Predict Browser Family

Yue Wen, *Aspiring Data Scientist*, *Columbia University*

***Abstract*—This report describes the methodology to extract browser family and major version using Random Forest model. The model is trained on the train data set, and the obtained results along with the limitations are presented and discussed.**

I. Introduction

The overall goal of the task is to build a machine learning model to extract browser family from user agents str.

The main algorithm is developed using a weighted random forest model. The reason to choose the specific model will be discussed in section IV in more detail.

The structure of the final data product included three key steps. The first step is to extract key words from agent string. The second section is to build feature matrix using the established key words to predict browser family. The third step is to use match browser family with key words extracted in the first step to further search the version.

The accuracy rates to predict the browser family and the version rate are both higher than 99.5% in the provided training data.

II. Methodology

This section is to present the methodology to extract key words, build feature matrix and build prediction model.

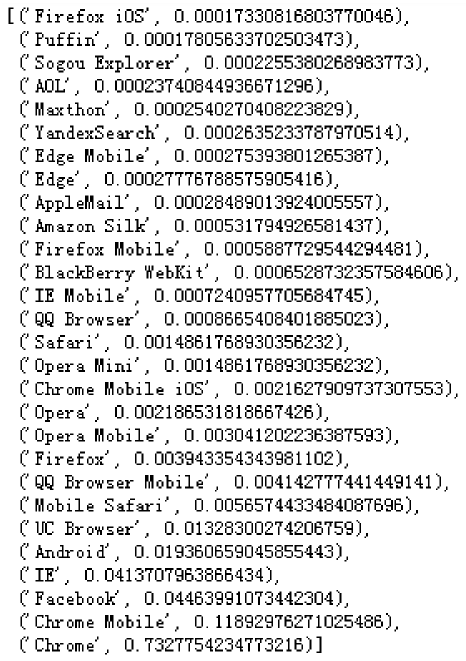
A. Data Set

The data sets contain two data sets: *data_coding_exercise.txt* and *test_data_coding_exercise.txt*. The model is trained using the entire train data set.

Both data sets contain three fields: user agents, user agent family and user agent's major version.

A simple observation of the dataset is that the data set is very imbalanced in terms of the number of records in each family. As shown in Fig.1, 'Chrome' makes up 73% of all the observations while "Firefox iOS" is less than 0.0002%. We will discuss the potential solutions further in more detail.

The training dataset is relatively very clean itself. Only 179 out 421215 observations have a "None" as the version. Those 179 observations are deleted for further work. However, it is important to note that we can delete the "bad" values here simply because it makes up such a small percentage of the dataset that we can simply remove them. In most other cases where the missing percentage is relatively larger, other imputation techniques might be considered and applied.



```
[('Firefox iOS', 0.00017330816803770046),
 ('Puffin', 0.00017805633702503473),
 ('Sogou Explorer', 0.00022553802689983773),
 ('AOL', 0.00023740844936671296),
 ('Maxthon', 0.00025402704082223829),
 ('YandexSearch', 0.00026352337870970514),
 ('Edge Mobile', 0.000275393801265387),
 ('Edge', 0.00027776788575905416),
 ('AppleMail', 0.00028489013924005557),
 ('Amazon Silk', 0.000531794926581437),
 ('Firefox Mobile', 0.0005887729544294481),
 ('BlackBerry WebKit', 0.0006528732357584606),
 ('IE Mobile', 0.0007240957705684745),
 ('QQ Browser', 0.0008665408401885023),
 ('Safari', 0.0014861768930356232),
 ('Opera Mini', 0.0014861768930356232),
 ('Chrome Mobile iOS', 0.0021627909737307553),
 ('Opera', 0.002186531818667426),
 ('Opera Mobile', 0.003041202236387593),
 ('Firefox', 0.003943354343981102),
 ('QQ Browser Mobile', 0.004142777441449141),
 ('Mobile Safari', 0.0056574433484087696),
 ('UC Browser', 0.01328300274206759),
 ('Android', 0.019360659045855443),
 ('IE', 0.0413707963866434),
 ('Facebook', 0.04463991073442304),
 ('Chrome Mobile', 0.11892976271025486),
 ('Chrome', 0.7327754234773216)]
```

Fig. 1. Percentage of the observations for each family in training dataset

B. Model

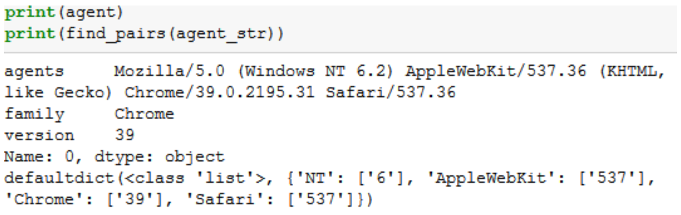
The established model contains three steps as described below.

