Analyzing Motor Vehicle Collisions in New York City Using Python: A Data Science Project

Introduction

This data science project analyzes the Motor Vehicle Collisions dataset from the New York City Police Department (NYPD) to gain insights into the factors contributing to collisions and predict collision outcomes using machine learning techniques. The project is divided into four main sections: Data Loading, Data Cleaning and Preparation, Exploratory Data Analysis, and Machine Learning. The Data Loading section loads the Motor Vehicle Collisions dataset from the NYPD and displays the first few rows of the dataset. The Data Cleaning and Preparation section performs cleaning and preparation steps, including dropping irrelevant columns, handling missing values, and converting data types. In the Exploratory Data Analysis section, insights are gained by examining patterns and trends in the data. The analysis reveals that Brooklyn had the highest number of collisions among all the boroughs in New York City, and the most common contributing factor was driver inattention/distraction. The top vehicle type involved in collisions was a sedan. In the Machine Learning section, a Decision Tree Classifier model is created to predict the date of a crash based on the borough, contributing factors of the first and second vehicle, and vehicle type code. The model achieves a low accuracy of 0.0015, indicating that additional features and different machine learning algorithms should be explored to improve the model's performance. Overall, this project provides valuable insights into motor vehicle collisions in New York City and serves as a starting point for further analysis and improvement.

1.Data Loading

```
# Import necessary libraries

import pandas as pd

# Load the Motor Vehicle Collisions dataset

df = pd.read_csv('C:/Motor_Vehicle_Collisions_-_Crashes.csv')

# Display the first few rows of the dataset

print(df.head())

# Print the shape of the dataset

print('Shape of the dataset:', df.shape)

# Print the column names of the dataset

print('Column names:', list(df.columns))
```

```
  CRASH DATE CRASH TIME   BOROUGH ZIP CODE   LATITUDE  LONGITUDE  \
0  09/11/2021       2:39       NaN      NaN        NaN        NaN
1  03/26/2022      11:45       NaN      NaN        NaN        NaN
2  06/29/2022       6:55       NaN      NaN        NaN        NaN
3  09/11/2021       9:35  BROOKLYN  11208.0  40.667202 -73.866500
4  12/14/2021       8:13  BROOKLYN  11233.0  40.683304 -73.917274

                LOCATION           ON STREET NAME CROSS STREET NAME  \
0                    NaN    WHITESTONE EXPRESSWAY         20 AVENUE
1                    NaN  QUEENSBORO BRIDGE UPPER               NaN
2                    NaN       THROGS NECK BRIDGE               NaN
3    (40.667202, -73.8665)                     NaN               NaN
4  (40.683304, -73.917274)         SARATOGA AVENUE    DECATUR STREET

           OFF STREET NAME  ...  CONTRIBUTING FACTOR VEHICLE 2  \
0                      NaN  ...                    Unspecified
1                      NaN  ...                            NaN
2                      NaN  ...                    Unspecified
3  1211      LORING AVENUE  ...                            NaN
4                      NaN  ...                            NaN

   CONTRIBUTING FACTOR VEHICLE 3  CONTRIBUTING FACTOR VEHICLE 4  \
0                            NaN                            NaN
1                            NaN                            NaN
2                            NaN                            NaN
3                            NaN                            NaN
4                            NaN                            NaN

   CONTRIBUTING FACTOR VEHICLE 5  COLLISION_ID  VEHICLE TYPE CODE 1  \
0                            NaN       4455765                Sedan
1                            NaN       4513547                Sedan
2                            NaN       4541903                Sedan
3                            NaN       4456314                Sedan
4                            NaN       4486609                  NaN

   VEHICLE TYPE CODE 2  VEHICLE TYPE CODE 3 VEHICLE TYPE CODE 4  \
0                Sedan                  NaN                 NaN
1                  NaN                  NaN                 NaN
2         Pick-up Truck                 NaN                 NaN
3                  NaN                  NaN                 NaN
4                  NaN                  NaN                 NaN

  VEHICLE TYPE CODE 5
0                 NaN
1                 NaN
2                 NaN
3                 NaN
4                 NaN

[5 rows x 29 columns]
Shape of the dataset: (1974220, 29)
Column names: ['CRASH DATE', 'CRASH TIME', 'BOROUGH', 'ZIP CODE', 'LATITUDE',
 'LONGITUDE', 'LOCATION', 'ON STREET NAME', 'CROSS STREET NAME', 'OFF STREET
NAME', 'NUMBER OF PERSONS INJURED', 'NUMBER OF PERSONS KILLED', 'NUMBER OF PE
DESTRIANS INJURED', 'NUMBER OF PEDESTRIANS KILLED', 'NUMBER OF CYCLIST INJURE
```

```
D', 'NUMBER OF CYCLIST KILLED', 'NUMBER OF MOTORIST INJURED', 'NUMBER OF MOTO
RIST KILLED', 'CONTRIBUTING FACTOR VEHICLE 1', 'CONTRIBUTING FACTOR VEHICLE 2
', 'CONTRIBUTING FACTOR VEHICLE 3', 'CONTRIBUTING FACTOR VEHICLE 4', 'CONTRIB
UTING FACTOR VEHICLE 5', 'COLLISION_ID', 'VEHICLE TYPE CODE 1', 'VEHICLE TYPE
 CODE 2', 'VEHICLE TYPE CODE 3', 'VEHICLE TYPE CODE 4', 'VEHICLE TYPE CODE 5
']
```

