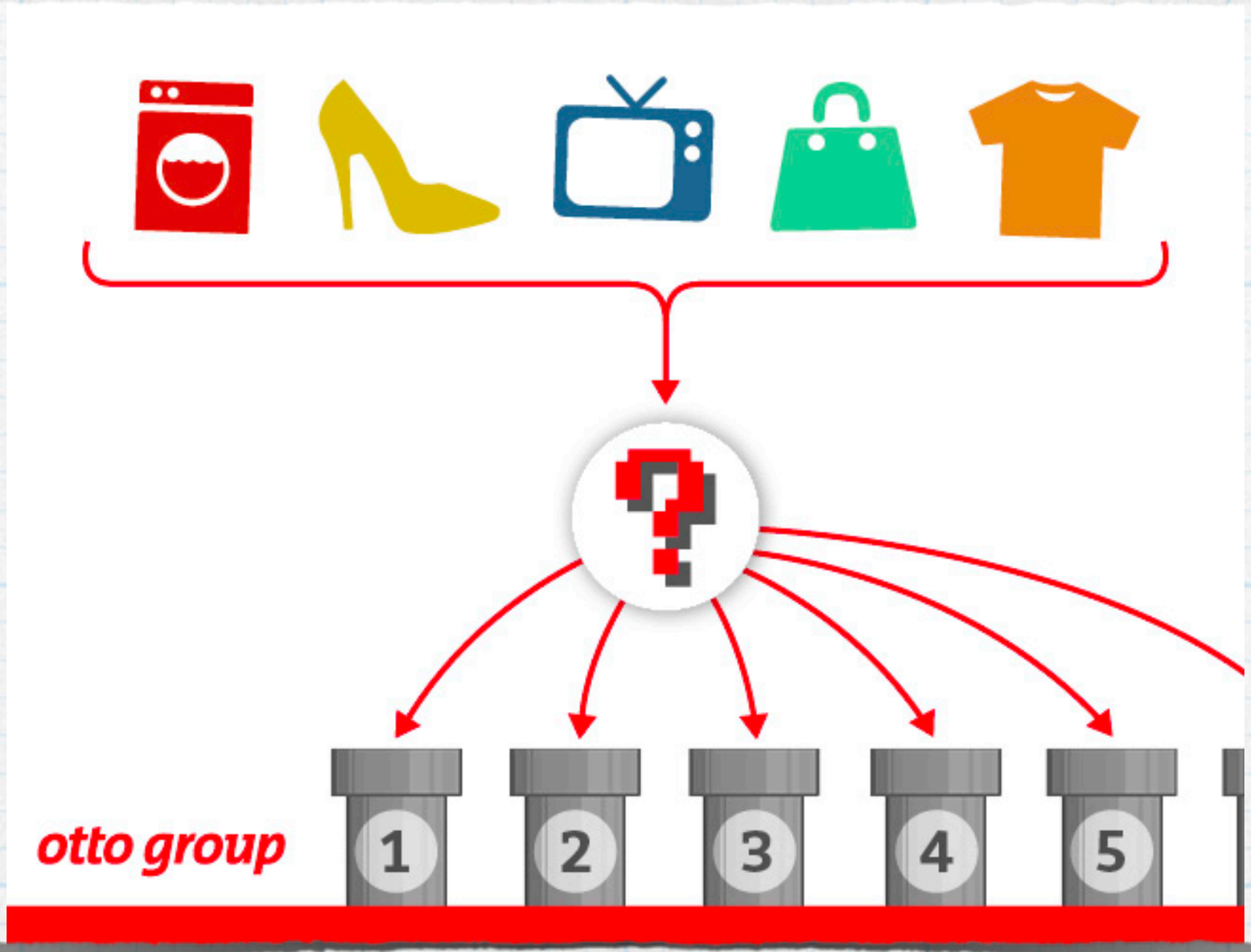
Product Classification



Author - Neha
Aug,2018

What's the problem?

* Otto group - one of the biggest e-commerce companies has a problem
* Thousands of products gets added every day
* For better analysis of products, they must be classified into correct product category
* Current methods not satisfactory, as identical products get classified into diff categories

Data

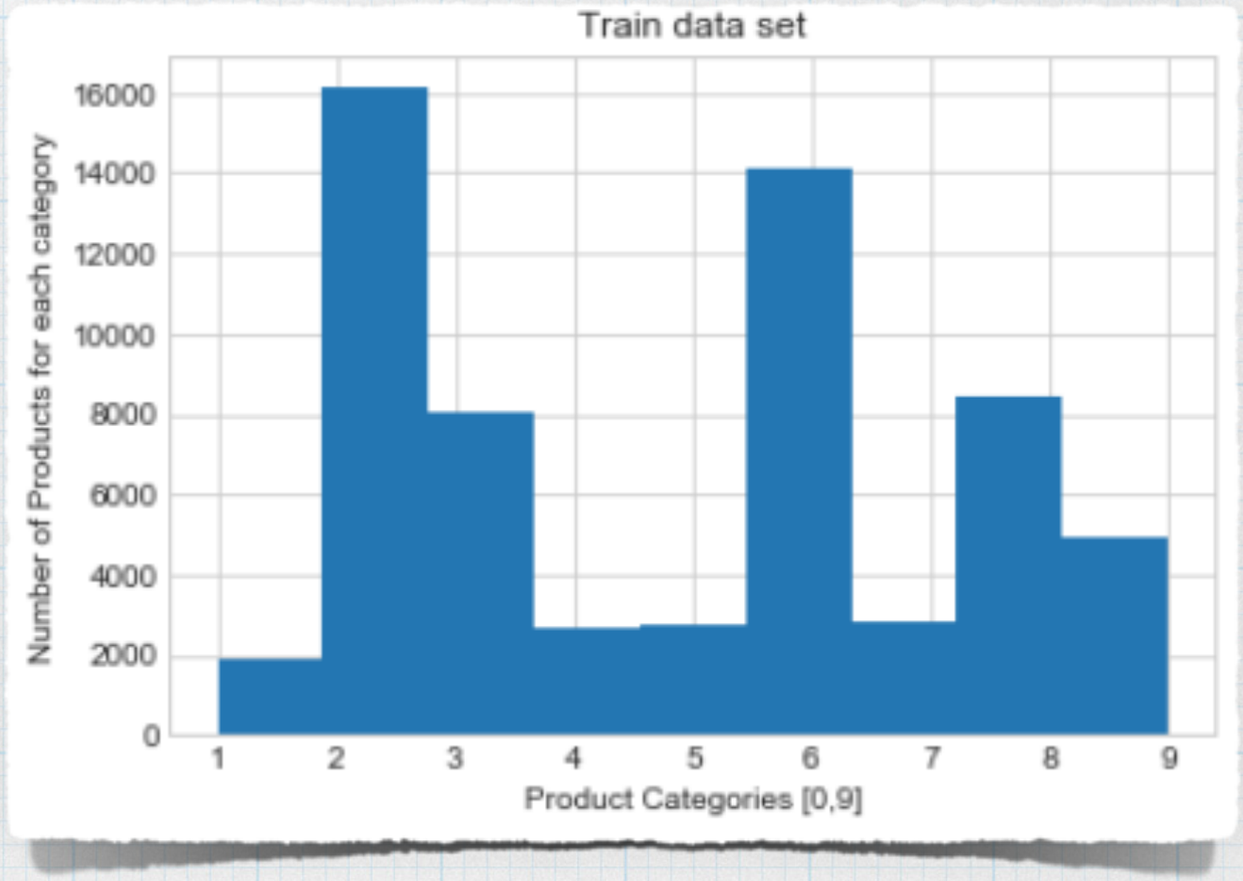
* 60K+ rows each corresponding to a product
* 93 columns or attributes related to products
* Overall 9 product categories

