Popularity Prediction for Articles Online

110550014 吳權祐
110550060 關立
110550082 王邦碩
110550130 劉秉驊

Introduction

Introduction

Social media nowadays has penetrated teenagers' life such as Dcard that more and more people are involved in discussing daily popular articles.

Therefore, we are interested in what the main themes the crowd delves into. And according to this, with AI analysizing, we can predict if an article will go viral.

In the future, we may utilize the outcome to make bloggers or writers rapidly understand the trend of this society, which accelerates the field of literature.

Related work

Related work

There are various models developed out of different proposes in AI field, which we are looking forward to a decent model meeting our main target and analyze the difference.

Random forest

Xgboost

Related work

We found out an expert predict the popularity with NY Times, using different models and algorithms . Thus, we refer to the API he uses and the mode of finding features.

Compared to thing that he emphasizes is the presentations of different models, we are more interested in seeing the feature engineering among different version data and the component of dataset as additional variables.

Thus, we can discuss further and find out the best outcome.

NYT-Article-Popularity

Dataset / Platform

Dataset / Platform

Data Source :

The New York Times.

Acquiring way :

Using web crawler in nytimes.py to fetch 2-year (2021-2022) articles though its API.

About 100,000 articles in total.

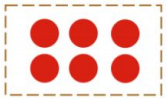
Randomly divide articles into ⅔ as train data and ⅓ as test data

Baseline

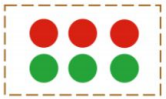
Entropy = $-\sum p_j \log_2 p_j$

Information Gain = $-p * \log_2 p - q * \log_2 q$

p：是的機率　q：否的機率



Info(6, 0) = $-\frac{6}{6}\log_2(\frac{6}{6}) - \frac{0}{6}\log_2(\frac{0}{6}) = 0$

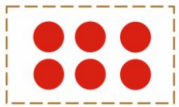


Info(3, 3) = $-\frac{3}{6}\log_2(\frac{3}{6}) - \frac{3}{6}\log_2(\frac{3}{6}) = 1$

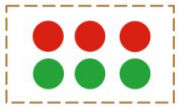
Gini = $1-\sum p_j^2$

Gini Impurity = $1 - (p^2+q^2)$

p：是的機率　q：否的機率



Info(6, 0) = $1-(1^2+0^2) = 0$



Info(3, 3) = = $1-(0.5^2+0.5^2) = 0.5$

- Mean Square Error

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2$$

- Mean Absolute Error

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right|$$

Decision Tree :

Classification Tree:

The main idea is calculating imgurity or information gain as the prediction for how we split into 2 choises.

Regression Tree :

Different from classification, we have to control the error to approach zero by formula MSE or MAE, while heading to the maximum depth.

Usually use in continuous data.

Baseline

Method :

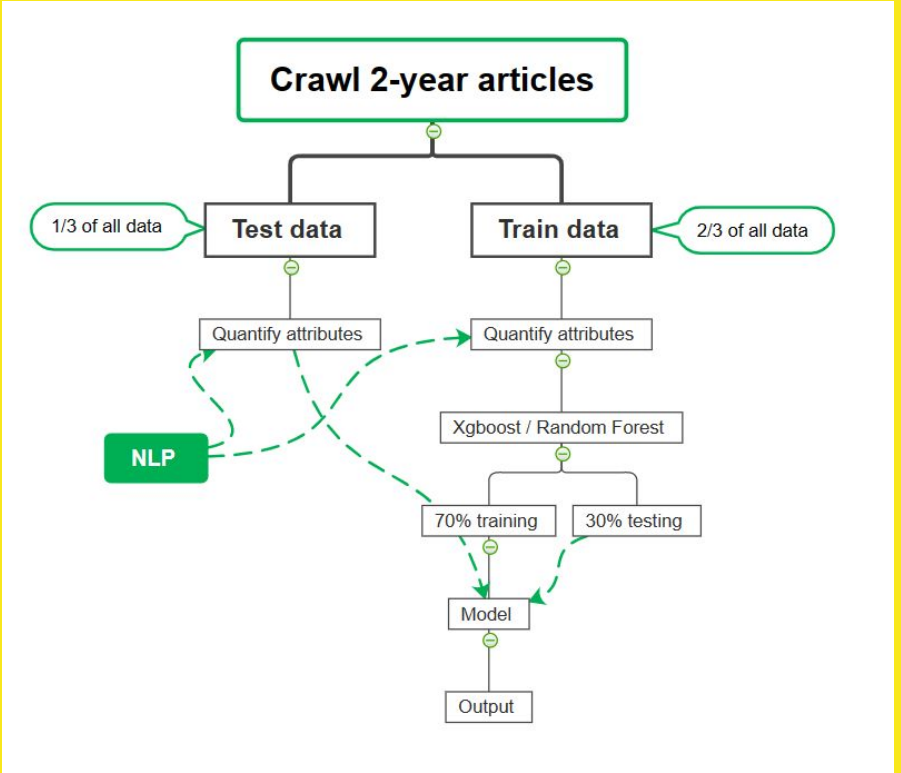
Decision Tree has been mother model of Xgboost and Random Forest, which are our approach to compare with.