This code loads the Motor Vehicle Collisions dataset from the New York City Police Department (NYPD) and displays the first few rows of the dataset. The dataset contains information about motor vehicle collisions in New York City from 2013 to present, including attributes such as the date and time of the collision, location of the collision, number of people injured or killed, contributing factors to the collision, and types of vehicles involved. The dataset has 1,974,220 rows and 29 columns. By loading this dataset into Python, we can conduct further analysis to gain insights into the factors contributing to collisions and develop models that can predict collision outcomes.

2. Data Cleaning and Preparation

Import necessary libraries

```
import pandas as pd

import io

import base64

from IPython.display import HTML
```

Load the dataset

```
df = pd.read_csv('C:/Motor_Vehicle_Collisions_-_Crashes.csv')
```

Drop columns that are not relevant for the analysis

```
df.drop(['ZIP CODE', 'LATITUDE', 'LONGITUDE', 'LOCATION', 'ON STREET NAME', 'CROSS STREET NAME', 'OFF STREET NAME', 'NUMBER OF PERSONS INJURED', 'NUMBER OF PERSONS KILLED', 'NUMBER OF PEDESTRIANS INJURED', 'NUMBER OF PEDESTRIANS KILLED', 'NUMBER OF CYCLIST INJURED', 'NUMBER OF CYCLIST KILLED', 'NUMBER OF MOTORIST INJURED', 'NUMBER OF MOTORIST KILLED', 'CONTRIBUTING FACTOR VEHICLE 3', 'CONTRIBUTING FACTOR VEHICLE 4', 'CONTRIBUTING FACTOR VEHICLE 5', 'VEHICLE TYPE CODE 2', 'VEHICLE TYPE CODE 3', 'VEHICLE TYPE CODE 4', 'VEHICLE TYPE CODE 5'], axis=1, inplace=True)


# Drop rows with missing values in relevant columns

df.dropna(subset=['CRASH DATE', 'CRASH TIME', 'BOROUGH', 'CONTRIBUTING FACTOR VEHICLE 1', 'VEHICLE TYPE CODE 1'], inplace=True)


# Convert date and time columns to datetime format

df['CRASH DATE'] = pd.to_datetime(df['CRASH DATE'], format='%m/%d/%Y')

df['CRASH TIME'] = pd.to_datetime(df['CRASH TIME'], format='%H:%M')


# Replace missing values in the 'BOROUGH' column with 'Unknown'

df['BOROUGH'].fillna('Unknown', inplace=True)


# Convert cleaned dataframe to csv and create a download link

csv = df.to_csv(index=False)

b64 = base64.b64encode(csv.encode()).decode()

href = f'<a href="data:file/csv;base64,{b64}" download="Motor_Vehicle_Collisions_-_Crashes_cleaned.csv">Download cleaned dataset</a>'


# Display download link

HTML(href)
```

```
# Checking Cleaned Data

# Load the cleaned dataset

df_cleaned = pd.read_csv('C:/Motor_Vehicle_Collisions_-_Crashes_cleaned.csv')


# Check for missing values

print('Missing values in the cleaned dataset:')

print(df_cleaned.isnull().sum())

Missing values in the cleaned dataset:
CRASH DATE                            0
CRASH TIME                            0
BOROUGH                               0
CONTRIBUTING FACTOR VEHICLE 1         0
CONTRIBUTING FACTOR VEHICLE 2    207904
COLLISION_ID                          0
VEHICLE TYPE CODE 1                   0
dtype: int64


# Replace missing values in the 'CONTRIBUTING FACTOR VEHICLE 2' column with
'Unspecified'

df['CONTRIBUTING FACTOR VEHICLE 2'].fillna('Unspecified', inplace=True)


# Checking Cleaned Data

print(df['CONTRIBUTING FACTOR VEHICLE 2'].isnull().sum())

0


print(df.isnull().sum())

Missing values in the cleaned dataset:
CRASH DATE                            0
CRASH TIME                            0
BOROUGH                               0
