Machine Learning Engineer Nanodegree

Capstone Project

Wesley Strange
March 28th, 2019

I. Definition

Project Overview

For my final project I will be using Chicago Crime data from years past to create a Machine Learning model to predict when violent crimes are most likely to happen. I believe the Police Department could benefit from this model by being able to better allocate their resources in attempt to prevent some of the more violent crimes that are being committed.

I live in the Northwest Suburbs of Chicago, so I would personally like to see the crime rates in Chicago reduced. I would like to feel safe when I'm out exploring the many beautiful sights that Chicago has to offer. I also chose this domain since Chicago has received a lot of negative media coverage in recent years due to the high number of murders and gun violence. According to the FBI crime dataset, in recent years Chicago has experience violent crime rates nearly three times higher than the national average (Uniform Crime Reporting (UCR) Program, 2019).

This type of problem has been solved before using Machine Learning. In the article "Once Upon a Crime: Towards Crime Prediction from Demographics and Mobile Data" they attempted to predict crime using a variety of data sources, including demographic and mobile phone data (Andrey Bogomolov, 2014).

Datasets and Inputs

The biggest question I faced was what data to feed into the Machine Learning model. I had a hard time determining what factors might come in to play to cause someone to commit a violent crime. I settled on the following data: Chicago crime, traffic, weather, professional sporting events/results, bitcoin price, stock market price, and holidays.

Chicago Crime Dataset (2001 - Present) - Target

This dataset contains the historical crime data for Chicago. I will only be using the data from May 2015 – Nov 2018. I will filter out all the non-violent crimes from the dataset, since I'm only interested in the violent crimes. I will aggregate the data grouped by day to get the sum of the number of violent crimes happening each day. I will create a new categorical variable called "High_Volume_Day" that is used to identify if a day qualifies as a high-volume violent crime day. This new variable will be the target value that we will be trying to predict. I will set a threshold so that approximately 25% of the days are identified as high-volume days. If the field value is 1 then it will be considered a high-volume violent crime day. Conversely, if the value is 0 then it will not be considered high-volume day. The distribution of the classes will not be balances, since I'm purposely limiting the number of high-volume days to about 25%. There will be about 1,200 records once the data has been aggregated by day.

The data was obtained from the Chicago Data Portal website <https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present-Dashboard/5cd6-ry5g>.

Chicago Weather (May 2015 - Present) - Features

I believe weather could come into play in a couple of different ways. First being peoples' overall mood due to the weather. Personally, I'm usually in a more upbeat mood when the weather is warm and sunny. On the flipside, when the weather is cold, rainy, or snowy, I'm more likely to be easily irritated. The second way I could see this coming into play is that people are more likely to be outside if the weather is warm and sunny, so they are probably more likely to interact with other people which might give them more opportunities to commit a violent crime.

Numerical features include: Air Temperature, Wet Bulb Temperature, Humidity, Rain Intensity, Interval Rain, Total Rain, Precipitation Type, Wind Direction, Wind Speed, Maximum Wind Speed, Barometric Pressure, Solar Radiation. There are several records for each day from multiple locations, so I will have to summarize the data into one record for each day.

The data was obtained from the Chicago Data Portal website <https://data.cityofchicago.org/Parks-Recreation/Beach-Weather-Stations-Automated-Sensors/k7hf-8y75>.

Chicago Sports Data - Features

It's hard to find anything that interests Chicagoans more than sports. Often you can tell the outcome of last night's big game by the morale in the office the next morning. Rarely a day goes by when I don't get stopped to chat about the Cubs chances in the upcoming season. Hence, I believe sports could play a role in the occurrence violent crimes.

Categorical features: Opponent, Home/Away, Result

The data was obtained from the Sport Reference suite of websites <https://www.sports-reference.com>.

Dow Jones Historical Data - Features

Money makes the world go around. When the market is booming, people are probably more likely to be in a good mood. When the market is slipping, people are more likely to be on edge. I can see this playing a big role.

Numerical feature: Gain/Loss (new value that will be calculated)

The data was obtained from Yahoo Finance <https://finance.yahoo.com/quote/%5Edji/history?ltr=1>.

Bitcoin Historical Data - Features

Same explanation as the Dow Jones explanation.

Numerical feature: Gain/Loss (new value that will be calculated)

The data was obtained from Coin Market Cap website <https://coinmarketcap.com/currencies/bitcoin/historical-data>.

National Holidays - Features

I believe people are more likely to be off work on days that are considered a National Holiday. Therefore, they're more likely to engage in social gatherings (parties, etc.) and drinking which could lead to more violent behavior.

Categorical feature: Holiday?

The list of recognized National Holidays was obtained from Wikipedia <https://en.wikipedia.org/wiki/Federal_holidays_in_the_United_States>. The actual dates were obtained by searching in Outlook Calendar.

Problem Statement

The problem is that there are too many violent crimes being committed in the Chicago neighborhoods. According to the FBI crime dataset, in recent years Chicago has experienced violent crime rates nearly three times the national average (Uniform Crime Reporting (UCR) Program, 2019). In 2016, the national average was 387 violent crimes per 100,000 residents while Chicago clocked in at 1,105. In 2017, the national average was 383 while Chicago was at 1,099 per 100,000 residents.