Input : The features
Output : The Tree model to get predict probability

```
part = 'dt'
data_list[part] = {}
train = pd.read_pickle('./feature/feat_v1.pkl')
X = train.drop(columns=['is_popular', 'group_by_subsection_name', 'feature_subsection_name'])
y = train['is_popular']

data_list[part]['X_train'], data_list[part]['X_test'], data_list[part]['y_train'], data_list[part]['y_test'] = \
    train_test_split(X, y, test_size=0.3, random_state=42)

dt = run_model(part, 'dt')
dt_param = {'dt__min_samples_leaf' : [0.01, 0.1, 1, 2, 3],
            'dt__max_leaf_nodes' : [10, 30, 50, 70, 100, 200]
            }

dt_gs = run_model(part, 'dt', mod_params=dt_param, grid_search=True)
data_list[part]['test'] = pd.read_pickle('./feature/feat_test_v1.pkl')
data_list[part]['test'] = data_list[part]['test']._get_numeric_data()

data_list[part]['final_actual'] = data_list[part]['test']['is_popular']
data_list[part]['test'] = data_list[part]['test'].drop(columns=['is_popular', 'feature_subsection_name', 'group_by_subsection_name'])

evaluate_model(part, dt_gs)
dt_gs.fit(X, y)
print(' ')
evaluate_model(part, dt_gs)
```

Main Approach

Main Approach



Main Approach

Features choosing :

5 ways to quantify.

Popularity :

The number of comments that satisfies the threshold we define.

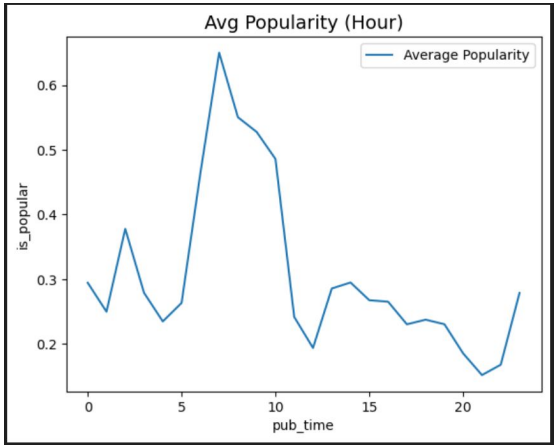
Analysis :

Calculate the correlation between popularity and the features in every unit as the representation of all the articles in the same unit.

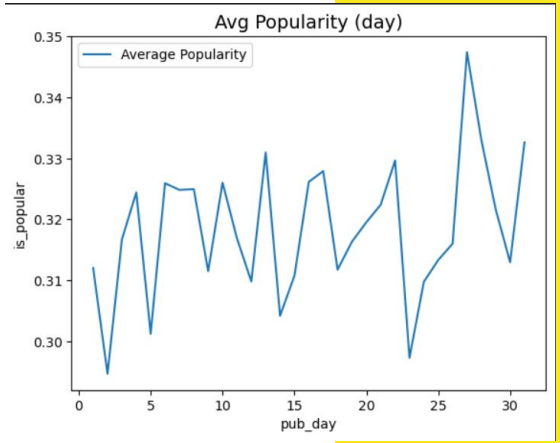
1. Choose the distinguishing time section of popularity

We are interested in whether time section affects people read articles. There may be a peak or a super valley that leads to positive or even negative effect.

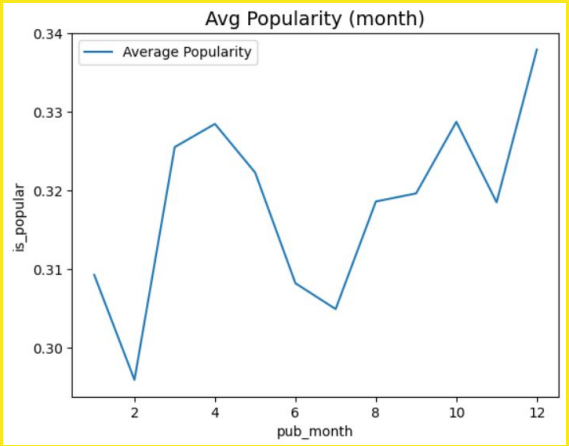
```
sns.lineplot(data = train.groupby('pub_time').mean()['is_popular'],  label='Average Popularity')
plt.title('Avg Popularity (Hour)', fontsize=14)
#plt.axvline(x = 3, linestyle= '--')

train['is_noon'] = train['pub_time'].apply(lambda x : 1 if (6 <= x and x <= 10) else 0)
```



We emphasize the best from 6 ~ 10.



There is no distinguishing section.



We emphasize the worst before February.

Main Approach

Also use the material in article

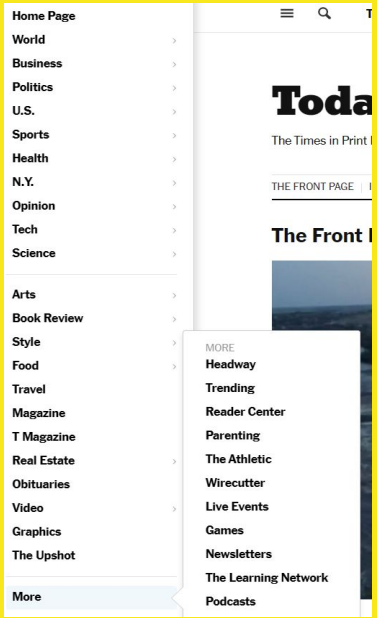
```
{'Op-Ed': 0.5198487712665406,
'News': 0.3318475317642158,
'Quote': 0.0,
'Correction': 0.0,
'briefing': 0.16906130268199235,
'Interactive Feature': 0.20671140939597316,
'Obituary (Obit)': 0.16155291170945524,
'Letter': 0.0022701475595913734,
'Editorial': 0.5085714285714286,
'Review': 0.25048923679060664,
'': 0.0,
'List': 0.013888888888888888,
'News Analysis': 0.5662650602409639,
"Editors' Note": 0.0,
'An Appraisal': 0.0,
'Biography': 0.0}
```

```
{'Opinion': 0.46581280788177337,
'U.S.': 0.21664034741413343,
'World': 0.19875933662488923,
'Today's Paper': 0.002232142857142857,
'Corrections': 0.0,
'Briefing': 0.01037117903930131,
'Style': 0.2549295774647887,
'Sports': 0.22435897435897437,
'Business Day': 0.22910835665733706,
'New York': 0.4203378602186154,
'Well': 0.7889784946236559,
'Movies': 0.1352973267866885,
'The Upshot': 0.6746575342465754,
'Climate': 0.42242703533026116,
'Science': 0.37409200968523004,
```

2. Group by section names from NYTimes

As common sense, personal hobbies induces people to read what we are attracted by.

