San Francisco: Crime Classification

April 20, 2019

Yang Yang Qian, Arthur Lima, Linda Dong

Problem - 1m [Linda]
EDA / Data Cleaning - 1m [Yang Yang]
Data Pipelining / Feature Engineering - 2m [Yang Yang]
Reduce dimensionality PCA - 1m [Arthur]
Decentralized model exploration: first run - 5m [Each explain own model]
Overall Results - 2m [Linda]

Challenge

- Predict the type of crime committed
  - from amongst 39 categories, e.g.:
    ASSAULT
    ROBBERY
    THEFT/LARCENY
  - using data sourced from the city government over 12 years (2003-2015)

- To evaluate model performance, our primary measure is (multi-class) log loss
  - aka: cross entropy

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij}),$$

Challenge

- Predict the type of crime committed
  - from amongst 39 categories, e.g.:
    ASSAULT
    ROBBERY
    THEFT/LARCENY
  - using data sourced from the city government over 12 years (2003-2015)

- To evaluate model performance, our primary measure is (multi-class) log loss
  - aka: cross entropy

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij}),$$

Exploratory Data Analysis

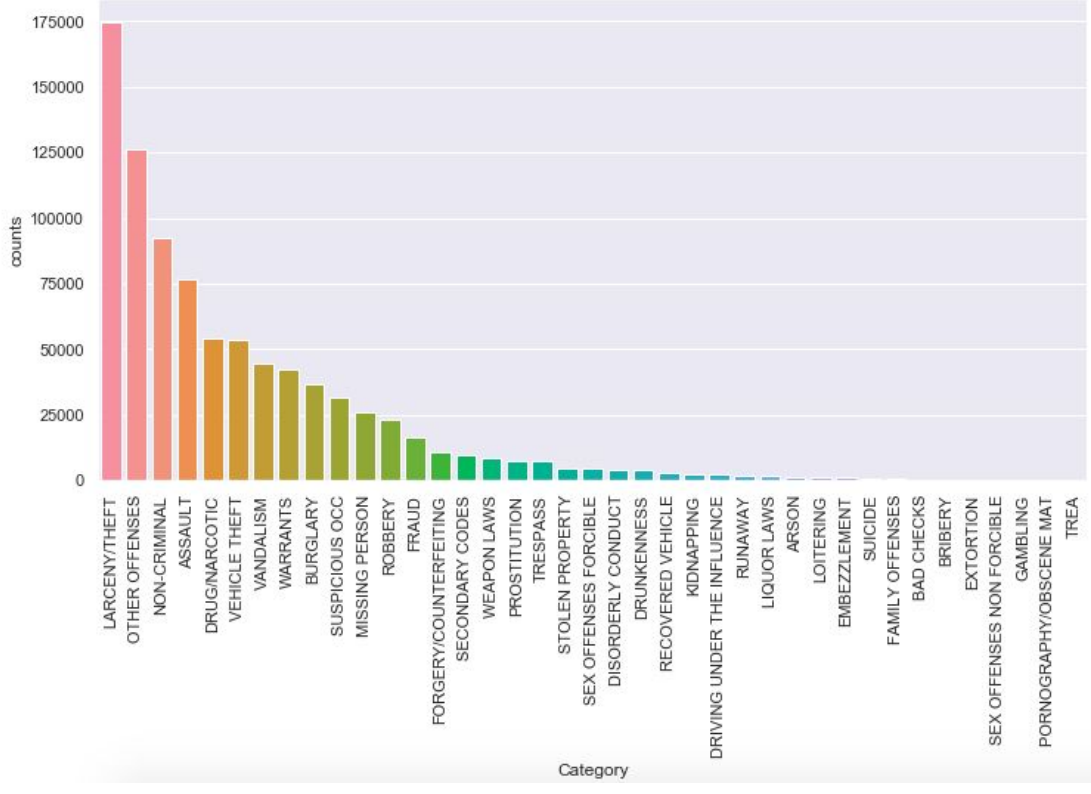
EDA

- About 800 thousand records in both train and test
- Train only has 9 columns
- Test only has 7 columns

2003-01-07 07:52:00	WARRANTS	WARRANT ARREST	Tuesday	SOUTHERN	ARREST, BOOKED	5TH ST / SHIPLEY ST	-122.402843	37.779829
2003-01-07 04:49:00	WARRANTS	ENROUTE TO OUTSIDE JURISDICTION	Tuesday	TENDERLOIN	ARREST, BOOKED	CYRIL MAGNIN STORTH ST / EDDY ST	-122.408495	37.784452
2003-01-07 03:52:00	WARRANTS	WARRANT ARREST	Tuesday	NORTHERN	ARREST, BOOKED	OFARRELL ST / LARKIN ST	-122.417904	37.785167
2003-01-07 03:34:00	WARRANTS	WARRANT ARREST	Tuesday	NORTHERN	ARREST, BOOKED	DIVISADERO ST / LOMBARD ST	-122.442650	37.798999
2003-01-07 01:22:00	WARRANTS	WARRANT ARREST	Tuesday	SOUTHERN	ARREST, BOOKED	900 Block of MARKET ST	-122.409537	37.782691
2003-01-06 23:30:00	WARRANTS	ENROUTE TO OUTSIDE JURISDICTION	Monday	BAYVIEW	ARREST, BOOKED	REVERE AV / INGALLS ST	-122.384557	37.728487
2003-01-06 23:14:00	WARRANTS	WARRANT ARREST	Monday	CENTRAL	ARREST, BOOKED	BUSH ST / HYDE ST	-122.417019	37.789110
2003-01-06 22:45:00	WARRANTS	WARRANT ARREST	Monday	SOUTHERN	ARREST, BOOKED	800 Block of BRYANT ST	-122.403405	37.775421
2003-01-06 22:45:00	WARRANTS	ENROUTE TO OUTSIDE JURISDICTION	Monday	SOUTHERN	ARREST, BOOKED	800 Block of BRYANT ST	-122.403405	37.775421

Category

- Categorical variable with 39 levels
- is what we are trying to predict
- Largest Category of crimes is theft, at around 20% of all crime
    - Larceny/Theft



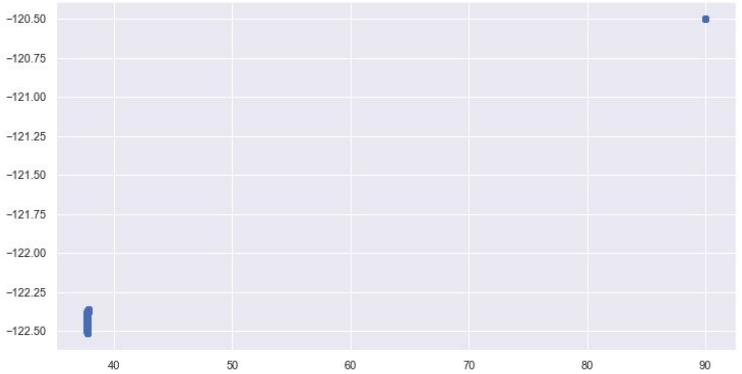
Address

- Can extract street, intersection, and various other features
- Probably correlated with X and Y

