In [ ]:

```
# imports
import os
import pandas as pd
import numpy as np

# load data
train = pd.read_csv('../data/processed/train_data_processed.csv')
test = pd.read_csv('../data/processed/test_data_processed.csv')
val = pd.read_csv('../data/processed/val_data_processed.csv')
```

In [ ]:

```
# more feature engineering
# use encoder to encode OCCURRED_ON_DATE column
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
train['OCCURRED_ON_DATE'] = le.fit_transform(train['OCCURRED_ON_DATE'])
test['OCCURRED_ON_DATE'] = le.transform(test['OCCURRED_ON_DATE'])
val['OCCURRED_ON_DATE'] = le.transform(val['OCCURRED_ON_DATE'])
```

In [ ]:

```
# save le
import joblib
joblib.dump(le, '../models/datetime_encoder.pkl')
```

Out[ ]:

```
['../models/datetime_encoder.pkl']
```

In [ ]:

```
#drop _id column

test = test.drop('_id', axis=1)
val = val.drop('_id', axis=1)
```

In [ ]:

```
# define the target variable
y_train = train['Severe_crimes']
y_test = test['Severe_crimes']
y_val = val['Severe_crimes']

# define the features
X_train = train.drop(['Severe_crimes'], axis=1)
X_test = test.drop(['Severe_crimes'], axis=1)
X_val = val.drop(['Severe_crimes'], axis=1)
```

In [ ]:

```
# build a random forest model which we selected as the best model from last week
from sklearn.ensemble import RandomForestClassifier
# the random forest model will have 1000 trees and a max depth of 10
rf = RandomForestClassifier(n_estimators=1000, max_depth=10, random_state=42)
# fit the model
rf.fit(X_train, y_train)

# evaluate the model
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_sc
# make predictions
y_pred = rf.predict(X_val)
# calculate the evaluation metrics
accuracy = accuracy_score(y_val, y_pred)
precision = precision_score(y_val, y_pred)
recall = recall_score(y_val, y_pred)
f1 = f1_score(y_val, y_pred)

# print the evaluation metrics
```

```
print('Accuracy: ', accuracy)
print('Precision: ', precision)
print('Recall: ', recall)
print('F1: ', f1)
```

```
Accuracy:  0.9943762120232709
Precision:  0.9923076923076923
Recall:  0.9186164801627671
F1:  0.954041204437401
```

In [ ]:

```
# add bootstrapping to the model
from sklearn.utils import resample
# define the number of bootstraps
n_bootstraps = 100
# create empty lists to store the evaluation metrics
accuracy_scores = []
precision_scores = []
recall_scores = []
f1_scores = []
# loop through the number of bootstraps
for i in range(n_bootstraps):
    # resample the data
    X_resampled, y_resampled = resample(X_train, y_train, random_state=i)
    # fit the model
    rf.fit(X_resampled, y_resampled)
    # make predictions
    y_pred = rf.predict(X_val)
    # calculate the evaluation metrics
    accuracy = accuracy_score(y_val, y_pred)
    precision = precision_score(y_val, y_pred)
    recall = recall_score(y_val, y_pred)
    f1 = f1_score(y_val, y_pred)
    # append the evaluation metrics to the lists
    accuracy_scores.append(accuracy)
    precision_scores.append(precision)
    recall_scores.append(recall)
    f1_scores.append(f1)

# calculate the mean and standard deviation of the evaluation metrics
accuracy_mean = np.mean(accuracy_scores)
accuracy_std = np.std(accuracy_scores)
precision_mean = np.mean(precision_scores)

precision_std = np.std(precision_scores)
recall_mean = np.mean(recall_scores)
recall_std = np.std(recall_scores)
f1_mean = np.mean(f1_scores)
f1_std = np.std(f1_scores)

# print the evaluation metrics
```

In [ ]:

```
# print the evaluation metrics
print('Accuracy: ', accuracy_mean, '+/-', accuracy_std)
print('Precision: ', precision_mean, '+/-', precision_std)
print('Recall: ', recall_mean, '+/-', recall_std)
print('F1: ', f1_mean, '+/-', f1_std)
```

```
Accuracy:  0.9942533936651584 +/- 0.0001544641647750288
Precision:  0.9932187827422044 +/- 0.0014498888540798896
Recall:  0.9158189216683621 +/- 0.002673664618708329
F1:  0.9529464309013826 +/- 0.001320692043123352
```

In [ ]:

```python
# get the more important features list
importances = rf.feature_importances_
# sort the importances in descending order
indices = np.argsort(importances)[::-1]
# get the feature names
features = X_train.columns

# show the list of features and their importances
for i in range(X_train.shape[1]):
    print(f'{features[indices[i]]}: {importances[indices[i]]}')
```

```
OFFENSE_DESCRIPTION: 0.7410702456668807
OFFENSE_CODE: 0.2352010942417921
HOUR: 0.008336455421575177
OCCURRED_ON_DATE: 0.005759008230516233
DISTRICT: 0.004799620122936456
DAY_OF_WEEK: 0.002475637220088287
MONTH: 0.0023579390962110186
```