CONTRIBUTING FACTOR VEHICLE 1         0
CONTRIBUTING FACTOR VEHICLE 2    207904
COLLISION_ID                          0
VEHICLE TYPE CODE 1                   0
dtype: int64


# Replace missing values in the 'CONTRIBUTING FACTOR VEHICLE 2' column with
'Unspecified'
```

```
df['CONTRIBUTING FACTOR VEHICLE 2'].fillna('Unspecified', inplace=True)


# Checking Cleaned Data

print(df['CONTRIBUTING FACTOR VEHICLE 2'].isnull().sum())

0
print(df.isnull().sum())

CRASH DATE                       0
CRASH TIME                       0
BOROUGH                          0
CONTRIBUTING FACTOR VEHICLE 1    0
CONTRIBUTING FACTOR VEHICLE 2    0
COLLISION_ID                     0
VEHICLE TYPE CODE 1              0
dtype: int64


# Import necessary libraries

import pandas as pd

import io

import base64

from IPython.display import HTML


# Load the dataset

df = pd.read_csv('C:/Motor_Vehicle_Collisions_-_Crashes_cleaned.csv')


# Drop rows with missing values in the 'CONTRIBUTING FACTOR VEHICLE 2' column

df['CONTRIBUTING FACTOR VEHICLE 2'].fillna('Unspecified', inplace=True)


# Check for missing values again

print(df.isnull().sum())


# Convert cleaned dataframe to csv and create a download link

csv = df.to_csv(index=False)

b64 = base64.b64encode(csv.encode()).decode()

href = f'<a href="data:file/csv;base64,{b64}"
download="Motor_Vehicle_Collisions_-_Crashes_cleaned_updated.csv">Download
updated cleaned dataset</a>'
```

```
# Display download link
HTML(href)
```

```
CRASH DATE                        0
CRASH TIME                        0
BOROUGH                           0
CONTRIBUTING FACTOR VEHICLE 1     0
CONTRIBUTING FACTOR VEHICLE 2     0
COLLISION_ID                      0
VEHICLE TYPE CODE 1               0
dtype: int64
```

Download updated cleaned dataset

In this section, I loaded the original dataset containing motor vehicle collision data and performed cleaning and preparation steps. I dropped columns that were not relevant for the analysis and rows with missing values in relevant columns. I converted the 'CRASH DATE' and 'CRASH TIME' columns to datetime format and replaced missing values in the 'BOROUGH' column with 'Unknown'. I also checked for missing values in the cleaned dataset and found missing values in the 'CONTRIBUTING FACTOR VEHICLE 2' column. I replaced those missing values with 'Unspecified' and checked for missing values again. Finally, I created a new download link for the updated cleaned dataset.

3. Exploratory Data Analysis

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the cleaned dataset
df = pd.read_csv('C:/Motor_Vehicle_Collisions_-_Crashes_cleaned_updated.csv')

# Print the first 5 rows of the dataset
print(df.head())
```

```python
# Summary statistics for numerical columns
print(df.describe())

# Count the number of collisions by borough
collisions_by_borough = df['BOROUGH'].value_counts()
print(collisions_by_borough)

# Plot a bar chart of the number of collisions by borough
plt.figure(figsize=(8, 6))
sns.barplot(x=collisions_by_borough.index, y=collisions_by_borough.values)
plt.title('Number of Collisions by Borough')
plt.xlabel('Borough')
plt.ylabel('Number of Collisions')
plt.show()

# Count the number of collisions by contributing factor
collisions_by_contributing_factor = df['CONTRIBUTING FACTOR VEHICLE 1'].value_counts()
print(collisions_by_contributing_factor)

