CS412 Final Project: Human or Robot?

Yilin Zhao(zhao100), Ka Ki Lai,Erica (kakilai2), Chenzhi Wang(cwang120)

Abstract

Why do you always fail in an online auction? An increasing number of robots used in the online bidding makes it more difficult for a human to land a winning bid. Based on the data sets provided by Kaggle, we implemented three classification algorithms trying to separate the bids from human and robots. In first section, a feature engineering method is stated to select the useful features. In the following three sections, we introduce the algorithm, discuss the features and parameters we are using and show the final performances of each algorithms.

Feature Engineering

Based on the provided **features.csv** we tried to add some new features to better describe the difference between robot and human. The new features are listed below

Features Name	Calculation
Fraud Score of Country	$\sum_{i=1}^{n} \frac{Frequency(country=I\&outcome=1)}{n \cdot Frequency(country=I)}$, where n is the appearance times of each bidder_id
Fraud Score of Auction	$\sum_{i=1}^{n} \frac{Frequency(auction=I\&outcome=1)}{n \cdot Frequency(auction=I)}$, where n is the appearance times of each bidder_id
Fraud Score of device	$\sum_{i=1}^{n} \frac{Frequency(device=I\&outcome=1)}{n \cdot Frequency(device=I)}$, where n is the appearance times of each bidder_id
Fraud Score of merchandise	$\sum_{i=1}^{n} \frac{Frequency(merchandise=I\&outcome=1)}{n \cdot Frequency(mewchandise=I)}$, where n is the appearance times of each bidder_id

Table 1.1 Introduction of new features

However, after we implemented several algorithms and did some variable selection we found that the original data file performs slightly better. Considering in addition that it has smaller number of features, we ultimately adopted the original feature.csv file.

Algorithms Description & Packages

1 Random Forest (Python)

Algorithm Introduction:

Random forests is an ensemble method of classification which combines k base classifiers to create an improved composite model for classification. For a given data set, k training data sets would be created to generate k learnt models. For a give data tuple, each of the classifier would return a class prediction. The ensemble will then make prediction on the data tuple according to the overall probability of class membership based on this composite classification model.
For Random forests, each decision tree is built by a random selection of attributes at each node to find the best split. By introducing randomness in best-split selection, correlation between trees built will be reduced since the split in each node are based on different predictors. Since Random forests would consider fewer features at each split during the process of tree building, the algorithm is more efficient, especially toward large datasets. It could also estimate variable importance to identify important predictors.

Packages used:

- To build the Classification Trees, the *DecisionTreeClassifier* from package *sklearn.tree* is used.
- To generate random samples to build tree, *shuffle* from package *sklearn.utils* is applied and it works as below:
  $X = [[1., 0.], [2., 1.], [0., 0.]]$
  $y = [0, 1, 2]$
  $X, y = shuffle(X, y, random\_state = 0)$
  $\#output:$
  $X = [[0., 0.], [2., 1.], [1., 0.]]$
  $y = [2, 1, 0]$

Data Preprocessing:

The Major step in data preprocessing include converting variable types and handling missing values. Since the package for tree building only support numeric variables, features are converted to numeric variables. Data preprocessing towards training and testing samples includes following steps:

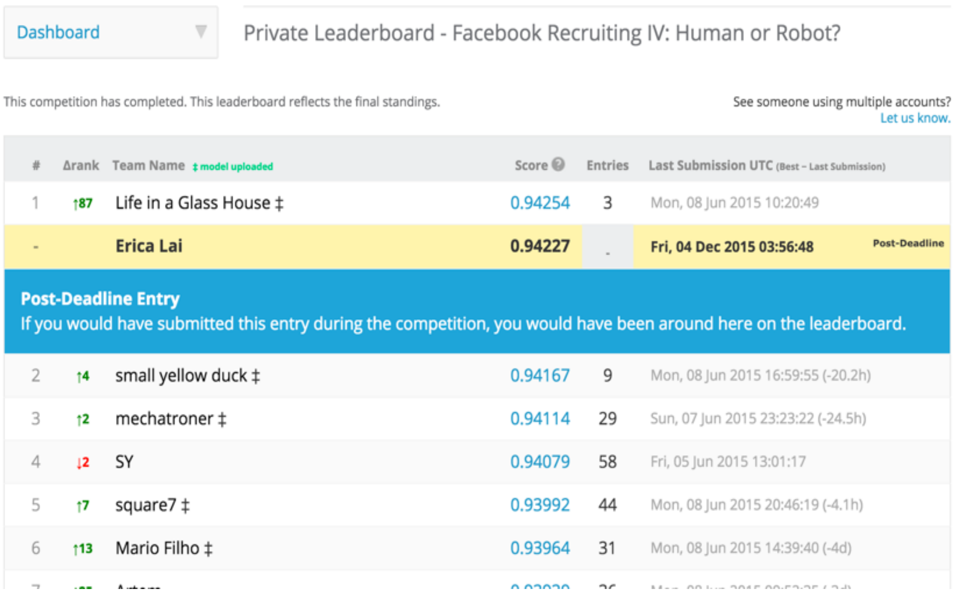
- Convert String variables **most_common_country** into Numeric variables $(0, 1, 2...n)$

- Convert Boolean variables **payment_account_prefix_same_as_address_prefix** and **sleep** into Numeric variables ($False = 0, True = 1$)
- Replace all missing value by -1e30

Pseudo code:

```
for $i$ $in$ $1 : N_{tree} = 800$ do
    Randomly bootstrap sample of size $N = 85\%$ of data points from $traindata$.
    Build Random Forest Tree, $Tree_i$ based on this subset by repeating following steps in each terminal node:
        • select $\sqrt{m}$ features at random from all $m$ features in the data set
        • pick the best feature, $M_k$ among the $\sqrt{m}$ features based on information gain for the best split.
        • Split the node by $M_K$ into two daughter nodes
    Predict the class probabilities of the test samples: The predicted class probability is the fraction of samples of the same class in a leaf.
end
For each data point in the test set, make a prediction on the class it belongs to by calculating the average class probability of all the trees:
Classification:
Let $P_b(x)$ be the prediction of class probability of test data point $x$ by the $Tree_b$,
The Overall Probability of Prediction for Classification = $\sum_{i=1}^{N_{tree}} P_i(x)/N_{tree}$.
where $N_{tree}$ is total number of trees built
```

Submission:



Private Leaderboard - Facebook Recruiting IV: Human or Robot?

This competition has completed. This leaderboard reflects the final standings.

See someone using multiple accounts? Let us know.

#	Δrank	Team Name ‡ model uploaded	Score	Entries	Last Submission UTC (Best – Last Submission)
1	↑87	Life in a Glass House ‡	0.94254	3	Mon, 08 Jun 2015 10:20:49
-		Erica Lai	0.94227	-	Fri, 04 Dec 2015 03:56:48 Post-Deadline
2	↑4	small yellow duck ‡	0.94167	9	Mon, 08 Jun 2015 16:59:55 (-20.2h)
3	↑2	mechatroner ‡	0.94114	29	Sun, 07 Jun 2015 23:23:22 (-24.5h)
4	↓2	SY	0.94079	58	Fri, 05 Jun 2015 13:01:17
5	↑7	square7 ‡	0.93992	44	Mon, 08 Jun 2015 20:46:19 (-4.1h)
6	↑13	Mario Filho ‡	0.93964	31	Mon, 08 Jun 2015 14:39:40 (-4d)
7	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Post-Deadline Entry
If you would have submitted this entry during the competition, you would have been around here on the leaderboard.