1) Step1: Extract key words

This step is to identify all the key words from the "user agent" fields for further building feature matrix.

After some initial exploration of the data set, we found that the representative words for each family are always followed by the major version number. Therefore, for each user agent string, we can extract a list of (word, number) pairs from the tokenized string list.

Fig.2 gives a simple example:



```
print(agent)
print(find_pairs(agent_str))
```

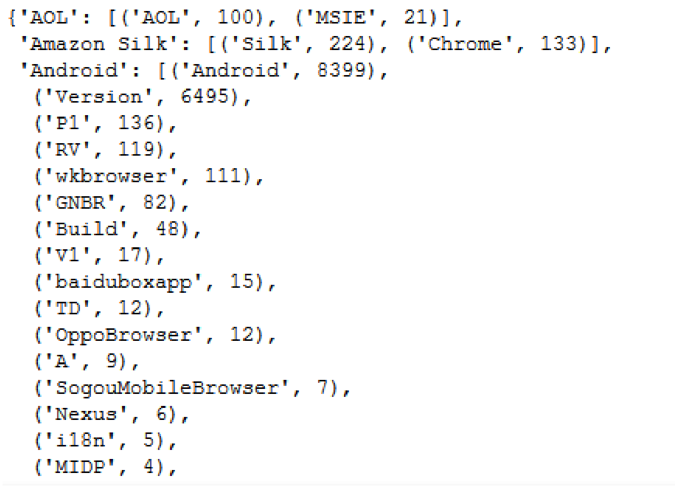
```
agents     Mozilla/5.0 (Windows NT 6.2) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/39.0.2195.31 Safari/537.36
family     Chrome
version    39
Name: 0, dtype: object
defaultdict(<class 'list'>, {'NT': ['6'], 'AppleWebKit': ['537'],
'Chrome': ['39'], 'Safari': ['537']})
```

Fig. 2. An example of finding (word, number) pair from agent string.

Then we noticed that 'Chrome' is the key word to be identified in this case, and it is followed by its version number '39'. Then for this specific agent string, we use '39' to locate the key word 'Chrome'.

Then, for each family, we performed the same procedure as discussed previously and count the occurrences for every key word.

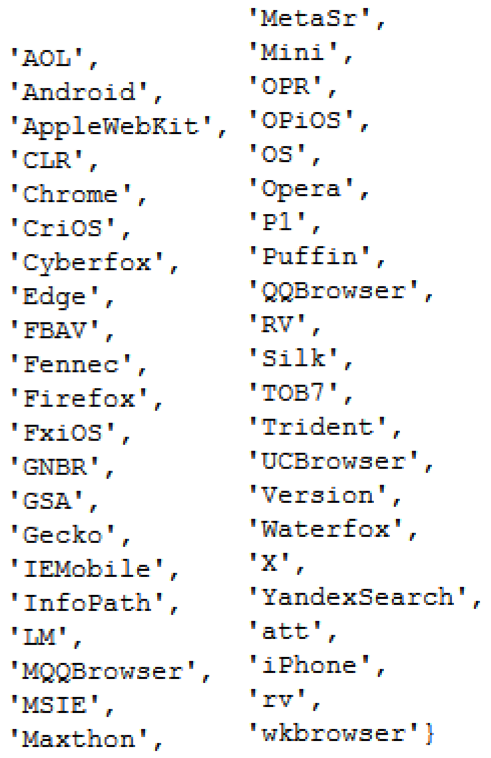
Fig.3 shows the extracted key words bag and corresponding occurrences for ‘AOL’, ‘Amazon Silk’, ‘Android’.



```
{'AOL': [('AOL', 100), ('MSIE', 21)],
 'Amazon Silk': [('Silk', 224), ('Chrome', 133)],
 'Android': [('Android', 8399),
  ('Version', 6495),
  ('P1', 136),
  ('RV', 119),
  ('wkbrowser', 111),
  ('GNBR', 82),
  ('Build', 48),
  ('V1', 17),
  ('baiduboxapp', 15),
  ('TD', 12),
  ('OppoBrowser', 12),
  ('A', 9),
  ('SogouMobileBrowser', 7),
  ('Nexus', 6),
  ('i18n', 5),
  ('MIDP', 4),
```

Fig. 3. An example of extracted key words and occurrences for each family.

