Understanding How Urban features Affect Crime In New York City

1. General background

Jane Jacobs wrote in her book *The Death and Life of Great American Cities* that in order for a street to be a safe place, “there must be eyes upon the street, eyes belonging to those we might call the natural proprietors of the street”. This assertion highlights that the safety of the urban environment depends on its urban design and the ability to bring a community together in space. Thus, inspired by this theory, we set out to study how the urban environment, as measured in datasets about its features, can influence the types and rates of crimes committed in the area. The hope is that by identifying urban features which are closely linked to crime rates and types of crime, we may provide better guidance on crime prevention and intervention in the city.

Understanding which urban features are correlated with certain types of crimes or crime incidence rates allow for better data-driven urban design. Using data on feature importance can help urban planners design safer cities, and improve existing urban landscapes through various urban intervention measures to improve the safety of a neighborhood. Better knowledge of the urban factors which affect crime rates and types can also help in better allocation of law enforcement resources to certain areas of the city. This allocation of resources can also be aided through the use of predictive modeling using machine learning to predict crime types and incidence in certain parts of the city based on the characteristics of the location.

In order to do this, we selected New York City as our area of study. The rationale for this is partly due to the availability of open source data for study in New York City. New York City Open Data provides many different types of urban data, as well as crime data, allowing us to better investigate the relationships between these factors.

2. Initial Exploration of Data and Observations

There were multiple iterations of our project model, and in this section we will be focusing on the ones used in our final model.

Data Name	Data Source	Data Type
NYPD Complaints [1]	Police Department (NYPD)	Geocoded CSV with over 200000 rows
NYC Police Precincts [2]	Police Department (NYPD)	Geocoded CSV with 77

		rows (as there are 77 precincts)
NYC Sidewalk Cafe [3]	New York City Department of Consumer and Worker Protection (DCWP)	Shapefile with over 200 rows
NYC Public Plaza [4]	NYC Department of Information Technology and Telecommunications (DoITT)	Shapefiles with over 1000 rows
NYC Street Centerline [5]	NYC Office of Technology and Information (OTI)	Shapefile with 120884 rows

NYPD Complaint and Police Precinct Dataset

To conduct exploratory data analysis on a data set with over two hundred thousand rows, we explored the NYPD Complaints dataset based on precincts. New York City is divided into 77 precincts and each is divided as much as possible with the boundaries of actual established neighborhoods. We were able to get a table of the number of complaints in each precinct by simply using **'groupby'** in python.

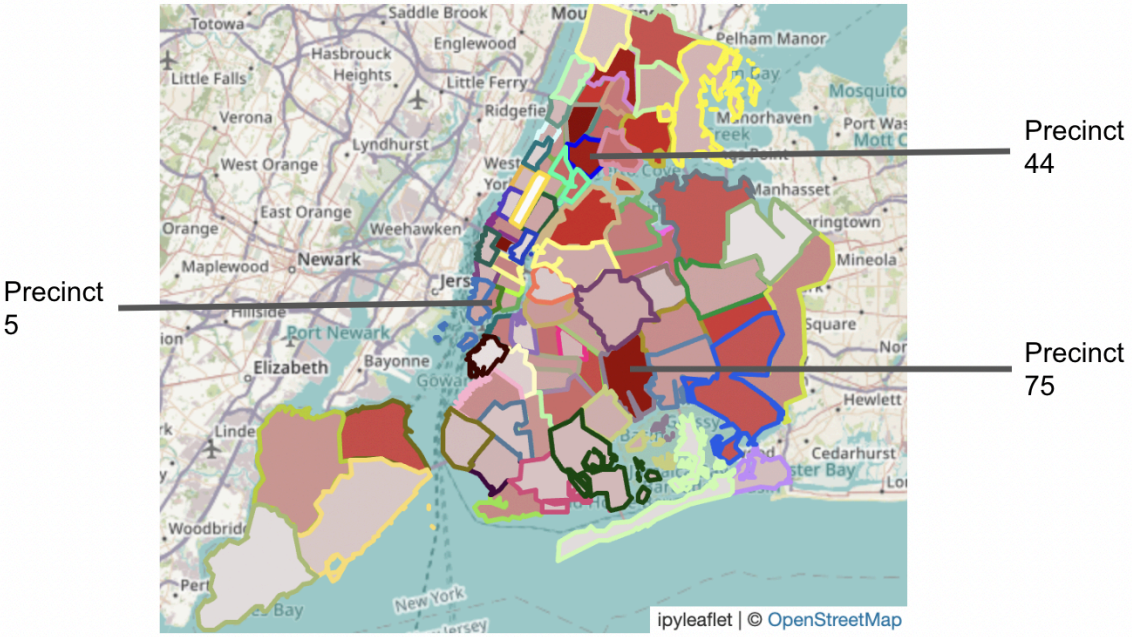
```
precinct_counts = pd.DataFrame(df_crime_complaints.value_counts(subset=['ADDR_PCT_CD']).rename_axis("Precinct").reset_index(name='counts'))
precinct_counts
```

Python

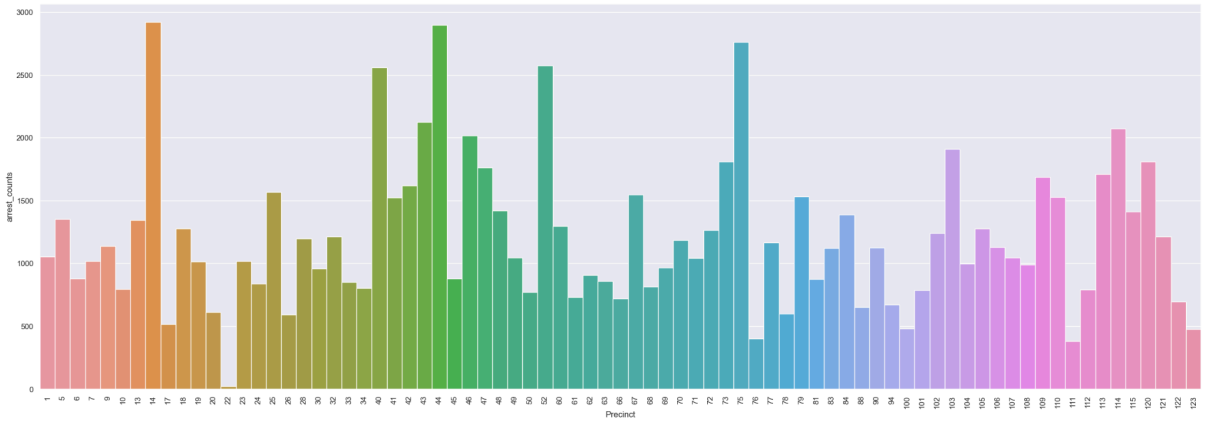
	Precinct	counts
0	75	8338
1	44	6320
2	14	6178
3	43	6036
4	114	5659
...	...	...
72	26	1492
73	100	1358
74	123	1263
75	76	1061
76	22	116

77 rows × 2 columns

However, it was difficult to understand the data, as we have no understanding of how each precinct number relates to actual known neighborhoods of New York City. This is the reason we had to use the NYC Police Precinct dataset as it gives us the geometry of each precinct and geocoded. Thus by merging it with the complaint data we can find out which precincts have the greatest number of complaints. The map below is the result of the merge of the two dataset and plotted above OpenStreetMap. where the darker red the color of polygons are, the higher number of counts of complaints in the precinct.