Thus, it's important to pick up these sections.



Main Approach

3. Choose the famous keywords in 2021 and 2022

Event (Ukrainian_Russian_war, covid, Storming of the US Capitol)

```
train['is_ukrainian_russian_war'] = train['keywords'].apply(lambda x: 1 if ('War and Armed Conflicts' in x or 'Ukraine' in x ) else 0)
train['is_covid'] = train['keywords'].apply(lambda x: 1 if 'Coronavirus (2019-nCoV)' in x else 0)
train['is_storm'] = train['keywords'].apply(lambda x: 1 if 'Storming of the US Capitol (Jan, 2021)' in x else 0)
print('ukrainian_russian_war     ', train.corr()['is_popular']['is_ukrainian_russian_war'])
print('Coronavirus               ', train.corr()['is_popular']['is_covid'])
print('Storming of the US Capitol', train.corr()['is_popular']['is_storm'])
```

```
ukrainian_russian_war      -0.01711591378491107
Coronavirus                -0.008319868561523065
Storming of the US Capitol 0.018062328389029523
```

Politics(party, congress, president)

```
train['is_party'] = train['keywords'].apply(lambda x: 1 if ('Democratic Party' in x or 'Republican Party' in x ) else 0)
train['is_congress'] = train['keywords'].apply(lambda x: 1 if ('House of Representatives' in x or 'Senate' in x) else 0)
train['is_president'] = train['keywords'].apply(lambda x: 1 if ('Trump, Donald J' in x or 'Biden, Joseph R Jr' in x)else 0)
print(train.corr()['is_popular']['is_party'])
print(train.corr()['is_popular']['is_congress'])
print(train.corr()['is_popular']['is_president'])
```

```
0.0747245023783232
0.03126890311545747
0.05524818799901686
```

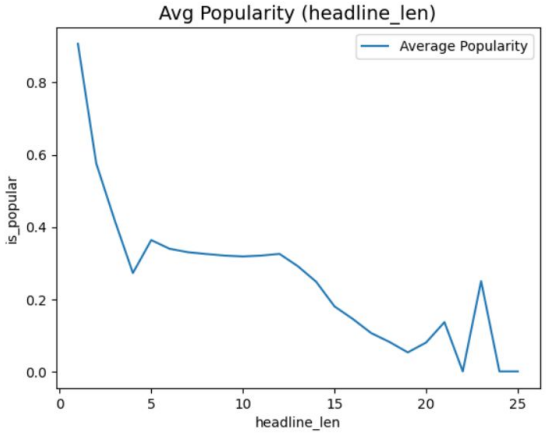
In 2 very years, various headlines springed out along the policy from the BIG countries like war between Ukraine and Russia.

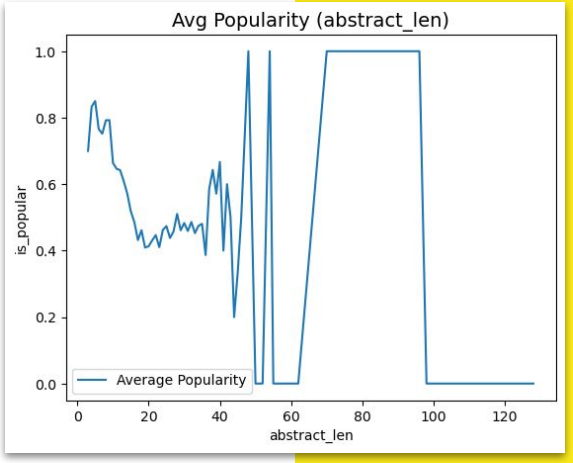
But we think these main themes are so dominant that we unite them as one group.

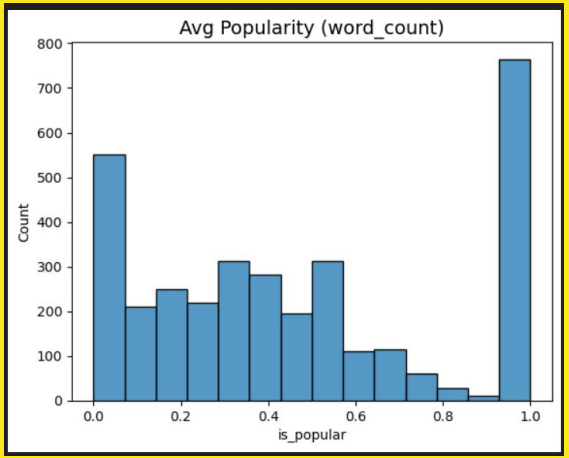
Main Approach

4. Analysize the length of different features

Sometimes, we need a neat information to let us comprehend what's going on, which the number of words may play a big role in our reading willing.







Lots of features found by NLP often appears in top 40 rank of correlation.

