Cycle Data clean part 03-Short Pass Holders

April 27, 2018

1 Cycle Data Cleaning Temporaray Pass Holder

Declaration : The central idea and coding is abstract from Kevin mark ham youtube video seriese, Introduction to machine learning with scikit-learn video series. You can find link under resources section.

2 Libraries

```
In [1]: # load libraries and set styles, options
        import os,csv
        import numpy as np
        import pandas as pd
        import seaborn as sns
        import warnings; warnings.simplefilter('ignore')
        #from IPython.display import HTML
        #HTML('<iframe src=http://www.seattle.gov/documents/departments/sdot/newmobilityprogra
```

```
In [2]: %matplotlib inline
```

2. Read and verify data

```
In [3]: # read in a CSV
        df = pd.read_csv('C:/Users/mrferozi/Desktop/GitHub/Bike/cycle-share-dataset-orignal/tri
```

```
In [4]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 286857 entries, 0 to 286856
Data columns (total 12 columns):
trip_id              286857 non-null int64
starttime            286857 non-null object
stoptime             286857 non-null object
bikeid               286857 non-null object
tripduration         286857 non-null float64
from_station_name    286857 non-null object
to_station_name      286857 non-null object
from_station_id      286857 non-null object
```

```
to_station_id        286857 non-null object
usertype             286857 non-null object
gender               181557 non-null object
birthyear            181553 non-null float64
dtypes: float64(2), int64(1), object(9)
memory usage: 26.3+ MB
```

```
In [5]: # Filter data and only consider trip duration more than 3 minute and exclude Short-Ter
        bikes = df[(df.tripduration >=300) & (df.usertype == 'Short-Term Pass Holder')]
```

```
In [6]: bikes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 101545 entries, 69 to 286856
Data columns (total 12 columns):
trip_id              101545 non-null int64
starttime            101545 non-null object
stoptime             101545 non-null object
bikeid               101545 non-null object
tripduration         101545 non-null float64
from_station_name    101545 non-null object
to_station_name      101545 non-null object
from_station_id      101545 non-null object
to_station_id        101545 non-null object
usertype             101545 non-null object
gender               0 non-null object
birthyear            0 non-null float64
dtypes: float64(2), int64(1), object(9)
memory usage: 10.1+ MB
```

- After applying fillter our data set shrink from 286857 to 143541 tuples

3 Handle missing values

What does "NaN" mean?

- "NaN" is not a string, rather it's a special value: `numpy.nan`.
- It stands for "Not a Number" and indicates a **missing value**.
- `read_csv` detects missing values (by default) when reading the file, and replaces them with this special value.

```
In [7]: # 'isnull' returns a DataFrame of booleans (True if missing, False if not missing)
        bikes.isnull().tail()
```

```
Out[7]:         trip_id  starttime  stoptime  bikeid  tripduration  from_station_name  \
        286852    False      False     False   False         False              False
```

```
        286853    False      False     False   False         False              False
        286854    False      False     False   False         False              False
        286855    False      False     False   False         False              False
        286856    False      False     False   False         False              False

                to_station_name  from_station_id  to_station_id  usertype  gender  \
        286852            False           False          False     False    True
        286853            False           False          False     False    True
        286854            False           False          False     False    True
        286855            False           False          False     False    True
        286856            False           False          False     False    True

                birthyear
        286852       True
        286853       True
        286854       True
        286855       True
        286856       True
```

As this study aware that short pass holders are the user who could be a tourist or not a regular customer. Due to fact that they are temporary users, Short pass holder not obliged to fill their personal information such as date of birth or gender information. That's why this study decided to get rid of those columns completely and Since we could not find any mean or average value to fill in those columns we have decided to exclude them from our study analysis for short pass holders.

```
In [8]: # 'nonnull' returns the opposite of 'isnull' (True if not missing, False if missing)
        bikes.notnull().tail()
```

```
Out[8]:         trip_id  starttime  stoptime  bikeid  tripduration  from_station_name  \
        286852     True       True      True    True          True               True
        286853     True       True      True    True          True               True
        286854     True       True      True    True          True               True
        286855     True       True      True    True          True               True
        286856     True       True      True    True          True               True

                to_station_name  from_station_id  to_station_id  usertype  gender  \
        286852             True            True           True      True   False
        286853             True            True           True      True   False
        286854             True            True           True      True   False
        286855             True            True           True      True   False
        286856             True            True           True      True   False

                birthyear
        286852      False
        286853      False
        286854      False
        286855      False
        286856      False
```

```
In [9]: # count the number of missing values in each Series
        bikes.isnull().sum()

Out[9]: trip_id                  0
        starttime                0
        stoptime                 0
        bikeid                   0
        tripduration             0
        from_station_name        0
        to_station_name          0
        from_station_id          0
        to_station_id            0
        usertype                 0
        gender              101545
        birthyear           101545
        dtype: int64
```