This is a supervised classification problem. The input features will include a lot of data gathered about Chicago (crime, traffic, weather, professional sporting events/results) and some general data (bitcoin price, stock market price, and holidays). The output will be one of two predictions: 1 – the day is predicted to be a high-crime day or 0 – the day is not predicted to be a high-crime day.

Solution Statement

The solution I'm proposing is to use a Machine Learning model to identify when violent crimes are most likely to occur. The ML model will take as input Chicago traffic, Chicago weather, Chicago sports, Financial Market data, whether it is a Holiday, and it will determine if violent crimes are more likely to happen that day. Having this information available would give the Police an opportunity to allocate more of their resources during those times when the violent crimes are more likely to occur in effort to prevent violent crimes from being committed or if crime has been committed, they will have a better chance of apprehending the criminal.

The supervised learning models that will be considered are Gaussian Naïve Bayes, Decision Trees, Ensemble Methods, K-Nearest Neighbors, Stochastic Gradient Descent Classifier, Support Vector Machines, Logistic Regression. Since the dataset is imbalanced, 3:1 in favor of not a high-volume crime day, I will be using the AUROC as the metric to evaluate the effectiveness of the ML model (Lador, 2017).

Metrics

AUROC (Area Under the Receiver Operating Characteristics) will be the main metric used to evaluate the effectiveness of the ML model we create for this project, since the dataset is imbalanced as explained in the previous section. The ROC curve is plotted with TPR against the FPR where TPR is on y-axis and FPR is on the x-axis.

Measure definitions:

TPR (True Positive Rate)

$$\text{TPR} = \text{Recall} = \text{True Positives} / (\text{True Positives} + \text{False Negatives})$$

FPR (False Positive Rate)

$$\text{FPR} = \text{False Positives} / (\text{True Negatives} + \text{False Positives})$$

II. Analysis

Data Exploration

Featureset Exploration

Features

- **Holiday?:** 0, 1 (0: No, 1: Yes)
    - This list includes all Federal holidays (business days) plus a custom list of additional holiday dates that I put together. This list includes weekend dates surrounding holidays and other holidays that are celebrated, but do not make the Federal holiday list.
    - Key additions: Valentine's Day, St. Patrick's Day, Cinco de Mayo, Halloween, New Year's Eve.
- **Barometric_Pressure:** continuous
- **Wind_Speed:** continuous
- **Air_Temperature:** continuous
- **Rain:** continuous
- **Humidity:** continuous
- **BC_Daily_Change:** continuous
    - New feature created by subtracting the BC_Open value from BC_Close value
- **DOW_Daily_Change:** continuous
    - New feature created by subtracting the DOW_Open value from the DOW_Close value
- **Home Game:** Home, Away, NA
    - New feature created by combining data from baseball, basketball, football and hockey
    - Home: at least one home game
    - Away: all games are away
    - NA: no games played that day
- **Fan Morale:** Good, Bad, Neutral
    - New feature created by combining data from baseball, basketball, football and hockey
    - Good: One or more wins, no losses
    - Bad: One or more losses, no wins
    - Neutral: At least one win and one loss or no game at all

Target

- **High Crime Day?:** 0, 1 (0: No, 1: Yes)
    - New target feature created based on the amount of crime committed on a given day.
    - I wanted to identify about 25% of the projects as high crime days. So, the threshold was set to 317 violent crimes per day. Anything equal to or greater than that amount was label a High Crime Day.

Featureset Statistical Information

	Barometric_Pressure	Wind_Speed	Air_Temperature	Rain	Humidity	BC_Daily_Change	DOW_Daily_Change
count	1271.000000	1271.000000	1271.000000	1271.000000	1267.000000	1280.000000	884.000000
mean	11.778168	66.071694	0.149257	3.164299	993.845771	3.145203	0.986837
std	10.233056	12.132837	0.556031	1.412045	6.582052	295.961063	157.405457
min	-19.305417	25.958333	0.000000	0.766667	970.991667	-2345.600000	-1041.839843
25%	3.770417	57.613872	0.000000	2.120833	989.823370	-9.362500	-56.395996
50%	12.908889	66.750000	0.000000	2.930000	993.679167	1.500000	8.800782
75%	20.699347	75.651515	0.028949	3.952277	997.981250	24.852500	73.819824
max	30.348182	91.173913	13.382083	9.795455	1015.971429	3633.600000	827.599609

Missing Values

There are many 'NaN' values that need to be addressed.

```
crime_data_updated.isnull().sum()
```