The histogram below shows how we used data visualization to approach our datasets initially to identify points of interest. Here, the number of crime complaints per precinct are represented in the bar graph, allowing us to make observations about the precincts with the highest crime rates.



There are a few interesting observations we found from exploring the complaint dataset that we presented in checkpoint 2.

1. Precinct 44 : 2 E 169th St, The Bronx has the highest complaint for murder
2. Precinct 75: 1000 Sutter Avenue, Brooklyn has the highest complaint count and majority of complaints are misdemeanor such as petit larceny
3. Precinct 5: 19 Elizabeth St: Chinatown, Little Italy has the highest arrest to complaint count. It was brought up in class that a possibility of this is due to asian hate since our data used consist mainly from 2022. Hence as an extension in our project we looked at how covid affected crime in NYC as well.

The first two observations pointed to us that there are large differences in type of crime in different parts of NYC which becomes our focus in our model later on.

NYC Sidewalk Cafe Data

The purpose for looking into Sidewalk Cafe Data was because our group was interested in looking at urban features in cities that affect crime. In our initial exploration, we found that sidewalk cafes were concentrated in the Manhattan areas and in our initial hypothesis we were expecting petit larceny crimes to be more frequent in areas with high counts of Sidewalk cafes.

NYC Public Plaza

The purpose for looking into Public Plaza Data was because public plazas represent how open the space is to residents and the community at large. Our hypothesis is that streets without public plazas might be more secluded, and also have less of a strong community around it, and thus increase the amount of crimes committed in the area.

Street Centreline

The Street Centreline dataset came into use much later on in the process of our project, as we realized that there was a need to move away from a precinct-centric view of crime, to recognising that proximity to certain streets influenced the nature and frequency of crime. This is supported by the fact that crimes in New York City tend to happen on a street level at certain hotspots in the city, and are less distributed according to precinct. Concurrently, it also solves a problem we had initially of having too small of a dataset in grouping crimes by precinct, allowing for more useful modeling.

3. First iteration of our model: Predicting percentage composition of crime in each Precinct

In this first iteration we were looking at predicting the percentage composition of crime committed in each precinct. We were interested if the urban features which we explored in the initial data analysis could help us in predicting the types of crime committed in each precinct. Our initial hypothesis for this was that places that are more touristy (shown by urban features such as higher number sidewalk cafes) would have generally have higher petit larceny crimes.

The steps for this model can be broken into the following steps:

1. Loading the datasets as Pandas dataframe as we were going to join them based on the geocoded precinct data (Police Precinct, Complaint, Arrest, Sidewalk Cafe, Plazas)

```
nypd_precincts=pd.read_csv("nypp.csv")
#need GeoDataFrame inorder to spatial join

from shapely import wkt

nypd_precincts['the_geom'] = geopandas.GeoSeries.from_wkt(nypd_precincts['the_geom'])
precincts = geopandas.GeoDataFrame(nypd_precincts, geometry='the_geom')
```

2. Spatial join with geopandas (combining the pandas dataframe and the precinct dataframe by giving each row in the various dataset (complaint, arrest, sidewalk cafe, plaza) a precinct number on whether it is within the geometry of the precinct (Police Precinct dataset)
   Example using Sidewalk cafe dataset:

```
sidewalk_cafe = pd.read_csv("./sidewalk_cafe.csv")
sidewalk_cafe = geopandas.GeoDataFrame(sidewalk_cafe, geometry=geopandas.points_from_xy(sidewalk_cafe.LONGITUDE, sidewalk_cafe.LATITUDE))
cafe_join = sidewalk_cafe.sjoin(precincts, how="left", predicate="within")
```

3. Summarize the dataset at a precinct level. For each precinct we found the number of plazas and sidewalk cafes.
   Below shows the precinct number and the number of cafes in each precinct:

```
cafecount = cafe_join["Precinct"].value_counts().to_frame('cafes')
cafecount
```

	cafes
19.0	123
6.0	107
114.0	85
20.0	84
9.0	68

4. Collate all possible features we want to include in the model in one dataframe, where shape_area is simply the total area of the precinct.

```
finalpredictivedata = precincttrainingdata[["cafes","plazas","Shape_Area"]]
finalpredictivedata["plazas"]=finalpredictivedata["plazas"].fillna(0)
finalpredictivedata
```

Precinct	cafes	plazas	Shape_Area
1.0	62.0	79.0	4.728646e+07
5.0	55.0	13.0	1.809453e+07
6.0	107.0	16.0	2.210333e+07
7.0	10.0	7.0	1.836667e+07
9.0	68.0	5.0	2.139539e+07
...	...	...	...
115.0	3.0	7.0	1.141197e+08
120.0	0.0	19.0	2.323380e+08
121.0	0.0	34.0	4.755776e+08
122.0	0.0	17.0	4.548530e+08
123.0	0.0	6.0	4.608627e+08

77 rows × 3 columns

5. Transform the data by encoding the offense descriptions, and grouping the data frame by precinct and the types of crimes committed in the precinct

```
from sklearn.linear_model import LinearRegression
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
```

```
preprocessor = ColumnTransformer([('encoder', OneHotEncoder(sparse=False), ["OFNS_DESC"])], remainder='passthrough')
transformed = preprocessor.fit_transform(df_crime_complaints)
X = pd.DataFrame(transformed, columns=preprocessor.get_feature_names_out())
X = X.drop(columns =['remainder__CMPLNT_NUM',
       'remainder__BORO_NM',
      'remainder__CMPLNT_FR_DT', 'remainder__CMPLNT_FR_TM',
      'remainder__CMPLNT_TO_DT', 'remainder__CMPLNT_TO_TM',
      'remainder__CRM_ATPT_CPTD_CD', 'remainder__HADEVELOPT',
      'remainder__JURIS_DESC', 'remainder__KY_CD', 'remainder__LAW_CAT_CD',
      'remainder__LOC_OF_OCCUR_DESC', 'remainder__PARKS_NM',
      'remainder__PD_CD', 'remainder__PD_DESC', 'remainder__PREM_TYP_DESC',
      'remainder__RPT_DT', 'remainder__Lat_Lon', 'remainder__X_COORD_CD',
      'remainder__Y_COORD_CD', 'remainder__Latitude', 'remainder__Longitude'])
X.rename(columns ={"remainder__ADDR_PCT_CD":"Precinct"}, inplace=True)
print(X.columns)
```

```
X = X.groupby(["Precinct"])['encoder__OFNS_DESC_(null)', 'encoder__OFNS_DESC_ADMINISTRATIVE CODE',
      'encoder__OFNS_DESC_AGRICULTURE & MRKTS LAW-UNCLASSIFIED',
      'encoder__OFNS_DESC_ALCOHOLIC BEVERAGE CONTROL LAW',
      'encoder__OFNS_DESC_ANTICIPATORY OFFENSES', 'encoder__OFNS_DESC_ARSON',
      'encoder__OFNS_DESC_ASSAULT 3 & RELATED OFFENSES',
      'encoder__OFNS_DESC_BURGLAR\'S TOOLS', 'encoder__OFNS_DESC_BURGLARY',
      'encoder__OFNS_DESC_CHILD ABANDONMENT/NON SUPPORT',
      'encoder__OFNS_DESC_CRIMINAL MISCHIEF & RELATED OF',
      'encoder__OFNS_DESC_CRIMINAL TRESPASS',
      'encoder__OFNS_DESC_DANGEROUS DRUGS',
      'encoder__OFNS_DESC_DANGEROUS WEAPONS',
      'encoder__OFNS_DESC_DISORDERLY CONDUCT',
      'encoder__OFNS_DESC_DISRUPTION OF A RELIGIOUS SERV',
      'encoder__OFNS_DESC_ENDAN WELFARE INCOMP',
```

