COMP 4353 Data Mining
Final Homework Assignment

- Geraldo Braho
- Student ID: 120302835

- 5/3/2017

Global Terrorism Database

Problem

This dataset contains check-ins in NYC and Tokyo collected for about 10 month (from 12 April 2012 to 16 February 2013). It contains 227,428 check-ins in New York city and 573,703 check-ins in Tokyo. Each check-in is associated with its time stamp, its GPS coordinates and its semantic meaning (represented by fine-grained venue-categories). This dataset is originally used for studying the spatial-temporal regularity of user activity in LBSNs.

Step 1 - Data Information

Generating the information about the dataset:

- number of columns and rows,
- names and data types of the columns,
- memory usage of the dataset.

In [4]:
```
# import libraries
import pandas as pd
import numpy as np
```

In [7]:
```
# read the data

dataSet=pd.read_csv("dataset_TSMC2014_NYC.csv")
```

In [8]:
```
# let's check the data
dataSet.head(10)
```

Out[8]:

	userId	venueId	venueCategoryId	venueCategory	la
0	470	49bbd6c0f964a520f4531fe3	4bf58dd8d48988d127951735	Arts & Crafts Store	4(
1	979	4a43c0aef964a520c6a61fe3	4bf58dd8d48988d1df941735	Bridge	4(
2	69	4c5cc7b485a1e21e00d35711	4bf58dd8d48988d103941735	Home (private)	4(
3	395	4bc7086715a7ef3bef9878da	4bf58dd8d48988d104941735	Medical Center	4(
4	87	4cf2c5321d18a143951b5cec	4bf58dd8d48988d1cb941735	Food Truck	4(
5	484	4b5b981bf964a520900929e3	4bf58dd8d48988d118951735	Food & Drink Shop	4(
6	642	4ab966c3f964a5203c7f20e3	4bf58dd8d48988d1e0931735	Coffee Shop	4(
7	292	4d0cc47f903d37041864bf55	4bf58dd8d48988d12b951735	Bus Station	4(
8	428	4ce1863bc4f6a35d8bd2db6c	4bf58dd8d48988d103941735	Home (private)	4(
9	877	4be319b321d5a59352311811	4bf58dd8d48988d10a951735	Bank	4(

In [9]:
```
# # of row, # of columns, data type of each column and memory usage
dataSet.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 227428 entries, 0 to 227427
Data columns (total 8 columns):
userId             227428 non-null int64
venueId            227428 non-null object
venueCategoryId    227428 non-null object
venueCategory      227428 non-null object
latitude           227428 non-null float64
longitude          227428 non-null float64
timezoneOffset     227428 non-null int64
utcTimestamp       227428 non-null object
dtypes: float64(2), int64(2), object(4)
memory usage: 13.9+ MB
```

Step 2 - Descriptive Statistics

Generating descriptive statistics of all columns (input and output) of the dataset.

Descriptive statistics for numerical columns include:

- count,
- mean,
- std,
- min,
- 25 percentile (Q1),
- 50 percentile (Q2, median),
- 75 percentile (Q3),
- max values of the columns.

In [11]:
```
# extract numerical data
# save it into new dataframe named numericalData
# generate descriptive statistics for numericalData
numericalData = dataSet.describe();
numericalData.columns = ['latitude', 'longitude', 'timezoneOffset', 'u
tcTimestamp'];
numericalData
```

Out[11]:

	latitude	longitude	timezoneOffset	utcTimestamp
count	227428.000000	227428.000000	227428.000000	227428.000000
mean	535.304817	40.754045	-73.974556	-253.392019
std	305.398774	0.072458	0.086209	43.234750
min	1.000000	40.550852	-74.274766	-420.000000
25%	287.000000	40.718330	-74.000633	-240.000000
50%	523.000000	40.747745	-73.983479	-240.000000
75%	793.000000	40.778374	-73.945709	-240.000000
max	1083.000000	40.988332	-73.683825	660.000000