5. Utilize NLP to quantify sentiment and abstract

NLP is well-known to handle human words in linguistics. Thus, we are going to use Textblob, Vader, and SnowNLP to distinguish some main aspects :

- Subjective / Objective
- Polarity :

  Words containing extreme factors and sentiments.

textblob

```
text = "i am happy today."
blob = TextBlob(text)
# get the sentiment of the text
sentiment = blob.sentiment
print(sentiment)
```

```
Sentiment(polarity=0.8, subjectivity=1.0)
```

Main Approach

Data name	Uneliminated	Eliminated
Modified	Modified_v1	Modified_v2
Unmodified	V1	V2

Modified_v1_0.3 :
Raise the threshold from median to top 30%

Construct 5 kinds of data via 3 additional ways

1. Modified :

   Eliminate articles of zero comment.

2. The elimination of high correlation :

   We drop the pair of attributes that have too high correlation score.

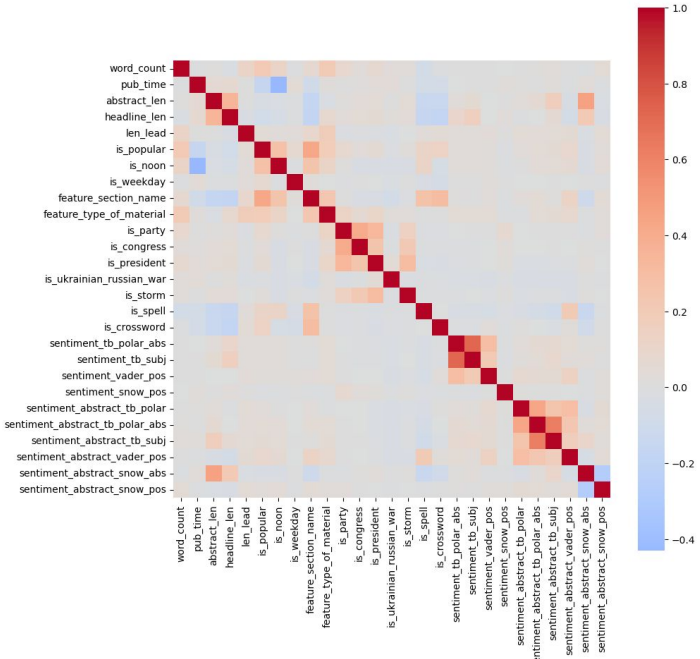
   (explain in the next slide)

3. Raise the threshold of popularity :

   We define the top 30% of the most comments to be popular instead of be higher than the median.

Main Approach

After dropping

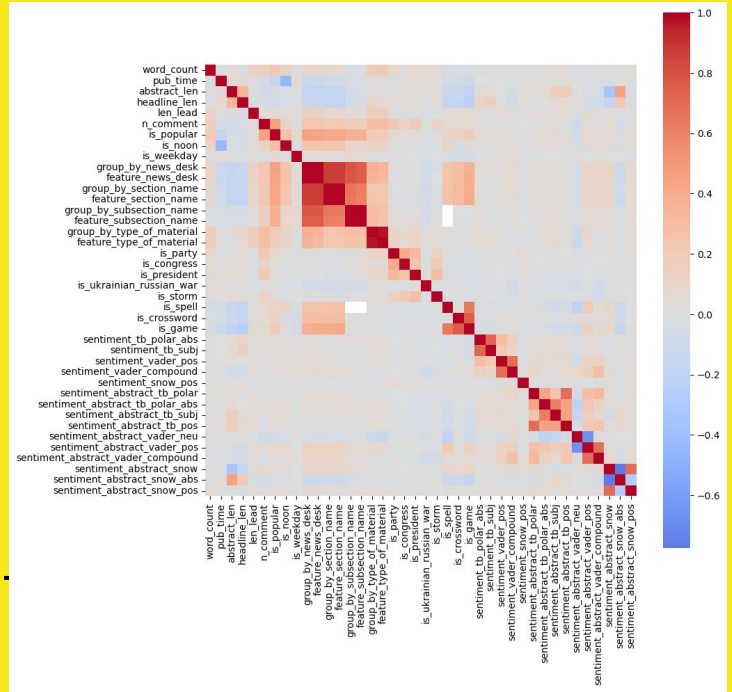


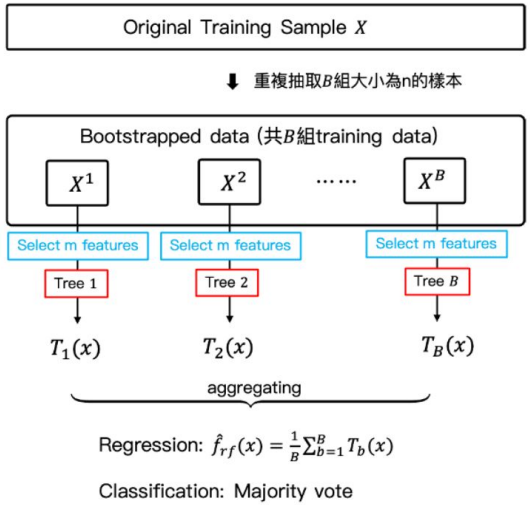
Independency of features :

Based on our analysis, we find many features are related, which these pairs may overaffect the results while models are learning.

Thus, we drop some pairs in order to make the results more independent.

