Avito Duplicate Ads Detection, Winners' Interview: 2nd Place, Team TheQuants | Mikel, Peter, Marios, & Sonny

Kaggle Team | 08.31.2016

The Avito Duplicate Ads competition ran on Kaggle from May to July 2016. Over 600 competitors worked to feature engineer their way to the top of the leaderboard by identifying duplicate ads based on their contents: Russian language text and images. TheQuants, made up of Kagglers Mikel, Peter, Marios, & Sonny, came in second place by generating features independently and combining their work into a powerful solution.

In this interview, they describe the many features they used (including text and images, location, price, JSON attributes, and clustered rows) as well as those that ended up in the "feature graveyard." In the end, a total of 587 features were inputs to 14 models which were ensembled through the weighted rank average of random forest and XGBoost models. Read on to learn how they cleverly explored and defined their feature space to carefully avoid overfitting in this challenge.

The basics

What was your background prior to entering this challenge?

Mikel Bober-Irizar: Past Predictive modelling competitions, financial predictions and medical diagnosis.

Peter Borrmann: Ph.D. in theoretical physics, research assistant professor as well as previous Kaggle experiences.

Marios Michailidis: I am a Part-Time PhD student at UCL , data science manager at dunnhumby and fervent Kaggler.

Sonny Laskar: I am an Analytics Consulting Manager with Microland working on implementing Big Data Solutions; mostly dealing with IT Operations data.

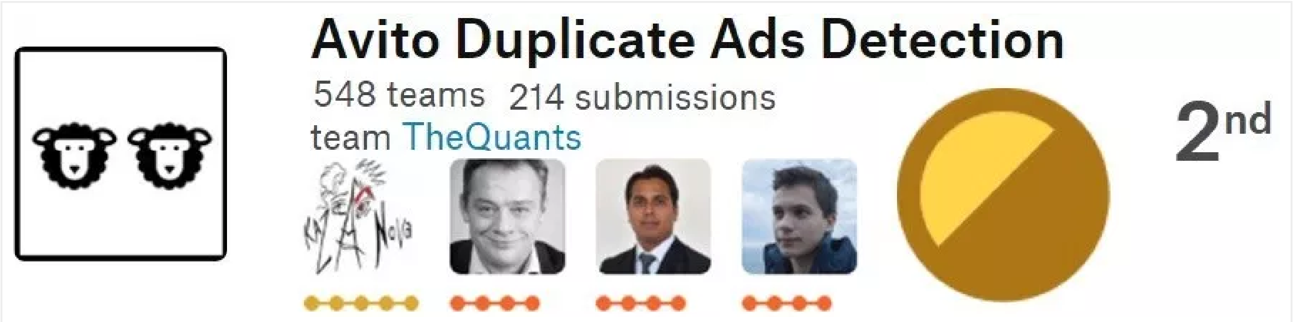
How did you get started with Kaggle?

Mikel Bober-Irizar: I wanted to learn about machine learning and use that knowledge to compete in competitions.

Peter Borrmann: I wanted to improve my skillset in the field.

Marios Michailidis: I wanted a new challenge and learn from the best.

Sonny Laskar: I got to know about Kaggle few years back when I was pursuing my MBA.