In [12]:
```
# extract categorical data (in case there is any)
# save it into new dataframe named categoricalData
# categoricalData = dataSet.select_dtypes(include=['object']).copy();
# categoricalData
```

In [13]:
```
# # determine distinct values in each column
# uniq = []
# for i in categoricalData:
#     uniq.append(categoricalData[i].unique())

# uniq
```

In [14]:
```
# # determine frequencies of the distinct values in each column
# freq = []
# for i in categoricalData:
#     freq.append(categoricalData[i].value_counts())
# freq
```

Step 3 - Analysis of the output column

In [16]:
```
# Calculating inter quartile range, Q3-Q1
IQR_value = numericalData.utcTimestamp.quantile(0.75) - numericalData.
utcTimestamp.quantile(0.25);
IQR_value
```

Out[16]: 440.7740669623248

In [17]:
```
# Calculating the range (difference between max and min value)
range_value = numericalData.utcTimestamp.max() - numericalData.utcTime
stamp.min();
range_value
```

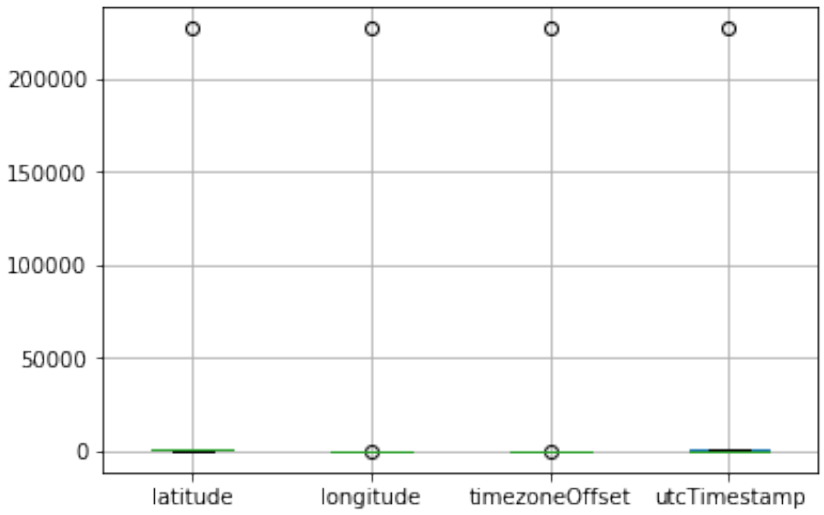
Out[17]: 227848.0

Step 4 - Box Plot

Box plots of all numerical columns

In [19]:
```
import matplotlib.pyplot as plt
%matplotlib inline
numericalData.boxplot()
```

Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x11b2144e0>



Pie plots for all categorical columns

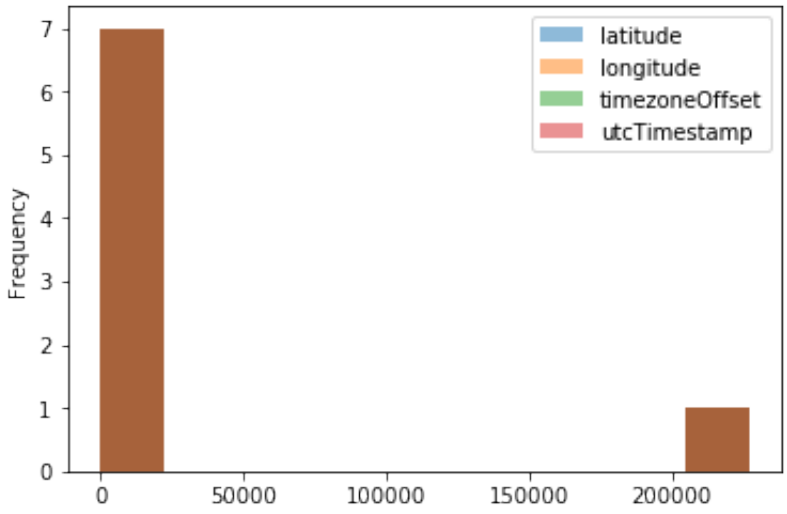
In [20]:
```
# pie plots for categorical data (if any)
#categoricalData.plot.pie(subplots=True, figsize=(50, 10))
```