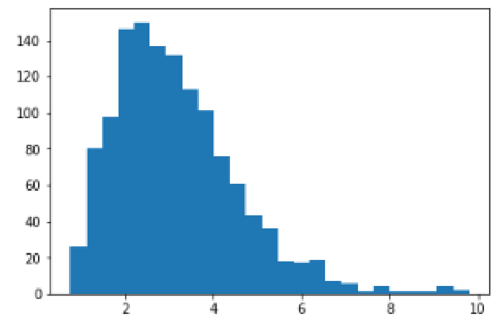
```
Date                    0
High Crime Day?         0
Barometric_Pressure     9
Wind_Speed              9
Air_Temperature         9
Rain                    9
Humidity               13
BC_Daily_Change         0
DOW_Daily_Change      396
Cubs_H/A              681
Cubs_Opp              681
Cubs_Result           681
Sox_H/A               680
Sox_Opp               680
Sox_Result            680
Basketball_H/A       1028
Basketball_Opp       1028
Basketball_Result    1028
Football_Result      1240
Football_H/A         1240
Football_Opp         1240
Hockey_H/A            992
Hockey_Opp            992
Hockey_Result         992
Holiday?                0
```

- DOW_Daily_Change
    - There are 396 NaN values since the DOW data is only captured during business days. For the days when the market was closed, I will set all NaN values to 0 since there would not be any change for those days.
- Barometric_Pressure, Wind_Speed, Air_Temperature, Rain, Humidity
    - I will set all NaN values to the mean for each of the metrics.
- There was a huge amount of NA values for the sporting event data.
    - I decided to combine all of the features into two new features – "Home Game" and "Fan Morale". This reduced the amount of NaN values to just 210 for which I assigned the value "NA".

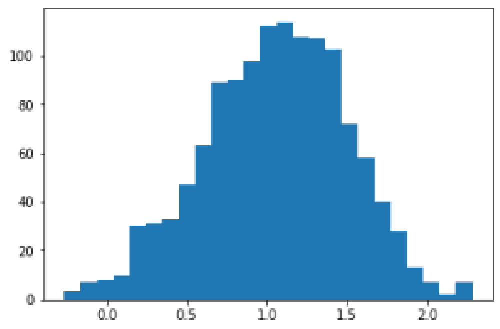
Transforming Features

Wind_Speed: the data was skewed to the right, so I applied the log to make it normally distributed.

`Wind_Speed`

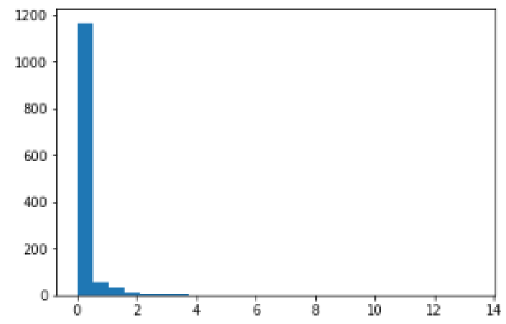


`Wind_Speed (Log)`

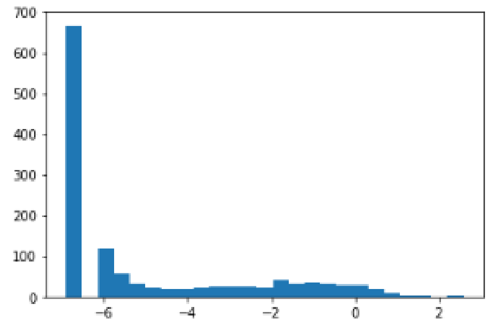


Rain: The values are all jammed on the left since it is unlikely to get substantial amount of rain in Chicago area. I took the log to make the values on the left more spread out.

`Rain`

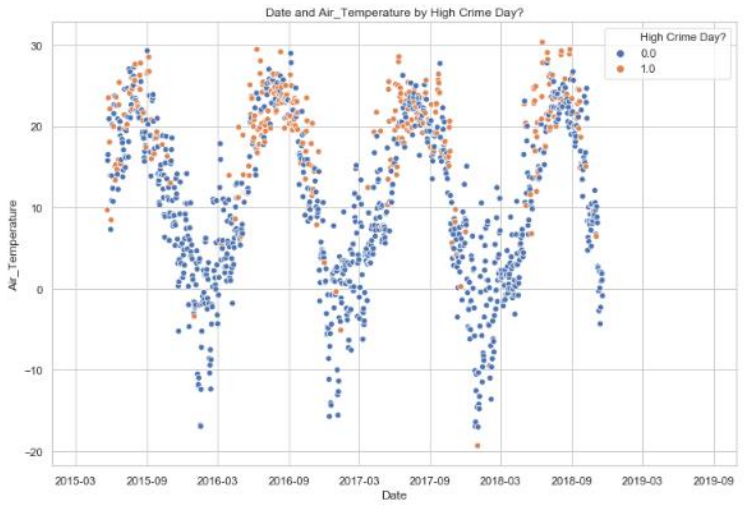


`Rain (Log)`



Exploratory Visualization - Chicago Crime by Weather and Date

The visualization below shows a noticeable seasonality trend where the higher crime days seem to take place mid-year when there is warmer weather. This leads me to believe that weather (especially temperature) is going to play a major role in determining whether or a not a day will be a high crime day.