Handle missing values depends on the dataset as well as the nature of analysis. As we have mention earlier, we have droping gender and birthyear columns

```
In [19]: bikes.drop(['gender', 'birthyear'], axis=1, inplace=True)

In [20]: bikes.shape

Out[20]: (101545, 10)

In [21]: # count the number of missing values in each Series
         bikes.isnull().sum()

Out[21]: trip_id              0
         starttime            0
         stoptime             0
         bikeid               0
         tripduration         0
         from_station_name    0
         to_station_name      0
         from_station_id      0
         to_station_id        0
         usertype             0
         dtype: int64
```

4 Data Transformation

converting data Categorical to Numaric for further use

- Converting category From Station ID and To Station ID to nominal

```
In [22]: bikes["from_station_id_cat"] = bikes["from_station_id"].astype('category')
         bikes.dtypes
```

```
Out[22]: trip_id                  int64
         starttime               object
         stoptime                object
         bikeid                  object
         tripduration           float64
         from_station_name       object
         to_station_name         object
         from_station_id         object
         to_station_id           object
         usertype                object
         from_station_id_cat   category
         dtype: object

In [23]: bikes["from_station_id_num"] = bikes["from_station_id_cat"].cat.codes

In [24]: bikes["to_station_id_cat"] = bikes["to_station_id"].astype('category')
         bikes.dtypes

Out[24]: trip_id                  int64
         starttime               object
         stoptime                object
         bikeid                  object
         tripduration           float64
         from_station_name       object
         to_station_name         object
         from_station_id         object
         to_station_id           object
         usertype                object
         from_station_id_cat   category
         from_station_id_num       int8
         to_station_id_cat     category
         dtype: object

In [25]: bikes["to_station_id_num"] = bikes["to_station_id_cat"].cat.codes
```

- Handlind Date and Time

For further, for our analysis we are dividing date columns in small portion. Which, could help us to understand dataset more clearly.

```
In [26]: # convert 'Time' to datetime format
         bikes['starttime'] = pd.to_datetime(bikes.starttime)

In [27]: bikes.dtypes

Out[27]: trip_id                        int64
         starttime             datetime64[ns]
         stoptime                      object
         bikeid                        object
```

```
tripduration                 float64
from_station_name             object
to_station_name               object
from_station_id               object
to_station_id                 object
usertype                      object
from_station_id_cat          category
from_station_id_num             int8
to_station_id_cat            category
to_station_id_num               int8
dtype: object
```

```
In [28]: bikes['Day'] = bikes.starttime.dt.weekday_name
```

```
In [29]: # convert 'Time' to datetime format
         bikes['stoptime'] = pd.to_datetime(bikes.stoptime)
```

```
In [30]: bikes["Day_cat"] = bikes["Day"].astype('category')
         bikes.dtypes
```

```
Out[30]: trip_id                       int64
         starttime            datetime64[ns]
         stoptime             datetime64[ns]
         bikeid                       object
         tripduration                float64
         from_station_name            object
         to_station_name              object
         from_station_id              object
         to_station_id                object
         usertype                     object
         from_station_id_cat        category
         from_station_id_num           int8
         to_station_id_cat          category
         to_station_id_num             int8
         Day                          object
         Day_cat                    category
         dtype: object
```

```
In [31]: bikes["Day_num"] = bikes["Day_cat"].cat.codes
```

```
In [32]: # convenient Series attributes are now available
         bikes["sthours"] = bikes.starttime.dt.hour
```

```
In [33]: # convenient Series attributes are now available
         bikes["stphours"] = bikes.stoptime.dt.hour
```

```
In [34]: bikes['tripduration_minutes']=bikes['tripduration']/60
         desred_decimals = 2
         bikes['tripduration_minutes'] = bikes['tripduration_minutes'].apply(lambda x: round(x
```

```
In [36]: bikes.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 101545 entries, 69 to 286856
Data columns (total 20 columns):
trip_id                 101545 non-null int64
starttime               101545 non-null datetime64[ns]
stoptime                101545 non-null datetime64[ns]
bikeid                  101545 non-null object
tripduration            101545 non-null float64
from_station_name       101545 non-null object
to_station_name         101545 non-null object
from_station_id         101545 non-null object
to_station_id           101545 non-null object
usertype                101545 non-null object
from_station_id_cat     101545 non-null category
from_station_id_num     101545 non-null int8
to_station_id_cat       101545 non-null category
to_station_id_num       101545 non-null int8
Day                     101545 non-null object
Day_cat                 101545 non-null category
Day_num                 101545 non-null int8
sthours                 101545 non-null int64
stphours                101545 non-null int64
tripduration_minutes    101545 non-null float64
dtypes: category(3), datetime64[ns](2), float64(2), int64(3), int8(3), object(7)
memory usage: 12.2+ MB
```

