Assignment_lyft_ZZ

July 30, 2019

1 Lyft Analytics Assignment

1.1 The goal is to recommend a Driver's Lifetime Value.

2 Table of Content

2.1 Part 1. Load the Data, overview of three tables

2.2 Part 2. Join dataframes

2.3 Part 3. Exploratory Data Analysis

2.4 Part 4. Modelling

```
In [137]: import sys
          import numpy as np
          import pandas as pd
          import datetime as dt
          from datetime import timedelta
          import math
          import matplotlib.pyplot as plt
          import seaborn as sns

          %matplotlib inline
          plt.style.use('fivethirtyeight')
```

2.5 Part 1. Load the Data, overview of three tables

```
In [138]: drivers = pd.read_csv('driver_ids.csv')
          rides_id = pd.read_csv('ride_ids.csv')
          rides_times = pd.read_csv('ride_timestamps.csv')
```

2.6 driver_ids.csv has

- driver_id: Unique identifier for a driver
- driver_onboard_date: Data on which driver was on_board

insights - 1. 937 drivers have an onboard date

2. There is no missing values, so we don't need to deal with missing values.However driver_onboard_date is not as a datetime format, therefore we need to convert it into a date-time format.

3. The driver onboard date varies between: '2016-03-28 00:00:00' and '2016-05-15 00:00:00' Take a look at driver_ids.csv

```
In [139]: drivers.head(10)
```

Out[139]:

	driver_id	driver_onboard_date
0	002be0ffdc997bd5c50703158b7c2491	2016-03-29 00:00:00
1	007f0389f9c7b03ef97098422f902e62	2016-03-29 00:00:00
2	011e5c5dfc5c2c92501b8b24d47509bc	2016-04-05 00:00:00
3	0152a2f305e71d26cc964f8d4411add9	2016-04-23 00:00:00
4	01674381af7edd264113d4e6ed55ecda	2016-04-29 00:00:00
5	01788cf817698fe68eaecd7eb18b2f72	2016-05-06 00:00:00
6	0213f8b59219e32142711992ca4ec01f	2016-04-07 00:00:00
7	021e5cd15ef0bb3ec20a12af99e142b3	2016-05-07 00:00:00
8	0258e250ca195cc6258cbdc75aecd853	2016-04-26 00:00:00
9	028b5a4dcd7f4924ebfabcf2e814c014	2016-05-06 00:00:00

```
In [140]: drivers.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 937 entries, 0 to 936
Data columns (total 2 columns):
driver_id              937 non-null object
driver_onboard_date    937 non-null object
dtypes: object(2)
memory usage: 14.7+ KB


In [141]: drivers['driver_onboard_date']=pd.to_datetime(drivers['driver_onboard_date'])

In [142]: drivers.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 937 entries, 0 to 936
Data columns (total 2 columns):
driver_id              937 non-null object
driver_onboard_date    937 non-null datetime64[ns]
dtypes: datetime64[ns](1), object(1)
memory usage: 14.7+ KB


In [143]: drivers.head()
```

Out[143]:

	driver_id	driver_onboard_date
0	002be0ffdc997bd5c50703158b7c2491	2016-03-29

```
1  007f0389f9c7b03ef97098422f902e62          2016-03-29
2  011e5c5dfc5c2c92501b8b24d47509bc          2016-04-05
3  0152a2f305e71d26cc964f8d4411add9          2016-04-23
4  01674381af7edd264113d4e6ed55ecda          2016-04-29
```

```
In [144]: drivers["driver_onboard_date"].max()
```

```
Out[144]: Timestamp('2016-05-15 00:00:00')
```

```
In [145]: drivers["driver_onboard_date"].min()
```

```
Out[145]: Timestamp('2016-03-28 00:00:00')
```

2.7 rides_id.csv has

- driver_id Unique identifier for a driver
- ride_id Unique identifier for a ride that was completed by the driver
- ride_distance Ride distance in meters
- ride_duration Ride durations in seconds
- ride_prime_time PrimeTime applied on the ride

Insights - 1.There is no missing values

2. There are 193,502 trips in total.

```
In [146]: rides_id.head(10)
```

```
Out[146]:                          driver_id                          ride_id  \
0  002be0ffdc997bd5c50703158b7c2491  006d61cf7446e682f7bc50b0f8a5bea5
1  002be0ffdc997bd5c50703158b7c2491  01b522c5c3a756fbdb12e95e87507eda
2  002be0ffdc997bd5c50703158b7c2491  029227c4c2971ce69ff2274dc798ef43
3  002be0ffdc997bd5c50703158b7c2491  034e861343a63ac3c18a9ceb1ce0ac69
4  002be0ffdc997bd5c50703158b7c2491  034f2e614a2f9fc7f1c2f77647d1b981
5  002be0ffdc997bd5c50703158b7c2491  03d6b9d80b8a96135cb9b25178e9e203
6  002be0ffdc997bd5c50703158b7c2491  04053c0ed21761e07f0b869cab5b7dd0
7  002be0ffdc997bd5c50703158b7c2491  0534d432e0186625f623aaee57af98be
8  002be0ffdc997bd5c50703158b7c2491  053a1621c0affcd2b9c517af5c2bc843
9  002be0ffdc997bd5c50703158b7c2491  066e92c52f59486de56cd7b8716a4ca6

   ride_distance  ride_duration  ride_prime_time
0           1811            327               50
1           3362            809                0
2           3282            572                0
3          65283           3338               25
4           4115            823              100
5           4832            917              100
6           1575            347                0
7           3056            687               25
8           3940           1143               75
9           3957            868               50
```

```
In [147]: rides_id.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 193502 entries, 0 to 193501
Data columns (total 5 columns):
driver_id          193502 non-null object
ride_id            193502 non-null object
ride_distance      193502 non-null int64
ride_duration      193502 non-null int64
ride_prime_time    193502 non-null int64
dtypes: int64(3), object(2)
memory usage: 7.4+ MB
```

2.8 ride_timestamps.csv has

- ride_id Unique identifier for a ride
- event event describes the type of event (see below)
- timestamp Time of event

Insights -

1. There are 970405 events in this dataset, each one has 5 event actions - requested at, accepted at, arrived at, picked up at, and dropped off at.

2. The ride time range varies between: '2016-03-28 05:48:18' and '2016-06-27 00:50:50'

3. There is one missing value at arrived_at for ride_id 72f0fa0bd86800e9da5c4dced32c8735. I will remove it.

```
In [148]: rides_times.head()
```

Out[148]:

	ride_id	event	timestamp
0	00003037a262d9ee40e61b5c0718f7f0	requested_at	2016-06-13 09:39:19
1	00003037a262d9ee40e61b5c0718f7f0	accepted_at	2016-06-13 09:39:51
2	00003037a262d9ee40e61b5c0718f7f0	arrived_at	2016-06-13 09:44:31
3	00003037a262d9ee40e61b5c0718f7f0	picked_up_at	2016-06-13 09:44:33
4	00003037a262d9ee40e61b5c0718f7f0	dropped_off_at	2016-06-13 10:03:05

```
In [149]: rides_times.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 970405 entries, 0 to 970404
Data columns (total 3 columns):
ride_id      970405 non-null object
event        970405 non-null object
timestamp    970404 non-null object
dtypes: object(3)
memory usage: 22.2+ MB
```

```
In [150]: #convert into datetime format
          #rides_times["timestamp"]=pd.to_datetime(rides_times["timestamp"])

In [151]: #reshape the dataframe, so it is easier to read
          #rides_times = rides_times.pivot(index='ride_id', columns='event', values='timestamp

In [152]: rides_times.head(2)

Out[152]:                              ride_id         event             timestamp
          0  00003037a262d9ee40e61b5c0718f7f0  requested_at  2016-06-13 09:39:19
          1  00003037a262d9ee40e61b5c0718f7f0   accepted_at  2016-06-13 09:39:51

In [153]: null_column=rides_times.columns[rides_times.isnull().any()]

In [154]: null_column

Out[154]: Index(['timestamp'], dtype='object')

In [155]: print(rides_times[rides_times["timestamp"].isnull()][null_column])

       timestamp
434222       NaN


In [156]: rides_times.drop(rides_times.index[[434220,434221,434222,434223,434224]],inplace=True
```

To avoid seasonality effect, we will consider 3 periods of 28 days

```
In [157]: rides_times['timestamp'].min()

Out[157]: '2016-03-28 05:48:18'

In [158]: rides_times['timestamp'].max()

Out[158]: '2016-06-27 00:50:50'
```

2.9 part 2. Join dataframes

Insights

1. 83 drivers joined LYFT (have an onboard date) but didn't complete any ride. Removed them because on board time are crucial for each driver, and removing 8% of the drivers should not impact the distribution of the number of rides per driver.

```
In [159]: #outer join
          combined_df_driver_id_ride_id = pd.merge(drivers,rides_id,how='outer',on='driver_id'

In [160]: combined_df_driver_id_ride_id.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 193585 entries, 0 to 193584
Data columns (total 6 columns):
driver_id              193585 non-null object
driver_onboard_date    185974 non-null datetime64[ns]
ride_id                193502 non-null object
ride_distance          193502 non-null float64
ride_duration          193502 non-null float64
ride_prime_time        193502 non-null float64
dtypes: datetime64[ns](1), float64(3), object(2)
memory usage: 10.3+ MB
```

```
In [161]: # When joining the driver_id to the ride_id table the number of unique drivers increa
          combined_df_driver_id_ride_id['driver_id'].nunique()

Out[161]: 1020

In [162]: combined_df_driver_id_ride_id[combined_df_driver_id_ride_id['driver_onboard_date'].is

Out[162]: 83

In [163]: combined_df_driver_id_ride_id[combined_df_driver_id_ride_id['ride_id'].isnull()]['dri

Out[163]: 83

In [164]: combined_df_driver_id_ride_id.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 193585 entries, 0 to 193584
Data columns (total 6 columns):
driver_id              193585 non-null object
driver_onboard_date    185974 non-null datetime64[ns]
ride_id                193502 non-null object
ride_distance          193502 non-null float64
ride_duration          193502 non-null float64
ride_prime_time        193502 non-null float64
dtypes: datetime64[ns](1), float64(3), object(2)
memory usage: 10.3+ MB

In [165]: combined_df_driver_id_ride_id.head()

Out[165]:                          driver_id driver_onboard_date  \
          0  002be0ffdc997bd5c50703158b7c2491          2016-03-29
          1  002be0ffdc997bd5c50703158b7c2491          2016-03-29
          2  002be0ffdc997bd5c50703158b7c2491          2016-03-29
          3  002be0ffdc997bd5c50703158b7c2491          2016-03-29
          4  002be0ffdc997bd5c50703158b7c2491          2016-03-29
```

```
                            ride_id  ride_distance  ride_duration  \
0  006d61cf7446e682f7bc50b0f8a5bea5         1811.0          327.0
1  01b522c5c3a756fbdb12e95e87507eda         3362.0          809.0
2  029227c4c2971ce69ff2274dc798ef43         3282.0          572.0
3  034e861343a63ac3c18a9ceb1ce0ac69        65283.0         3338.0
4  034f2e614a2f9fc7f1c2f77647d1b981         4115.0          823.0

   ride_prime_time
0             50.0
1              0.0
2              0.0
3             25.0
4            100.0
```

