Large Scale Machine Learning - Assignment 4

Abhishek Sinha, Arun Sai, Ashish Bora

May 6, 2016

1 Feature Extraction

In this assignment we didn't focus much on data cleaning. We instead concentrated on feature extraction part. For data cleaning we relied on the public scripts that were posted on Kaggle. These scripts removed redundant columns and identified outliers and appropriately modified their feature values. Below we describe some of the features we extracted.

1.1 Random Feature Combinations

Here we generated new features by combining original features using arithmetic operators : $+, -, *, /$.

feature	AUC (Validation Set)
$pca4 - ind\_var30$	0.7459055
$n0 + var15$	0.7423217
$pca\_all0 + num\_var13\_0$	0.7143601
$pca4 + num\_op\_var39\_ult1$	0.7129791
$pca1 + var15$	0.7128941
$ind\_var41\_0 - pca2$	0.7096776

Table 1: Table shows some of the important feature combinations found. Because of the huge number of possible combinations we randomly generated combinations and picked the top few and used them in training. The last column shows the AUC obtained on the validation set by fitting a decision tree($depth = 3$) using the feature on the left column. Features Names: $pca*$ denotes features obtained through PCA on the original dataset and $n0$ represents number of zeros.

1.2 Random Rotation of Feature Space

This is related to the question we posted on piazza about training better decision trees by finding a rotated feature space on which decision trees can generalize better. We couldn't come up with a principled & efficient approach to do this. So we randomly rotated the feature space and trained a model using XGB on the rotated space. However random rotations didn't work well (AUC approximately dropped by 10%).

1.3 KNN

This feature is based on the KNN classifier. Here we first built a kd-tree on the combined dataset of train and test sets. We then extracted features such as : number of positive labels that are close to a given point. This feature resulted in over fitting (the public LB score is significantly lower than the CV score obtained when this feature was used in training).

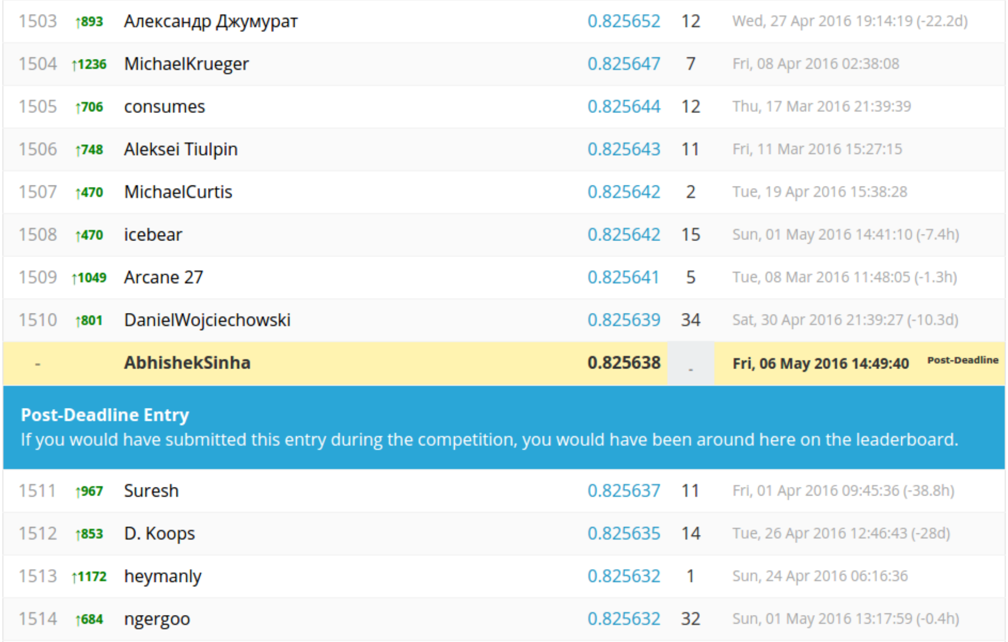
1.4 Other Features

- *PCA:* Extracted the top 10 principal components using the entire training data and using only the important features.
- *KMeans:* Here we clustered the training and test datasets and added the resulting cluster id of each point as a feature.

2 Training

We trained the following models: XGB, Random Forests, Extremely Random Trees, LR, Neural Nets. Needless to say, XGB performed the best out of these classifiers. To reduce the training time, we trained most of our models with ∼50 features that are important. This didn't significantly change the public LB score.

2.1 XGB



1503	↑893	Александр Джумурат	0.825652	12	Wed, 27 Apr 2016 19:14:19 (-22.2d)
1504	↑1236	MichaelKrueger	0.825647	7	Fri, 08 Apr 2016 02:38:08
1505	↑706	consumes	0.825644	12	Thu, 17 Mar 2016 21:39:39
1506	↑748	Aleksei Tiulpin	0.825643	11	Fri, 11 Mar 2016 15:27:15
1507	↑470	MichaelCurtis	0.825642	2	Tue, 19 Apr 2016 15:38:28
1508	↑470	icebear	0.825642	15	Sun, 01 May 2016 14:41:10 (-7.4h)
1509	↑1049	Arcane 27	0.825641	5	Tue, 08 Mar 2016 11:48:05 (-1.3h)
1510	↑801	DanielWojciechowski	0.825639	34	Sat, 30 Apr 2016 21:39:27 (-10.3d)
-		**AbhishekSinha**	**0.825638**	-	**Fri, 06 May 2016 14:49:40** Post-Deadline
Post-Deadline Entry If you would have submitted this entry during the competition, you would have been around here on the leaderboard.					
1511	↑967	Suresh	0.825637	11	Fri, 01 Apr 2016 09:45:36 (-38.8h)
1512	↑853	D. Koops	0.825635	14	Tue, 26 Apr 2016 12:46:43 (-28d)
1513	↑1172	heymanly	0.825632	1	Sun, 24 Apr 2016 06:16:36
1514	↑684	ngergoo	0.825632	32	Sun, 01 May 2016 13:17:59 (-0.4h)

Figure 1: XGB

2.2 Logistic Regression

We tried logisitc regression with L1 and L2 regularization. The C parameter was selected by standard 5-fold cross validation with logarithmic sampling and roc-auc metric. The score with L1 regularization was considerably better (0.7924) as compared to L2 regularization (0.615). The dataset is very sparse. Since L1 encourages sparse weight vectors, it probably does a better job of finding good features from a very sparse dataset.

2.3 Neural Nets

We used the Keras script given in the forums. This uses 3 fully connected layers from 81 to 400, 400 to 200 and finally 200 to 1. It uses *tanh* non-linearities. There is also a sigmoid at the end and the loss used is again binary cross entropy. The whole network is trained with the Adam Optimizer. It seems that the author faced problems of skewed data. To address this, they just end up using only around 6000 training examples (3000 each of positive and negative) so as to make their dataset balanced. Despite getting rid of so much data, this network achieved a public Kaggle score of 0.789466.

Post-Deadline: Fri, 06 May 2016 16:59:47 keras.csv 0.789466 □

nn-kaggle-script

Edit description

Figure 2: Neural Network

3 Ensemble

We experimented with various ensembling techniques for this assignment. We ensembled all the classifiers described in the previous section. Here are different approaches we tried:

- *Weighted average:* average based on the CV score.
- *Rank averaging:* weighted average after converting to ranks.
- *Blending:* Here we trained a meta model which takes the predictions from all the base classifiers and learns to combine their predictions. We experimented with various meta models: LR, RF, XGB.

While blending (with XGB/RF as meta models) performed better than simple averaging, it didn't significantly outperform the best XGB model (*i.e,* we didn't see much gains using blending).