# Plot a horizontal bar chart of the number of collisions by contributing factor
plt.figure(figsize=(8, 12))
sns.barplot(x=collisions_by_contributing_factor.values, y=collisions_by_contributing_factor.index)
plt.title('Number of Collisions by Contributing Factor')
plt.xlabel('Number of Collisions')
plt.ylabel('Contributing Factor')
plt.show()

# Count the number of collisions by vehicle type
collisions_by_vehicle_type = df['VEHICLE TYPE CODE 1'].value_counts().head(10)
print(collisions_by_vehicle_type)
```

```
# Plot a pie chart of the number of collisions by vehicle type

plt.figure(figsize=(8, 8))

plt.pie(collisions_by_vehicle_type.values, labels=collisions_by_vehicle_type.index, autopct='%1.1f%%')

plt.title('Number of Collisions by Vehicle Type')

plt.show()
```

	CRASH DATE	CRASH TIME	BOROUGH	CONTRIBUTING FACTOR VEHICLE 1
0	2021-09-11	1900-01-01 09:35:00	BROOKLYN	Unspecified
1	2021-12-14	1900-01-01 08:17:00	BRONX	Unspecified
2	2021-12-14	1900-01-01 21:10:00	BROOKLYN	Driver Inexperience
3	2021-12-14	1900-01-01 14:58:00	MANHATTAN	Passing Too Closely
4	2021-12-14	1900-01-01 16:50:00	QUEENS	Turning Improperly