6. Use multi-output linear regression to create a predictive model.

```
We create our multi-output regressor here, train it, and score it.
```

```
from sklearn.datasets import make_regression
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.multioutput import MultiOutputRegressor
from sklearn.ensemble import RandomForestRegressor

clf = MultiOutputRegressor(RandomForestRegressor(n_estimators=25,max_depth=2, random_state=10))
clf.fit(X_train, y_train)
predicted=clf.predict(X_test)
clf.score(X_test, y_test)
```

✓ 0.6s

```
-0.4713537988431775
```

7. Calculate the accuracy of the model by dividing the predicted result by the expected result, giving a percentage score.

```
accuracy = predicteddata/testdata
accuracy.replace([np.inf, -np.inf], 0, inplace=True)
accuracy.fillna(0, inplace=True)
accuracy
```

✓ 0.6s

Precinct	encoder__OFNS_DESC_ADMINISTRATIVE CODE	encoder__OFNS_DESC_AGRICULTURE & MRKTS LAW-UNCLASSIFIED	encoder__OFNS_DESC_ALCOHOLIC BEVERAGE CONTROL LAW
49	0.403687	0.000000	0.000000
71	0.650860	0.000000	0.000000
44	0.969070	0.931498	0.000000
121	0.736465	0.000000	0.000000
101	0.614008	0.420662	0.000000
102	0.527531	1.002682	0.000000
104	0.529217	0.000000	0.178165
81	0.728355	0.564640	0.000000
62	1.508679	0.000000	0.000000
100	0.946394	0.194274	0.000000
19	4.449535	0.000000	0.000000
84	0.577794	0.000000	0.000000
60	1.140196	0.653314	0.000000
6	2.806205	0.000000	0.000000
114	1.010724	0.000000	0.057818
45	0.663172	0.000000	0.058135

8. Observe the mean and standard deviation of the accuracy for each type of crime, and use the better predicted ones as dependent variables in future iterations of the model.

	mean	std
encoder__OFNS_DESC_INTOXICATED/IMPAIRED DRIVING	1.000000	0.000000
encoder__OFNS_DESC_KIDNAPPING	1.000000	0.000000
encoder__OFNS_DESC_JOSTLING	1.000000	0.000000
encoder__OFNS_DESC_UNLAWFUL POSS. WEAP. ON SCHOOL	1.000000	0.000000
encoder__OFNS_DESC_HOMICIDE-NEGLIGENT-VEHICLE	1.000000	0.000000
encoder__OFNS_DESC_OFFENSES RELATED TO CHILDREN	1.000000	0.000000
encoder__OFNS_DESC_OTHER STATE LAWS	1.000000	0.000000
encoder__OFNS_DESC_ESCAPE 3	1.000000	0.000000
encoder__OFNS_DESC_DISRUPTION OF A RELIGIOUS SERV	1.000000	0.000000
encoder__OFNS_DESC_FELONY SEX CRIMES	1.000000	0.000000
encoder__OFNS_DESC_ANTICIPATORY OFFENSES	1.000000	0.000000
encoder__OFNS_DESC_PROSTITUTION & RELATED OFFENSES	1.000000	0.000000
encoder__OFNS_DESC_NYS LAWS-UNCLASSIFIED VIOLATION	0.997032	0.011873
encoder__OFNS_DESC_FRAUDULENT ACCOSTING	0.991920	0.032320
encoder__OFNS_DESC_LOITERING/GAMBLING (CARDS, DIC	0.989285	0.042859
encoder__OFNS_DESC_ALCOHOLIC BEVERAGE CONTROL LAW	0.981618	0.046946
encoder__OFNS_DESC_DISORDERLY CONDUCT	0.983981	0.055243
encoder__OFNS_DESC_CHILD ABANDONMENT/NON SUPPORT	0.983864	0.064544
encoder__OFNS_DESC_HOMICIDE-NEGLIGENT,UNCLASSIFIE	0.979008	0.083969
encoder__OFNS_DESC_ASSAULT 3 & RELATED OFFENSES	0.001275	0.141412
encoder__OFNS_DESC_ENDAN WELFARE INCOMP	0.947926	0.154085
encoder__OFNS_DESC_GAMBLING	0.942080	0.179372
encoder__OFNS_DESC_PETIT LARCENY	0.001760	0.205113
encoder__OFNS_DESC_HARRASSMENT 2	0.049025	0.215234

One problem we faced when attempting this was the low accuracy of the model when we tried to score it. This was likely due to the fact that our independent variables were too few, and also due to the fact that our decision to categorize the crimes committed in NYC by precinct left us only with 77 unique data points. Consequently, our model struggled to find

meaningful relationships within the training data, and so had limited accuracy when predicting.

Another problem we faced was also the overselection of dependent variables. However, after carrying out step 8 and observing the accuracy of our model in predicting some crime proportions over others, we could re-adjust the model to accept new output expectations. For instance, we tweaked the model to predict assault, harassment, murder, robbery, rape, and petit larceny, and consequently attained a better score than the initial model.

```
Y=Y_bak[['encoder__OFNS_DESC_ASSAULT 3 & RELATED OFFENSES',
         'encoder__OFNS_DESC_HARRASSMENT 2',
         'encoder__OFNS_DESC_MURDER & NON-NEGL. MANSLAUGHTER',
         'encoder__OFNS_DESC_PETIT LARCENY',
         'encoder__OFNS_DESC_RAPE',
         'encoder__OFNS_DESC_ROBBERY']]
Y
```

✓ 0.4s

```
Scoring it again, we see an improvement in the score
```

```
clf = MultiOutputRegressor(RandomForestRegressor(n_estimators=25,max_depth=2, random_state=10))
clf.fit(X_train, y_train)
predicted=clf.predict(X_test)
clf.score(X_test, y_test)
```

✓ 0.1s

```
0.2594034748473971
```

As seen above, our model's score improved from being -0.47 to 0.26, demonstrating how the model is held back by the attempt to predict too many types of crimes at once. This also highlights the different nature of crimes committed in New York City, such that not all of them experience the same relationship with urban features. Thus, we decided to progress with a classifier model instead, as it will allow for more easy tweaking of our independent and dependent data to improve our model's predictive score, as a consequence of the ability to visualise the data in a confusion matrix, and also analyse the importance of each input feature.