feat_1	feat_2	feat_3	feat_4	feat_5	feat_6	feat_7	feat_8	feat_9	feat_10	...	feat_85	feat_86	feat_87	feat_88	feat_89	feat_90	feat_91	feat_92	feat_93	target
1	0	0	0	0	0	0	0	0	0	...	1	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	...	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	...	0	0	0	0	0	0	0	0	0	1
1	0	0	1	6	1	5	0	0	1	...	0	1	2	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	...	1	0	0	0	0	1	0	0	0	1

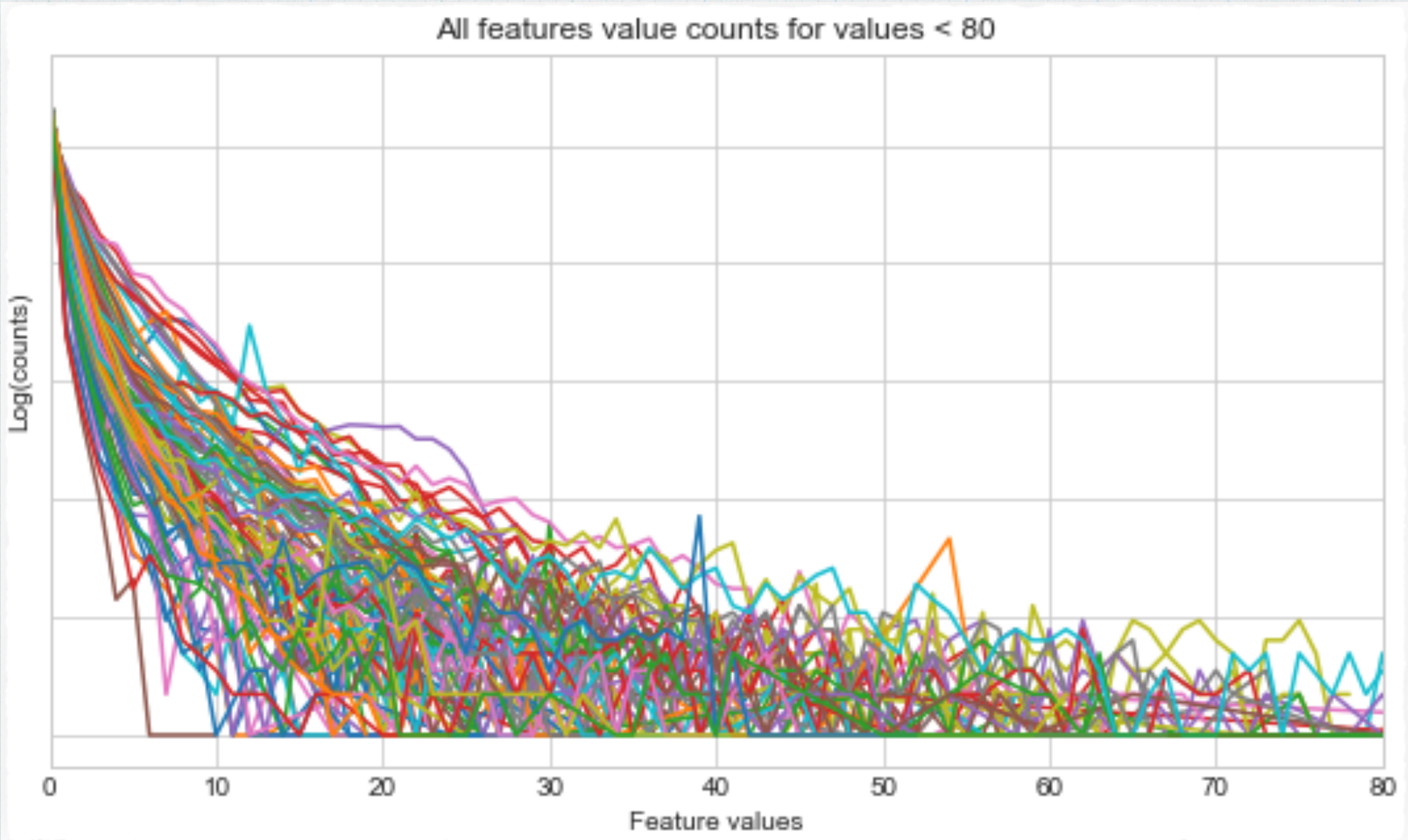
Challenges

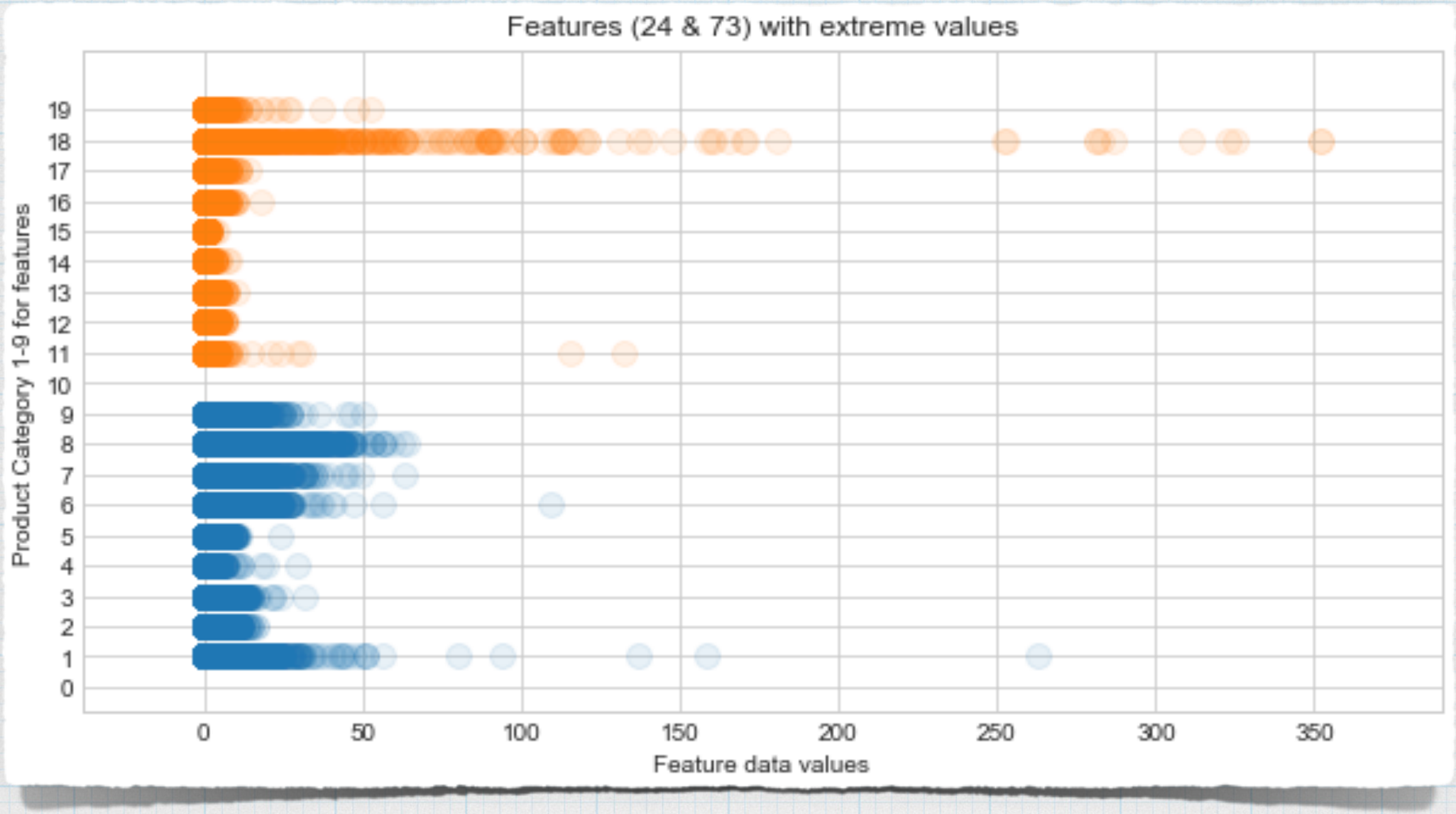
* Only numbers - no meaningful description

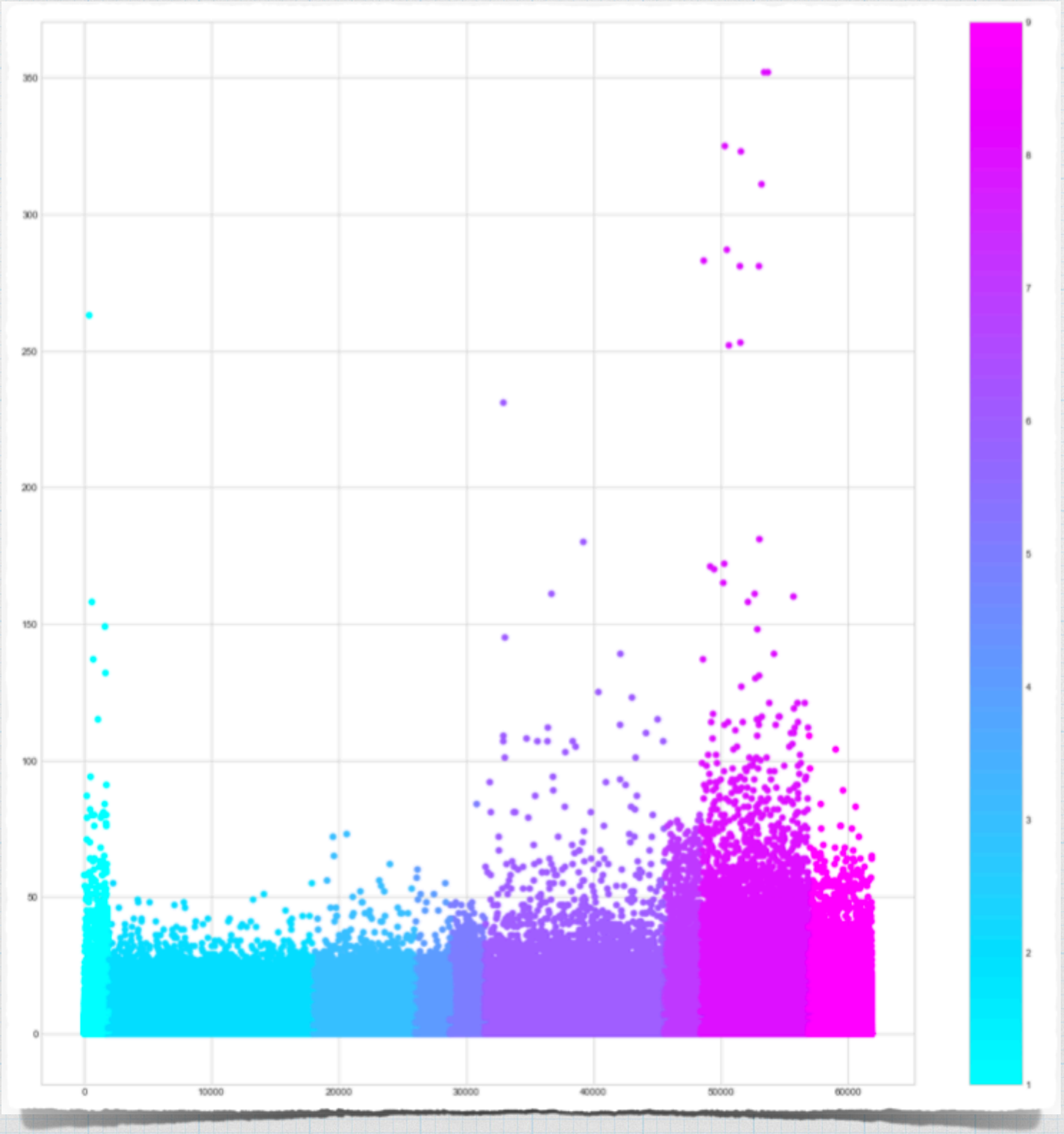
* Class imbalance - data amount varies for different product categories



How does the data look?

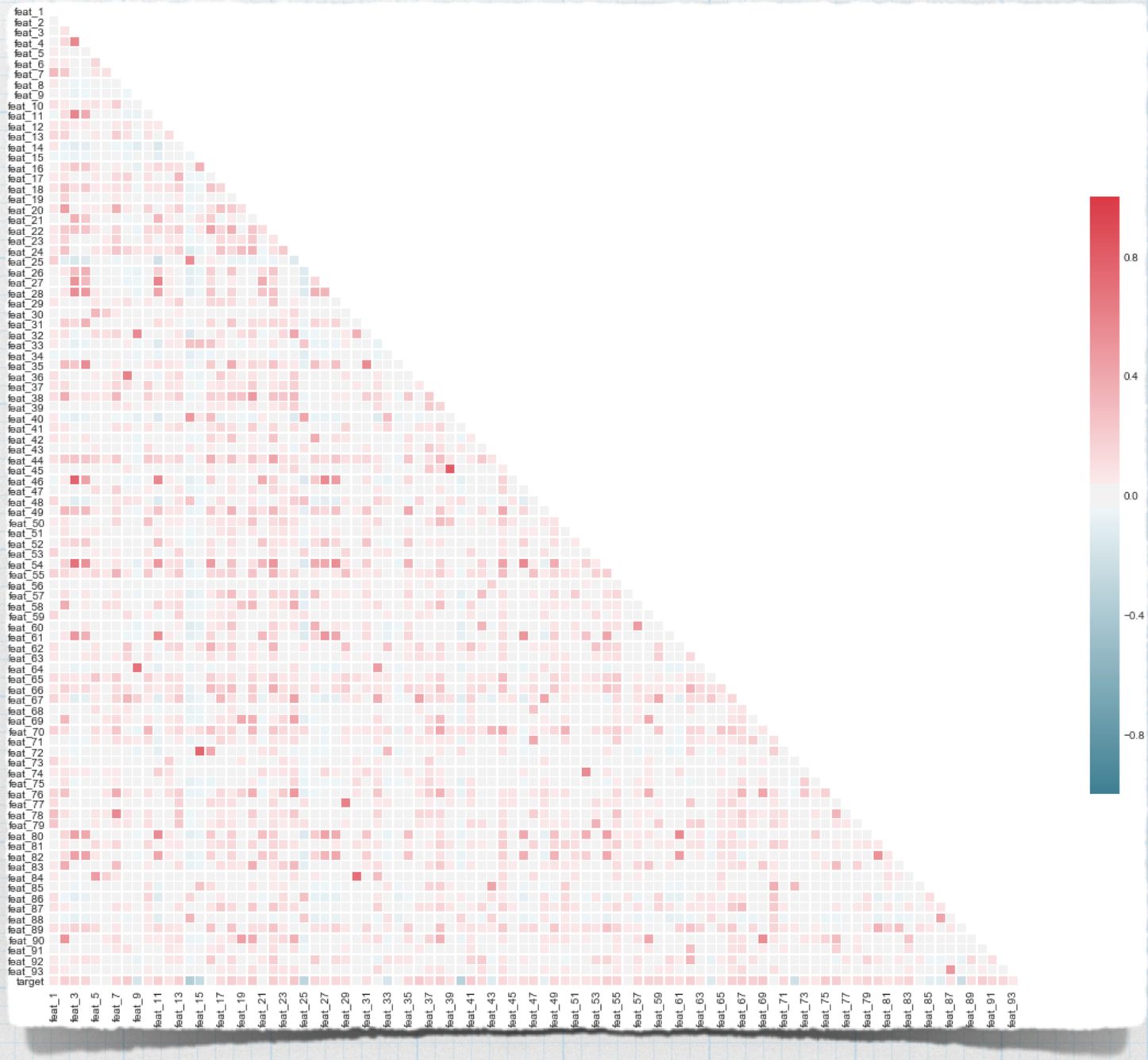



Features (24 & 73) with extreme values
Product Category 1-9 for features
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
0
50
100
150
200
250
300
350
Feature data values


350
300
250
200
150
100
50
0
0
10000
20000
30000
40000
50000
60000
9
8
7
6
5
4
3
2
1

Statistical Analysis

* Are some features co-related?
* Highly correlated pairs
* (8, 36), (39, 45) etc.



Correlations significant?

