Multi-class classification problem: Crimes in San-Francisco

Galina Alperovich

Abstract—From a dataset with nearly 12 years of crime reports from across all of San Francisco's neighborhoods, with given time and location, we were trying to predict the category of crime that occurred. The tasks of classification was preceded by the data processing and feature engineering. Several classification techniques are considered, such as Neural Networks, k-nearest neighbors, Logistic Regression, Random Forest, Gradient Boosting.

Index Terms—machine learning, data mining, multi-class classification

1 Introduction

KAGGLE competitions [1] are the popular activities in a data science community. It is a platform for predictive modelling and analytics competitions where companies and researchers post their data, and then statisticians and data miners from all over the world compete to produce the best models.

In a current work we have considered one of these Kaggle's competitions which is called "San Francisco Crime Classification" [2]. It is a multi-class classification problem where with a given long history of crime records (12 years) competitors were supposed to predict crime category.

Usually for such tasks many approaches and algorithms should be experimented with. Also before experimenting with different approaches we have made standard procedures as outlier detection, missing data imputation and scaling. It is very important to properly prepare data in order to get reasonable results (one of the rule in computer science: garbage in - garbage out). As one of the preparation steps we have extracted many new features from original dataset, and it gave us large increase in a evaluation score.

Detailed pre-processing procedures and selected algorithms are described below.

2 Working with dataset

Considered dataset contains incidents derived from SFPD Crime Incident Reporting system. The data ranges from 1/1/2003 to 5/13/2015. The training set and test set rotate every week, meaning week 1,3,5,7... belong to test set, week 2,4,6,8 belong to training set. Both train and test set have around 800 000 records.

In original train dataset there were the following features:

1) **Datetime stamp**
2) **X and Y coordinates (latitude and longitude)**
3) **Police department**
4) **Weekday**
5) Intersections of streets
6) SF District
7) Crime decision (arrested/booked/...)

• *E-mail: shchegal@fel.cvut.cz*

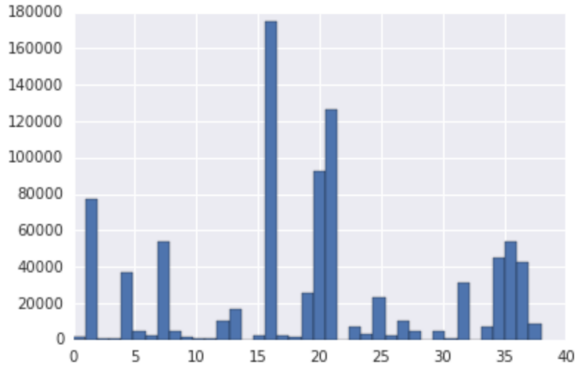

Fig. 1: Distribution of crimes in SF (names are encoded by numbers)

Features which are selected with a bold font are presented in a train set.

In order to prepare original dataset we needed to process data firstly. After the first careful look into dataset several things were found: some categories had only 1-2 records which is not a representative sample, so it was not possible to use these information for prediction and the decision was to remove such category. There were an outliers in X and Y coordinates. Also data originally sorted by a datetime, so we needed to shuffle it in cross-validation step.

2.1 Data visualization and exploration

Here are presented some images which help to understand data better.

On the figure 1 you can see crime category distribution. It is clear that it is not uniform and some categories are extremely frequent and some of them otherwise less

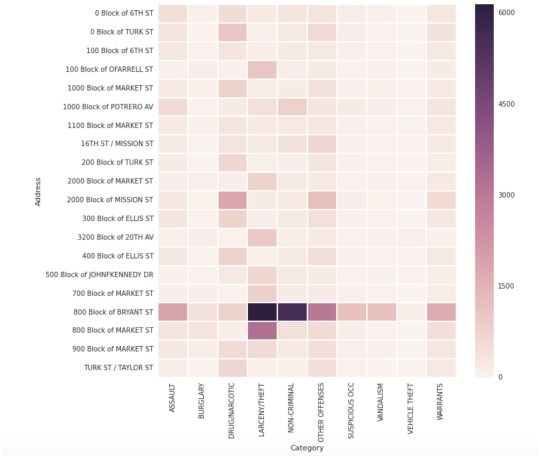

Fig. 2: Crime category vs Address.

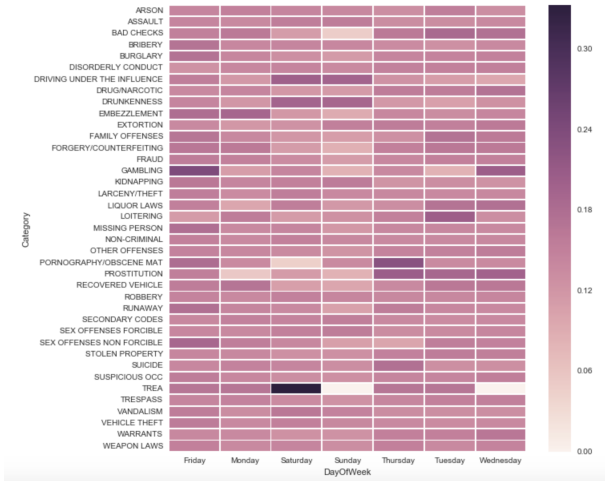

Fig. 3: Crime category vs Weekday.