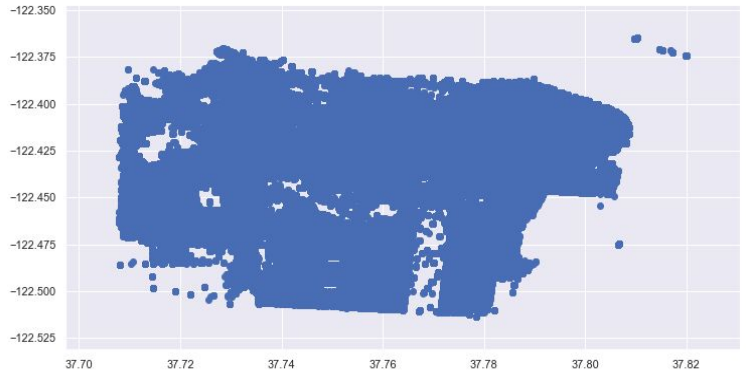
```
The top Adresses are: 800 Block of BRYANT ST      26533
800 Block of MARKET ST       6581
2000 Block of MISSION ST     5097
1000 Block of POTRERO AV     4063
900 Block of MARKET ST       3251
Name: Address, dtype: int64
```

X and Y

- Latitude and Longitude
- Appears to have some outliers, some mis-coded locations
- We have outliers in both train and test
- Rounding the values; don’t need that much precision



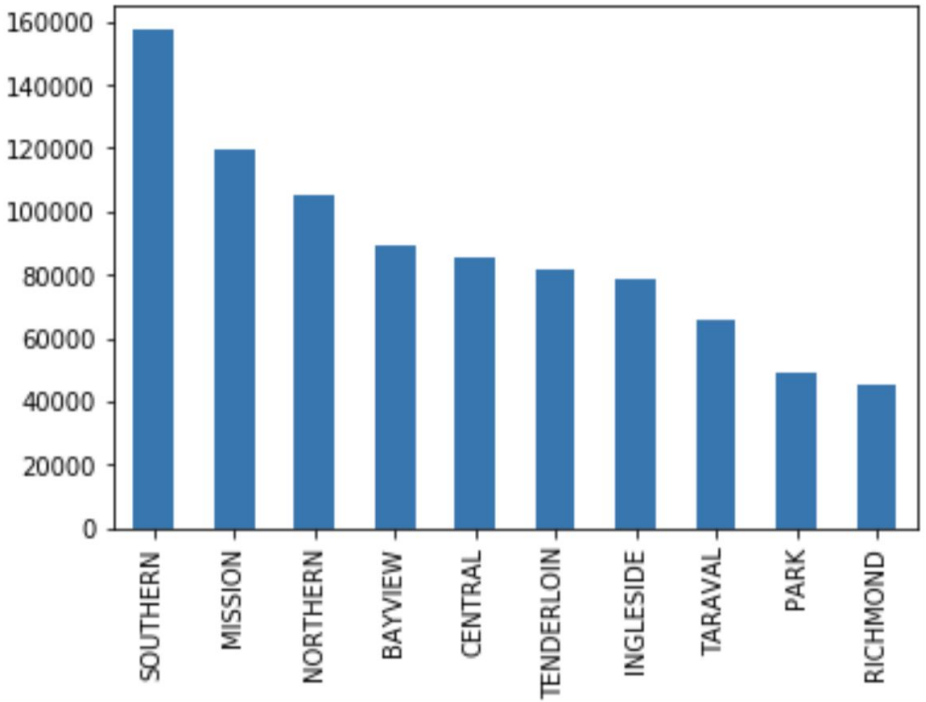
before



after

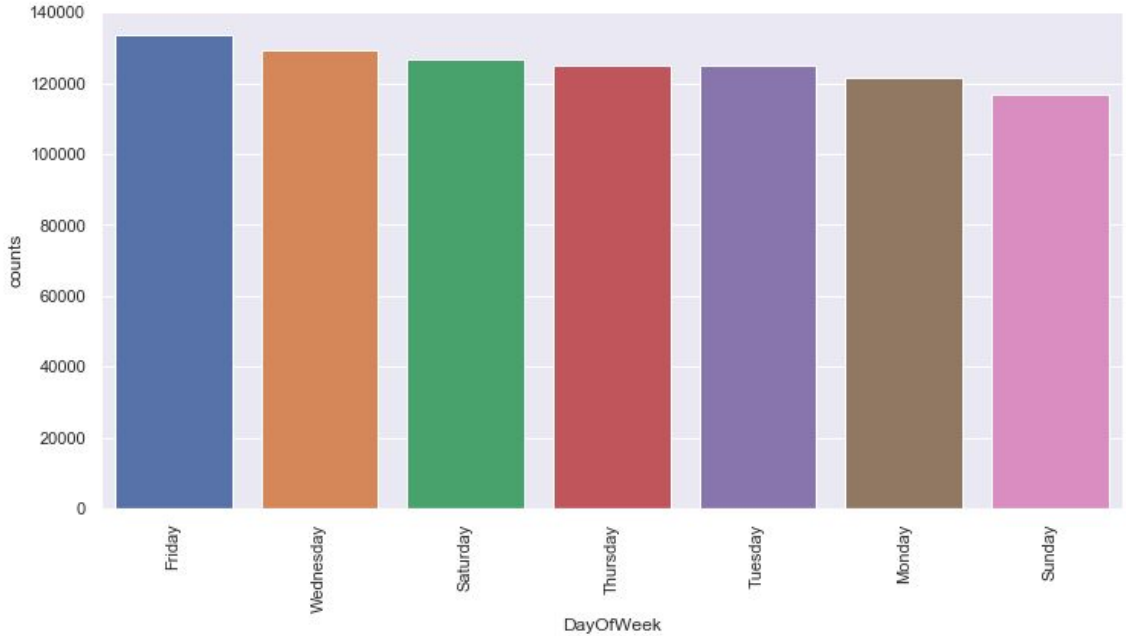
Police District

- 10 police districts
- Probably correlated with longitude and latitude
- Southern district dominates



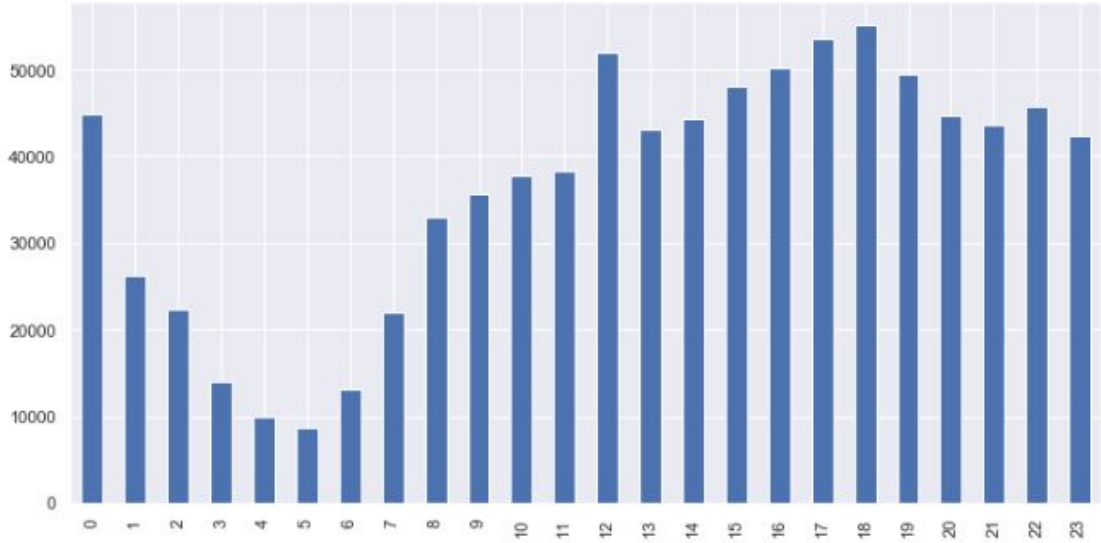
DayOfWeek

- Categorical variable with 7 levels
- Looks like number of reported crimes is highest on Friday
- Highest vs lowest is about 12% decrease