2.10 Join the Combined Driver_id/Ride_id table to the Ride_timestamp

Insights

1683 rides (0.8%) don't have an associated timestamp are being removed

```
In [166]: combined_all = pd.merge(combined_df_driver_id_ride_id,rides_times,how='outer',on='ri
```

```
In [167]: combined_all.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 979167 entries, 0 to 979166
Data columns (total 8 columns):
driver_id              932857 non-null object
driver_onboard_date    922806 non-null datetime64[ns]
ride_id                979084 non-null object
ride_distance          932774 non-null float64
ride_duration          932774 non-null float64
ride_prime_time        932774 non-null float64
event                  970400 non-null object
timestamp              970400 non-null object
dtypes: datetime64[ns](1), float64(3), object(4)
memory usage: 67.2+ MB
```

```
In [168]: #the number of unique rides is higher than the one obtained with the Ride_id table 1
          combined_all['ride_id'].nunique()
```

```
Out[168]: 202764
```

```
In [169]: #The existing rides with no associated timestamp (1683 represents 0.8%) have to be r
          # However,the drivers that didn't complete any rides are kept in the dataframe
          combined_all[combined_all['timestamp'].isnull() &
                                                  combined_all['ride_id'].notnull()]['ride
```

```
Out[169]: 8684

In [170]: combined_all= combined_all[((combined_all['ride_id'].isnull())
                          & (combined_all['timestamp'].isnull())) |
                          (( combined_all['ride_id'].notnull()) &
                          (combined_all['timestamp'].notnull()))]

In [171]: combined_all.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 970483 entries, 0 to 979166
Data columns (total 8 columns):
driver_id              924173 non-null object
driver_onboard_date    921123 non-null datetime64[ns]
ride_id                970400 non-null object
ride_distance          924090 non-null float64
ride_duration          924090 non-null float64
ride_prime_time        924090 non-null float64
event                  970400 non-null object
timestamp              970400 non-null object
dtypes: datetime64[ns](1), float64(3), object(4)
memory usage: 66.6+ MB
```

3 ----------

From the Ride rates

Using the following assumption on ride costs

Base Fare - $2.00 Cost per Mile - $1.15 Cost per Minute - $0.22 Service Fee - $1.75 Minimum Fare - $5.00 Maximum Fare - $400.00

Therefore, we need to do the unit conversions.

Conversion:

ride_distance ---> meter to mile

ride_duration ---> second to minute

```
In [172]: combined_all['ride_duration_minutes']=combined_all['ride_duration'].map(lambda x: int
          combined_all['ride_distance_miles']=combined_all['ride_distance']*0.000621371
          combined_all.head()

Out[172]:                          driver_id driver_onboard_date  \
          0  002be0ffdc997bd5c50703158b7c2491          2016-03-29
          1  002be0ffdc997bd5c50703158b7c2491          2016-03-29
          2  002be0ffdc997bd5c50703158b7c2491          2016-03-29
          3  002be0ffdc997bd5c50703158b7c2491          2016-03-29
          4  002be0ffdc997bd5c50703158b7c2491          2016-03-29
```

```
                              ride_id  ride_distance  ride_duration  \
0  006d61cf7446e682f7bc50b0f8a5bea5         1811.0          327.0
1  006d61cf7446e682f7bc50b0f8a5bea5         1811.0          327.0
2  006d61cf7446e682f7bc50b0f8a5bea5         1811.0          327.0
3  006d61cf7446e682f7bc50b0f8a5bea5         1811.0          327.0
4  006d61cf7446e682f7bc50b0f8a5bea5         1811.0          327.0

   ride_prime_time           event            timestamp  \
0             50.0    requested_at  2016-04-23 02:13:50
1             50.0     accepted_at  2016-04-23 02:14:15
2             50.0      arrived_at  2016-04-23 02:16:36
3             50.0    picked_up_at  2016-04-23 02:16:40
4             50.0  dropped_off_at  2016-04-23 02:22:07

   ride_duration_minutes  ride_distance_miles
0                    5.0             1.125303
1                    5.0             1.125303
2                    5.0             1.125303
3                    5.0             1.125303
4                    5.0             1.125303
```

In [173]:
```
# Function to compute the ride price

def compute_ride_price(distance_miles, duration_minute, prime):
    price = 2+1.75+distance_miles*1.15+duration_minute*0.22
    price = price *((100+prime)/100.0)
    price.loc[price>400]=400
    price.loc[price<5]=5
    return price
```

In [174]:
```
combined_all['total_revenue_$'] = compute_ride_price(
                                                    combined_all['ride_distance_miles
                                                    combined_all['ride_duration_minut
                                                    combined_all['ride_prime_time']

combined_all.head()
```

Out[174]:
```
                              driver_id driver_onboard_date  \
0  002be0ffdc997bd5c50703158b7c2491          2016-03-29
1  002be0ffdc997bd5c50703158b7c2491          2016-03-29
2  002be0ffdc997bd5c50703158b7c2491          2016-03-29
3  002be0ffdc997bd5c50703158b7c2491          2016-03-29
4  002be0ffdc997bd5c50703158b7c2491          2016-03-29

                              ride_id  ride_distance  ride_duration  \
0  006d61cf7446e682f7bc50b0f8a5bea5         1811.0          327.0
1  006d61cf7446e682f7bc50b0f8a5bea5         1811.0          327.0
```

```
2  006d61cf7446e682f7bc50b0f8a5bea5         1811.0          327.0
3  006d61cf7446e682f7bc50b0f8a5bea5         1811.0          327.0
4  006d61cf7446e682f7bc50b0f8a5bea5         1811.0          327.0

   ride_prime_time           event            timestamp  \
0             50.0    requested_at  2016-04-23 02:13:50
1             50.0     accepted_at  2016-04-23 02:14:15
2             50.0      arrived_at  2016-04-23 02:16:36
3             50.0    picked_up_at  2016-04-23 02:16:40
4             50.0  dropped_off_at  2016-04-23 02:22:07

   ride_duration_minutes  ride_distance_miles  total_revenue_$
0                    5.0             1.125303         9.216147
1                    5.0             1.125303         9.216147
2                    5.0             1.125303         9.216147
3                    5.0             1.125303         9.216147
4                    5.0             1.125303         9.216147
```

In [175]:
```
# Rides only contain information per ride_id
#It contains 83 drivers that didn't complete any rides
rides = combined_all.groupby('ride_id', as_index=False).agg({
                                                              'driver_id': laml
                                                              'driver_onboard_
                                                              'ride_distance':
                                                              'ride_duration'
                                                              'ride_prime_time
                                                              'timestamp': laml
                                                              'ride_duration_mi
                                                              'ride_distance_mi
                                                              'total_revenue_$
                                                              })
```

In [176]:
```
rides.head()
```

Out[176]:
```
                            ride_id                         driver_id  \
0  00003037a262d9ee40e61b5c0718f7f0  d967f5296732fa55266b5f1314e7447b
1  00005eae40882760d675da5effb89ae3  0656192a402808805282e60761bda088
2  000061d42cf29f73b591041d9a1b2973  c468a648519cd42da75e6aa9dadf733e
3  00006efeb0d5e3ccad7d921ddeee9900  689bdf87fb2de49f98bf4946cfaa5068
4  0000d9b24d8ccdd991b76258e616fa01                               NaN

  driver_onboard_date  ride_distance  ride_duration  ride_prime_time  \
0          2016-04-09         3698.0         1112.0              0.0
1          2016-04-30         3016.0          479.0             25.0
2          2016-04-01         4084.0          406.0             75.0
3          2016-04-04         1646.0          332.0             75.0
4                 NaT            NaN            NaN              NaN
```

```
             timestamp  ride_duration_minutes  ride_distance_miles  \
0  2016-06-13 09:39:19                   18.0             2.297830
1  2016-05-14 05:23:21                    7.0             1.874055
2  2016-05-16 15:43:09                    6.0             2.537679
3  2016-05-11 19:29:36                    5.0             1.022777
4  2016-04-26 18:11:38                    NaN                  NaN

   total_revenue_$
0        10.352504
1         9.306454
2        13.979579
3        10.545838
4              NaN
```

In [177]:
```
## Drivers contain information per driver
# drivers doesn t contain the 83 drivers because it
drivers = rides.groupby('driver_id', as_index=False).agg({
                                                'ride_id': lambda x: x.count(),
                                                'driver_onboard_date': lambda x: pd.to_da
                                                'timestamp': lambda x: pd.to_datetime(x.
                                                'total_revenue_$': lambda x: x.sum(),
                                                'ride_distance_miles': lambda x: x.sum()
                                                'ride_duration': lambda x: x.sum(),
                                                }).rename(columns={
                                                                    'ride_id': '#_rides'
                                                                    'timestamp': 'last_da
                                                                    'ride_distance_miles
                                                                    'ride_duration':'tota
                                                    })
drivers.head()
```

Out[177]:
```
                          driver_id  #_rides driver_onboard_date  \
0  002be0ffdc997bd5c50703158b7c2491       277          2016-03-29
1  007f0389f9c7b03ef97098422f902e62        31          2016-03-29
2  011e5c5dfc5c2c92501b8b24d47509bc        34          2016-04-05
3  0152a2f305e71d26cc964f8d4411add9       191          2016-04-23
4  01674381af7edd264113d4e6ed55ecda       375          2016-04-29

            last_date  total_revenue_$  total_distance_miles  \
0 2016-06-23 10:06:26      3618.368651           1081.363873
1 2016-06-22 13:17:40       328.347501             73.030355
2 2016-06-12 20:22:22       489.169288            167.554554
3 2016-06-26 10:16:37      2622.200548            914.185249
4 2016-06-24 13:03:37      5418.915437           1940.941796

   total_duration_seconds
0                221238.0
1                 20497.0
```

```
2                 29205.0
3                174521.0
4                357443.0
```

```
In [178]: drivers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 844 entries, 0 to 843
Data columns (total 7 columns):
driver_id                 844 non-null object
#_rides                   844 non-null int64
driver_onboard_date       837 non-null datetime64[ns]
last_date                 844 non-null datetime64[ns]
total_revenue_$           844 non-null float64
total_distance_miles      844 non-null float64
total_duration_seconds    844 non-null float64
dtypes: datetime64[ns](2), float64(3), int64(1), object(1)
memory usage: 52.8+ KB
```

```
In [179]: # Those are the drivers who joined Lyft but never completed a ride
          inexistant_rides = combined_all.loc[combined_all['ride_id'].isnull()]
          never_activated_drivers = inexistant_rides.groupby('driver_id', as_index=False).agg(
                                                                  'ride_id': lambda x: x.co
                                                                  'driver_onboard_date': la
                                                                  'total_revenue_$': lambda
                                                                  'ride_distance_miles': la
                                                                  'ride_duration': lambda x
                                                          }).rename(columns={
                                                                      'ride_id': '#_rides'
                                                                      'ride_distance_miles
                                                                      'ride_duration':'tota
                                                              })

          never_activated_drivers['last_date']=never_activated_drivers['driver_onboard_date']
```

```
In [180]: #add those never activated drivers to drivers dataset
          drivers = pd.concat([drivers, never_activated_drivers])
```

```
In [181]: drivers['total_#_days']=(drivers['last_date']-drivers['driver_onboard_date']+timedelt
          drivers['rides_per_day'] =drivers['#_rides']/drivers['total_#_days']
          drivers['rides_per_day'].loc[drivers['rides_per_day'].isnull()]=0
```

```
In [182]: drivers.head()
```

```
Out[182]:    #_rides                        driver_id driver_onboard_date  \
          0      277  002be0ffdc997bd5c50703158b7c2491          2016-03-29
          1       31  007f0389f9c7b03ef97098422f902e62          2016-03-29
          2       34  011e5c5dfc5c2c92501b8b24d47509bc          2016-04-05
```

```
3      191  0152a2f305e71d26cc964f8d4411add9          2016-04-23
4      375  01674381af7edd264113d4e6ed55ecda          2016-04-29

            last_date  total_distance_miles  total_duration_seconds  \
0 2016-06-23 10:06:26           1081.363873                221238.0
1 2016-06-22 13:17:40             73.030355                 20497.0
2 2016-06-12 20:22:22            167.554554                 29205.0
3 2016-06-26 10:16:37            914.185249                174521.0
4 2016-06-24 13:03:37           1940.941796                357443.0

   total_revenue_$  total_#_days  rides_per_day
0      3618.368651          87.0       3.183908
1       328.347501          86.0       0.360465
2       489.169288          69.0       0.492754
3      2622.200548          65.0       2.938462
4      5418.915437          57.0       6.578947
```

```
In [183]: drivers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 927 entries, 0 to 82
Data columns (total 9 columns):
#_rides                   927 non-null int64
driver_id                 927 non-null object
driver_onboard_date       920 non-null datetime64[ns]
last_date                 927 non-null datetime64[ns]
total_distance_miles      927 non-null float64
total_duration_seconds    927 non-null float64
total_revenue_$           927 non-null float64
total_#_days              920 non-null float64
rides_per_day             927 non-null float64
dtypes: datetime64[ns](2), float64(5), int64(1), object(1)
memory usage: 72.4+ KB
```