Original





Main Idea : Draw independent trees

$$s^2 = \frac{1}{B}\sum_{b=1}^{B}(\hat{\theta}_j^* - \bar{\theta})^2 = \frac{1}{B}\sum_{b=1}^{B}(\hat{\theta}_j^*)^2 - (\frac{1}{B}\sum_{b=1}^{B}\hat{\theta}_j^*)^2$$

s^2: sample variance of **theta**
theta : Bootstrap replicator
B : the number of dataset

Random Forest :

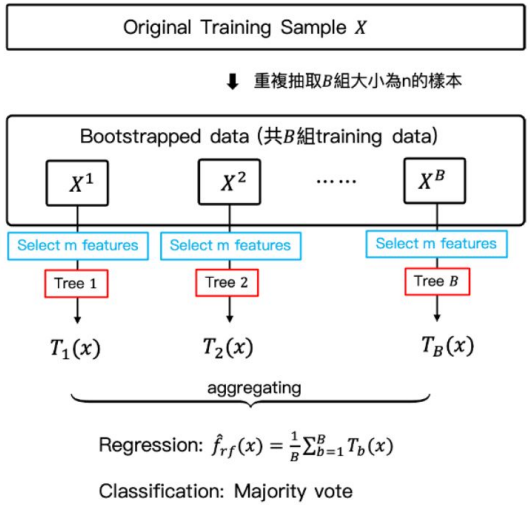
Main Idea :

1. Bootstrap keeps drawing samples.
2. Bagging constructs independent trees.
3. Process based on Decision Tree.

Input : Our Features

Bootstrap :

1. Draw and put back several sets of samples to estimate variance among dataset.
2. To decrease variance and raise accuracy of models.



Main Idea : Draw independent trees

$$\hat{f}_{avg}(x*) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{b}(x^*)$$

Regression :
f(x) : model result

Random Forest :

Process :

1. Draw data from Bootstrap samples.
2. Construct random forest trees for each drawn data.
3. Select few features in single node and run the process of decision tree.

Output : The forest model

1. Regression : Average all the results.
2. Classification : Majority vote

$$\hat{C}^{B} = majority\ vote\{\hat{C}_b(x)\}_1^B$$

—**Classification** :
$\hat{C}_b$: the class prediction of the bth random forest tree.

Main Approach

For each version data :

Let models learn the best parameters on train data within an appropriate range.

For 5 versions of data, we choose 5 best combinations respectively to run on their test data.

Random Forest :

```
rf_params = {'rf__n_estimators': [100, 200, 300, 400, 500],
             'rf__max_depth': [10, 20, 30, 40, 50, 60],
             'rf__min_samples_leaf': [i for i in range(2, 18)],
            }
```

n_estimators : the number of trees

max_depth : the maximum number of branches

min_samples_leaf : the minimum number of samples in all nodes

$$\hat{y} = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i)$$

$$where\ F = \{f(x) = w_{q(x)}\}(q : R^m \to T, w \in R^T)$$

Main function :

Estimation **y** that sums up every tree **f(x)** with scores **q**

Algorithm 1: Exact Greedy Algorithm for Split Finding

Input: I, instance set of current node
Input: d, feature dimension
$gain \leftarrow 0$
$G \leftarrow \sum_{i \in I} g_i$, $H \leftarrow \sum_{i \in I} h_i$
for $k = 1$ *to* m **do**
  $G_L \leftarrow 0$, $H_L \leftarrow 0$
  for j *in sorted*(I, *by* $\mathbf{x}_{jk}$) **do**
    $G_L \leftarrow G_L + g_j$, $H_L \leftarrow H_L + h_j$
    $G_R \leftarrow G - G_L$, $H_R \leftarrow H - H_L$
    $score \leftarrow \max(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$
  end
end
Output: Split with max score

Greedy algorithm to split leaves

Xgboost :

Main Idea : Boosting

1. Generate dependent trees one by one.
2. Let the latter tree to support the former.
3. Sum up the result of the estimation.

Input / Process : Our Features

Gradient Boosting Decision Tree

1. Searching all the features in single sample node.
2. Greedy : Select the highest Gain value to branch that includes left, right leaf scores, or no leaf score and the complexity of splitting a leaf.

$$Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right] - \gamma$$

Gain function :

First term is left leaf score, **second** is the right.
Third is no leaf score, **last** is complexity of splitting.

$$l(y_i, y^i) = \frac{1}{2}(y_i - y^i)^2$$

Loss function :
Calculate MSE between actual value and estimation

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda ||w||^2$$

Gamma, lambda : coefficients
T : the number of the nodes
w : the score of the node

$$Obj^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \Omega(f_i)$$
$$= \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) + constant$$

Goal: find f_t to minimize this

Object function :
Sample in a node maps to an **Object function**, which we want to modify.

Xgboost :

Output : The forest model

1. After training a tree, approximate the result to the actual value by Loss value, and also estimate the complexity of the very tree.
2. Then construct another tree according to the previous one, which is used to modify the error and approach the actual.

Main idea of construct tree

- Start from constant prediction, add a new function each time

$$\hat{y}_i^{(0)} = 0$$
$$\hat{y}_i^{(1)} = f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i)$$
$$\hat{y}_i^{(2)} = f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i)$$
$$\dots$$
$$\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)$$

New function

Model at training round t

Keep functions added in previous round

Main Approach

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda||w||^2$$

$$\text{obj}(\theta) = \sum_{i}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

$$Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right] - \gamma$$

For each version data :

Let models learn the best parameters on train data within an appropriate range.

For 5 versions of data, we choose 5 best combinations respectively to run on their test data.

Xgboost :

```
xgb_params = {
              'xgb__learning_rate' : [0.1, 0.2, 0.3],
              'xgb__max_depth' : [3, 5],
              'xgb__n_estimators' : [100, 200, 300],
              'xgb__reg_alpha' : [0.1, 1.0, 1.8, 2.0],
              'xgb__reg_lambda' : [0.1, 1.0, 2.0, 2.4, 3.0],
              'xgb__gamma' : [1.0, 2.0, 2.5, 3.0],
            }
```

- learning_rate : the weight of trees.
- gamma : the cost of splitting branches that we hope to suppress.
- We decrease maximum depth and n_estimators to lighten the pressure of computer.

