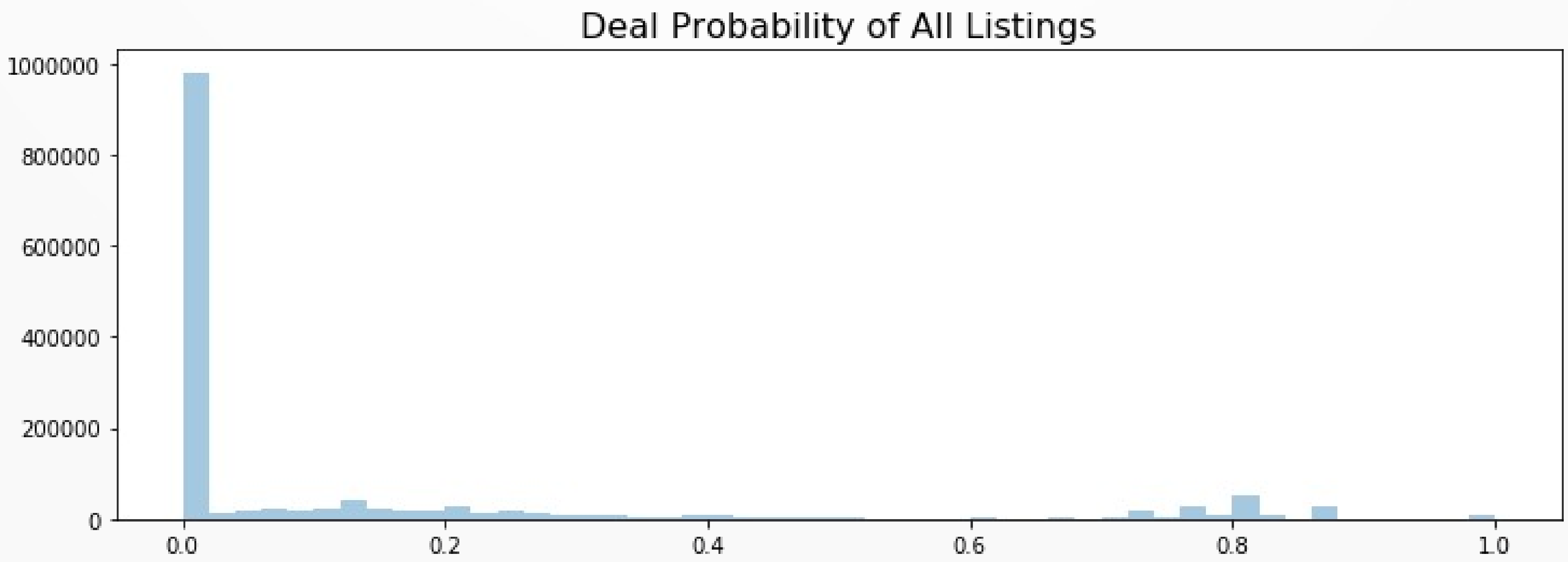
Deal Probability for Online Sellers

Helping sellers know what to expect from their listings.

Importance of Online Sellers

- **Platforms that rely on used goods sales:** Ebay, Craigslist, (Less)Amazon, Avito(Russia).
- Online sellers are the lifeblood of these platforms. You must keep them happy.
- Most online sellers have no idea what they're doing, as shown by these 1.5M Listings on Avito.ru

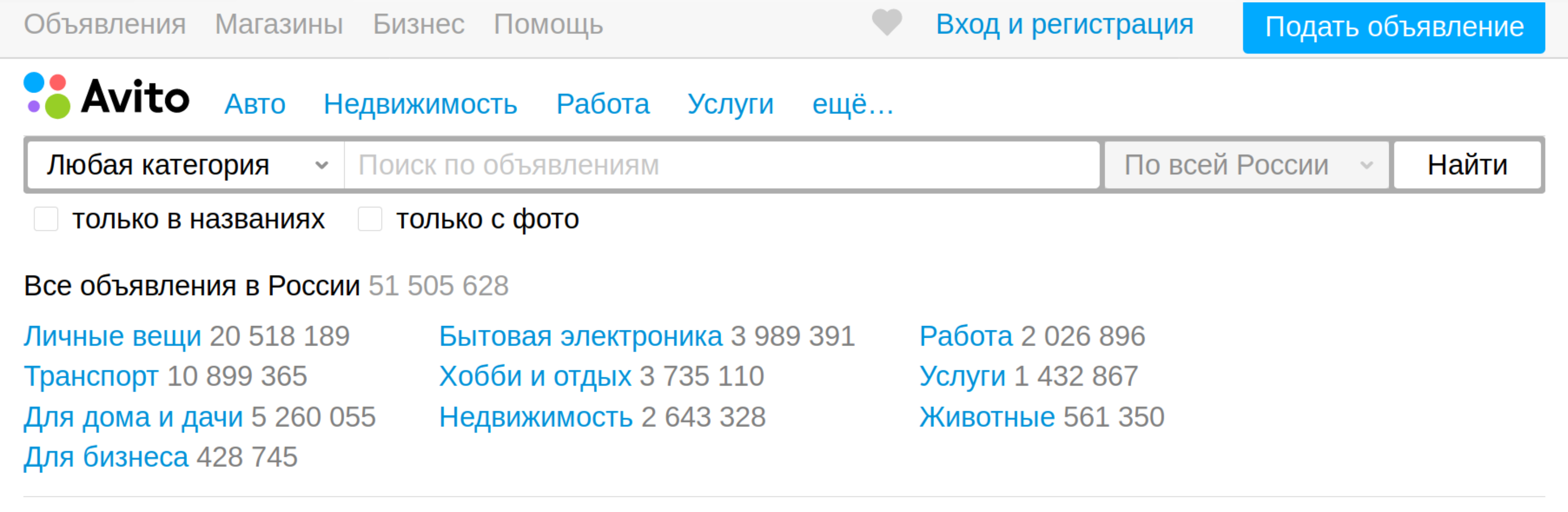


Struggles of Online Sellers

- **Tiny details** in a product listing can make a **big difference** in buyer’s interest.
- Even with an optimized listing, **demand for a product may simply not exist**, frustrating sellers who may have over-invested in marketing.
- Sellers online sometimes feel frustrated with both too little demand (indicating something is **wrong with the product or the product listing**) or too much demand (indicating a hot item with a good description was **underpriced**).

