Cs 699

Project Report

Qiren Sun, Yuhan Wang

03/26/2019

Statement of mining goal

TikTok, also known as **Douyin** (Chinese: 抖音; pinyin: *Dǒuyīn*; literally: "vibrating sound") in China, is a media app for creating and sharing short videos. Owned by ByteDance, the media app was launched as Douyin in China in September 2016 and introduced to the overseas market as TikTok one year later. It is a leading short video platform in Asia, United States, and other parts of the world. In 2018, the application gained popularity and became the most downloaded app in the U.S. in October 2018.

As of 2018, it is available in over 150 markets, and in 75 languages. The application allows users to create short videos of 15 seconds. In July 2018, the app had more than 500 million users globally.

For this project, we want to predict whether an video generator is the top 25 video generator in the app TikTok according all attributes we select like fans, comment, opus number and etc.,

Dataset description

This dataset includes the statistics of the video generators (fans numbers, opus numbers, comment numbers, etc.) in Tik Tok.

Attribute:

User Id: video generator’s id in Tik Tok (short video platform)

Name: video generator’s name

Rank Date: video generator’s rank date (for instance, video generator’s rank in 2/10/2019)

New Rank Index: video generator’s rank index (nri(New Rank Index) higher, video generator’s rank higher)

Type: video generator’s video type

Follower Number: video generator’s follower number (it’s the follower number at the rank date)

Follower Number Increase: it’s the video generator’s follower number increase at the rank date

Repost Number: repost number by other Tik Tok’s users at the rank date

Opus Number: video generator’s opus number at the rank date

Like Number: video generators are liked by other Tik Tok’s users at the rank date

Comment Number: other Tik Tok’s users comment number at the rank date

Rank Position: video generator’s current rank

Huoshan Fans Number: Huoshan(short video platform) current fans number

Original Music be Used Number: original music be used by other Tik Tok’s users

Toutiao Fans: Toutiao(a Beijing-based news and information content platform) current fans number

Fans Total: video generator's current fans number in Tik Tok, Huoshan, and Toutiao.

Douyin Fans: video generator's current fans in Tik Tok

Original Music Number: video generator's current original music number

Like Opus Number: video generator's current opus are liked by other Tik Tok's users

Like Number: video generators are liked by other Tik Tok's users at present

Opus Number: video generator's current opus number

Dynamic Number: video generator's total works

Attribute Selection

Here are the several attribute combinations we used in this project.

1. CfsSubsetEval : Evaluates the worth of a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them.

GreedyStepwise : Performs a greedy forward or backward search through the space of attribute subsets.

Attribute:
'opusNum','originalMusicBeUsedNum'

2. ClassifierAttributeEval : Evaluates the worth of an attribute by using a user-specified classifier.
Ranker : Ranks attributes by their individual evaluations.

Attribute:
'opusNum','followerNumInc','type','followerNum','name','likeNum','commentNum','uid'

3. CorrelationAttributeEval : Evaluates the worth of an attribute by measuring the correlation (Pearson's) between it and the class.
Ranker : Ranks attributes by their individual evaluations.

Attribute:
'likeNum','commentNum','likeNum','douyinFansNum','fansTotal','followerNum','repostNum', 'toutiaoFansNum','originalMusicBeUsedNum','originalMusicNum','type'

4. ReliefFAttributeEval : Evaluates the worth of an attribute by repeatedly sampling an instance and considering the value of the given attribute for the nearest instance of the same and different class.
Ranker : Ranks attributes by their individual evaluations.

Attribute:
'name','likeNum','commentNum','originalMusicBeUsedNum','toutiaoFansNum','repostNum','originalMusicNum','followerNumInc','huoshanFansNum'

5. Chosen by myself
Attribute:
'commentNum','followerNum','followerNumInc','type','repostNum','likeNum','opusNum', 'likeOpusNum','fansTotal','douyinFansNum','dynamicNum'

We didn't select 'nri' because it is hard to figure out the meaning of this attribute. 'nri' may equal to w1*'fansTotal'+w2*'likeNum'+…+w*x, but we don't know the value of w and the exact number of x.
We selected the attribute through Weka and implemented the classification through sklearn (python).