Dates

- Dates can be transformed into various features
- Binary, categorical, cyclical
- For hours-of-day it looks like there were some peaks around lunch time, and during evening rush hour



Takeaways

- We are working with static data, no more incoming data
- Our data reflects reported crimes, rather than actual crimes
- We have enough data to run most models
- An abundance of features; need feature selection
- Strategies to address:
    - PCA to project down to fewer dimensions
    - Choose a model that deals well with skewed data / long tail

Feature Engineering

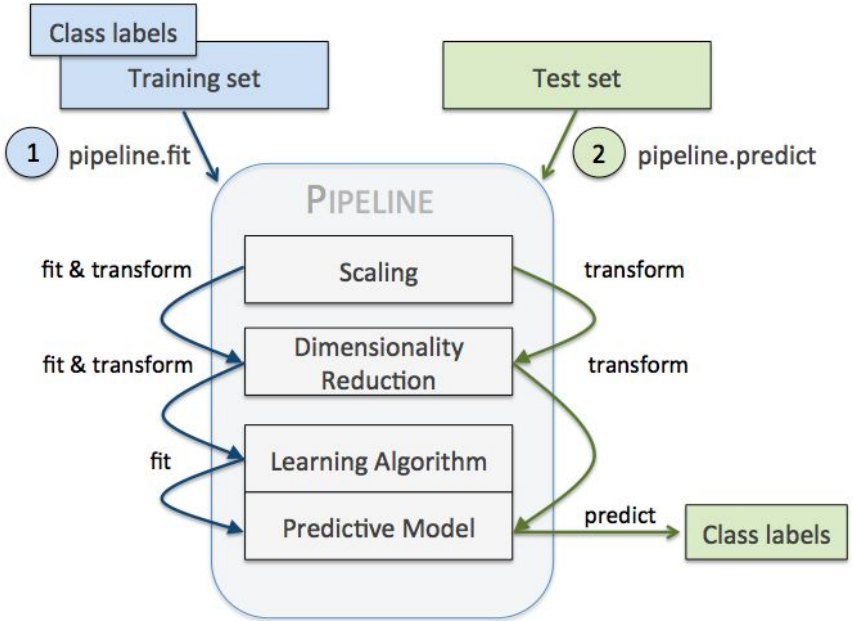
Feature Engineering Approach

Invested into building a feature extraction pipeline

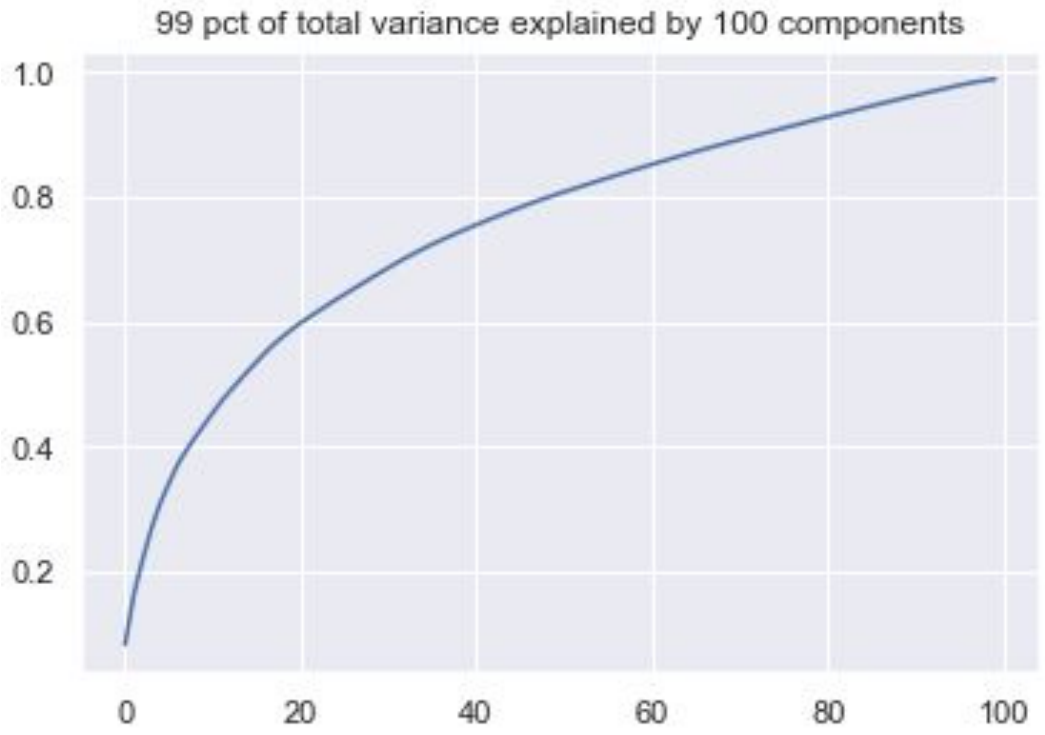
- From 6 base features
- To over 37 thousand features