Evaluation Metric

Evaluation Metric

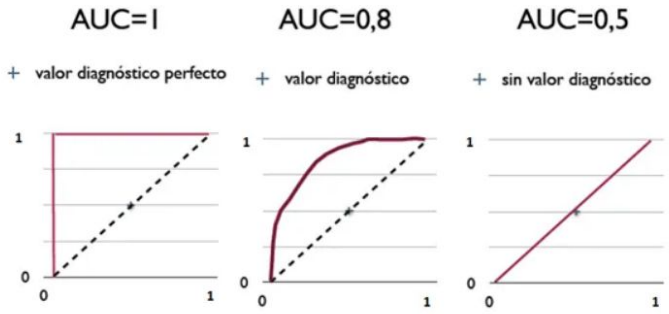
Term	Positive	Negative
True	TP	TN
False	FP	FN

Positive : Models predict it is popular.
Negative : Models predict it is NOT popular.
True : Actual situation suits what models predict.
False : Actual situation DOESN'T suit what models predict.

True Positive Rate (TPR): TP / (TP + FN)
False Positive Rate (FPR): FP / (TN + FP)

Accuracy : (TP + TN) / (TP + FP + FN + TN)
Precision : TP / (TP + FP)
Recall : TP / (TP + FN)
F-Score : The average score of Precision and Recall

Evaluation Metric



ROC - AUC : Area ranged by TPR as x-axis and FPR as y-axis evaluates the ability of the models.

For each threshold in the same model, plotting the (FPR, TPR) coordinates in the ROC space, we obtain the ROC curve specific to that model.

The left-top point of the ROC space has the highest TPR and the lowest FPR, so the curve that is closer to the left-top corner has the better classification ability.

This tells that the AUC of the ROC curve could be a score to measure the performance of models.

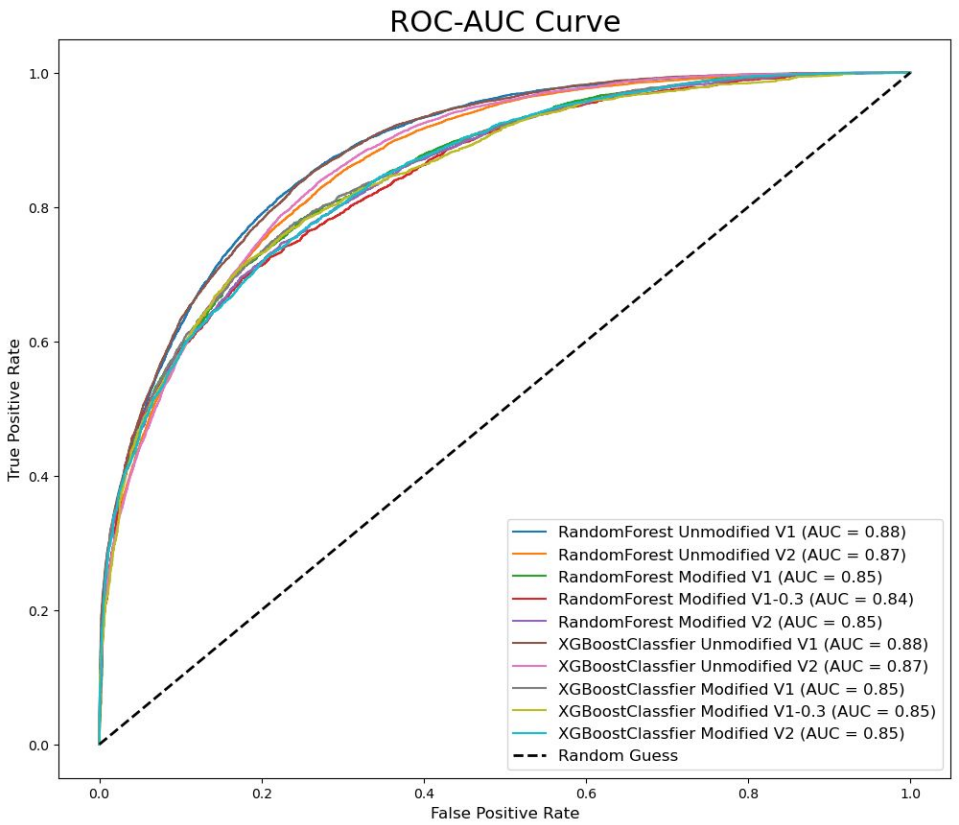
Results & Analysis

Results & Analysis

Hypothesis:

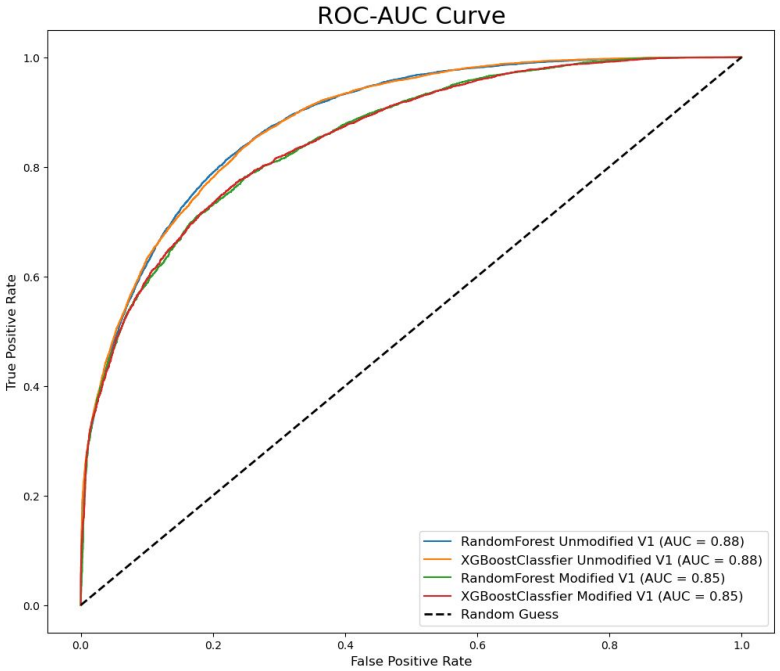
Given the abundance of zero-commented data in the dataset, we anticipate a substantial impact on the outcome due to the distinctive features of such data.