Plot 1. Submission of Random Forests

Score: 0.94227

2 Support Vector Machine (Matlab)

Algorithm Introduction:

Support vector machine(SVM) is a supervised learning model used for classification and regression analysis in machine learning. Given a training data set, the SVM separates the different categories by a clear gap, which is called a hyper-plane, in a high-dimensional space. This hyper-plane is constructed as wide as possible to make it has the largest distance to the nearest training data of any category. Test data are then mapped into the same space and be predicted into different categories based on the which side of the hyper-plane they fall on.
In 1992, Boser, Guyon and Vapnik introduced the kernel trick to maximize the margin (distance between classes) of the hyper-plane, which makes SVMs an efficiently perform a non-linear classification. In non-linear classification, spline kernel, polynomial kernel and Gaussian radial basis function kernel are commonly used to map the features into a transformed feature space.

Data Processing:

The SVM requires all features to be numeric and without missing data. Therefore, data processing in Matlab of training and testing data set is provided below:

- Convert String variables **most_common_country** into Numeric variables $(0, 1, 2...n)$.
- Convert Boolean variables **payment_account_prefix_same_as_address_prefix** and **sleep** into Numeric variables $(False = 0, True = 1)$.
- Replace all missing value by -9999.
- Separate the test dataset from train dataset based on (**outcome** = -1).

Algorithm Issue:

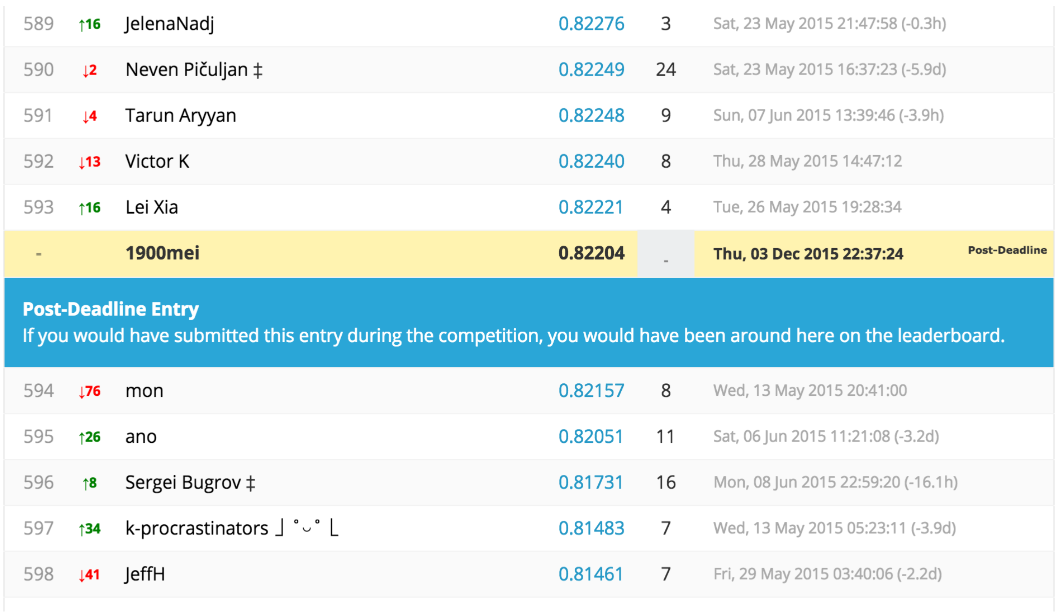
- Feature Selection: The over 400 features caused a over-fitting issue in the SVM classification. After we tried the PCA on the training data and also check the eigenvalues of the co-variance matrix of all the features, We found **n_bids** and **bids_per_auction_mean** are the most powerful features among them.
- Imbalance data: It is highly imbalanced in the training data set with over 1900 samples in the negative class but only 108 samples in positive class. Therefore, we did the cross validation by selecting 500 negs and 100 pos from the training set every time.
- Kernel Selection: There are several kernel functions we can use in the SVM training, after we tried the linear, polynomial and RBF kernel on the training set, the RBF kernel outperformed the other two.
- Two parameters are used in the RBF SVM model. The box constraint **C** tells the SVM optimization how much you want to avoid miss-classifying each train sample.The parameter γ is involved in the RBF kernel function. In order to optimize the hyperparameters of the RBF SVM, we did multiple grid searchs and get **C=46** and γ=**0.0001** give the best performance.

Pseudo code:

```
for $i$ $in$ $1 : CV = 10$ do
    Randomly select samples of size $neg = 500$ and $pos = 100$ data points from $traindata$.
    Construct SVM models and predict the categories of each test samples.
end
Apply the Majority Vote Classifier on each of the test sample point and make the final prediction of the class on the test set.
The algorithm of SVM model is below:
```

- Maximize the dual problem of
  $\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(X_i, X_j)$
  where $k(X_i, X_j) = exp(-\gamma \parallel X_i - X_j \parallel^2$
  constraint to:
  $0 \leq \alpha_i \leq C$
  $\sum_{i=1}^{n} \alpha_i y_i = 0$
- $w = \sum_i \alpha_i y_i X_i$
- bias is calculated by $b = \frac{1}{n_{SV}} \sum_{i=1} n_{SV} (w \cdot X_i - y_i)$

Submission:



589	↑16	JelenaNadj	0.82276	3	Sat, 23 May 2015 21:47:58 (-0.3h)
590	↓2	Neven Pičuljan ‡	0.82249	24	Sat, 23 May 2015 16:37:23 (-5.9d)
591	↓4	Tarun Aryyan	0.82248	9	Sun, 07 Jun 2015 13:39:46 (-3.9h)
592	↓13	Victor K	0.82240	8	Thu, 28 May 2015 14:47:12
593	↑16	Lei Xia	0.82221	4	Tue, 26 May 2015 19:28:34
-		**1900mei**	**0.82204**	-	**Thu, 03 Dec 2015 22:37:24** Post-Deadline

Post-Deadline Entry
If you would have submitted this entry during the competition, you would have been around here on the leaderboard.

594	↓76	mon	0.82157	8	Wed, 13 May 2015 20:41:00
595	↑26	ano	0.82051	11	Sat, 06 Jun 2015 11:21:08 (-3.2d)
596	↑8	Sergei Bugrov ‡	0.81731	16	Mon, 08 Jun 2015 22:59:20 (-16.1h)
597	↑34	k-procrastinators ˩ ˚◡˚ ˪	0.81483	7	Wed, 13 May 2015 05:23:11 (-3.9d)
598	↓41	JeffH	0.81461	7	Fri, 29 May 2015 03:40:06 (-2.2d)

Plot 2. Submission of SVM

3 Adaboost based on Classification Tree (R)

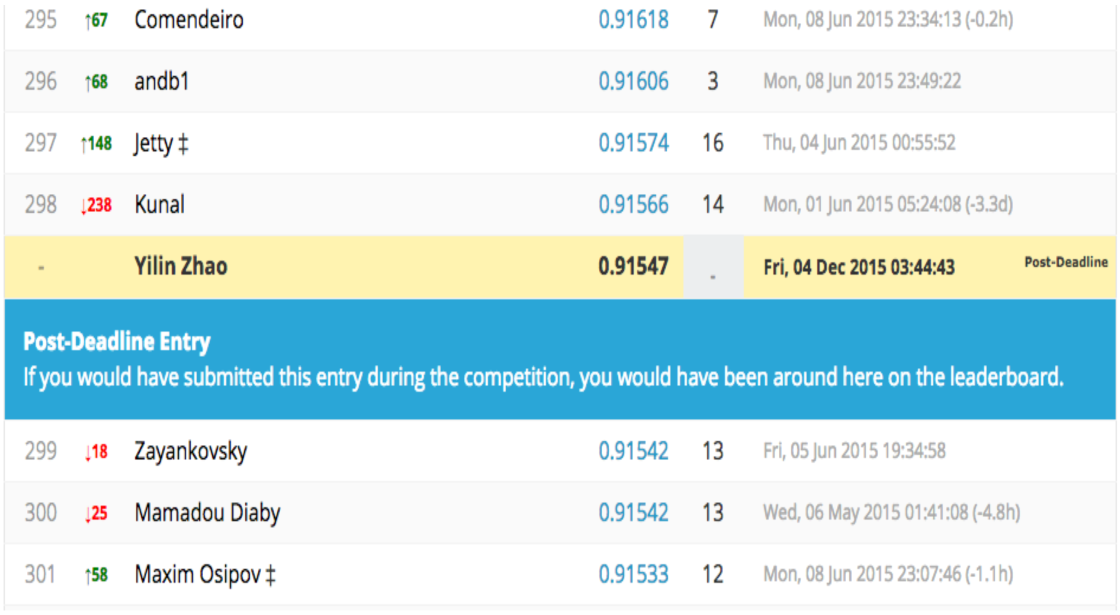
Data Processing:

- Separate the original data set into training and test data set based on **outcome**.
- Replace all the missing values by -9999.
- Convert **outcome** from numeric to factor since we are going to build a classification tree in R and the outcome should be categorical.
- Check the standard deviation of each feature and remove the features with $sd = 0$.

Algorithm Introduction:
Adaboost is a good way to combine many other types of learning algorithms to improve their performance. The output of the other learning algorithms ('weak learners') is combined into a weighted sum that represents the final output of the boosted classifier. We use Classification Tree (*rpart* package) as the weak learners in our Adaboost algorithm. Classification tree can handle mix types of features in data and also has good tolerance of missing value.The main programming tool in this section is R.
Pseudo code:

```
for $i$ in $1 : 1000$ do
    Randomly sub-sample 50% data points from traindata.
    Build classification tree based on this subset and predict the test data.
    Record the prediction result as a weak hypothesis.
end
Then we can get a bunch of weak hypothesis which can be directly used in following Adaboost algorithm.
Initialization:
$D_k(i)$: Example i weight after learner k
N: number of observations.
$D_0(i) = 1/N(observations)$
$\alpha_k$: learner k's weight.
K: number of learners.
for $k$ in $1 : K$ do
    $\epsilon_k \leftarrow \sum_{i=1}^{N} D_{k-1}(i)\delta(h_k(x_i) \neq y_i)$
    $\alpha_k \leftarrow \frac{1}{2} \log \frac{1-\epsilon}{\epsilon}$
    $z_k \leftarrow \sum_{i=1}^{N} D_{k-1}(i)exp(-\alpha_k y_i h_k(i))$
    $D_k(i) \leftarrow \frac{D_{k-1}(i)exp(-\alpha_k y_i h_k(i))}{z_k}$
end
Final hypothesis: $H(x) = \sum_{k=1}^{K} \alpha_k h_k(x)$.
```

Final Score of Adaboost is: 0.915.
Submission:



295	↑67	Comendeiro	0.91618	7	Mon, 08 Jun 2015 23:34:13 (-0.2h)	
296	↑68	andb1	0.91606	3	Mon, 08 Jun 2015 23:49:22	
297	↑148	Jetty ‡	0.91574	16	Thu, 04 Jun 2015 00:55:52	
298	↓238	Kunal	0.91566	14	Mon, 01 Jun 2015 05:24:08 (-3.3d)	
-		**Yilin Zhao**	**0.91547**	-	**Fri, 04 Dec 2015 03:44:43**	**Post-Deadline**
299	↓18	Zayankovsky	0.91542	13	Fri, 05 Jun 2015 19:34:58	
300	↓25	Mamadou Diaby	0.91542	13	Wed, 06 May 2015 01:41:08 (-4.8h)	
301	↑58	Maxim Osipov ‡	0.91533	12	Mon, 08 Jun 2015 23:07:46 (-1.1h)	