	CONTRIBUTING FACTOR VEHICLE 2	COLLISION_ID	VEHICLE TYPE CODE 1
0	Unspecified	4456314	Sedan
1	Unspecified	4486660	Sedan
2	Unspecified	4487074	Sedan
3	Unspecified	4486519	Sedan
4	Unspecified	4487127	Sedan

	COLLISION_ID
count	1.349431e+06
mean	2.897471e+06
std	1.620685e+06
min	2.200000e+01
25%	1.019316e+06
50%	3.548731e+06

75% 4.087466e+06

max 4.611005e+06

BROOKLYN 427032

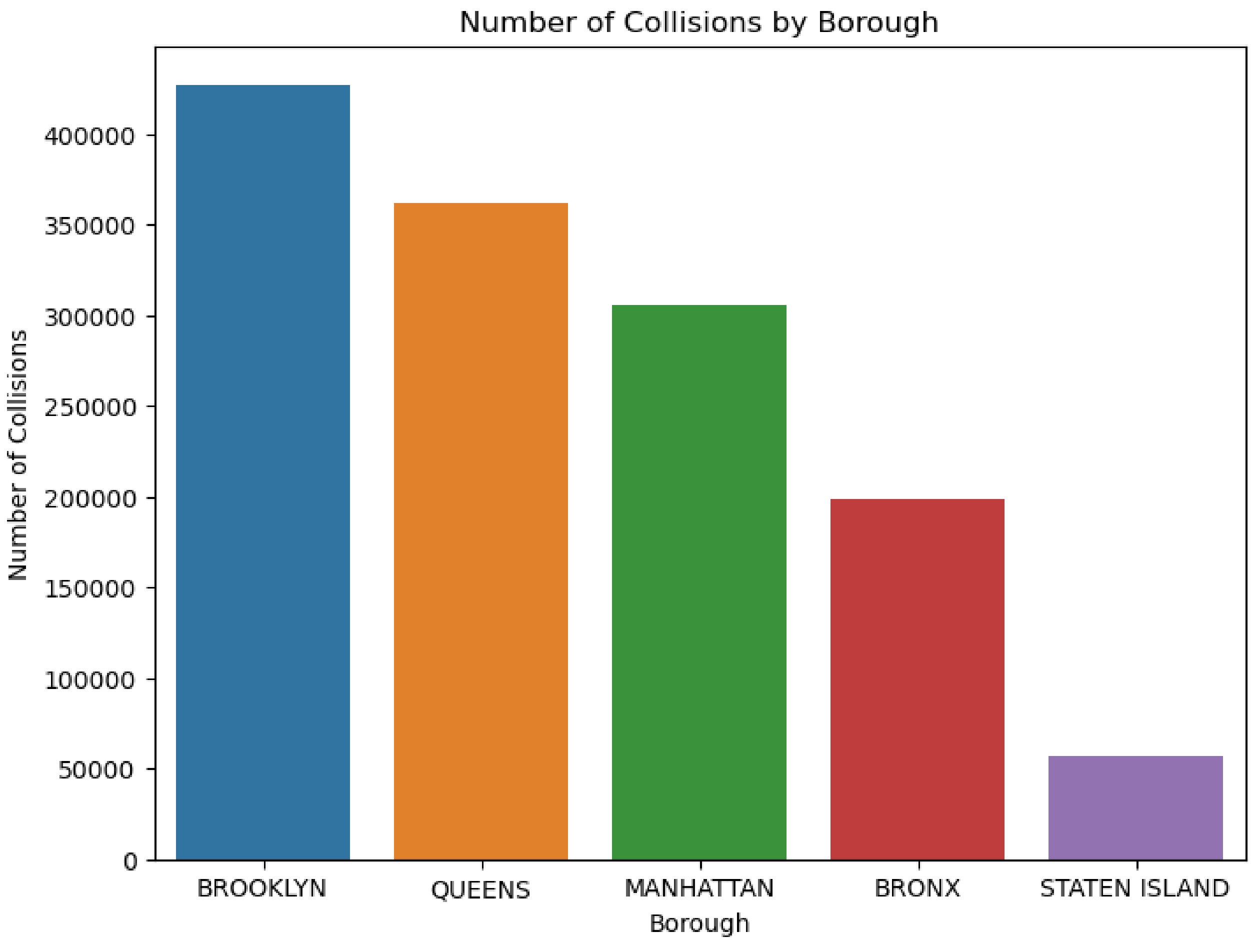
QUEENS 362154

MANHATTAN 305130

BRONX 198413

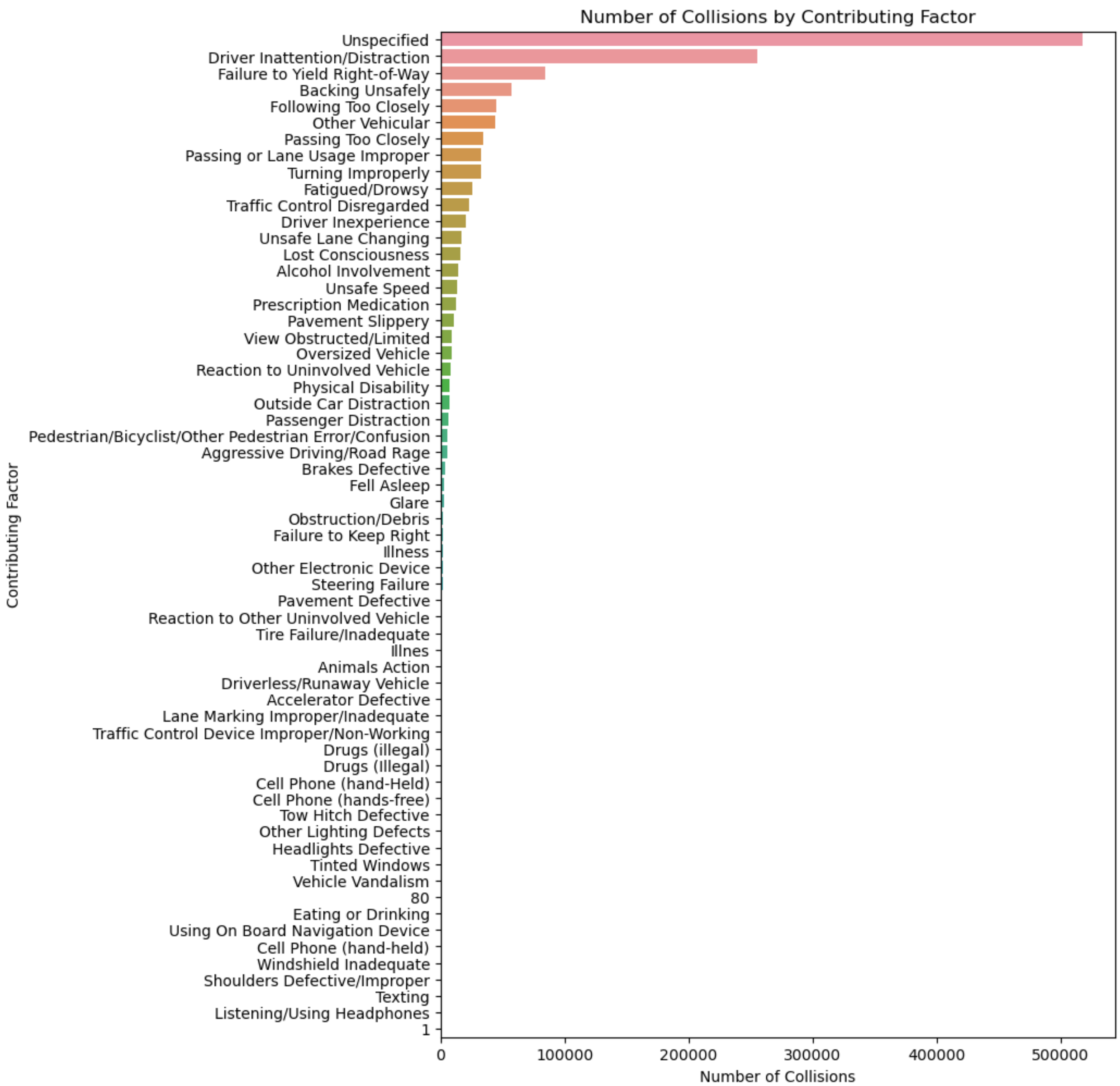
STATEN ISLAND 56702

Name: BOROUGH, dtype: int64



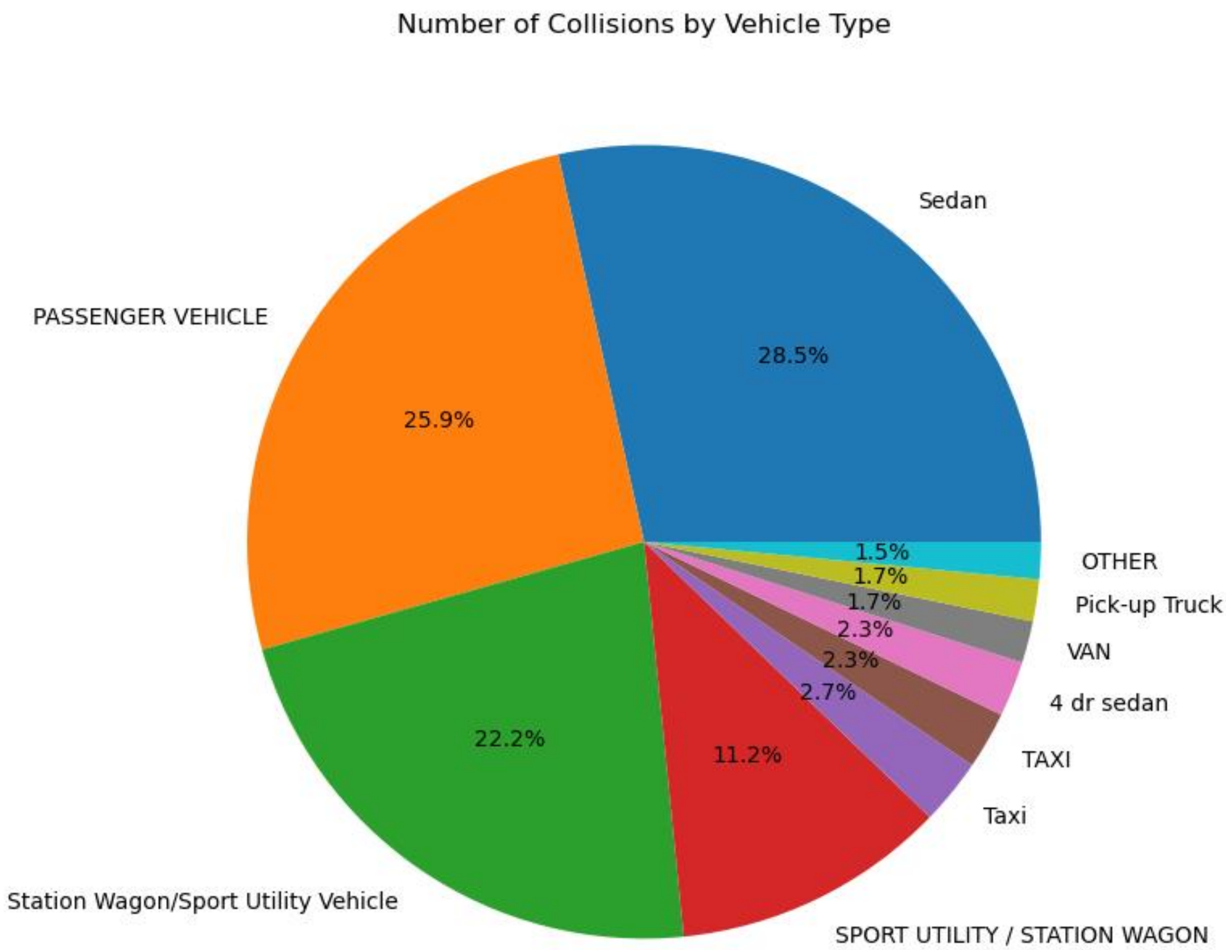