Algorithms and Techniques

Algorithms (Modern Machine Learning Algorithms: Strengths and Weaknesses, 2017)

Decision Trees

- Strengths:
    - Easy to understand and implement.
    - Performs well on imbalanced datasets.
    - Allows for a data driven exploration of non-linear relationships.
    - Easy to interpret and visualize. Handles large data sets well.
    - Can predict binary target variables as well as quantitative target variables.
- Weaknesses:
    - Small changes in the data can lead to different split which can undermine the interpretability of the model.
    - Not very reproducible on future data.
- Why is this a good candidate?
    - Perform well on imbalanced datasets, handles large data sets well (roughly 45k records in dataset), and can predict binary target variables.
- How they are trained:
    - The decision tree is split based on the attibrute that provides the most information gain. The key is to find a balance between overfitting (too many branches) the tree and underfitting (not enough branches).

Logistic Regression

- Strengths:
    - Fast training time.
    - Easy to implement and understand.
    - Outputs have nice probabilistic interpretation.
    - Algorithm can be regularized to avoid overfitting.
    - Logistic models can be updated easily with new data using stochastic gradient descent.
- Weaknesses:
    - Underperforms when there are multiple or non-linear decision boundaries.
    - Not flexible enough to naturally capture more complex relationships.
- Why is this a good candidate?
    - Easy to implement, fast training time and can predict binary target variables.
- How they are trained:
    - Uses decision boundary to classify probabilities into positive and negative classes.

Support Vector Machines

- Strengths:
    - Moderate training time.
    - Model non-linear decision boundaries.
    - Many kernels to choose from.
    - Fairly robust against overfitting.
    - Good for large feature sets.
- Weaknesses:
    - Memory intensive.
    - Trickier to tune due to the importance of picking the right kernel.
    - Don't scale well to larger datasets.
- Why is this a good candidate?

    - Good for large feature sets (we have 103 after one-hot encoding) and can use penalized classification to care for the imbalanced dataset.
- How they are trained:
    - Support Vector Classifiers use a hyperplane to determine how to classify a prediction. The hyperplane is determined by choosing the line that creates the greatest possible margin between the hyperplane and any point within the training set (Bambrick, 2019).

Techniques

Apply Supervised Machine Learning Models and Evaluate Model Performance
The supervised learning models that will be used are Decision Trees, Support Vector Machines, and Logistic Regression. I will apply each model to the dataset. I will review the results of each model to determine which model's performance was the best. I will then use that model to improve upon.

Improve Model
I will use GridSearchCV to fine tune the chosen model in hopes to achieve even greater performance.

Final Model Evaluation
I will compare the AUROC results of the optimized model to that of the Naïve Predictor.

Extract Feature Importance
Lastly, I will extract the feature importance values to determine which features are providing the most predictive power.

Benchmark

Naïve Predictor

The model will be evaluated on whether it gets a result better than random choice. This model will be referred to as the "Naive Predictor". The Naïve Predictor will be the value we get if we chose a model that always predicted '1' (i.e. the day is considered a high volume violent crime day). For the ML model to be useful I have limited the number of days that are considered "High Volume" to just over 25%, otherwise the Police would be on high alert too often and the model wouldn't be useful. To identify whether a day is a high-volume day I summed up the number of violent crimes by day and then used 317 crimes as the threshold that identifies approximately 25% of the projects as high-volume days.

- 324 (# of high crime days) / 1280 (# of total days) = 0.253125
- AUROC = 0.5

III. Methodology

Data Preprocessing

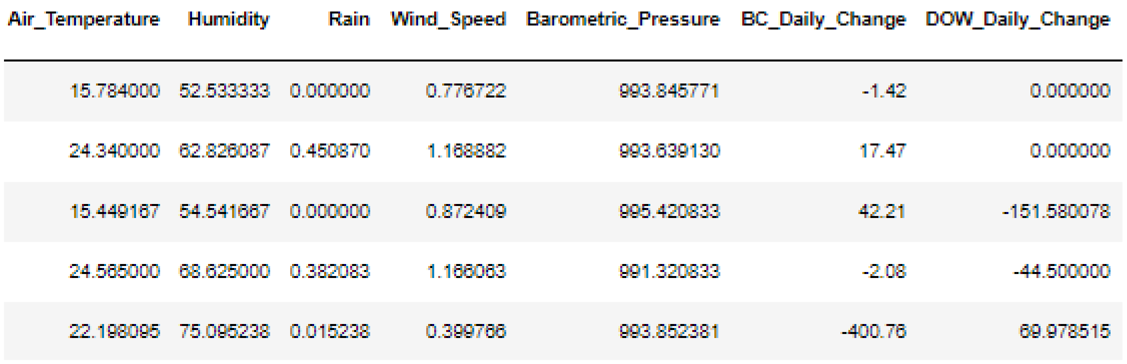
Step 1: Split Data into Feature and Target Sets

The first thing I did was split out the “High Crime Day?” (target variable) into its own list. I also dropped the “Date” feature from the features Data Frame since it will not be used to train the ML Models.