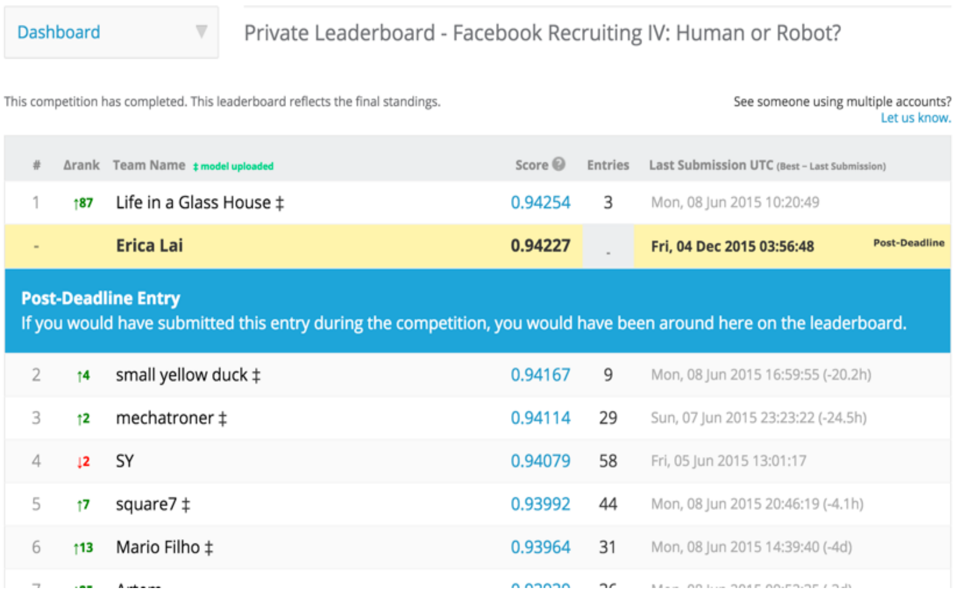
Post-Deadline Entry
If you would have submitted this entry during the competition, you would have been around here on the leaderboard.

Plot 3. Submission of Adaboost

Summary

Three algorithms, including Random Forest, Support Vector Machine and Adaboost are applied in this classification project and all of them performed an over 80% accuracy on the test data set. Random Forest returns the highest accuracy of 94.227% which outperforms most of the submissions.

Final Submission:



Dashboard ▼ Private Leaderboard - Facebook Recruiting IV: Human or Robot?

This competition has completed. This leaderboard reflects the final standings.

See someone using multiple accounts? Let us know.

#	Δrank	Team Name ‡ model uploaded	Score	Entries	Last Submission UTC (Best – Last Submission)
1	↑87	Life in a Glass House ‡	0.94254	3	Mon, 08 Jun 2015 10:20:49
-		Erica Lai	0.94227	.	Fri, 04 Dec 2015 03:56:48 Post-Deadline

Post-Deadline Entry
If you would have submitted this entry during the competition, you would have been around here on the leaderboard.

#	Δrank	Team Name	Score	Entries	Last Submission UTC
2	↑4	small yellow duck ‡	0.94167	9	Mon, 08 Jun 2015 16:59:55 (-20.2h)
3	↑2	mechatroner ‡	0.94114	29	Sun, 07 Jun 2015 23:23:22 (-24.5h)
4	↓2	SY	0.94079	58	Fri, 05 Jun 2015 13:01:17
5	↑7	square7 ‡	0.93992	44	Mon, 08 Jun 2015 20:46:19 (-4.1h)
6	↑13	Mario Filho ‡	0.93964	31	Mon, 08 Jun 2015 14:39:40 (-4d)

Plot 4. Final Submission

Final Score: 0.94227

Work Distribution

Algorithm Implementation	Team Member
Random Forests (Python)	Ka Ki Lai (Erica)
SVM (Matlab)	Chenzhi Wang
Adaboost (R)	Yilin Zhao

Table 2 Work Distribution