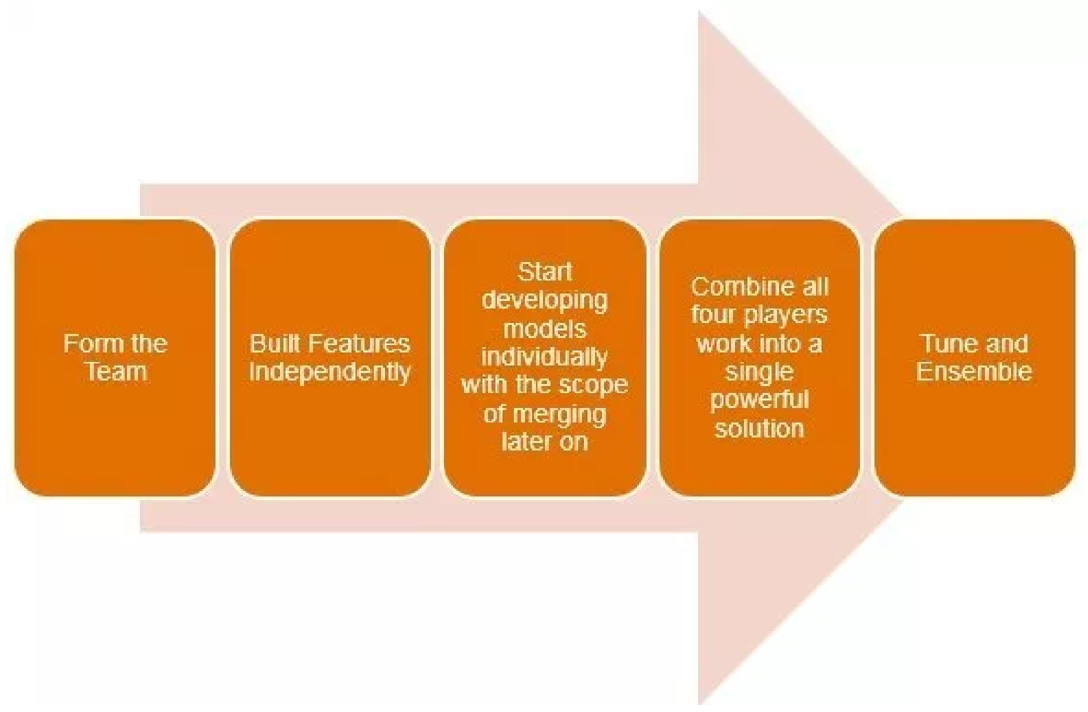


THE QUANTS TEAM

Summary

Our approach to this competition was divided into several parts :

1. Merging early based on standing of the leaderboard.
2. Generate features independently (on cleaned or raw data) that would potentially capture the similarity between the contents of 2 ads and could be further divided to more categories (like text similarities or image similarities).
3. Build a number of different classifiers and regressors independently with a hold out sample.
4. Combine all members' work
5. Ensemble the results through weighted rank average of a 2-layer meta-model network (StackNet).


Form the Team
Built Features Independently
Start developing models individually with the scope of merging later on
Combine all four players work into a single powerful solution
Tune and Ensemble

APPROACH SUMMARY.

Data Cleaning and Feature Engineering

Data Cleaning

In order to clean the text, we applied stemming using the NLTK Snowball Stemmer, and removed stopwords/punctuation as well as transforming to lowercase. In some situations we also removed non-alphanumeric characters too.

Feature Engineering vol 1: Features we actually used

In order to pre-emptively find over-fitting features, we built a script that looks at the changes in the properties (histograms and split purity) of a feature over time, which allowed us to quickly (200ms/feature) identify overfitting features without having to run overnight XGBoost jobs.

After removing overfitting features, our final feature space had 587 features derived from different themes:

General:

- CategoryID, parentCategoryID raw CategoryID, parentCategoryID one-hot (except overfitting ones).
- Price difference / mean.
- Generation3probability (output from model trained to detect generationmethod=3).

Location:

- LocationID & RegionID raw.
- Total latitude/longtitude.
- SameMetro, samelocation, same region etc.
- Distance from city centres (Kalingrad, Moscow, Petersburg, Krasnodar, Makhachkala, Murmansk, Perm, Omsk, Khabarovsk, Kluichi, Norilsk)
  Gaussian noise was added to the location features to prevent overfitting to specific locations, whilst allowing XGBoost to create its own regions.

All Text:

- Length / difference in length.
- nGrams Features (n = 1,2,3) for title and description (Both Words and Characters).
  - Count of Ngrams (#, Sum, Diff, Max, Min).
  - Length / difference in length.
  - Count of Unique Ngrams.
  - Ratio of Intersect Ngrams.
  - Ratio of Unique Intersect Ngrams.
- Distance Features between the titles and descriptions:
  - Jaccard
  - Cosine
  - Levenshtein
  - Hamming
- Special Character Counting & Ratio Features:

    - Counting & Ratio features of Capital Letters in title and description.
    - Counting & Ratio features of Special Letters (digits, punctuations, etc.) in title and description.
- Similarity between sets of words/characters.
- Fuzzywuzzy distances.
- jellyfish distances.
- Number of overlapping sets of n words (n=1,2,3).
- Matching moving windows of strings.
- Cross-matching columns (eg. title1 with description2).

Bag of words:

For each of the text columns, we created a bag of words for both the intersection of words and the difference in words and encoded these in a sparse format resulting in ~80,000 columns each. We then used this to build Naive Bayes, SGD and similar models to be used as features.

Price Features:

- Price Ratio.
- Is both/one price NaN.
- Total Price.

JSON Features:

- Attribute Counting Features.
    - Sum, diff, max, min.
- Count of Common Attributes Names.
- Count of Common Attributes Values.
- Weights of Evidence on keys/values, XGBoost model on sparse encoded attributes.