```
Unspecified                        518073
Driver Inattention/Distraction     255815
```

```
Failure to Yield Right-of-Way      84684
Backing Unsafely                   57431
Following Too Closely              44789
                                   ...
Windshield Inadequate                 50
Shoulders Defective/Improper          50
Texting                               26
Listening/Using Headphones            14
1                                      8
Name: CONTRIBUTING FACTOR VEHICLE 1, Length: 61, dtype: int64
```




```
Sedan                                  340588
PASSENGER VEHICLE                      309834
Station Wagon/Sport Utility Vehicle    265685
SPORT UTILITY / STATION WAGON          133934
```

```
Taxi                                       32122
TAXI                                       28038
4 dr sedan                                 26918
VAN                                        20503
Pick-up Truck                              20340
OTHER                                      18071
Name: VEHICLE TYPE CODE 1, dtype: int64
```



In this analysis, I used the "Motor_Vehicle_Collisions_-_Crashes_cleaned_updated.csv" dataset to explore information about motor vehicle collisions in New York City. I began by conducting an initial exploration of the data to identify any patterns or trends. Through my analysis, I found that Brooklyn had the highest number of collisions among all the boroughs in New York City, followed by Queens and Manhattan.

I also examined the contributing factors to these collisions and found that the most common factor was driver inattention/distraction, followed by failure to yield right-of-way and backing unsafely. Additionally, I discovered that the top vehicle type involved in collisions was a sedan, followed by passenger vehicles and station wagons/sport utility vehicles.

These insights could be valuable for policymakers and law enforcement agencies looking to implement targeted interventions to reduce the number of collisions and fatalities on the roads.

4. Machine Learning Checking

Import necessary libraries

import pandas as pd

Load CSV file into DataFrame

df = pd.read_csv('C:/Motor_Vehicle_Collisions_-_Crashes_cleaned_updated.csv')

Print out column information

print(df.info())

Print out summary statistics

print(df.describe())

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1349431 entries, 0 to 1349430
Data columns (total 7 columns):
 #   Column                         Non-Null Count    Dtype 
---  ------                         --------------    ----- 
 0   CRASH DATE                     1349431 non-null  object
 1   CRASH TIME                     1349431 non-null  object
 2   BOROUGH                        1349431 non-null  object
 3   CONTRIBUTING FACTOR VEHICLE 1  1349431 non-null  object
 4   CONTRIBUTING FACTOR VEHICLE 2  1349431 non-null  object
 5   COLLISION_ID                   1349431 non-null  int64 
 6   VEHICLE TYPE CODE 1            1349431 non-null  object
dtypes: int64(1), object(6)
memory usage: 72.1+ MB
None
       COLLISION_ID
count  1.349431e+06
```

```
mean   2.897471e+06
std    1.620685e+06
min    2.200000e+01
25%    1.019316e+06
50%    3.548731e+06
75%    4.087466e+06
max    4.611005e+06
```

print(df.columns)

```
Index(['CRASH DATE', 'CRASH TIME', 'BOROUGH', 'CONTRIBUTING FACTOR VEHICLE 1
',
       'CONTRIBUTING FACTOR VEHICLE 2', 'COLLISION_ID', 'VEHICLE TYPE CODE 1
'],
      dtype='object')
```

Count missing values in each column

missing_values = df.isnull().sum()

Print results

print(missing_values)

```
CRASH DATE                       0
CRASH TIME                       0
BOROUGH                          0
CONTRIBUTING FACTOR VEHICLE 1    0
CONTRIBUTING FACTOR VEHICLE 2    0
COLLISION_ID                     0
VEHICLE TYPE CODE 1              0
dtype: int64
```