4 Part 3 - EDA!

Insights

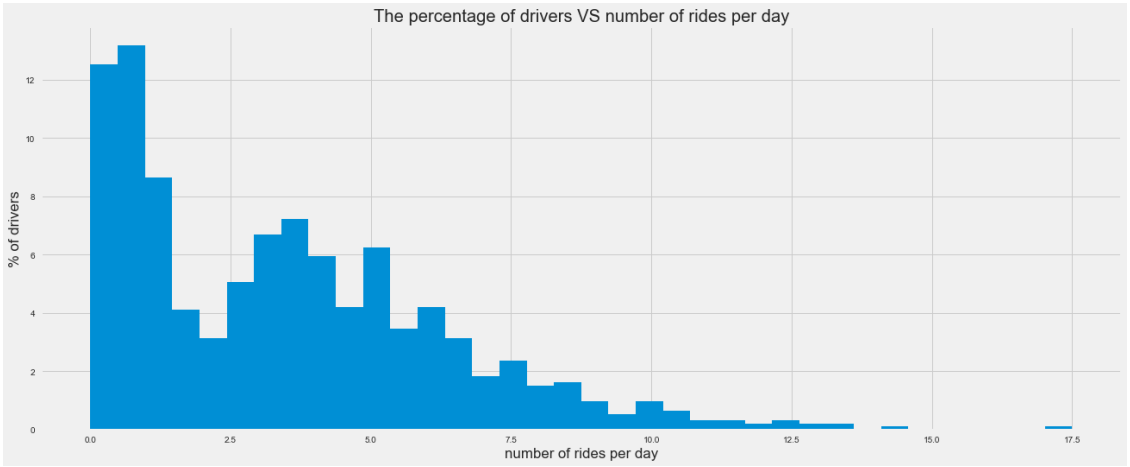
1.The driver behavior varies SIGNIFICANTLY across drivers, they don't act alike

```
In [184]: drivers.describe()
```

```
Out[184]:           #_rides  total_distance_miles  total_duration_seconds  \
          count  927.000000            927.000000              927.000000
          mean   199.372168            859.940847           171399.894283
          std    180.870521            793.237956           156255.371603
          min      0.000000              0.000000                0.000000
```

```
25%      39.000000            162.546304          32730.500000
50%     203.000000            766.887389         169015.000000
75%     317.000000           1358.658760         270937.000000
max     919.000000           4118.571884         779797.000000

       total_revenue_$  total_#_days  rides_per_day
count       927.000000    920.000000     927.000000
mean       2734.397058     51.106522       3.474992
std        2488.067272     26.047365       2.866441
min           0.000000      1.000000       0.000000
25%         519.490216     34.000000       0.954451
50%        2647.977744     56.000000       3.148148
75%        4332.596786     72.000000       5.264481
max       12525.436478     91.000000      17.500000
```

In [185]:
```
ndrivers = len(drivers['driver_id'])
drivers.hist(column='rides_per_day', bins=36, figsize=(20,8), weights=np.ones(ndrive
plt.xlabel("number of rides per day")
plt.ylabel("% of drivers")
plt.title("The percentage of drivers VS number of rides per day")
plt.savefig('#_rides_day.png')
plt.show()
```



In [186]:
```
# Here, we need at least 2 periods of 28 days to correlate churn with revenue in the
# Churners are the drivers who didn t complete any ride during 28 days

rides_at_least_56_days_data = rides[pd.to_datetime(rides['driver_onboard_date']) <=
                                    pd.to_datetime('2016-05-02 00:00:00') ]
```

In [187]:
```
# first 28 periods rides data

rides_first_28_period = rides_at_least_56_days_data[(pd.to_datetime(rides_at_least_5
```

```
                                                          pd.to_datetime(rides_at_least_56
rides_first_28_period.head()
```

Out[187]:

	ride_id	driver_id	driver_onboard_date	ride_distance	ride_duration	ride_prime_time	timestamp	ride_duration_minutes	ride_distance_miles	total_revenue_$
1	00005eae40882760d675da5effb89ae3	0656192a402808805282e60761bda088	2016-04-30	3016.0	479.0	25.0	2016-05-14 05:23:21	7.0	1.874055	9.306454
10	000290b418595bfe228d2bf4d4e331df	4aa585f63fbe04dcded4019662cc47f8	2016-04-17	3019.0	852.0	0.0	2016-04-30 23:30:19	14.0	1.875919	8.987307
16	0003bd8128cced32fcb88aa3805b0a72	8dc9d28aa6ab0af5dcc98420a9bd65e0	2016-04-23	6970.0	1291.0	25.0	2016-04-26 21:21:21	21.0	4.330956	16.688249
18	00046fb60349e54c9845d667d7c897fd	39a7bc235caf53556b15dd28ab5a7157	2016-05-02	2314.0	490.0	100.0	2016-05-08 01:59:20	8.0	1.437852	14.327061
22	000527ec3aa73cc922d8a4858edf9c06	a6fe0a06612bc9d9fd3f7dc1fd9615f0	2016-04-11	5796.0	527.0	0.0	2016-04-13 22:55:05	8.0	3.601466	9.651686

```
In [188]: #rides second period
rides_second_28_period = rides_at_least_56_days_data[(pd.to_datetime(rides_at_least_5
                                                       pd.to_datetime(rides_at_least_5
                                                      (pd.to_datetime(rides_at_least_5
                                                       pd.to_datetime(rides_at_least_5


rides_first_28_period.head()
```

Out[188]:

	ride_id	driver_id	driver_onboard_date	ride_distance	ride_duration	ride_prime_time
1	00005eae40882760d675da5effb89ae3	0656192a402808805282e60761bda088				
10	000290b418595bfe228d2bf4d4e331df	4aa585f63fbe04dcded4019662cc47f8				
16	0003bd8128cced32fcb88aa3805b0a72	8dc9d28aa6ab0af5dcc98420a9bd65e0				
18	00046fb60349e54c9845d667d7c897fd	39a7bc235caf53556b15dd28ab5a7157				
22	000527ec3aa73cc922d8a4858edf9c06	a6fe0a06612bc9d9fd3f7dc1fd9615f0				

```
1           2016-04-30         3016.0          479.0             25.0
10          2016-04-17         3019.0          852.0              0.0
16          2016-04-23         6970.0         1291.0             25.0
18          2016-05-02         2314.0          490.0            100.0
22          2016-04-11         5796.0          527.0              0.0

                timestamp  ride_duration_minutes  ride_distance_miles  \
1   2016-05-14 05:23:21                    7.0             1.874055
10  2016-04-30 23:30:19                   14.0             1.875919
16  2016-04-26 21:21:21                   21.0             4.330956
18  2016-05-08 01:59:20                    8.0             1.437852
22  2016-04-13 22:55:05                    8.0             3.601466

    total_revenue_$
1          9.306454
10         8.987307
16        16.688249
18        14.327061
22         9.651686
```

In [189]:
```
# drivers first 28 period
drivers_first_28_period = rides_first_28_period.groupby('driver_id', as_index=False)
                                                'ride_id': lambda x: x.count(),
                                                'driver_onboard_date': lambda x: pd.to_da
                                                'timestamp': lambda x: pd.to_datetime(x.
                                                'total_revenue_$': lambda x: x.sum(),
                                                'ride_distance_miles': lambda x: x.sum()
                                                'ride_duration': lambda x: x.sum(),
                                                'ride_prime_time': lambda x: x[x>0].coun
                                                'timestamp': lambda x: x.str.slice(start
                                                #'total_revenue_$': lambda x: x[rides_fi
                                                }).rename(columns={
                                                                    'ride_id': '#_rides'
                                                                    'timestamp': 'last_da
                                                                    'ride_distance_miles
                                                                    'ride_duration':'tota
                                                                    'total_revenue_$':'fi
                                                                    'timestamp': '#_activ
                                                                    'ride_prime_time': '
                                                                    #'total_revenue_$':
                                                    })
drivers_first_28_period.head()
```

Out[189]:
```
                          driver_id  #_rides driver_onboard_date  \
0  002be0ffdc997bd5c50703158b7c2491       109          2016-03-29
1  007f0389f9c7b03ef97098422f902e62         7          2016-03-29
2  011e5c5dfc5c2c92501b8b24d47509bc        12          2016-04-05
3  0152a2f305e71d26cc964f8d4411add9        56          2016-04-23
```

```
4  01674381af7edd264113d4e6ed55ecda      176          2016-04-29

   #_active_days  first_28_days_revenue_$  total_distance_miles  \
0             21              1312.907194            386.305108
1              3                87.102761             18.624974
2              5               170.542045             58.779211
3             15               743.278028            264.131763
4             20              2494.205956            878.087322

   total_duration_seconds  %_prime_time
0                 78271.0      0.376147
1                  5471.0      0.428571
2                  9929.0      0.416667
3                 46599.0      0.214286
4                165773.0      0.261364
```

In [190]:
```
#Drivers second periof of 28 days

drivers_second_28_period = rides_second_28_period.groupby('driver_id', as_index=Fals
                                                  'ride_id': lambda x: x.count(),
                                                  'driver_onboard_date': lambda x: pd.to_da
                                                  'timestamp': lambda x: pd.to_datetime(x.
                                                  'total_revenue_$': lambda x: x.sum(),
                                                  'ride_distance_miles': lambda x: x.sum()
                                                  'ride_duration': lambda x: x.sum(),
                                                  }).rename(columns={
                                                                     'ride_id': '#_rides'
                                                                     'timestamp': 'last_da
                                                                     'ride_distance_miles
                                                                     'ride_duration':'tota
                                                                     'total_revenue_$':'s
                                                      })
drivers_second_28_period.head()
```

