Safety Recommenders Report

# Data Sourcing

Python applications download data from DC websites:

# Wrangling

1. Eliminated rows in the CSV file that contain no data in any column. We eliminated these rows in order to eliminate \_\_\_.
2. Replace values in the offense\_text column to eliminate values with the slash (/):
   1. “theft/other” to “theft”
   2. “theft f/auto” to “auto theft”
   3. “assault w/dangerous weapon” to “assault with weapon”

We made these changes to improve the readability of the results.

# Machine Learning

We employ supervised learning – generalizing from know examples. We provide algorithms with a set of inputs and desired outputs and the algorithm produces the learned output given a new input.

## Questions

What kind of crime (UCR Rank) can be predicted based on:

* Latitude and longitude
* Month, day and hour
* A combination of location and time

## Problem Type

This is a multiclass classification problem with the desired data point for each crime being a Uniform Crime Report Ranking (UCR Rank):

| **ucrrank** | **offense\_key** |
| --- | --- |
| 1 | violent|homicide |
| 2 | violent|sex abuse |
| 3 | violent|assault w/dangerous weapon |
| 4 | violent|robbery |
| 5 | property|burglary |
| 6 | property|theft/other |
| 7 | property|theft f/auto |
| 8 | property|motor vehicle theft |
| 9 | property|arson |

For a specific data point (crime), the UCR Rank is its label. We could think of this as a classification or a linear regression task.

In addition to looking at UCR Ranks as categories of crimes or we can see the rankings as a continuum where 1 is the most serious, violent crime (i.e., homicide), 9 is the least serious crime (i.e. arson), and other crimes fit between those two on the numerical scale.

…able to build a model able to generalize accurately from the training set to the test set.

# Django & Machine Learning

<https://us.pycon.org/2016/schedule/presentation/1614/> -- Ben and Rebecca

# Intro to ML

## Mglearn problem

<https://github.com/amueller/mglearn/issues/5>

<https://github.com/amueller/introduction_to_ml_with_python/issues/3> [I did this]

mglearn project from its official github repository and copy mglearn folder to Anaconda3\Lib\site-packages

I changed the following import line in the mglearn/plot\_animal\_tree.py and mglearn/plot\_interactive\_tree.py files from:

from scipy.misc import imread

to

from scipy.misc.pilutil import imread. [I did this]

<https://github.com/amueller/mglearn/issues/2>

$ easy\_install Pillow

Worked:

### [hugovk](https://github.com/hugovk) **commented** [**12 days ago**](https://github.com/python-pillow/Pillow/issues/2479#issuecomment-389525813)

|  |
| --- |
| [**@bburns**](https://github.com/bburns) Please see [#2945](https://github.com/python-pillow/Pillow/issues/2945) and [conda-forge/pillow-feedstock#45](https://github.com/conda-forge/pillow-feedstock/issues/45), it's a problem with conda.  One suggested workaround:  conda remove pillow  pip install pillow  Or:  conda update –all [I did this] |
| <https://github.com/python-pillow/Pillow/issues/2479> Exploring Data X, y = mglearn.datasets.load\_extended\_boston()  print("X.shape {}".format(X.shape))  from sklearn.datasets import load\_iris  iris\_dataset = load\_iris()  print("Keys of iris\_dataset: \n{}".format(iris\_dataset.keys()))  print(iris\_dataset['DESCR'][:3193] + "\n...")  print("Target names: {}".format(iris\_dataset['target\_names']))  print("Feature names: \n{}".format(iris\_dataset['feature\_names'])) Naïve Bayes Classifiers  | **ALGORITHM** | **STRENGTHS** | **WEAKNESSES** | **NOTES** | **REGULARIZATION/ PARAMETER NOTES** | | --- | --- | --- | --- | --- | |  |  |  |  |  | | BernoulliNB |  |  |  | Assumes binary data;  How often every feature of each class is not zero;  Bigger alpha = less complex;  Sparse count data like text | | Decision Tree |  |  |  | Tweak tree depth;  See feature importance | | GAUSSIANNB |  |  |  | Apply to continuous data;  Average value and standard deviation of each feature for each class;  Very high dimension data | | KERNELIZED SUPPORT VECTOR MACHINES |  |  |  | Better generalization in low-dimensional spaces | | LINEAR MODELS |  |  |  | C – smaller – simpler;  good for very large datasets;  good when large number of features compared to number of samples;  SGDClassifier and SGDRegressor support scalability | | LINEAR REGRESSION |  |  |  | Alpha – larger = simpler; fast to train and predict; work well with small datasets and scale to large | | LINEARSVC |  |  |  |  | | LOGISTIC REGRESSION |  |  |  | l1 – only a few features important, so also easier interpretability;  l2 – default;  Use solver=’sag’ for hundreds of thousands or millions of records | | MultinomialNB |  |  |  | Assumes count data, e.g. words in sentence; Takes into account average value of each feature for each class;  Bigger alpha = less complex;  Sparse count data like text;  Performs better than BernoulliNB with relatively large number of nonzero features (i.e., large documents) | | NAÏVE BAYES CLASSIFIERS |  |  |  | Train faster than linear, slightly worse generalization than LogisticRegression and LinearSVC;  To make a prediction a data point is compared to the statistics for each of the classes, and the best matching class is predicted;  Fast to train and predict;  Very well on hi-dimensional sparse data;  Good on large datasets where training might take too long even with linear model | | RIDGE |  |  |  | Use solver=’sag’ for hundreds of thousands or millions of records | |

# June 2, 2018

Build and Validation of Crime SVM Classifier took 18944.776 seconds

Validation scores are as follows:

precision 0.330272

recall 0.428137

accuracy 0.428137

f1 0.335200

dtype: float64

Fitted model written to:

C:\Users\Judith\projects\safetyrecommender\crime-svm-classifier.pickle

Build and Validation of Crime kNN Classifier took 6.247 seconds

Validation scores are as follows:

precision 0.334133

recall 0.402764

accuracy 0.402764

f1 0.347662

dtype: float64

Fitted model written to:

C:\Users\Judith\projects\safetyrecommender\crime-knn-classifier.pickle

Build and Validation of Logistic Regression took 72.228 seconds

Validation scores are as follows:

precision 0.178081

recall 0.421921

accuracy 0.421921

f1 0.250435

dtype: float64

Fitted model written to:

C:\Users\Judith\projects\safetyrecommender\logistic-regression.pickle

Out[39]:

Timestamp('2018-06-01 17:38:42.576546')

Build and Validation of Gaussian NB took 0.833 seconds

Validation scores are as follows:

precision 0.278147

recall 0.420225

accuracy 0.420225

f1 0.285300

dtype: float64

Fitted model written to:

C:\Users\Judith\projects\safetyrecommender\gaussian-nb.pickle

Build and Validation of RandomForestClassifier took 13.205 seconds

Validation scores are as follows:

precision 0.328266

recall 0.347587

accuracy 0.347587

f1 0.336913

dtype: float64

Fitted model written to:

C:\Users\Judith\projects\safetyrecommender\randomforestclassifier.pickle

Created a Jupyter notebook that represents all phases of the machine learning process:

1. Ingestion, including the http request for current crime data in csv format from: <https://datagate.dc.gov/search/open/crimes?daterange=2years&details=true&format=csv>
2. Data wrangling