Solution: **Deal Probability**

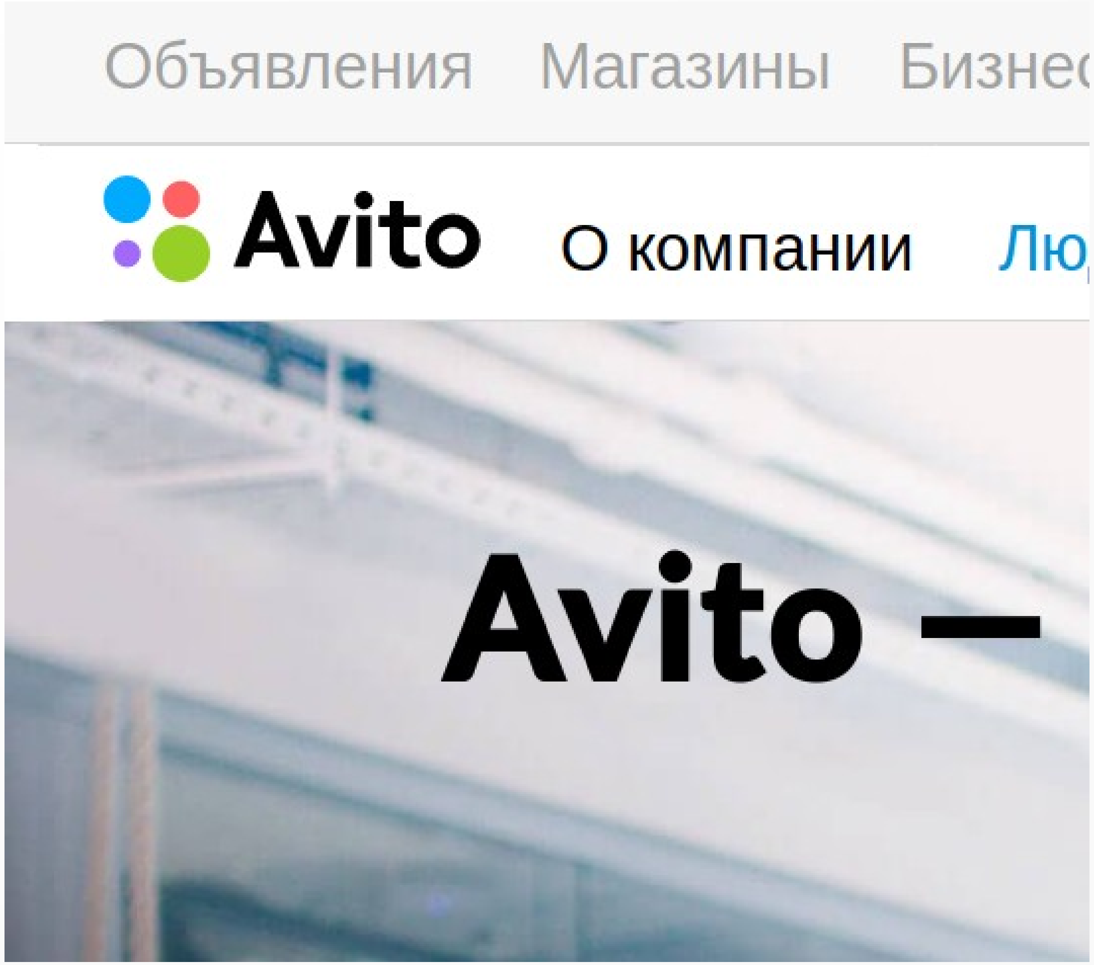
- Recommendations aimed at improving the experience of placing ads, and consequently prevent sellers from migrating to other platforms.
- **How to best optimize their listing.**
  - Bot: “We noticed your ad description is too short. Ads with at least N number of words have higher chances of selling.”
- **Realistic expectation of buyer’s interest.**
  - Bot: “Based on similar ads, yours has 99% chances of selling. Cash is coming your way!”


Объявления Магазины Бизнес Помощь
Вход и регистрация
Подать объявление
Avito
Авто Недвижимость Работа Услуги ещё…
Любая категория
Поиск по объявлениям
По всей России
Найти
только в названиях
только с фото
Все объявления в России 51 505 628
Личные вещи 20 518 189
Транспорт 10 899 365
Для дома и дачи 5 260 055
Для бизнеса 428 745
Бытовая электроника 3 989 391
Хобби и отдых 3 735 110
Недвижимость 2 643 328
Работа 2 026 896
Услуги 1 432 867
Животные 561 350

Avito’s Data

About Avito

- **Avito.ru** is a Russian classified advertisements website with sections devoted to general goods for sale, jobs, real estate, personals, cars for sale, and services.
- Avito.ru is the most popular classifieds site in Russia and is the **second biggest classifieds site in the world** after Craigslist.



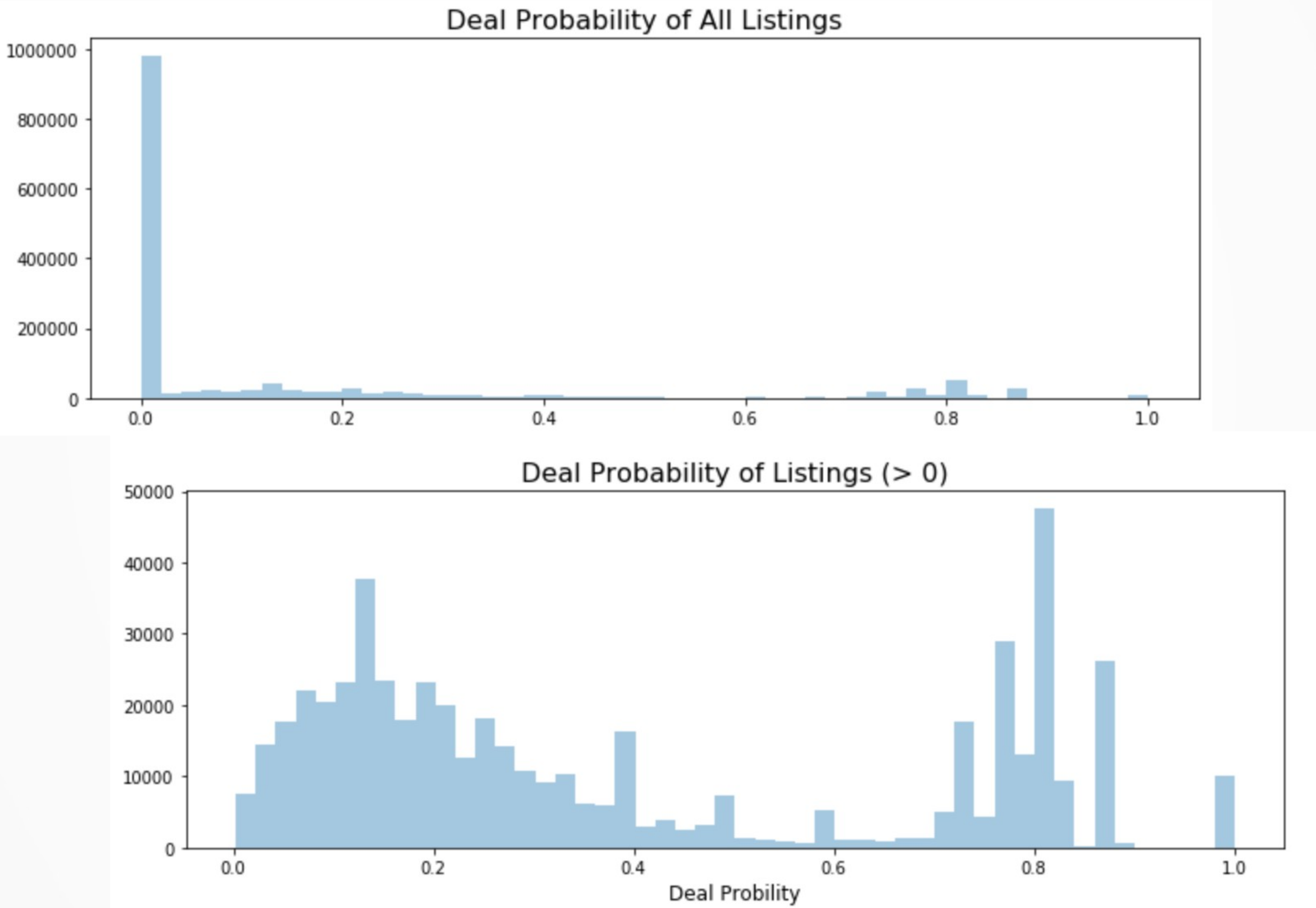
About the Dataset

- It’s big. 1.5+Million ads.
- It’s in Russian, with Cyrillic alphabet.
- Titles and descriptions offer endless NLP opportunities.
- Plenty of categorical data to binarize.

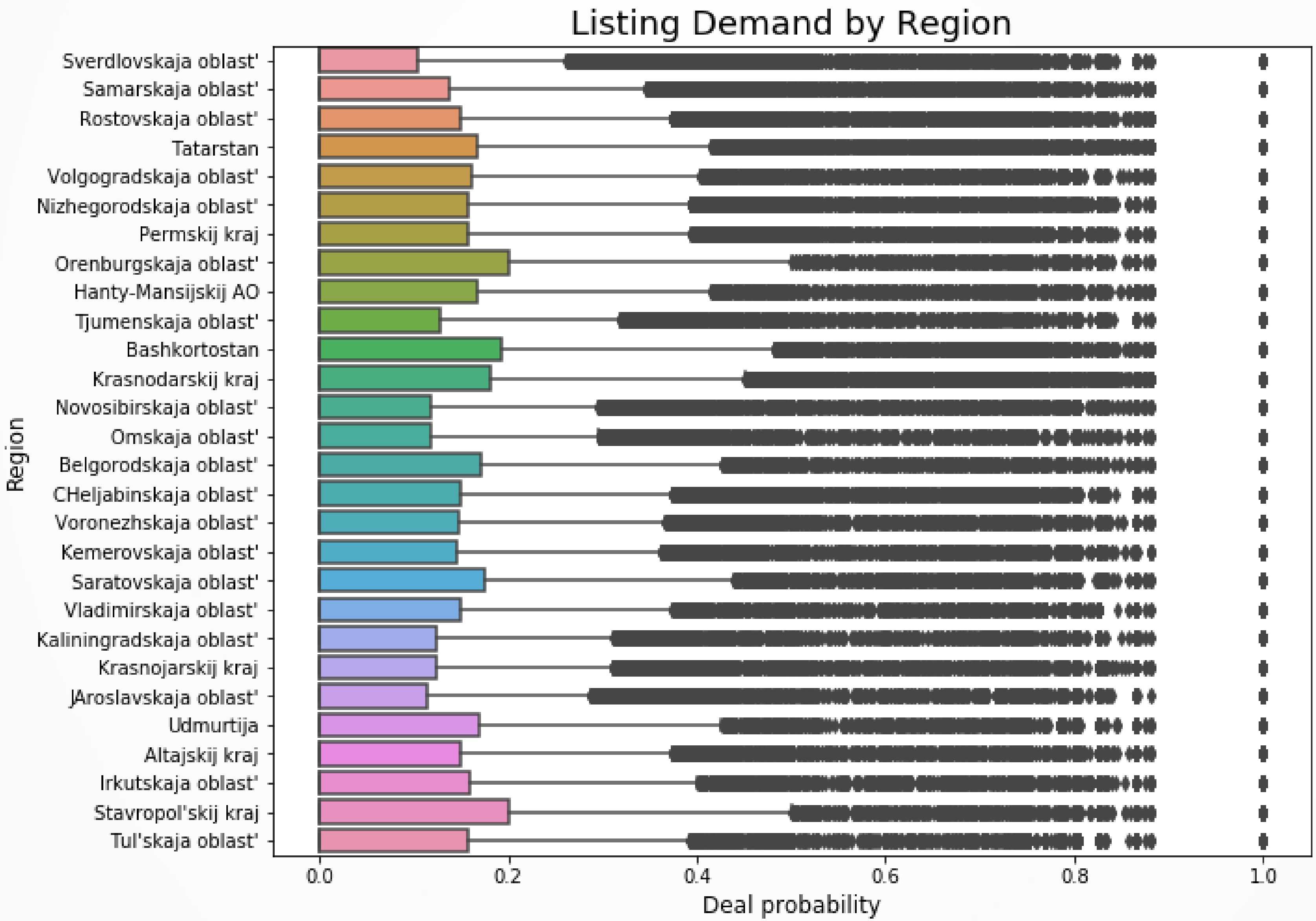
Out[3]:

	item_id	user_id	region	city	parent_category_name	category_name	param_
0	b912c3c6a6ad	e00f8ff2eaf9	Свердловская область	Екатеринбург	Личные вещи	Товары для детей и игрушки	Постельны принадлежнос
1	2dac0150717d	39aeb48f0017	Самарская область	Самара	Для дома и дачи	Мебель и интерьер	Друг
2	ba83aefab5dc	91e2f88dd6e3	Ростовская область	Ростов-на-Дону	Бытовая электроника	Аудио и видео	Видео, DVD Blu-ray плеер
3	02996f1dd2ea	bf5cccea572d	Татарстан	Набережные Челны	Личные вещи	Товары для детей и игрушки	Автомобильны крес
4	7c90be56d2ab	ef50846afc0b	Волгоградская область	Волгоград	Транспорт	Автомобили	С пробег

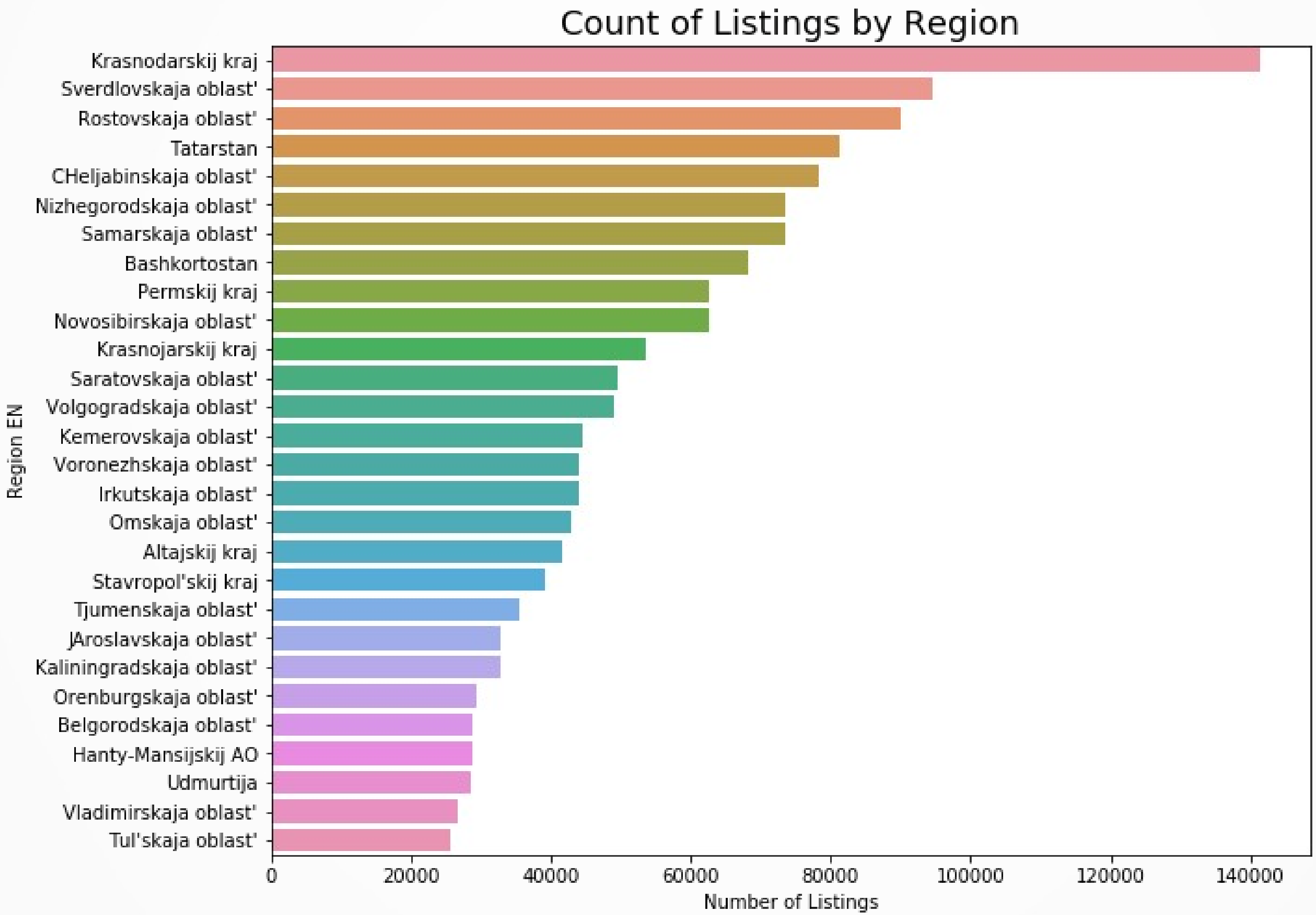
- **Outcome variable is very non-normal.** There's three distinct groups. (Zero Range, Lower Range, Upper Range)