5 Removing Duplicated Tuples

```
In [37]: # detect duplicate trip_id codes: True if an item is identical to a previous item
         bikes.trip_id.duplicated().tail()

Out[37]: 286852    False
         286853    False
         286854    False
         286855    False
         286856    False
         Name: trip_id, dtype: bool
```

The above codes shows that we do not have any duplicated rows in our dataset

```
In [38]: # Breaking date column for further Analysis
         bikes['bmonth'] = bikes['starttime'].dt.month

In [39]: # Breaking date column for further Analysis
         bikes['Date'] = bikes['starttime'].dt.date
```

```
In [40]: bikes.dtypes
```

```
Out[40]: trip_id                          int64
         starttime                datetime64[ns]
         stoptime                 datetime64[ns]
         bikeid                           object
         tripduration                    float64
         from_station_name                object
         to_station_name                  object
         from_station_id                  object
         to_station_id                    object
         usertype                         object
         from_station_id_cat            category
         from_station_id_num                int8
         to_station_id_cat              category
         to_station_id_num                  int8
         Day                              object
         Day_cat                        category
         Day_num                            int8
         sthours                           int64
         stphours                          int64
         tripduration_minutes            float64
         bmonth                            int64
         Date                             object
         dtype: object
```

```
In [41]: bikes.Day_num.tail()
```

```
Out[41]: 286852    6
         286853    6
         286854    6
         286855    6
         286856    6
         Name: Day_num, dtype: int8
```