Out[190]:
```
                          driver_id  #_rides driver_onboard_date  \
0  002be0ffdc997bd5c50703158b7c2491       42          2016-03-29
1  007f0389f9c7b03ef97098422f902e62       15          2016-03-29
2  011e5c5dfc5c2c92501b8b24d47509bc       17          2016-04-05
3  0152a2f305e71d26cc964f8d4411add9       93          2016-04-23
4  01674381af7edd264113d4e6ed55ecda      190          2016-04-29

            last_date  second_28_days_revenue_$  total_distance_miles  \
0 2016-05-22 14:57:33                613.963974            166.278880
1 2016-05-19 15:56:56                152.819293             39.226530
2 2016-05-26 07:22:38                268.260552             94.615541
3 2016-06-17 08:21:46               1331.836383            465.414957
4 2016-06-23 14:36:52               2784.087152           1015.189105
```

```
   total_duration_seconds
0                 34449.0
1                 10049.0
2                 16457.0
3                 90484.0
4                182815.0
```

In [191]:
```
#Adding an column "ride_per_day" to drivers 1st period

drivers_first_28_period['rides_per_day'] =drivers_first_28_period['#_rides']/28
drivers_first_28_period.head()
```

Out[191]:
```
                          driver_id  #_rides driver_onboard_date  \
0  002be0ffdc997bd5c50703158b7c2491       109          2016-03-29
1  007f0389f9c7b03ef97098422f902e62         7          2016-03-29
2  011e5c5dfc5c2c92501b8b24d47509bc        12          2016-04-05
3  0152a2f305e71d26cc964f8d4411add9        56          2016-04-23
4  01674381af7edd264113d4e6ed55ecda       176          2016-04-29

   #_active_days  first_28_days_revenue_$  total_distance_miles  \
0             21              1312.907194            386.305108
1              3                87.102761             18.624974
2              5               170.542045             58.779211
3             15               743.278028            264.131763
4             20              2494.205956            878.087322

   total_duration_seconds  %_prime_time  rides_per_day
0                 78271.0      0.376147       3.892857
1                  5471.0      0.428571       0.250000
2                  9929.0      0.416667       0.428571
3                 46599.0      0.214286       2.000000
4                165773.0      0.261364       6.285714
```

In [192]:
```
drivers_revenue_per_period = pd.merge(drivers_first_28_period.iloc[:,[0,1,3,4,7]],dr
```

In [193]:
```
drivers_revenue_per_period.head()
```

Out[193]:
```
                          driver_id  #_rides  #_active_days  \
0  002be0ffdc997bd5c50703158b7c2491       109             21
1  007f0389f9c7b03ef97098422f902e62         7              3
2  011e5c5dfc5c2c92501b8b24d47509bc        12              5
3  0152a2f305e71d26cc964f8d4411add9        56             15
4  01674381af7edd264113d4e6ed55ecda       176             20

   first_28_days_revenue_$  %_prime_time  second_28_days_revenue_$
0              1312.907194      0.376147                613.963974
1                87.102761      0.428571                152.819293
2               170.542045      0.416667                268.260552
3               743.278028      0.214286               1331.836383
4              2494.205956      0.261364               2784.087152
```

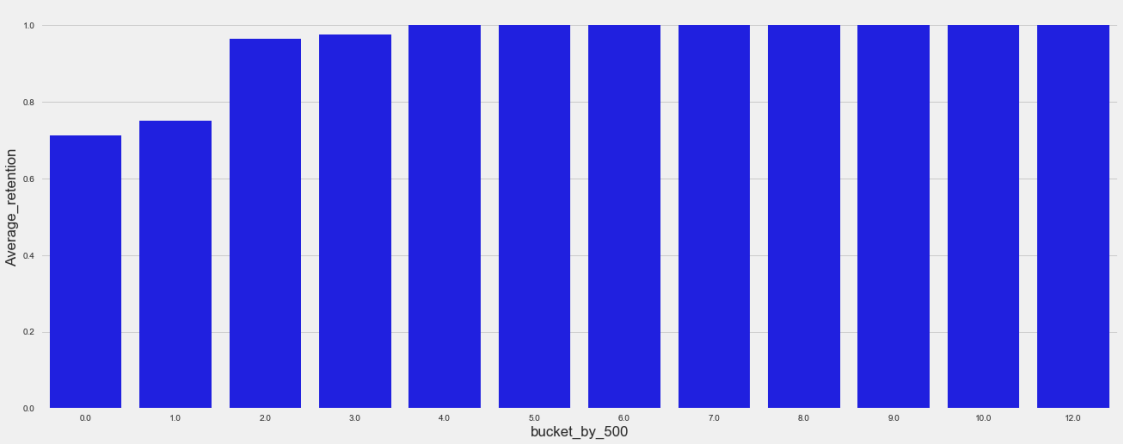
```
In [194]: drivers_revenue_per_period['Churn_2nd_period'] = drivers_revenue_per_period['second_

In [195]: drivers_revenue_per_period['retention'] = 1- drivers_revenue_per_period['Churn_2nd_pe

In [196]: drivers_revenue_per_period['bucket_by_500'] = drivers_revenue_per_period['first_28_da

In [197]: churn_per_revenue_bucket = drivers_revenue_per_period.groupby('bucket_by_500', as_ind
                                                    'driver_id': lambda x: x.count(),
                                                    'Churn_2nd_period': np.mean,
                                                    'retention': np.mean
                                                    }).rename(columns={
                                                                       'driver_id': '#_drive
                                                                       'Churn_2nd_period':
                                                                       'retention': 'Average
                                                        })
          churn_per_revenue_bucket
```

Out[197]:

	bucket_by_500	#_drivers	Average_churn	Average_retention
0	0.0	174	0.287356	0.712644
1	1.0	152	0.250000	0.750000
2	2.0	109	0.036697	0.963303
3	3.0	84	0.023810	0.976190
4	4.0	49	0.000000	1.000000
5	5.0	40	0.000000	1.000000
6	6.0	17	0.000000	1.000000
7	7.0	12	0.000000	1.000000
8	8.0	3	0.000000	1.000000
9	9.0	2	0.000000	1.000000
10	10.0	3	0.000000	1.000000
11	12.0	1	0.000000	1.000000

```
In [198]: plt.figure(figsize=(20,8))
          sns.barplot(x='bucket_by_500',y='Average_retention',data=churn_per_revenue_bucket,col
          #plt.savefig('retention.png')
```

Out[198]: <matplotlib.axes._subplots.AxesSubplot at 0x1a28bdbdd8>



```
In [199]: drivers_revenue_per_period.to_pickle('/Users/zoezhou/Documents/churn_df.pickle')

In [200]: drivers_revenue_per_period.head(30)
```

Out[200]:

	driver_id	#_rides	#_active_days
0	002be0ffdc997bd5c50703158b7c2491	109	21
1	007f0389f9c7b03ef97098422f902e62	7	3
2	011e5c5dfc5c2c92501b8b24d47509bc	12	5
3	0152a2f305e71d26cc964f8d4411add9	56	15
4	01674381af7edd264113d4e6ed55ecda	176	20
5	0213f8b59219e32142711992ca4ec01f	203	22
6	0258e250ca195cc6258cbdc75aecd853	110	18
7	02e440f6c209206375833cef02e0cbae	47	11
8	036f3d94e7c65e4e3574822d31c72656	53	21
9	039c5afbca8e03e4c18d9c8ea94140ac	29	17
10	03f2b5c74cb89f39e58711699e76bf39	82	14
11	03f5278eb43475aa6790f5be32463755	98	17
12	0430df9a3eb327122c57ee3a64765000	43	8
13	04c4ffa5a385eab86fa7e422263d2999	46	11
14	052bba06c5fc0bdea4bc2f9cb92b37c7	51	10
15	05addf442c147875efa5cf53453ad47b	165	20
16	0656192a402808805282e60761bda088	142	18
17	06c848ab3a7fc5421e82e98850a81710	173	26
18	07dd442e3e0b9f0f9b0d69c7b47cbb06	114	18
19	081d8ba3bc9a00a481df02bd9d0a4c53	33	6
20	08a1491d6a804e0af969f08252ddbbd8	190	26
21	08b2b063cce8d02495c4b880293f153c	200	22
22	0938ed763cb3129ae63607aaf69daff5	22	6
23	0afc0241296972b583debd7c5f5c707c	84	7
24	0b631e16fa61f7321da18cf35a076d5f	75	13
25	0c02bd2b09f7193103279ab9b760b777	70	12
26	0ca501b2a1d72e80e0d0cd2c25bdd124	77	13
27	0e7f0f05c7e193b1774c2e5713741cd4	68	14
28	0eff1404b137a5562642f0f706e59f25	45	11
29	0f057c0c73054f569a59a0880b91cbb0	7	2

	first_28_days_revenue_$	%_prime_time	second_28_days_revenue_$
0	1312.907194	0.376147	613.963974
1	87.102761	0.428571	152.819293
2	170.542045	0.416667	268.260552
3	743.278028	0.214286	1331.836383
4	2494.205956	0.261364	2784.087152
5	2516.720877	0.241379	441.628027
6	1527.668793	0.300000	2257.075312
7	641.259187	0.212766	149.797530

8	745.005119	0.113208	1117.309922
9	318.130982	0.275862	138.357121
10	1018.583527	0.256098	695.972874
11	1345.648190	0.357143	1140.965739
12	504.332784	0.186047	NaN
13	789.657192	0.434783	114.057993
14	680.510118	0.274510	35.450385
15	1975.525445	0.345455	2015.722527
16	2109.353535	0.274648	2198.201654
17	1994.460700	0.375723	2024.870447
18	1359.521015	0.131579	849.981604
19	363.042230	0.303030	232.012095
20	2752.667626	0.373684	3604.204887
21	3033.671690	0.325000	3343.870329
22	212.150908	0.136364	144.679167
23	1211.522124	0.345238	2197.591023
24	834.257221	0.506667	2681.318297
25	742.716762	0.071429	1607.393476
26	1005.852858	0.350649	1796.687403
27	1019.717570	0.176471	1544.404502
28	462.291178	0.155556	NaN
29	122.973636	0.000000	43.873913

	Churn_2nd_period	retention	bucket_by_500
0	0	1	2.0
1	0	1	0.0
2	0	1	0.0
3	0	1	1.0
4	0	1	4.0
5	0	1	5.0
6	0	1	3.0
7	0	1	1.0
8	0	1	1.0
9	0	1	0.0
10	0	1	2.0
11	0	1	2.0
12	1	0	1.0
13	0	1	1.0
14	0	1	1.0
15	0	1	3.0
16	0	1	4.0
17	0	1	3.0
18	0	1	2.0
19	0	1	0.0
20	0	1	5.0
21	0	1	6.0
22	0	1	0.0
23	0	1	2.0

24	0	1	1.0
25	0	1	1.0
26	0	1	2.0
27	0	1	2.0
28	1	0	0.0
29	0	1	0.0

5 Drivers Value

5.1 first 28 days

```
In [201]: drivers_value_first_28_period = rides_first_28_period.groupby('driver_id', as_index=
                                                    'ride_id': lambda x: x.count(),
                                                    'driver_onboard_date': lambda x: pd.to_da
                                                    'timestamp': lambda x: pd.to_datetime(x.
                                                    'total_revenue_$': {'first_28_days_revenu
                                                    'ride_distance_miles': lambda x: x.sum()
                                                    'ride_duration': lambda x: x.sum(),
                                                    'ride_prime_time': lambda x: x[x>0].coun
                                                    'timestamp': lambda x: x.str.slice(start=
                                                    #'total_revenue_$': lambda x: x[rides_fi
                                                    }).rename(columns={
                                                                        'ride_id': '#_rides'
                                                                        'timestamp': 'last_da
                                                                        'ride_distance_miles
                                                                        'ride_duration':'tota
                                                                        #'total_revenue_$':'
                                                                        'timestamp': '#_activ
                                                                        'ride_prime_time': '%
                                                                        #'total_revenue_$':
                                                        })
          drivers_value_first_28_period.head()
```