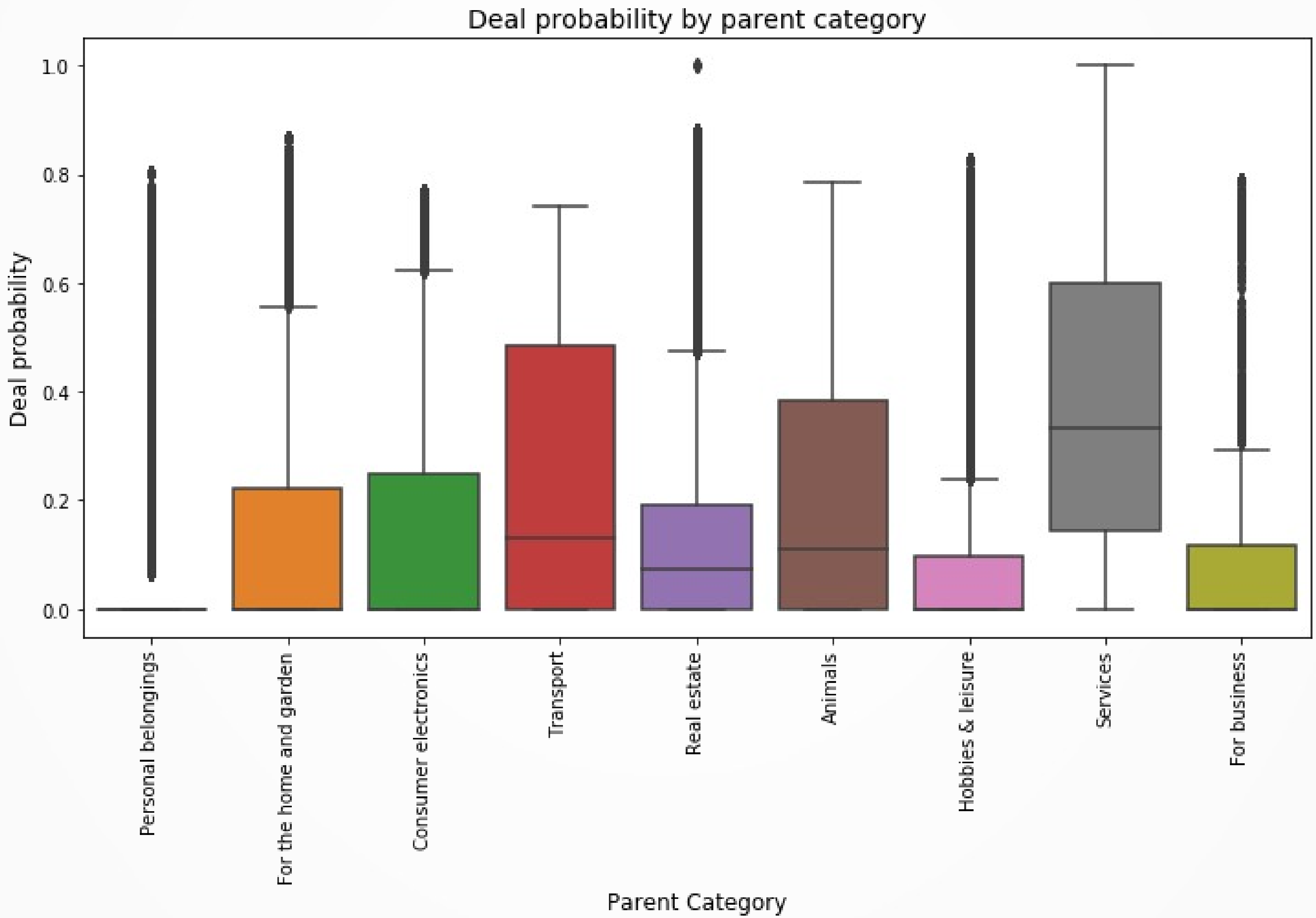
Demand Distribution by Region



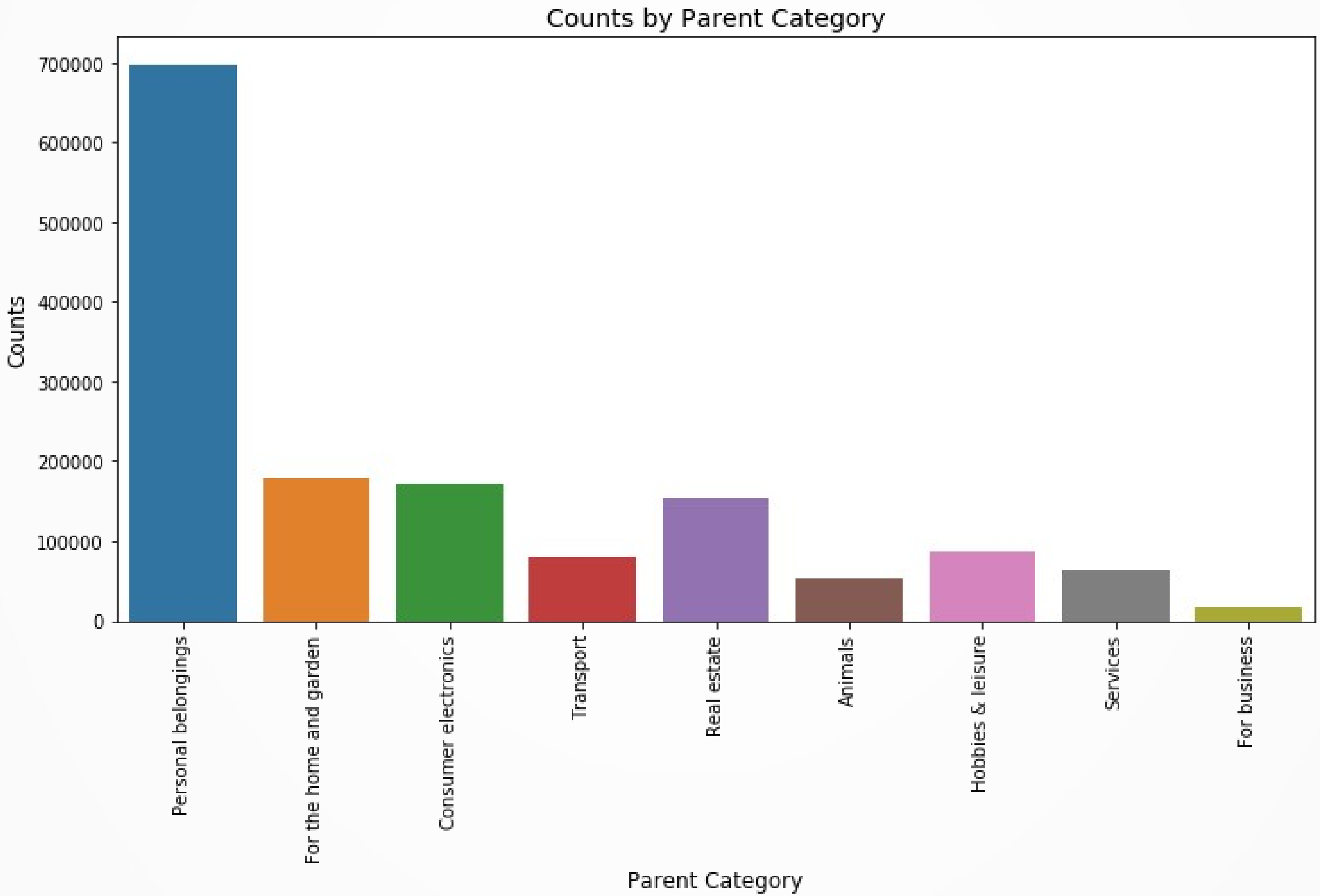
Listing Counts by Region



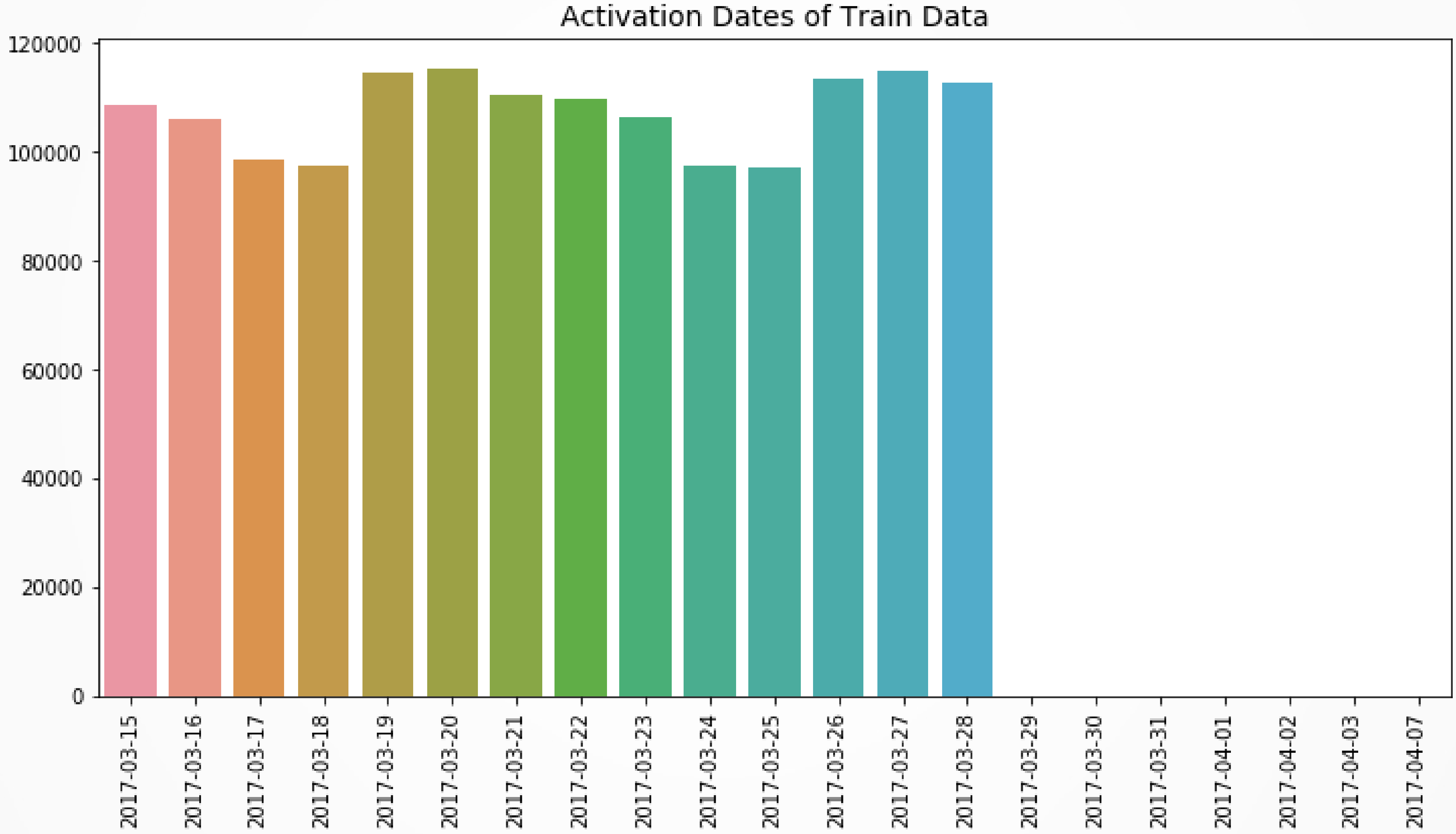
Deal Probability by Parent Category



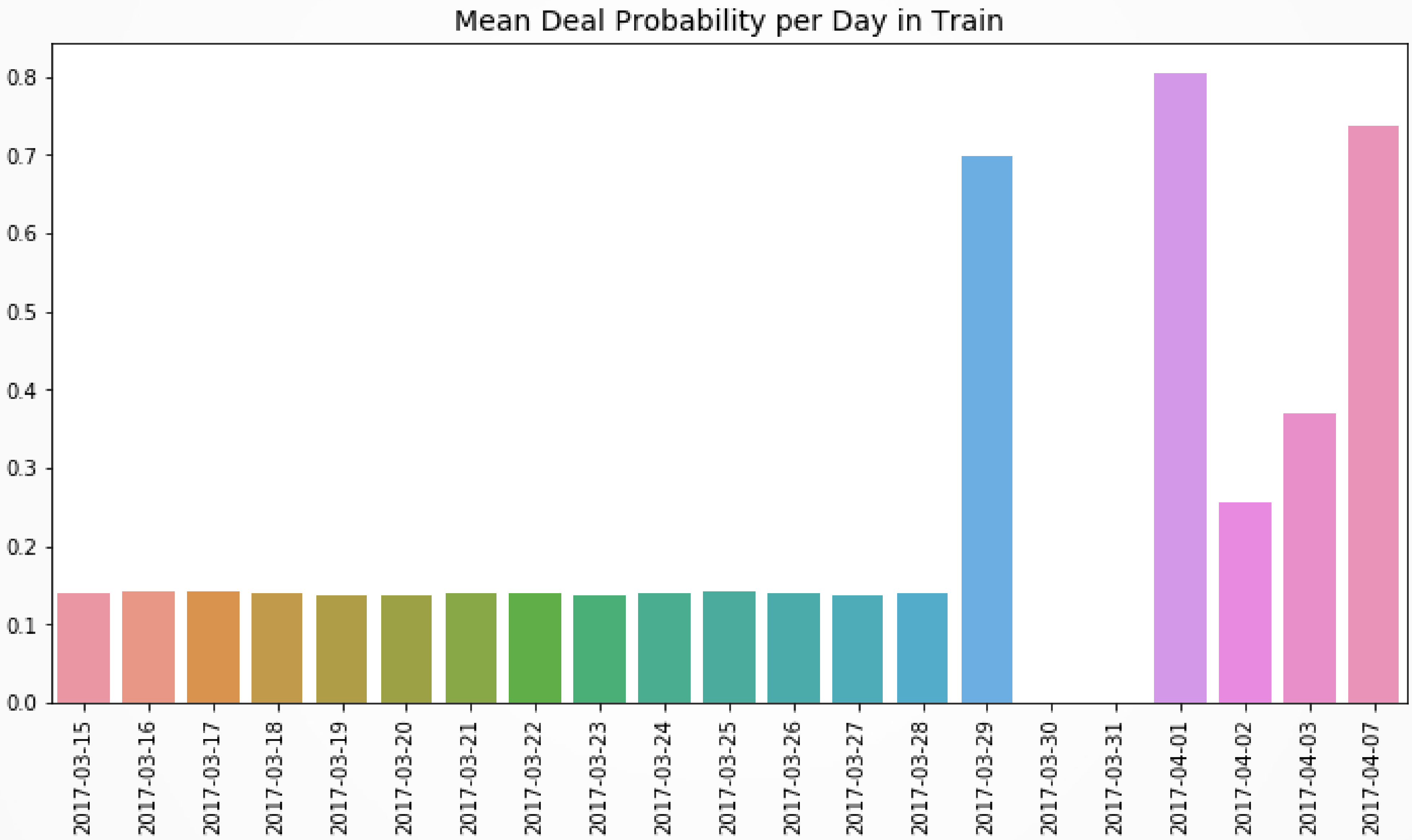
Ad Count by Parent Category



Count of Ad Activation Dates



Daily Mean Deal Probability



Feature Engineering Pipeline

Generation of feature sets in detail...

`sklearn.cross_decomposition`.PLSRegression

vs

`sklearn.decomposition`.TruncatedSVD

- Natural Language Processing involves applying Singular Value Decomposition onto a Document-Frequency Matrix.
- SVD is unsupervised (decomposition), PLSR is supervised (cross-decomposition).
- PLSR is ideal for use when many features are correlated and/or the number of features exceeds the number of datapoints.
- NLP for explanatory vs predictive purposes.

Cross-Decomposition of TF-IDF Vectors with BiGrams¶

- This process consists of extracting term frequency vectors using the text in each ad as a document. Tokens for unigrams and bigrams will be included in this stage. Lastly, the resulting matrix will be reduced to the smallest number of components that retain all potential predictive power.
- Perform onto both titles and descriptions and retain separate components for each.

- **About this step**
-
- Russian stopwords from NLTK library.

```
# Sample of russian stopwords from NLTK
nltk.corpus.stopwords.words('russian')[:10]
```

```
['и', 'в', 'во', 'не', 'что', 'он', 'на', 'я', 'с', 'со']
```

- Let vectorizer compute BiGrams along with single terms. TriGrams had very poor performance.
- Vectorizer `lowercase=False` on titles and descriptions.
- Filtering terms: set `min_df=0.00005`, which reduced 200k+ features down to 30k+.