Training Set:

Dataset*0.7

Testing Set:

Dataset*0.3

Labels:

1 (top 25) and 0 (not in top 25)

Data processing Procedure

Clean_data.py:
Substituting the Chinese name into integer and the rankPosition with 0(top 25) and 1(not in top 25).

API.py:
This app is used to retrieve TOP 50 types in TikTok video such as entertainment, games, sports, science and etc. In each class, we select top 50 videos among them and formed the dataset.

Attribute_selectionClassfy.py:
This file is used to select attributes we retrieved and classify them. We used KNN, Gaussian Naïve Bayes, SVC, MLP in order to classify attributes and data. Before we use these classifiers, we preprocess the dataset to get the 70% of data as the training set and the 30% of the data as the testing set. In the last section, we invoke these four classifiers mentioned above.

Code Explanation

Packages

In [ ]:
```
import csv
import pandas as pd
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from matplotlib.ticker import MaxNLocator
import numpy as np
import matplotlib.pyplot as plt
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve, auc
```

Attribute Selection

In [ ]:
```
#1. ClassifierAttributeEval
X1=dm[['opusNum','followerNumInc','type','followerNum','name','likeNum','commentNum','uid']]

#2. CfsSubsetEval
X2=dm[['opusNum','originalMusicBeUsedNum']]


#3. CorrelationAttributeEval
X3=dm[['likeNum','commentNum','likeNum','douyinFansNum','fansTotal','followerNum','repostNum',
    'toutiaoFansNum','originalMusicBeUsedNum','originalMusicNum','type']]

#4. ReliefFAttributeEval
X4=dm[['name','likeNum','commentNum','originalMusicBeUsedNum','toutiaoFansNum','repostNum','originalMusicNum',
      'followerNumInc','huoshanFansNum']]


#Chosen by myself
X5=dm[['commentNum','followerNum','followerNumInc','type','repostNum','likeNum','opusNum',
      'likeOpusNum','fansTotal','douyinFansNum','dynamicNum']]

Y=dm[['rankPosition']].values
```

Preprocess data to get the training set(70%) and test set(30%)

In [ ]:
```
def preprocess(X,Y,selection):
    global X_train,X_test,Y_train,Y_test,x_valid,y_valid,X_test,Y_test
    le=LabelEncoder()
    z=StandardScaler().fit_transform(X)
    Y=le.fit_transform(Y)
    X_train,X_test,Y_train,Y_test=train_test_split(z,Y,test_size=0.3,random_state=0)
    x_valid=X_test[-20:,:]
    y_valid=Y_test[-20:]
    X_test=X_test[:-20,:]
    Y_test=Y_test[:-20]
    return
```

1. KNN (k-nearest neighbors)

In [ ]:
```python
def er_rate():
    global error_rate
    error_rate=[]
    for k in range(1,31,2):
        knn_classifier= KNeighborsClassifier(n_neighbors=k)
        knn_classifier.fit(X_train,Y_train)
        pred_k=knn_classifier.predict(X_test)
        error_rate.append(np.mean(pred_k!=Y_test))
    #y_test=np.array(np.random.randint(0,2,(52,)))
    #y_test=y_test.astype(float)

    plt.figure(figsize=(10,4))
    ax=plt.gca()
    ax.xaxis.set_major_locator(MaxNLocator(integer=True))
    plt.plot(range(1,31,2),error_rate,color='red',linestyle='dashed',
             marker='o',markerfacecolor='black',markersize=10)
    plt.title('Error rate vs. k for Tik Tok Subset')
    plt.xlabel('number of neighbors: k')
    plt.ylabel('Error Rate')
    return


def KNN(x_valid,y_valid,k):
    knn_classifier= KNeighborsClassifier(n_neighbors=k)
    knn_classifier.fit(X_train,Y_train)
    y_score=knn_classifier.predict_proba(X_test)[:,1]

    new_instance=x_valid
    prediction=knn_classifier.predict(new_instance.reshape(len(y_valid),-1))

    #print('KNN Prediction: ',prediction)
    #print('KNN Valid: ', y_valid)
    print('KNN Correct', np.mean(prediction==y_valid))
    #c_m=confusion_matrix(Y_test,prediction)

    fpr_rf, tpr_rf, _ = roc_curve(Y_test, y_score)
    print('TP rate: '+ str(np.mean(tpr_rf).tolist()))
    print('FP rate: '+ str(np.mean(fpr_rf).tolist()))
    print('ROC Area: '+str(roc_auc_score(Y_test, y_score).tolist()))
    print('\n')
    return
```