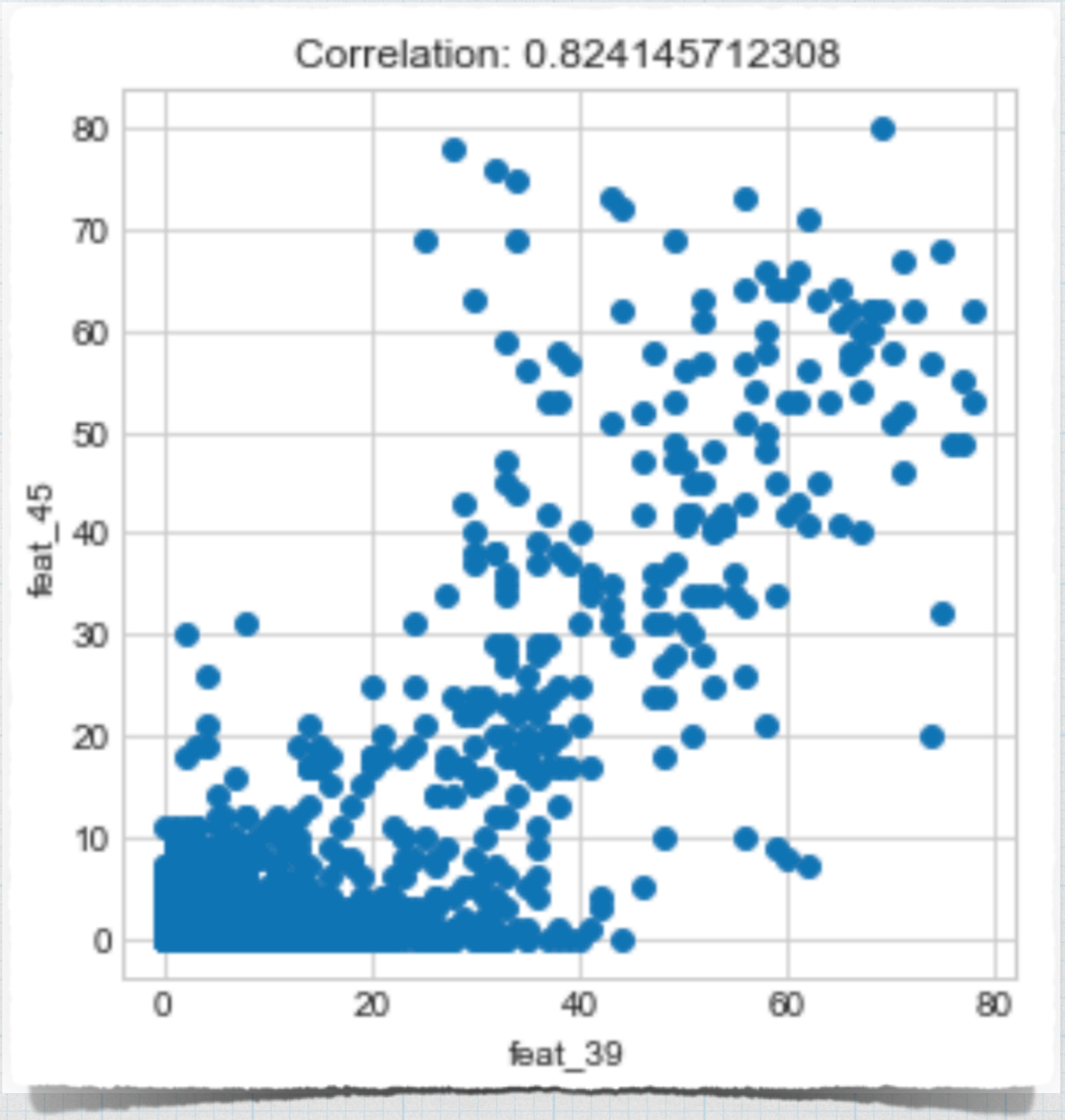
* Null Hypothesis Ho:
  ρ (correlation coefficient) = 0

* Alternate Hypothesis Ha:
  $\rho \neq 0$

* At significance level
  (α = 0.01 or 1%)

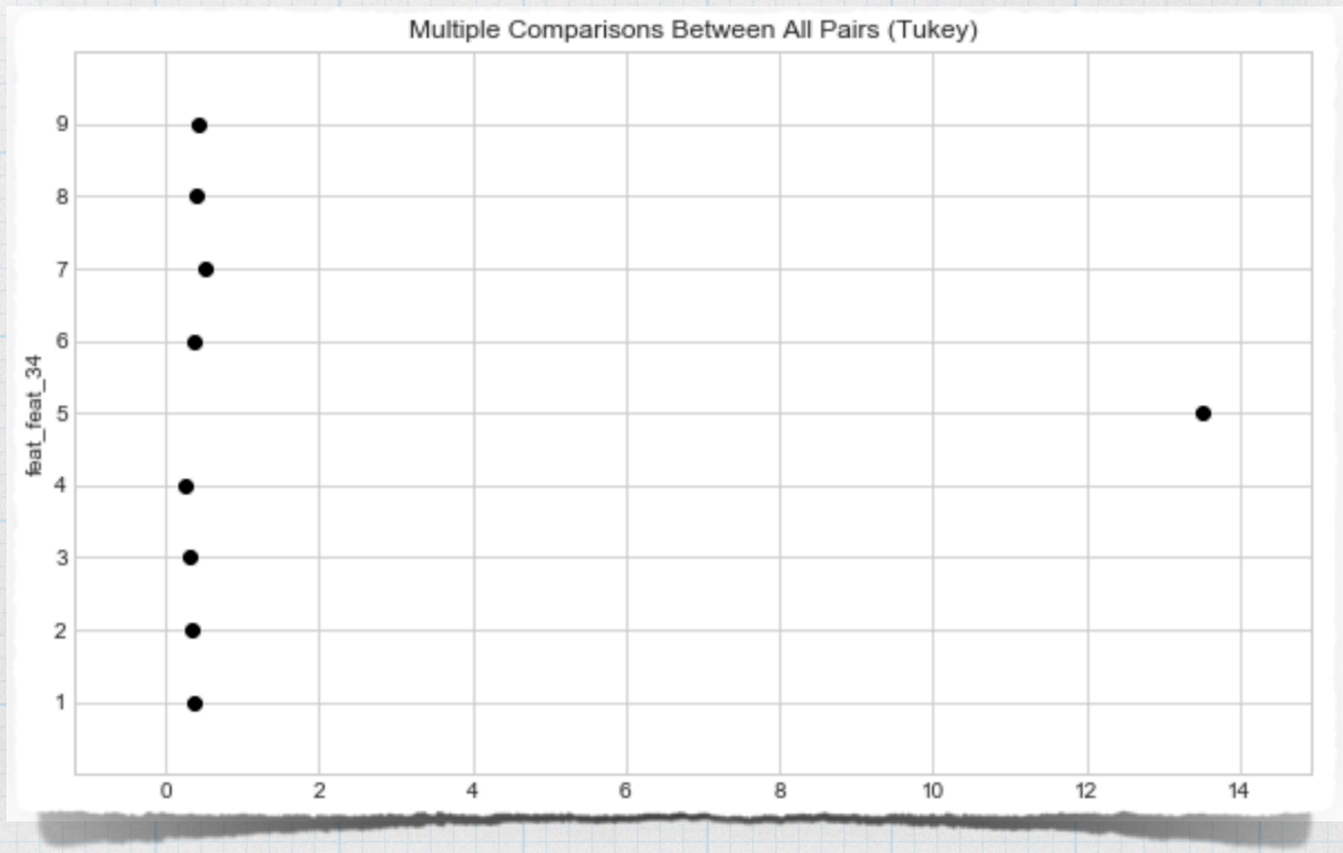
* Result: The p values were equal to 0 and we rejected the null hypothesis that the population correlation is zero.

* The pairs of features [(8, 36), (39, 45), (3, 46), (3, 54), (9, 64), (15, 72), (29, 77), (30, 84)] are statistically linearly correlated at significance level of 1%

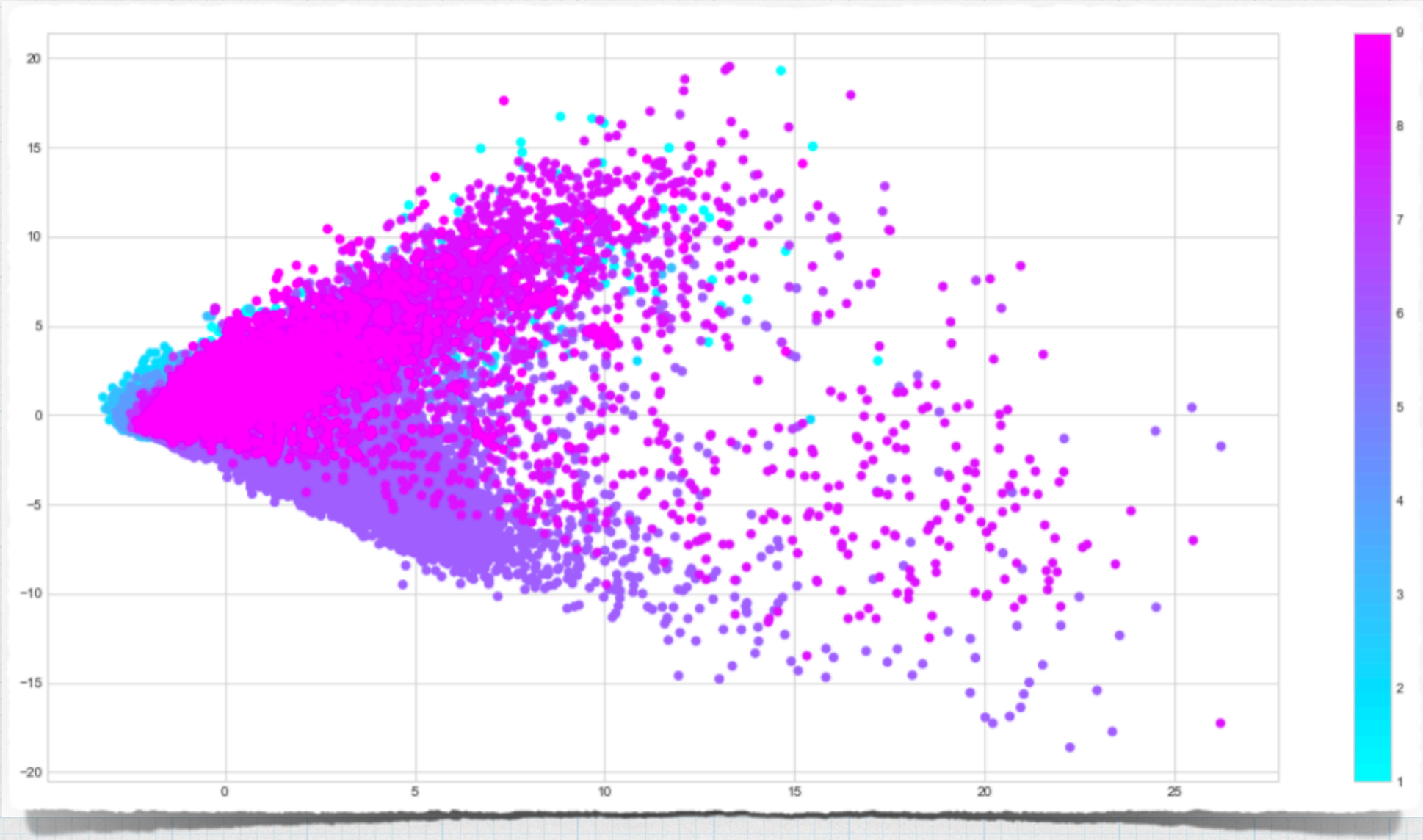


Imp features - ANOVA

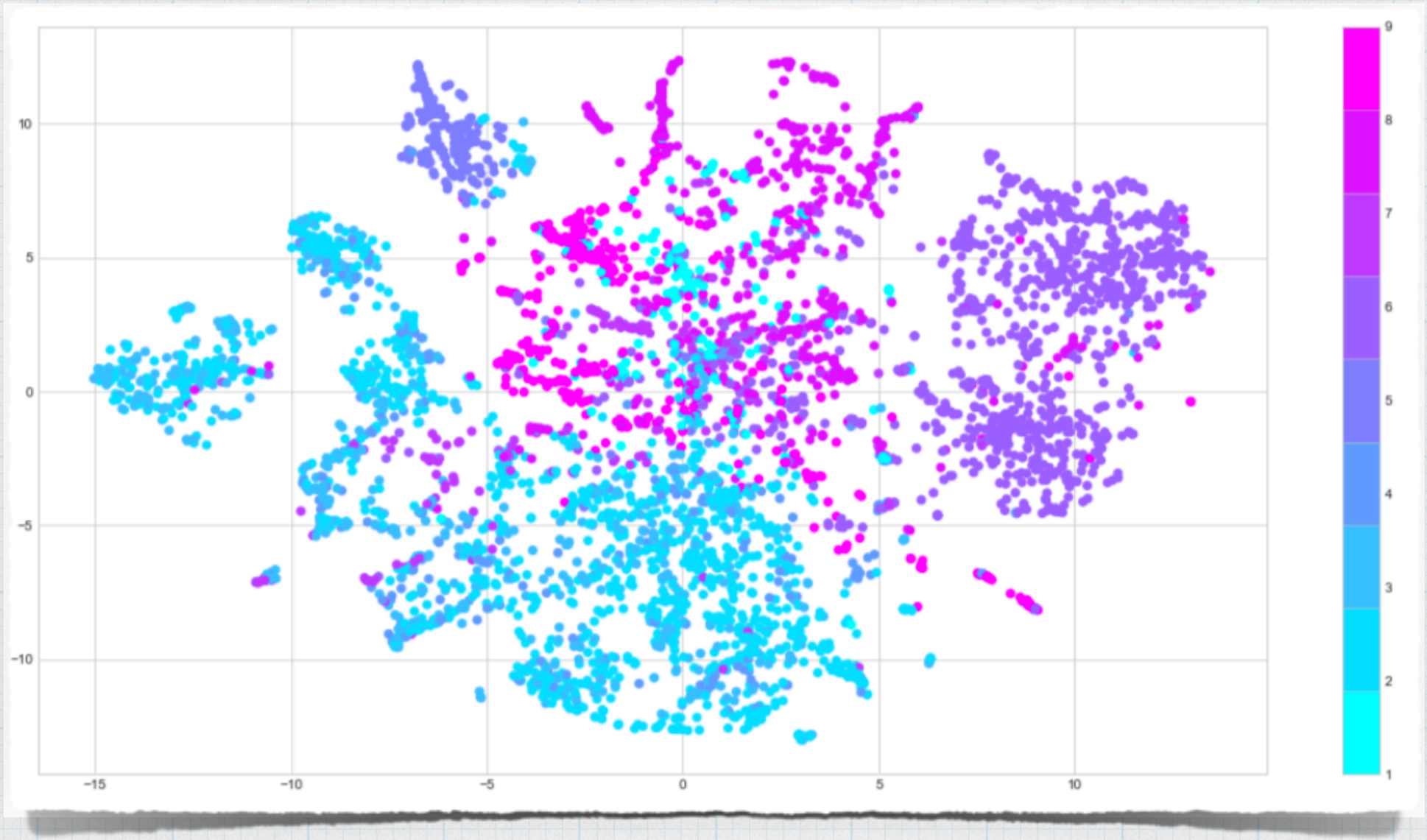
* How good a feature is to differentiate among classes
* Assumptions
  - homogeneity of variance
  - normally distributed classes
  - independent samples