Step 5 - Distribution of Columns

Histogram of all numerical columns

In [22]:
```
numericalData.plot.hist(alpha=0.5)
```

Out[22]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x11b6fb828>
```

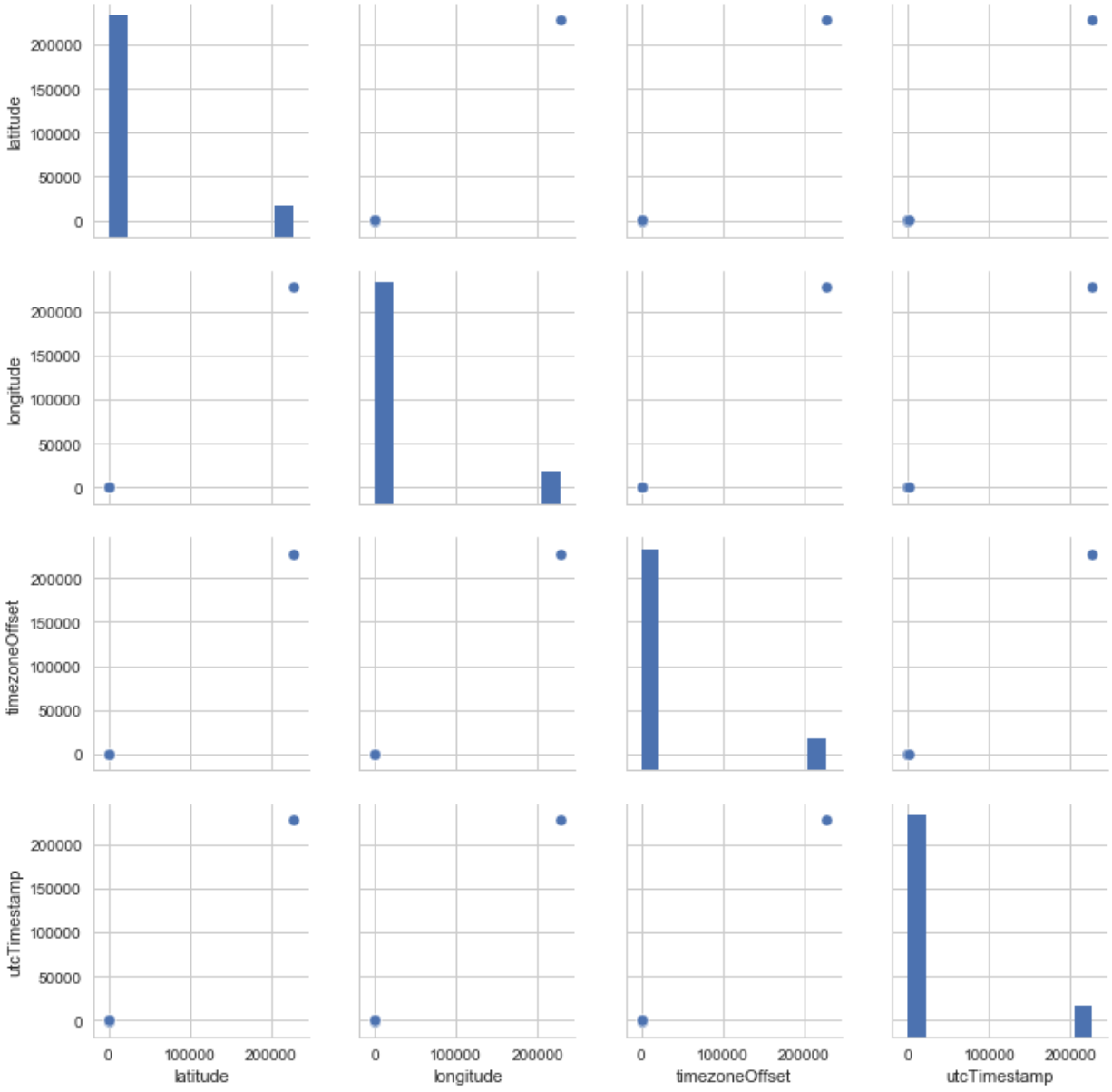


Step 6 - Pairwise plot

Pairwise scatter plot of all numerical input and output columns

In [28]:
```
import seaborn as sns
sns.set_style("whitegrid")
tips = sns.load_dataset("tips")
sns.pairplot(numericalData)
```

Out[28]: <seaborn.axisgrid.PairGrid at 0x120c393c8>



Step 7 - Cross-Correlation of Input Columns

Cross-correlation matrix for input columns using pearson correlation coefficient

