[0][0]
percentile = lambda x: x / highest_prime_t
prime_time_sum['prime_time_percentile'] = prime_time_sum['ride_prime_time'].apply(percentile)
```

In [13]:
```
prime_time_percentile = prime_time_sum.drop(columns='ride_prime_time').sort_values(by='driver_id')
prime_time_percentile.head()
```

Out[13]:

driver_id	prime_time_percentile
002be0ffdc997bd5c50703158b7c2491	0.273189
007f0389f9c7b03ef97098422f902e62	0.031766
011e5c5dfc5c2c92501b8b24d47509bc	0.034307
0152a2f305e71d26cc964f8d4411add9	0.104193
01674381af7edd264113d4e6ed55ecda	0.238882

Driver Retention (Sorted)

In [14]:
```
# Sorted Driver Retention by Driver id
driv_rent = driver_v_retention
driv_rent.sort_values(by='driver_id', inplace=True)

max_driv_rent = driv_rent['retention_period (in days)'].max()
driv_rent_perc = lambda x : x / max_driv_rent

driv_rent['rent_perc'] = driv_rent['retention_period (in days)'].apply(driv_rent_perc)
driv_rent = driv_rent[['driver_id', 'rent_perc']]

driv_rent.head()
```

Out[14]:

	driver_id	rent_perc
0	002be0ffdc997bd5c50703158b7c2491	0.955556
1	007f0389f9c7b03ef97098422f902e62	0.944444
2	011e5c5dfc5c2c92501b8b24d47509bc	0.755556
3	0152a2f305e71d26cc964f8d4411add9	0.688889
4	01674381af7edd264113d4e6ed55ecda	0.622222

Number of Rides per Driver

In [15]:
```
num_rides_per_driver = ride_ids[['driver_id', 'ride_id']].groupby('drive
r_id').count()
num_rides_per_driver.rename(columns={'ride_id' : 'number_of_rides'}, inp
lace=True)
num_rides_per_driver.reset_index(level=0, inplace=True)
num_rides_per_driver.sort_values(by='driver_id', inplace=True)

max_num_rides = num_rides_per_driver['number_of_rides'].max()
num_rides_percentile = lambda x : x / max_num_rides
num_rides_per_driver['num_rides_percentile'] = num_rides_per_driver['num
ber_of_rides'].apply(num_rides_percentile)

num_rides_per_driver = num_rides_per_driver[['driver_id', 'num_rides_per
centile']]

num_rides_per_driver.head()
```

Out[15]:

	driver_id	num_rides_percentile
0	002be0ffdc997bd5c50703158b7c2491	0.301415
1	007f0389f9c7b03ef97098422f902e62	0.033732
2	011e5c5dfc5c2c92501b8b24d47509bc	0.036997
3	0152a2f305e71d26cc964f8d4411add9	0.207835
4	01674381af7edd264113d4e6ed55ecda	0.408052

Driver Distance Sum

In [16]:
```
sum_ride_dist = ride_ids[['driver_id', 'ride_distance']]
sum_ride_dist = sum_ride_dist.groupby('driver_id').sum().sort_values(by=
'driver_id')
max_ride_dist = sum_ride_dist['ride_distance'].max()

dist_percentile = lambda x : x / max_ride_dist

sum_ride_dist['dist_percentile'] = sum_ride_dist['ride_distance'].apply(
dist_percentile)
sum_ride_dist.drop(columns='ride_distance', inplace=True)

sum_ride_dist.head()
```

Out[16]:

driver_id	dist_percentile
002be0ffdc997bd5c50703158b7c2491	0.262558
007f0389f9c7b03ef97098422f902e62	0.017732
011e5c5dfc5c2c92501b8b24d47509bc	0.040683
0152a2f305e71d26cc964f8d4411add9	0.221967
01674381af7edd264113d4e6ed55ecda	0.471266

Time to Accept Request in Seconds

In [17]:
```
# # Get requested at and accepted at events for each ride
# time_filt = timestamps[(timestamps['event'] == 'accepted_at') | (times
tamps['event'] == 'requested_at')][['ride_id', 'event','timestamp']]
# ride_filt = ride_ids[['driver_id', 'ride_id']]

