Week 3 Reports

Train Test Split:

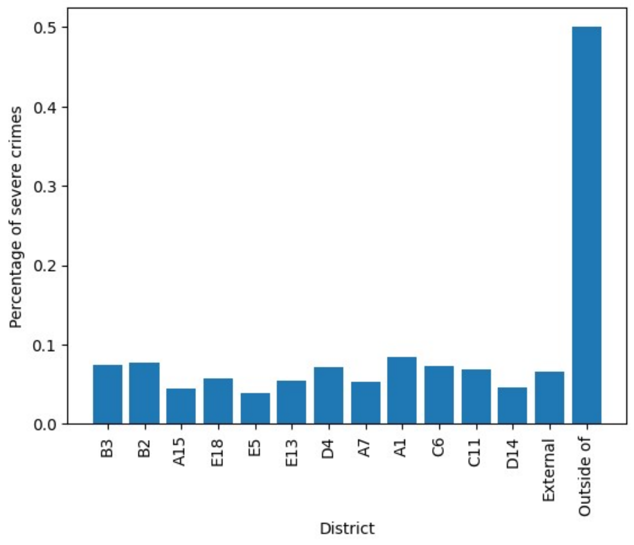
I actually split this dataset into three pieces: Training, Testing, and Validation. Unlike other datasets that is posted by Boston government, since 2024 just started, they also put 2024's first month's data into this data frame, so I split out these data from 2024 as my testing data and will not touch it until the model is fully baked.

For my training and validation data, I use Sklearn's Train_test_split function and use 80-20 portion. The dataset itself is large enough so I am not warried about splitting too much to validation data. 20% of validation data seems enough for me. 80% of the data for training ensures that the model has access to a large enough dataset to learn the underlying patterns. This is important because the performance of machine learning models generally improves with more data, up to a certain point. A larger training set provides a rich variety of examples from which the model can learn, leading to a more accurate and robust model.

EDAs: This would not be a linear regression problem so I did not do any outlier EDAs in my data frame.

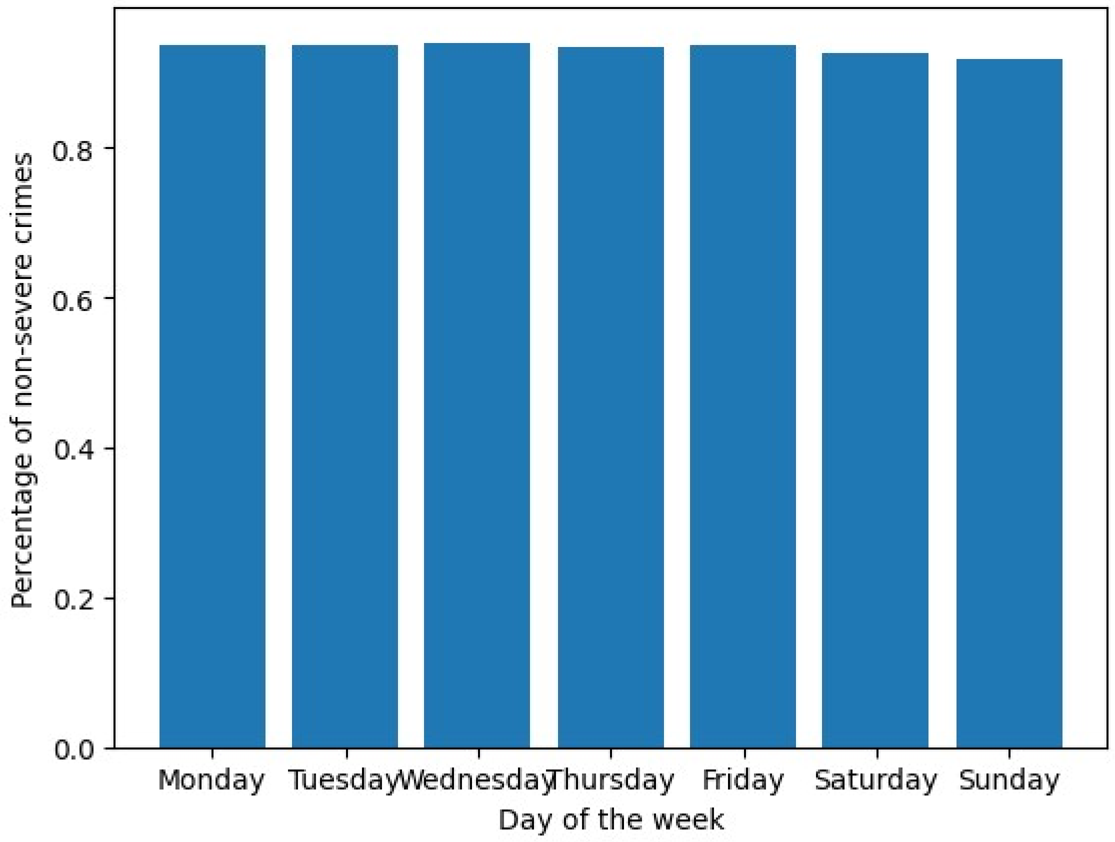
I did one univariate analysis and a lot of bivariate analysis and also a correlation matrix for me to analyses the dataset.