Pipelines and custom transformers made it easier to produce results

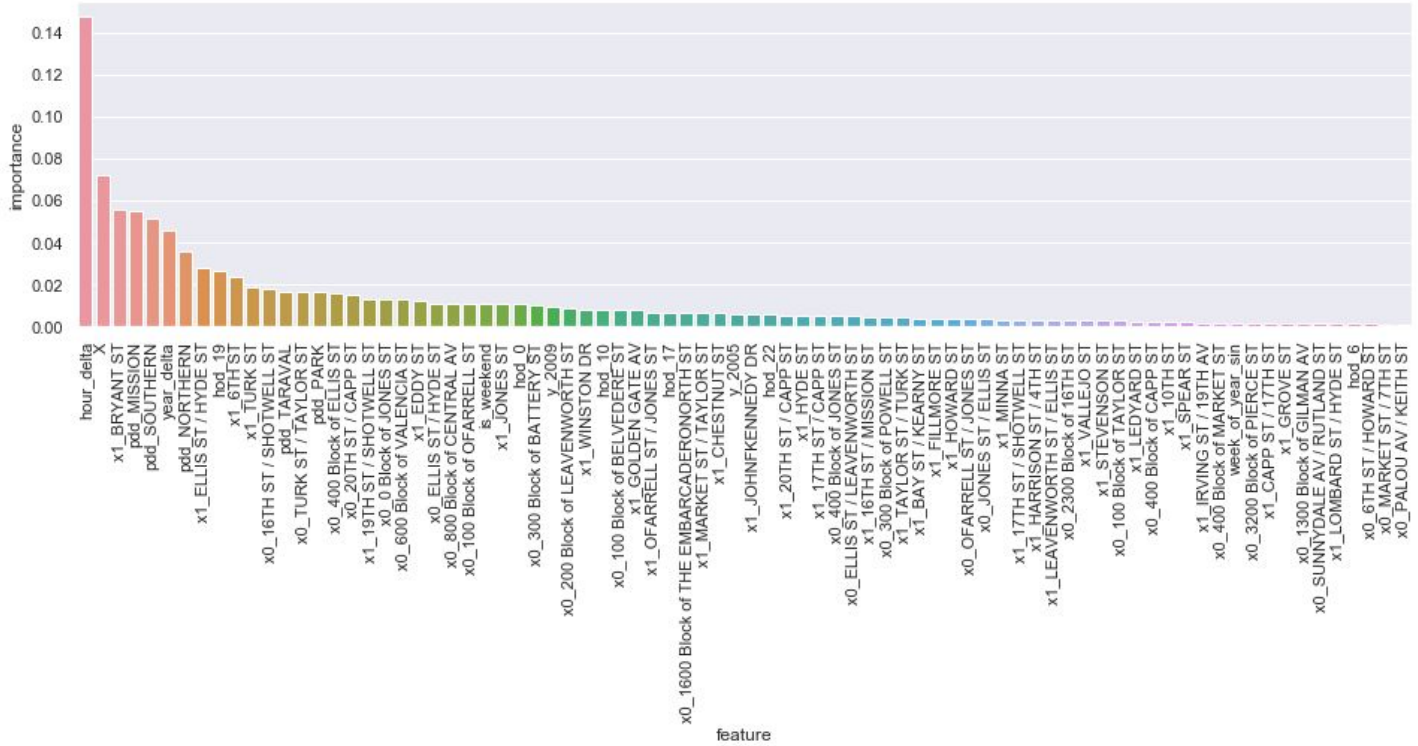
- Handled scaling, imputing, and other transformations in a repeatable manner
- Centralized repetitive code
- Pipe transformed features into grid-search-cross-validation



PCA


99 pct of total variance explained by 100 components
1.0
0.8
0.6
0.4
0.2
0
20
40
60
80
100

Random Forest & Feature Importance



Model Selection and Tuning

Choosing models

Methodology:

- Decentralized exploration
- GridSearchCV
  - scoring='neg_log_loss'
  - Optimizing
    - Hyperparameters
    - Manual Choices of variable
    - PCA
    - Cross-Validation

Models:

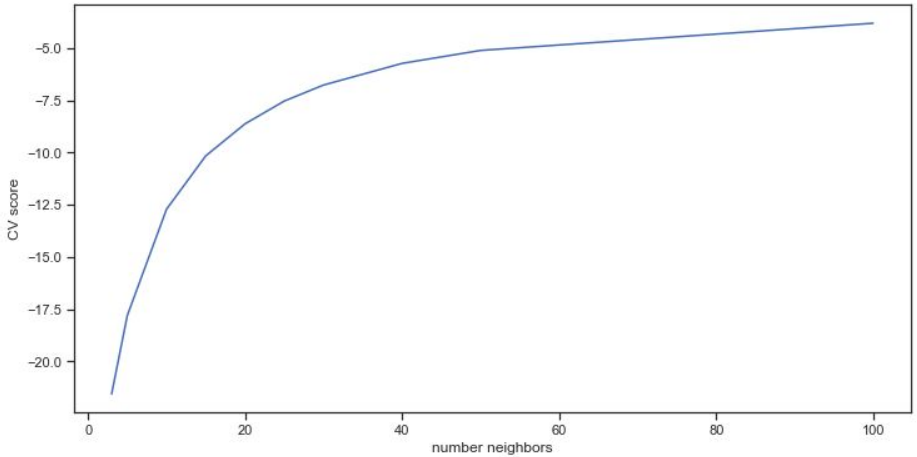
- KNN
- Logistic Regression
- Random Forest
- Gradient Boosted Tree
- Multi-Layer Perceptron

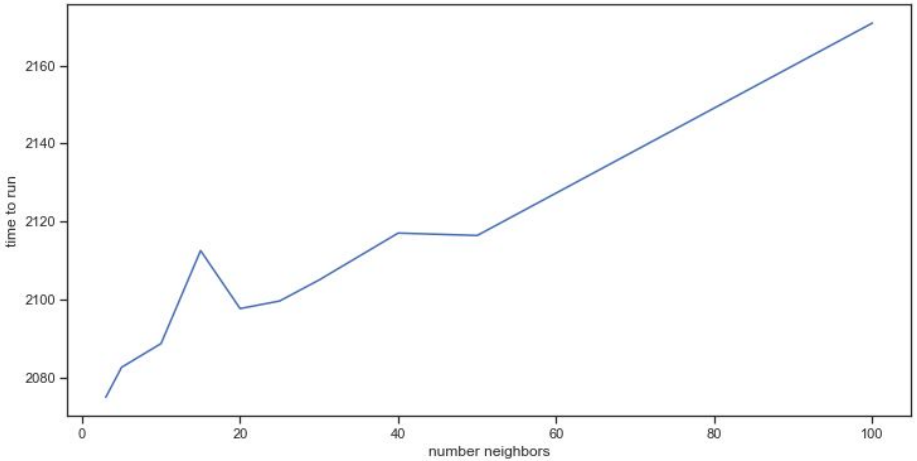
KNN

PCA

```
Best parameter (CV score=-4.389):
{'n_neighbors': 100}
```

sklearn.neighbors.KNeighborsClassifier





- After 80 neighbors, the incremental gain in the CV score is narrowing
- Best CV score at -3.802

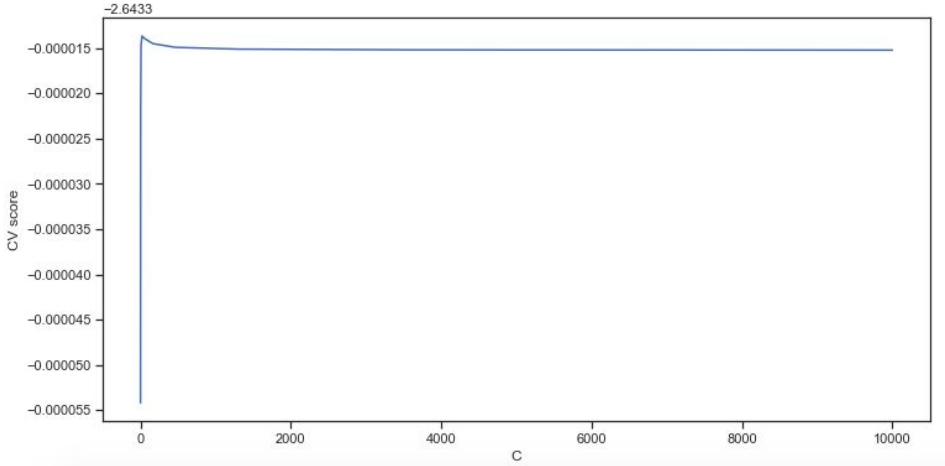
- The more neighbors you run, the more time it takes.
- Lowest time is 2080 seconds = ~34 minutes

