Presentation of Kaggle competition ”San Francisco Crime Classification”

Emil Melnikov, Zilya Yagafarova
Supervisor: Kamil Salakhiev

Innopolis University

Competition and dataset

A competition ”San Francisco Crime Classification” is about prediction the category of crimes that occurred in the city by the bay (`https://www.kaggle.com/c/sf-crime`). The dataset contains incidents derived from SFPD Crime Incident Reproting System. There are 9 columns:

Dates - timestamp of the crime incident

Category - category of the crime incident (only in train.csv). This is the target variable we are going to predict.

Descript - detailed description of the crime incident (only in train.csv)

DayOfWeek - the day of the week

PdDistrict - name of the Police Department District

Resolution - how the crime incident was resolved (only in train.csv)

Address - the approximate street address of the crime incident

X - Longitude

Y - Latitude

There are 878049 rows in train set and 884262 rows in test set.

Feature exploration

The crime distribution is presented on the Figure 1. There are 39 types of crimes in total, among which the top 5 frequent are theft, other offenses, non-criminal, assault, and drug/narcotic.

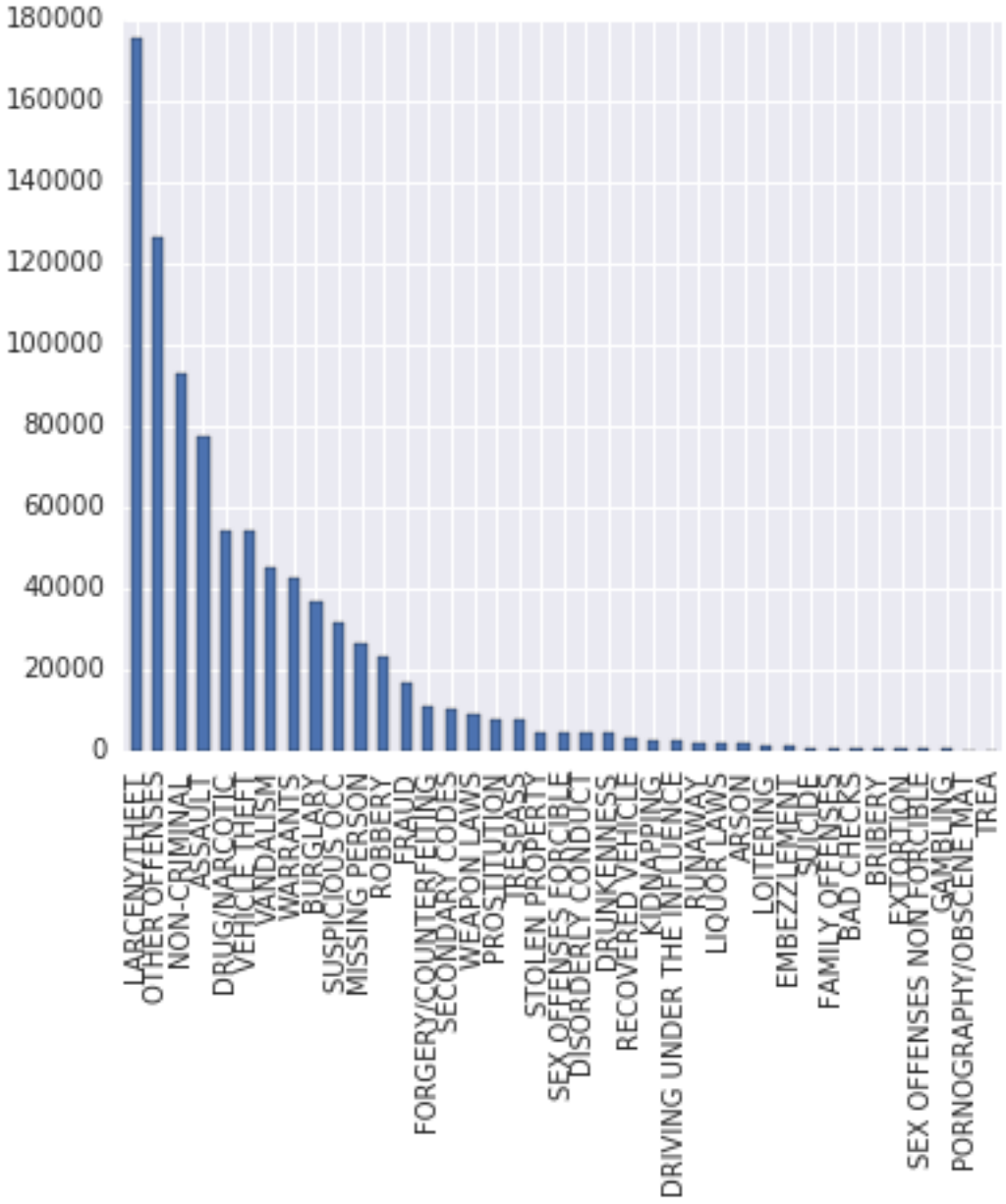

Figure 1: Crime distribution

We also looked at how the occurrences of crimes vary from different police department districts. From Figure 2 we could figure out that Southern Police Department District handles the most crimes, almost 4 times more than Richmond Police Department District.

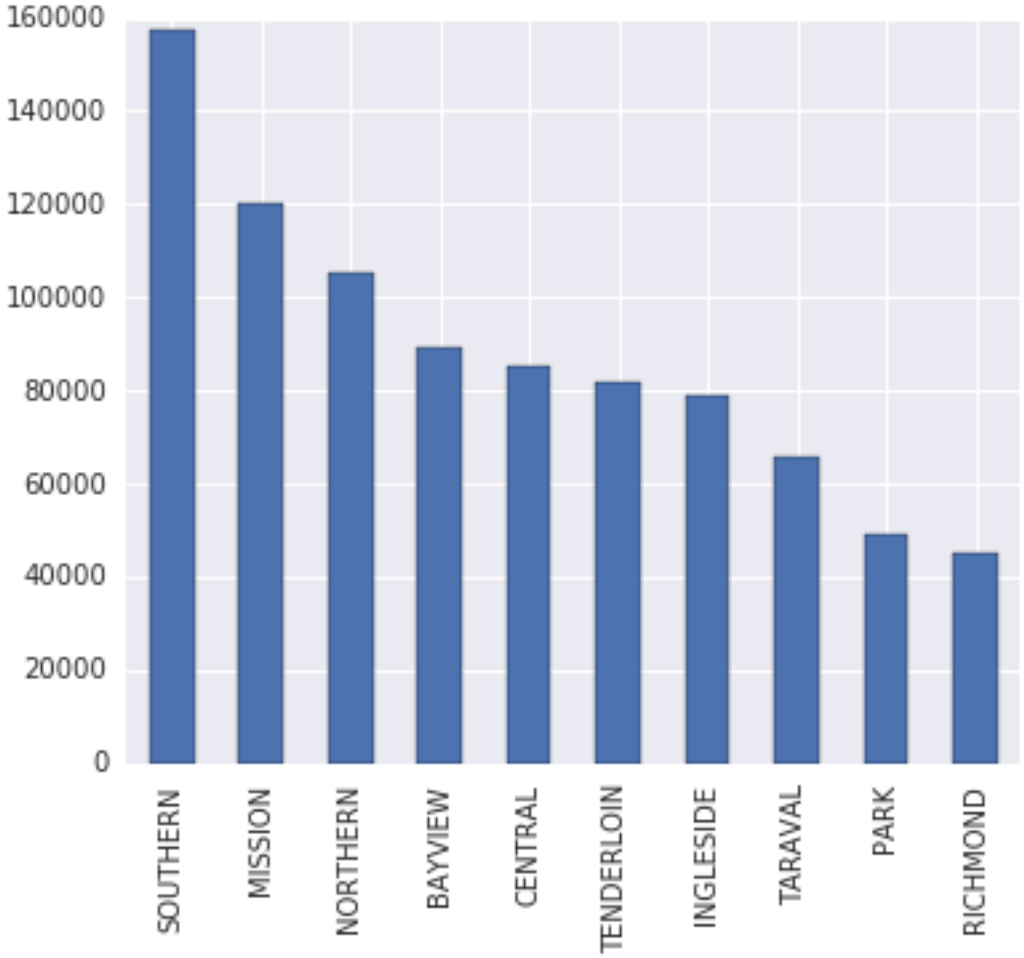

Figure 2: Police department district crime distribution

We can see the distribution of day of week on the Figure 3. The highest criminal occurrence was on Fridays and lowest was on Sundays.

Also, we've used Kaggle forums to find useful graphs and visualizations done by other competition participants. The following graphs were the most helpful for gaining insight into the data: time and spatial distributions, crimes by weekday, crimes by day/district/year, timeseries

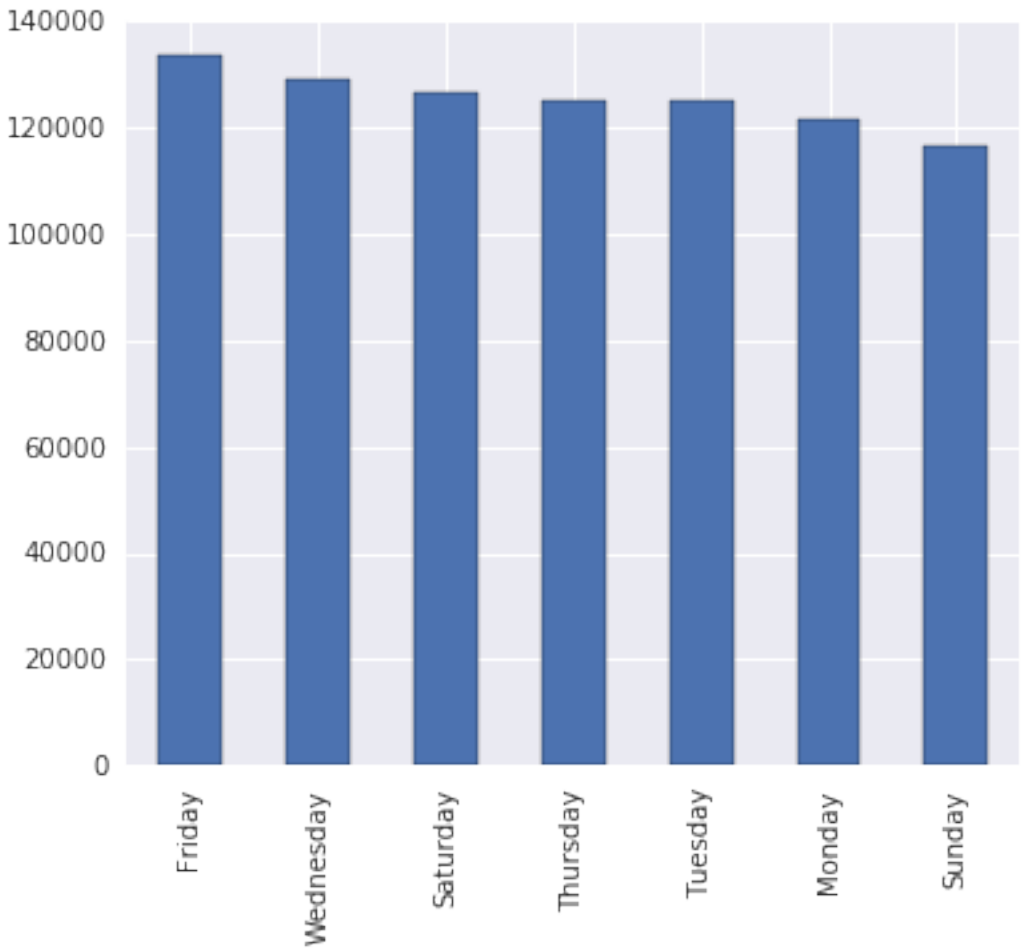

Figure 3: The amount of crime depending of day

and heatmaps).

Submissions evaluation

Submissions are evaluated using the multi-class logarithmic loss. Each incident has been labeled with one true class. For each incident, we must submit a set of predicted probabilities (one for every class). We must submit a csv file with the incident id, all candidate class names, and a probability for each class.

Feature selection

We adopted features:

- **Category:** we converted the category to integer values using the method LabelEncoder.
- **Hours, PdDistrict, DayOfWeek:** we converted the districts, weekday, and hour (from Dates) into binarized arrays using get_dummies method and combined them into a new dataframe.
- **Address:** we cleaned Address feature from numbers and tried to process it with one-hot and label encoders. After all, we stopped at LabelEncoder which gave better results probably because there are some dependencies between addresses.

We experimented with 2 combinations of feature sets:

1. Hours, PdDistrict, DayOfWeek.
2. Hours, PdDistrict, DayOfWeek, Address.

We decided not to use X and Y features because of there were a few hundred entries with Longitude and Latitude given as -120.5 and 90 respectively. The description of the incident and the resolution are both not used in the test data and they are not useful.