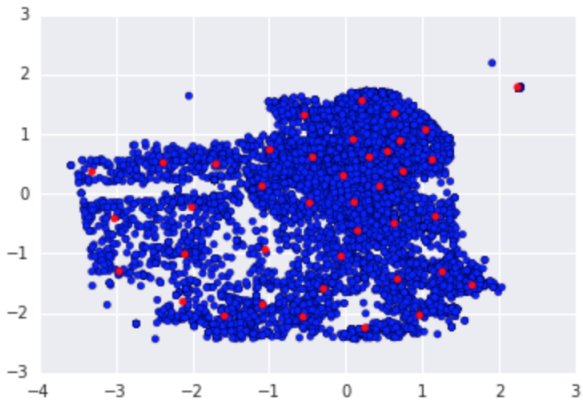

Fig. 4: Clustering of X and Y coordinates, number of clusters is 40

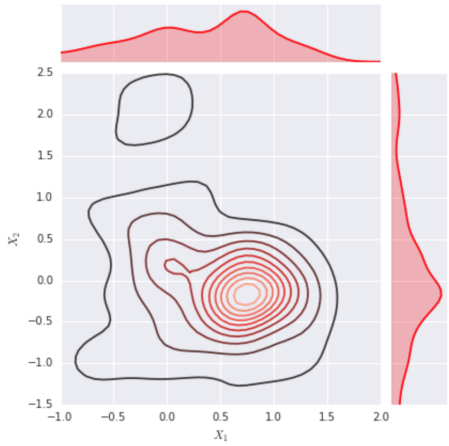

Fig. 5: PCA transformation on X and Y coordinates: first 2 components

populated. The most frequent categories as follows: 'LARCENY/THEFT', 'OTHER OFFENSES', 'NON-CRIMINAL', 'ASSAULT'.

On the figure 2 we can see a heatmap of two variables: address and crime category. From this picture we can conclude that some of locations have very high criminal activity and some of them not. Also different crimes are concentrated in a differnet locations.

On the figure 3 we can see heatmap of crime category and weekday. It can be seen that some crimes are concentrated in a different weekdays. For example TREA (Trespassing or loitering near posted industrial property) is highly frequent on Saturday, GAMBLING on Friday, DRUNKENNESS on Saturday. Some of days are highly criminal in all types of crime, for example Friday.

2.2 Feature extraction

We have extracted features from three original fields: Datetime, both X and Y coordinates.

There are 5 extracted features from datetime:

- year
- hour
- month
- day
- absolute value of minutes minus 30

From X and Y coordinates 52 features have been extracted:

- three variants of rotated Cartesian coordinates (rotated by 30, 45, 60 degree each X and Y)
- Polar coordinates (i.e. the 'r' and the angle 'theta')
- centers of 40 clusters and 40 distances to all centers (see figure 4)
- 2 first components after PCA transformation (see figure 5)

With these additional features we get more spatial information than just raw X and Y.

2.3 Data processing

After extracting the features we standardized them by removing the mean and scaling to unit variance. Centering

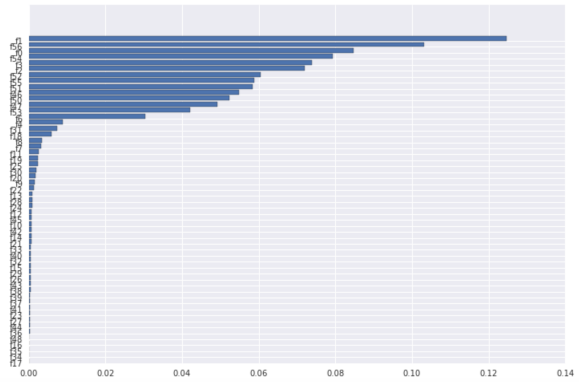

Fig. 6: Feature importance based on XGBoost algorithm

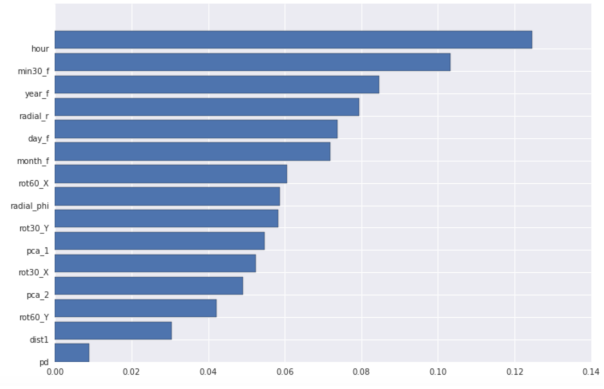

Fig. 7: Feature importance based on XGBoost algorithm, Top-15 features

and scaling happen independently on each feature by computing the relevant statistics on the samples in the training set.