Out[201]:

	driver_id	#_rides	driver_onboard_date	#_active_days	total_revenue_$	
		<lambda>	<lambda>	<lambda>	first_28_days_revenue_$	revenue_prime_time
0	002be0ffdc997bd5c50703158b7c2491	109	2016-03-29	21	1312.907194	582.286988
1	007f0389f9c7b03ef97098422f902e62	7	2016-03-29	3	87.102761	50.744614
2	011e5c5dfc5c2c92501b8b24d47509bc	12	2016-04-05	5	170.542045	80.358454
3	0152a2f305e71d26cc964f8d4411add9	56	2016-04-23	15	743.278028	203.510823
4	01674381af7edd264113d4e6ed55ecda	176	2016-04-29	20	2494.205956	709.985233

```
                         total_distance_miles total_duration_seconds  \
  revenue_no_prime_time              <lambda>                <lambda>
0             730.620206           386.305108                78271.0
1              36.358147            18.624974                 5471.0
2              90.183591            58.779211                 9929.0
3             539.767206           264.131763                46599.0
4            1784.220724           878.087322               165773.0

  %_prime_time
      <lambda>
0     0.376147
1     0.428571
2     0.416667
3     0.214286
4     0.261364
```

In [202]:
```
first_period_drivers_value = drivers_value_first_28_period['total_revenue_$']
first_period_drivers_value['driver_id'] = drivers_value_first_28_period['driver_id']
```

In [203]:
```
# Lyft commision on each ride is 25%
# If prime time, the value of a driver should increase -> 70% of 25%
# If no prime time, the value of a driver should increase -> 50% of 25%

first_period_drivers_value['Driver_value_if_prime_time_first_period'] =  0.7 * (first
first_period_drivers_value['Driver_value_if_no_prime_time_first_period'] =  0.5 * (fi
first_period_drivers_value['Total_Driver_value_first_period'] =  first_period_drivers
first_period_drivers_value.head()
```

Out[203]:
```
   first_28_days_revenue_$  revenue_prime_time  revenue_no_prime_time  \
0              1312.907194          582.286988             730.620206
1                87.102761           50.744614              36.358147
2               170.542045           80.358454              90.183591
3               743.278028          203.510823             539.767206
4              2494.205956          709.985233            1784.220724

                          driver_id  Driver_value_if_prime_time_first_period  \
0  002be0ffdc997bd5c50703158b7c2491                                101.900223
1  007f0389f9c7b03ef97098422f902e62                                  8.880307
2  011e5c5dfc5c2c92501b8b24d47509bc                                 14.062729
3  0152a2f305e71d26cc964f8d4411add9                                 35.614394
4  01674381af7edd264113d4e6ed55ecda                                124.247416

   Driver_value_if_no_prime_time_first_period  Total_Driver_value_first_period
0                                   91.327526                       193.227749
1                                    4.544768                        13.425076
2                                   11.272949                        25.335678
3                                   67.470901                       103.085295
4                                  223.027590                       347.275006
```

5.2 Second 28 days

In [204]:
```
#second 28 day period

drivers_value_second_28_period = rides_second_28_period.groupby('driver_id', as_index
                                                    'ride_id': lambda x: x.count(),
                                                    'driver_onboard_date': lambda x: pd.to_da
                                                    'timestamp': lambda x: pd.to_datetime(x.
                                                    'total_revenue_$': {'second_28_days_rever
                                                    'ride_distance_miles': lambda x: x.sum()
                                                    'ride_duration': lambda x: x.sum(),
                                                    'ride_prime_time': lambda x: x[x>0].count
                                                    'timestamp': lambda x: x.str.slice(start=
                                                    #'total_revenue_$': lambda x: x[rides_fi
                                                    }).rename(columns={
                                                                        'ride_id': '#_rides'
                                                                        'timestamp': 'last_da
                                                                        'ride_distance_miles
                                                                        'ride_duration':'tota
                                                                        #'total_revenue_$':'
                                                                        'timestamp': '#_activ
                                                                        'ride_prime_time': '%
                                                                        #'total_revenue_$':
                                                        })
drivers_value_second_28_period.head()
```

Out[204]:

	driver_id	#_rides	driver_onboard_date	#_active_days	total_revenue_$			total_distance_miles	total_duration_seconds
		<lambda>	<lambda>	<lambda>	second_28_days_revenue_$	revenue_prime_time	revenue_no_prime_time	<lambda>	<lambda>
0	002be0ffdc997bd5c50703158b7c2491	42	2016-03-29	11	613.963974	428.923974	185.040000	166.278880	34449.0
1	007f0389f9c7b03ef97098422f902e62	15	2016-03-29	5	152.819293	57.499176	95.320117	39.226530	10049.0
2	011e5c5dfc5c2c92501b8b24d47509bc	17	2016-04-05	6	268.260552	123.910647	144.349906	94.615541	16457.0
3	0152a2f305e71d26cc964f8d4411add9	93	2016-04-23	19	1331.836383	430.404322	901.432061	465.414957	90484.0
4	01674381af7edd264113d4e6ed55ecda	190	2016-04-29	19	2784.087152	796.331660	1987.755492	1015.189105	182815.0

```
  %_prime_time
      <lambda>
0     0.523810
1     0.333333
2     0.529412
3     0.279570
4     0.247368
```

In [205]:
```
second_period_drivers_value = drivers_value_second_28_period['total_revenue_$']
second_period_drivers_value['driver_id'] = drivers_value_second_28_period['driver_id
```

In [206]:
```
# Lyft commision on each ride is 25%
# If prime time, the value of a driver should increase -> 70% of 25%
# If no prime time, the value of a driver should increase -> 50% of 25%

second_period_drivers_value['Driver_value_if_prime_time_second_period'] =  0.7 * (se
second_period_drivers_value['Driver_value_if_no_prime_time_second_period'] =  0.5 *
second_period_drivers_value['Total_Driver_value_second_period'] =  second_period_dri
second_period_drivers_value.head()
```

Out[206]:

	second_28_days_revenue_$	revenue_prime_time	revenue_no_prime_time	\
0	613.963974	428.923974	185.040000	
1	152.819293	57.499176	95.320117	
2	268.260552	123.910647	144.349906	
3	1331.836383	430.404322	901.432061	
4	2784.087152	796.331660	1987.755492	

	driver_id	Driver_value_if_prime_time_second_period	\
0	002be0ffdc997bd5c50703158b7c2491	75.061695	
1	007f0389f9c7b03ef97098422f902e62	10.062356	
2	011e5c5dfc5c2c92501b8b24d47509bc	21.684363	
3	0152a2f305e71d26cc964f8d4411add9	75.320756	
4	01674381af7edd264113d4e6ed55ecda	139.358040	

	Driver_value_if_no_prime_time_second_period	\
0	23.130000	
1	11.915015	
2	18.043738	
3	112.679008	
4	248.469437	

	Total_Driver_value_second_period
0	98.191695
1	21.977370
2	39.728101
3	187.999764
4	387.827477

In [207]:
```
# There are drivers who didn't complete any rides during the second 28 days period

total_value_per_driver= pd.merge(first_period_drivers_value.iloc[:,[3,6]],second_peri
total_value_per_driver['Total_Driver_value_second_period'].fillna(0,inplace=True)
total_value_per_driver.head(10)
```

Out[207]:

	driver_id	Total_Driver_value_first_period	Total_Driver_value_second_period
0	002be0ffdc997bd5c50703158b7c2491	193.227749	98.191695
1	007f0389f9c7b03ef97098422f902e62	13.425076	21.977370
2	011e5c5dfc5c2c92501b8b24d47509bc	25.335678	39.728101
3	0152a2f305e71d26cc964f8d4411add9	103.085295	187.999764
4	01674381af7edd264113d4e6ed55ecda	347.275006	387.827477
5	0213f8b59219e32142711992ca4ec01f	354.747175	63.324945
6	0258e250ca195cc6258cbdc75aecd853	217.274932	321.647758
7	02e440f6c209206375833cef02e0cbae	89.643502	20.736770
8	036f3d94e7c65e4e3574822d31c72656	98.207703	154.941765
9	039c5afbca8e03e4c18d9c8ea94140ac	45.901925	17.294640

In [208]:
```
total_value_per_driver['Average_value']=(total_value_per_driver['Total_Driver_value_
```

In [209]:
```
total_value_per_driver.head()
```

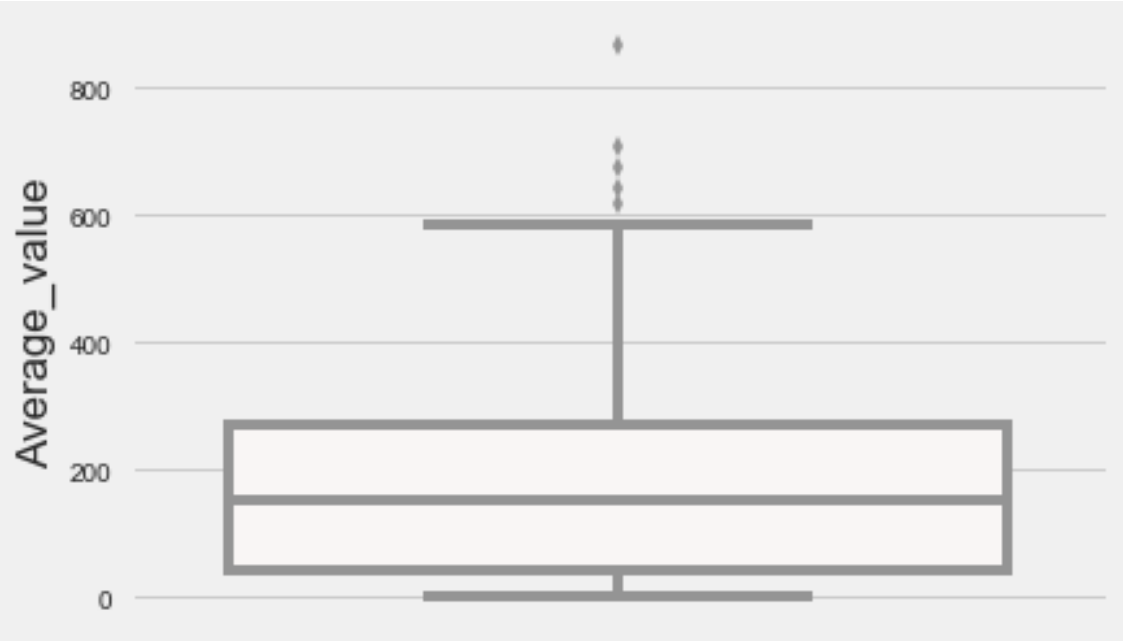
Out[209]:

	driver_id	Total_Driver_value_first_period	Total_Driver_value_second_period	Average_value
0	002be0ffdc997bd5c50703158b7c2491	193.227749	98.191695	145.709722
1	007f0389f9c7b03ef97098422f902e62	13.425076	21.977370	17.701223
2	011e5c5dfc5c2c92501b8b24d47509bc	25.335678	39.728101	32.531890
3	0152a2f305e71d26cc964f8d4411add9	103.085295	187.999764	145.542529
4	01674381af7edd264113d4e6ed55ecda	347.275006	387.827477	367.551242

