Corporate Investment Decision-making

组名：一点点

[陆之东 乐雅馨 冯圣飞 杭慧丽 王莹]

CONTENTS

01 Description

02 Preparation

03 The Development Process

04 Optimization

01

Description

Description

Our program aims to employ the F early warning model to recognize magnitude of risks based on principal component analysis and logistic regression analysis to decide which company to invest in.

02

Preparation

Preparation

Data to collect

Preparations

The Model We Used

Data to Collect

The SEC’s EDGAR database provides free public access to corporate information, allowing us to quickly research a company’s financial information and operations by reviewing registration statements, prospectuses and periodic reports filed on Forms 10-K and 10-Q.

Preparations

01% HTML Document & communication

02% Python Library

parse HTML Document

realize HTTP communication

03% Regular expression

04% Tools and calculation methods to analyze financial statements.

The Model We Used

$F=0.1774+1.1091W1+1.9271W2+0.1074W3+0.0302W4+0.04961W5$

An early warning system (EWS) is a system which is used for identifying current situations and predicting the risk level

The F early warning system has been improved upon the Z early warning system, including cash flow as a predictive variable.

A simplified F early warning system model:

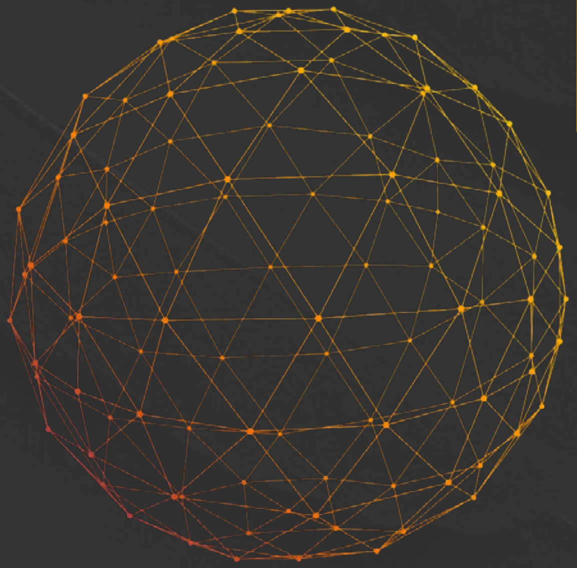
$F=-0.1774+1.1091W1+1.9271W2.$

03

The Development Process

3.1

Part 1



Skeleton

As some unpredictable errors may appear when Python program runs for a long period, we decide to divide this part of program into three steps, so that the running time of each step can be decreased and this kind of error can be avoided.

Skeleton

Step 1

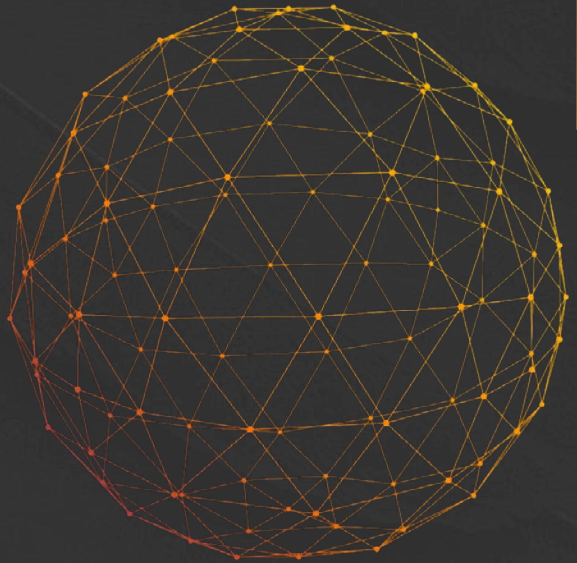
Search “CHINA” in the SEC.gov | EDGAR | Search Tools. Get the CIK codes. Save those in “10-XList”.

Step 2

Get the URLs.
Save them into the file ”chartUrls”.