After noticing that some words have very low occurrences, we can state that those words are just noisy words that are accidently chosen as they are followed accidently by the same version number. Therefore, we delete all the words that have very low occurrences. Then we finally get the 43 key words bag:



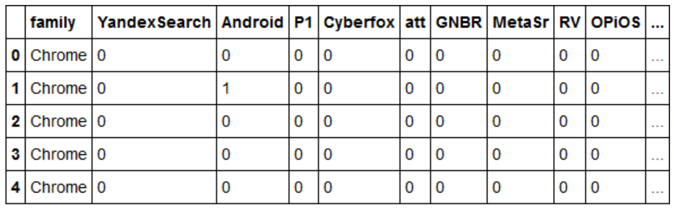
```
                   'MetaSr',
'AOL',             'Mini',
'Android',         'OPR',
'AppleWebKit',     'OPiOS',
'CLR',             'OS',
'Chrome',          'Opera',
'CriOS',           'P1',
'Cyberfox',        'Puffin',
'Edge',            'QQBrowser',
'FBAV',            'RV',
'Fennec',          'Silk',
'Firefox',         'TOB7',
'FxiOS',           'Trident',
'GNBR',            'UCBrowser',
'GSA',             'Version',
'Gecko',           'Waterfox',
'IEMobile',        'X',
'InfoPath',        'YandexSearch',
'LM',              'att',
'MQQBrowser',      'iPhone',
'MSIE',            'rv',
'Maxthon',         'wkbrowser'}
```

Fig. 4. All the extracted key words

So far, we successfully extracted all the key words. The next step is to use those key words to build feature matrix.

2) *Step2: Build feature matrix and label to predict browser version.*

In step 1, we have successfully extracted key words. This step is simply to create features to indicate the presence of key words in the string. Therefore, the feature matrix has 43 predictors, and each of them is a dummy variable, with ‘1’ representing the corresponding key word is contained in the agent string while “0’ representing the key word does not exist in the string,



	family	YandexSearch	Android	P1	Cyberfox	att	GNBR	MetaSr	RV	OPiOS	...
0	Chrome	0	0	0	0	0	0	0	0	0	...
1	Chrome	0	1	0	0	0	0	0	0	0	...
2	Chrome	0	0	0	0	0	0	0	0	0	...
3	Chrome	0	0	0	0	0	0	0	0	0	...
4	Chrome	0	0	0	0	0	0	0	0	0	...

Fig. 5. Head of the feature matrix

Then we transfer the 28 family string to integer range from 0 to 27.

It is important to note that it is safe and convenient to encode categorical variable like this because a tree model is applied further. However, if a regression model is applied, we should perform one hot encoding to get a better result.

Then we use a weighted Random forest model with tuned parameters to predict the label using the feature matrix.

3) *Step3: predict the major version*

Notice that in step 2), the browser family is predicted, so we can perform a simple match in this case to find the major version.

Firstly, we use the predicted browser family to search the key word in their corresponding family using the dictionary showed in Fig3 by the order from high occurrences to low occurrences. For instance, if an ‘Amazon Silk’ user agent string has both “Silk” and “Chrome”, then “Silk” will be picked up because it has a higher frequency.

Secondly, after we get the key word, i.e. match word for the specific string, we picked up the number following the key word as its major version.