Model selection

We split the train data into a training and validation set with ratio 0.6 using Scikit-learns train_test_split function. We applied Naive Bayes classifier, AdaBoost Classifier, KNeighbors Classifier, Logistic Regression, Gradient Boosting Classifier and used as our accuracy measure the Logarithmic Loss metric. Table 1 demonstrates our results evaluated by log loss metric.

For all models we tried (except KNN and AdaBoost, which were either too slow or too inaccurate), we plotted learning curves (see Figures 4, 5, 6, 7, 8) to see whether a particular model would benefit from all our training data. Naive Bayes and Logistic Regression are too simple for our data, while Gradient Boosting Trees leave much room for improvement by increasing model complexity with increasing the number of boosted trees. This can be seen from the slow convergence of training and cross-validation scores. We've used standard accuracy score for the learning curves, because logarithmic loss for small dataset samples could miss some classes, resulting in error on test set. Nevertheless, accuracy score is working just fine for investigating bias-variance trade-off. We've used only a small subset of the training set because, otherwise, it would take too much time, and the general trend can still be observed even on these small subsets.

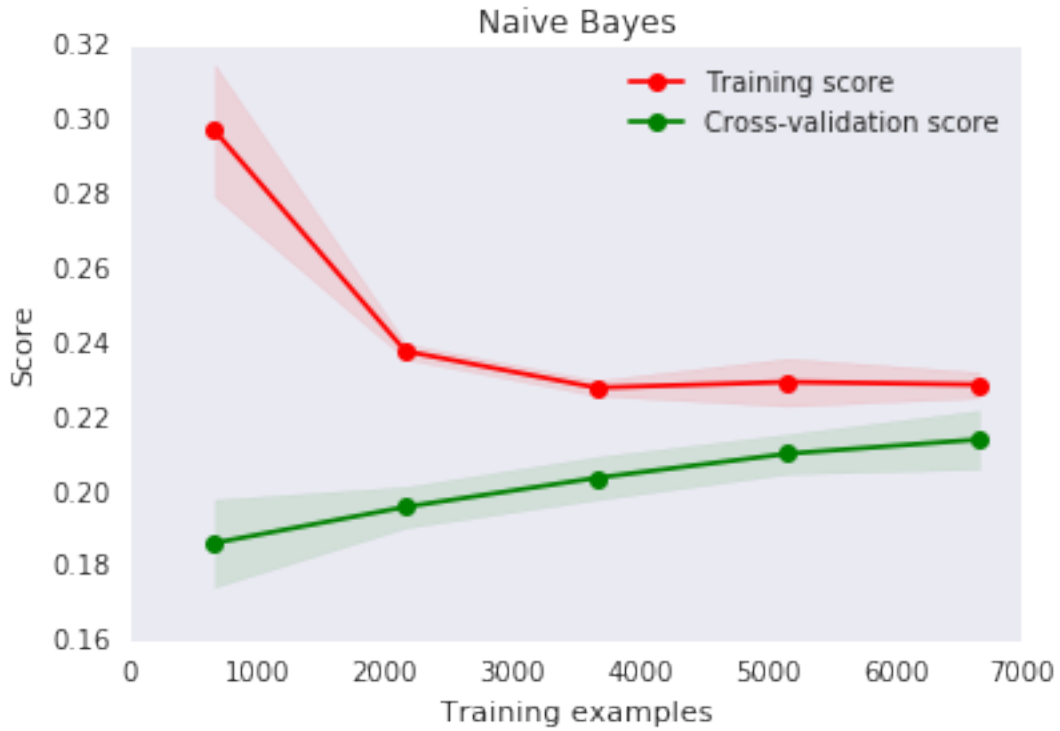

Figure 4: Learning curve: Naive Bayes

Figure 5: Learning curve: Logistic regression ($C = 1$)

After submitting Gradient Boosting Classifier with n_estimators=70 using set 2 we scored 2.52996 (team "Zilya") in the Public Leaderboard. The range in the Public Leaderboard is [1.95936; 34.53878], the median is 2.6133, the mean is 8.8250824461175075. We are on the 642 place among the total 6774 participants.

As a final note, Logistic Regression has showed quite good results, while being very fast to train ($\approx$ 2 minutes), and, as we can see from the learning curve, it doesn't require a very large dataset.

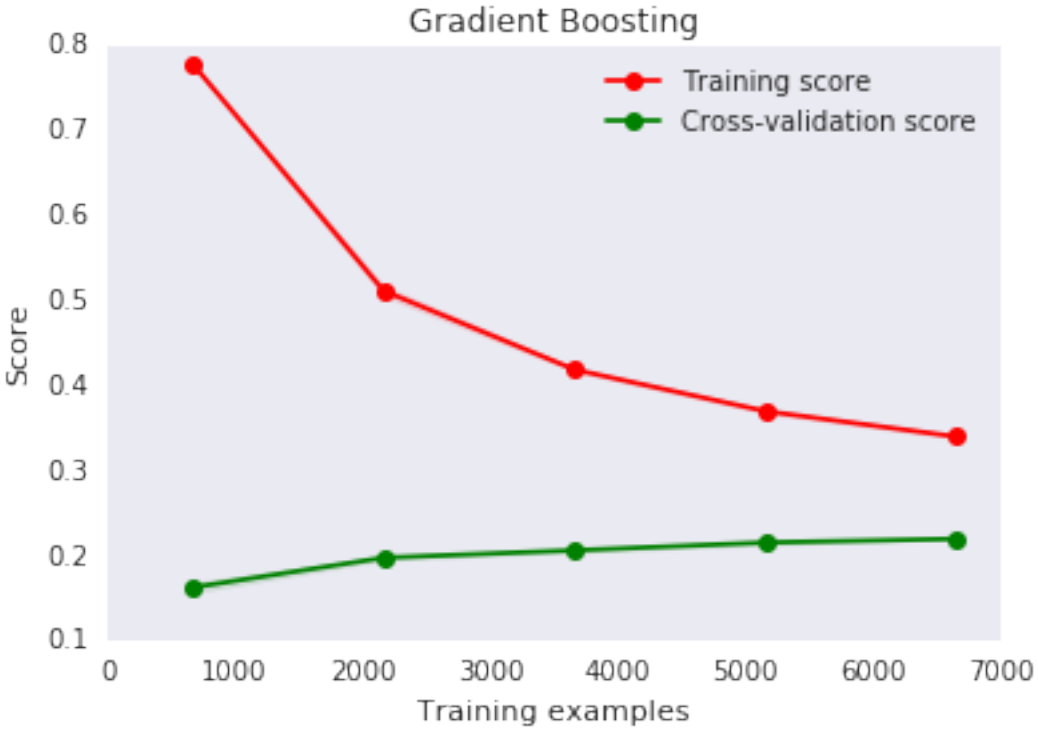

Figure 6: Learning curve: Gradient Boosting (40 trees)

Table 1: Model scores

Model	Set	Score	Settings
Naive Bayes classifier	1	2.5847142954194253	
Naive Bayes classifier	2	2.5555147705662731	
AdaBoost Classifier	2	3.1687059895128398	n_estimators=128
KNeighbors Classifier	2	2.5670426980397347	n_neighbors=1875, weights uniform
KNeighbors Classifier	2	2.5653473838229162	n_neighbors=2000, weights uniform
Logistic Regression	1	2.59367372421341	C=.01
Logistic Regression	2	2.5751351150170638	C=.01
Logistic Regression	2	2.5680290149864726	C=1
Gradient Boosting Classifier	1	2.58836083903685	n_estimators=50
Gradient Boosting Classifier	2	2.5118887513986259	n_estimators=50
Gradient Boosting Classifier	2	2.5208627693584873	n_estimators=40
Gradient Boosting Classifier	2	2.5082266486962368	n_estimators=60
Gradient Boosting Classifier	2	2.50518261159915	n_estimators=70