4. Second iteration of our model: Predicting type of crime

A struggle we faced with our first model was the low number of training and test dataset and the low accuracy and reliability as a result of it. Hence, instead of grouping the complaint dataset by precincts we analyze them as individual data points which increased the total number of data used in our model from the 77 precincts to 256797 the total number of complaints we have in our complaint dataset. However, we did not want to exclude our hard

work and initial interest in urban features as they were definitely still useful especially in predicting types of crimes.

The steps in this iteration of the model are as follows:

1. Dropping irrelevant columns

```
df = df_crime_complaints.copy()

df = df.drop(columns = ['CMPLNT_NUM','BORO_NM','CMPLNT_TO_DT','CMPLNT_TO_TM','CRM_ATPT_CPTD_CD','HADEVELOPT',
'JURIS_DESC','KY_CD','LAW_CAT_CD','LOC_OF_OCCUR_DESC','PARKS_NM','PD_CD','PD_DESC','PREM_TYP_DESC',
'RPT_DT','X_COORD_CD','Y_COORD_CD'])
```

2. Label encode the type of crime (because random forest classification is unable to deal with the string type)

```
from sklearn.preprocessing import LabelEncoder

lb_make = LabelEncoder()
df['OFNS_DESC'] = lb_make.fit_transform(df_crime_complaints['OFNS_DESC'])
```

3. Cleaning the data so that it is usable in random forest classification. For example, the date column is in the format of mm/dd/yyyy hence we had to first use the method string.split('/') to obtain the month and year as its own column

```
#split date into columns
df[["month", "day", "year"]] = df["CMPLNT_FR_DT"].str.split("/", expand = True)
df= df.drop(columns = 'CMPLNT_FR_DT')

#get only the hour in order to do random forest
df['CMPLNT_FR_TM'] = pd.to_datetime(df['CMPLNT_FR_TM'])
# extract hour from the timestamp column to create an time_hour column
df['hour'] = df['CMPLNT_FR_TM'].dt.hour
df= df.drop(columns = 'CMPLNT_FR_TM')
```

4. Add column of distance to nearest sidewalk cafe

```
distance_to_cafe = gdf.geometry.apply(lambda x: sidewalk.distance(x).min())
['nearest_cafe'] = distance_to_cafe
df
```

5. Add column of distance to nearest plaza

```
nearest_plaza = gdf.geometry.apply(lambda x: plazas.distance(x).min())
df['nearest_plaza'] = nearest_plaza
df
```

6. Add column of nearest street id

```
def nearest(point,street):
    precinct = point["ADDR_PCT_CD"]
    streets_to_check = street.loc[street["Precinct"]==precinct]
    distances = streets_to_check.distance(point["geometry"]).idxmin()
    dist = street.iloc[distances,:]["full_stree_id"]
    return dist
```

```
street_join= street_join.to_crs(2263)
gdf=gdf.to_crs(2263)
nearest_street = gdf.apply(lambda x: nearest(x,street_join),  axis =1)
df["nearest_street"] = nearest_street
```

7. Run the random forest classifier with the encoded type of crime as Y and dropping columns we are not using in our model.

```
y2 = df["OFNS_DESC_ID"]
X2 = df.drop(['OFNS_DESC','OFNS_DESC_ID','Lat_Lon','geometry', 'day','Latitude','Longitude'], axis=1)


from sklearn.model_selection import train_test_split
X_train2, X_test2, y_train2, y_test2 = train_test_split(X2, y2, test_size=0.3, random_state = 44)
from sklearn.ensemble import RandomForestClassifier
rf_model2 = RandomForestClassifier(n_estimators=500, max_features="auto")
rf_model2.fit(X_train2, y_train2)

predictions = rf_model2.predict(X_test2)
predictions
```

The reason for adding the step 6 where we labeled each crime complaint to the nearest street is because we expected there to still be influence based on the location on the type of crime other than the features such as proximity to sidewalk cafes and plazas or hour of the day but importantly where the crime was committed. The initial plan was to use longitude and latitude however the model would consider it as two separate features instead of one.

This step was a lot more challenging than we planned for it to be and it was the result of the NYC Street Name data set having over 100000 rows where we were trying to match the longitude and latitude of each complaint (200000+) from the complaint dataset. It was taking exponential time in order to run the function of finding the minimum distance to a street. Furthermore, we wanted the id of the street (which we first used as a label encoder for the 100000+ street names) instead of the nearest distance. Hence this is what the nearest function defined in step 6 helps in as it breaks down the searching process to only match the complaints with streets names that are in the same precinct instead of all 100000+ street names. The code still took 40 minutes to run each time and here we can learn to export our dataset to avoid rerunning the code every single time we work on the project.

Here are some interesting findings from the above method.
Firstly, with the inclusion of street names we are able to identify the street with the most number of complaints. We found this to be the street Broadway, which we recognise to be a

really long street hence it might have been a factor to why there are higher complaint counts. However, it is also undeniable that Broadway is special street in NYC given its historical and economic significance.

```
#df["nearest_street"].value_counts()
reet_join_dict[street_join_dict["full_stree_id"]==2183]
```

	full_stree	full_stree_id
377	BROADWAY	2183

Secondly, we were able to score our model's accuracy which we found to be 0.29 (which is low, we know but it is an improvement!).

```
rf_model2.score(X_test2,y_test2)
```

```
0.2947429906542056
```

This is the best accuracy we could get even playing with some of the conditions of our random forest classifier such as n_estimators which is the number of decision trees to build before taking the average of the predictions. However, given that there are 59 type of crime and some are very similar in nature we believe that this is fairly decent accuracy! In the next step we will be discussing more improvements that can be done to the model.

Lastly, we were able to run our random forest classification and see each of the feature's importance. This is useful as we can observe each features' importance on the type of crime based on the model and compare it to real life expectations of each feature.

```
importance = rf_model2.feature_importances_
lumns = X2.columns
i = 0

while i <len(columns):
    print (f" Importance of the feature' '{columns[i]}' is {round(importance[i]*100,2)}%.")
    i += 1
```

```
Importance of the feature' 'ADDR_PCT_CD' is 9.78%.
Importance of the feature' 'month' is 14.51%.
Importance of the feature' 'year' is 0.57%.
Importance of the feature' 'hour' is 19.47%.
Importance of the feature' 'nearest_cafe' is 19.7%.
Importance of the feature' 'nearest_plaza' is 19.42%.
Importance of the feature' 'nearest_street' is 16.55%.
```

The result above shows us that distance to the nearest cafe and plaza matters the most in our random forest classifier in the type of crime. This points to an area of possible further exploration, what if we focused on predicting the type of crime in areas with similar distance

to nearest cafe and plaza (which can be taken as proxies to where people crowd). For example, selecting a sufficient amount of data with short distances to nearest cafe and plaza, hence they are all supposedly places with high count of people and redoing the random forest classifier might be able to produce a model with a higher accuracy as it would be for a more specific area.

Another possibility of future exploration is to simply drop certain crime types and have our model only work on predicting these fewer types of crimes rather than 59 different types of crime. This is a possible avenue as we found out that our model was better able to predict certain crimes such as petit larceny. Out of the test data set there were 16505 supposedly petit larceny complaints. Our model was able to predict it 9323 of the times right. This is much greater than the average accuracy of our model. Hence this suggests that a possible future direction of predicting crimes in NYC is to narrow down on specific crimes as some may correlate with temporal and spatial features much more than other crimes (which makes sense since petit larceny crimes happen frequently where crowds are while other crimes such as fraud are less spatially and temporally influenced).

5. Third iteration of our model (applying the second model to just Manhattan island precincts, tagging each crime with respective neighborhood and its features)

As mentioned previously, we were wondering if narrowing to certain areas such as Manhattan island would be able to give us more accurate scores given that the areas there would probably be more similar to one another. We specifically chose Manhattan island as well as we expect it to be higher in crimes that our model is better able to predict such as petit larceny given the high tourist counts.

We followed the same steps in the previous section but only with the complaints that are filed in manhattan island. The code below shows how we selected complaints only from manhattan island as the precinct number of manhattan is between 1 and 34.

```
df_manhattan = df
# df_manhattan["ADDR_PCT_CD"] = df["ADDR_PCT_CD"].astype('int')

_manhattan = df_manhattan[df_manhattan["ADDR_PCT_CD"]<=34]
df_manhattan
```

We again ran our random forest classifier and this time round we found a higher score of 0.33 compared to our model 2 that had 0.29. The most interesting observation from running this model is seeing the change in difference of importance in the features. When we ran the model on just manhattan island we saw that the feature importance of distance to nearest plaza and cafe greatly reduced while hour greatly increased. This makes sense as there is probably much less variation in those two features when we are looking only at manhattan hence random forest classifier gives it a lower importance as a feature.

```python
importance = rf_model3.feature_importances_
columns = X3.columns
i = 0

while i <len(columns):
    print (f" Importance of the feature' '{columns[i]}' is {round(importance[i]*100,2)}%.")
    i += 1
```

```
Importance of the feature' 'ADDR_PCT_CD' is 7.31%.
Importance of the feature' 'month' is 17.71%.
Importance of the feature' 'year' is 0.89%.
Importance of the feature' 'hour' is 25.88%.
Importance of the feature' 'nearest_cafe' is 17.17%.
Importance of the feature' 'nearest_plaza' is 17.24%.
Importance of the feature' 'nearest_street' is 13.8%.
```

6. Conclusion and reflection on the project

This project has come really far from our initial interest in crime and urban features. We were initially limited by our data as there were no datasets that we could find that simply included both of them. Hence, looking towards using geographic information systems (GIS) in python was a significant step in this project. Combining the different datasets was definitely challenging. We initially tried to solve this by summarizing the different datasets to precinct level of analysis. This proved to be a naive solution as we found that we were unable to produce models with such little data (77 data points from the 77 precincts). Therefore, we came to relook at our dataset and instead each of them as unique data points and map urban features to each of them rather than each precinct. This presented a brand new set of challenges such as increased time-complexity in our codes (we learnt that patience is a virtue that data scientists or at least those without supercomputers need).

In our conclusion we would also like to highlight a repeating challenge in dealing with geospatial datasets. As all of us were unfamiliar with GIS at the time of working on the project, there was a lot of trial and error in understanding how to use methods in GIS packages such as geopandas. For example, a recurring problem was the error message asking us to set a coordinate reference system (CRS) but what are we supposed to set it to? The worst problem arises when python does not even warn us about checking the CRS and we find ourselves finding ridiculous distances for the nearest distance to sidewalk cafe and plaza because of the wrong CRS (after taking a ridiculously long time to run the code). A key learning point here is when using geospatial data, always check the CRS and understand what they mean. Global CRS are usually in units of degrees rather than meters hence when dealing with localized datasets it is preferable to set it to a local CRS.

Finally, after overcoming the numerous challenges we see the fruit of our hard work in the final model that seemed to be quite reasonable in its accuracy and the fact that it corroborates features that we see in real life that affects crime.

Moreover, another study has found similar results in using spatial-temporal features in predicting the type of crime in NYC. They suggest that rather based on streets (which we have done) or politically demarcated areas such as precincts, crimes were more spatially affected by distance to hotspots [6]. Hence a possible further exploration of this project would be to use models to predict the locations of these hotspots which might be more telling of crime in NYC.

Crimes are committed based on large pools of people as there are higher crime opportunities in those areas [7]. In addition to the possible future plans discussed in section 4 in evaluation of our second model, another dataset we definitely can look into is the population of people working or living in those areas.

Follow-up on possibility of COVID-19 affecting crime in Section 2: Investigating COVID-19

The project has opened up opportunities to incorporate and implement various other possible indicators of street crime. Here, we slightly diverged from the project and analyzed the effects of diseases like COVID-19 on street crime in New York.

COVID-19 brought a drastic change in all our lives, specifically in our environment. We speculated that COVID-19 would affect the street crime numbers too. Although COVID-19 is non-existent these days, new mutants or diseases similar to nature in COVID-19 seem inevitable. Hence, by modeling COVID-19's effect on street crime, we can use the model to analyze the effects the diseases, similar in nature to COVID, may have on street crime in the future

To explore COVID data, we imported NYPD Complaints historic data, so we could access the complaint data in COVID years. For the sake of brevity, we classified the year 2020 filled with complaints made during the COVID 19 epidemic. We will compare the 2020 crime data to 2019 crime data, which was COVID free. Our rationale was to compare a similar amount of data points, instead of all years with COVID in relation to all years with non-Covid. In that case, Covid data only accounted for 2.5 years, whereas non-Covid for the rest of the data in the historic data.

1) We load the data and drop any duplicates for consistency in the data.

```
df_crime_complaints = pd.read_csv("NYPD_Complaint_Data_Historic.csv")
df_crime_complaints.drop_duplicates()
df_crime_complaints
```

The NYPD Historic data has identical columns to the data complaints data we analyzed above. Hence, we followed the same steps in cleaning and processing the data.

