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

In [ ]:
```
# load the model
import joblib
model = joblib.load('../models/rf_model_week10.pkl')
```

In [ ]:
```
# load the test data
test = pd.read_csv('../data/processed/test_data_processed.csv')

# load the label encoder
from sklearn.preprocessing import LabelEncoder
le_datetime = joblib.load('../models/datetime_encoder.pkl')

# use the label encoder to transform the datetime column
test['OCCURRED_ON_DATE'] =  le_datetime.transform(test['OCCURRED_ON_DATE'])
```

In [ ]:
```
# show test data first
print(test.head())
```

```
     _id  OFFENSE_CODE  OFFENSE_DESCRIPTION  DISTRICT  OCCURRED_ON_DATE  \
0  20848           801                    6         0                 0
1  20849          3018                  100         0                 0
2  20851           801                    6         0                 0
3  20852          3410                  105         5                 0
4  20854           724                    7         9                 1

   MONTH  DAY_OF_WEEK  HOUR  Severe_crimes
0      1            0     0              1
1      1            0     2              0
2      1            0    11              1
3      1            0    11              0
4      1            1     0              0
```

In [ ]:
```
# remove _id column
test = test.drop('_id', axis=1)
test.head()
```

Out[ ]:

	OFFENSE_CODE	OFFENSE_DESCRIPTION	DISTRICT	OCCURRED_ON_DATE	MON
0	801	6	0	0	
1	3018	100	0	0	
2	801	6	0	0	
3	3410	105	5	0	
4	724	7	9	1	

In [ ]:
```
# choose random 5 sample from the test data
test_sample = test.sample(5, random_state=3331) # set random state for reproduci
test_sample
```

Out[ ]:

	OFFENSE_CODE	OFFENSE_DESCRIPTION	DISTRICT	OCCURRED_ON_DATE	M
508	3207	85	8	1	
969	3802	74	4	3	
377	613	51	8	0	
2788	801	6	3	14	
2083	3201	86	10	10	

In [ ]:

```
# remove the target column from the test sample
test_sample = test_sample.drop('Severe_crimes', axis=1)
test_sample
```

Out[ ]:

	OFFENSE_CODE	OFFENSE_DESCRIPTION	DISTRICT	OCCURRED_ON_DATE	M
508	3207	85	8	1	
969	3802	74	4	3	
377	613	51	8	0	
2788	801	6	3	14	
2083	3201	86	10	10	

In [ ]:

```
# use the model to make predictions
predictions = model.predict(test_sample)
predictions
```

Out[ ]:

```
array([0, 0, 0, 1, 0], dtype=int64)
```

In [ ]:

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

# print the feature importance
print("Feature ranking:")
for f in range(test_sample.shape[1]):
    print("%d. feature %s (%f)" % (f + 1, features[indices[f]], importances[indi
```

```
Feature ranking:
1. feature OFFENSE_DESCRIPTION (0.741070)
2. feature OFFENSE_CODE (0.235201)
3. feature HOUR (0.008336)
4. feature OCCURRED_ON_DATE (0.005759)
5. feature DISTRICT (0.004800)
6. feature DAY_OF_WEEK (0.002476)
7. feature MONTH (0.002358)
```

In [ ]:

```
test_target = test['Severe_crimes']
test_features = test.drop('Severe_crimes', axis=1)
```

In [ ]:
```
# predict the full test data
predictions = model.predict(test_features)
```

In [ ]:
```
# evaluate the model
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(test_target, predictions)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
Accuracy: 99.50%
```

In [ ]:
```
# re do the model training process

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

# use the label encoder to transform the datetime column
train['OCCURRED_ON_DATE'] =  le_datetime.transform(train['OCCURRED_ON_DATE'])
val['OCCURRED_ON_DATE'] =  le_datetime.transform(val['OCCURRED_ON_DATE'])

# remove _id column

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

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

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

In [ ]:
```
# remove the protected features to avoid bias
X_train = X_train.drop('DISTRICT', axis=1)
X_val = X_val.drop('DISTRICT', axis=1)
```

In [ ]:
```
# train the model
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=1000, max_depth=10, random_state=42)

rf.fit(X_train, y_train)
```

Out[ ]:
```
▾                    RandomForestClassifier
RandomForestClassifier(max_depth=10, n_estimators=1000, random_state=4
2)
```

In [ ]:
```
# use this model to make predictions
test_features = test_features.drop('DISTRICT', axis=1)
predictions = rf.predict(test_features)

# evaluate the model
accuracy = accuracy_score(test_target, predictions)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
Accuracy: 99.55%
```

In [ ]:

```
#merge the test data with the predictions
test['predictions'] = predictions
test.head()
```

Out[ ]:

	OFFENSE_CODE	OFFENSE_DESCRIPTION	DISTRICT	OCCURRED_ON_DATE	MON
0	801	6	0	0	
1	3018	100	0	0	
2	801	6	0	0	
3	3410	105	5	0	
4	724	7	9	1	

In [ ]:

```
# load other encoders
le_district = joblib.load('../models/le_district.pkl')
le_description = joblib.load('../models/le_description.pkl')

# use these encoders to transform the columns back to their original values of t
test['DISTRICT'] = le_district.inverse_transform(test['DISTRICT'])
test['OFFENSE_DESCRIPTION'] = le_description.inverse_transform(test['OFFENSE_DES
test['OCCURRED_ON_DATE'] = le_datetime.inverse_transform(test['OCCURRED_ON_DATE'

test.head()
```

Out[ ]:

	OFFENSE_CODE	OFFENSE_DESCRIPTION	DISTRICT	OCCURRED_ON_DATE	M
0	801	ASSAULTSIMPLE	A1	01-01	
1	3018	SICKINJUREDMEDICALPOLICE	A1	01-01	
2	801	ASSAULTSIMPLE	A1	01-01	
3	3410	TOWEDMOTORVEHICLE	C11	01-01	
4	724	AUTOTHEFT	E13	01-02	