In [ ]:
```
# imports
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt

# inport label encoder

from sklearn.preprocessing import LabelEncoder
```

In [ ]:
```
# read in train,test, and val data
train = pd.read_csv('../data/processed/train_data.csv')
test = pd.read_csv('../data/processed/test_data.csv')
val = pd.read_csv('../data/processed/val_data.csv')
```

In [ ]:
```
# show all column names in train data
print(train.columns)
```

```
Index(['_id', 'OFFENSE_CODE', 'OFFENSE_DESCRIPTION', 'DISTRICT',
       'REPORTING_AREA', 'SHOOTING', 'OCCURRED_ON_DATE', 'YEAR', 'MONTH',
       'DAY_OF_WEEK', 'HOUR', 'STREET', 'Severe_crimes'],
      dtype='object')
```

In [ ]:
```
# find the number of missing values in each column
print(train.isnull().sum())
# find the number of blank values in each column
print(train.isna().sum())
```

```
_id                    0
OFFENSE_CODE           0
OFFENSE_DESCRIPTION    0
DISTRICT               0
REPORTING_AREA         0
SHOOTING               0
OCCURRED_ON_DATE       0
YEAR                   0
MONTH                  0
DAY_OF_WEEK            0
HOUR                   0
STREET                 0
Severe_crimes          0
dtype: int64
_id                    0
OFFENSE_CODE           0
OFFENSE_DESCRIPTION    0
DISTRICT               0
REPORTING_AREA         0
SHOOTING               0
OCCURRED_ON_DATE       0
YEAR                   0
MONTH                  0
DAY_OF_WEEK            0
HOUR                   0
STREET                 0
Severe_crimes          0
dtype: int64
```

In [ ]:
```
# find duplicate rows
print(train.duplicated().sum())
```

```
0
```

In [ ]:

```
# remove id column
train = train.drop(columns=['_id'])
# find duplicate rows
print(train.duplicated().sum())
```

```
185
```

In [ ]:

```
# remove duplicate rows
train = train.drop_duplicates()
# find duplicate rows
print(train.duplicated().sum())
```

```
0
```

In [ ]:

```
train.head()
```

Out[ ]:

	OFFENSE_CODE	OFFENSE_DESCRIPTION	DISTRICT	REPORTING_AREA	SHOOTING	OCCURRED_C
0	520	BURGLARY - RESIDENTIAL	C6	194	0	20 08:
1	3821	M/V ACCIDENT - INVOLVING PEDESTRIAN - NO INJURY	E13	303	0	20 18:
2	3114	ASSAULT - SIMPLE	E13	912	1	20 00:
3	3801	M/V ACCIDENT - OTHER	D4	167	0	20 10:
4	3502	MISSING PERSON - LOCATED	E5	691	0	20 13:

In [ ]:

```
# remove columns that are not needed

# remove REPORTING_AREA, SHOOTING
train = train.drop(columns=['REPORTING_AREA', 'SHOOTING'])
train.head()
```

Out[ ]:

	OFFENSE_CODE	OFFENSE_DESCRIPTION	DISTRICT	OCCURRED_ON_DATE	YEAR	MONTH	DAY_
0	520	BURGLARY - RESIDENTIAL	C6	2023-09-05 08:03:00+00	2023	9	
1	3821	M/V ACCIDENT - INVOLVING PEDESTRIAN - NO INJURY	E13	2023-11-13 18:57:00+00	2023	11	
2	3114	ASSAULT - SIMPLE	E13	2023-09-11 00:06:00+00	2023	9	
3	3801	M/V ACCIDENT - OTHER	D4	2023-09-02 10:48:00+00	2023	9	
4	3502	MISSING PERSON - LOCATED	E5	2023-04-27 13:30:00+00	2023	4	

In [ ]:
```
# remove YEAR since its all 2023
train = train.drop(columns=['YEAR'])
```

In [ ]:
```
# check the data types of each column
train.dtypes
```

Out[ ]:
```
OFFENSE_CODE            int64
OFFENSE_DESCRIPTION    object
DISTRICT               object
OCCURRED_ON_DATE       object
MONTH                   int64
DAY_OF_WEEK            object
HOUR                    int64
STREET                 object
Severe_crimes           int64
dtype: object
```