Results & Analysis



The ROC-AUC of the models we trained

Results & Analysis



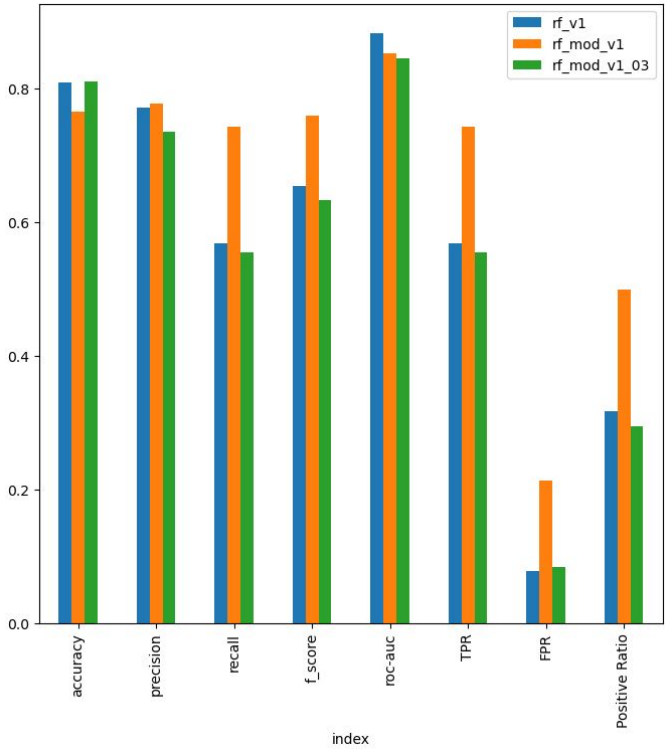
1. Model v.s. Model :

To analysize the presentation of models, we can see the area under the ROC line, as known as AUC, as the prediction ability of the model. The higher AUC score is, the more powerful the model is.

Obviously, compared with Xgboost, Random Forest shows the largely same powerful prediction ability in version 1 and even in modified version 1, which we generally take >0.8 score both obtain as wonderful admiration.

From this phenomenon, we could say that there is nearly NO DIFFERENCE between Xgboost and Random Forest and they are good enough as well.