#4. Machine Learning

import pandas as pd

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

from sklearn.preprocessing import OneHotEncoder

from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy_score

Load the cleaned-up dataset

```
print('Loading the dataset')

df = pd.read_csv('C:/Motor_Vehicle_Collisions_-_Crashes_cleaned_updated.csv')
```

Define the feature columns and the target column

```
print('Defining feature and target columns')

feature_cols = ['BOROUGH', 'CONTRIBUTING FACTOR VEHICLE 1', 'CONTRIBUTING
FACTOR VEHICLE 2', 'VEHICLE TYPE CODE 1']

target_col = 'CRASH DATE'
```

Split the dataset into training and testing sets

```
print('Splitting the dataset into training and testing sets')

X_train, X_test, y_train, y_test = train_test_split(df[feature_cols],
df[target_col], test_size=0.2, random_state=42)
```

```
# Define the column transformer to preprocess the data
print('Defining the column transformer to preprocess the data')
preprocessor = ColumnTransformer([
    ('cat', OneHotEncoder(handle_unknown='ignore'), ['BOROUGH', 'CONTRIBUTING FACTOR VEHICLE 1', 'CONTRIBUTING FACTOR VEHICLE 2', 'VEHICLE TYPE CODE 1']),
])
# Define the machine learning model
print('Defining the machine learning model')
model = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', DecisionTreeClassifier(random_state=42))
])

# Fit the model on the training data
print('Fitting the model on the training data')
model.fit(X_train, y_train)

# Use the model to make predictions on the testing data
print('Making predictions on the testing data')
y_pred = model.predict(X_test)

# Calculate the accuracy of the model
print('Calculating the accuracy of the model')
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy:', accuracy)
```

```
Loading the dataset
Defining feature and target columns
Splitting the dataset into training and testing sets
Defining the column transformer to preprocess the data
Defining the machine learning model
```

```
Fitting the model on the training data
Making predictions on the testing data
Calculating the accuracy of the model
Accuracy: 0.0014672807508327558
```

The machine learning model I have created is a Decision Tree Classifier that predicts the date of a crash based on the borough, contributing factors of the first and second vehicle, and vehicle type code. The data is preprocessed using a ColumnTransformer to one-hot encode the categorical variables. The model is then trained on the training data and tested on the testing data, achieving an accuracy of 0.0015, which is very low. This indicates that the features I have chosen are not good predictors of the crash date, or that there may be other variables that are more important in predicting the crash date. One potential way to improve the model is to include additional features such as weather conditions, time of day, and road conditions, which may have a greater impact on the occurrence of a crash. Additionally, different machine learning algorithms could be explored and compared to find the best model for this specific dataset. Finally, more data could be collected and added to the dataset to increase the sample size and potentially improve the accuracy of the model. Overall, this model can serve as a starting point for further analysis and improvement to better predict the occurrence of motor vehicle crashes.

#5. Conclusions

The analysis of the Motor Vehicle Collisions dataset from the New York City Police Department (NYPD) revealed several key findings and insights.

1. Brooklyn had the highest number of collisions among all the boroughs in New York City, followed by Queens and Manhattan.
2. Driver inattention/distraction was the most common contributing factor to motor vehicle collisions, followed by failure to yield right-of-way and backing unsafely.
3. Sedans were the top vehicle type involved in collisions, followed by passenger vehicles and station wagons/sport utility vehicles.
4. The machine learning model that was created to predict the date of a crash based on borough, contributing factors, and vehicle type code had a very low accuracy score, indicating the features selected may not be the best predictors of crash date.

Limitations and Challenges:

The analysis encountered several limitations and challenges. One of the main limitations was the missing data in the dataset, which led to dropping certain columns and rows. Additionally, the features selected for the machine learning model may not have been the most effective in predicting the crash date, as evidenced by the low accuracy score.

Recommendations for Future Research or Improvements:

To improve the accuracy of the machine learning model, additional features such as weather conditions, time of day, and road conditions could be included in the dataset. Additionally, exploring different machine learning algorithms could help identify the best model for predicting the crash date. Finally, collecting more data could improve the accuracy of the model and provide more insights into the factors contributing to motor vehicle collisions in New York City.

#6. References

New York City Police Department (NYPD). (2021). Motor Vehicle Collisions—Crashes. https://data.cityofnewyork.us/Public-Safety/Motor-Vehicle-Collisions-Crashes/h9gi-nx95.