Data Science and Machine Learning (IM5033)

Dealing with Boy & Girl Classification on Kaggle

呂晟維 [a]， 華崧淇 [b]

[a] 國立中央大學資訊管理學系 106403551@cc.ncu.edu.tw

[b] 國立中央大學資訊管理學系 schua1013@g.ncu.edu.tw

Table Content and Teamwork

Data explore & Design	Visualize	V
	Determine our topic - ensemble	V
Data Pre-processing	Scale outliers	
	Deal with imbalanced	
	Encode nominal category columns	V
	Encode self-intro	V
	Add BMI column	V
	Bin numeric columns	V
	Build up our cleaning module	V
Modeling	Random forest	V
	SVM	
	Adaboost	V
	Hard voting	V

(checked is my engagement)

Design of Work

In this competition, we aim to accomplish 2 main targets:

- Engage in automaton of data cleaning
- Focus on ensemble learning

Implementation of Data Pre-Processing

In this phrase, we explore the data and conduct data Pre-Processing with following methods. In order to minimize copy paste and maximize code reuse, we aim to modulize our data Pre-Processing scripts in a separate .py file and utilize the module by importing in .ipynb notebook.

Deal with imbalanced

The first thing to do is explore raw data, the gender class has a distribution of 187:57 on boy to girl. Since we apply SMOTE + tomeklink to make them equal.
Note: SMOTE+ENN or random over sampling are recommended either.

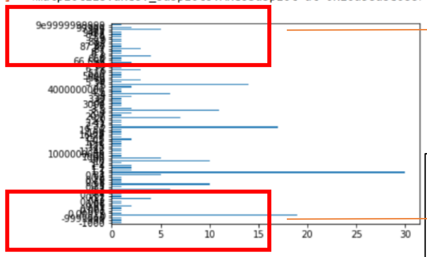
Scale outliers

We can find there're several missing value, big-num value and also scientific notation in numeric columns (height, weight, IQ…). So let's deal with outliers first. We have 2 options for scaling:

- Replace the outliers with specific min/max value. (according to real world)
- Scale whole data. (ex: minmax scaler, std scaler, etc.)

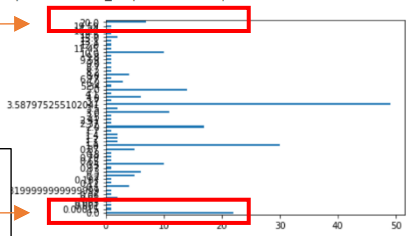
We choice the first option, but apparently its must more human demanded. In practical, me have to declare the min/max value for each numeric column. For instance, we set "yt" range from 200 ~ 0, edit the records > 200 and < 0 to 200 and 0.

```
In [30]: # 原本的yt
         origin_df['yt'].value_counts().sort_index().plot(kind='barh')
Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x16a58abe688>
```



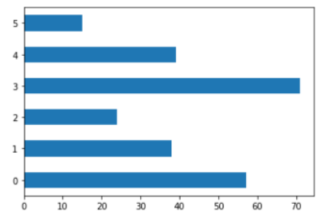
Origin Distribution (full of outliers)

```
In [32]: # scale outlier後的yt
         scaleOutliers(origin_df)['yt'].value_counts().sort_index().plot(kind='barh')
Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x16a58ca3408>
```



Scale Outliers Distribution

```
In [28]: df['yt'].value_counts().sort_index().plot(kind='barh')
Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x16a58909888>
```



After Binning

Encode nominal category columns

“Star sign” and “Phone os” these two categorical columns are consisted with strings. Convert them into label 1~12 and 1~4 using df.replace().

Encode self-intro

Self intro column contains meaningful text value, since traditional machine learning algorithms cannot understand raw text inputs, the only thing to do is discard it or encode it. Under time limitation, it’s hard to apply NLP or word2vector. But we find a significant rule: if self intro contains “Handsome” the record must be a boy, however, if self intro contains “Beautiful” there’s only 11/11 chances this he or she is a girl. As a result, why not transform “Handsome” into 1, the others into 0 ?

Add BMI column

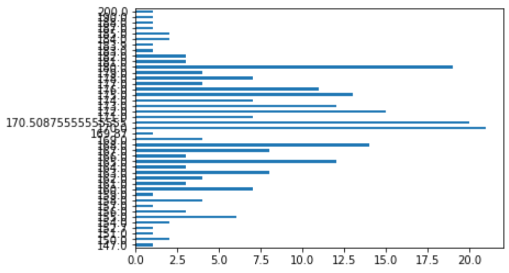
Boy and girl ranges from different BMI, adding a new column to emphasize column “weght” and “height”.

Bin numeric columns (a.k.a. bucketing)

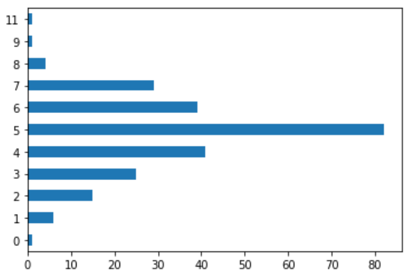
For numeric columns, we have already scaled them by min and max values. That’s not enough since we only squeeze the full range of data and have not dealt with the distributions. To pursue a well cleaned dataset, we bin each numeric (continuous) column into 7~12 bins. Binning gives us less information but facilitates the analysis and reduces noise, especially in “YT”, “FB friends” those columns with big offsets.

Note: pd.cut or pd.qcut are different methods and provide different meaning.

Note: If you have applied scaler through the whole data, it’s less required for binning. (However we took the harder way ☹)



before Binning



After Binning

Build up our cleaning module

Finally, the last step of data cleaning, wrap each step into methods and put them in a python module, ready for imported whenever necessary. It can process both training and testing data. Note: “over/under sampling” is only performed at training, be careful.

Implementation of Modeling – Voting Classification

The ensemble voting classification approach contains all the models that we’ve implemented. We will introduce the sub-classifier it contains and show their accuracy as well.

Hard vs Soft voting

Voting method has “Hard vote” and “Soft vote” modes, “Hard vote” counts the ballots to determine result, “Soft vote” calculate the average of probabilities (can also customize weights). To simplify, we choose the hard vote method.

Classifiers

We have 5 classifiers with different algorithms, which are:

- SVM (kernel='rbf’)
- Logistic Regression (multi_class='ovr' for binary classification)
- Random Forest Classifier (n_estimators=50)
- Naive bayes
- AdaBoost Classifier (n_estimators=100)

The resulting accuracy of each algorithms, naïve bayes performs very badly…:

- Accuracy: 0.86 (+/- 0.02) [SVM]
- Accuracy: 0.84 (+/- 0.01) [Logistic Regression]
- Accuracy: 0.91 (+/- 0.02) [Random Forest]
- Accuracy: 0.55 (+/- 0.02) [naive Bayes]
- Accuracy: 0.88 (+/- 0.02) [AdaBoost]
- Accuracy: 0.88 (+/- 0.02) [Ensemble]

For the case of research, we test each model separately and find that random forest has the highest training and validation accuracy. However, voting classifier does not have significant improvement.

Note: We apply a 80%-20% train-validation split during training the model, test set is the no-solution csv file, so don’t worried about whether to make a train-validation-test split.

Further Improvements

Due to time limitation, we spend less attention on figuring the confusion matrix and ROC curve. If you have time, please look into the recalls and list the common False Negative cases (true condition is girl but mis-classify as boy, since girl is the minority class). You could easily find out the mistakes made by classifier and come up with countermeasures.