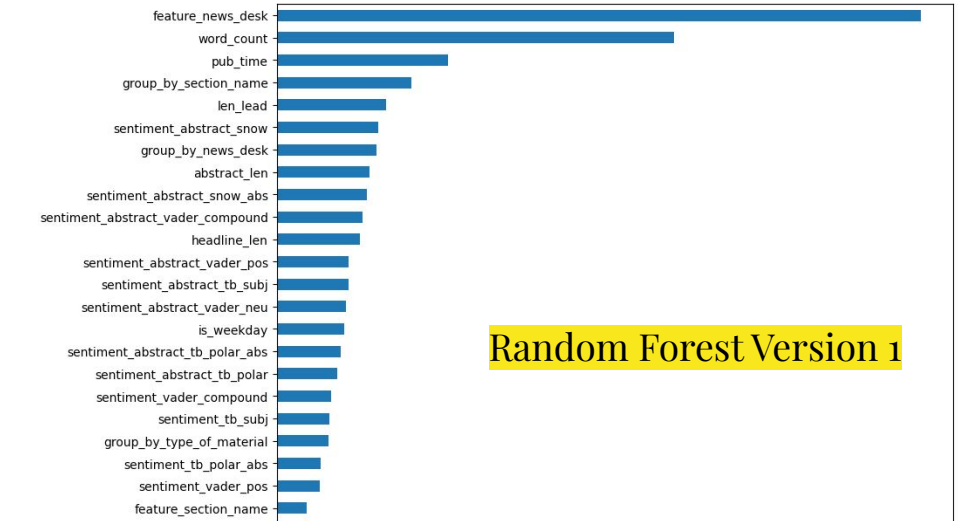
Results & Analysis



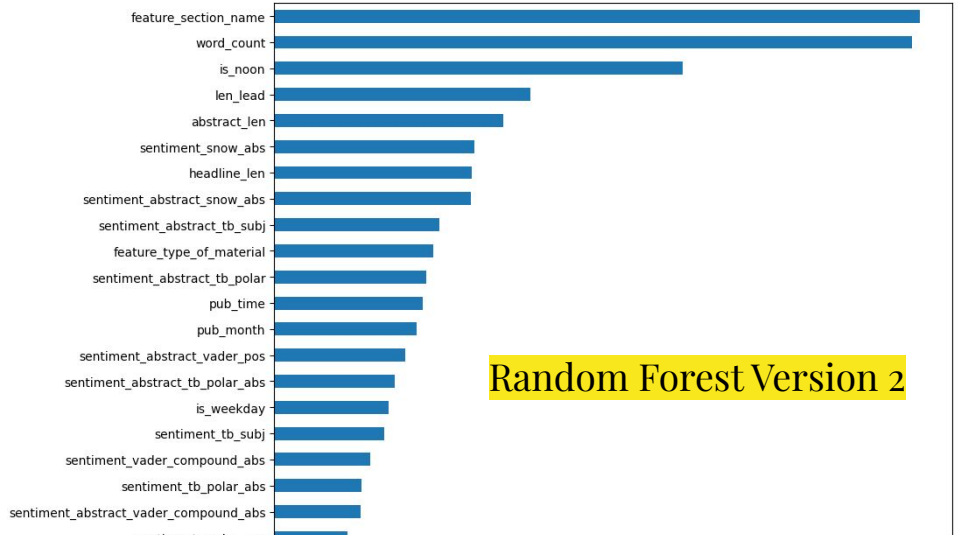
2. Modified v.s. Unmodified :

For V1, Roc-Auc and Precision perform better than those of modified_v1_0.3. The zero-commented data truly affects the result, which fits the hypothesis we guess.

Since modified_v1 possesses higher ratio of positive data, it has advantage of predicting positive result, which increases not only Recall, F-score, and TPR, but also FPR simultaneously.



Random Forest Version 1



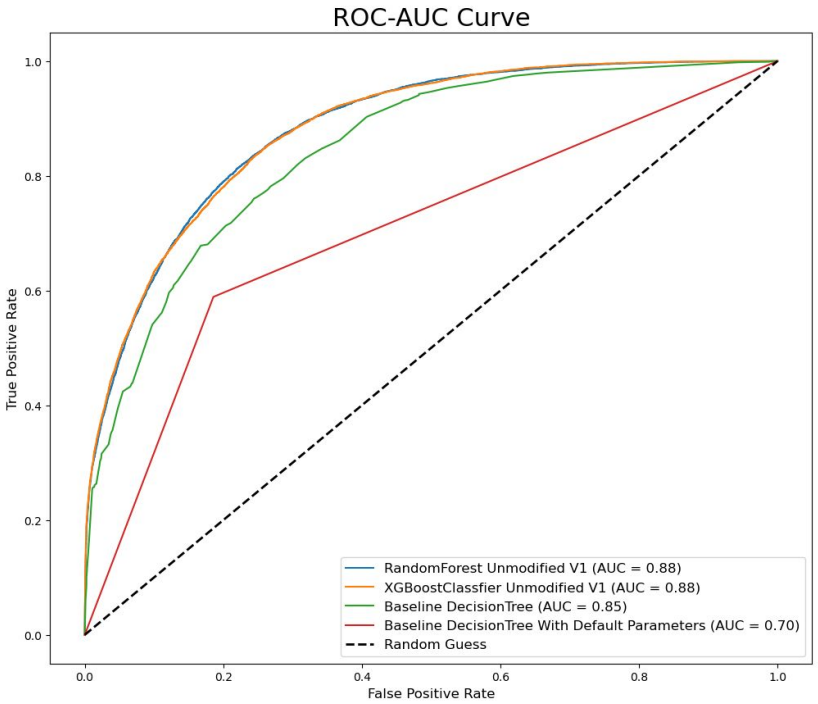
Random Forest Version 2

3. Version 1 v.s. Version 2:

In version 2, we eliminate the pairs of high correlation features, that is much dependent pairs.

However, we apparently ignore the role of importance they're involved in the process of model learning, which might bring about an opposite consequence if we don't take it into account.

Results & Analysis



4. Beat Baseline :

Seeing that the performance of the model trained by V1 feature is better than the others,

Therefore, we choose V1 to train the baseline model and compare with our models trained by the same data.

Contrasting to the baseline, regardless of Xgboost or Random Forest model, our approach diplayes an subtle growth on predicting popularity, that is our main purpose.

Limit

Limit

1. Data amount :

We look forward to acquire much more articles.

First, huge database is closer to the natural environment, which is supposed to be a normal distribution in the analysis.

Second, more different, powerful features may provide accurate and influent information for the learning of models.

Limit

2. Hardware limitation :

Besides that the 100,000 articles fetched by web crawler take several hours all the time, hardware also limits the depth our models reach and the branches.

According to the model nature, trees always draw and evaluate the sole feature once, which is lack of reference to multiple factors on each decision. Although we can see the tremendous cost of amending the error of splitting, however, boosting the efficiency of model learning is almost in our sight.

Practice

Practice

More and more Youtubers, vlogers, or even music productors are now aware of BREAKING the algorithm for higher watch on their achievements.

But we know it's risky to fight against the monster like Google. Then why don't we CONSTRUCT our own algorithm.

By utilizing our research, just let models learn what the trend of the popularity looks like. Perhaps in the near future, we are helping not only flourish the global journalism providing new aspects of jobs, but also enlarge the stage for those who are seizing an opportunity to be shiny under the spotlight.

Conclusion

Conclusion

1. To our surprise, there's no tremendous difference of performance between baseline model and our models. But due to optimizing the designation of Xgboost and Random Forest, we are still aware of the subtle progress.

2. We underrate some implicit features that have such influence on model learning, and we also regard integrity of data as an important role producing biased results from models. The more various data is, the more complete environment our models can learn in.

Other

Others

Github:

- Source code

Reference :

- NYT-Article-Popularity: Predicting online news popularity using New York Times articles in 2020
- The New York Times API
- Random forest
- Xgboost

Contribution :

- 吳權祐 : Web crawl、Xgboost
- 關立 : Random Forest
- 王邦碩 : Data process (feature)
- 劉秉驊 : Slide、Video

Presentation

END