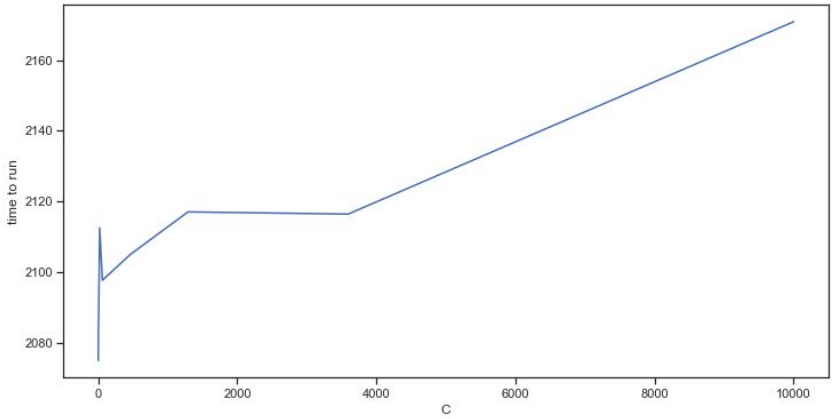
Logistic Regression

PCA

```
Best parameter (CV score=-2.681):
{'C': 0.01, 'penalty': 'l1'}
```

`sklearn.linear_model`.LogisticRegression





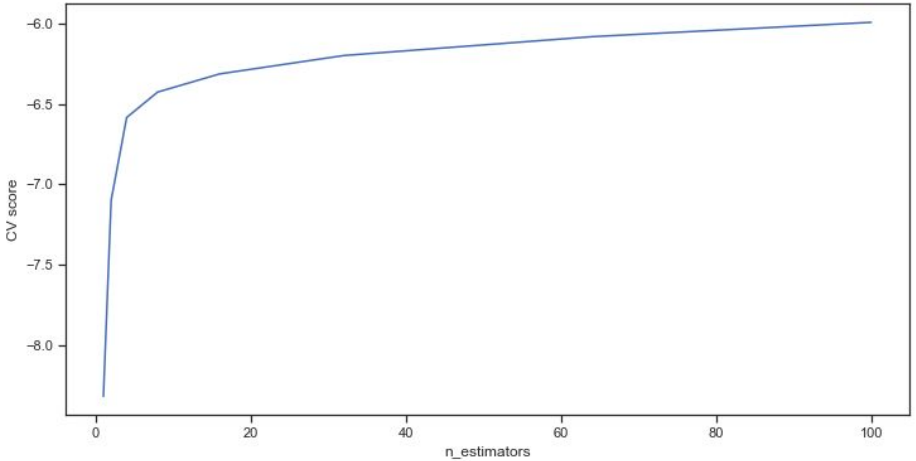
- Little or no gain on tuning C
- Lowest CV score of -2.6433136686256247

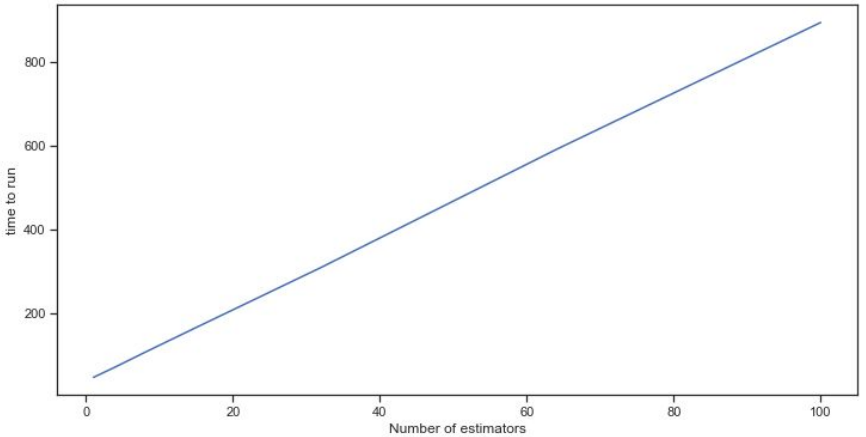
- The bigger the C, the more time it takes to run
- The lowest run time was of 2082 seconds = ~ 34.7 minutes

Random Forest

PCA

`sklearn.ensemble`.RandomForestClassifier





- Little or no gain after number of estimators after 60
- Lowest CV score of -5.994343979788256

- The bigger the number of estimators, the more time it takes to run
- The lowest run time was of 48.31 seconds

Gradient Boosted Tree

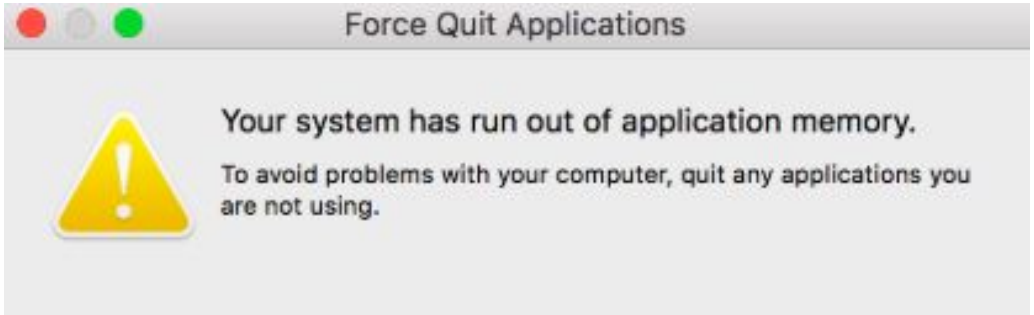
PCA

`sklearn.ensemble`.GradientBoostingClassifier

Features:
* longitude
* latitude
* is late night

Parameters: default settings

- Takes too long to run iterations.
  - We were able to run just one time before it crashed the computer
- Best CV score: -2.493
  - Works well with skewed data, which is the case



Gradient Boosted Tree [Placeholder for Results]

`sklearn.ensemble`.GradientBoostingClassifier

Features:
* longitude, latitude
* longitude, latitude, is_latenight
* longitude, latitude, is_latenight, is_weekend
* longitude, latitude, is_latenight, is_weekend, month_of_year, one-hot police district

Hyperparameters:
* n_estimators: [100, 200, 500, 1000]
* max_depth: [2, 3, 5, 10, 20, 30]

Best CV score: X

ADD PLOTS IF MODEL ACTUALLY FINISHES RUNNING

MultiLayer Perceptron

`sklearn.neural_network`.MLPClassifier

Key model parameters

Features:
longitude, latitude, is_weekend, is_latenight, month of year, one-hot police district

Hyperparameters:
3 hidden layers of 100, 50, 100 nodes

Pros

- Feature selection and engineering matters less
- Allows non-linear decision boundaries
- Universal function approximator which can be used (in theory) to map any given inputs to output