Also we have imputed outliers in a X,Y coordinates with mean values.

2.4 Feature importance

There are many approaches how to estimate importance of the features in a dataset. Since we have tried Gradient Boosting algorithm we can use feature importance from the weight value which is the number of times a feature is used to split the data across all trees.
After running Gradient Boosting algorithms with 15 decision trees, we have got the following estimation of features importance (see figure 6).

15 the most important features as follows: hour, min30_f, year_f, radial_r, day_f, month_f, rot60_X, radial_phi, rot30_Y, pca_1, rot30_X, pca_2, rot60_Y, dist1, pd. Description of each feature can be seen in Appendix.

Feature min30_f is a second main feature describing the fact that some crimes can be mapped to spacetime precisely and some can be mapped only approximately. min30_f is also reflecting the fact of time distribution symmetry.

3 Evaluation and Error Metric

Submissions for the competition are evaluated using the multi-class logarithmic loss [4]. For each incident, we must submit a set of predicted probabilities (one for every class). On the Kaggle side each incident has been labeled with one true class. The formula is then,

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

where N is the number of cases in the test set, M is the number of class labels, $\log$ is the natural logarithm, y_{ij} is 1 if observation i is in class j and 0 otherwise, and p_{ij} is the predicted probability that observation i belongs to class j.

The resulting probabilities for a given incident are not required to sum to one because they are rescaled prior to being scored (each row is divided by the row sum). In order to avoid the extremes of the log function, predicted probabilities are replaced with $\max(\min(p, 1-10^{-15}), 10^{-15})$

4 Neural Networks

In our work we tried many different configurations of the neural networks with different parameters. Keras framework for Python was used. We have experimented with the following parameters:

- Number of layers
- Activation functions
- Dropout value
- Number of neurons in each layer
- Weight and activity regularization
- Learning rate in SGD

Stochastic gradient descent was used as an optimization method and back-propagation as learning algorithm.

4.0.1 *Input and output layers*

Input data of the neural network was 46 spatial and datetime-related features which were described in details before (46 features instead of 57 because models have been running before rotated modification were introduced). Number of neurons in input layer varied in the experiments from 70 to 500. Number of neurons in output layer is fixed and equals 39 (number of classes). Number of hidden layers varied from 2 to 5.

4.0.2 *Topology and neuron types*

The best result with 5-fold cross-validation gave network with the following specification (see Python code):

```
from keras.layers import Dense, Activation, Dropout
from keras.models import Sequential
from keras.optimizers import SGD

model = Sequential()
model.add(Dense(300, input_dim=46, init='uniform'))
model.add(Activation('tanh'))
model.add(Dropout(0.5))
model.add(Dense(100, init='uniform'))
model.add(Activation('tanh'))
model.add(Dropout(0.5))
model.add(Dense(39, init='uniform'))
```

```
model.add(Activation('softmax'))

sgd = SGD(lr=0.1, decay=1e-6, momentum=0.9, nesterov
    =True)
model.compile(loss='categorical_crossentropy',
              optimizer=sgd,
              metrics=['accuracy'])
```

Input layer has 300 neurons, one hidden layer has 100 neurons, activation functions are tangents for input and hidden layers and softmax for output layer, value of dropouts is 0.5. SGD optimizer with learning rate 0.1, momentum 0.9.

4.1 Early stopping

In order to avoid overfitting we implemented Early Stopping callback which allows to track error on validation set, stop fitting earlier if it is not increasing after 5 epochs and save intermediate weights to the file.

4.2 Comparison of different specifications

Comparison of different specification can be seen in a publicly available Google document. The best score gave network with specifications described above and multi-class logloss value is 2.68 (cross-validated mean) and on public leaderboard is 2.69.

5 Other classification algorithms

Besides Neural network approach we also tried several other algorithms with different parameters:

- KNN for 2, 4, 6, 16, 32 and 64 neighbours (euclidian distance)
- Random forest (with different number of trees)
- Logistic regression
- Gradient Boosting (many parameters were tried)

6 Experiments

6.1 Working environment

Some of the experiments were conducted in a local machine 2.5 GHz Intel Core i5, 8 GB 1600 MHz DDR3, Mac OSX, 10.10.5.

Vast majority of experiments were conducted on MetaCentrum Virtual Organization [5] which operates and manages distributed computing infrastructure consisting of computing and storage resources.

A total of **86** jobs on MetaCentrum were started and around **20** jobs on the local machine.