All graphs I did are in the figure folder, and I will show some of these graphs here.



Clearly, there are some district that have higher probability of Severe crimes happening.

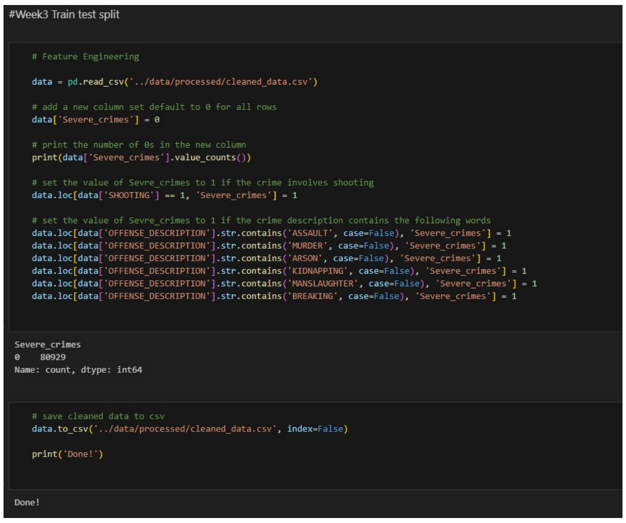
But these graphs show that day of the week does not affect severe crimes to happen:



Problems:

1. Low rate of gun shooting

This problem is found 2 weeks ago but I manage to fix it this week, I create a new column before train test split called severe crimes that makes the portion of target more even.



#Week3 Train test split

```python
# Feature Engineering

data = pd.read_csv('../data/processed/cleaned_data.csv')

# add a new column set default to 0 for all rows
data['Severe_crimes'] = 0

# print the number of 0s in the new column
print(data['Severe_crimes'].value_counts())

# set the value of Sevre_crimes to 1 if the crime involves shooting
data.loc[data['SHOOTING'] == 1, 'Severe_crimes'] = 1

# set the value of Sevre_crimes to 1 if the crime description contains the following words
data.loc[data['OFFENSE_DESCRIPTION'].str.contains('ASSAULT', case=False), 'Severe_crimes'] = 1
data.loc[data['OFFENSE_DESCRIPTION'].str.contains('MURDER', case=False), 'Severe_crimes'] = 1
data.loc[data['OFFENSE_DESCRIPTION'].str.contains('ARSON', case=False), 'Severe_crimes'] = 1
data.loc[data['OFFENSE_DESCRIPTION'].str.contains('KIDNAPPING', case=False), 'Severe_crimes'] = 1
data.loc[data['OFFENSE_DESCRIPTION'].str.contains('MANSLAUGHTER', case=False), 'Severe_crimes'] = 1
data.loc[data['OFFENSE_DESCRIPTION'].str.contains('BREAKING', case=False), 'Severe_crimes'] = 1
```

```
Severe_crimes
0    80929
Name: count, dtype: int64
```

```python
# save cleaned data to csv
data.to_csv('../data/processed/cleaned_data.csv', index=False)

print('Done!')
```

```
Done!
```

2. All variables do not have a strong correlation with my target variable. But I will not use linear regression anyway and there should be a pattern that can be found by models.