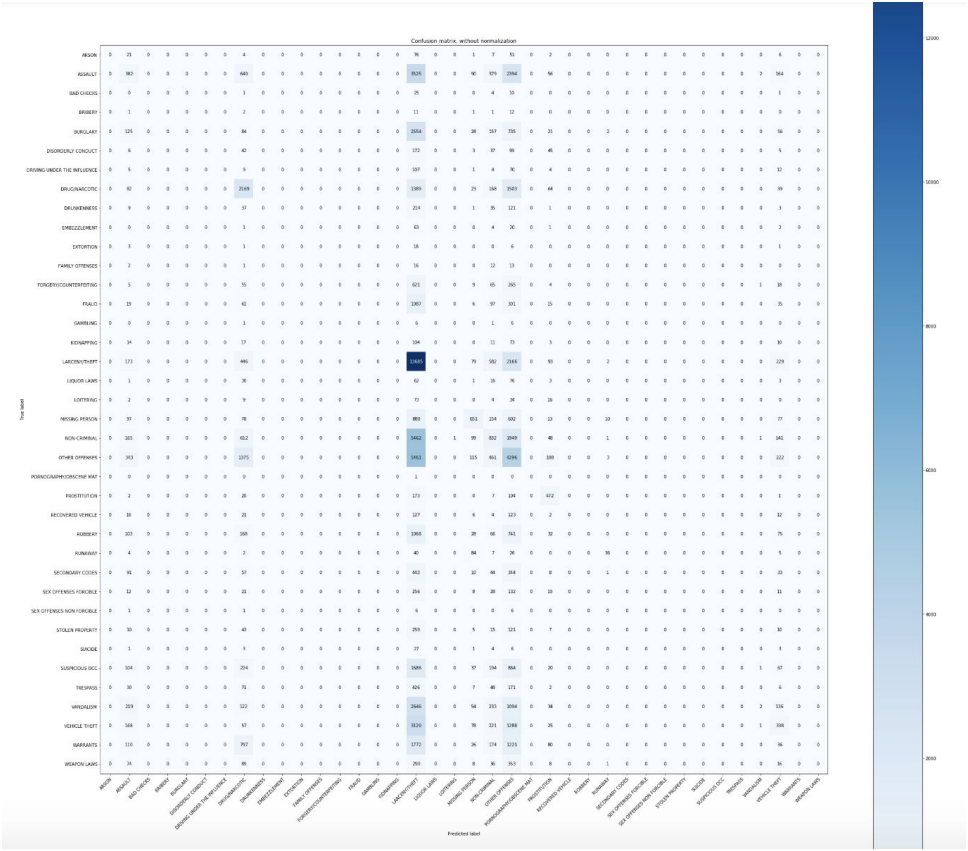
Cons

- Training is computational-resource-heavy
- Not guaranteed to converge to a global minimum

Lowest CV score: -2.464 | Runtime: ~16 hours

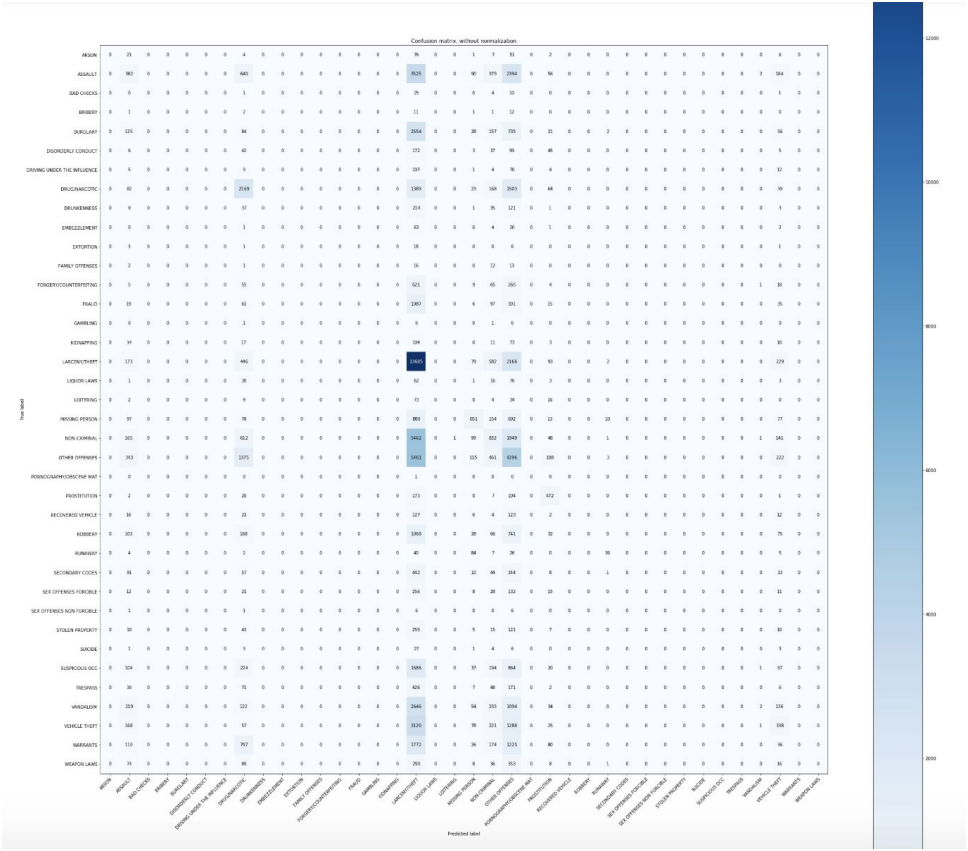
Model Validation

- Split into train (90%) and dev sets (10%)
- Confusion matrix >
  - Most dev data were miscategorized as either:
    - theft/larceny
    - other offenses
    - drug/narcotic
  - Usefulness was limited
    - Not enough time for much tuning and rerunning
- Log loss comparison
  - train_logloss: -2.464
  - dev_logloss: -2.475