Step 2: Min-Max Scaler

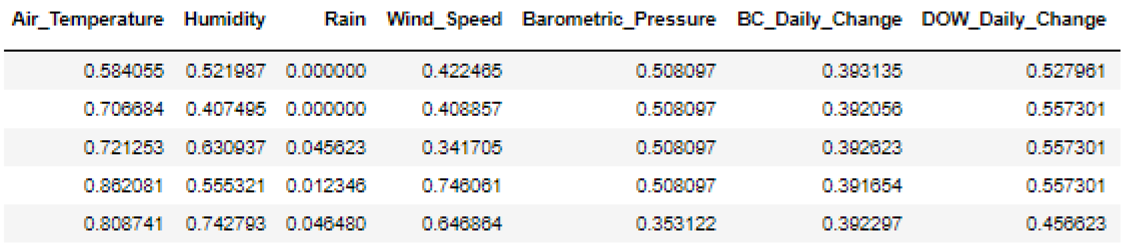
Second, I normalized the numerical features by using min-max scaler to scale all numerical features to a value between 0 and 1. Applying scaling in this manner maintained the feature’s distribution and ensured that each feature was treated equally by the ML Learner.

Sample taken before min-max scaling:



Air_Temperature	Humidity	Rain	Wind_Speed	Barometric_Pressure	BC_Daily_Change	DOW_Daily_Change
15.784000	52.533333	0.000000	0.776722	993.845771	-1.42	0.000000
24.340000	62.826087	0.450870	1.168882	993.639130	17.47	0.000000
15.449167	54.541667	0.000000	0.872409	995.420833	42.21	-151.580078
24.565000	68.625000	0.382083	1.166063	991.320833	-2.08	-44.500000
22.198095	75.095238	0.015238	0.399766	993.852381	-400.76	69.978515

Sample taken after min-max scaling:



Air_Temperature	Humidity	Rain	Wind_Speed	Barometric_Pressure	BC_Daily_Change	DOW_Daily_Change
0.584055	0.521987	0.000000	0.422465	0.508097	0.393135	0.527961
0.706684	0.407495	0.000000	0.408857	0.508097	0.392056	0.557301
0.721253	0.630937	0.045623	0.341705	0.508097	0.392623	0.557301
0.862081	0.555321	0.012346	0.746061	0.508097	0.391654	0.557301
0.808741	0.742793	0.046480	0.646864	0.353122	0.392297	0.456623

Step 3: One-Hot Encoding

Third, I used one-hot encoding to convert the non-numerical features to numerical features. One-hot encoding creates a "dummy" variable for each possible category of each non-numeric feature.

“Home Game” split into 3 new variables:

- Home Game_NA
- Home Game_No
- Home Game_Yes

“Fan Morale” split into 3 new variables:

- Fan Morale_Bad
- Fan Morale_Good
- Fan Morale_Neutral

Step 4: Training / Testing Split

Fourth, I used the train_test_split module from sklearn.cross_validation to split the features and target data into training and testing sets. I allocated 80% of the data to the training set and 20% to the testing set.

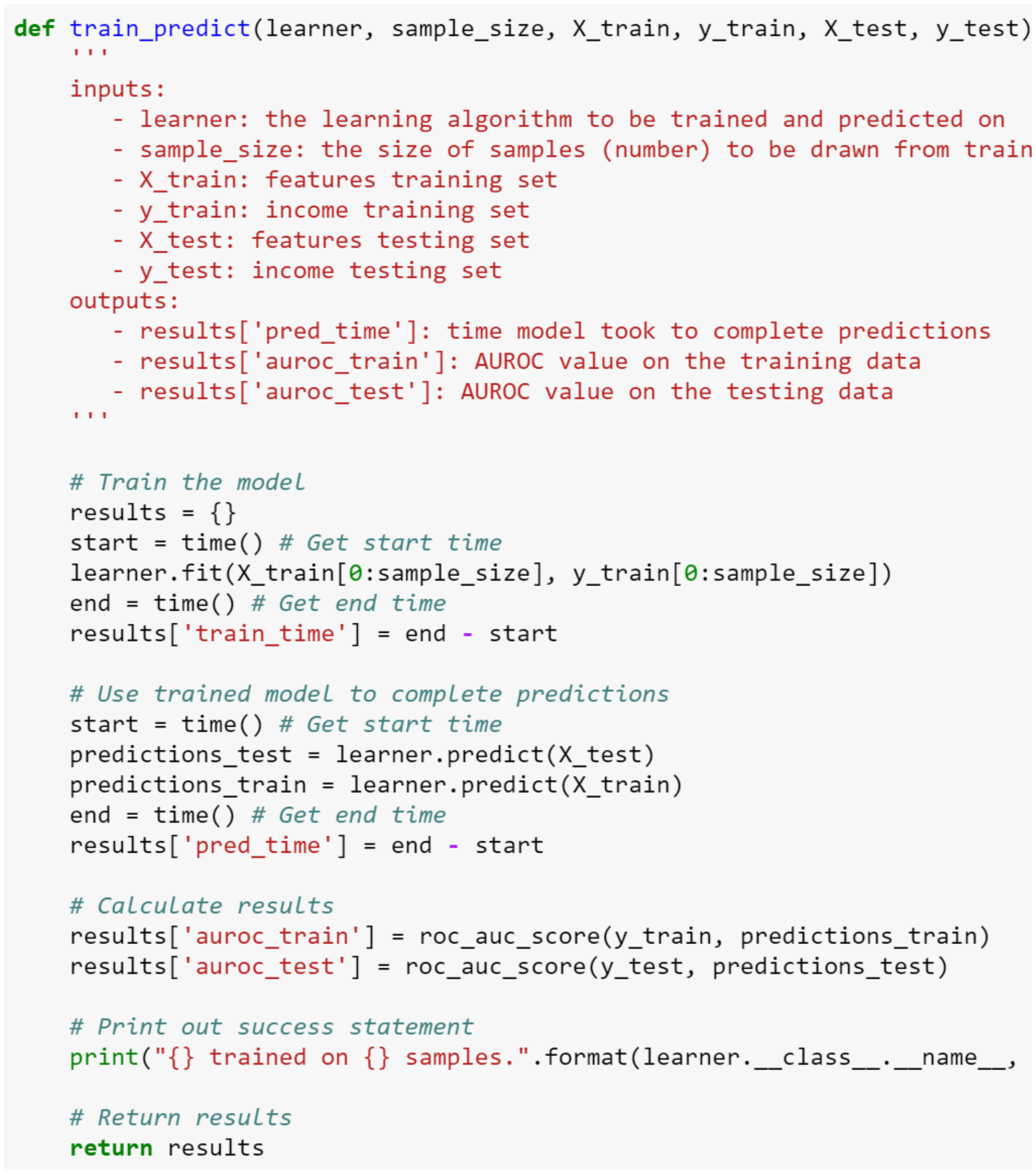
- Training set has 1024 samples
- Testing set has 256 samples