Challenges of PLSR Reduction

- **PLSR doesn't handle CSR matrices.** Only dense format data.
- With 1.5Million rows on 30k columns, you WILL run **out of memory.**
- **Solution**: Decompose column ranges at a time.
  - Iterative PLSR decomposes ranges of columns at a time, so process doesn't run out of memory.

```
Columns: 61500-63000
Prelim score for column range: 0.010685719787845938
Aggregate cv: [0.17766326 0.17203679 0.17633762 0.17824416]
Columns: 63000-64500
Prelim score for column range: 0.014578818745921374
Aggregate cv: [0.17883687 0.17353379 0.17777242 0.17950047]
Columns: 64500-66000
Prelim score for column range: 0.011591843160968396
Aggregate cv: [0.18006346 0.17490881 0.1788604  0.1805137 ]
Columns: 66000-67138
Prelim score for column range: 0.009172436403075301
Aggregate cv: [0.18092212 0.17574777 0.179737   0.18130798]
Decomposing Aggregate...
Aggregate cv after decomposition: [0.18097245 0.1757806  0.1
797841  0.18141601]
Minutes: 46.37753241459529
```

```python
vec = feature_extraction.text.TfidfVectorize
    stop_words=ru_stop,
    lowercase=False,
    ngram_range=(1,2),
    min_df=0.000005)
# Fitting on train and test as merged lists
vec.fit(train[var].astype(str).tolist() + test[var].ast
print('N tokens:',len(vec.get_feature_names()))
```

```
N tokens: 67138
```

Discrete Vector Cross-Decomposition

- This consists of splitting the dependent variable into discrete ranges and creating a vocabulary for each range.
- Then vectorize and cross-decompose each vocabulary independently. Resulting components for each vocabulary will reflect the presence of terms common in a certain discrete range of target.

- **Make target groups:** rows where deal probability is in either zero, lower or upper range.