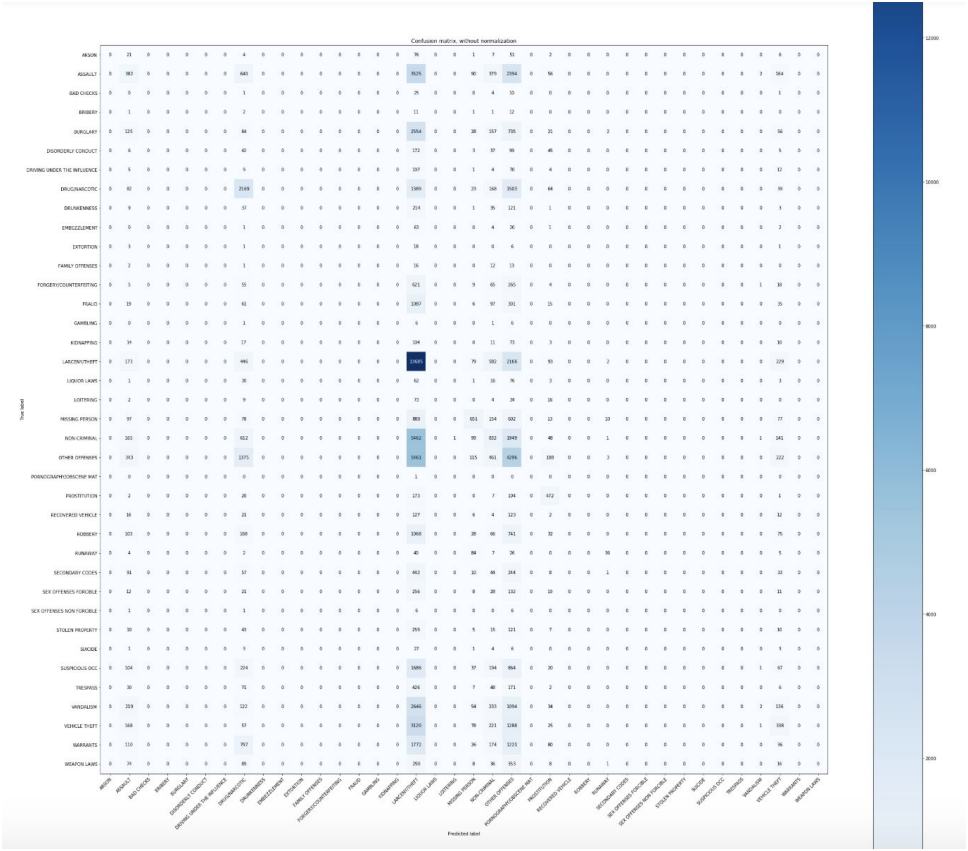
Model Validation

- Split into train (90%) and dev sets (10%)
- Confusion matrix >
  - Most dev data were miscategorized as either:
    - theft/larceny #1
    - other offenses
    - drug/narcotic
  - Usefulness was limited
    - Not enough time for much tuning and rerunning
- Log loss comparison
  - train_logloss: -2.464
  - dev_logloss: -2.475



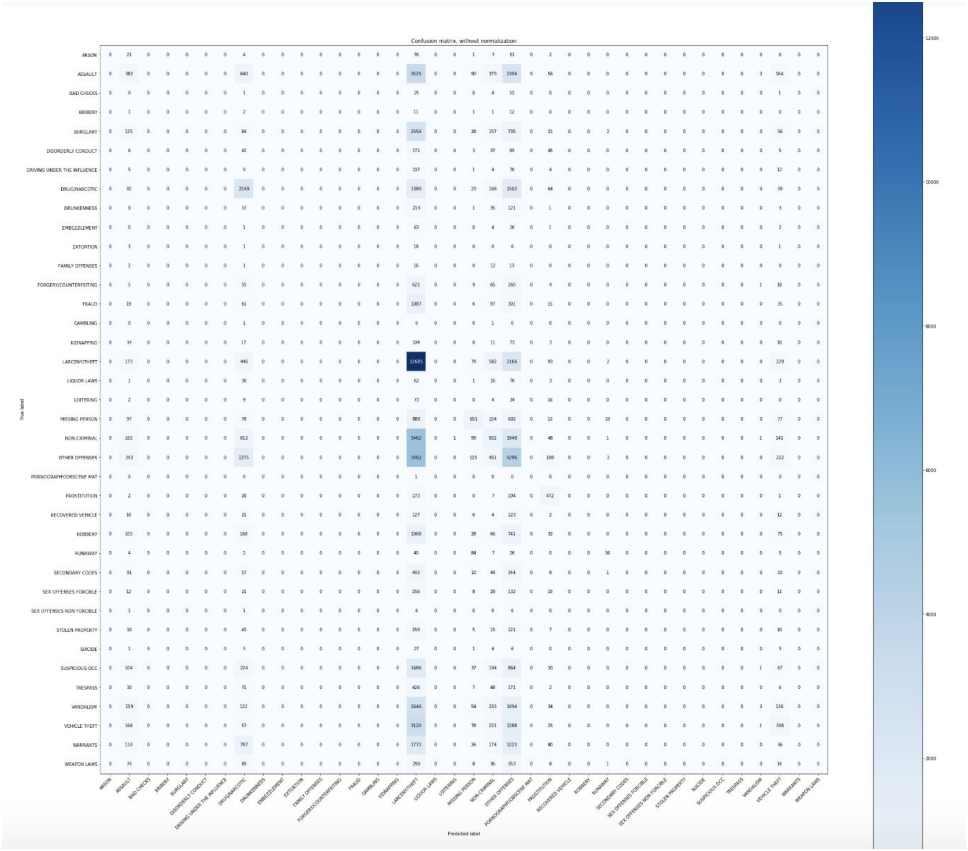
Model Validation

- Split into train (90%) and dev sets (10%)
- Confusion matrix >
  - Most dev data were miscategorized as either:
    - theft/larceny #1
    - other offenses #2
    - drug/narcotic
  - Usefulness was limited
    - Not enough time for much tuning and rerunning
- Log loss comparison
  - train_logloss: -2.464
  - dev_logloss: -2.475



Model Validation

- Split into train (90%) and dev sets (10%)
- Confusion matrix >
  - Most dev data were miscategorized as either:
    - theft/larceny #1
    - other offenses #2
    - drug/narcotic #5
  - Usefulness was limited
    - Not enough time for much tuning and rerunning
- Log loss comparison
  - train_logloss: -2.464
  - dev_logloss: -2.475

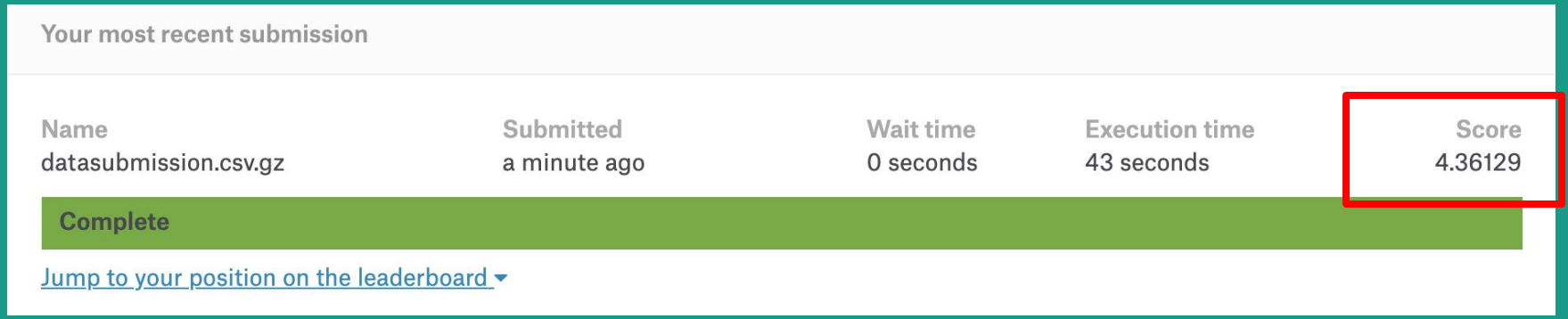


Results

Results - CV Scores

MODEL	CV Score with raw features	CV Score with PCA
KNN	-3.802	-4.389
Logistic Regression	-2.643	-2.681
Random Forest	-5.994	-
Gradient Boosted Tree	-2.493	-
MultiLayer Perceptron	-2.484	-