In [210]:
```
total_value_per_driver['Average_value'].describe()
```

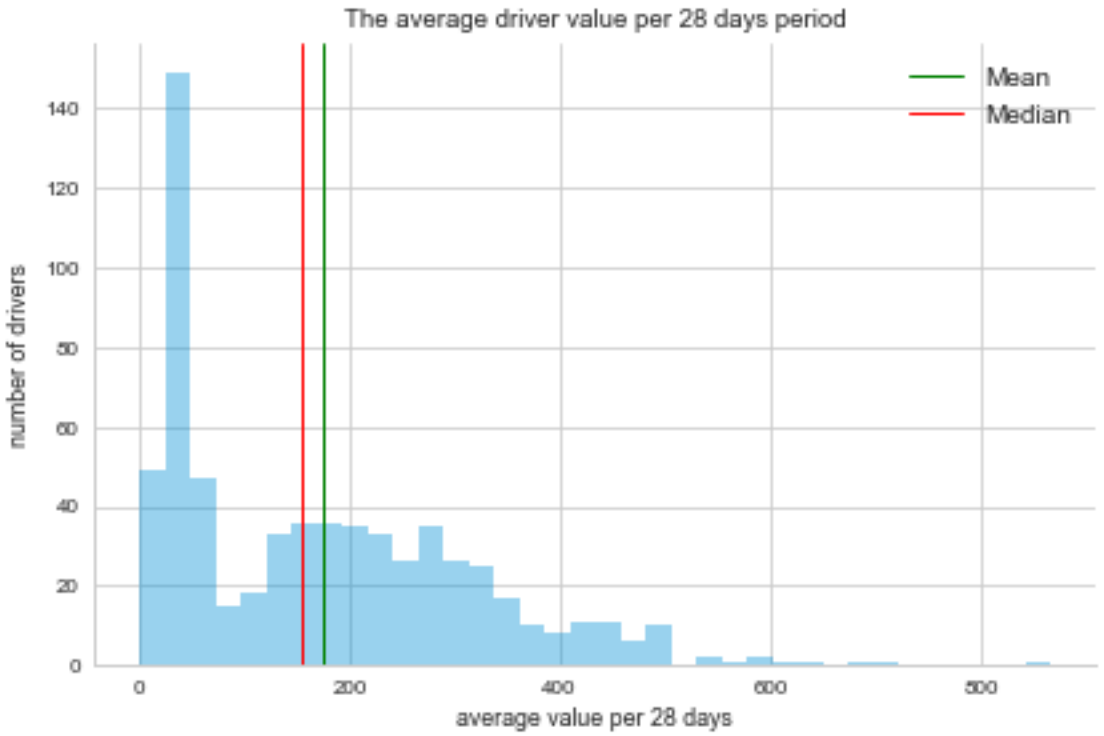
```
Out[210]: count    646.000000
          mean     173.773234
          std      142.538999
          min        0.713571
          25%       41.113086
          50%      153.618596
          75%      269.146842
          max      866.220860
          Name: Average_value, dtype: float64
```

```
In [211]: #boxplot for average value per driver
          sns.boxplot(y=total_value_per_driver['Average_value'], palette="vlag")
          sns.set_context('paper')
          sns.set_style("whitegrid")
          sns.despine(bottom=False)
          plt.show()
```




```
In [212]: ax = sns.distplot(total_value_per_driver['Average_value'],bins=36, hist=True, kde=Fa
          ax.set_title('The average driver value per 28 days period')
          sns.set_context('poster')
          sns.set(style='whitegrid')
          sns.despine(bottom=False)
          ax.set_ylabel('number of drivers')
          ax.set_xlabel('average value per 28 days')
          ax.axvline(total_value_per_driver['Average_value'].mean(), color = 'green', linewidth
          ax.axvline(total_value_per_driver['Average_value'].median(), color = 'red', linewidth
          ax.legend(['Mean', 'Median'])
```

```
plt.savefig("average_driver_value.png")
plt.show()
```




```
In [213]: plt.savefig('Average driver value per 28 days perdiod.png')
```

```
<Figure size 432x288 with 0 Axes>
```

```
In [214]: total_value_per_driver.to_pickle('/Users/zoezhou/Documents/total_value_per_driver.pi
```

6 ----------

7 Part 4 - Modelling

7.0.1 Crucial point:It follows a geometric distribution. The expected value of represent the average lifetime of a driver

7.0.2 E(X)=1/probability_of_churn

```
In [215]: import numpy as np
          import pandas as pd
          import matplotlib.pyplot as plt
          import math
          from datetime import timedelta
```

```
import warnings
warnings.filterwarnings("ignore")

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score,
from sklearn.preprocessing import StandardScaler
from sklearn import feature_selection
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LinearRegression, LogisticRegression

import pylab
pylab.rcParams['figure.figsize'] = 12, 8

% matplotlib inline
```

7.1 Load the data

Highlight - the dataset is imbalanced

```
In [216]: total_value = pd.read_pickle('/Users/zoezhou/Documents/total_value_per_driver.pickle
```

```
In [217]: total_value.head()
```

Out[217]:

	driver_id	Total_Driver_value_first_period	\
0	002be0ffdc997bd5c50703158b7c2491	193.227749	
1	007f0389f9c7b03ef97098422f902e62	13.425076	
2	011e5c5dfc5c2c92501b8b24d47509bc	25.335678	
3	0152a2f305e71d26cc964f8d4411add9	103.085295	
4	01674381af7edd264113d4e6ed55ecda	347.275006	

	Total_Driver_value_second_period	Average_value
0	98.191695	145.709722
1	21.977370	17.701223
2	39.728101	32.531890
3	187.999764	145.542529
4	387.827477	367.551242

```
In [218]: churn_df = pd.read_pickle('/Users/zoezhou/Documents/churn_df.pickle')
```

```
In [219]: churn_df.head()
```

Out[219]:

	driver_id	#_rides	#_active_days	\
0	002be0ffdc997bd5c50703158b7c2491	109	21	
1	007f0389f9c7b03ef97098422f902e62	7	3	
2	011e5c5dfc5c2c92501b8b24d47509bc	12	5	
3	0152a2f305e71d26cc964f8d4411add9	56	15	
4	01674381af7edd264113d4e6ed55ecda	176	20	

```
   first_28_days_revenue_$  %_prime_time  second_28_days_revenue_$  \
0              1312.907194      0.376147                613.963974
1                87.102761      0.428571                152.819293
2               170.542045      0.416667                268.260552
3               743.278028      0.214286               1331.836383
4              2494.205956      0.261364               2784.087152

   Churn_2nd_period  retention  bucket_by_500
0                 0          1            2.0
1                 0          1            0.0
2                 0          1            0.0
3                 0          1            1.0
4                 0          1            4.0
```

```
In [220]: churn_df['Churn_2nd_period'].value_counts()
```

```
Out[220]: 0    552
          1     94
          Name: Churn_2nd_period, dtype: int64
```

14.5% of the total drivers churned second 28 days period.

```
In [221]: churn_df.head()
```

```
Out[221]:                          driver_id  #_rides  #_active_days  \
          0  002be0ffdc997bd5c50703158b7c2491      109             21
          1  007f0389f9c7b03ef97098422f902e62        7              3
          2  011e5c5dfc5c2c92501b8b24d47509bc       12              5
          3  0152a2f305e71d26cc964f8d4411add9       56             15
          4  01674381af7edd264113d4e6ed55ecda      176             20

             first_28_days_revenue_$  %_prime_time  second_28_days_revenue_$  \
          0              1312.907194      0.376147                613.963974
          1                87.102761      0.428571                152.819293
          2               170.542045      0.416667                268.260552
          3               743.278028      0.214286               1331.836383
          4              2494.205956      0.261364               2784.087152

             Churn_2nd_period  retention  bucket_by_500
          0                 0          1            2.0
          1                 0          1            0.0
          2                 0          1            0.0
          3                 0          1            1.0
          4                 0          1            4.0
```

7.1.1 The features to predict driver's churn are:

- Number of rides
- The total ride price per driver

- The number of active days (at least one ride)
- The percentage of rides with prime time

```
In [222]: X_all = churn_df[[col for col in churn_df.columns if col in ['#_rides','first_28_days
          target = churn_df['Churn_2nd_period']

In [223]: X_all.shape

Out[223]: (646, 4)

In [224]: X_all
```

Out[224]:

	#_rides	#_active_days	first_28_days_revenue_$	%_prime_time
0	109	21	1312.907194	0.376147
1	7	3	87.102761	0.428571
2	12	5	170.542045	0.416667
3	56	15	743.278028	0.214286
4	176	20	2494.205956	0.261364
5	203	22	2516.720877	0.241379
6	110	18	1527.668793	0.300000
7	47	11	641.259187	0.212766
8	53	21	745.005119	0.113208
9	29	17	318.130982	0.275862
10	82	14	1018.583527	0.256098
11	98	17	1345.648190	0.357143
12	43	8	504.332784	0.186047
13	46	11	789.657192	0.434783
14	51	10	680.510118	0.274510
15	165	20	1975.525445	0.345455
16	142	18	2109.353535	0.274648
17	173	26	1994.460700	0.375723
18	114	18	1359.521015	0.131579
19	33	6	363.042230	0.303030
20	190	26	2752.667626	0.373684
21	200	22	3033.671690	0.325000
22	22	6	212.150908	0.136364
23	84	7	1211.522124	0.345238
24	75	13	834.257221	0.506667
25	70	12	742.716762	0.071429
26	77	13	1005.852858	0.350649
27	68	14	1019.717570	0.176471
28	45	11	462.291178	0.155556
29	7	2	122.973636	0.000000
..	...	...	...	...
616	118	23	1674.481656	0.254237
617	37	4	445.896965	0.081081
618	106	13	1414.539028	0.292453
619	37	10	489.738973	0.432432
620	241	24	2872.776502	0.315353

```
621       52              20              739.794399      0.346154
622       24               6              337.830254      0.500000
623       26               6              330.976537      0.230769
624      123              19             2082.149690      0.130081
625       43               6              595.301802      0.255814
626       27              16              386.726970      0.555556
627      230              24             3028.274289      0.513043
628       38               6              380.497420      0.184211
629       47              16              572.401999      0.042553
630       43              15              756.268232      0.279070
631       80              12             1228.450299      0.450000
632       71              19             1098.548065      0.183099
633       82              14             1184.282668      0.146341
634       94              16             1336.364440      0.212766
635       71              14              942.399087      0.352113
636       97              16             1179.777977      0.247423
637       81              13             1507.570488      0.481481
638       32               4              447.546735      0.312500
639      277              22             3285.730983      0.209386
640      152              19             1897.907369      0.289474
641      168              17             1996.329480      0.291667
642       33               7              472.505454      0.393939
643      110              18             1646.700987      0.318182
644      221              23             2723.968159      0.276018
645       29              13              270.975182      0.103448

[646 rows x 4 columns]
```

8 Split data

```
In [225]: X_train, X_test, y_train, y_test = train_test_split(X_all,target,test_size=0.3)

In [226]: X_train.shape

Out[226]: (452, 4)
```