Implementation

Step 1: Training / Predicting Pipeline

First, I created a training and predicting pipeline that I used to quickly train various models using various sizes of training data and perform predictions on the testing data.

The train_predict function accepts as input: the learner, sample size, training data, and testing data. The function first trains the model. Then, the function uses the trained model to complete its predictions for the testing data. Lastly, the model calculates the AUROC score for both the training and testing data and returns the results.



```python
def train_predict(learner, sample_size, X_train, y_train, X_test, y_test)
    '''
    inputs:
       - learner: the learning algorithm to be trained and predicted on
       - sample_size: the size of samples (number) to be drawn from train
       - X_train: features training set
       - y_train: income training set
       - X_test: features testing set
       - y_test: income testing set
    outputs:
       - results['pred_time']: time model took to complete predictions
       - results['auroc_train']: AUROC value on the training data
       - results['auroc_test']: AUROC value on the testing data
    '''

    # Train the model
    results = {}
    start = time() # Get start time
    learner.fit(X_train[0:sample_size], y_train[0:sample_size])
    end = time() # Get end time
    results['train_time'] = end - start

    # Use trained model to complete predictions
    start = time() # Get start time
    predictions_test = learner.predict(X_test)
    predictions_train = learner.predict(X_train)
    end = time() # Get end time
    results['pred_time'] = end - start

    # Calculate results
    results['auroc_train'] = roc_auc_score(y_train, predictions_train)
    results['auroc_test'] = roc_auc_score(y_test, predictions_test)

    # Print out success statement
    print("{} trained on {} samples.".format(learner.__class__.__name__,

    # Return results
    return results
```

Step 2: Initial Model Evaluation

Second, I wrote nested for statements to iterate through a list of learners (Decision Tree, SVM, and Logistic Regression) and list of sample sizes (1%, 10%, 100%). Each iteration was passed through the training / prediction pipeline. The results of each iteration were printed out at the end.

Results:

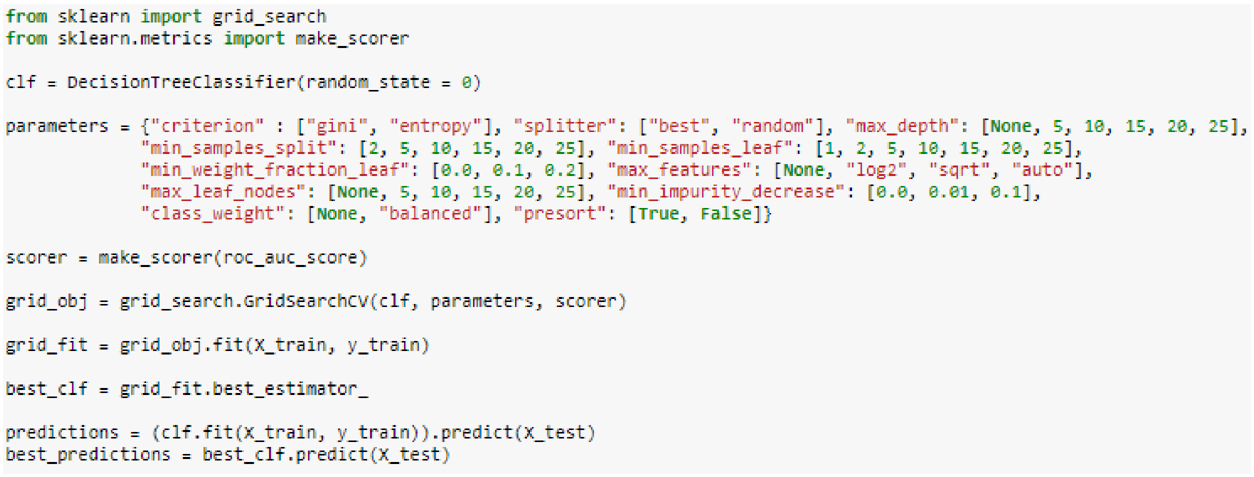
Learner	**AUROC_Test (trained on 1% Data)**	**AUROC_Test (trained on 10% Data)**	**AUROC_Test (trained on 100% Data)**
Decision Tree Classifier	0.573	**0.708**	**0.702**
Logistic Regression	0.623	0.588	0.687
SVC	0.500	0.500	0.500