Submission Results



Submission Results

kaggle log loss: 4.361

train_logloss: -2.464

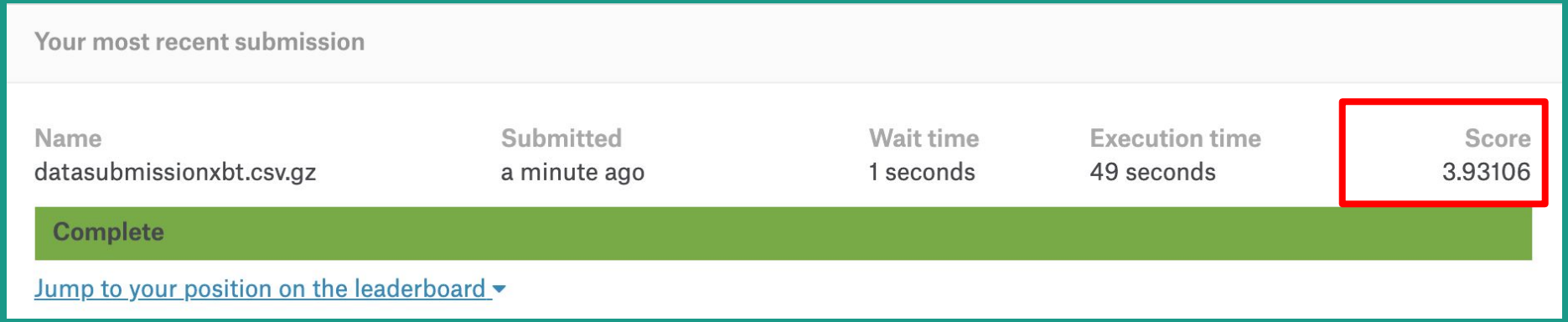
dev_logloss: -2.475

- Looks like we overfit
- Strategies to addressing overfitting
  - Introduce regularization / penalties (L1, L2) for weights
    - This isn't a configurable parameter for MLP
    - Probably better off with a different model
  - Get more data, especially from underrepresented classes
    - Oversample from minority classes
    - Incorporate supplemental data sources

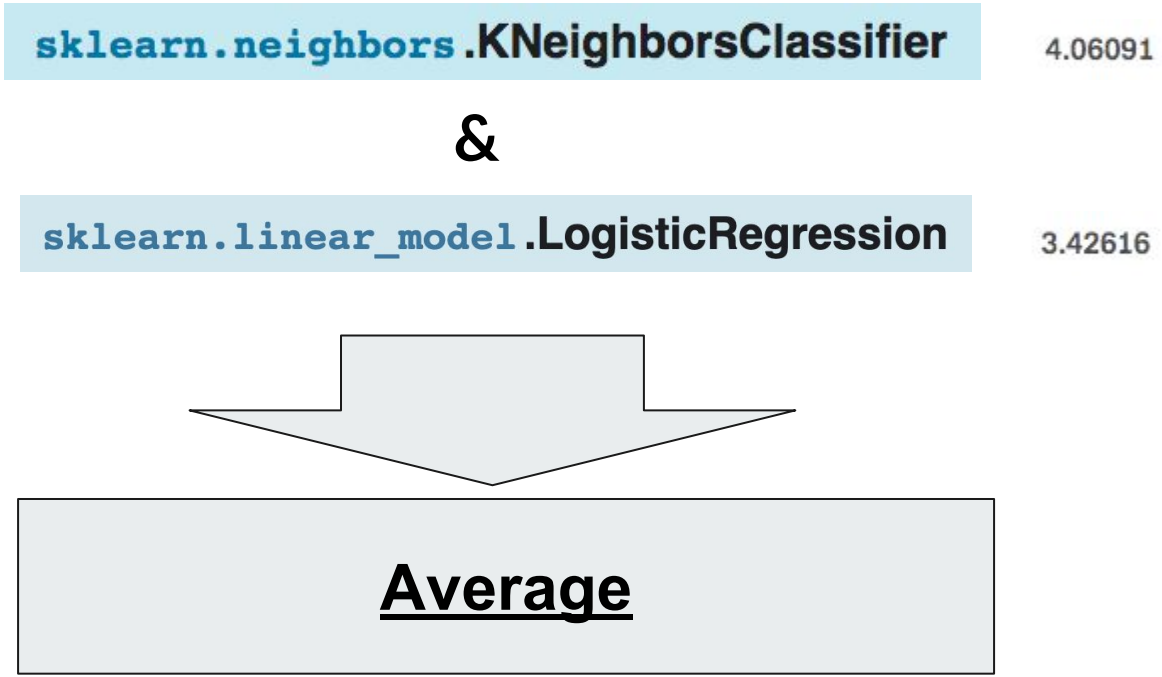
Results - Kaggle Scores

MODEL	CV Score with raw features	CV Score with PCA	Kaggle score
Logistic Regression	-2.643	-2.681	3.42
Gradient Boosted Tree	-2.493	-	3.93
MultiLayer Perceptron	-2.484	-	4.36
Voting (average of KNN+PCA, log+PCA)	-	-	**3.23**

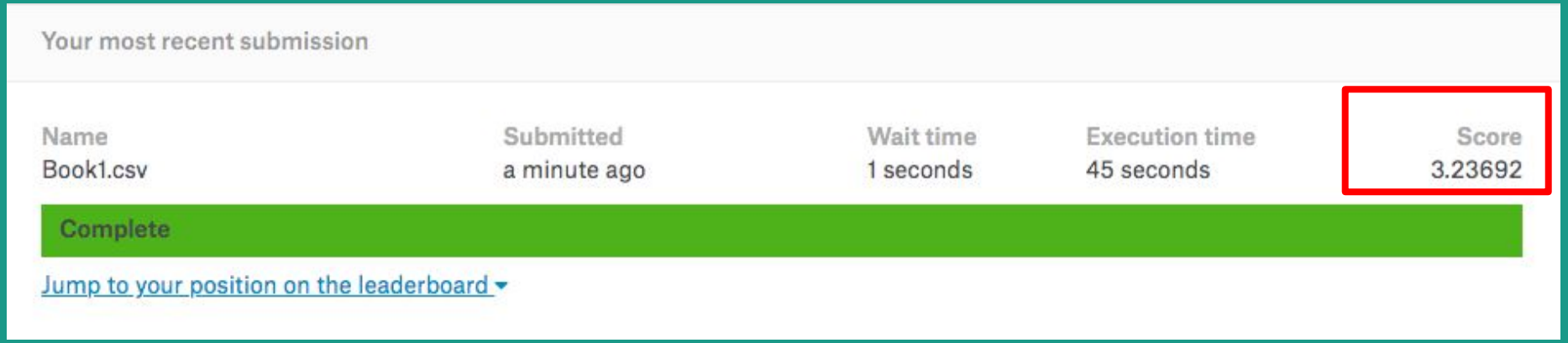
Submission Results



Methodologies "Voting"



Submission Results



Evolution of our Kaggle score

Baseline	Mid-Term				Submission
22.1	7.6	4.3	3.9	3.4	3.2

Lessons Learned

- Computation power matters (get a cluster!)
  - Model fitting takes time with full training sets
  - We can't always use mini train sets because some models need lots of data to perform well
- Start early!
  - Even with relatively manageable data size and feature set, we didn't get a chance to tune models fully
  - Explore supplemental data sources
- PCA helps with feature selection
- **Building a pipeline** helps teammates scale!
  - Improves code cleanliness and standardization
  - Reduces downstream toil

Thank you!

Questions?