---

6 Finalizing Bike Data and narrow down for year value

```
In [42]: # convert 'Time' to datetime format
         bikes['starttime'] = pd.to_datetime(bikes.starttime)
```

```
In [43]: # Breaking date column for further Analysis
         bikes['year'] = bikes['starttime'].dt.year
```

```
In [44]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 286857 entries, 0 to 286856
```

```
Data columns (total 10 columns):
trip_id              286857 non-null int64
starttime            286857 non-null object
stoptime             286857 non-null object
bikeid               286857 non-null object
tripduration         286857 non-null float64
from_station_name    286857 non-null object
to_station_name      286857 non-null object
from_station_id      286857 non-null object
to_station_id        286857 non-null object
usertype             286857 non-null object
dtypes: float64(1), int64(1), object(8)
memory usage: 21.9+ MB
```

```
In [45]: # convert 'Time' to datetime format
         bikes['Date'] = pd.to_datetime(bikes.Date)
         bikes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 101545 entries, 69 to 286856
Data columns (total 23 columns):
trip_id                 101545 non-null int64
starttime               101545 non-null datetime64[ns]
stoptime                101545 non-null datetime64[ns]
bikeid                  101545 non-null object
tripduration            101545 non-null float64
from_station_name       101545 non-null object
to_station_name         101545 non-null object
from_station_id         101545 non-null object
to_station_id           101545 non-null object
usertype                101545 non-null object
from_station_id_cat     101545 non-null category
from_station_id_num     101545 non-null int8
to_station_id_cat       101545 non-null category
to_station_id_num       101545 non-null int8
Day                     101545 non-null object
Day_cat                 101545 non-null category
Day_num                 101545 non-null int8
sthours                 101545 non-null int64
stphours                101545 non-null int64
tripduration_minutes    101545 non-null float64
bmonth                  101545 non-null int64
Date                    101545 non-null datetime64[ns]
year                    101545 non-null int64
dtypes: category(3), datetime64[ns](3), float64(2), int64(5), int8(3), object(7)
memory usage: 14.5+ MB
```

```
In [46]: bikes.Date.tail()
```

```
Out[46]: 286852   2016-08-31
         286853   2016-08-31
         286854   2016-08-31
         286855   2016-08-31
         286856   2016-08-31
         Name: Date, dtype: datetime64[ns]
```

```
In [48]: # df2 belong to bikes dataset
         df2 = bikes[(bikes.year != 2014) ]
         df2.head(100)
```

```
Out[48]:        trip_id           starttime            stoptime    bikeid  \
         20239    25091 2015-01-01 00:24:00 2015-01-01 00:48:00  SEA00325
         20242    25094 2015-01-01 01:36:00 2015-01-01 01:42:00  SEA00327
         20244    25100 2015-01-01 01:54:00 2015-01-01 02:36:00  SEA00374
         20245    25102 2015-01-01 01:56:00 2015-01-01 02:36:00  SEA00469
         20246    25104 2015-01-01 02:01:00 2015-01-01 02:09:00  SEA00483
         20247    25105 2015-01-01 02:02:00 2015-01-01 02:09:00  SEA00194
         20248    25106 2015-01-01 02:10:00 2015-01-01 02:32:00  SEA00470
         20249    25109 2015-01-01 02:22:00 2015-01-01 02:34:00  SEA00342
         20250    25111 2015-01-01 02:26:00 2015-01-01 02:56:00  SEA00249
         20253    25115 2015-01-01 03:24:00 2015-01-01 04:40:00  SEA00157
         20256    25128 2015-01-01 08:23:00 2015-01-01 08:56:00  SEA00486
         20257    25129 2015-01-01 08:23:00 2015-01-01 08:56:00  SEA00176
         20258    25130 2015-01-01 08:49:00 2015-01-01 09:27:00  SEA00401
         20259    25131 2015-01-01 10:14:00 2015-01-01 10:33:00  SEA00204
         20262    25134 2015-01-01 11:09:00 2015-01-01 11:31:00  SEA00434
         20264    25136 2015-01-01 11:34:00 2015-01-01 11:45:00  SEA00107
         20266    25138 2015-01-01 11:47:00 2015-01-01 16:20:00  SEA00031
         20267    25139 2015-01-01 11:48:00 2015-01-01 16:20:00  SEA00389
         20268    25140 2015-01-01 11:56:00 2015-01-01 12:11:00  SEA00147
         20269    25142 2015-01-01 12:00:00 2015-01-01 12:12:00  SEA00210
         20272    25145 2015-01-01 12:15:00 2015-01-01 12:57:00  SEA00147
         20273    25146 2015-01-01 12:16:00 2015-01-01 12:57:00  SEA00200
         20275    25149 2015-01-01 12:52:00 2015-01-01 13:04:00  SEA00371
         20278    25152 2015-01-01 12:53:00 2015-01-01 13:05:00  SEA00457
         20279    25153 2015-01-01 12:56:00 2015-01-01 13:16:00  SEA00364
         20280    25154 2015-01-01 12:58:00 2015-01-01 13:16:00  SEA00192
         20283    25157 2015-01-01 13:00:00 2015-01-01 13:20:00  SEA00086
         20284    25158 2015-01-01 13:03:00 2015-01-01 13:17:00  SEA00466
         20285    25159 2015-01-01 13:03:00 2015-01-01 13:22:00  SEA00419
         20286    25160 2015-01-01 13:05:00 2015-01-01 13:21:00  SEA00213
         ...        ...                 ...                 ...       ...
         20336    25216 2015-01-01 15:19:00 2015-01-01 15:40:00  SEA00193
         20338    25218 2015-01-01 15:20:00 2015-01-01 15:40:00  SEA00305
         20339    25219 2015-01-01 15:20:00 2015-01-01 16:56:00  SEA00469
```

```
20343    25223 2015-01-01 15:30:00 2015-01-01 15:42:00  SEA00100
20344    25224 2015-01-01 15:30:00 2015-01-01 15:42:00  SEA00246
20345    25225 2015-01-01 15:39:00 2015-01-01 16:05:00  SEA00481
20349    25230 2015-01-01 15:47:00 2015-01-01 17:55:00  SEA00248
20353    25234 2015-01-01 16:08:00 2015-01-01 16:16:00  SEA00304
20354    25235 2015-01-01 16:09:00 2015-01-01 16:36:00  SEA00481
20355    25236 2015-01-01 16:14:00 2015-01-01 16:31:00  SEA00234
20359    25240 2015-01-01 16:27:00 2015-01-01 16:54:00  SEA00486
20360    25241 2015-01-01 16:27:00 2015-01-01 16:56:00  SEA00457
20361    25242 2015-01-01 16:27:00 2015-01-01 16:56:00  SEA00082
20362    25243 2015-01-01 16:37:00 2015-01-01 17:06:00  SEA00234
20369    25250 2015-01-01 17:27:00 2015-01-01 17:32:00  SEA00151
20370    25251 2015-01-01 17:28:00 2015-01-01 17:50:00  SEA00387
20373    25254 2015-01-01 18:06:00 2015-01-01 18:25:00  SEA00387
20377    25258 2015-01-01 19:30:00 2015-01-01 20:00:00  SEA00121
20378    25259 2015-01-01 19:33:00 2015-01-01 20:01:00  SEA00236
20380    25261 2015-01-01 19:36:00 2015-01-01 20:02:00  SEA00287
20381    25262 2015-01-01 19:37:00 2015-01-01 20:01:00  SEA00446
20382    25263 2015-01-01 20:05:00 2015-01-01 20:37:00  SEA00287
20384    25265 2015-01-01 20:12:00 2015-01-01 20:37:00  SEA00446
20385    25266 2015-01-01 20:13:00 2015-01-01 20:37:00  SEA00101
20386    25267 2015-01-01 20:15:00 2015-01-01 20:37:00  SEA00121
20387    25268 2015-01-01 20:15:00 2015-01-01 20:37:00  SEA00047
20390    25271 2015-01-01 20:43:00 2015-01-01 20:52:00  SEA00446
20391    25272 2015-01-01 20:43:00 2015-01-01 20:52:00  SEA00280
20392    25273 2015-01-01 20:43:00 2015-01-01 20:52:00  SEA00047
20393    25274 2015-01-01 20:44:00 2015-01-01 20:52:00  SEA00121

       tripduration                                from_station_name  \
20239      1403.479          Lake Union Park / Valley St & Boren Ave N
20242       373.146                            12th Ave & E Mercer St
20244      2527.307                           Summit Ave & E Denny Way
20245      2401.616                           Summit Ave & E Denny Way
20246       463.284                                 3rd Ave & Broad St
20247       374.388                                 3rd Ave & Broad St
20248      1341.749                              Westlake Ave & 6th Ave
20249       713.799                              Westlake Ave & 6th Ave
20250      1764.223             Cal Anderson Park / 11th Ave & Pine St
20253      4578.327  Seattle Aquarium / Alaskan Way S & Elliott Bay...
20256      2007.481                      Pier 69 / Alaskan Way & Clay St
20257      2002.652                      Pier 69 / Alaskan Way & Clay St
20258      2288.270                              2nd Ave & Blanchard St
20259      1145.254                       Summit Ave E & E Republican St
20262      1294.421          Lake Union Park / Valley St & Boren Ave N
20264       666.286                        Eastlake Ave E & E Allison St
20266     16390.548                       NE 42nd St & University Way NE
20267     16321.122                       NE 42nd St & University Way NE
20268       902.576          Lake Union Park / Valley St & Boren Ave N
```

20269	699.075	Lake Union Park / Valley St & Boren Ave N
20272	2502.464	E Blaine St & Fairview Ave E
20273	2449.811	E Blaine St & Fairview Ave E
20275	740.454	3rd Ave & Broad St
20278	682.798	3rd Ave & Broad St
20279	1219.227	E Pine St & 16th Ave
20280	1132.444	E Pine St & 16th Ave
20283	1166.184	9th Ave N & Mercer St
20284	801.236	Cal Anderson Park / 11th Ave & Pine St
20285	1111.246	9th Ave N & Mercer St
20286	923.087	9th Ave N & Mercer St
...	...	...
20336	1248.700	3rd Ave & Broad St
20338	1182.491	3rd Ave & Broad St
20339	5778.924	Summit Ave & E Denny Way
20343	719.348	2nd Ave & Pine St
20344	730.831	2nd Ave & Pine St
20345	1531.261	Seattle Aquarium / Alaskan Way S & Elliott Bay...
20349	7665.406	2nd Ave & Spring St
20353	516.595	E Pine St & 16th Ave
20354	1633.725	Pier 69 / Alaskan Way & Clay St
20355	1010.440	12th Ave & E Denny Way
20359	1640.057	Pier 69 / Alaskan Way & Clay St
20360	1742.493	Pier 69 / Alaskan Way & Clay St
20361	1749.466	12th Ave & E Mercer St
20362	1754.521	6th Ave S & S King St
20369	336.879	Bellevue Ave & E Pine St
20370	1309.895	2nd Ave & Pine St
20373	1152.240	E Pine St & 16th Ave
20377	1832.885	1st Ave & Marion St
20378	1726.987	1st Ave & Marion St
20380	1549.438	1st Ave & Marion St
20381	1475.038	1st Ave & Marion St
20382	1959.209	2nd Ave & Blanchard St
20384	1503.815	2nd Ave & Blanchard St
20385	1471.379	2nd Ave & Blanchard St
20386	1353.139	2nd Ave & Blanchard St
20387	1309.542	2nd Ave & Blanchard St
20390	552.783	3rd Ave & Broad St
20391	564.614	3rd Ave & Broad St
20392	564.241	3rd Ave & Broad St
20393	494.042	3rd Ave & Broad St

	to_station_name	from_station_id	\
20239	12th Ave & E Mercer St	SLU-17	
20242	Summit Ave & E Denny Way	CH-15	
20244	Summit Ave & E Denny Way	CH-01	
20245	Summit Ave & E Denny Way	CH-01	

20246	2nd Ave & Vine St	BT-01
20247	2nd Ave & Vine St	BT-01
20248	Westlake Ave & 6th Ave	SLU-15
20249	Westlake Ave & 6th Ave	SLU-15
20250	NE 42nd St & University Way NE	CH-08
20253	Seattle Aquarium / Alaskan Way S & Elliott Bay...	WF-04
20256	Pier 69 / Alaskan Way & Clay St	WF-01
20257	Pier 69 / Alaskan Way & Clay St	WF-01
20258	Harvard Ave & E Pine St	BT-05
20259	Occidental Park / Occidental Ave S & S Washing...	CH-03
20262	Eastlake Ave E & E Allison St	SLU-17
20264	15th Ave NE & NE 40th St	EL-05
20266	NE 42nd St & University Way NE	UD-02
20267	NE 42nd St & University Way NE	UD-02
20268	E Blaine St & Fairview Ave E	SLU-17
20269	E Blaine St & Fairview Ave E	SLU-17
20272	PATH / 9th Ave & Westlake Ave	EL-03
20273	PATH / 9th Ave & Westlake Ave	EL-03
20275	Seattle Aquarium / Alaskan Way S & Elliott Bay...	BT-01
20278	Seattle Aquarium / Alaskan Way S & Elliott Bay...	BT-01
20279	6th Ave S & S King St	CH-07
20280	6th Ave S & S King St	CH-07
20283	PATH / 9th Ave & Westlake Ave	DPD-01
20284	6th Ave S & S King St	CH-08
20285	PATH / 9th Ave & Westlake Ave	DPD-01
20286	PATH / 9th Ave & Westlake Ave	DPD-01
...	...	...
20336	2nd Ave & Pine St	BT-01
20338	2nd Ave & Pine St	BT-01
20339	NE 47th St & 12th Ave NE	CH-01
20343	Harvard Ave & E Pine St	CBD-13
20344	Harvard Ave & E Pine St	CBD-13
20345	Pier 69 / Alaskan Way & Clay St	WF-04
20349	PATH / 9th Ave & Westlake Ave	CBD-06
20353	E Harrison St & Broadway Ave E	CH-07
20354	Pier 69 / Alaskan Way & Clay St	WF-01
20355	6th Ave S & S King St	CH-06
20359	Seattle Aquarium / Alaskan Way S & Elliott Bay...	WF-01
20360	Seattle Aquarium / Alaskan Way S & Elliott Bay...	WF-01
20361	Lake Union Park / Valley St & Boren Ave N	CH-15
20362	Seattle Aquarium / Alaskan Way S & Elliott Bay...	ID-04
20369	2nd Ave & Pine St	CH-12
20370	E Pine St & 16th Ave	CBD-13
20373	2nd Ave & Pine St	CH-07
20377	2nd Ave & Blanchard St	CBD-05
20378	2nd Ave & Blanchard St	CBD-05
20380	2nd Ave & Blanchard St	CBD-05
20381	2nd Ave & Blanchard St	CBD-05

20382			3rd Ave & Broad St		BT-05	
20384			3rd Ave & Broad St		BT-05	
20385			3rd Ave & Broad St		BT-05	
20386			3rd Ave & Broad St		BT-05	
20387			3rd Ave & Broad St		BT-05	
20390			2nd Ave & Blanchard St		BT-01	
20391			2nd Ave & Blanchard St		BT-01	
20392			2nd Ave & Blanchard St		BT-01	
20393			2nd Ave & Blanchard St		BT-01	

	to_station_id	usertype	...	to_station_id_num	Day	\
20239	CH-15	Short-Term Pass Holder	...	20	Thursday	
20242	CH-01	Short-Term Pass Holder	...	11	Thursday	
20244	CH-01	Short-Term Pass Holder	...	11	Thursday	
20245	CH-01	Short-Term Pass Holder	...	11	Thursday	
20246	BT-03	Short-Term Pass Holder	...	1	Thursday	
20247	BT-03	Short-Term Pass Holder	...	1	Thursday	
20248	SLU-15	Short-Term Pass Holder	...	38	Thursday	
20249	SLU-15	Short-Term Pass Holder	...	38	Thursday	
20250	UD-02	Short-Term Pass Holder	...	47	Thursday	
20253	WF-04	Short-Term Pass Holder	...	59	Thursday	
20256	WF-01	Short-Term Pass Holder	...	57	Thursday	
20257	WF-01	Short-Term Pass Holder	...	57	Thursday	
20258	CH-09	Short-Term Pass Holder	...	18	Thursday	
20259	PS-04	Short-Term Pass Holder	...	30	Thursday	
20262	EL-05	Short-Term Pass Holder	...	26	Thursday	
20264	UW-04	Short-Term Pass Holder	...	52	Thursday	
20266	UD-02	Short-Term Pass Holder	...	47	Thursday	
20267	UD-02	Short-Term Pass Holder	...	47	Thursday	
20268	EL-03	Short-Term Pass Holder	...	25	Thursday	
20269	EL-03	Short-Term Pass Holder	...	25	Thursday	
20272	SLU-07	Short-Term Pass Holder	...	37	Thursday	
20273	SLU-07	Short-Term Pass Holder	...	37	Thursday	
20275	WF-04	Short-Term Pass Holder	...	59	Thursday	
20278	WF-04	Short-Term Pass Holder	...	59	Thursday	
20279	ID-04	Short-Term Pass Holder	...	29	Thursday	
20280	ID-04	Short-Term Pass Holder	...	29	Thursday	
20283	SLU-07	Short-Term Pass Holder	...	37	Thursday	
20284	ID-04	Short-Term Pass Holder	...	29	Thursday	
20285	SLU-07	Short-Term Pass Holder	...	37	Thursday	
20286	SLU-07	Short-Term Pass Holder	...	37	Thursday	
...	...	...	...	...	...	
20336	CBD-13	Short-Term Pass Holder	...	9	Thursday	
20338	CBD-13	Short-Term Pass Holder	...	9	Thursday	
20339	UD-07	Short-Term Pass Holder	...	49	Thursday	
20343	CH-09	Short-Term Pass Holder	...	18	Thursday	
20344	CH-09	Short-Term Pass Holder	...	18	Thursday	
20345	WF-01	Short-Term Pass Holder	...	57	Thursday	

20349	SLU-07	Short-Term Pass Holder	...	37	Thursday	
20353	CH-02	Short-Term Pass Holder	...	12	Thursday	
20354	WF-01	Short-Term Pass Holder	...	57	Thursday	
20355	ID-04	Short-Term Pass Holder	...	29	Thursday	
20359	WF-04	Short-Term Pass Holder	...	59	Thursday	
20360	WF-04	Short-Term Pass Holder	...	59	Thursday	
20361	SLU-17	Short-Term Pass Holder	...	40	Thursday	
20362	WF-04	Short-Term Pass Holder	...	59	Thursday	
20369	CBD-13	Short-Term Pass Holder	...	9	Thursday	
20370	CH-07	Short-Term Pass Holder	...	16	Thursday	
20373	CBD-13	Short-Term Pass Holder	...	9	Thursday	
20377	BT-05	Short-Term Pass Holder	...	3	Thursday	
20378	BT-05	Short-Term Pass Holder	...	3	Thursday	
20380	BT-05	Short-Term Pass Holder	...	3	Thursday	
20381	BT-05	Short-Term Pass Holder	...	3	Thursday	
20382	BT-01	Short-Term Pass Holder	...	0	Thursday	
20384	BT-01	Short-Term Pass Holder	...	0	Thursday	
20385	BT-01	Short-Term Pass Holder	...	0	Thursday	
20386	BT-01	Short-Term Pass Holder	...	0	Thursday	
20387	BT-01	Short-Term Pass Holder	...	0	Thursday	
20390	BT-05	Short-Term Pass Holder	...	3	Thursday	
20391	BT-05	Short-Term Pass Holder	...	3	Thursday	
20392	BT-05	Short-Term Pass Holder	...	3	Thursday	
20393	BT-05	Short-Term Pass Holder	...	3	Thursday	

	Day_cat	Day_num	sthours	stphours	tripduration_minutes	bmonth	\
20239	Thursday	4	0	0	23.39	1	
20242	Thursday	4	1	1	6.22	1	
20244	Thursday	4	1	2	42.12	1	
20245	Thursday	4	1	2	40.03	1	
20246	Thursday	4	2	2	7.72	1	
20247	Thursday	4	2	2	6.24	1	
20248	Thursday	4	2	2	22.36	1	
20249	Thursday	4	2	2	11.90	1	
20250	Thursday	4	2	2	29.40	1	
20253	Thursday	4	3	4	76.31	1	
20256	Thursday	4	8	8	33.46	1	
20257	Thursday	4	8	8	33.38	1	
20258	Thursday	4	8	9	38.14	1	
20259	Thursday	4	10	10	19.09	1	
20262	Thursday	4	11	11	21.57	1	
20264	Thursday	4	11	11	11.10	1	
20266	Thursday	4	11	16	273.18	1	
20267	Thursday	4	11	16	272.02	1	
20268	Thursday	4	11	12	15.04	1	
20269	Thursday	4	12	12	11.65	1	
20272	Thursday	4	12	12	41.71	1	
20273	Thursday	4	12	12	40.83	1	

20275	Thursday	4	12	13	12.34	1
20278	Thursday	4	12	13	11.38	1
20279	Thursday	4	12	13	20.32	1
20280	Thursday	4	12	13	18.87	1
20283	Thursday	4	13	13	19.44	1
20284	Thursday	4	13	13	13.35	1
20285	Thursday	4	13	13	18.52	1
20286	Thursday	4	13	13	15.38	1
...	...	...	...	...	...	...
20336	Thursday	4	15	15	20.81	1
20338	Thursday	4	15	15	19.71	1
20339	Thursday	4	15	16	96.32	1
20343	Thursday	4	15	15	11.99	1
20344	Thursday	4	15	15	12.18	1
20345	Thursday	4	15	16	25.52	1
20349	Thursday	4	15	17	127.76	1
20353	Thursday	4	16	16	8.61	1
20354	Thursday	4	16	16	27.23	1
20355	Thursday	4	16	16	16.84	1
20359	Thursday	4	16	16	27.33	1
20360	Thursday	4	16	16	29.04	1
20361	Thursday	4	16	16	29.16	1
20362	Thursday	4	16	17	29.24	1
20369	Thursday	4	17	17	5.61	1
20370	Thursday	4	17	17	21.83	1
20373	Thursday	4	18	18	19.20	1
20377	Thursday	4	19	20	30.55	1
20378	Thursday	4	19	20	28.78	1
20380	Thursday	4	19	20	25.82	1
20381	Thursday	4	19	20	24.58	1
20382	Thursday	4	20	20	32.65	1
20384	Thursday	4	20	20	25.06	1
20385	Thursday	4	20	20	24.52	1
20386	Thursday	4	20	20	22.55	1
20387	Thursday	4	20	20	21.83	1
20390	Thursday	4	20	20	9.21	1
20391	Thursday	4	20	20	9.41	1
20392	Thursday	4	20	20	9.40	1
20393	Thursday	4	20	20	8.23	1

	Date	year
20239	2015-01-01	2015
20242	2015-01-01	2015
20244	2015-01-01	2015
20245	2015-01-01	2015
20246	2015-01-01	2015
20247	2015-01-01	2015
20248	2015-01-01	2015

```
20249 2015-01-01  2015
20250 2015-01-01  2015
20253 2015-01-01  2015
20256 2015-01-01  2015
20257 2015-01-01  2015
20258 2015-01-01  2015
20259 2015-01-01  2015
20262 2015-01-01  2015
20264 2015-01-01  2015
20266 2015-01-01  2015
20267 2015-01-01  2015
20268 2015-01-01  2015
20269 2015-01-01  2015
20272 2015-01-01  2015
20273 2015-01-01  2015
20275 2015-01-01  2015
20278 2015-01-01  2015
20279 2015-01-01  2015
20280 2015-01-01  2015
20283 2015-01-01  2015
20284 2015-01-01  2015
20285 2015-01-01  2015
20286 2015-01-01  2015
...          ...   ...
20336 2015-01-01  2015
20338 2015-01-01  2015
20339 2015-01-01  2015
20343 2015-01-01  2015
20344 2015-01-01  2015
20345 2015-01-01  2015
20349 2015-01-01  2015
20353 2015-01-01  2015
20354 2015-01-01  2015
20355 2015-01-01  2015
20359 2015-01-01  2015
20360 2015-01-01  2015
20361 2015-01-01  2015
20362 2015-01-01  2015
20369 2015-01-01  2015
20370 2015-01-01  2015
20373 2015-01-01  2015
20377 2015-01-01  2015
20378 2015-01-01  2015
20380 2015-01-01  2015
20381 2015-01-01  2015
20382 2015-01-01  2015
20384 2015-01-01  2015
20385 2015-01-01  2015
```

```
20386 2015-01-01  2015
20387 2015-01-01  2015
20390 2015-01-01  2015
20391 2015-01-01  2015
20392 2015-01-01  2015
20393 2015-01-01  2015

[100 rows x 23 columns]
```

7 Saving Bike data

```
In [49]: df2.to_csv('C:\\Users\\mrferozi\\Desktop\\GitHub\\Bike\\dataset\\cycle\\trip_clean_sh
```

7.1 Resources

References:*From the video series: Introduction to machine learning with scikit-learn* - scikit-learn documentation: Cross-validation, Model evaluation - scikit-learn issue on GitHub: MSE is negative when returned by cross_val_score - Section 5.1 of An Introduction to Statistical Learning (11 pages) and related videos: K-fold and leave-one-out cross-validation (14 minutes), Cross-validation the right and wrong ways (10 minutes) - Scott Fortmann-Roe: Accurately Measuring Model Prediction Error - Machine Learning Mastery: An Introduction to Feature Selection - Harvard CS109: Cross-Validation: The Right and Wrong Way - Journal of Cheminformatics: Cross-validation pitfalls when selecting and assessing regression and classification models