8.0.1 model #1 - Logistic regression

```
In [227]: lr = LogisticRegression()

In [228]: #fit the model
          lr.fit(X_train,y_train)

Out[228]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                    intercept_scaling=1, max_iter=100, multi_class='warn',
                    n_jobs=None, penalty='l2', random_state=None, solver='warn',
                    tol=0.0001, verbose=0, warm_start=False)

In [229]: print(lr.coef_)
```

```
[[ 0.01134125 -0.16528373 -0.00125279 -0.08206313]]
```

```
In [230]: lr.score(X_train,y_train)
```

```
Out[230]: 0.8473451327433629
```

```
In [231]: Y_train_pred = lr.predict(X_train)
          Y_test_pred = lr.predict(X_test)
```

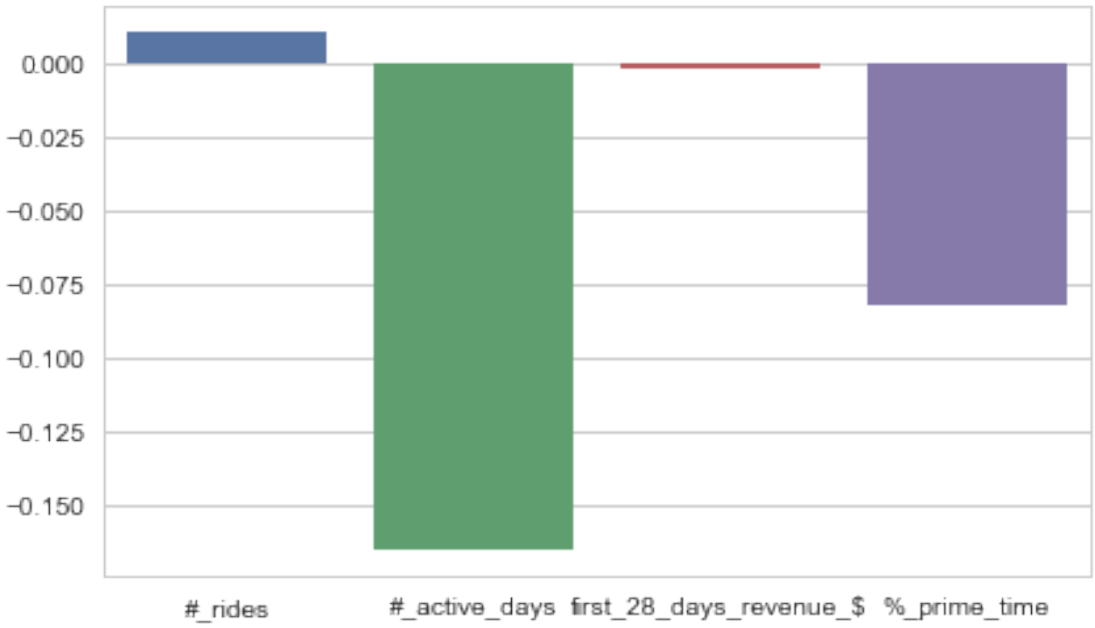
```
In [232]: #Train
          print(classification_report(y_train, Y_train_pred))
```

```
              precision    recall  f1-score   support

           0       0.86      0.98      0.92       386
           1       0.38      0.08      0.13        66

   micro avg       0.85      0.85      0.85       452
   macro avg       0.62      0.53      0.52       452
weighted avg       0.79      0.85      0.80       452
```

```
In [233]: #Test
          print(classification_report(y_test, Y_test_pred))
```

```
              precision    recall  f1-score   support

           0       0.85      0.99      0.92       166
           1       0.00      0.00      0.00        28

   micro avg       0.85      0.85      0.85       194
   macro avg       0.43      0.49      0.46       194
weighted avg       0.73      0.85      0.78       194
```

```
In [234]: sns.barplot(x=X_train.columns,y=lr.coef_[0])
          plt.show()
```




```
In [235]: # The expected lifetime is equal to 1/probabaility_of_churn

          lifetime_28_days_period = ([1/x[1] for x in lr.predict_proba(X_all)])

In [236]: X_all['Driver_Lifetime']= lifetime_28_days_period

In [237]: X_all['driver_id']= churn_df['driver_id']

In [238]: X_all
```

Out[238]:

	#_rides	#_active_days	first_28_days_revenue_$	%_prime_time
0	109	21	1312.907194	0.376147
1	7	3	87.102761	0.428571
2	12	5	170.542045	0.416667
3	56	15	743.278028	0.214286
4	176	20	2494.205956	0.261364
5	203	22	2516.720877	0.241379
6	110	18	1527.668793	0.300000
7	47	11	641.259187	0.212766
8	53	21	745.005119	0.113208
9	29	17	318.130982	0.275862
10	82	14	1018.583527	0.256098
11	98	17	1345.648190	0.357143
12	43	8	504.332784	0.186047
13	46	11	789.657192	0.434783
14	51	10	680.510118	0.274510
15	165	20	1975.525445	0.345455

16	142	18	2109.353535	0.274648
17	173	26	1994.460700	0.375723
18	114	18	1359.521015	0.131579
19	33	6	363.042230	0.303030
20	190	26	2752.667626	0.373684
21	200	22	3033.671690	0.325000
22	22	6	212.150908	0.136364
23	84	7	1211.522124	0.345238
24	75	13	834.257221	0.506667
25	70	12	742.716762	0.071429
26	77	13	1005.852858	0.350649
27	68	14	1019.717570	0.176471
28	45	11	462.291178	0.155556
29	7	2	122.973636	0.000000
..	...	...	...	...
616	118	23	1674.481656	0.254237
617	37	4	445.896965	0.081081
618	106	13	1414.539028	0.292453
619	37	10	489.738973	0.432432
620	241	24	2872.776502	0.315353
621	52	20	739.794399	0.346154
622	24	6	337.830254	0.500000
623	26	6	330.976537	0.230769
624	123	19	2082.149690	0.130081
625	43	6	595.301802	0.255814
626	27	16	386.726970	0.555556
627	230	24	3028.274289	0.513043
628	38	6	380.497420	0.184211
629	47	16	572.401999	0.042553
630	43	15	756.268232	0.279070
631	80	12	1228.450299	0.450000
632	71	19	1098.548065	0.183099
633	82	14	1184.282668	0.146341
634	94	16	1336.364440	0.212766
635	71	14	942.399087	0.352113
636	97	16	1179.777977	0.247423
637	81	13	1507.570488	0.481481
638	32	4	447.546735	0.312500
639	277	22	3285.730983	0.209386
640	152	19	1897.907369	0.289474
641	168	17	1996.329480	0.291667
642	33	7	472.505454	0.393939
643	110	18	1646.700987	0.318182
644	221	23	2723.968159	0.276018
645	29	13	270.975182	0.103448

	Driver_Lifetime	driver_id
0	31.780404	002be0ffdc997bd5c50703158b7c2491

1	2.080258	007f0389f9c7b03ef97098422f902e62
2	2.575569	011e5c5dfc5c2c92501b8b24d47509bc
3	11.067888	0152a2f305e71d26cc964f8d4411add9
4	54.103127	01674381af7edd264113d4e6ed55ecda
5	56.877186	0213f8b59219e32142711992ca4ec01f
6	25.106682	0258e250ca195cc6258cbdc75aecd853
7	6.064999	02e440f6c209206375833cef02e0cbae
8	28.908911	036f3d94e7c65e4e3574822d31c72656
9	12.229716	039c5afbca8e03e4c18d9c8ea94140ac
10	10.002724	03f2b5c74cb89f39e58711699e76bf39
11	19.726793	03f5278eb43475aa6790f5be32463755
12	3.713215	0430df9a3eb327122c57ee3a64765000
13	7.282869	04c4ffa5a385eab86fa7e422263d2999
14	5.331637	052bba06c5fc0bdea4bc2f9cb92b37c7
15	32.629503	05addf442c147875efa5cf53453ad47b
16	35.681971	0656192a402808805282e60761bda088
17	80.939380	06c848ab3a7fc5421e82e98850a81710
18	19.405478	07dd442e3e0b9f0f9b0d69c7b47cbb06
19	2.847016	081d8ba3bc9a00a481df02bd9d0a4c53
20	171.404909	08a1491d6a804e0af969f08252ddbbd8
21	112.238342	08b2b063cce8d02495c4b880293f153c
22	2.708494	0938ed763cb3129ae63607aaf69daff5
23	4.549782	0afc0241296972b583debd7c5f5c707c
24	7.694402	0b631e16fa61f7321da18cf35a076d5f
25	6.167019	0c02bd2b09f7193103279ab9b760b777
26	9.010555	0ca501b2a1d72e80e0d0cd2c25bdd124
27	11.498065	0e7f0f05c7e193b1774c2e5713741cd4
28	5.121147	0eff1404b137a5562642f0f706e59f25
29	1.924673	0f057c0c73054f569a59a0880b91cbb0
..	...	...
616	61.240727	f17cfca756365f6863a241ea96ab9f75
617	2.381576	f1b4411717c78f67380366c2a16a4d1e
618	10.574351	f2a9db857bbd5fe385ed59d8f2e89621
619	5.049873	f395649fb47860aebc4817c7a6ea90e6
620	80.431194	f54b6feed73d306d44d8fba250bafea8
621	25.230009	f5efba2f8019f9eff955fa20312d0639
622	3.014198	f696de645de36b56677457d2d3136524
623	2.909533	f758703e18f588f2370783f8b779e664
624	49.474749	f7858e1e354a9fa26b3055bc12a4ee5b
625	3.197362	f86eb77e1cefe28e9f0e9d3775fae261
626	11.857450	f91254f1c1b3112ef3464e477d23c9e8
627	112.127118	f9aa6d69d74e786544027a1ab3049f44
628	2.766468	fac81ea6cbd540c89c7eee17e851a233
629	11.470278	faebff3d5429ff2036c125a91df765c8
630	12.921907	fb83fc6555a4f700fd92630d9fcb9cea
631	9.744932	fb903879c556260ae2604ae0c45cb92a
632	26.607574	fba8372d56b91b1bff7b71d970b5af58
633	11.980349	fc3504d2efaaaa976a33b3c856927155

```
634        17.225075  fc83b793850ea70d9e898afd0b3ef592
635        10.343835  fd2130d0d215069168dc2f79c1f5ae44
636        13.925570  fd39748ba122e84e9ac492e6fb7c7a05
637        15.507714  fda96e6cd3395dfae3e59cd4ac95f7d7
638         2.493301  fde60697758e68d617f471e49f65db75
639        64.098049  fdff1a7205bc3b9ab1dc5dc223782fc5
640        29.061804  fe35f74209d1056dd315ddb17681203d
641        20.027105  fe469488a23d4bdda47b83a659dcc103
642         3.517963  fed19d671569afe8a2f9fa0953dd25ca
643        29.025197  ff419a3476e21e269e340b5f1f05414e
644        70.879912  ff714a67ba8c6a108261cd81e3b77f3a
645         6.388156  fff482c704d36a1afe8b8978d5486283

[646 rows x 6 columns]
```

```
In [239]: value_per_driver = pd.read_pickle('/Users/zoezhou/Documents/total_value_per_driver.p
```

```
In [240]: driver_lifetime_value = pd.merge(X_all,value_per_driver,how='inner',on='driver_id')
```

```
In [241]: #driver ltv=driver life time * average valueAve
          driver_lifetime_value['Driver_lifetime_value'] = driver_lifetime_value['Driver_Lifet
```

```
In [242]: driver_lifetime_value.head()
```