In [29]:
```
numericalData.corr(method = 'pearson')
```

Out[29]:

	latitude	longitude	timezoneOffset	utcTimestamp
latitude	1.000000	0.999992	0.999991	0.999995
longitude	0.999992	1.000000	1.000000	0.999991
timezoneOffset	0.999991	1.000000	1.000000	0.999992
utcTimestamp	0.999995	0.999991	0.999992	1.000000

Step 8 - Identify Correlated Columns

Those input columns with pearson coefficient greater than or equal to 0.8

In [30]:
```
numericalData.corr(method = 'pearson') >= 0.8
```

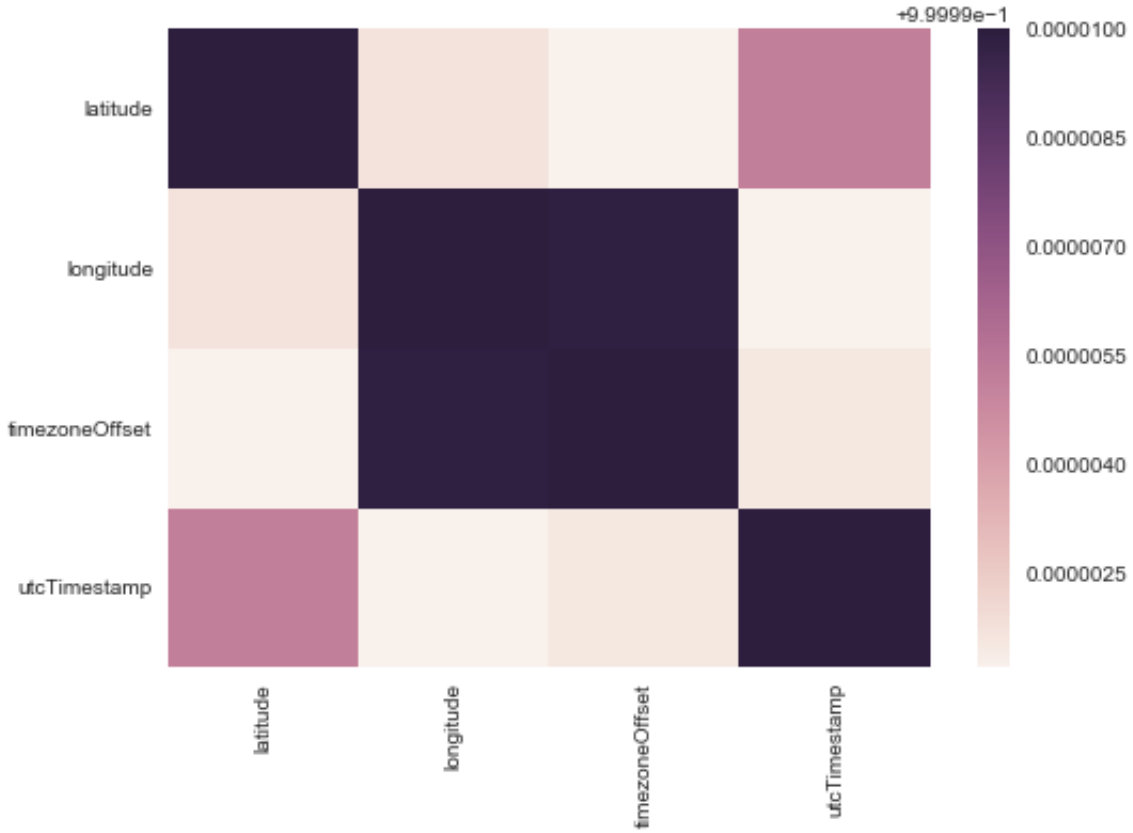
Out[30]:

	latitude	longitude	timezoneOffset	utcTimestamp
latitude	True	True	True	True
longitude	True	True	True	True