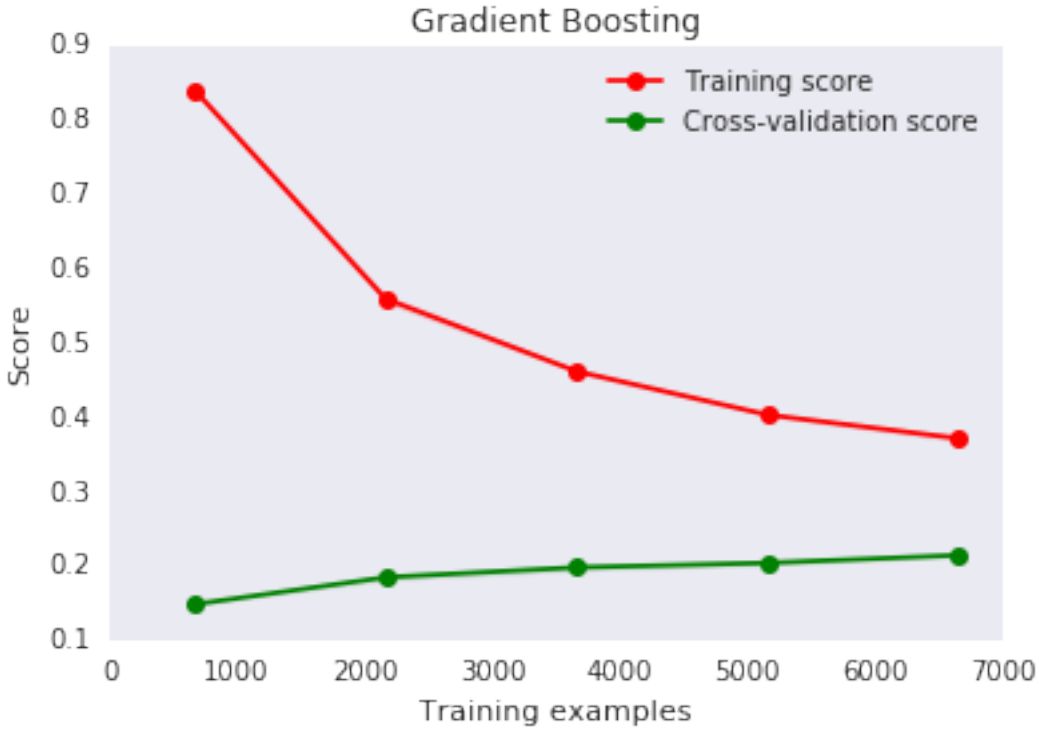

Figure 7: Learning curve: Gradient Boosting (60 trees)

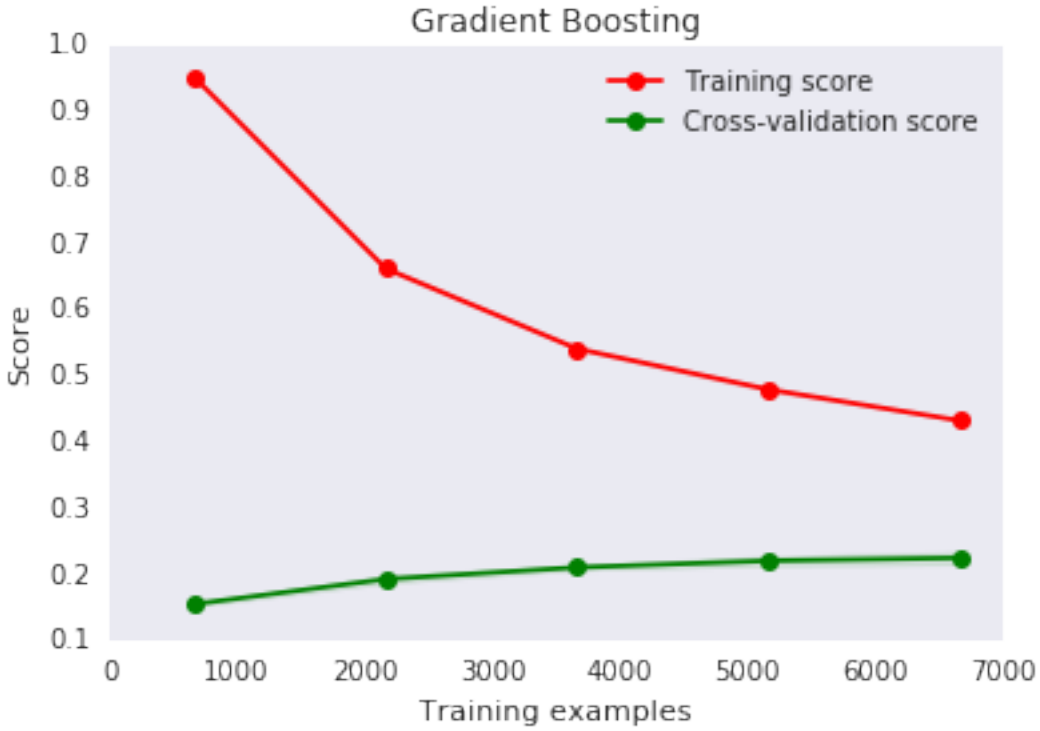

Figure 8: Learning curve: Gradient Boosting (100 trees)