Out[242]:

	#_rides	#_active_days	first_28_days_revenue_$	%_prime_time	\
0	109	21	1312.907194	0.376147	
1	7	3	87.102761	0.428571	
2	12	5	170.542045	0.416667	
3	56	15	743.278028	0.214286	
4	176	20	2494.205956	0.261364	

	Driver_Lifetime	driver_id	\
0	31.780404	002be0ffdc997bd5c50703158b7c2491	
1	2.080258	007f0389f9c7b03ef97098422f902e62	
2	2.575569	011e5c5dfc5c2c92501b8b24d47509bc	
3	11.067888	0152a2f305e71d26cc964f8d4411add9	
4	54.103127	01674381af7edd264113d4e6ed55ecda	

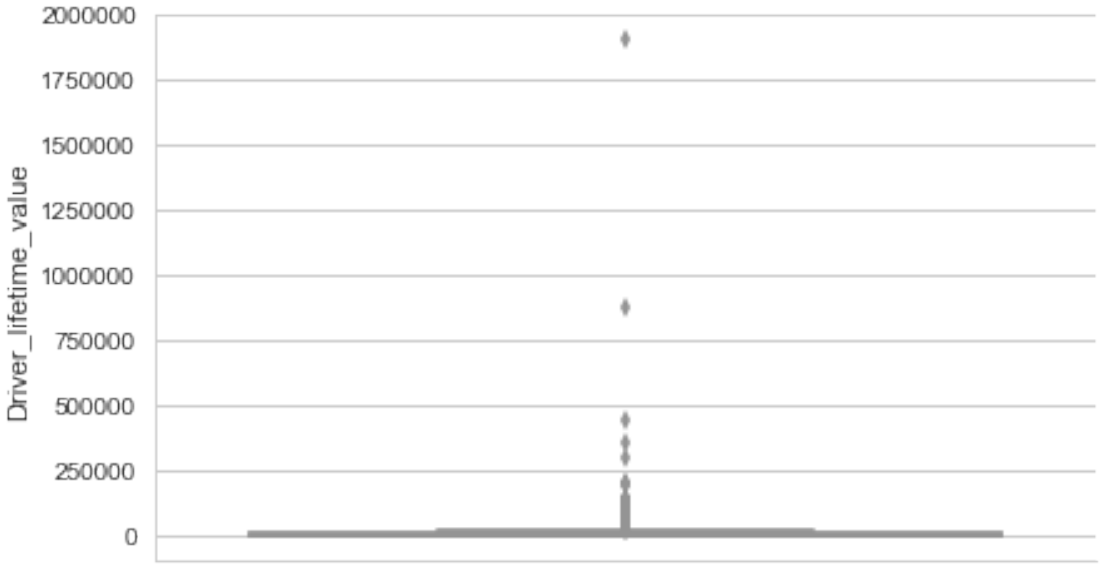
	Total_Driver_value_first_period	Total_Driver_value_second_period	\
0	193.227749	98.191695	
1	13.425076	21.977370	
2	25.335678	39.728101	
3	103.085295	187.999764	
4	347.275006	387.827477	

	Average_value	Driver_lifetime_value
0	145.709722	4630.713869
1	17.701223	36.823106
2	32.531890	83.788113

```
3     145.542529            1610.848479
4     367.551242           19885.671344
```

Boxplot of driver lifetime value

```
In [243]: sns.boxplot(y=driver_lifetime_value['Driver_lifetime_value'], palette="vlag")
          sns.set_context('paper')
          sns.set_style("darkgrid")
          sns.despine(bottom=False)
          plt.show()
```

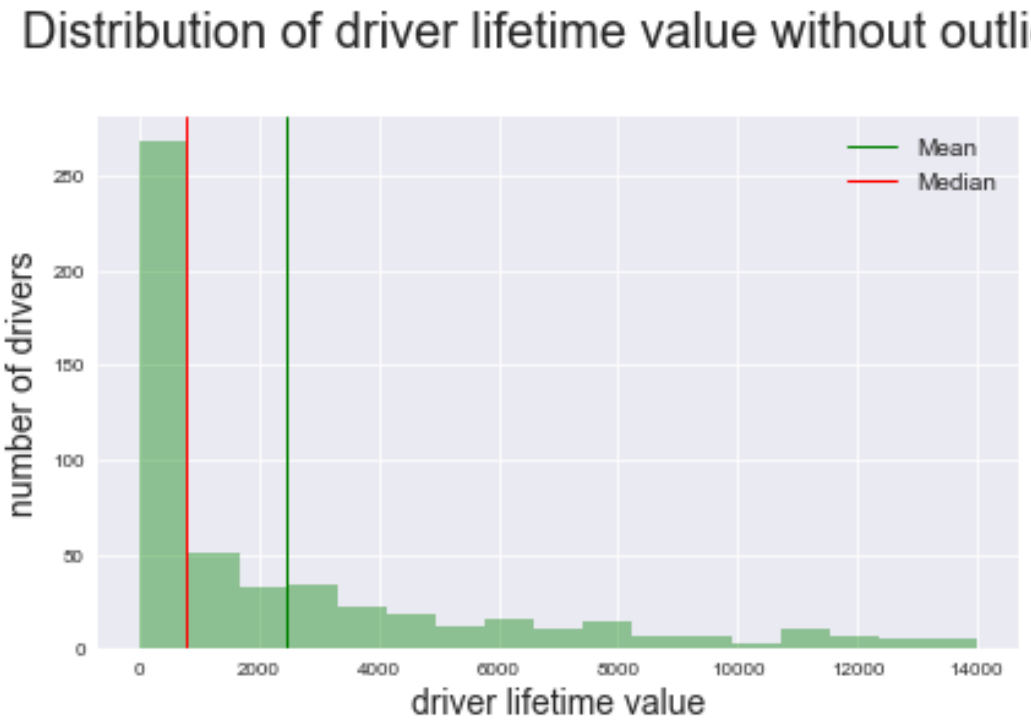



```
In [244]: driver_lifetime_value['Driver_lifetime_value'].describe()

Out[244]: count    6.460000e+02
          mean     1.664964e+04
          std      8.866205e+04
          min      1.276384e+00
          25%      1.745443e+02
          50%      1.823200e+03
          75%      8.903690e+03
          max      1.901254e+06
          Name: Driver_lifetime_value, dtype: float64
```

There are outliers in driver_lifetime_value, so we should remove the outliers.
Upper_limit=Q3+1.5*IQR*=*5687+1.55575*=5687+8362.5=14049.5

```
In [245]: driver_lifetime_value_new = driver_lifetime_value[driver_lifetime_value.Driver_lifeti
```

```
In [246]: ax = sns.distplot(driver_lifetime_value_new['Driver_lifetime_value'], hist=True, kde=
          ax.set_title('Distribution of driver lifetime value without outliers\n', size = 20)
          sns.set_context('poster')
          sns.set(style='darkgrid')
          sns.despine(bottom=False)
          ax.set_ylabel('number of drivers', size = 14)
          ax.set_xlabel('driver lifetime value', size = 14)
          ax.axvline(driver_lifetime_value_new['Driver_lifetime_value'].mean(), color = 'green
          ax.axvline(driver_lifetime_value_new['Driver_lifetime_value'].median(), color = 'red
          ax.legend(['Mean', 'Median'])
          plt.show()
```




```
In [247]: driver_lifetime_value_new['Driver_lifetime_value'].describe()

Out[247]: count      524.000000
          mean      2465.414940
          std       3375.828081
          min          1.276384
          25%        130.360499
          50%        771.022770
          75%       3549.842133
          max      14017.938155
          Name: Driver_lifetime_value, dtype: float64
```

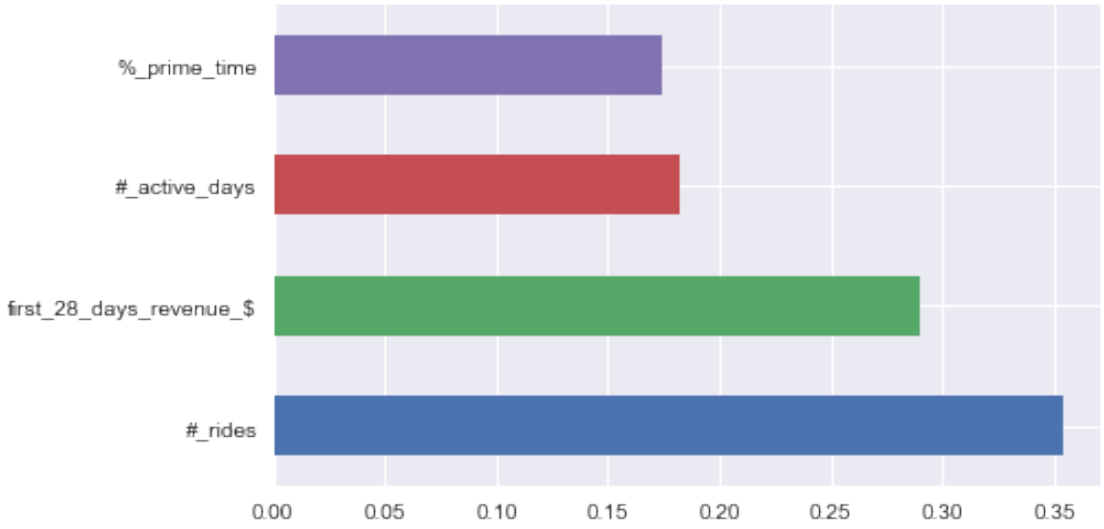
The driver lifetime value after removing the outliers has mean of 1972 and median of 529.

9 ----------

9.1 Random Forest Classifier is used to get the importance of each feature

A quick analysis shows that the number of active days is strong predictor, followed by first 28 days revenue and the number of ride.

```
In [248]: rfc = RandomForestClassifier()
          rfc.fit(X_train,y_train)
          y_pred = rfc.predict(X_test)

In [249]: def significance(cols):
              if cols['p-value']<0.001:
                  return '***'
              if cols['p-value']<0.01:
                  return '**'
              if cols['p-value']<0.1:
                  return '*'
              else:
                  return ''

In [250]: c=feature_selection.f_classif(X_train,y_train)[1].tolist()
          summary=pd.concat([pd.DataFrame(data=np.transpose(abs(rfc.feature_importances_)),inde
                                          columns=['Coef']), pd.DataFrame(c,index=X_all.iloc[:
          summary['significance']=pd.DataFrame(summary.apply(significance,axis=1))

          df_relevant_feature_rfc = summary.sort_values(by='p-value',ascending=True)
          df_relevant_feature_rfc

Out[250]:                            Coef       p-value significance
          #_active_days            0.182419  2.785359e-17          ***
          first_28_days_revenue_$  0.289725  9.057031e-11          ***
          #_rides                  0.353554  1.663170e-10          ***
          %_prime_time             0.174302  3.644326e-02            *

In [251]: import matplotlib.pyplot as plt
          %matplotlib inline

          feature_importances = pd.Series(rfc.feature_importances_, index = X_train.columns)
          feature_importances.nlargest(4).plot(kind='barh')

Out[251]: <matplotlib.axes._subplots.AxesSubplot at 0x1a28eefc88>
```



Thanks for reading