timezoneOffset	True	True	True	True
utcTimestamp	True	True	True	True

Step 9 - Cross-Correlation Heatmap

Heatmap plot for cross-correlation matrix of input column

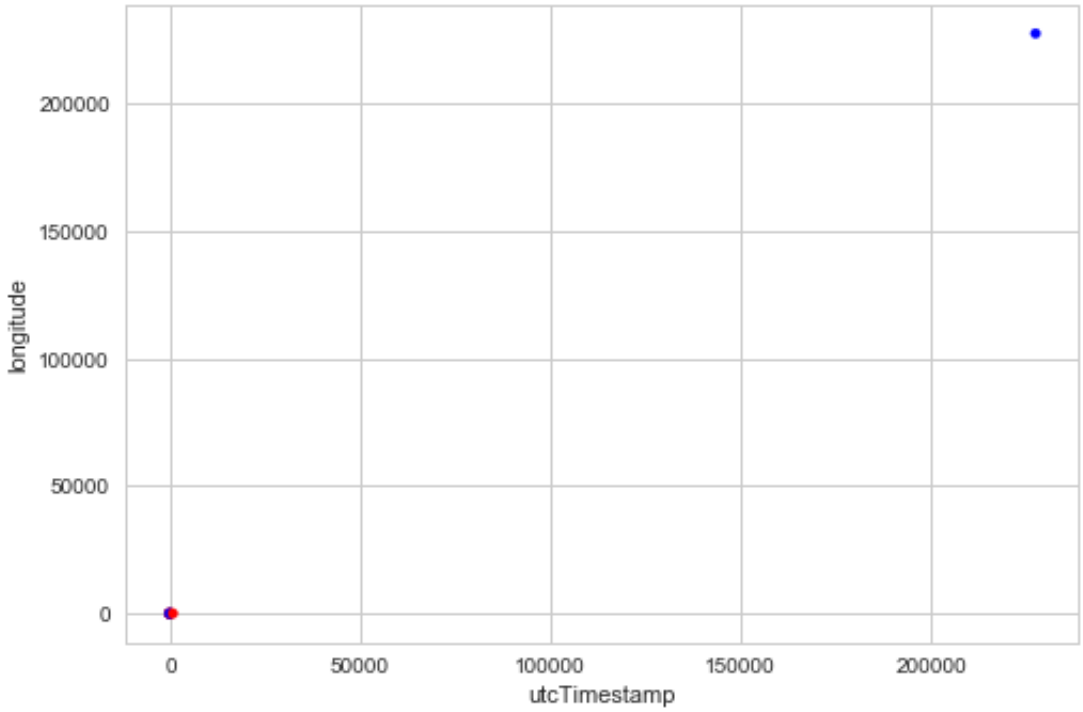
In [32]:
```
ax = sns.heatmap(numericalData.corr(method = 'pearson').T)

# turn the axis label
for item in ax.get_yticklabels():
    item.set_rotation(0)

for item in ax.get_xticklabels():
    item.set_rotation(90)

plt.show()
```



Step 10 - Output versus Input Plot

In [33]:
```
numericalData.plot.scatter(x='utcTimestamp', y='longitude', c=['b','r'])
```

Out[33]: <matplotlib.axes._subplots.AxesSubplot at 0x1227ea9b0>



In [ ]: