

DATA 606: Capstone Term Paper

# Introduction

The Baltimore Police Agency launched a massive overhaul to its new Records Management Systems in May 2020. This improvement enabled the department to migrate from a paper-based system to a completely digital reporting environment. As a consequence of this major shift, we had significant difficulties in appropriately transferring data from the new records system to the previous Open Data Baltimore system. The "Arrests" dataset is one of many open datasets made publicly accessible by Baltimore's police department on the city's Open Data website (<https://www.baltimorepolice.org/crime-stats/open-data> ). This information is provided to us in an effort to foster greater openness and data exchange between the local administration and the residents of the city. This dataset contains arrest records for offenses such as assault, theft, and property damage in the City of Baltimore. [1]

**Our business question: What methodology that we’ve learned over the course of our graduate student journey will provide the most accurate model in the least amount of time?** To answer this question, we sought to use their database to create a model that can predict which district a crime occurs in based on various details related to the arrest of the perpetrator, such as what he or she was charged with, their gender, etc. Our hope with this model is that the department can then use this model to efficiently spread their resources, tackling the more likely arrests that would be made in a certain district. In addition, we hope that the department can modify their policing efforts to decrease bias in said policing efforts in certain districts (if any are found).

# Accessing the Data

We begin where all great classification problems begin, acquiring the data. Due to different strengths of our computers, we used a mix of anaconda, jupyter lab and google colab to access the data. We used a simple .read\_csv method and checked the general info of the dataframe so we’re aware of what we’re working with.

# Features

The following are a complete list of the raw features at our disposal:

RangeIndex: 175923 entries, 0 to 175922

Data columns (total 20 columns):

| **#** | **Column** | **Non-Null Count** | **Dtype** |
| --- | --- | --- | --- |
| 0 | X | 90128 | float64 |
| 1 | Y | 90128 | float64 |
| 2 | RowID | 175923 | int32 |
| 3 | ArrestNumber | 167697 | float64 |
| 4 | Age | 175836 | float64 |
| 5 | Gender | 175884 | object |
| 6 | Race | 175884 | object |
| 7 | ArrestDateTime | 175923 | object |
| 8 | ArrestLocation | 91488 | object |
| 9 | IncidentOffence | 175923 | object |
| 10 | IncidentLocation | 91488 | object |
| 11 | Charge | 156541 | object |
| 12 | ChargeDescription | 175923 | object |
| 13 | District | 89891 | object |
| 14 | Post | 89891 | object |
| 15 | Neighborhood | 89870 | object |
| 16 | Latitude | 90128 | float64 |
| 17 | Longitude | 90128 | float64 |
| 18 | GeoLocation | 175923 | object |
| 19 | shape | 0 | object |

-----------------------------------------------------------

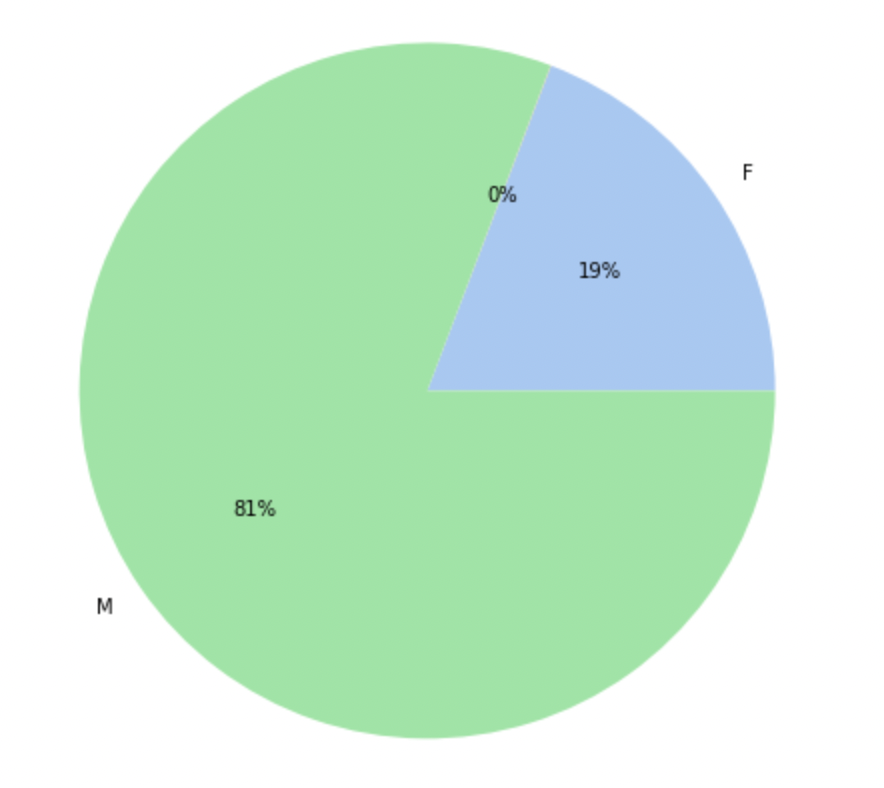
dtypes: float64(6), int32(1), object(13)

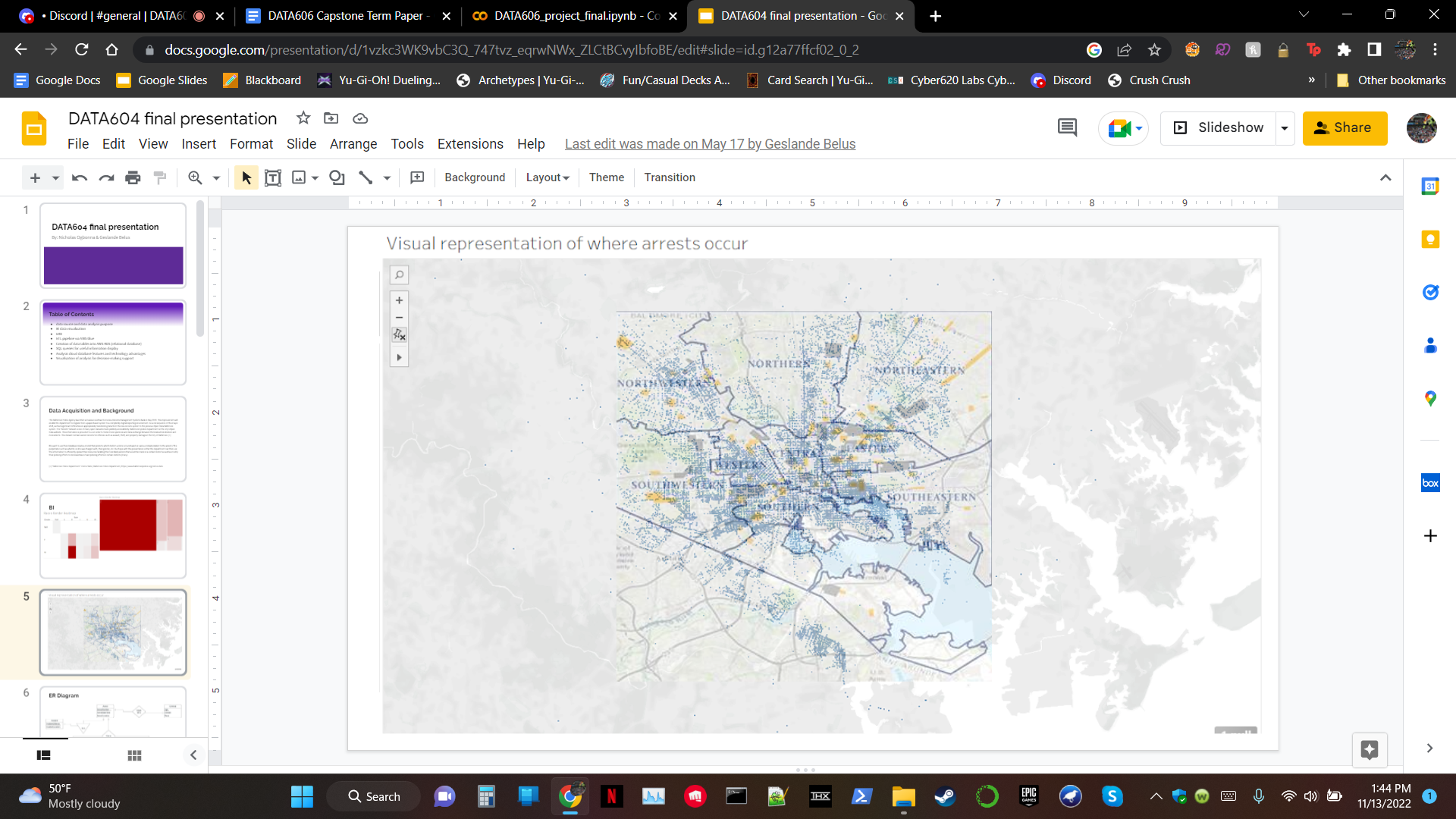
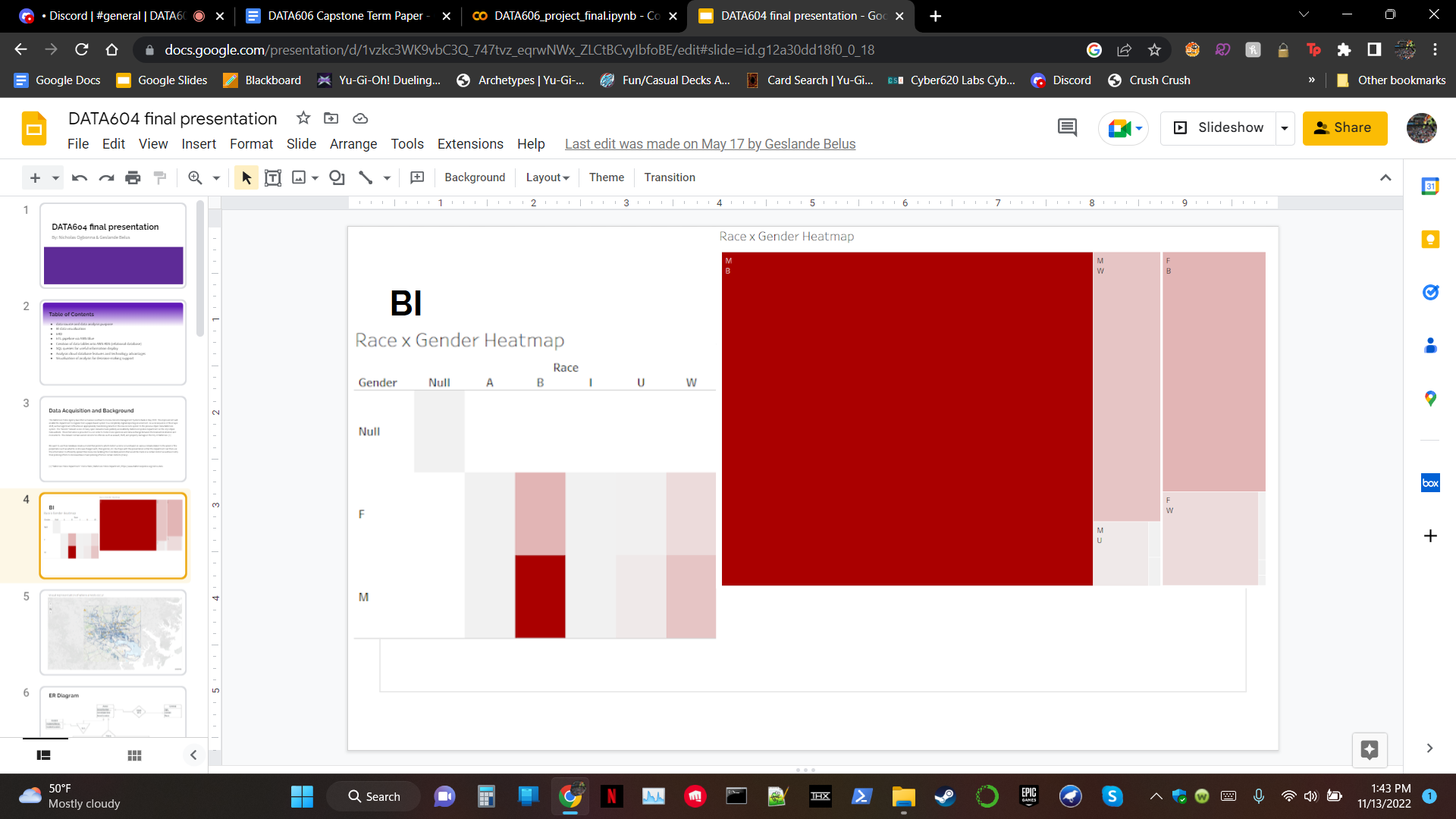
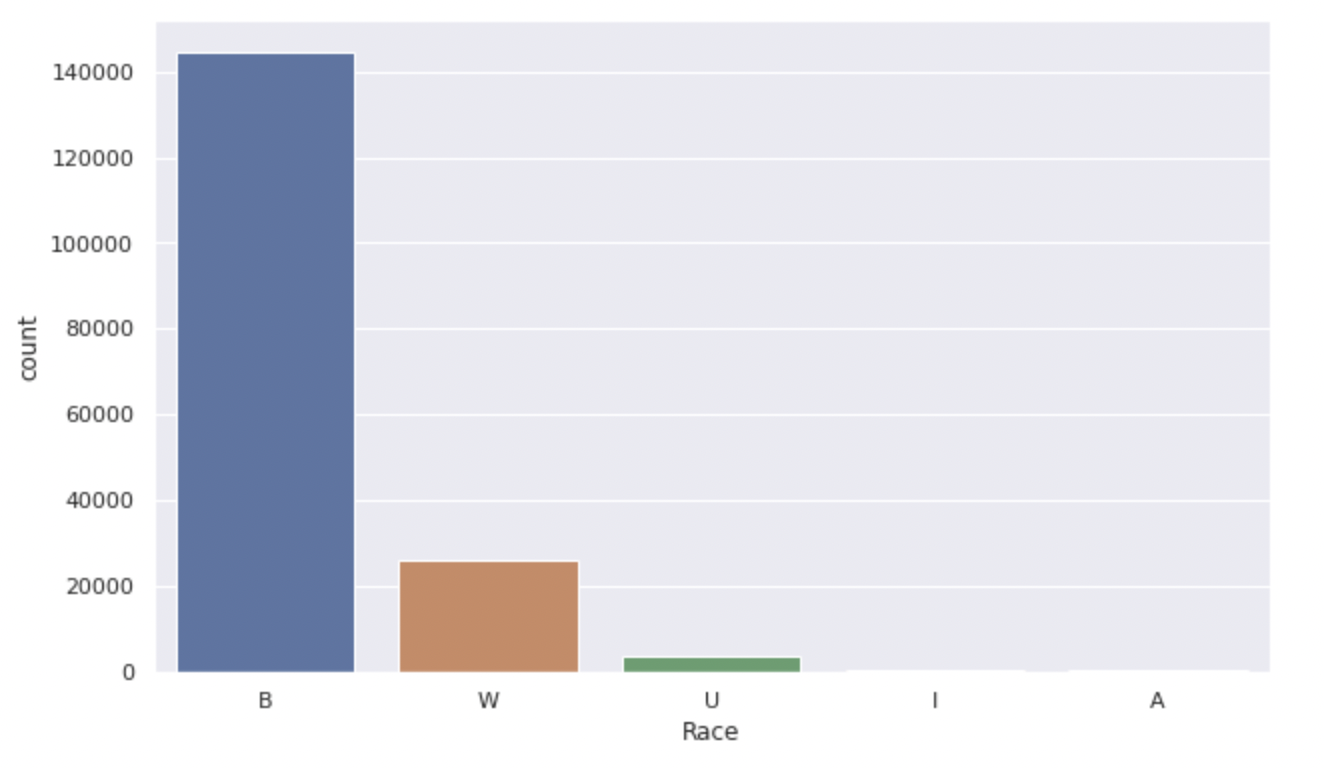
memory usage: 26.2+ MB

# Exploratory Data Analysis (EDA)

Exploratory Data Analysis refers to “the critical process of performing initial investigations on data to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations. Exploratory Data Analysis, or EDA is one of the most important steps in Data Analysis. During this step, the data analyst thoroughly examines the dataset to ensure it is in the form they need for modeling.” [2]





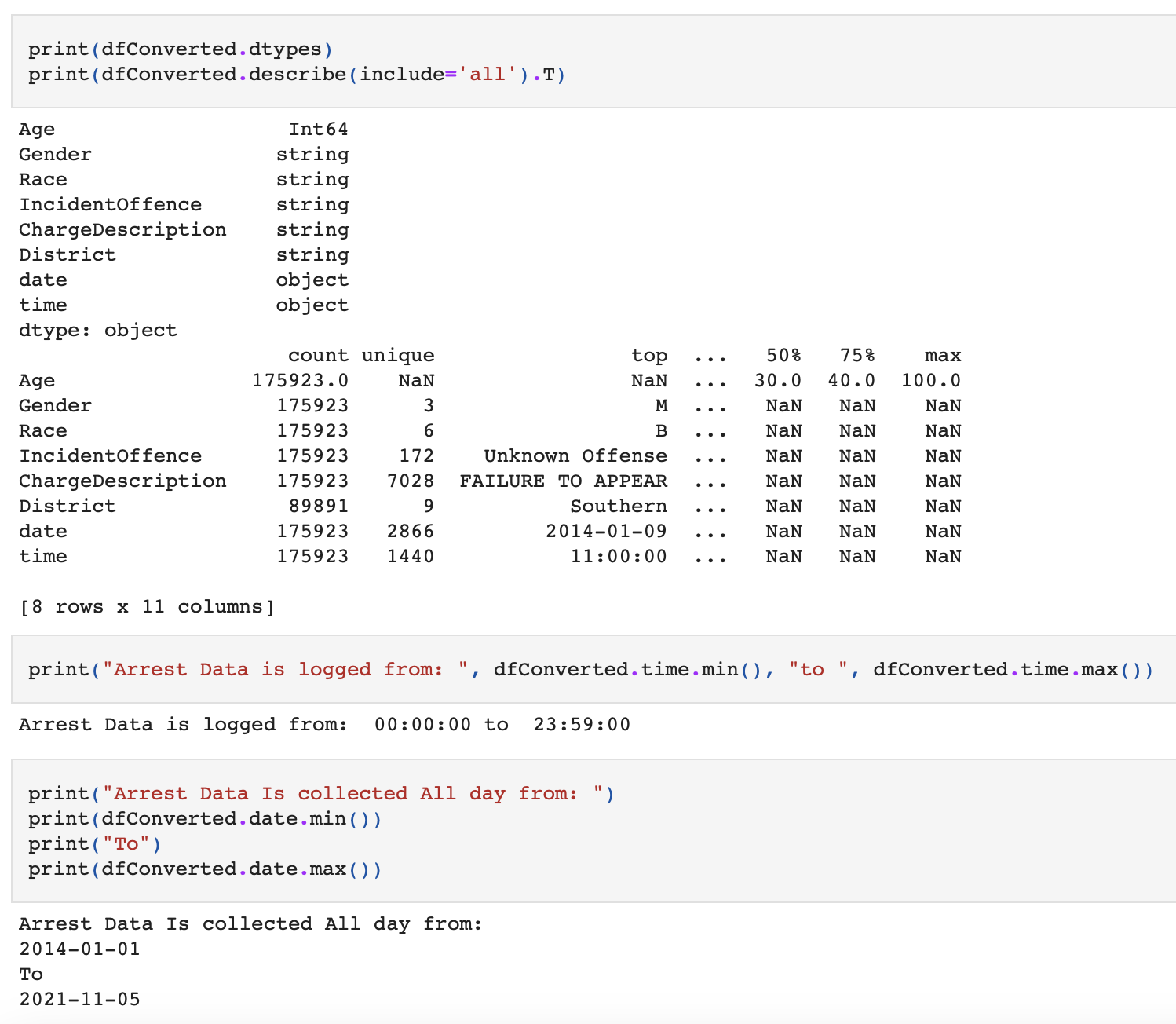


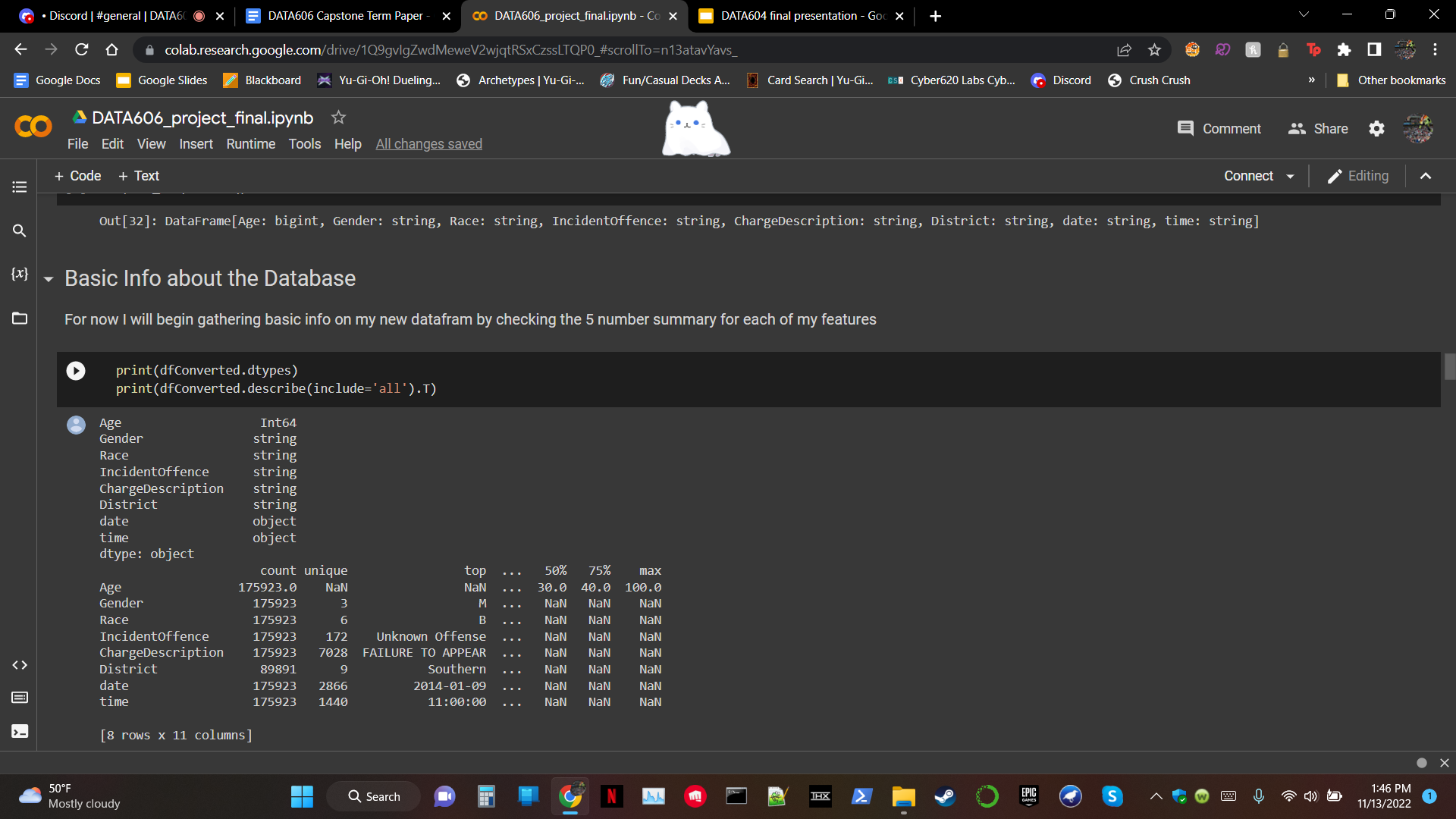
# 

# 

**Dataset basics**

For now we will begin gathering basic info on my new dataframe by checking the 5 number summary for each of my features

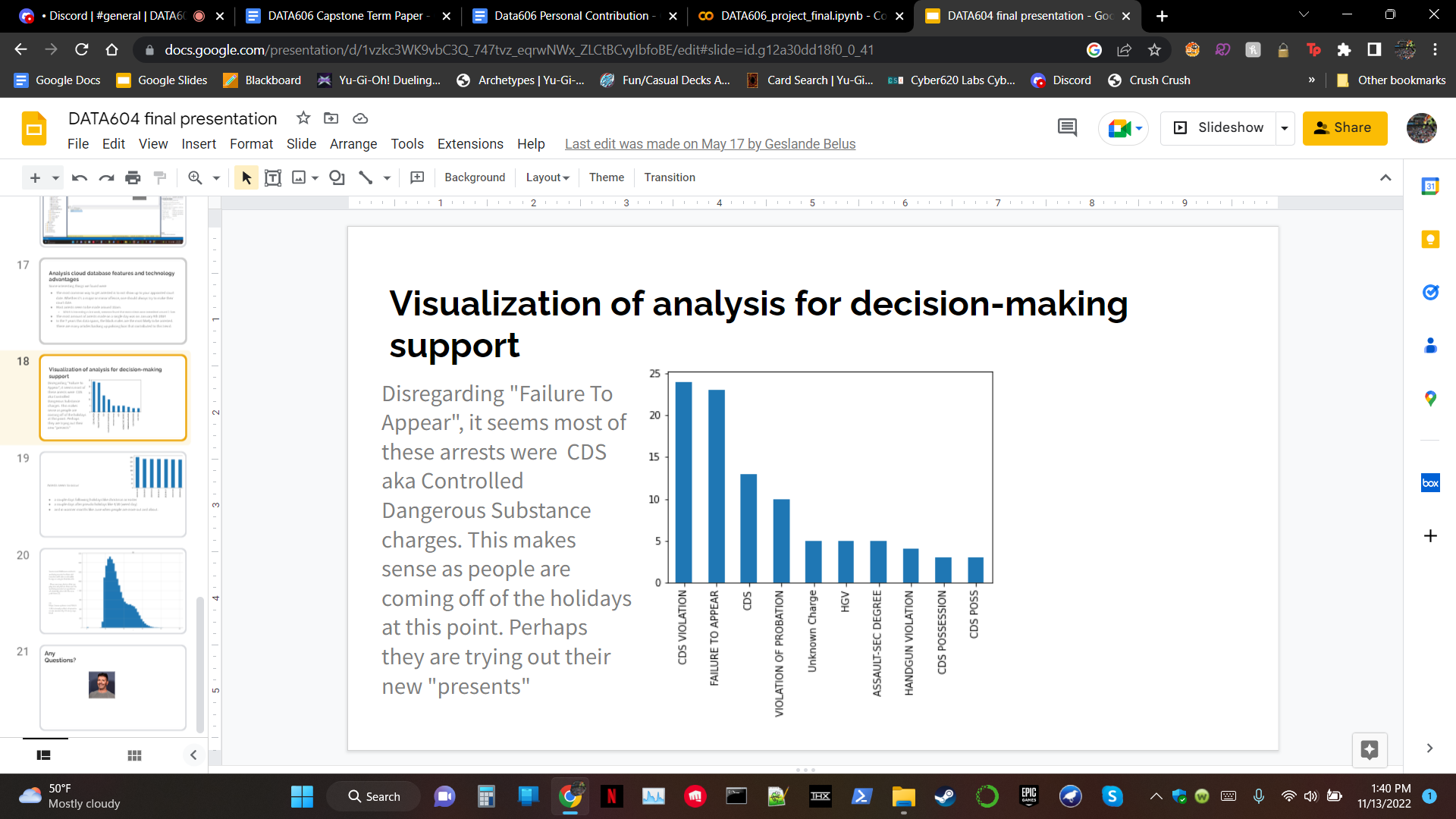


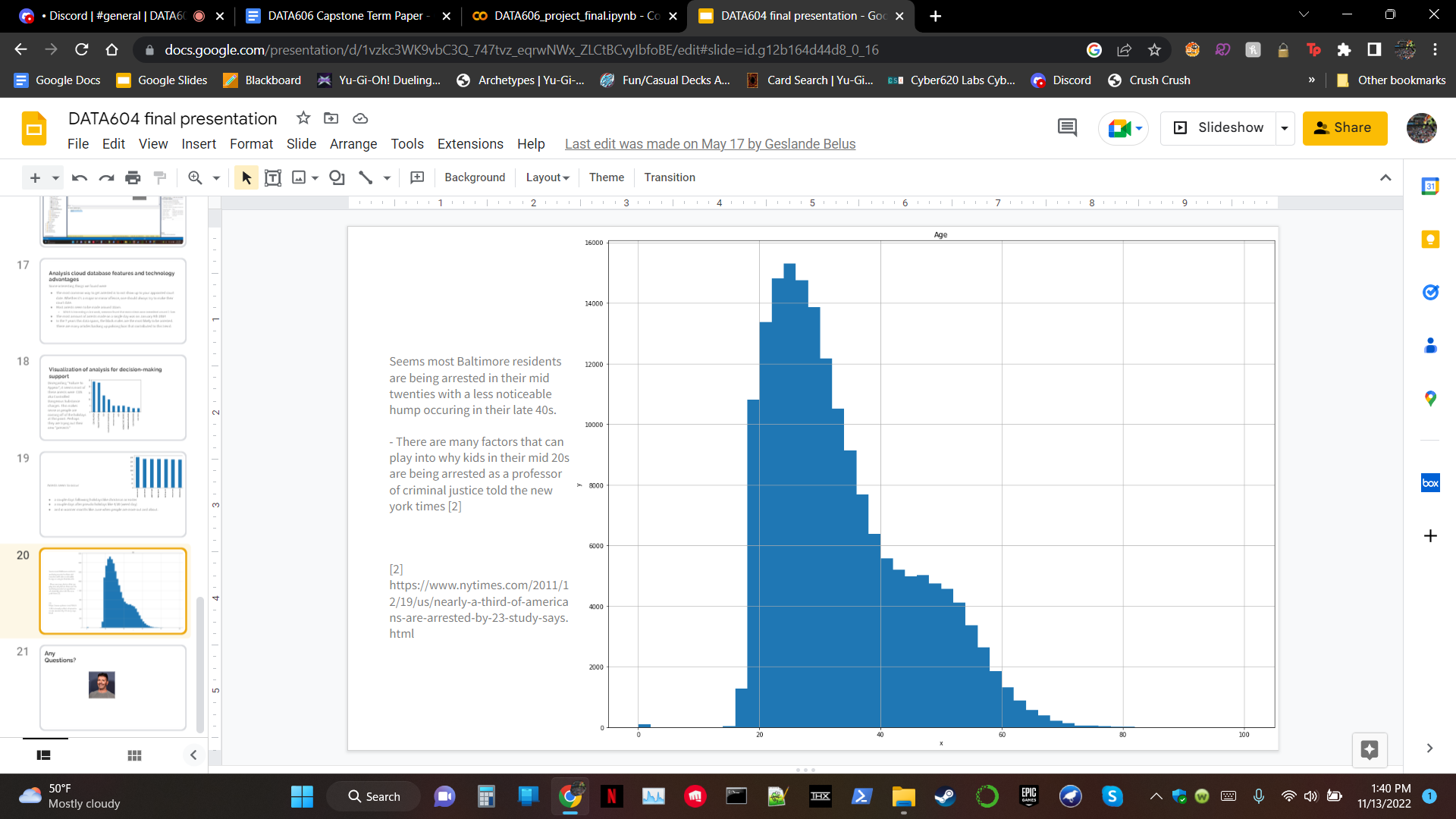
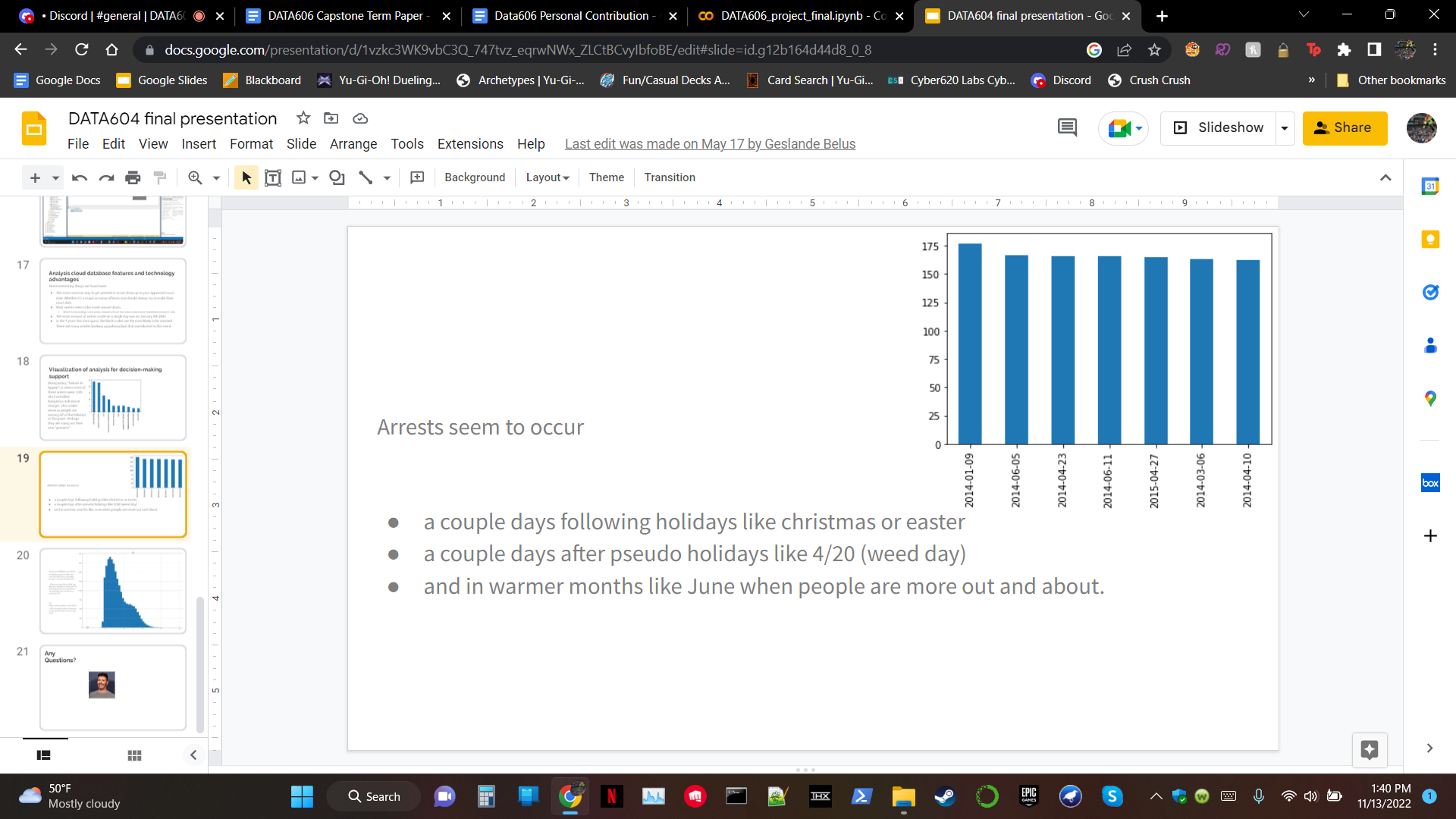


**5 Number Summary Analysis:**

From the 5 number summary alone, we acquired a wealth of information:

* The most frequent reason for being arrested is not showing up to your appointed court date. Whether it’s a major or minor offense, one should always do their best to make their court date.
* The most common time to be arrested is around 11am.
* The highest number of arrests made on a single day was on January 9th, 2014
* In the 7 years this data spans, black males are the most likely to be arrested. Here is an article backing up policing bias that contributes to this trend. We will leave some articles below.
  + https://www.nytimes.com/2011/12/19/us/nearly-a-third-of-americans-are-arrested-by-23-study-says.html





**Multicollinearity check**

Decision trees were one of the methods we wanted to apply, so we needed to make sure that the columns were not highly correlated to each other.

The first thing we checked for is if there is any multicollinearity in the numerical data



We only have 1 numerical variable, so it would make sense that it is fully correlated with itself. Now, how would we check for multicollinearity with our other variables? According to Deepanshu Bhalla of Listen Data [3], "For categorical variables, multicollinearity can be detected with Spearman rank correlation coefficient for ordinal variables and chi-square test for nominal variables."

A nominal variable has no inherent ordering. Race, for example, is a categorical variable because there are 6 categories (W, B, A, I, U, and most recently O) and you can't place them in an order naturally.

A variable with an ordinal value has a distinct ordering. The date and time columns would both be ordinal, as the objects have a clear timeline. i.e. 2014-01-09 comes before 2014-01-10 and 03:40:00 is earlier than 11:00:00

# Modeling, Processing, and Cleaning

In our data, starting 2014-01-01 and ending 2021-11-05, there were multiple features that were not applicable to the classification problem or held the same data as other features, and were dropped as a result. ‘X’, ‘Y’, ‘RowID’ and ‘ArrestNumber’ were all unnecessary indexing variables. Additionally, ‘ArrestLocation’, ‘IncidentLocation’ and ‘Neighborhood’ were dropped, as they provided location information that would cause data leakage. ‘Latitude’, ‘Longitude’ and ‘Geolocation’ were dropped for a similar reason. The ‘Shape’ feature was mostly filled with N/A values, and was not associated with the model in any way. ‘Charge’ holds the same information as ‘Charge Description’, and ‘Post’ is a 3 digit code that reflects the charge, and as such, is redundant. The leftover columns were as follows:

* Age
* Gender
* Race
* IncidentOffence
* ChargeDescription
* ArrestDateTime
* District (the target column)

To start, we put the data in a pyspark dataframe, and created a copy as a pandas dataframe, then dropped each of the features noted above, leaving us with a dataset with dimensions 175923x7. Next the date and time were separated to lower sparsity when using OHE, as there were 45088 non-unique values in the ArrestDateTime column.

Next, N/A values were dealt with. We had 87 in Age, 39 in Gender, 39 in Race, 86032 in District, and none in the other features. Given that District is the target variable, the rows without a given district were dropped. To handle the other N/A values, a ‘U’ was put in place to denote “unknown”, while taking precautions to look for and include any data that used a different marker for an unknown. The Race column was an exception, as an N/A denoted “other”, rather than “unknown”, so an ‘O’ was used in place of a ‘U’.

Before modeling, we built a pipeline to separate the District column, apply the Standard Scaler, one-hot encode the non-numeric features, and impute missing values. We then used train\_test\_split with a test size of .25, splitting the data into 67418 training and 22473 tests.

* To begin, we did logistic regression with grid search, then succeeded in further refining it using the best correlation strength.
* We attempted to use ADABoost, but it resulted in lower accuracy.
* Next, we tried a decision tree classifier, further refining it using max and minimum ranges.
* Our last unique model was SVM, which was refined using a grid search. SVM took an extremely long amount of time to run, but returned with the highest accuracy.
* However, we used pyspark to expand our options in parameters for logistic regression, which resulted in what was by far the highest accuracy model, while not taking too much longer to run than SVM.

# Results and Conclusion

* The Logistic Regression model produced the second highest accuracy (23.6 percent) in around a minute and a half. It appears that attempting to change the Logistic regression resulted in a decrease in accuracy.
* The SVM model produced the highest accuracy (25 percent) in 1483.9 seconds, or a little under 25 minutes. This was after a preliminary grid search in an attempt to determine best parameters (taking nearly 24 hours). As this was already beginning to be prohibitively slow, even on a powerful home PC, further refinement was not considered.
* Since Logistic Regression had such high accuracy without having an extreme runtime, we decided to try it with different parameters using pyspark. This resulted in an accuracy of 50.02 percent in 3146.6 seconds (52 minutes 26 Seconds), more than doubling our previous model accuracy.

# References

[1] “Baltimore Police Department.” Crime Stats | Baltimore Police Department, <https://www.baltimorepolice.org/crime-stats>.

[2] Ijraset. “IJRASET Journal for Research in Applied Science and Engineering Technology.” *An Enhanced Approach of Finding Probability of Road Accidents By Using ANOVA*, https://www.ijraset.com/research-paper/enhanced-approach-of-finding-probability-of-road-accidents.

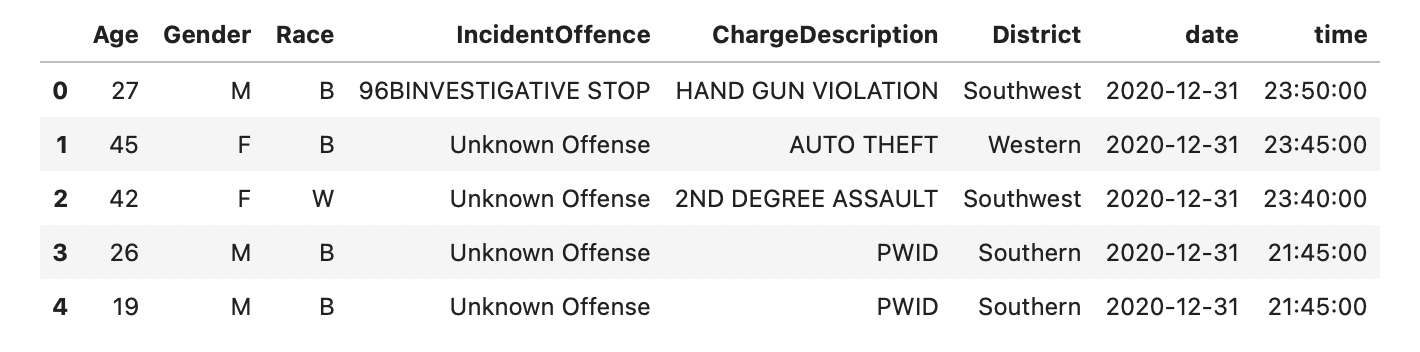
[3] “Detecting Multicollinearity In Categorical Variables, <https://www.listendata.com/2015/04/detecting-multicollinearity-in-categorical-variables.html>

After the initial investigations into the data, the first thing we noticed is that there are some features that will not contribute to our classification problem:

* 'X', 'Y', 'RowID', and 'ArrestNumber': are all similar indexing variables that will provide nothing to the classification
* 'ArrestLocation', 'IncidentLocation', 'Neighborhood': This wouldn't fit my classification problem as my model shouldn't need location information to determine the district the crime happened
* 'Latitude', 'Longitude', and 'GeoLocation' all provide the same information. This wouldn't fit my classification problem as my model shouldn't need location information to determine the district the crime happened
* 'Shape': Shape is a useless feature column that provides no information and is mostly filled with NA values
* Charge, post: Charge and Charge Description hold what is basically the same information. Post is a 3 digit code relating to someone's charge. Thus when I throw them into a OHE they will provide redundancy which is bad.

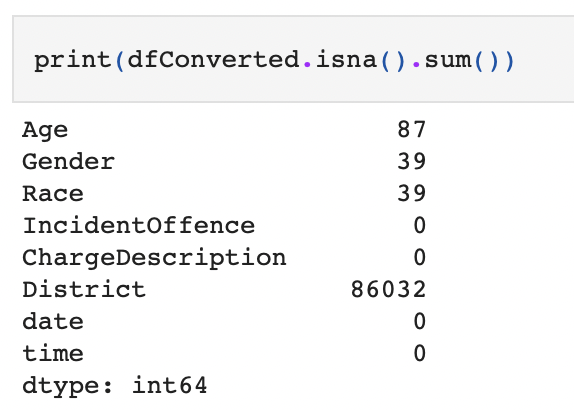
Therefore, I will drop those columns before moving on with the EDA.

For our analysis, it is better to have Date and time of the arrests made in separate columns instead of having a timestamp, so we have split the ArrestDateTime column into two columns and dropped the column. Now we have our final dataframe which we want for our analysis.



**Dealing with missing values (if any):**

Now that I have all the relevant columns for my classification, I need to ensure that the values in said column are not going to cause my modeling pipeline issues. I will begin by first checking how many NA values I have in each column.



We seem to have NA values in Age, Gender, and Race (Ignore District for now). Due to the sheer size of the data, we can fill those with a value to denote that they were "unknown". But if they already have a value denoting "unknown" that would muddy the purity of the data. So before making that modification, We'll check for unique values for gender and race.

* If **age** has no "unknown" value already then substituting 0 for all

Non positive age should suffice

* We will check this by simply checking if all ages are positive numbers
* Gender should have 2 unique values. If it does then we know that NA values are most likely due to the gender being unknown at the time of the arrest according to my father who is a correctional officer
* Similarly we will check the unique values of each of the values in race. If the values all relate to a particular race and/or contain a letter denoting "other race", we will consider my "unknown" hypothesis to be true and will modify the NAs accordingly.

After using appropriate means to handle the NA values in all our columns, we have our final dataframe which we can use for our analysis.