Step 3

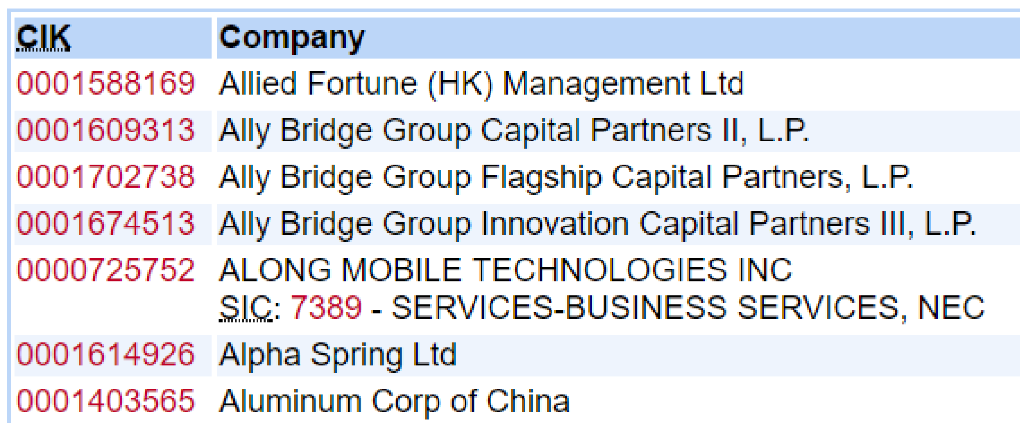
Visit the statements through the URLs in ”chartUrls”.



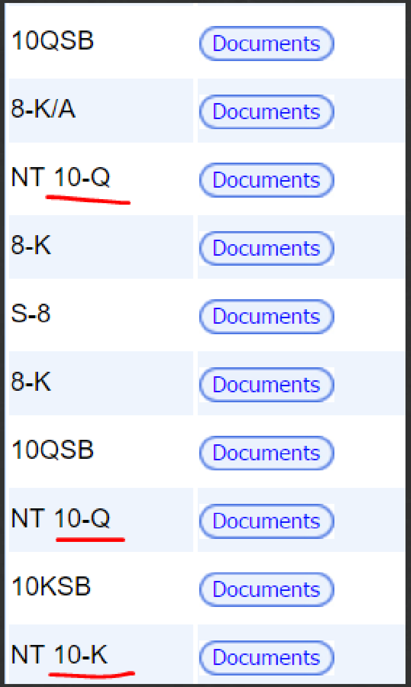
More Details

Step One

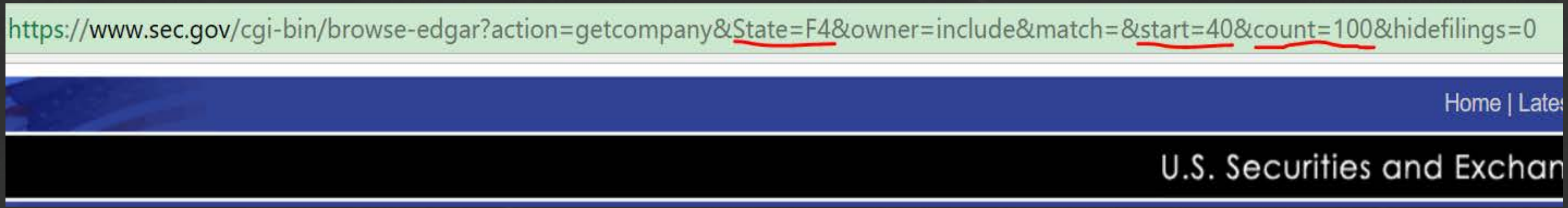
Obtain the URL and analyze its composition: “country=F4” “start” “count” . Set the valid range to 100. Put CIK code into the CIK list. Visit the URL. Search through the web page . Write the URL into “10-Xlist” . Get all the URL.



CIK	Company
0001588169	Allied Fortune (HK) Management Ltd
0001609313	Ally Bridge Group Capital Partners II, L.P.
0001702738	Ally Bridge Group Flagship Capital Partners, L.P.
0001674513	Ally Bridge Group Innovation Capital Partners III, L.P.
0000725752	ALONG MOBILE TECHNOLOGIES INC SIC: 7389 - SERVICES-BUSINESS SERVICES, NEC
0001614926	Alpha Spring Ltd
0001403565	Aluminum Corp of China



10QSB	Documents
8-K/A	Documents
NT 10-Q	Documents
8-K	Documents
S-8	Documents
8-K	Documents
10QSB	Documents
NT 10-Q	Documents
10KSB	Documents
NT 10-K	Documents



https://www.sec.gov/cgi-bin/browse-edgar?action=getcompany&State=F4&owner=include&match=&start=40&count=100&hidefilings=0