In [ ]:
```
# change OCCURRED_ON_DATE to datetime
train['OCCURRED_ON_DATE'] = pd.to_datetime(train['OCCURRED_ON_DATE'])
```

In [ ]:
```
# change day of week to numbers monday = 0, sunday = 6
train['DAY_OF_WEEK'] = train['OCCURRED_ON_DATE'].dt.dayofweek
train.head()

# remove year from OCCURRED_ON_DATE
train['OCCURRED_ON_DATE'] = train['OCCURRED_ON_DATE'].dt.strftime('%m-%d')
```

In [ ]:
```
train.head()
```

Out[ ]:

	OFFENSE_CODE	OFFENSE_DESCRIPTION	DISTRICT	OCCURRED_ON_DATE	MONTH	DAY_OF_WEE
0	520	BURGLARY - RESIDENTIAL	C6	09-05	9	
1	3821	M/V ACCIDENT - INVOLVING PEDESTRIAN - NO INJURY	E13	11-13	11	
2	3114	ASSAULT - SIMPLE	E13	09-11	9	
3	3801	M/V ACCIDENT - OTHER	D4	09-02	9	
4	3502	MISSING PERSON - LOCATED	E5	04-27	4	

In [ ]:

```python
# Remove spaces in OFFENSE_DESCRIPTION
train['OFFENSE_DESCRIPTION'] = train['OFFENSE_DESCRIPTION'].str.replace(' ', '')
remove = ['-', '(', ')', '/']
for r in remove:
    train['OFFENSE_DESCRIPTION'] = train['OFFENSE_DESCRIPTION'].str.replace(r, '')




train.head()
```

Out[ ]:

	OFFENSE_CODE	OFFENSE_DESCRIPTION	DISTRICT	OCCURRED_ON_DATE	M
0	520	BURGLARYRESIDENTIAL	C6	09-05	
1	3821	MVACCIDENTINVOLVINGPEDESTRIANNOINJURY	E13	11-13	
2	3114	ASSAULTSIMPLE	E13	09-11	
3	3801	MVACCIDENTOTHER	D4	09-02	
4	3502	MISSINGPERSONLOCATED	E5	04-27	

In [ ]:

```python
# do the same for test and val data
test = test.drop(columns=['REPORTING_AREA', 'SHOOTING', 'YEAR'])

val = val.drop(columns=['REPORTING_AREA', 'SHOOTING', 'YEAR'])

# change OCCURRED_ON_DATE to datetime
test['OCCURRED_ON_DATE'] = pd.to_datetime(test['OCCURRED_ON_DATE'])
val['OCCURRED_ON_DATE'] = pd.to_datetime(val['OCCURRED_ON_DATE'])

# change day of week to numbers monday = 0, sunday = 6
test['DAY_OF_WEEK'] = test['OCCURRED_ON_DATE'].dt.dayofweek
val['DAY_OF_WEEK'] = val['OCCURRED_ON_DATE'].dt.dayofweek
```

```
# remove year from OCCURRED_ON_DATE
test['OCCURRED_ON_DATE'] = test['OCCURRED_ON_DATE'].dt.strftime('%m-%d')
val['OCCURRED_ON_DATE'] = val['OCCURRED_ON_DATE'].dt.strftime('%m-%d')

# Remove spaces in OFFENSE_DESCRIPTION
test['OFFENSE_DESCRIPTION'] = test['OFFENSE_DESCRIPTION'].str.replace(' ', '')
val['OFFENSE_DESCRIPTION'] = val['OFFENSE_DESCRIPTION'].str.replace(' ', '')

for r in remove:
    test['OFFENSE_DESCRIPTION'] = test['OFFENSE_DESCRIPTION'].str.replace(r, '')
    val['OFFENSE_DESCRIPTION'] = val['OFFENSE_DESCRIPTION'].str.replace(r, '')

# check the data types of each column
train.dtypes
```

Out[ ]:

```
OFFENSE_CODE            int64
OFFENSE_DESCRIPTION    object
DISTRICT               object
OCCURRED_ON_DATE       object
MONTH                   int64
DAY_OF_WEEK             int32
HOUR                    int64
STREET                 object
Severe_crimes           int64
dtype: object
```

In [ ]:

```
# remove street column
train = train.drop(columns=['STREET'])
test = test.drop(columns=['STREET'])
val = val.drop(columns=['STREET'])
```