Based on the results, I chose to move forward using the Decision Tree Classifier.

Refinement

GridSearchCV and Make_Scorer

I used GridSearchCV and make_scorer to fine tune the model to achieve even greater scores. The roc_auc_score was used to evaluate the models to determine the best parameters to use to achieve the highest AUROC score.



```
from sklearn import grid_search
from sklearn.metrics import make_scorer

clf = DecisionTreeClassifier(random_state = 0)

parameters = {"criterion" : ["gini", "entropy"], "splitter": ["best", "random"], "max_depth": [None, 5, 10, 15, 20, 25],
             "min_samples_split": [2, 5, 10, 15, 20, 25], "min_samples_leaf": [1, 2, 5, 10, 15, 20, 25],
             "min_weight_fraction_leaf": [0.0, 0.1, 0.2], "max_features": [None, "log2", "sqrt", "auto"],
             "max_leaf_nodes": [None, 5, 10, 15, 20, 25], "min_impurity_decrease": [0.0, 0.01, 0.1],
             "class_weight": [None, "balanced"], "presort": [True, False]}

scorer = make_scorer(roc_auc_score)

grid_obj = grid_search.GridSearchCV(clf, parameters, scorer)

grid_fit = grid_obj.fit(X_train, y_train)

best_clf = grid_fit.best_estimator_

predictions = (clf.fit(X_train, y_train)).predict(X_test)
best_predictions = best_clf.predict(X_test)
```

Final Model (Best CLF)

DecisionTreeClassifier(class_weight='balanced', criterion='entropy', max_depth=None, max_features=None, max_leaf_nodes=10, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, presort=True, random_state=0, splitter='best')

Metric	**Unoptimized Model**	**Optimized Model**	**Reduced Data (Top 5 Feature Imp)**
AUROC Score	0.702	**0.7678**	**0.7678**

IV. Results

Model Evaluation and Validation

Model Evaluation

Final Model (Best CLF)

DecisionTreeClassifier(class_weight='balanced', criterion='entropy', max_depth=None, max_features=None, max_leaf_nodes=10, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, presort=True, random_state=0, splitter='best')

Parameter Breakdown (sklearn.tree.DecisionTreeClassifier, 2019)

Class_weight = 'balanced': This indicates that the classes will be balanced out by replicating the smaller class until you have as many samples as in the larger one.

Criterion = 'entropy': This indicates that the function to measure the quality of a split will be entropy (information gain).

max_depth = None: This indicates that there is no limit to how deep the tree can be and thus will be expanded until all the leaves contain less than min_samples_split samples.

max_features = None: This indicates that there is no limit to the number of features to consider when looking for the best split.

max_leaf_nodes = 10: This indicates that the limit of leaf nodes is 10.

min_impurity_decrease = 0.0: This indicates that a node will be split if the split induces a decrease of the impurity greater than or equal to 0.0, so basically it will always split if there is any decrease to the impurity.

min_impurity_split = None: Deprecated in favor of min_impurity_decrease.

min_samples_leaf = 1: This indicates that at least 1 sample is required to be at a leaf node.

min_samples_split = 2: This indicates that at least 2 samples are required to split an internal node.

min_weight_fraction_leaf = 0.0: This indicates that there is not a minimum weighted fraction of the sum of weights required to be at a leaf node.

presort = True: This indicates that the data will be presorted to speed up the finding of best splits in fitting. This could slow down the training process for large datasets, but since our dataset is relatively small shouldn't be a problem.

splitter = 'best': The indicates that the strategy used to choose the split at each node will be to choose the best split.

Validation

I used the train_test_split module to split the features and target data into training and testing sets. 80% of the data was used to train the model, the remaining 20% was used to validate the model. Since, the validation set was not used to train the model, the robustness of the model was tested using unseen data, so the results can be trusted.

However, the model relies heavily on the "Air_Temperature" feature giving it a feature importance value of 0.6464. So, an unusually cold day in the summer or warm day in the winter would most likely receive poor results from the model.

Justification

The optimized model and the reduced data model both outperformed the benchmark model by almost 54%.

Metric	**Benchmark Model**	**Optimized Model**	**Reduced Data (Top 5 Feature Imp)**
AUROC Score	0.5	**0.7678**	**0.7678**

The purpose of this model is for Chicago PD to be able to identify when violent crimes are most likely to occur. They would then be able to better allocate their resources during those time when the violent crimes are more likely to occur in effort to prevent violent crimes from being committed or if crime has been committed, they will have a better chance of apprehending the criminal.