	CMPLNT_NUM	CMPLNT_FR_DT	CMPLNT_FR_TM	CMPLNT_TO_DT	CMPLNT_TO_TM	ADDR_PCT_CD	RPT_DT	KY_CD	OFNS_DESC	PD_CD	...	SUSP_SEX	TRANSIT_DISTRICT	Latitude	Longitude	Lat_Lon	PATROL_BORO	STATION_NAME	VIC_AGE_GROUP	VIC_RACE	VIC_SEX
0	506547392	03/29/2018	20:30:00	NaN	NaN	32.0	03/30/2018	351	CRIMINAL MISCHIEF & RELATED OF	254.0	...	NaN	NaN	40.810877	-73.941064	(40.810877241, -73.941064151)	PATROL BORO MAN NORTH	NaN	25-44	WHITE	F
1	629632833	02/06/2018	23:15:00	NaN	NaN	52.0	02/07/2018	341	PETIT LARCENY	333.0	...	F	NaN	40.873671	-73.908014	(40.873671035, -73.908013649)	PATROL BORO BRONX	NaN	UNKNOWN	UNKNOWN	D
2	787203902	11/21/2018	00:15:00	11/21/2018	00:20:00	75.0	11/21/2018	341	PETIT LARCENY	321.0	...	F	NaN	40.651782	-73.885457	(40.651782232, -73.885456761)	PATROL BORO BKLYN NORTH	NaN	UNKNOWN	UNKNOWN	D
3	280364018	06/09/2018	21:42:00	06/09/2018	21:43:00	10.0	06/10/2018	361	OFF. AGNST PUB ORD SENSBLTY &	639.0	...	M	NaN	40.759310	-73.994706	(40.759310399, -73.994706072)	PATROL BORO MAN SOUTH	NaN	18-24	WHITE HISPANIC	F
4	985800320	11/10/2018	19:40:00	11/10/2018	19:45:00	19.0	11/10/2018	341	PETIT LARCENY	333.0	...	F	NaN	40.764536	-73.970728	(40.764535539, -73.970728388)	PATROL BORO MAN NORTH	NaN	UNKNOWN	UNKNOWN	D
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5203937	632890596	12/08/2011	14:15:00	12/08/2011	22:00:00	44.0	12/09/2011	359	OFFENSES AGAINST PUBLIC ADMINI	749.0	...	M	NaN	40.824144	-73.917173	(40.824144257, -73.917172879)	PATROL BORO BRONX	NaN	18-24	WHITE HISPANIC	F
5203938	905387511	04/15/2010	17:00:00	NaN	NaN	52.0	04/15/2010	361	OFF. AGNST PUB ORD SENSBLTY &	639.0	...	M	NaN	40.860460	-73.903799	(40.860460111, -73.903798573)	PATROL BORO BRONX	NaN	18-24	UNKNOWN	M
5203939	348532820	03/29/2007	11:10:00	03/29/2007	11:20:00	14.0	03/29/2007	341	PETIT LARCENY	333.0	...	NaN	NaN	40.751864	-73.993707	(40.751863853, -73.993706898)	PATROL BORO MAN SOUTH	NaN	NaN	UNKNOWN	D
5203940	785516627	03/28/2007	10:05:00	03/28/2007	10:05:00	43.0	03/28/2007	347	INTOXICATED & IMPAIRED DRIVING	905.0	...	NaN	NaN	40.826284	-73.859704	(40.826284436, -73.859704288)	PATROL BORO BRONX	NaN	45-64	BLACK	F
5203941	753764143	03/05/2010	23:00:00	03/05/2010	23:30:00	70.0	03/08/2010	578	HARRASSMENT 2	638.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

2) The important part was to convert the column with the Report Date into Date-Time Index. This allowed us to create another column, ‘year’, within our dataframe. The column was filled by accessing the year of the Report Date which we converted into Date-Time Index. Then, through conditioning values, we filter data and store complaints made in Covid Year (2020) in covid_data, whereas complaints made in Non-Covid year (2019) in non_covid.

```
df_crime_complaints['RPT_DT'] = pd.to_datetime(df_crime_complaints['RPT_DT']).dt.date

df_crime_complaints['RPT_DT']
df_crime_complaints['year'] = pd.DatetimeIndex(df_crime_complaints['RPT_DT']).year

covid_data = df_crime_complaints.loc[df_crime_complaints['year'] == 2020]
non_covid = df_crime_complaints.loc[df_crime_complaints['year'] == 2019]
```

3) Next, we made some initial explorations, by looking at the number of crimes in each precinct in both 2019 and 2020. Apparently, from our initial glance, the number of crimes in 2020 decreased from 2019, perhaps signifying the effect of COVID has on crime level. However, a guiding question for us would be whether this difference is statistically significant.

```
precinct_counts_for_covid = pd.DataFrame(covid_data.value_counts(subset=['ADDR_PCT_CD']).rename_axis("Precinct").reset_index(name='counts'))
precinct_counts_for_covid['']
```

	Precinct	counts
0	75.0	12814
1	40.0	10630
2	44.0	9579
3	43.0	9369
4	47.0	8966
...	...	...
72	100.0	2594
73	111.0	2483
74	123.0	2024
75	76.0	2004
76	22.0	200

```
precinct_counts_for_non_covid = pd.DataFrame(non_covid.value_counts(subset=['ADDR_PCT_CD']).rename_axis("Precinct").reset_index(name='counts'))
precinct_counts_for_non_covid
```

	Precinct	counts
0	75.0	14256
1	40.0	13040
2	44.0	11760
3	52.0	9892
4	14.0	9775
...	...	...
72	100.0	2792
73	111.0	2624
74	76.0	2435
75	123.0	2146
76	22.0	255

4) Next, using the geocoded data of NYPD precincts, we merge it with the data with crime count in each precinct in covid and non-covid year. We created a Count Density using the NYPD Precinct data, dividing the number of crimes in the precinct by its shape area. In this way, we are able to judge crime rates better, since the number of crimes occurring in a precinct could be dependent on the area of the precinct.

```
merged_precincts_for_non_covid = pd.merge(nypd_precincts, precinct_counts_for_non_covid, on=["Precinct"])
merged_precincts_for_non_covid["CountDensity"] = merged_precincts_for_non_covid["counts"]/merged_precincts_for_non_covid["Shape_Area"]
merged_precincts_for_non_covid.sort_values(by=["CountDensity"], ascending=False, inplace=True)
merged_precincts_for_non_covid.reset_index(inplace=True)
merged_precincts_for_non_covid.drop(columns=["index"],inplace=True)
# merged_precincts["the_geom"].replace("MULTIPOLYGON ", "", regex=True, inplace=True)
# merged_precincts["the_geom"].replace("\(", "", regex=True, inplace=True)
# merged_precincts["the_geom"].replace("\)", "", regex=True, inplace=True)
merged_precincts_for_non_covid
```

	the_geom	Precinct	Shape_Leng	Shape_Area	counts	CountDensity
0	MULTIPOLYGON (((-73.97464798076302 40.75336712...	14	20973.820132	2.051113e+07	9775	0.000477
1	MULTIPOLYGON (((-73.9413174228129 40.807713346...	28	17106.149405	1.529402e+07	5169	0.000338
2	MULTIPOLYGON (((-73.98155997085203 40.74388188...	13	27719.207375	2.950766e+07	7709	0.000261
3	MULTIPOLYGON (((-73.98863862848766 40.72293372...	5	18807.124911	1.809453e+07	4702	0.000260
4	MULTIPOLYGON (((-73.93224093714262 40.78969488...	23	19427.884542	2.146668e+07	5564	0.000259
...	...	...	...	...	...	...
72	MULTIPOLYGON (((-74.15945602438066 40.64144833...	121	136811.464647	4.755776e+08	5635	0.000012
73	MULTIPOLYGON (((-73.74461376070957 40.77894737...	111	103218.168638	2.596987e+08	2624	0.000010
74	MULTIPOLYGON (((-74.05050806403247 40.56642203...	122	154842.244366	4.548530e+08	4370	0.000010
75	MULTIPOLYGON (((-73.94923327082499 40.79687199...	22	32712.763701	3.831220e+07	255	0.000007
76	MULTIPOLYGON (((-74.1698258238223 40.561090420...	123	120813.800441	4.608627e+08	2146	0.000005