In [ ]:

```
# encode all non-numeric columns
# reset the label encoder
le_description = LabelEncoder()
le_district = LabelEncoder()


# fit and transform the label encoder
train['OFFENSE_DESCRIPTION'] = le_description.fit_transform(train['OFFENSE_DESCRIPT
train['DISTRICT'] = le_district.fit_transform(train['DISTRICT'])
```

In [ ]:

```
train.head()
```

Out[ ]:

	OFFENSE_CODE	OFFENSE_DESCRIPTION	DISTRICT	OCCURRED_ON_DATE	MONTH	DAY_OF_WEE
0	520	15	6	09-05	9	
1	3821	69	9	11-13	11	
2	3114	6	9	09-11	9	
3	3801	70	8	09-02	9	
4	3502	62	11	04-27	4	

In [ ]:

```
# do the same for test and val data
# add new label if the value is unseen in the training data
test['OFFENSE_DESCRIPTION'] = test['OFFENSE_DESCRIPTION'].map(lambda s: '<unknown>'
val['OFFENSE_DESCRIPTION'] = val['OFFENSE_DESCRIPTION'].map(lambda s: '<unknown>' i

# add new label if the value is unseen in the training data
test['DISTRICT'] = test['DISTRICT'].map(lambda s: '<unknown>' if s not in le_distri
```

```python
val['DISTRICT'] = val['DISTRICT'].map(lambda s: '<unknown>' if s not in le_district

# add <unknown> to the classes
le_description.classes_ = np.append(le_description.classes_, '<unknown>')
le_district.classes_ = np.append(le_district.classes_, '<unknown>')



# transform the label encoder
test['OFFENSE_DESCRIPTION'] = le_description.transform(test['OFFENSE_DESCRIPTION'])
val['OFFENSE_DESCRIPTION'] = le_description.transform(val['OFFENSE_DESCRIPTION'])

test['DISTRICT'] = le_district.transform(test['DISTRICT'])
val['DISTRICT'] = le_district.transform(val['DISTRICT'])

# check the data types of each column
train.dtypes
```

Out[ ]:

```
OFFENSE_CODE            int64
OFFENSE_DESCRIPTION     int32
DISTRICT                int32
OCCURRED_ON_DATE       object
MONTH                   int64
DAY_OF_WEEK             int32
HOUR                    int64
Severe_crimes           int64
dtype: object
```

In [ ]:

```python

```

In [ ]:

```python
# show number of values in each column
train.nunique()
# show number of 1 and 0 in the Severe_crimes column
train['Severe_crimes'].value_counts()
```

Out[ ]:

```
Severe_crimes
0    57521
1     4171
Name: count, dtype: int64
```

In [ ]:

```python
# save the processed data
train.to_csv('../data/processed/train_data_processed.csv', index=False)
test.to_csv('../data/processed/test_data_processed.csv', index=False)
val.to_csv('../data/processed/val_data_processed.csv', index=False)
```

WEEK 5

in week 4 we did encoding of all object varible, The reason i combine week4 and week5 is to fix the problem in week4's code

In [ ]:

```python
# show number of unique values in each column
train.nunique()
```

Out[ ]:

```
OFFENSE_CODE           116
OFFENSE_DESCRIPTION    117
DISTRICT                14
OCCURRED_ON_DATE       365
MONTH                   12
DAY_OF_WEEK              7
HOUR                    24
Severe_crimes            2
dtype: int64
```

New features are already created within earlier notebooks that the column severe crime is the new feature.

In [ ]:
```
# add synthetic data to the training data
# add 1000 rows of synthetic data
synthetic_data = train.sample(n=1000, replace=True)
train = pd.concat([train, synthetic_data])

# show number of unique values in each column
train.nunique()
```

Out[ ]:
```
OFFENSE_CODE           116
OFFENSE_DESCRIPTION    117
DISTRICT                14
OCCURRED_ON_DATE       365
MONTH                   12
DAY_OF_WEEK              7
HOUR                    24
Severe_crimes            2
dtype: int64
```

In [ ]:
```
# count the number of 1 and 0 in the Severe_crimes column
train['Severe_crimes'].value_counts()
```

Out[ ]:
```
Severe_crimes
0    58454
1     4238
Name: count, dtype: int64
```