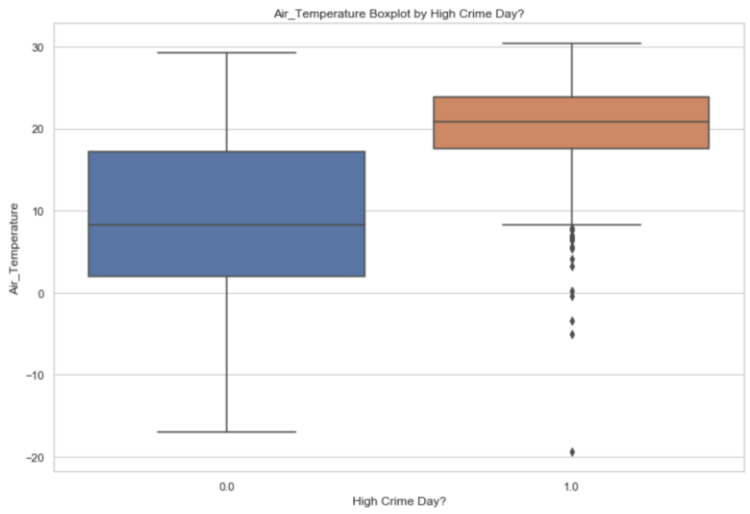
Given the nature of this problem and the fact that lives are at stake, the AUROC score of 0.7678 may not be adequate to consider the problem solved. While it performs significantly better than the benchmark model, I would consider continuing to iterate on the model by including additional data before relying on the model to make decisions about how many officers to deploy on a given day.

V. Conclusion

Free-Form Visualization

The visualization below backs up the model's determination that "Air_Temperature" is by far the most important feature in determining whether a day is going to be a "High Crime Day". The model gave a feature importance score of 0.6464 to Air_Temperature.

As you can see, the days classified as high crime days tend to be much warmer than non-high crime days. The 25th percentile for the high crime days is higher than the 75th percentile for the non-high crime days days. In addition, the median for the high crime days is also significantly higher at close to 3 times that of the non-high crime days.



Reflection

Steps taken:

1. I read in all of the data from excel and csv files.
2. I prepared the data for exploratory data analysis by adding new features, modifying existing features, and deleting features that wouldn't be used.
3. I used exploratory data analysis techniques to explore the data to uncover patterns in the data and to determine if any features needed to be transformed to make the ML model more accurate.
4. I prepared the data for input in to the ML model by transforming features that had skewed distributions, using Min-Max Scaler to scale all the numerical features to values between 0 and 1, and used One-Hot Encoding to change all of the categorical features to numerical.
5. I established a Benchmark Model, so the ML model's effectiveness could be evaluated.
6. I applied various supervised ML models (Decision Tree Classifier, Logistic Regression, and SVC) to the data.
7. I used GridSerach and Make_Scorer to refine the model to achieve greater scores.
8. I evaluated the model performance and extracted feature importance.

I found it to be interesting how big of a role the temperature played. Temperature received a feature importance value of 0.6464 which was by far the highest value. The second highest was 0.25 which was assigned to DOW_Daily_Change.

I also thought it was interesting that holidays received a feature importance value of 0.0. I thought that holidays would at least play a small role.

The most difficult aspect of the project for me was to get the data into a format that would be useful for the ML model. Each of the CSV and Excel files had many features that needed to be removed or used to calculate new features. There was also some messy data, so the data had to be cleaned before it could be input into the ML model.

Improvement

I think the model could be improved by using the calendar month as an input choice. Right now, the model is relying heavily on the temperature to determine if a day will be classified as a high crime day, but if there is an unusually warm day in the winter or unusually cold day in the summer the model could be thrown off. If we added in the calendar month, then this could help the model to not overreact when the temperature is unusually warm or cold. This would help the model pick up on the seasonality trend of the crime data as well. I would expect the updated ML model to achieve a greater AUROC score.

Bibliography

Andrey Bogomolov, B. (2014, 09 10). *Once Upon a Crime: Towards Crime Prediction*. Retrieved from arXiv: https://arxiv.org/pdf/1409.2983.pdf

Lador, S. M. (2017, 09 05). *What metrics should be used for evaluating a model on an imbalanced data set?* Retrieved from Towards Data Science: https://towardsdatascience.com/what-metrics-should-we-use-on-imbalanced-data-set-precision-recall-roc-e2e79252aeba

Modern Machine Learning Algorithms: Strengths and Weaknesses. (2017, 05 16). Retrieved from Elite Data Science: https://elitedatascience.com/machine-learning-algorithms

Narkhede, S. (2018, 06 26). *Understanding AUC - ROC Curve*. Retrieved from Towards Data Science: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

sklearn.tree.DecisionTreeClassifier. (2019, 03 28). Retrieved from Scikit Learn: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

Uniform Crime Reporting (UCR) Program. (2019, 01 26). Retrieved from FBI: https://www.fbi.gov/services/cjis/ucr