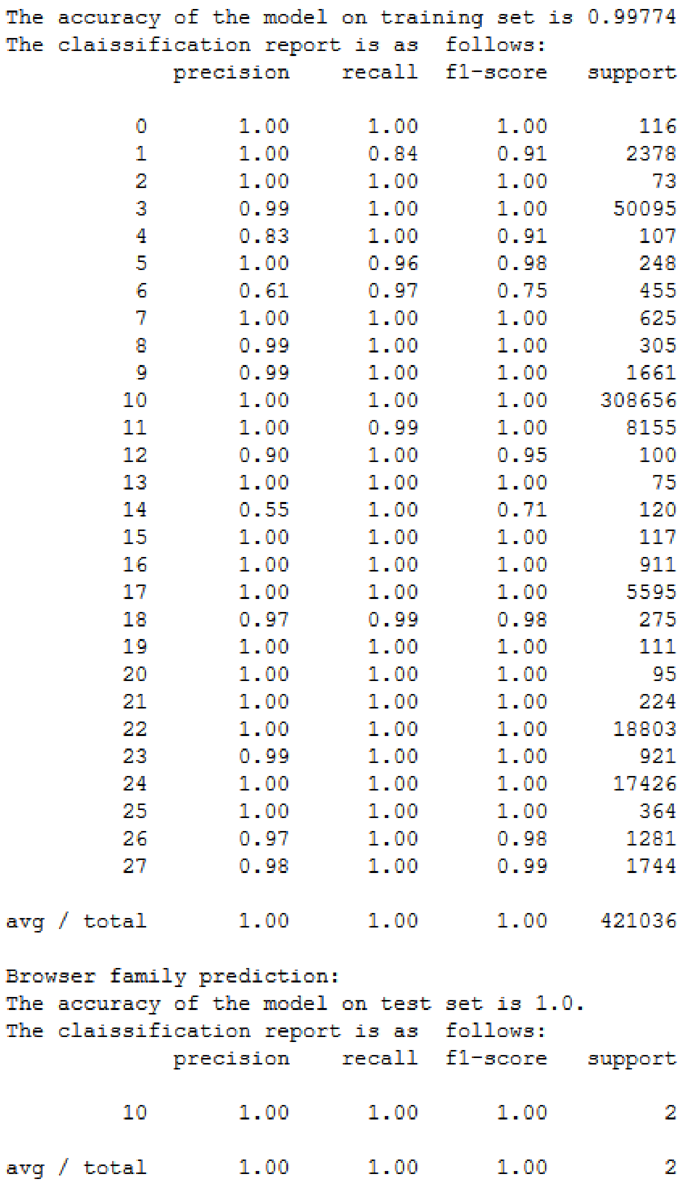
III. Result and Evaluation

In this section, we will discuss the classification results of the model.

A. *Predict the browser family*

Fig.5 showed the classification result of the developed random forest model.

We can tell the accuracy rate is very high. However, in the case where the dataset is very imbalanced, the accuracy rate is not the only metric to evaluate the classification model. Therefore, it is necessary to examine other metrics such precision, recall, F1-socre further.



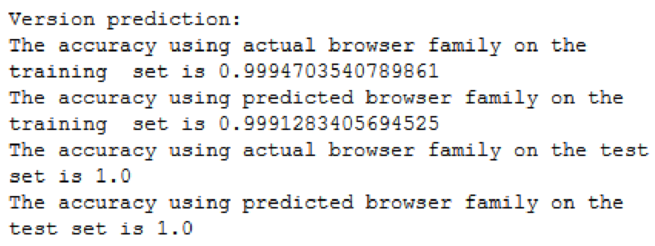
```
The accuracy of the model on training set is 0.99774
The claissification report is as  follows:
             precision    recall  f1-score   support

          0       1.00      1.00      1.00       116
          1       1.00      0.84      0.91      2378
          2       1.00      1.00      1.00        73
          3       0.99      1.00      1.00     50095
          4       0.83      1.00      0.91       107
          5       1.00      0.96      0.98       248
          6       0.61      0.97      0.75       455
          7       1.00      1.00      1.00       625
          8       0.99      1.00      1.00       305
          9       0.99      1.00      1.00      1661
         10       1.00      1.00      1.00    308656
         11       1.00      0.99      1.00      8155
         12       0.90      1.00      0.95       100
         13       1.00      1.00      1.00        75
         14       0.55      1.00      0.71       120
         15       1.00      1.00      1.00       117
         16       1.00      1.00      1.00       911
         17       1.00      1.00      1.00      5595
         18       0.97      0.99      0.98       275
         19       1.00      1.00      1.00       111
         20       1.00      1.00      1.00        95
         21       1.00      1.00      1.00       224
         22       1.00      1.00      1.00     18803
         23       0.99      1.00      1.00       921
         24       1.00      1.00      1.00     17426
         25       1.00      1.00      1.00       364
         26       0.97      1.00      0.98      1281
         27       0.98      1.00      0.99      1744

avg / total       1.00      1.00      1.00    421036

Browser family prediction:
The accuracy of the model on test set is 1.0.
The claissification report is as  follows:
             precision    recall  f1-score   support

         10       1.00      1.00      1.00         2

avg / total       1.00      1.00      1.00         2
```

Fig. 6. Classification result of browser family

When we examined the classification report, we can see it performed very well for almost every family.

B. *Predict the major version*

The step is more like a match/search model, we report the accuracy as below:



```
Version prediction:
The accuracy using actual browser family on the
training  set is 0.9994703540789861
The accuracy using predicted browser family on the
training  set is 0.9991283405694525
The accuracy using actual browser family on the test
set is 1.0
The accuracy using predicted browser family on the
test set is 1.0
```

Fig. 7. Classification result of major version

We can tell the accuracy rate is very high as well. Notice here, we give two version of accuracy rate, one is to use the ground truth browser family to predict major version, another is to use the predicted browser family to predict major version. We can tell the model performed well for both cases.

IV. Discussion

A. *Encode categorical label*

We encoded the categorical label to integer. One alternative way is to do one hot encoding, build 28 models and then use 'one versus all' multiclass classification techniques. However, it will be very computationally expensive. Therefore, transferring the categorical label to integer might be the most practical way in this case.

B. *Why random forest model*

As we discussed above, we encoded the categorical label to integer, so tree methods might be the first choice. In this case, logistic regression will assume the order relation of the encoded categorical label while tree models do not make such assumptions.

Besides, we chose random forest model instead of a gradient boosting model because the developed features are very strong predictors, which makes the classifier itself a strong learner.

C. *Imbalanced data set*

As shown in Fig.1, the dataset is very imbalanced. We came up with the following potential solutions.

1) *Weighted random forest model*

A weighted random forest model is used here to address the problem. When predicting the result, the model will adjust weights inversely proportional to class frequencies in the input data.

In general, for any tree models, we can apply the same practice in the case where the dataset is imbalanced.

2) *Downsample and Oversample*

We can also downsample the majority class or oversample the minority class to make the dataset more balanced. Especially in cases where the cost of false positive error and false negative error is different, it is more important to transform data before we start to build the model.

In this case, we can down sample 'Chrome' and 'Chrome Mobile' as well as oversample the rest of the browser families to make the size of each class similar.

D. *Tradeoff between computation expenses and model performance*

It takes around one minute to run each step discussed previously, adding up to around 3 minutes in total.

For the modeling part, training the random forest model is relatively computationally expensive than some naïve model such as Naïve Bayes.

At that point, whether to select a simpler model or a complicated model with higher accuracy really depends more on the business case.

Sometimes, it is more beneficial to have a simpler and faster model. For example, if a real-time prediction result for user application is needed, we might prefer a simpler model with a faster speed, even the result might be worse than a complicated model.

On the other hand, if the business focuses more on the prediction results. For example, the prediction results will be used further for fraud detection, then the accuracy rate might be the most important metric for us to consider.

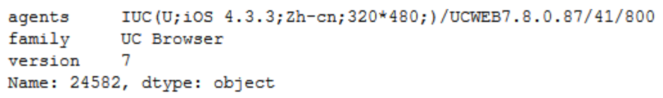
E. Make new prediction and update new model Analysis

After we extract the key words, the key words bag can be further used to build new features to predict the outcome. It is not necessary to re-extract key words every time when we want to make a new prediction. However, it might be beneficial for us to update the key words from time to time as more data become available.

V. LIMITATION AND POTENTIAL SOLUTIONS

A. Extract key words techniques

We extract key words from the tokenized string with any punctuation and white space as separator. In this case, we failed to extract key words if there is no white space/punctuation between the key word and the major version. An example is shown below.



```
agents     IUC(U;iOS 4.3.3;Zh-cn;320*480;)/UCWEB7.8.0.87/41/800
family     UC Browser
version    7
Name: 24582, dtype: object
```

Fig. 8. An example of failure to extract key word.

However, only 7 observations follow such pattern, we might consider this as an error in data.

On the other hand, assuming that a considerable proportion of data follow such pattern, we can use 'regular expression' to search words ending with a numerical number for further processing.

B. Dependence between version number and browser family

The extracting key words technique utilizes both version number and browser family. However, we might run into cases where the ground truth of version number or browser family is unknown.

In this case, we come up with a solution as follows.

If we only know the browser family without knowing version number, we can extract all the words and then simply select those with high frequency as key words.

VI. CONCLUSION

The report demonstrates the methodology to classify browser family and infer major version from user agents. From the business side, the developed model might have further potential implications and applications in the cyber security industry, such as network monitoring and suspicious users detection.
Meanwhile, it is a very interesting and challenging mini-project to work on!

Yue Wen received the B.S degree from Tsinghua University, China, and currently a master candidate in the department of industrial engineering & operations research at Columbia University. Her interests mainly focus on data science, including machine learning and statistical modeling. She has worked as an energy data scientist intern at Vermont Energy Investment Corporation during 2017 summer.