Image Features:

- # of Images in each Set.
- Difference Hashing of images.
- Hamming distance between each pair of images.
- Pairwise comparison of file size of each image.
- Pairwise comparison of dimension of each image.
- BRISK keypoint/descriptor matching.
- Image histogram comparisons.
- Dominant colour analysis.
- Uniqueness of images (how many other items have the same images).
- Difference in number of images.

Clusters:

We found clusters of rows by grouping rows which contain the same items (eg. if row1 has items 123, 456 and row2 has items 456, 789 they are in the same cluster). We discovered that the size of these clusters was a very good feature (larger clusters were more likely to be non-duplicates), as well as the fact that clusters always the same generationMethod. Adding cluster-size features gave us a 0.003 to 0.004 improvement.

Feature Engineering vol 2 : The ones that did not make it

Overfitting was probably the biggest problem throughout the competition, and lots of features which (over)performed in validation didn't do so well on the leaderboard. This is likely because the very powerful features learn to recognise specific products or sellers that do not appear in the test set. Hence, a feature graveyard was a necessary evil.

TF-IDF:

This was something we tried very early into the competition, adapting our code from the Home Depot competition. Unfortunately, it overfitted very strongly, netting us 0.98 val-auc and only 0.89 on LB. We tried adding noise, reducing complexity, but in the end we gave up.

Word2vec:

We tried both training a model on our cleaned data and using the pretrained model posted in the forums. We tried using word-mover distance from our model as features, but they were rather weak (0.70AUC) so in the end we decided to drop these for simplicity. Using the pre-trained model did not help, as the authors used MyStem for stemming (which is not open-source) so we could not replicate their data cleaning. After doing some transformations on the pre-trained model to try and make it work with our stemming (we got it down to about 20% missing words), it scored the same as our custom word2vec model.

Advanced cluster features:

We tried to expand the gain from our cluster features in several ways. We found that taking the mean prediction for the cluster as well as cluster_size * (1-cluster_mean) provided excellent features in validation (50% of gain in xgb importance), however these overfitted. We also tried taking features such as the standard deviation of locations of items in a cluster, but these overfitted too.

Grammar features:

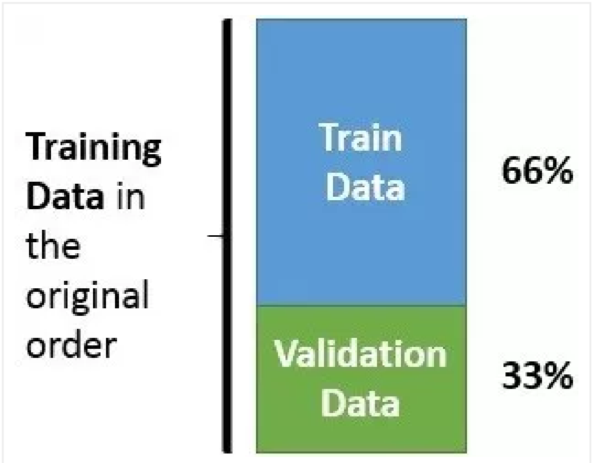
We tried building features to *fingerprint* different types of sellers, such as usage of capital letters, special characters, newlines, punctuation etc. However while these helped a lot in CV, they overfitted on the leaderboard.

Brand violations:

We built some features based around words that could never appear together in duplicate listings. (For example, if one item wrote 'iPhone 4s' but the other one wrote 'iPhone 5s', they could not be duplicates). While they worked well at finding non-duplicates, there were just too few cases where these violations occurred to make a difference to the score.

Validation Scheme

Initially, we were using a random validation set before switching to a set of non-overlapping items, where none of the items in the valset appeared in the train set. This performed somewhat better, however we had failed to notice that the training set was ordered based on time! We later noticed this (inspired by this post) and switched to using last 33% as a valset.



VALIDATION SCHEME.

This set correlated relatively well with the leaderboard until the last week, when we were doing meta-modelling and it fell apart - at a point where it would be too much work to switch to a better set. This hurt us a lot towards the end of the competition.

Modelling

Modelling vol 1 : The ones that made it

In this section we built various models (classifiers and regressors) on different input data each time (since the modelling process was overlapping with the feature engineering process. All models were training with the first 67% of the training data and validated on the remaining 33%. All predictions were saved (so that they can be used later for meta modelling. The most dominant models were:

XGBoost:

Trained with all 587 of our final features with 1000 estimators, maximum depth equal to 20 and minimum child of 10, and particularly high Eta (0.1) - bagged 5 times. We also replaced *nan* values with -1 and *Infinity* values with 99999.99. It scored **0.95143** on private leaderboard. Bagging added 0.00030 approximately.