Dimension Reduction

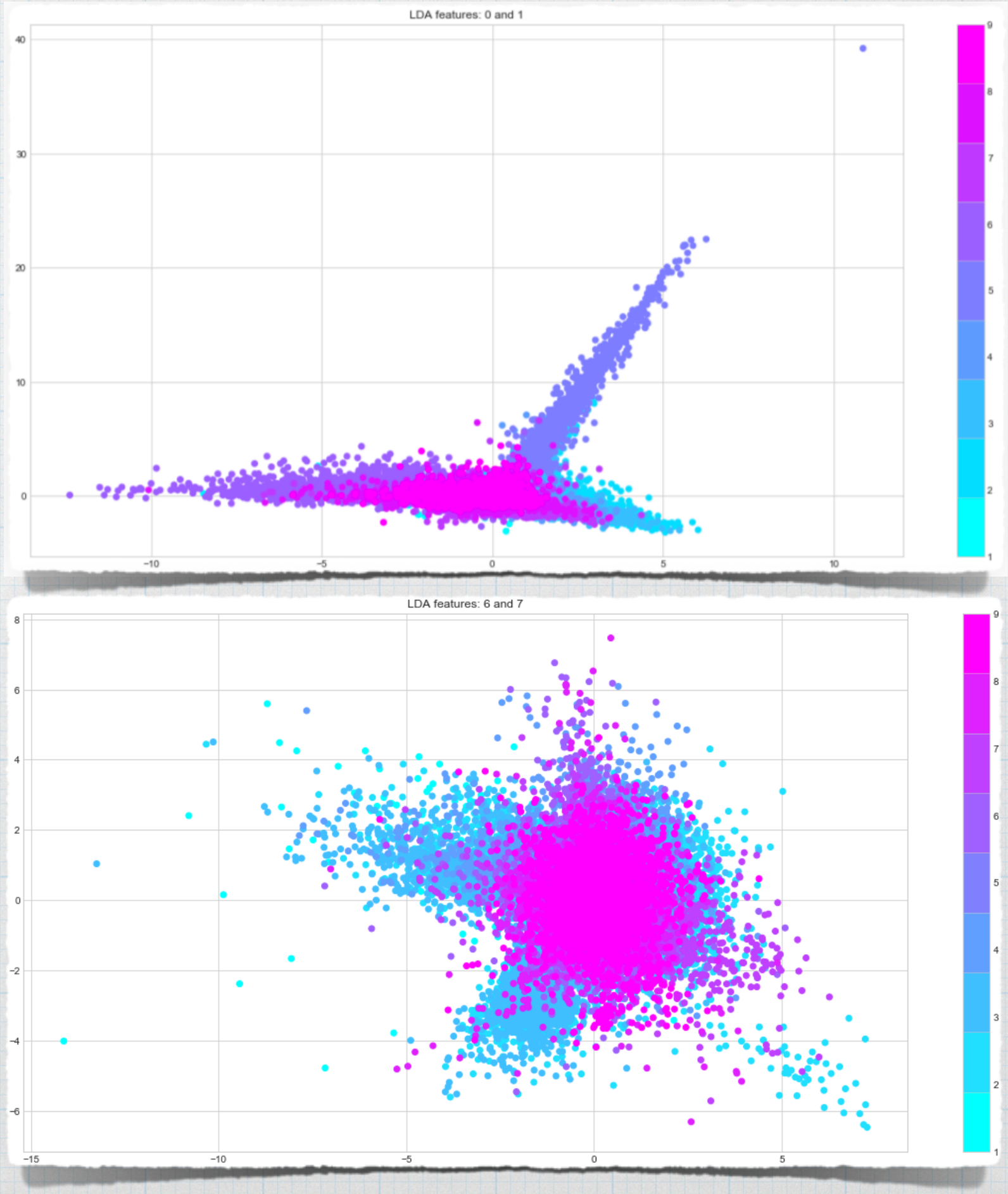


PCA - transform the variables to new set, known as principal components (pc) and are orthogonal. 1st pc retains maximum variation and it reduces as of move on to 2nd, 3rd pc and so on



t-SNE - It finds patterns in the data by identifying observed clusters based on similarity of data points with multiple features

LDA - It maximizes the separability among the known classes



ML Modeling

* Overfitting and Underfitting
* Model Evaluation Criteria - Accuracy, Precision, Recall, Log loss

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

* Cross Validation, Held out test set
* DRY principle - using Python functions

Baseline Results

* Baseline - always predict the most occurring class i.e. "2" in this case. The results will serve as comparison point for all our future predictions.
  Accuracy: 0.26 or 26%
  Log loss: 25.54

* Logistic Regression - multiple models with L1 & L2 regularization and different algorithms.

Parameters	Hyperparams	Test Results				
l		Log Loss	Precision	Recall	F1	Accuracy
liblinear, l1, ovr	C:10	0.66987	0.76	0.76	0.73	0.75614
saga, l1, ovr	C:10	0.67156	0.76	0.76	0.73	0.75549
saga, l1, multinomial	C:10	0.63743	0.76	0.76	0.75	0.76471
lbfgs, l2, ovr	C:10	0.67034	0.76	0.76	0.73	0.75501
lbfgs, l2, multinomial	C:100	0.63614	0.76	0.77	0.75	0.76568

Fix class imbalances

* Oversampling - generate more samples for the minority classes. Random Oversampling, SMOTE, ADASYN

* Undersampling - reduce the data from the majority classes. Random Undersampling, ClusterCentroids, NearMiss, ENN, IntanceHardnessThreshold

* Combination - SMOTEENN

* Class weights - weights applied inversely proportional to the number of rows for each class

Class Balacning Method	Test Results				
	Log Loss	Precision	Recall	F1	Accuracy
Normalized, Naïve Random Over Sampling	0.79605	0.77	0.71	0.73	0.71057
Normalized,ADASYN Over Sampling	0.67975	0.76	0.76	0.73	0.75663
Normalized,ADASYN Over Sampling	0.8348	0.75	0.69	0.7	0.68714
SMOTE random over sampling	0.78492	0.77	0.71	0.72	0.70814
Prototype generation under sampling	0.99595	0.76	0.59	0.62	0.58759
Random under sampling	0.81376	0.77	0.71	0.73	0.71202
Near miss under sampling - version1	2.35544	0.79	0.63	0.7	0.54945
Near miss under sampling - version2	1.03002	0.72	0.57	0.59	0.57143
Near miss under sampling - version3	0.89183	0.72	0.7	0.71	0.70023
