Data Quality and Statistics

Authors

Rakshit Sareen (rs5606)
Saurabh Mahajan (sm6921)
Sneha Ghosh (sg3533)

Abstract

Data cleaning plays an important role in data analysis applications. The goal for first part of the project is to clean the data obtained from NYC crime dataset. Data cleaning is done by finding missing values, incorrect values using type checks and other validations and filtering using reference dataset. We use these techniques to clean data for all columns and then merge the cleaned and corrected columns to obtain the final dataset. This dataset will be used in the second part of project for data analysis and data exploration.

Introduction

NYC is the most populated city in US. People from diverse background live here and it is crucial to analyse crimes and try to take actions for safety. Data analysis will allow us to visualize data trends by aggregating data using various attributes. We could analyze amount of crimes happening yearly, monthly, in different boroughs and also the types of crimes which happen.

Since the dataset consists of around 5 million rows, we have utilized the big data Hadoop framework for data analysis. Before data analysis, data cleaning is done to remove invalid data from the dataset so that our analysis does not get affected by invalid data. Some of the data was also corrected to reduce the incorrect data wherever possible. We used pyspark for data cleaning.

GitHub Repository -

https://github.com/snehaghosh91/BigDataProject

Results and discussion

- **Column 0**
  This column is a unique identifier for each row.

Script : col0.py
The property we checked for this column was basically that every value in the column is unique. One output file generated : **col1_statistics.out** which contains the count for the valid and invalid keys.
INVALID: 0
VALID : 5580035

- **Column 1 and Column 3 and Column 5**
  These two columns represent the From and To Date of the crime.
  Script : col1_3.py
  The script validates all the dates in the column 1,3 and 5 of the data.
  These were tricky columns as the combination of both to and from columns determine whether it is invalid data or not.
  I decided to mark it invalid only if both the values are empty.
  If one of the values is not empty, then we mark the dates as VALID. We follow the below semantic and mark the dates as EXACT, RANGE, ENDPOINT.
  EXACT : If only FROM Date is provided
  ENDPOINT : If only TO Date is provided
  RANGE : If both dates are provided.

  OUTPUT Files:
  Col3_valid_data.out : Valid To Dates
  Col3_invalid_data.out : Invalid To Dates
  Col1_invalid_data.out : Invalid From Dates
  Col1_valid_data.out : Valid From Dates
  Col1_3_valid_data.out : Corrected data
  exactDates.out : Columns which have only from date
  rangeDates.out : Columns which have both from and to date
  endPointDates.out : Columns which have only to date

- **Column 2 and Column 4**
  These two columns represent the from and to time of the crime represented by the row.
  Script : col2-4.py
  The script cleans the data related to time. It also corrects the data in which the time was incorrect, such as 24:00:00, as there is no such time. It has been changed to 00:00:00.
  The tagging for the data is INVALID and VALID. INVALID can be any reasons such as time wrongly recorded, no proper formatting of the time, empty string.

  OUTPUT Files:
  Col2_invalid_data.out : contains invalid from time data
  Col2_valid_data.out : contains valid from time data

Col2_corrected.out : corrected from time
Col4_invalid_data.out : contains invalid to time data
Col4_valid_data.out : contains valid to time data
Col4_corrected.out : corrected to time
Col2_statistics.out : statistics for INVALID and VALID from time
Col4_statistics.out : statistics for INVALID and VALID to time

- **Column 6 - Offense Classification Code**
  Script col6_7.py - Code checks if the classification code is not empty and is a three digit integer.
  3 output files are generated -
  1. **col6_invalid_data.out** - Contains all invalid values in the column.
     Result - There are no invalid values in this column.