Keras: Neural Network

Trained with all our final features, transformed with standard scaler as well as with logarithm plus 1, where all negative features have been replaced with zero. The main architecture involved 3 hidden layers with 800 hidden units plus 60% dropout. The main activation function was Softmax and all intermediate ones were standard rectifiers (Relu). We bagged it 10 times. It scored **0.94912** on private leaderboard. It gave +0.00080-90 when rank-averaged with the XGBoost model

Modelling vol 2: The ones that didn't

We build a couple of deeper Xgboost models with higher Eta (0.2) that although performed well in cv, they overfitted the leaderboard.

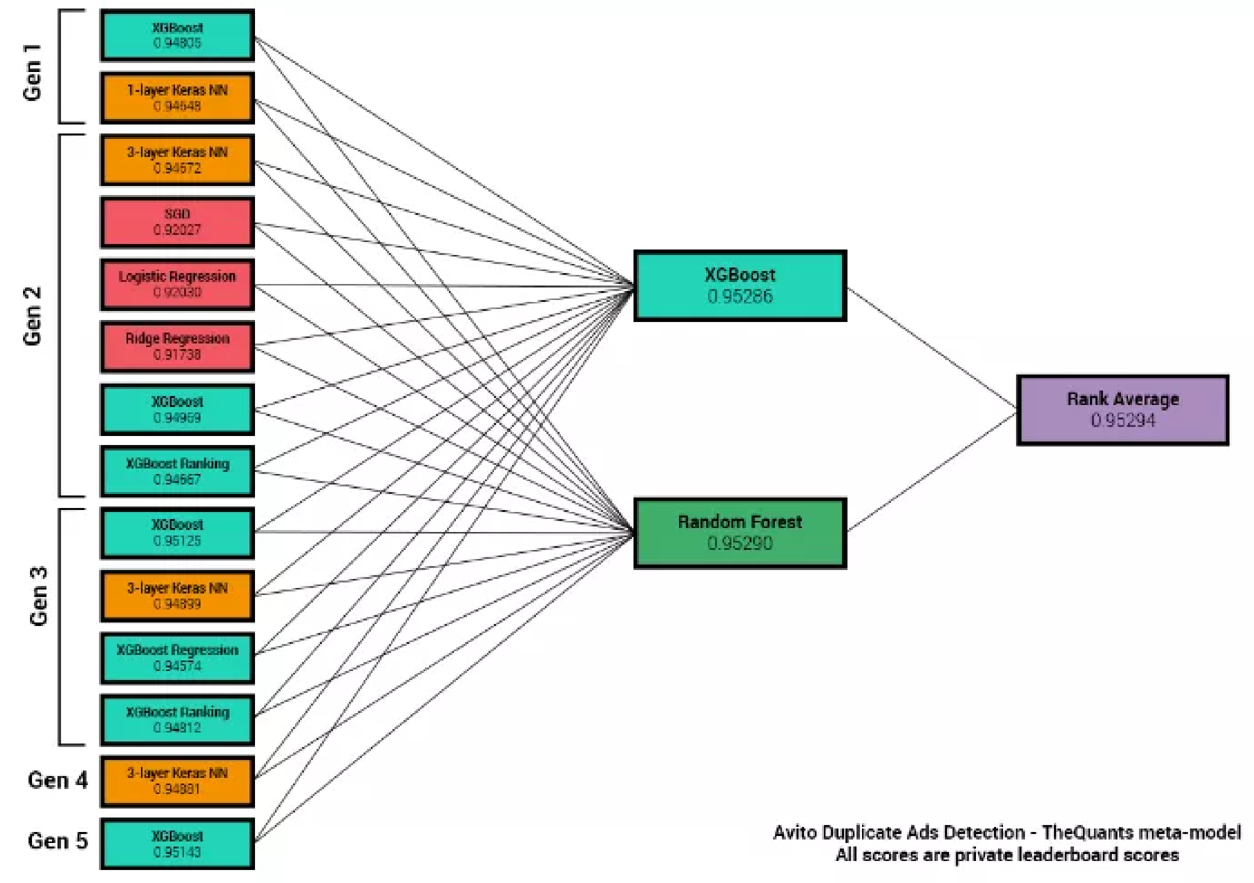
We used a couple of models to predict generation method in order to use that as feature for meta-modelling but it did not add anything so we removed it.