EditedNearestNeighbours	0.82391	0.76	0.75	0.73	0.7521
InstanceHardnessThreshold	1.91493	0.72	0.7	0.71	0.52666
SMOTEENN under and over sampling	1.07592	0.78	0.64	0.65	0.63849
Weighted classes (weights inversely proportional to the number of rows for each class)	0.73284	0.76	0.75	0.75	0.75323
Logistic Regression with one vs all approach (9 times for each class)	0.67151	0.76	0.76	0.73	0.75533
Logistic Regression with one vs all approach (9 times for each class) - Balanced	0.94957	0.75	0.73	0.74	

Other Models

* Multinomial Naive Bayes
* Support Vector Machine
* KNN
* Ensemble - Random Forest, Gradient Boosting
* Stacking - Meta classifier applied to the results from first layer classifiers

Model	Parameters	Log Loss Result
MultinomialNB	alpha=1.0	3.45009
SVC	C=1.0, gamma='auto', kernel='rbf'	0.56515
KNeighborsClassifier	n_neighbors=5	1.43225
RandomForestClassifier	max_depth=10, max_features=20,n_estimators=500	0.90412
GradientBoostingClassifier	max_depth=5, max_features=20, n_estimators=1000	0.49744
as (Logistic, GaussianNB, SVC) and meta classifier as LogisticRegression		1.34294
as (KNN, Logistic, RandomForest) and meta classifier as LogisticRegression		1.89438
EnsembleVoteClassifier with LogisticRegression, RandomForest, GradientBoost	weights=[1,1,2] respectively	0.51784

Final Model

The best model is the GradientBoost followed by SVM, RandomForest and LogisticRegression.

Summary of GradientBoost model results:

- Log Loss: 0.49
- Accuracy: 82%
- Precision: 0.82
- Recall: 0.82
- F1: 0.82

Recommendation

* Since consistent analysis of the products is crucial for the Otto Group, I would recommend to first classify the products by using the Predictive Model.

* Once the products are classified into correct categories, smoother and more accurate analysis can be done.

* The model is not perfect, as all models are wrong. But it gives accuracy of 82%, i.e. our prediction of the product category is correct 82 out 100 times.

Conclusions

* Otto group product classification turned out to be a hard challenge
* Data very unique in several senses. The signal from the data seems to be evenly spread. No single feature is contributing hugely as compared to others.
* Feature engineering would have been possible only if there was some meaning associated with data.
* Stacking and ensemble models could improve the predictions significantly.
* Since data is huge Neural networks are worth trying.