- **Make strings:** Join the titles of each group into 3 long corpuses to be used as documents.
- **Get TF-IDF matrix** and transpose.
- **Indicator** of which document has the **highest frequency.**
- The index of each group are the terms most common in that range.

	0	1	2	group
ёрочка	0.000000	0.000000	0.000015	2
ёршик	0.000000	0.000000	0.000046	2
ёта	0.000000	0.000024	0.000012	1
ёх	0.000040	0.000009	0.000018	0
ёхкомфорчатая	0.000067	0.000000	0.000000	0

	up_voc	low_voc	zero_voc
0	ВАЗ	м²	Продам
1	iPhone	эт	Платье
2	Коляска	квартира	Куртка
3	LADA	сот	платье
4	Samsung	участке	Туфли

Discrete Vector Sums

- Similar to previous procedure, vocabularies are created for discrete ranges of target.
- Instead of decomposing the vectors of those vocabularies, you simply sum their frequencies along the row axis of the term frequency matrix. This results in a single variable for each vocabulary, which represents the aggregate frequency of a vocabulary's terms per ad.

```python
vec = feature_extraction.text.TfidfVectorizer(
    stop_words=ru_stop,
    lowercase=False,
    #max_features=8600,
    #ngram_range=(1,2),
    #min_df=0.0005,
    #max_df=0.0005,
    vocabulary=vocabs.up_voc.dropna()
)
vec.fit(train['title'].astype(str).tolist()+test['title'].astype(str).tolist())
print(len(vec.get_feature_names()))
```

16262

```python
# Word counts for train. CSR Matrix, tokens ordered alphabetically
counts = vec.transform(train['title'].astype(str).tolist())

sums = counts.sum(axis=1)

sums = sums.tolist()

sumsdf['upvoc'] = [i[0] for i in sums]
```

Sentiment Analysis

- An NLP library called **polyglot** offers multi-language tools, such as Sentiment-Analysis and Named-Entity-Recognition in Russian.