Meta-Modelling

The previous modelling process generated 14 different models including linear models as well as XGBoosts and NNs, that were later used for meta-modelling

For validation purposes we splitted the remaining (33%) data again into 67-33 in order to tune the hyper parameters of our meta-models that used as input the aforementioned 14 models. Sklearn's Random Forest which performed slightly better than XGBoost (0.95290 vs 0.95286). Their rank average yielded our best Leaderboard score of **0.95294**

The Modelling and Meta-Modelling process is also illustrated below :



THE QUANTS ENSEMBLE.

Thanks

Thanks to the competitors for the challenge, Kaggle for hosting, Avito for organizing. Thanks to the open source community and the research that makes it all possible.

Teamwork

How did your team form?

Early on into the competition Peter & Sonny & Mikel formed a team as they held the top 3 spots at the time, and decided to join forces to see how far they could go. Later on, Marios was spotted lurking at the bottom of the leaderboard, and was asked to join because of his extensive Kaggle experience.

How did your team work together?

We were all quite independent, branching out and each working on our own features as there was lots of ground to cover, while also brainstorming and discussing ideas together. At the end we came together to consolidate everything into one featurespace and to build models for it.

Bios

Mikel Bober-Irizar (anokas) is a young and ambitious Data Scientist and Machine Learning Enthusiast. He has been participating in various predictive modelling competitions, and has also developed algorithms for various problems, including financial prediction and medical diagnosis. Mikel is currently finishing his studies at Royal Grammar School, Guildford, UK, and plans to go on to study Math or Computer Science.

Priv.-Doz. Dr. Peter Borrmann (NoName) is head of The Quants Consulting focusing on quantitative modelling and strategy. Peter studied in Göttingen, Oldenburg and Bremen and has a Ph.D. in theoretical physics. He habilitated at the University of Oldenburg where he worked six years as a research assistant professor. Before starting his own company Peter worked at IBM Business Consulting Services in different roles.

Marios Michailidis (KazAnova) is Manager of Data Science at Dunnhumby and part-time PhD in machine learning at University College London (UCL) with a focus on improving recommender systems. He has worked in both marketing and credit sectors in the UK Market and has led many analytics projects with various themes including: Acquisition, Retention, Uplift, fraud detection, portfolio optimization and more. In his spare time he has created KazAnova, a GUI for credit scoring 100% made in Java. He is former Kaggle #1.

Sonny Laskar (Sonny Laskar) is an Analytics Consulting Manager at Microland (India) where he is building IT Operations Analytics platform. He has over eight years of experience spread across IT Infrastructure, Cloud and Machine learning. He holds an MBA from India's premiere B School IIM, Indore. He is an avid break dancer and loves solving logic puzzles.

Share this:

Related

AVITO DUPLICATE ADS DETECTION FEATURE ENGINEERING RANDOM FOREST XGBOOST

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name*

Email*

Website

☐

Save my name, email, and website in this browser for the next time I comment.

☐ Notify me of follow-up comments by email.

☐ Notify me of new posts by email.

Submit

The Official Blog of Kaggle.com

Search

Categories

- DATA NOTES (18)
- DATA SCIENCE NEWS (63)
- KAGGLE NEWS (148)
- KERNELS (48)
- OPEN DATASETS (12)
- PULSE OF THE COMPETITION (1)
- STUDENTS (4)
- TUTORIALS (54)
- UNCATEGORIZED (3)

WINNERS' INTERVIEWS (228)

Popular Tags

#1 KAGGLER ANNUAL SANTA COMPETITION BINARY CLASSIFICATION COMMUNITY COMPUTER VISION CONVOLUTIONAL NEURAL NETWORKS DARK MATTER DATA NOTES DATA SCIENCE CAREERS DATA VISUALIZATION DEEP NEURAL NETWORKS DELOITTE DIABETES DIABETIC RETINOPATHY DRAPER SATELLITE IMAGE CHRONOLOGY EEG DATA ELO CHESS RATINGS COMPETITION EUROVISION CHALLENGE FLIGHT QUEST HERITAGE HEALTH PRIZE HOW MUCH DID IT RAIN? IMAGE CLASSIFICATION KAGGLE DATASETS KAGGLE INCLASS KERNELS LOGISTIC REGRESSION MARCH MANIA MEETUP MULTICLASS CLASSIFICATION NATURAL LANGUAGE PROCESSING OPEN DATA OPEN DATA SPOTLIGHT PRACTICE FUSION PRODUCT PRODUCT NEWS PROFILING TOP KAGGLERS RECRUITING REGRESSION PROBLEM SCIKIT-LEARN SCRIPTS OF THE WEEK TOURISM FORECASTING TUTORIAL VIDEO SERIES WIKIPEDIA CHALLENGE XGBOOST

Archives

Select Month ▾