2. Gaussian Naive Bayes

In [ ]:
```python
def Naive_Bayesian(x_valid,y_valid):
    NB_classifier = GaussianNB().fit(X_train, Y_train)
    y_score=NB_classifier.predict_proba(X_test)[:,1]

    new_instance = x_valid
    prediction = NB_classifier.predict(new_instance)

   # print('NB Prediction: ',prediction)
   # print('NB Valid: ', y_valid)
    print('NB Correct', np.mean(prediction==y_valid))
    #c_m=confusion_matrix(Y_test,prediction)

    fpr_rf, tpr_rf, _ = roc_curve(Y_test, y_score)
    print('TP rate: '+ str(np.mean(tpr_rf).tolist()))
    print('FP rate: '+ str(np.mean(fpr_rf).tolist()))
    print('ROC Area: '+str(roc_auc_score(Y_test, y_score).tolist()))
    print('\n')
    return
```

3. SVC (Support Vector Machines)

In [ ]:
```python
def SVC_classify(X_test,Y_test):
    clf=SVC(gamma='auto')
    clf.fit(X_train,Y_train)
    y_score=clf.fit(X_train,Y_train).decision_function(X_test)

    prediction=clf.predict(X_test.reshape(len(Y_test),-1))
    correct=np.mean(prediction==Y_test)
    #print('SVC Prediction: ',prediction)
    #print('SVC Valid: ', Y_test)
    print('SVC Correct', correct)
    #c_m=confusion_matrix(Y_test,prediction)

    fpr_rf, tpr_rf, _ = roc_curve(Y_test, y_score)
    print('TP rate: '+ str(np.mean(tpr_rf).tolist()))
    print('FP rate: '+ str(np.mean(fpr_rf).tolist()))
    print('ROC Area: '+str(roc_auc_score(Y_test, y_score).tolist()))
    print('\n')
    return
```

4. MLP (multilayer perceptron)

A feedforward neural network

```
In [ ]: def mlp_classifier(X_test,Y_test):
            mlp=MLPClassifier(hidden_layer_sizes=(50,), max_iter=50, alpha=1e-4,
                              solver='sgd', verbose=10, tol=1e-4, random_state=1,
                              learning_rate_init=.1)
            #mlp.fit(X_train,Y_train)
            y_score=mlp.fit(X_train,Y_train).predict_proba(X_test)[:,1]
            prediction=mlp.predict(X_test.reshape(len(Y_test),-1))
            correct=np.mean(prediction==Y_test)
            print('\nMLP Correct', correct)
            #c_m=confusion_matrix(Y_test,prediction)

            fpr_rf, tpr_rf, _ = roc_curve(Y_test, y_score)
            print('TP rate: '+ str(np.mean(tpr_rf).tolist()))
            print('FP rate: '+ str(np.mean(fpr_rf).tolist()))
            print('ROC Area: '+str(roc_auc_score(Y_test, y_score).tolist()))
            print('\n')

            return
```

Invoke the four different classifier

```
In [ ]: dict1={'Attribute Selection':X1,'CfsSubsetEval':X2,'CorrelationAttributeEval':X3,
               'ReliefFAttributeEval':X4,'Chosen by myself':X5}
        for selection,i in dict1.items():
            print(selection)
            preprocess(i,Y,selection)
            er_rate()
            num=error_rate.index(min(error_rate))
            k=2*num+1
            KNN(X_test,Y_test,k)
            Naive_Bayesian(X_test,Y_test)
            SVC_classify(X_test,Y_test)
            mlp_classifier(X_test,Y_test)
            print('\n')
```