  2. **col6_valid_data.out** - Contains all valid values in the column. CMPLNT_NUM - unique value to identify rows, KY_CD - column value
     Result -
     Command - head -5 col6_valid_data.out
     ```
     CMPLNT_NUM  KY_CD
     101109527     113
     153401121     101
     569369778     117
     968417082     344
     ```

  3. **col6_statistics.out** - Contains statistics about the column like invalid elements count, valid elements count and invalid value which occurs maximum number of times.
     Result -
     Command - head -5 col6_statistics.out
     ```
     INVALID COUNT:  0
     VALID COUNT:    5580035
     ```

- **Column 7 - Description of offense**
  Script col6_7.py - Code checks if the description of offense is empty. Checks if the description is valid for its key by finding the description for each key which has max count and comparing it to the description.
  4 output files are generated -
  1. **col7_invalid_data.out** - Contains all invalid values in the column. CMPLNT_NUM - unique value to identify rows, OFNS_DESC - column value, REASON - tells why value is invalid.
     Result -

Command - head -5 col7_invalid_data.out

```
CMPLNT_NUM   OFNS_DESC  REASON
932125924               EMPTY VALUE
327111538               EMPTY VALUE
651408610               EMPTY VALUE
737618153               EMPTY VALUE
```

2. **col7_valid_data.out -** Contains all valid values in the column. CMPLNT_NUM - unique value to identify rows, OFNS_DESC - column value
   Result -
   Command - head -5 col7_valid_data.out

```
CMPLNT_NUM   OFNS_DESC
710792599     ASSAULT 3 & RELATED OFFENSES
423573333     SEX CRIMES
302356516     ROBBERY
349433504     CRIMINAL MISCHIEF & RELATED OF
```

3. **col7_statistics.out -** Contains statistics about the column like invalid elements count, valid elements count and invalid value which occurs maximum number of times.
   Result -
   Command - head -5 col7_statistics.out

```
INVALID COUNT:  22593
VALID COUNT:    5557442
MAX INVALID OCCURRENCE          EMPTY VALUE     18892
```

4. **col7_corrected.out -** Obtained by merging the valid data with data obtained by correcting invalid data. CMPLNT_NUM - unique value to identify rows, OFNS_DESC - valid/corrected column value
   Result -
   Command - head -5 col7_corrected.out

```
CMPLNT_NUM   OFNS_DESC
710792599     ASSAULT 3 & RELATED OFFENSES
423573333     SEX CRIMES
354521636     GRAND LARCENY OF MOTOR VEHICLE
827898780     DANGEROUS DRUGS
```

- **Column 8 - Internal Classification Code**
  Script col8_9.py - Code checks if the classification code is not empty and is a three digit integer.
  3 output files are generated -
  1. **col8_invalid_data.out -** Contains all invalid values in the column. CMPLNT_NUM - unique value to identify rows, PD_CD - column value, REASON -

tells why value is invalid.
Result -
Command - head -5 col8_invalid_data.out
```
CMPLNT_NUM  PD_CD  REASON
153401121          EMPTY VALUE
940141475          EMPTY VALUE
586976434          EMPTY VALUE
388875685          EMPTY VALUE
```

2. **col8_valid_data.out** - Contains all valid values in the column. CMPLNT_NUM - unique value to identify rows, PD_CD - column value
   Result -
   Command - head -5 col8_valid_data.out
```
CMPLNT_NUM  PD_CD
101109527   729
569369778   503
968417082   101
641637920   101
```

3. **col8_statistics.out** - Contains statistics about the column like invalid elements count, valid elements count and invalid value which occurs maximum number of times.
   Result -
   Command - head -5 col8_statistics.out
```
INVALID COUNT:     4909
VALID COUNT: 5575126
MAX INVALID OCCURRENCE          EMPTY VALUE  4909
```

- **Column 9 - Description of internal classification**
  Script col8_9.py - Code checks if the description of internal classification is empty. Checks if the description is valid for its key by finding the description for each key which has max count and comparing it to the description.
  3 output files are generated -
  1. **col9_invalid_data.out** - Contains all invalid values in the column. CMPLNT_NUM - unique value to identify rows, PD_DESC - column value, REASON - tells why value is invalid.
     Result -
     Command - head -5 col9_invalid_data.out
```
CMPLNT_NUM  PD_DESC  REASON
735957494          EMPTY VALUE
566621021          EMPTY VALUE
843046086          EMPTY VALUE
872932532          EMPTY VALUE
```

2. **col9_valid_data.out** - Contains all valid values in the column. CMPLNT_NUM - unique value to identify rows, PD_DESC - column value
   Result -
   Command - head -5 col9_valid_data.out
   ```
   CMPLNT_NUM  PD_DESC
   423573333     SODOMY 1
   710792599     ASSAULT 3
   827898780     CONTROLLED SUBSTANCE, POSSESSI
   138050170     LARCENY,GRAND FROM BUILDING (NON-RESIDENCE) UNATTENDED
   ```

3. **col9_statistics.out** - Contains statistics about the column like invalid elements count, valid elements count and invalid value which occurs maximum number of times.
   Result -
   Command - head -5 col9_statistics.out
   ```
   INVALID COUNT:        4909
   VALID COUNT: 5575126
   MAX INVALID OCCURRENCE              EMPTY VALUE  4909
   ```

4. **col9_corrected.out** - Obtained by merging the valid data with data obtained by correcting invalid data. CMPLNT_NUM - unique value to identify rows, PD_DESC - valid/corrected column value
   Result -
   Command - head -5 col9_corrected.out
   ```
   CMPLNT_NUM  PD_DESC
   710792599     ASSAULT 3
   423573333     SODOMY 1
   827898780     CONTROLLED SUBSTANCE, POSSESSI
   349433504     CRIMINAL MISCHIEF,UNCLASSIFIED 4
   ```

- **Column 10 - Crime was successfully completed or attempted**
  Script col10.py - Code checks if the values in column are not empty and are valid values - ATTEMPTED , COMPLETED.
  3 output files are generated -
  1. **col10_invalid_data.out** - Contains all invalid values in the column. CMPLNT_NUM - unique value to identify rows, CRM_ATPT_CPTD_CD - column value, REASON - tells why value is invalid.
     Result -
     Command - head -5 col10_invalid_data.out
     ```
     CMPLNT_NUM  CRM_ATPT_CPTD_CD  REASON
     448788620              EMPTY VALUE
     363472497              EMPTY VALUE
     ```

```
559717358           EMPTY VALUE
181835873           EMPTY VALUE
```

2. **col10_valid_data.out** - Contains all valid values in the column. CMPLNT_NUM - unique value to identify rows, CRM_ATPT_CPTD_CD - column value.
   Result -
   Command - head -5 col10_valid_data.out
   ```
   CMPLNT_NUM  CRM_ATPT_CPTD_CD
   101109527   COMPLETED
   153401121   COMPLETED
   569369778   COMPLETED
   968417082   COMPLETED
   ```

3. **col10_statistics.out** - Contains statistics about the column like invalid elements count, valid elements count and invalid value which occurs maximum number of times.
   Result -
   Command - head -5 col10_statistics.out
   ```
   INVALID COUNT:      7
   VALID COUNT: 5580028
   MAX INVALID OCCURRENCE          EMPTY VALUE  7
   ```

- **Column 11 - Level of Offense**
  Script col11.py - Code checks if the values in column are not empty and are valid values - FELONY, MISDEMEANOR, VIOLATION.
  3 output files are generated -
  1. **col11_invalid_data.out** - Contains all invalid values in the column. CMPLNT_NUM - unique value to identify rows, LAW_CAT_CD - column value, REASON - tells why value is invalid.
     Result - There are no invalid values in this column.
  2. **col11_valid_data.out** - Contains all valid values in the column. CMPLNT_NUM - unique value to identify rows, LAW_CAT_CD - column value.
     Result -
     Command - head -5 col11_valid_data.out
     ```
     CMPLNT_NUM  LAW_CAT_CD
     101109527   FELONY
     153401121   FELONY
     569369778   FELONY
     968417082   MISDEMEANOR
     ```

  3. **col11_statistics.out** - Contains statistics about the column like invalid elements count, valid elements count and invalid value which occurs maximum

number of times.
Result -
Command - head -5 col11_statistics.out
```
INVALID COUNT:        0
VALID COUNT: 5580035
```

- **Column 12 - Jurisdiction responsible for incident.**
  Script col12.py - Code checks if the values in column are not empty and are not invalid values - OTHER (identified from manual analysis).
  3 output files are generated -
  1. **col12_invalid_data.out** - Contains all invalid values in the column. CMPLNT_NUM - unique value to identify rows, JURIS_DESC - shows column value, REASON - tells why value is invalid.
     Result -
     Command - head -5 col12_invalid_data.out
     ```
     CMPLNT_NUM   JURIS_DESC    REASON
     675941712    OTHER         INVALID
     277565054    OTHER         INVALID
     389301033    OTHER         INVALID
     514698127    OTHER         INVALID
     ```

  2. **col12_valid_data.out** - Contains all valid values in the column. CMPLNT_NUM - unique value to identify rows, JURIS_DESC - column value.
     Result -
     Command - head -5 col12_valid_data.out
     ```
     CMPLNT_NUM   JURIS_DESC
     101109527    N.Y. POLICE DEPT
     153401121    N.Y. POLICE DEPT
     569369778    N.Y. POLICE DEPT
     968417082    N.Y. POLICE DEPT
     ```

  3. **col12_statistics.out** - Contains statistics about the column like invalid elements count, valid elements count and invalid value which occurs maximum number of times.
     Result -
     Command - head -5 col12_statistics.out
     ```
     INVALID COUNT:        14964
     VALID COUNT: 5565071
     MAX INVALID OCCURRENCE    OTHER INVALID         14964
     ```

- **Column 13- Borough name**
  Script col13.py - Code checks if the values in column are not empty and are valid values - QUEENS, MANHATTAN, BRONX, STATEN ISLAND, BROOKLYN.

3 output files are generated -

1. **col13_invalid_data.out** - Contains all invalid values in the column. CMPLNT_NUM - unique value to identify rows, BORO_NM - shows column value, REASON - tells why value is invalid.
   Result -
   Command - head -5 col13_invalid_data.out
   ```
   CMPLNT_NUM  BORO_NM    REASON
   187370390              EMPTY VALUE
   284743219              EMPTY VALUE
   219649982              EMPTY VALUE
   699763801              EMPTY VALUE
   ```

2. **col13_valid_data.out** - Contains all valid values in the column. CMPLNT_NUM - unique value to identify rows, BORO_NM - column value.
   Result -
   Command - head -5 col13_valid_data.out
   ```
   CMPLNT_NUM  BORO_NM
   101109527   BRONX
   153401121   QUEENS
   569369778   MANHATTAN
   968417082   QUEENS
   ```

3. **col13_statistics.out** - Contains statistics about the column like invalid elements count, valid elements count and invalid value which occurs maximum number of times.
   Result -
   Command - head -5 col13_statistics.out
   ```
   INVALID COUNT:      463
   VALID COUNT: 5579572
   MAX INVALID OCCURRENCE          EMPTY VALUE  463
   ```

- **Column 14 - Precinct**
  Script col14.py - Code checks if the values in column are not empty, are integers and are valid values.
  Valid values for Precincts are obtained from this link -
  https://www1.nyc.gov/site/nypd/bureaus/patrol/precincts-landing.page
  Manual Analysis after data cleaning - Analysing the data reference three values were not integers - Midtown South Precinct, Midtown North Precinct and Central Park Precinct. These need to be converted into integers as the values in crime dataset were all integers. Before converting these values the data was wrongly marked as invalid so adding the mapping of String precincts to integers helped fix the issue. **Mapping String to integers**- Midtown South Precinct - 14, Midtown North Precinct - 18, Central Park Precinct - 22

3 output files are generated -

1. **col14_invalid_data.out** - Contains all invalid values in the column. CMPLNT_NUM - unique value to identify rows, ADDR_PCT_CD - shows column value, REASON - tells why value is invalid.
   Result -
   Command - head -5 col14_invalid_data.out
   ```
   CMPLNT_NUM   ADDR_PCT_CD    REASON
   594173303                EMPTY VALUE
   713215539                EMPTY VALUE
   602049379                EMPTY VALUE
   989678893                EMPTY VALUE
   ```

2. **col14_valid_data.out** - Contains all valid values in the column. CMPLNT_NUM - unique value to identify rows, ADDR_PCT_CD - column value.
   Result -
   Command - head -5 col14_valid_data.out
   ```
   CMPLNT_NUM   ADDR_PCT_CD
   101109527      44
   153401121      103
   569369778      28
   968417082      105
   ```

3. **col14_statistics.out** - Contains statistics about the column like invalid elements count, valid elements count and invalid value which occurs maximum number of times.
   Result -
   Command - head -5 col14_statistics.out
   ```
   INVALID COUNT:        390
   VALID COUNT: 5579645
   MAX INVALID OCCURRENCE              EMPTY VALUE  390
   ```

- **Column 15 - Specific location of occurrence**
  Script col15.py - Code checks if the values in column are not empty and are valid values - FRONT OF, INSIDE, OPPOSITE OF, REAR OF, OUTSIDE.
  3 output files are generated -
  1. **col15_invalid_data.out** - Contains all invalid values in the column. CMPLNT_NUM - unique value to identify rows, LOC_OF_OCCUR_DESC - shows column value, REASON - tells why value is invalid.
     Result -
     Command - head -5 col15_invalid_data.out
     ```
     CMPLNT_NUM   LOC_OF_OCCUR_DESC          REASON
     569369778                EMPTY VALUE
     ```

```
898496564           EMPTY VALUE
566081066           EMPTY VALUE
584555879           EMPTY VALUE
```

2. **col15_valid_data.out -** Contains all valid values in the column. CMPLNT_NUM - unique value to identify rows, LOC_OF_OCCUR_DESC - column value.
   Result -
   Command - head -5 col15_valid_data.out
   ```
   CMPLNT_NUM  LOC_OF_OCCUR_DESC
   101109527   INSIDE
   153401121   OUTSIDE
   968417082   INSIDE
   641637920   FRONT OF
   ```

3. **col15_statistics.out -** Contains statistics about the column like invalid elements count, valid elements count and invalid value which occurs maximum number of times.
   Result -
   Command - head -5 col15_statistics.out
   ```
   INVALID COUNT:      1223605
   VALID COUNT: 4356430
   MAX INVALID OCCURRENCE          EMPTY VALUE  1223392
   ```

- **Column 16 - Specific description of premises**
  Script col16.py -Code checks if the values in column are not empty and are not invalid values - OTHER (identified from manual analysis).
  3 output files are generated -
  1. **col16_invalid_data.out -** Contains all invalid values in the column. CMPLNT_NUM - unique value to identify rows, PREM_TYP_DESC - shows column value, REASON - tells why value is invalid.
     Result -
     Command - head -5 col16_invalid_data.out
     ```
     CMPLNT_NUM  PREM_TYP_DESC  REASON
     153401121                  EMPTY VALUE
     569369778   OTHER          INVALID
     641637920   OTHER          INVALID
     340513307   OTHER          INVALID
     ```

  2. **col16_valid_data.out -** Contains all valid values in the column. CMPLNT_NUM - unique value to identify rows, PREM_TYP_DESC - column value.
     Result -
     Command - head -5 col16_valid_data.out
     ```
     CMPLNT_NUM  PREM_TYP_DESC
     ```

```
101109527    BAR/NIGHT CLUB
968417082    RESIDENCE-HOUSE
365661343    DRUG STORE
608231454    STREET
```

3. **col16_statistics.out** - Contains statistics about the column like invalid elements count, valid elements count and invalid value which occurs maximum number of times.
   Result -
   Command - head -5 col16_statistics.out
   ```
   INVALID COUNT:      183608
   VALID COUNT: 5396427
   MAX INVALID OCCURRENCE    OTHER        INVALID       148410
   ```

- **Column 17 - Name of NYC park**
  Script col17.py - Code checks if the values in column are not empty.
  3 output files are generated -
  1. **col17_invalid_data.out** - Contains all invalid values in the column. CMPLNT_NUM - unique value to identify rows, PARKS_NM - shows column value, REASON - tells why value is invalid.
     Result -
     Command - head -5 col17_invalid_data.out
     ```
     CMPLNT_NUM  PARKS_NM  REASON
     101109527            EMPTY VALUE
     153401121            EMPTY VALUE
     569369778            EMPTY VALUE
     968417082            EMPTY VALUE
     ```

  2. **col17_valid_data.out** - Contains all valid values in the column. CMPLNT_NUM - unique value to identify rows, PARKS_NM - column value.
     Result -
     Command - head -5 col17_valid_data.out
     ```
     CMPLNT_NUM  PARKS_NM
     590638275    MADISON SQUARE PARK
     557672328    COLUMBUS PARK AT MANHATTAN
     253843712    ARCILLA PLAYGROUND
     189160748    CENTRAL PARK
     ```

  3. **col17_statistics.out** - Contains statistics about the column like invalid elements count, valid elements count and invalid value which occurs maximum number of times.
     Result -
     Command - head -5 col17_statistics.out

```
INVALID COUNT:	5567497
VALID COUNT: 12538
MAX INVALID OCCURRENCE		EMPTY VALUE	5567497
```

- **Column 18 - Name of NYCHA housing development**
  Script col18.py - Code checks if the values in column are not empty, and are valid values.
  Valid values for **Housing development** are obtained from this link - http://www1.nyc.gov/site/nycha/about/developments.page (Heading - Development Maps)
  3 output files are generated -
  1. **col18_invalid_data.out** - Contains all invalid values in the column. CMPLNT_NUM - unique value to identify rows, HADEVELOPT - shows column value, REASON - tells why value is invalid.
     Result -
     Command - head -5 col18_invalid_data.out
     ```
     CMPLNT_NUM  HADEVELOPT  REASON
     101109527              EMPTY VALUE
     153401121              EMPTY VALUE
     569369778              EMPTY VALUE
     968417082              EMPTY VALUE
     ```

  2. **col18_valid_data.out** - Contains all valid values in the column. CMPLNT_NUM - unique value to identify rows, HADEVELOPT - column value.
     Result -
     Command - head -5 col18_valid_data.out
     ```
     CMPLNT_NUM  HADEVELOPT
     251546004   MARCY
     824663386   MORRIS I
     978579954   FARRAGUT
     609719707   BORINQUEN PLAZA I
     ```

  3. **col18_statistics.out** - Contains statistics about the column like invalid elements count, valid elements count and invalid value which occurs maximum number of times.
     Result -
     Command - head -5 col18_statistics.out
     ```
     INVALID COUNT:	5359618
     VALID COUNT: 220417
     MAX INVALID OCCURRENCE		EMPTY VALUE	5302218
     ```

- **Column 19 - X-Coordinate of the location of crime**

Script col19-22.py - Code checks if the values in column are not empty and integer values with the New York City limits.
3 output files are generated -

1. **col19_invalid_data.out** - Contains all invalid values in the column. CMPLNT_NUM - unique value to identify rows, X_COORD_CD - shows column value.
2. **col19_valid_data.out** - Contains all valid values in the column. CMPLNT_NUM - unique value to identify rows, X_COORD_CD - column value.
3. **col19_statistics.out** - Contains statistics about the column like invalid elements count, valid elements count and invalid value which occurs maximum number of times.

- **Column 20 - Y-Coordinate of the location of crime**
  Script col19-22.py - Code checks if the values in column are not empty and integer values with the New York City limits.
  3 output files are generated -
  2. **col20_invalid_data.out** - Contains all invalid values in the column. CMPLNT_NUM - unique value to identify rows, Y_COORD_CD - shows column value.
  2. **col20_valid_data.out** - Contains all valid values in the column. CMPLNT_NUM - unique value to identify rows, Y_COORD_CD - column value.
  3. **col20_statistics.out** - Contains statistics about the column like invalid elements count, valid elements count and invalid value which occurs maximum number of times.

- **Column 21 - Latitude of the location of crime**
  Script col19-22.py - Code checks if the values in column are not empty and decimal values with the New York City limits.
  3 output files are generated -
  3. **col21_invalid_data.out** - Contains all invalid values in the column. CMPLNT_NUM - unique value to identify rows, Latitude - shows column value.
  2. **col21_valid_data.out** - Contains all valid values in the column. CMPLNT_NUM - unique value to identify rows, Latitude - column value.
  3. **col21_statistics.out** - Contains statistics about the column like invalid elements count, valid elements count and invalid value which occurs maximum number of times.

- **Column 22 - Longitude of the location of crime**
  Script col19-22.py - Code checks if the values in column are not empty and decimal values with the New York City limits.
  3 output files are generated -
  4. **col22_invalid_data.out** - Contains all invalid values in the column. CMPLNT_NUM - unique value to identify rows, Longitude - shows column value.

2. **col22_valid_data.out** - Contains all valid values in the column. CMPLNT_NUM - unique value to identify rows, Longitude - column value.
3. **col22_statistics.out** - Contains statistics about the column like invalid elements count, valid elements count and invalid value which occurs maximum number of times.

- **Column 23 - Latitude, Longitude of the location of crime**
  Script col23.py - Code checks if the values in column are not empty and decimal values match the values that are printed the column 21 and 22.
  3 output files are generated -
  5. **col23_invalid_data.out -** Contains all invalid values in the column. CMPLNT_NUM - unique value to identify rows, Lat_Lon - shows column value.
  2. **col23_valid_data.out** - Contains all valid values in the column. CMPLNT_NUM - unique value to identify rows, Lat_Lon - column value.
  3. **col23_statistics.out** - Contains statistics about the column like invalid elements count, valid elements count and invalid value which occurs maximum number of times.

Individual Contributions

Rakshit - Worked on columns 0 to 5 for data cleaning and data correction. The data contained from and to dates and time of the crime.

- Observation was that some entries had wrong data in terms of time and some was empty. I have corrected the data and produced relevant data files.
- The tricky part of this data was that To date of the crime had many null/empty values, hence we cannot just throw that tuple because it is not present.
- We categorised the dates into 4 major categories, INVALID, EXACT, RANGE and ENDPOINT. Everything apart from INVALID is VALID data and is used to produce valid/corrected data.
- Throwing away any empty value did not make sense because that would have reduced data, hence I performed cross column validations.
- All the data is categorized and tagged in separate files which are mentioned above in the column details.
- Regular expression were used to validate the formatting of data and has been corrected wherever possible.
- The dates and time we also checked for correct semantic values and range.
- The output files contain detailed summary for statistics for the data and tagging.
- The merging is possible because the unique identifier is maintained in the data set of the valid and the invalid files. This unique identifier is used to merge the data at a later stage.
- Scripts were not written for individual columns but rather based on semantics, such as cleaning for dates data went into single script and for time, a different single script.

- I go by the name buzzLY in the github repo. Also, initial commits do not contain my username but the name is still visible.

Saurabh - Worked on columns 19 to 23 for data cleaning and data correction.

- ➔ The columns contained data for the locations of the crime. All the rows with empty columns were marked invalid while the other values were checked if the were valid locations within the city.
- ➔ Merged the valid and corrected columns to obtain final cleaned dataset.

Sneha - **Worked on columns 6 to 18 for data cleaning and data correction**.

- ➔ Manual analysis of data to look for invalid values. Eg - “Other” value was present in column 12 and column 16. This was identified by manually analyzing the distinct values for every column and identifying the invalid values.
- ➔ Cross column validations done for key and description(columns 6, 7 and columns 8, 9). Idea used - For a particular key, calculate the count of each description and find the description with max count. The other description values for the key are considered invalid. For data correction, the other description values are replaced with the description having max count.
- ➔ Created common code in helper.py which is used for validating all columns 6 to 18.
- ➔ Research for Reference data - Created reference data file for Precinct[2] and Housing Development[3]. Modified reference dataset for “Precinct” by mapping String values to integers to prevent incorrect validation of precinct data. Used this dataset to validate those column values.
- ➔ Data type validation was performed for integer values. Values which were enum are also validated using the list of valid values. This list is obtained by manual analysis.
- ➔ For every column separate files are generated - valid, invalid, statistics, corrected. Corrected file is obtained by correcting values in the column whenever possible. Statistics file contains the summary about count of valid data, invalid data and invalid data which has max frequency.

Summary

- Among the invalid data, count of empty values was maximum.
- There were instances where same key had different descriptions. This was fixed by replacing such descriptions with the description having the highest count for a particular key.
- For the column “Housing Development”[3] and “Precinct”[2], some invalid values were obtained by using a different data set as reference.
- By manual data analysis, some invalid values (like “Other”) were obtained for some columns.
- The original NYPD crime file 1396 MB in size is reduced to 903MB file after Data Cleaning.

References

1. https://data.cityofnewyork.us/Public-Safety/NYPD-Complaint-Data-Historic/qgea-i56i
2. https://www1.nyc.gov/site/nypd/bureaus/patrol/precincts-landing.page
3. http://www1.nyc.gov/site/nycha/about/developments.page