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())
```

```
OFFENSE_CODE           0
OFFENSE_DESCRIPTION    0
DISTRICT               0
OCCURRED_ON_DATE       0
MONTH                  0
DAY_OF_WEEK            0
HOUR                   0
STREET                 0
Severe_crimes          0
dtype: int64
OFFENSE_CODE           0
OFFENSE_DESCRIPTION    0
DISTRICT               0
OCCURRED_ON_DATE       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 18:5
2	3114	INVESTIGATE PROPERTY	E13	912	1	20 00:(
3	3801	M/V ACCIDENT - OTHER	D4	167	0	20 10:4
4	3502	MISSING PERSON - LOCATED	E5	691	0	20 13:3

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	INVESTIGATE PROPERTY	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	INVESTIGATE PROPERTY	E13	09-11	9	
3	3801	M/V ACCIDENT - OTHER	D4	09-02	9	
4	3502	MISSING PERSON - LOCATED	E5	04-27	4	

In [ ]:
```
# encode all non-numeric columns
le = LabelEncoder()


# encode OFFENSE_DESCRIPTION
train['OFFENSE_DESCRIPTION'] = le.fit_transform(train['OFFENSE_DESCRIPTION'])

# encode DISTRICT
train['DISTRICT'] = le.fit_transform(train['DISTRICT'])

# encode street
train['STREET'] = le.fit_transform(train['STREET'])

# show the first 5 rows of the train data
train.head()

# save the encoder settings ( so we can use it later to decode the data or use it c
import joblib
joblib.dump(le, '../models/label_encoder.pkl')
```

Out[ ]:
```
['../models/label_encoder.pkl']
```

In [ ]:
```
# 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    57263
1     4164
Name: count, dtype: int64
```

In [ ]:
```
# do the same for test and val data
test = test.drop(columns=['_id', 'REPORTING_AREA', 'SHOOTING', 'YEAR'])
test['OCCURRED_ON_DATE'] = pd.to_datetime(test['OCCURRED_ON_DATE'])
test['DAY_OF_WEEK'] = test['OCCURRED_ON_DATE'].dt.dayofweek
test['OCCURRED_ON_DATE'] = test['OCCURRED_ON_DATE'].dt.strftime('%m-%d')
test['OFFENSE_DESCRIPTION'] = le.transform(test['OFFENSE_DESCRIPTION'])
test['DISTRICT'] = le.transform(test['DISTRICT'])
test['STREET'] = le.transform(test['STREET'])

val = val.drop(columns=['_id', 'REPORTING_AREA', 'SHOOTING', 'YEAR'])
val['OCCURRED_ON_DATE'] = pd.to_datetime(val['OCCURRED_ON_DATE'])
val['DAY_OF_WEEK'] = val['OCCURRED_ON_DATE'].dt.dayofweek
val['OCCURRED_ON_DATE'] = val['OCCURRED_ON_DATE'].dt.strftime('%m-%d')
val['OFFENSE_DESCRIPTION'] = le.transform(val['OFFENSE_DESCRIPTION'])
val['DISTRICT'] = le.transform(val['DISTRICT'])
val['STREET'] = le.transform(val['STREET'])

# 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)
```