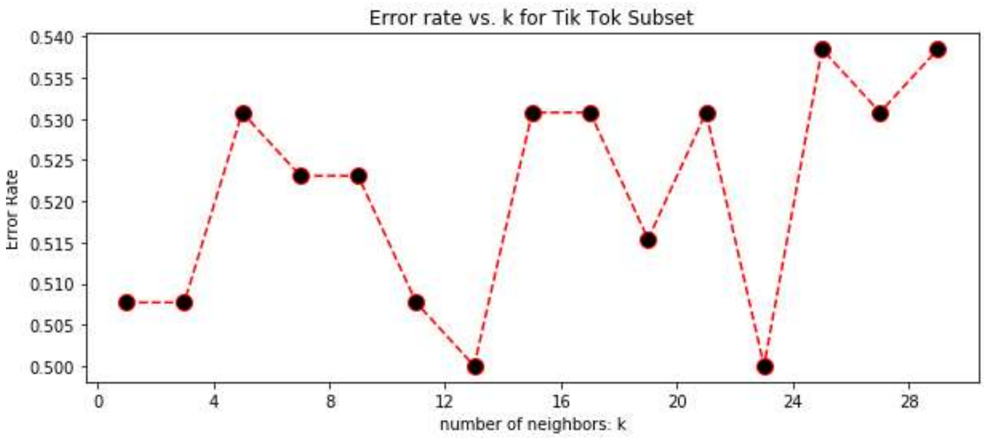
Data Mining Result

Attribute_Selection&Classify.py (Python)

1. k-NN (***k*-nearest neighbors**)

For k in range(1,31,2), I selected the k had the lowest error rate.

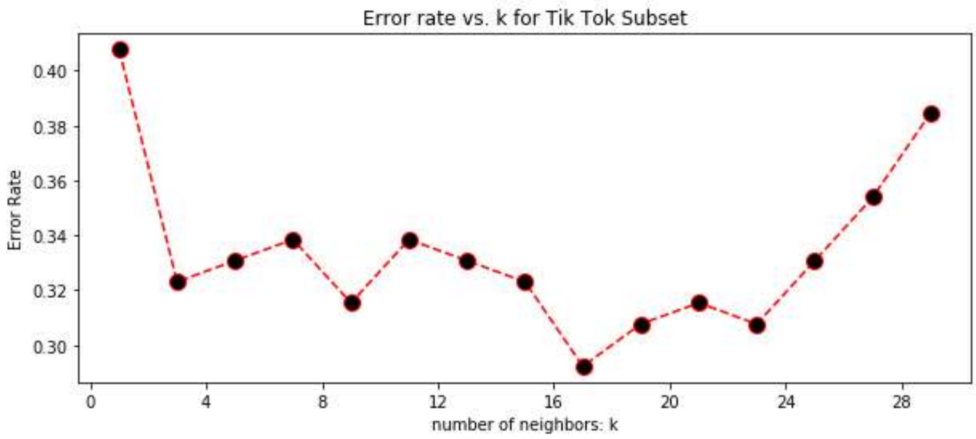
CfsSubsetEval



KNN Correct 0.5
TP rate: 0.6219780219780221

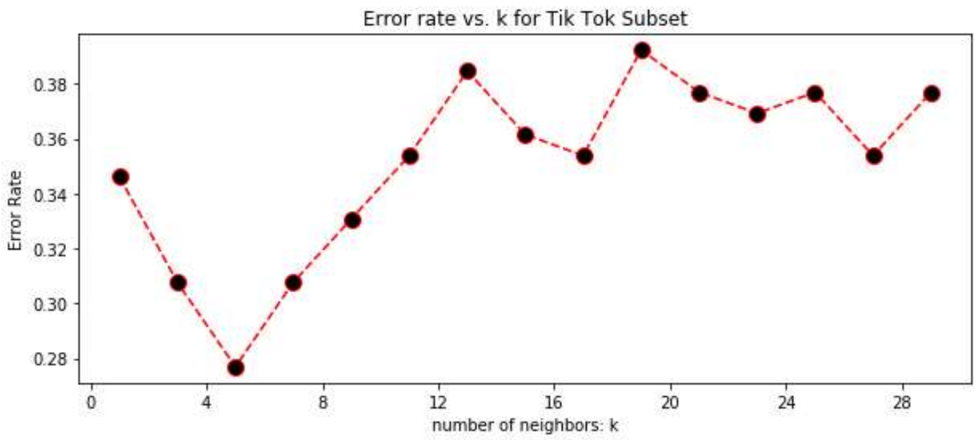
FP rate: 0.6175824175824175
ROC Area: 0.5128994082840237

ClassifierAttributeEval



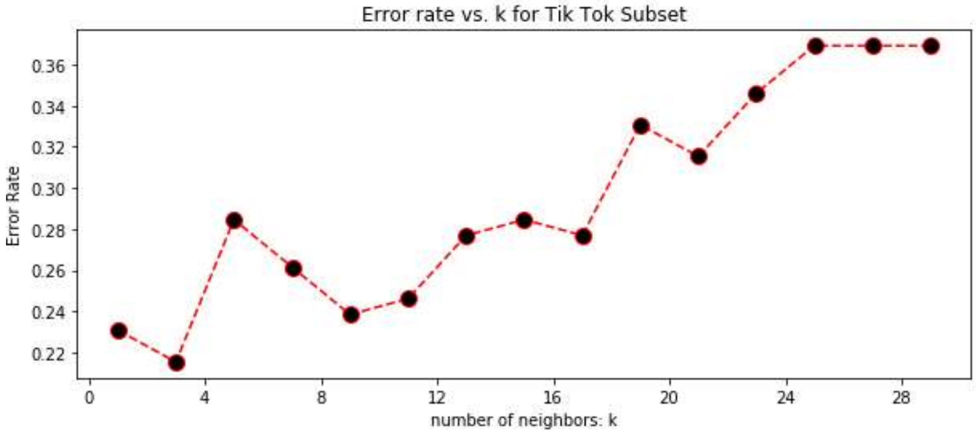
KNN Correct 0.7076923076923077
TP rate: 0.5100591715976331
FP rate: 0.3195266272189349
ROC Area: 0.7602366863905325

CorrelationAttributeEval



KNN Correct 0.7230769230769231
TP rate: 0.567032967032967
FP rate: 0.3494505494505495
ROC Area: 0.781775147928994

ReliefFAttributeEval



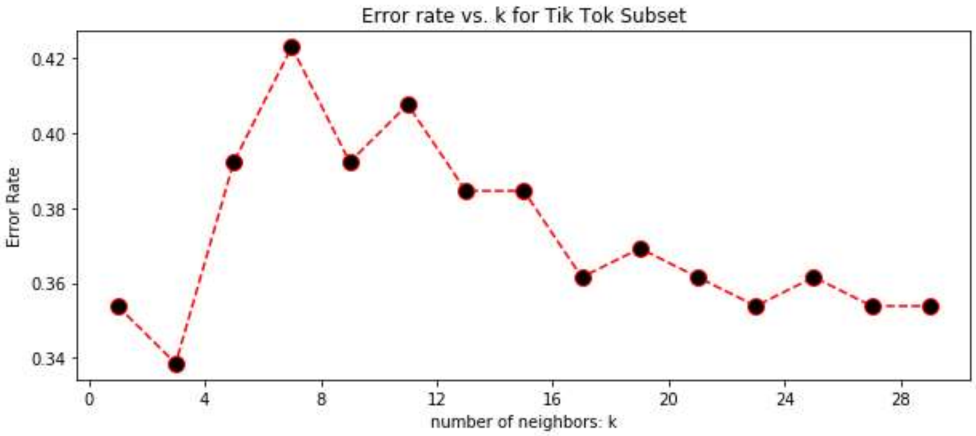
KNN Correct 0.7846153846153846

TP rate: 0.6123076923076923

FP rate: 0.35692307692307695

ROC Area: 0.8359763313609468

Chosen by myself



KNN Correct 0.6615384615384615

TP rate: 0.5230769230769231

FP rate: 0.38461538461538464

ROC Area: 0.7002366863905325

2. NB (Naïve Bayes)

CfsSubsetEval

NB Correct 0.4846153846153846

TP rate: 0.22403846153846152

FP rate: 0.25384615384615383

ROC Area: 0.5022485207100592

ClassifierAttributeEval

NB Correct 0.6307692307692307

TP rate: 0.6495436766623206
FP rate: 0.45084745762711864
ROC Area: 0.6972781065088757

CorrelationAttributeEval

NB Correct 0.5846153846153846
TP rate: 0.5778357235984355
FP rate: 0.43546284224250326
ROC Area: 0.6231952662721894

ReliefFAttributeEval

NB Correct 0.5846153846153846
TP rate: 0.6093406593406593
FP rate: 0.4706043956043956
ROC Area: 0.6371597633136095

Chosen by myself

NB Correct 0.6307692307692307
TP rate: 0.6243589743589745
FP rate: 0.48743589743589744
ROC Area: 0.6424852071005918

3. SVC(C-Support Vector Classification)

ClassifierAttributeEval

SVC Correct 0.7
TP rate: 0.6870703764320786
FP rate: 0.3250409165302782
ROC Area: 0.81301775147929

CfsSubsetEval
SVC Correct 0.4846153846153846
TP rate: 0.30871794871794866
FP rate: 0.3282051282051282
ROC Area: 0.4847337278106509

CorrelationAttributeEval
SVC Correct 0.6384615384615384
TP rate: 0.6290275761973875
FP rate: 0.40435413642960816
ROC Area: 0.7114792899408284

ReliefFAttributeEval
SVC Correct 0.6538461538461539
TP rate: 0.6447552447552448

FP rate: 0.44391608391608395
ROC Area: 0.6979881656804734

Chosen by myself
SVC Correct 0.6692307692307692
TP rate: 0.6824383164005805
FP rate: 0.49259796806966627
ROC Area: 0.7017751479289941

4. MLPClassifier (Multi-layer Perceptron classifier)

ClassifierAttributeEval
MLP Correct 0.8
TP rate: 0.7701357466063348
FP rate: 0.23619909502262443
ROC Area: 0.9048520710059171

CfsSubsetEval
MLP Correct 0.49230769230769234
TP rate: 0.44807692307692304
FP rate: 0.4634615384615385
ROC Area: 0.4840236686390532

CorrelationAttributeEval
MLP Correct 0.6307692307692307
TP rate: 0.6053981106612686
FP rate: 0.37246963562753044
ROC Area: 0.7150295857988166

ReliefFAttributeEval
MLP Correct 0.6538461538461539
TP rate: 0.6500754147812972
FP rate: 0.4117647058823529
ROC Area: 0.7301775147928994

Chosen by myself
MLP Correct 0.6692307692307692
TP rate: 0.6578249336870027
FP rate: 0.4610079575596817
ROC Area: 0.695621301775148

Model selection:

Among all 20 models, we select the best attribute combination of each algorism first. For KNN, the best combination would be the ReliefFAttributeEval with the highest accuracy, highest ROC area and relatively higher TP rate and lower FP rate. For Naïve Bayes, the best performance combination is the ClassifierAttributeEval with highest accuracy, highest ROC area and relatively higher TP rate and lower FP rate. For SVC, the best combination is also the ClassifierAttributeEval with highest accuracy, highest ROC area, highest TP rate and lowest FP rate. Last but not least, for MLPClassifier, it is also the ClassifierAttributeEval with the best performance. It has highest accuracy, highest ROC area, highest TP rate and lowest FP rate.

Overall, ClassfierAttributeEval seems like the best combination of attribute among all combinations. With all four algorisms, the MLPClassfier has the best performance of measurement comparing to ReliefAttributeEval of KNN and ClassigierAttributeEval of the other two algorisms. As a result, it would be rational to select ClassifierAttributeEval of MLPClassfier algorism as the best model among 20 of them.

Personal Contribution:

Qiren Sun: Dataset Selection, Choosing algorisms.
Yuhan Wang: Dataset Selection, Using API to get dataset.
Together: Performing test using python and weka, drawing conclusion upon the test results.

Conclusion:

For this project, we want to predict whether a video generator is the top 25 video generator in the app TikTok. After implementing five different attribute selection algorithms, we used four different classifiers to get the accuracy, TP rate, FP rate, and ROC Area. The final result showed that the best choice is ClassifierAttributeEval of MLPClassfier algorism. Using these attributes and combinations, we can easily predict the ranking of the video posters. It is essential to know since there are a lot of online media companies nowadays looking for employ those media producers and invest in them.