Working programming language is Python, integrated development environment are PyCharm and Jupyter (IPython notebook). The most used Python libraries are keras, xgboost, numpy, scikit-learn, pandas for computation and matplotlib with seaborn for vizualization.

6.2 Cross-validation

In our experiments we have used stratified 5-fold cross-validation procedure in order to avoid overfitting.

6.3 Comparison of different classifiers

Comparing different algorithms for this classification task we can look at final multi-class logloss values for all of them:

Algorithm	Multi-class logloss
Neural Network	2.68
Logistic regression	2.69
Gradient Boosting	**2.31**
Random forest	2.38
KNN (for all k)	> 10

TABLE 1: Multi-class logloss for final models

7 Discussions

Usually Kaggle competition is a very toe-to-toe, all competitors are fighting for a thousandth of an evaluation score, and it is very difficult to compete. Also usually complicated ensemble models are used which require huge computational resources.

In some of our experiments we have used blending techniques which take several classifiers output and combine them with some coefficients but the final score was not better than Gradient Boosting model itself. More experiments with ensembles are needed, they are the key for a place in a top results on Kaggle leaderboard. Actually Gradient Boosting itself is an ensemble of weak decision tree classifiers, that is why it is very often gives the best result: more than half of the winning solutions in machine learning challenges hosted at Kaggle adopt XGBoost [6].

Despite of a big number of experiments with neural networks we considered only small part of possible configurations for this tasks. More complicated architecture can be considered, as well as greater number of neurons, type of layers (for example recurrent since we have date dependent dataset). Probably blending of a more complicated network with Gradient Boosting will give higher score.

8 Conclusion

In current work we have been solving multi-class classification problem in one of the Kaggle competition. We processed given datasets, extracted new spatial and date-related features which gave a big increase in a score. Several algorithms with different parameters were considered including Neural Networks, k-nearest neighbors algorithms, Logistic Regression, Random Forest, XGBoost. In total around 100 experiments were conducted in MetaCentrum environment and local machine. To get higher place on a public Kaggle leaderboard it needed to combine different classifiers using ensemble techniques which requires a lot of computational resources.
Best result was given by Gradient Boosting algorithm, final multi-class logloss score is 2.31527 and it is 281 place out of 2111.

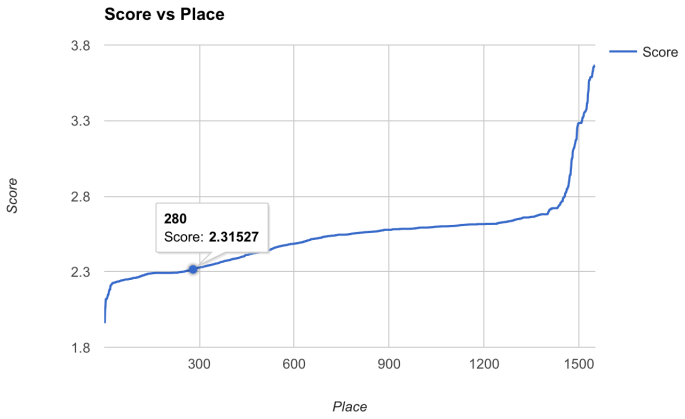

Fig. 8: Score vs Place in a Public leaderboard. Final score of current work is shown.

APPENDIX A
FEATURES LIST IN AN ORIGINAL DATASET

Datetime
X coordinate
Y coordinate
Weekday
Police department

TABLE 2: Features list in an original dataset

APPENDIX B
FEATURES LIST IN A PROCESSED DATASET

year	hour
month	day
police department (encoded)	closest cluster center (encoded)
distance to 1st cluster	distance to 2nd cluster
. . .	distance to 40th cluster
first PCA component	second PCA component
rot. 45 of X	rot. 45 of Y
rot. 30 of X	rot. 30 of Y
rot. 60 of X	rot. 60 of Y
radial r	radial phi
abs(min - 30)	

TABLE 3: Features list after dataset processing

REFERENCES

[1] Kaggle competitions website: https://www.kaggle.com
[2] The page of "San Francisco Crime Classification" task: https://www.kaggle.com/c/sf-crime
[3] Documentation for boosting trees: https://xgboost.readthedocs.io/en/latest/model.html
Reference API: http://xgboost.readthedocs.io/en/latest/python/python_api.html
[4] Logarithmic loss also called cross-entropy: https://en.wikipedia.org/wiki/Cross_entropy
https://www.kaggle.com/wiki/MultiClassLogLoss
[5] MetaCentrum VO website: http://metavo.metacentrum.cz/en/
[6] XGBoost: Implementing the Winningest Kaggle Algorithm in Spark and Flink: http://www.kdnuggets.com/2016/03/xgboost-implementing-winningest-kaggle-algorithm-spark-flink.html