Features

- Tokenization (165 Languages)
- Language detection (196 Languages)
- Named Entity Recognition (40 Languages)
- Part of Speech Tagging (16 Languages)
- Sentiment Analysis (136 Languages)
- Word Embeddings (137 Languages)
- Morphological analysis (135 Languages)
- Transliteration (69 Languages)

Weaknesses of Polyglot

- **Tokenization engine.** Documents must be manually pre-processed to ensure the number of detected sentences matches the number of rows.
- Bound to mismatches. Only worked on titles.

```
title...
Cleaning train text.
Polyglot parsing.

Detector is not able to detect

N detected sentences: 1503424
Actual N rows: 1503424
Computing entity sentiments.
0.0 percent done.
0.05 percent done.
0.1 percent done.
```

Binary CountVectorizer

- Several categorical variables in this data have thousands of unique values which would increase the dimensional space unreasonably if binarizing in dense format.
- A binary CountVectorizer does the heavy lifting of populating dummy counts in sparse format, and PLSR reduces the numerous columns to a few core components.

Binary CountVectorizer as Dummy

```
param_1 p1plsr
N tokens: 387
Columns: 0-387
Prelim score for column range: 0.15655498110674948
Aggregate cv: [0.15676329 0.15322856 0.15637188 0.15657073]
(1503424, 10) (508438, 10)
param_2 p2plsr
N tokens: 280
Columns: 0-280
Prelim score for column range: 0.12558359907063832
Aggregate cv: [0.16264688 0.15924979 0.16186965 0.16246216]
(1503424, 20) (508438, 20)
param_3 p3plsr
N tokens: 1269
Columns: 0-1000
Prelim score for column range: 0.05751873133971
Aggregate cv: [0.16334192 0.15999341 0.16264773 0.16316857]
(1503424, 30) (508438, 30)
```

Target-Sorted Label Encodings

- Normally, label encoding isn't recommended for machine learning because the algorithm will interpret the code numbers as meaningful information.
- However, encodings can convey useful information if categorical values are sorted by their mean outcome value. This way, each label's code will represent an approximation of the target outcome.

Evaluations

Testing feature-sets and algorithms.

Baseline LinearRegression()

```
CV Scores:
 [0.2164809  0.21699175 0.21778808 0.21968502 0.21857161 0.2
1801418
 0.21727649 0.21810469 0.21669987 0.21771957]
Mean CV score:
 0.217733215497144
```

- Scores are in RMSE.
- CV is on X_dev and y_dev. Right figure is on X_val, y_val.
- Output must be between 0 and 1.
- Modification is limiting output to range.

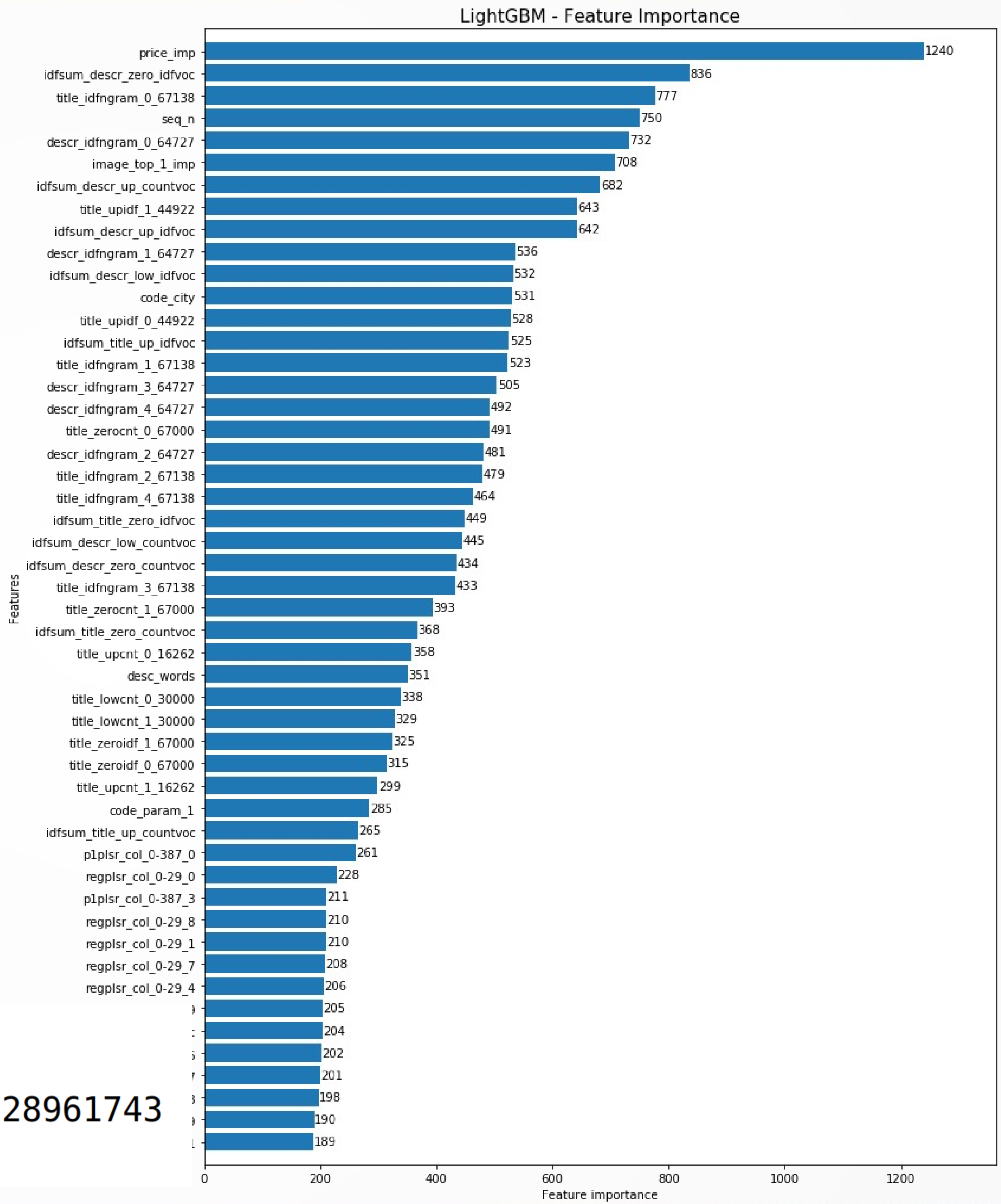
```
Less than zero: 17953
Over one: 246
RMSE without modification: 0.217884963631909
RMSE fit-to-range: 0.2175233073416146
```

PLSR 50 Components

- Took resulting 153 features engineered and reduced them to 50 components.
- Scores remained the same on PLSR's own prediction.

```
Less than zero: [17953]
Over one: [247]
RMSE without modification: 0.21789044528359755
RMSE fit-to-range: 0.21752825004059145
```

- **LightGBM** is a gradient boosting framework that uses tree based learning algorithms. It is designed to be distributed and efficient.
- Results below are using all 153 features.




```
Less than zero: 12674
Over one: 139
RMSE without modification: 0.20283926728961743
RMSE fit-to-range: 0.2027659024821594
```

LGBM on PLSR 50 Components

- Transformed all features with previous round of PLSR and fed them into LGBM. Scores are not better than LGBM on all features.

```
Less than zero: 4996
Over one: 50
RMSE without modification: 0.21042460691349882
RMSE fit-to-range: 0.2104147368480157
```

SelectFromModel(30 features)

- Compared feature selection techniques:

	Score	Selector
CoefRidge	0.21945	SelectFromModel(estimator=Ridge(alpha=1.0, cop...
FRegression	0.219495	SelectKBest(k=30, score_func=<function f_regre...
Importances	0.221562	SelectFromModel(estimator=ExtraTreesRegressor(...
CoefLasso	0.227862	SelectFromModel(estimator=Lasso(alpha=1.0, cop...

- Ridge gave the best scores.

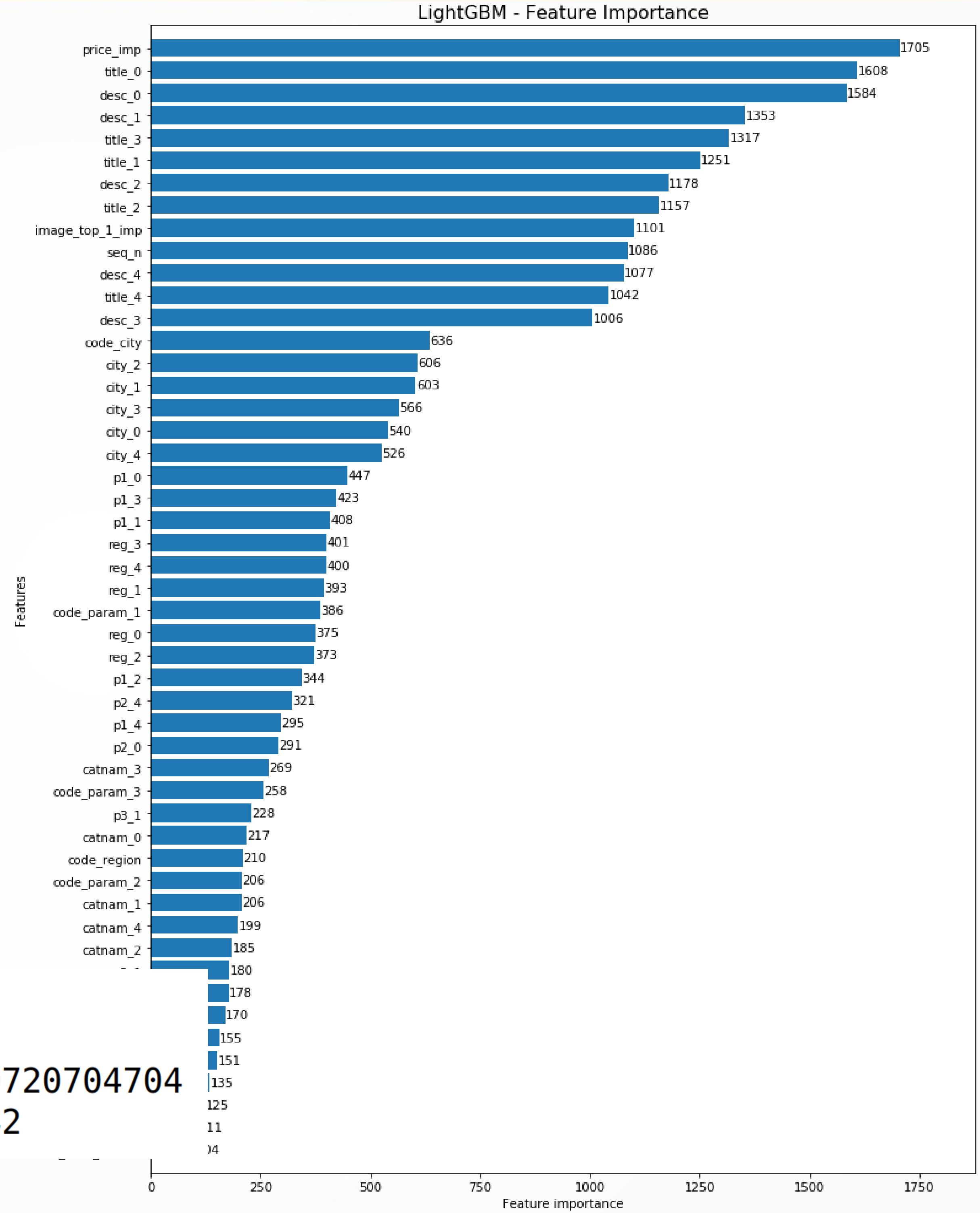
```
CV Scores: [0.21923302 0.21989553 0.21890318 0.21946869 0.21
97511 ]
Selection by Ridge Coefs: 0.219450302790982
```

LGB with 30 Ridge-Sel Features

- Took top 30 features based on SelectFromModel(Ridge), and did a train/test split for LGBM.

```
Less than zero: 5962
Over one: 49
RMSE without modification: 0.21136754065569408
RMSE fit-to-range: 0.2113501733446004З
```

- Took each original feature (title, descr, category, city) and did PLSR on all the features created from it.
- LightGBM using PLSR 10 components for each original feature.




```
Less than zero: 8101
Over one: 53
RMSE without modification: 0.20615780720704704
RMSE fit-to-range: 0.20612838300391242
```

Results Table

Predictor	RMSE X_val
LinearRegression. All engineered features.(153)	0.2175
PLSR 50 components.	0.2175
LGB. All engineered features.	0.2027
LGB on PLSR 50 Comp	0.2104
SelectFromModel(Ridge, 30 features)	0.2194
LGB with 30 Ridge Selected Features	0.2113
LGB with PLSR by Original Features	0.2061
Multi Layer Perceptron	0.2603
Keras	0.2110
RandomForestRegressor(100 trees)	0.2106

Conclusions

About the final product.

Product Summary I

- Based on several comparisons, the above-described feature-engineering pipeline followed by LightGBM is the best approach at predicting the deal probability of an online ad.
- **Why it Works.**
- It works due to the robustness and variety of feature-extraction and decomposition techniques. While feature extraction can generate a lot of data, this is only useful in a reduced dimensional space. Decomposing large CSR matrices produces predictively powerful components.

Product Summary II

- **What Problem it Solves**
- Online platforms for selling used goods rely on regular people selling their belongings online to achieve high-traffic. These sellers blindly sell their things with erroneous expectations and bad listing practices, thus becoming frustrated with online sales.
- **How it solves the problem.**
- Helping sellers understand the demand of their listings contributes in several ways: Informed sellers can optimize their listings for maximum deal probability and also optimize their choice of goods to sell, based on the deal probability of particular categories.

Product Summary III

- **How will it work in production?**
- In a production environment this model would learn from the sales information of a historic time window in order to predict the deal probability of new ads.
- Necessary maintenance would involve adjusting some of the feature-engineering procedures to ensure they are capturing the most valuable information.