# # Matches rides with driver id
# time_event = pd.merge(ride_filt, time_filt, on='ride_id')
# to_time_obj = lambda x: pd.to_datetime(x, format = '%Y-%m-%d %H:%M:%
S') # datetime obj w/h secs
# time_convt = time_event['timestamp'].apply(to_time_obj)
```

In [18]:
```
# time_event['timestamp'] = time_convt
# updated_t_e = time_event[['driver_id', 'ride_id', 'timestamp']]
# # grouped_time_event = updated_t_e.groupby('ride_id').diff()
# # grouped_time_event
```

In [19]:
```
req_to_acc = pd.read_csv('request_to_accept.csv')
req_to_acc = req_to_acc[['ride_id', 'wait_num']]

driv_rid = ride_ids[['driver_id', 'ride_id']]

wait_driv = pd.merge(req_to_acc, driv_rid, on='ride_id')
max_wait = wait_driv['wait_num'].max()

wait_percentile = lambda x : x / max_wait

wait_driv['wait_percentile'] = wait_driv['wait_num'].apply(wait_percenti
le)
wait_driv = wait_driv[['driver_id', 'wait_percentile']]
```

Putting the Formula Together

In [20]:
```
comp_driv = pd.merge(prime_time_percentile, num_rides_per_driver, on='dr
iver_id')
comp_driv = pd.merge(comp_driv, driv_rent, on='driver_id')
# comp_driv.drop(columns='retention_period (in days)_x') <-- Was getting
an extra column earlier
comp_driv.rename(columns={'retention_period (in days)' : 'rentention_in_
days'},
                 inplace=True)
comp_driv = pd.merge(comp_driv, sum_ride_dist, on='driver_id')
comp_driv = pd.merge(comp_driv, wait_driv, on='driver_id')
comp_driv.head()
```

Out[20]:

	driver_id	prime_time_percentile	num_rides_percentile	rent_perc	dist
0	002be0ffdc997bd5c50703158b7c2491	0.273189	0.301415	0.955556	
1	002be0ffdc997bd5c50703158b7c2491	0.273189	0.301415	0.955556	
2	002be0ffdc997bd5c50703158b7c2491	0.273189	0.301415	0.955556	
3	002be0ffdc997bd5c50703158b7c2491	0.273189	0.301415	0.955556	
4	002be0ffdc997bd5c50703158b7c2491	0.273189	0.301415	0.955556	

In [21]:
```
pt_weight = comp_driv['prime_time_percentile'] * 25/100
# rides_weight = comp_driv['num_rides_percentile'] * 25/100
ride_length_weight = comp_driv['num_rides_percentile']/comp_driv['dist_p
ercentile'] * 25/100
rent_weight = comp_driv['rent_perc'] * 25/100
# dist_weight = comp_driv['dist_percentile'] * 25/100
wait_weight = comp_driv['wait_percentile'] * 25/100

comp_driv['formula'] = pt_weight + rent_weight - wait_weight + ride_leng
th_weight
print("min: " + str(comp_driv['formula'].min()) + " max: " + str(comp_dr
iv['formula'].max()))

formula_min = comp_driv['formula'].min()
min_shift = lambda x : x + formula_min * -1
comp_driv['shifted_formula'] = comp_driv['formula'].apply(min_shift)
print("new min: " + str(comp_driv['shifted_formula'].min()) + " new max:
" + str(comp_driv['shifted_formula'].max()))

comp_driv.head()
```

```
min: -0.015094850611428207 max: 0.7708769583584127
new min: 0.0 new max: 0.785971808969841
```

Out[21]:

	driver_id	prime_time_percentile	num_rides_percentile	rent_perc	dist
0	002be0ffdc997bd5c50703158b7c2491	0.273189	0.301415	0.955556	
1	002be0ffdc997bd5c50703158b7c2491	0.273189	0.301415	0.955556	
2	002be0ffdc997bd5c50703158b7c2491	0.273189	0.301415	0.955556	
3	002be0ffdc997bd5c50703158b7c2491	0.273189	0.301415	0.955556	
4	002be0ffdc997bd5c50703158b7c2491	0.273189	0.301415	0.955556	

In [22]:
```
comp_driv.to_csv('shifted_driver_formula.csv', index=False)
```

Do All Drivers act alike?

In [23]:
```
# Get requested at and accepted at events for each ride
time_filt = timestamps[(timestamps['event'] == 'accepted_at')][['ride_i
d', 'event','timestamp']]
ride_filt = ride_ids[['driver_id', 'ride_id']]

# Matches rides with driver id
time_event = pd.merge(ride_filt, time_filt, on='ride_id')
to_time_obj = lambda x: pd.to_datetime(x, format = '%Y-%m-%d %H:%M:%S').
time() # datetime obj w/h secs
time_convt = time_event['timestamp'].apply(to_time_obj)

time_event['timestamp'] = time_convt
time_event = time_event[['driver_id','timestamp']]
```

In [32]:
```
# grouped = time_event.groupby('driver_id')
# f = lambda x: mode(x)[0]
# for i in grouped:
#     a_driv_ts = i[1]['timestamp']
# #     print(a_driv_ts.mode())

#     print(a_driv_ts.mode())
#     break
```

In [ ]:

In [ ]: