**HW2 Train Dataset**

**Tyler Mann**

**Appendix A:**

**Import in the different packages as needed, numpy is the significant package needed to run the data frames. I created the path to the dataset from my computer, read the dataset, and then printed it.**

**Appendix B:**

**Shows the details of the printed data the columns and rows.**

**Appendix C:**

**Shows the first 5 rows of the dataset and their columns including missing data. “Origin Call” and “Origin Stand” is they seem to be specific locations so trying to identify those locations based on the “Trip ID” may help or looking at the “Polyline” data. If not since there is so much of it missing it may have to not be included.**

**Appendix D:**

**Shows the column headers and what types they are, integers, floats, object, Booleans.**

**Appendix E:**

**This shows the counts of the column types.**

**Appendix F:**

**If they are considered “object” then it would show as true for the columns, if not it will show as false. I then split them into number variables and category variables and print each of them.**

**Appendix G:**

**This shows true or false for the rows in each column. If the data in the column is null it will say true, if it is not, it will say false. It does this to all rows within the dataset. I then want to sort how many are null for both the number and category variables. The number variables show a lot of missing values while category variables do not.**

**Appendix H:**

**This shows the shape of the data frame, how many columns and how many rows in the entire dataset.**

**Appendix I:**

**If I want to randomly choose a sample within the dataset, I must do a random permutation based on the parameters set of 1% of the total dataset, this will amount to around 17,107 random rows. These rows are captured within an array and then I print out the sample.**

**Appendix J:**

**This is the descriptive statistics of the dataset.**

**Appendix J & K:**

**If you look at the statistical analysis there is very little to no change between the population and sample statistics.**

**#Appendix A**

**import** **numpy** **as** **np**

**import** **scipy** **as** **sp**

**import** **matplotlib** **as** **mpl**

**import** **seaborn** **as** **sns**

**import** **pandas** **as** **pd**

In [2]:

path1 = r"C:\Users\tyler\OneDrive\Desktop\Tyler stuff\Predictive Modeling\train.csv"

In [4]:

df = pd.read\_csv(path1)

In [5]:

**#Appendix B**

print(df)

TRIP\_ID CALL\_TYPE ORIGIN\_CALL ORIGIN\_STAND TAXI\_ID \

0 1372636858620000589 C NaN NaN 20000589

1 1372637303620000596 B NaN 7.0 20000596

2 1372636951620000320 C NaN NaN 20000320

3 1372636854620000520 C NaN NaN 20000520

4 1372637091620000337 C NaN NaN 20000337

... ... ... ... ... ...

1710665 1404171463620000698 C NaN NaN 20000698

1710666 1404171367620000670 C NaN NaN 20000670

1710667 1388745716620000264 C NaN NaN 20000264

1710668 1404141826620000248 B NaN 12.0 20000248

1710669 1404157147620000079 B NaN 34.0 20000079

TIMESTAMP DAY\_TYPE MISSING\_DATA \

0 1372636858 A False

1 1372637303 A False

2 1372636951 A False

3 1372636854 A False

4 1372637091 A False

... ... ... ...

1710665 1404171463 A False

1710666 1404171367 A False

1710667 1388745716 A False

1710668 1404141826 A False

1710669 1404157147 A False

POLYLINE

0 [[-8.618643,41.141412],[-8.618499,41.141376],[...

1 [[-8.639847,41.159826],[-8.640351,41.159871],[...

2 [[-8.612964,41.140359],[-8.613378,41.14035],[-...

3 [[-8.574678,41.151951],[-8.574705,41.151942],[...

4 [[-8.645994,41.18049],[-8.645949,41.180517],[-...

... ...

1710665 [[-8.612469,41.14602],[-8.612487,41.145993],[-...

1710666 [[-8.610138,41.140845],[-8.610174,41.140935],[...

1710667 []

1710668 [[-8.630712,41.154885],[-8.63073,41.154813],[-...

1710669 [[-8.615538,41.140629],[-8.615421,41.140746],[...

[1710670 rows x 9 columns]

In [6]:

**#Appendix C**

df.head(5)

Out[6]:

|  | **TRIP\_ID** | **CALL\_TYPE** | **ORIGIN\_CALL** | **ORIGIN\_STAND** | **TAXI\_ID** | **TIMESTAMP** | **DAY\_TYPE** | **MISSING\_DATA** | **POLYLINE** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1372636858620000589 | C | NaN | NaN | 20000589 | 1372636858 | A | False | [[-8.618643,41.141412],[-8.618499,41.141376],[... |
| **1** | 1372637303620000596 | B | NaN | 7.0 | 20000596 | 1372637303 | A | False | [[-8.639847,41.159826],[-8.640351,41.159871],[... |
| **2** | 1372636951620000320 | C | NaN | NaN | 20000320 | 1372636951 | A | False | [[-8.612964,41.140359],[-8.613378,41.14035],[-... |
| **3** | 1372636854620000520 | C | NaN | NaN | 20000520 | 1372636854 | A | False | [[-8.574678,41.151951],[-8.574705,41.151942],[... |
| **4** | 1372637091620000337 | C | NaN | NaN | 20000337 | 1372637091 | A | False | [[-8.645994,41.18049],[-8.645949,41.180517],[-... |

In [7]:

**#Appendix D**

df.dtypes

Out[7]:

TRIP\_ID int64

CALL\_TYPE object

ORIGIN\_CALL float64

ORIGIN\_STAND float64

TAXI\_ID int64

TIMESTAMP int64

DAY\_TYPE object

MISSING\_DATA bool

POLYLINE object

dtype: object

In [8]:

**#Appendix E**

df.dtypes.value\_counts()

Out[8]:

int64 3

object 3

float64 2

bool 1

dtype: int64

In [9]:

**#Appendix F**

df.dtypes == 'object'

Out[9]:

TRIP\_ID False

CALL\_TYPE True

ORIGIN\_CALL False

ORIGIN\_STAND False

TAXI\_ID False

TIMESTAMP False

DAY\_TYPE True

MISSING\_DATA False

POLYLINE True

dtype: bool

In [10]:

num\_vars = df.columns[df.dtypes != 'object']

cat\_vars = df.columns[df.dtypes == 'object']

In [11]:

print(num\_vars)

print(cat\_vars)

Index(['TRIP\_ID', 'ORIGIN\_CALL', 'ORIGIN\_STAND', 'TAXI\_ID', 'TIMESTAMP',

'MISSING\_DATA'],

dtype='object')

Index(['CALL\_TYPE', 'DAY\_TYPE', 'POLYLINE'], dtype='object')

In [12]:

**#Appendix G**

df[num\_vars].isnull()

Out[12]:

|  | **TRIP\_ID** | **ORIGIN\_CALL** | **ORIGIN\_STAND** | **TAXI\_ID** | **TIMESTAMP** | **MISSING\_DATA** |
| --- | --- | --- | --- | --- | --- | --- |
| **0** | False | True | True | False | False | False |
| **1** | False | True | False | False | False | False |
| **2** | False | True | True | False | False | False |
| **3** | False | True | True | False | False | False |
| **4** | False | True | True | False | False | False |
| **...** | ... | ... | ... | ... | ... | ... |
| **1710665** | False | True | True | False | False | False |
| **1710666** | False | True | True | False | False | False |
| **1710667** | False | True | True | False | False | False |
| **1710668** | False | True | False | False | False | False |
| **1710669** | False | True | False | False | False | False |

1710670 rows × 6 columns

In [13]:

df[num\_vars].isnull().sum().sort\_values(ascending=**False**)

Out[13]:

ORIGIN\_CALL 1345900

ORIGIN\_STAND 904091

MISSING\_DATA 0

TIMESTAMP 0

TAXI\_ID 0

TRIP\_ID 0

dtype: int64

In [14]:

df[cat\_vars].isnull().sum().sort\_values(ascending=**False**)

Out[14]:

POLYLINE 0

DAY\_TYPE 0

CALL\_TYPE 0

dtype: int64

In [15]:

*#***Appendix H**

df.shape

Out[15]:

(1710670, 9)

In [16]:

**#Appendix I**

**#Create an object to select 17,107 random samples**

sampler = np.random.permutation(17107)

In [17]:

**#This is an array of the random sample chosen by Python**

sampler

Out[17]:

array([14603, 5197, 12962, ..., 9279, 16657, 5419])

In [18]:

df.take(sampler)

Out[18]:

|  | **TRIP\_ID** | **CALL\_TYPE** | **ORIGIN\_CALL** | **ORIGIN\_STAND** | **TAXI\_ID** | **TIMESTAMP** | **DAY\_TYPE** | **MISSING\_DATA** | **POLYLINE** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **14603** | 1372921659620000687 | A | 2002.0 | NaN | 20000687 | 1372921659 | A | False | [[-8.589645,41.182263],[-8.59005,41.183361],[-... |
| **5197** | 1372749816620000561 | B | NaN | 15.0 | 20000561 | 1372749816 | A | False | [[-8.585676,41.148585],[-8.585712,41.148927],[... |
| **12962** | 1372877277620000254 | B | NaN | 23.0 | 20000254 | 1372877277 | A | False | [[-8.612568,41.145966],[-8.61246,41.145948],[-... |
| **1294** | 1372672878620000036 | A | 15689.0 | NaN | 20000036 | 1372672878 | A | False | [[-8.676324,41.154975],[-8.676198,41.154876],[... |
| **15883** | 1372933347620000549 | C | NaN | NaN | 20000549 | 1372933347 | A | False | [[-8.627211,41.151942],[-8.627103,41.151996],[... |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **9499** | 1372826248620000483 | B | NaN | 14.0 | 20000483 | 1372826248 | A | False | [[-8.611065,41.149476],[-8.611128,41.14944],[-... |
| **3655** | 1372703326620000352 | A | 4796.0 | NaN | 20000352 | 1372703326 | A | False | [[-8.604081,41.149692],[-8.604099,41.149683],[... |
| **9279** | 1372821868620000195 | B | NaN | 60.0 | 20000195 | 1372821868 | A | False | [[-8.609499,41.151204],[-8.609418,41.151321],[... |
| **16657** | 1372945026620000331 | B | NaN | 57.0 | 20000331 | 1372945026 | A | False | [[-8.610885,41.145741],[-8.610822,41.145966],[... |
| **5419** | 1372749935620000153 | A | 7956.0 | NaN | 20000153 | 1372749935 | A | False | [[-8.605683,41.165388],[-8.605278,41.165397],[... |

17107 rows × 9 columns

In [19]:

**#The view of the shape of the random sample**

df.sample(n=17107).shape

Out[19]:

(17107, 9)

In [20]:

**#The random sampler can also be applied this way. This method allows you to reduce the size of the random sample**

df.sample(n=17107)

Out[20]:

|  | **TRIP\_ID** | **CALL\_TYPE** | **ORIGIN\_CALL** | **ORIGIN\_STAND** | **TAXI\_ID** | **TIMESTAMP** | **DAY\_TYPE** | **MISSING\_DATA** | **POLYLINE** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **83108** | 1374139988620000038 | C | NaN | NaN | 20000038 | 1374139988 | A | False | [[-8.609553,41.15124],[-8.609571,41.151681],[-... |
| **989949** | 1391225092620000361 | C | NaN | NaN | 20000361 | 1391225092 | A | False | [[-8.613693,41.147703],[-8.613432,41.147631],[... |
| **43781** | 1373409354620000388 | A | 2002.0 | NaN | 20000388 | 1373409354 | A | False | [[-8.67996,41.156469],[-8.679906,41.156532],[-... |
| **385075** | 1380015218620000018 | A | 3606.0 | NaN | 20000018 | 1380015218 | A | False | [[-8.659845,41.169762],[-8.65989,41.16978],[-8... |
| **1675302** | 1403570936620000672 | C | NaN | NaN | 20000672 | 1403570936 | A | False | [[-8.600148,41.182749],[-8.599788,41.182659],[... |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **1657279** | 1403280092620000597 | A | 3633.0 | NaN | 20000597 | 1403280092 | A | False | [[-8.605044,41.177466],[-8.605044,41.177484],[... |
| **1559539** | 1401552533620000329 | B | NaN | 42.0 | 20000329 | 1401552533 | A | False | [[-8.612046,41.172624],[-8.612064,41.172633],[... |
| **658732** | 1384886762620000598 | B | NaN | 12.0 | 20000598 | 1384886762 | A | False | [[-8.630703,41.154813],[-8.630676,41.154822],[... |
| **535578** | 1382605096620000320 | C | NaN | NaN | 20000320 | 1382605096 | A | False | [[-8.623494,41.157801],[-8.623548,41.15772],[-... |
| **1469794** | 1400059383620000101 | B | NaN | 36.0 | 20000101 | 1400059383 | A | False | [[-8.649306,41.154246],[-8.64927,41.154246],[-... |

17107 rows × 9 columns

In [21]:

**#Appendix J**

**#descriptive statistics for the Train dataset**

print (df.describe(include='all'))

TRIP\_ID CALL\_TYPE ORIGIN\_CALL ORIGIN\_STAND TAXI\_ID \

count 1.710670e+06 1710670 364770.000000 806579.000000 1.710670e+06

unique NaN 3 NaN NaN NaN

top NaN B NaN NaN NaN

freq NaN 817881 NaN NaN NaN

mean 1.388622e+18 NaN 24490.363018 30.272381 2.000035e+07

std 9.180944e+15 NaN 19624.290043 17.747840 2.112405e+02

min 1.372637e+18 NaN 2001.000000 1.000000 2.000000e+07

25% 1.380731e+18 NaN 6593.000000 15.000000 2.000017e+07

50% 1.388493e+18 NaN 18755.000000 27.000000 2.000034e+07

75% 1.396750e+18 NaN 40808.000000 49.000000 2.000052e+07

max 1.404173e+18 NaN 63884.000000 63.000000 2.000098e+07

TIMESTAMP DAY\_TYPE MISSING\_DATA POLYLINE

count 1.710670e+06 1710670 1710670 1710670

unique NaN 1 2 1703650

top NaN A False []

freq NaN 1710670 1710660 5901

mean 1.388622e+09 NaN NaN NaN

std 9.180944e+06 NaN NaN NaN

min 1.372637e+09 NaN NaN NaN

25% 1.380731e+09 NaN NaN NaN

50% 1.388493e+09 NaN NaN NaN

75% 1.396750e+09 NaN NaN NaN

max 1.404173e+09 NaN NaN NaN

**#The random sampler can also be applied this way. This method allows you to reduce the size of the random sample**

sa = df.sample(n=17107)

In [43]:

**#Appendix K**

**#descriptive statistics for the Train dataset SAMPLE**

print (sa.describe(include='all'))

TRIP\_ID CALL\_TYPE ORIGIN\_CALL ORIGIN\_STAND TAXI\_ID \

count 1.710700e+04 17107 3622.000000 8028.000000 1.710700e+04

unique NaN 3 NaN NaN NaN

top NaN B NaN NaN NaN

freq NaN 8134 NaN NaN NaN

mean 1.388639e+18 NaN 24781.648537 30.190583 2.000035e+07

std 9.174212e+15 NaN 19711.655679 17.685451 2.106353e+02

min 1.372637e+18 NaN 2001.000000 1.000000 2.000000e+07

25% 1.380660e+18 NaN 6654.000000 15.000000 2.000016e+07

50% 1.388544e+18 NaN 18954.000000 27.000000 2.000034e+07

75% 1.396855e+18 NaN 41110.250000 47.000000 2.000052e+07

max 1.404170e+18 NaN 63882.000000 63.000000 2.000090e+07

TIMESTAMP DAY\_TYPE MISSING\_DATA POLYLINE

count 1.710700e+04 17107 17107 17107

unique NaN 1 1 17051

top NaN A False []

freq NaN 17107 17107 57

mean 1.388639e+09 NaN NaN NaN

std 9.174212e+06 NaN NaN NaN

min 1.372637e+09 NaN NaN NaN

25% 1.380660e+09 NaN NaN NaN

50% 1.388544e+09 NaN NaN NaN

75% 1.396855e+09 NaN NaN NaN

max 1.404170e+09 NaN NaN NaN