77 rows × 6 columns

```
merged_precincts_for_covid = pd.merge(nypd_precincts, precinct_counts_for_covid, on=["Precinct"])
merged_precincts_for_covid["CountDensity"] = merged_precincts_for_covid["counts"]/merged_precincts_for_covid["Shape_Area"]
merged_precincts_for_covid.sort_values(by=["CountDensity"], ascending=False, inplace=True)
merged_precincts_for_covid.reset_index(inplace=True)
merged_precincts_for_covid.drop(columns=["index"],inplace=True)
# merged_precincts["the_geom"].replace("MULTIPOLYGON ", "", regex=True, inplace=True)
# merged_precincts["the_geom"].replace("\(", "", regex=True, inplace=True)
# merged_precincts["the_geom"].replace("\)", "", regex=True, inplace=True)
merged_precincts_for_covid
```

	the_geom	Precinct	Shape_Leng	Shape_Area	counts	CountDensity
0	MULTIPOLYGON (((-73.97464798076302 40.75336712...	14	20973.820132	2.051113e+07	6810	0.000332
1	MULTIPOLYGON (((-73.9413174228129 40.807713346...	28	17106.149405	1.529402e+07	4206	0.000275
2	MULTIPOLYGON (((-73.93224093714262 40.78969488...	23	19427.884542	2.146668e+07	5401	0.000252
3	MULTIPOLYGON (((-73.93444684733386 40.83599434...	32	25583.751835	2.302456e+07	5286	0.000230
4	MULTIPOLYGON (((-73.89138023291999 40.86170058...	46	29392.712842	3.833342e+07	8775	0.000229
...	...	...	...	...	...	...
72	MULTIPOLYGON (((-74.15945602438066 40.64144833...	121	136811.464647	4.755776e+08	4899	0.000010
73	MULTIPOLYGON (((-73.74461376070957 40.77894737...	111	103218.168638	2.596987e+08	2483	0.000010
74	MULTIPOLYGON (((-74.05050806403247 40.56642203...	122	154842.244366	4.548530e+08	4235	0.000009
75	MULTIPOLYGON (((-73.94923327082499 40.79687199...	22	32712.763701	3.831220e+07	200	0.000005
76	MULTIPOLYGON (((-74.1698258238223 40.561090420...	123	120813.800441	4.608627e+08	2024	0.000004

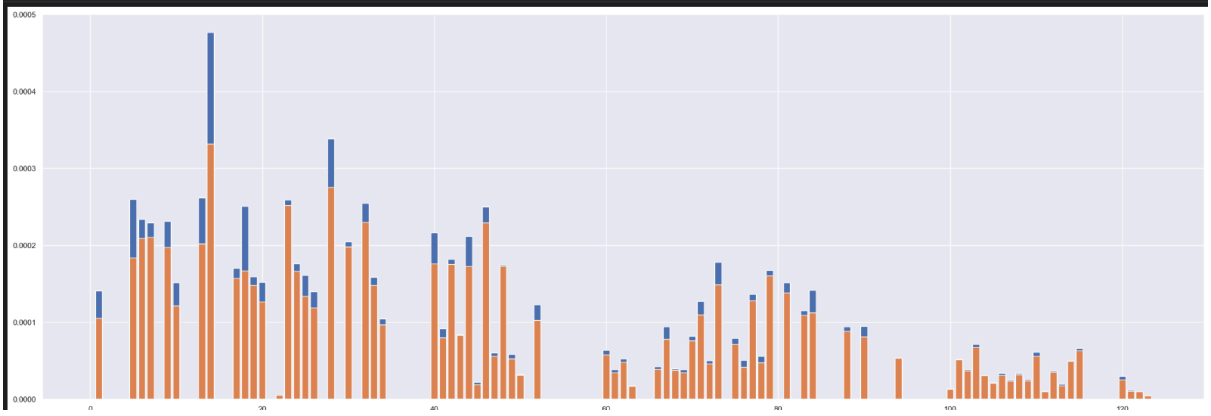
5) We now present some visualizations from our data exploration

a) Crime Density vs Precinct Number

In this histogram, we visualize the change in crime density in each precinct in NYC. We overlapped the crime density count of 2019 and 2020. The blue histogram corresponds to the crime count density in 2020, whereas the orange in 2019. As apparent, every precinct has a decrease in crime count density, supporting our speculation that COVID-19 will have caused a decrease in crime rate. The association also implies that crime rate is dependent on the population density in an area, at least in NYC.

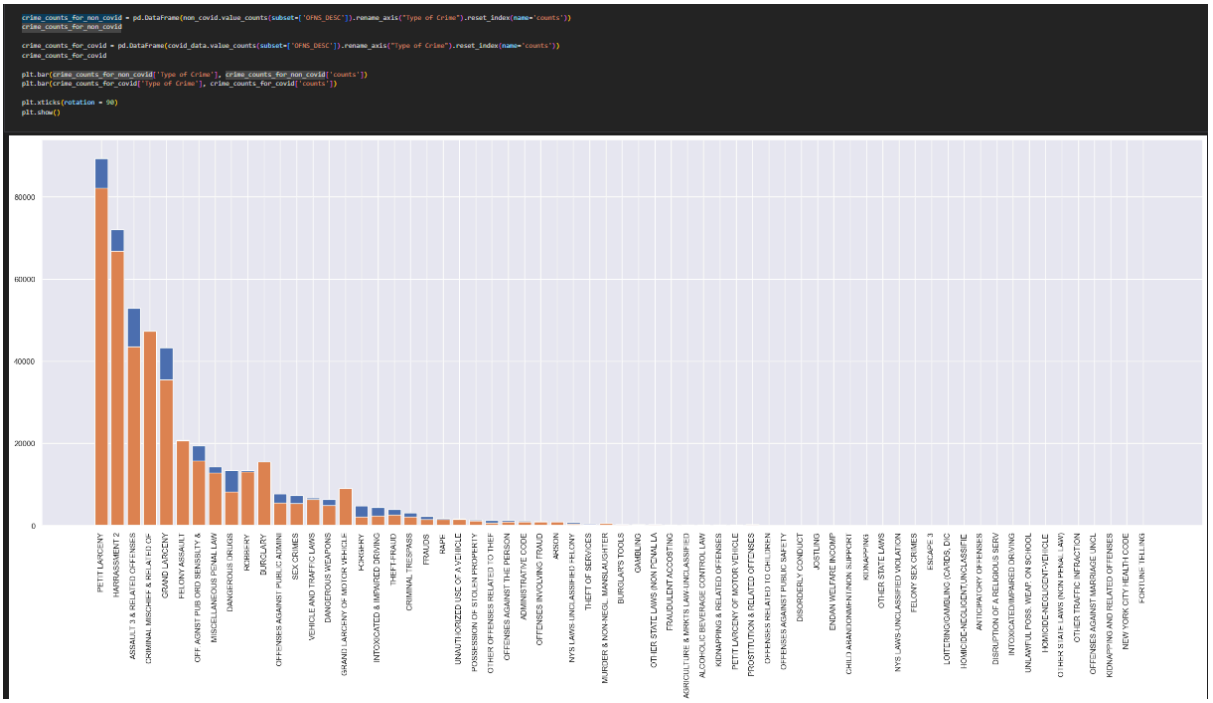
```
plt.bar(merged_precincts_for_non_covid['Precinct'], merged_precincts_for_non_covid['CountDensity'])

plt.ylim(0, 0.0005)
plt.bar(merged_precincts_for_covid['Precinct'], merged_precincts_for_covid['CountDensity'])
plt.show()
```



b) Number of Crimes vs Type of Crime: Similarly, here we compare the change in crime counts for every type of crime from 2019 to 2020. The blue histogram represents the 2019 data, whereas, the orange represents 2020. We

see a decrease in all types of crime, except Burglary and Grand Motor Larceny. We hypothesize that these occurrences too are due to lack of population on the streets during COVID-19. Perhaps, there were more complaints reported for burglary, because of the increased number of people inside homes and the decreased theft opportunities outdoors.



6) Crime cases vs Covid Cases: Using a scatter plot and a line of best fit, we graphed the number of covid cases and crime counts in New York City. To do this, we mined data of the everyday COVID 19 cases in NYC in 2020

```
covid19cases = pd.read_csv("covid19cases.csv")
covid19cases.drop_duplicates()
covid19cases


covid19cases['date_of_interest'] = pd.to_datetime((covid19cases['date_of_interest']))
covid19cases['date_of_interest'] = pd.to_datetime(covid19cases['date_of_interest']).dt.date


covid19cases['year'] = pd.DatetimeIndex(covid19cases['date_of_interest']).year

covid19cases2020= covid19cases.loc[covid19cases['year'] == 2020]
```

We then merged the total number of covid cases in NYC to the total number of crimes in NYC, with respect to date.

```
everydaycasescount = pd.DataFrame(covid_data.value_counts(subset=['RPT_DT']).rename_axis("date_of_interest").reset_index(name='CRIME Cases'))
everydaycasescount

merged_covid = pd.merge(everydaycasescount, covid19cases2020, on=["date_of_interest"])
merged_covid = merged_covid.rename({'CASE_COUNT': 'COVID CASES'}, axis='columns')
merged_covid.head(50)
```

✓ 0.2s

	date_of_interest	CRIME Cases	COVID CASES	PROBABLE_CASE_COUNT	HOSPITALIZED_COUNT	DEATH_COUNT	PROBABLE_DEATH_COUNT	CASE_COUNT_7DAY_AVG	ALL_
0	2020-06-02	1502	570	10	62	37	8	551	
1	2020-03-11	1441	155	0	79	1	0	46	
2	2020-10-21	1429	654	167	53	5	1	538	
3	2020-09-08	1417	393	23	26	2	0	250	
4	2020-10-06	1396	571	72	47	3	1	555	
5	2020-03-04	1380	5	0	2	0	0	0	
6	2020-10-20	1378	601	160	72	5	0	528	
7	2020-10-02	1369	605	82	49	3	0	534	

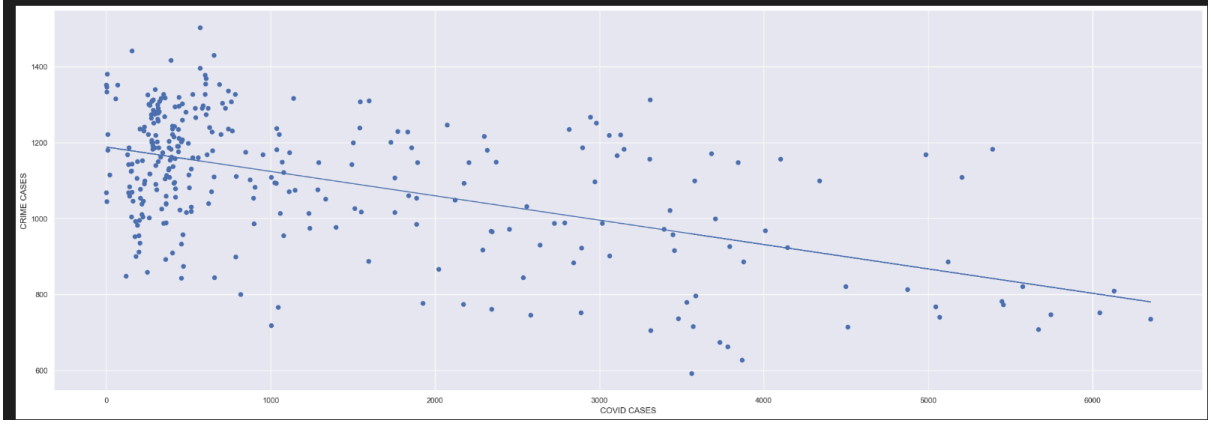
Next, we plot a scatter plot of Covid Cases against Crime Cases in NYC, which clearly shows a negative correlation.

```
x = merged_covid['COVID CASES']
y = merged_covid['CRIME Cases']

a, b = np.polyfit(x, y, 1)
plt.scatter(x, y)
plt.plot(x, a*x+b)
plt.xlabel("COVID CASES")
plt.ylabel("CRIME CASES")
```

✓ 1.4s

```
Text(0, 0.5, 'CRIME CASES')
```



While we took a slight turn in our conclusive report, we have explored new possibilities to create further models predicting crime cases: in this case, Covid-19. Looking at our analysis, we see possibilities of implementing multiple models, perhaps using a Multiple Linear Regression, Random Forest Regression, or Gradient Boost regression. We can test out in the future, and have left our project open to further exploration and extrapolation.

Reference and Data Source

1. Police Department (NYPD, "NYPD Complaint Data Current (Year To Date)," *Cityofnewyork.us*, Oct. 28, 2016. https://data.cityofnewyork.us/Public-Safety/NYPD-Complaint-Data-Current-Year-To-Date-/5uac-w243
2. Police Precincts NYC Open Data *NYC Open Data*, 2018. https://data.cityofnewyork.us/Public-Safety/Police-Precincts/78dh-3ptz

3. Sidewalk Café Licenses and Applications *Cityofnewyork.us*, Mar. 16, 2016. https://data.cityofnewyork.us/Business/Sidewalk-Caf-Licenses-and-Applications/qcdj-rwhu
4. “Public Plazas | NYC Open Data,” *NYC Open Data*, 2018. https://data.cityofnewyork.us/Transportation/Public-Plazas/m4mp-ji5y
5. “NYC Street Centerline (CSCL) | NYC Open Data,” *NYC Open Data*, 2018. https://data.cityofnewyork.us/City-Government/NYC-Street-Centerline-CSCL-/exjm-f27b (accessed Nov. 20, 2022).
6. A. A. Almuhanna, M. M. Alrehili, S. H. Alsubhi, and L. Syed, “Prediction of Crime in Neighbourhoods of New York City using Spatial Data Analysis,” *2021 1st International Conference on Artificial Intelligence and Data Analytics (CAIDA)*, Apr. 2021, doi: 10.1109/caida51941.2021.9425120.
7. V. Ceccato, “Pandemic Restrictions and Spatiotemporal Crime Patterns in New York, São Paulo, and Stockholm - Vania Ceccato, Tulio Kahn, Christopher Herrmann, Anders Östlund, 2022,” *Journal of Contemporary Criminal Justice*, 2022. https://journals.sagepub.com/doi/full/10.1177/10439862211038471#bibr9-10439862211038471