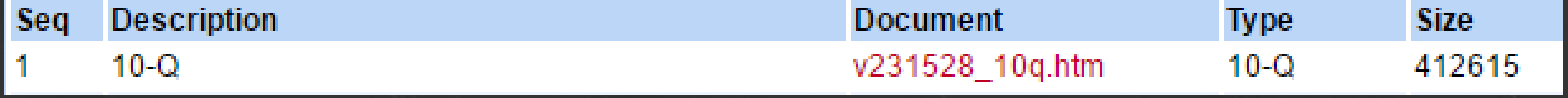
More Details

Step Two

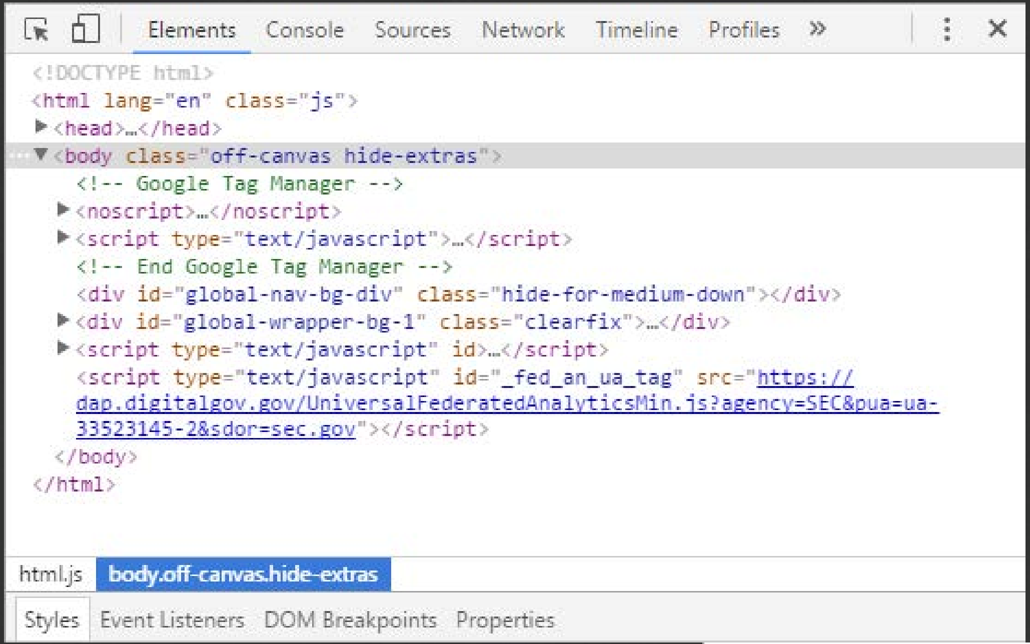
Search through the index , find the corresponding URL and save it in “chartUrls”.

Step Three

Judge every <tr> in the file. Take the average value of first two items. Save them in data.txt in the form of “CIK\tTCA\tTCL\tTSE\tNI\n”

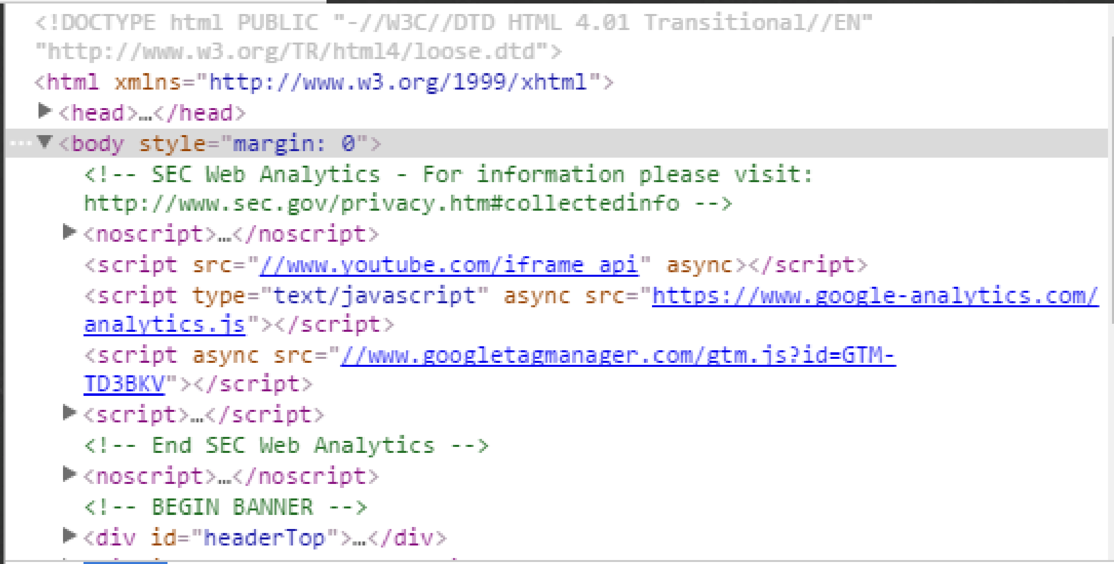


Seq	Description	Document	Type	Size
1	10-Q	v231528_10q.htm	10-Q	412615



```
<!DOCTYPE html>
<html lang="en" class="js">
▶<head>…</head>
▼<body class="off-canvas hide-extras">
    <!-- Google Tag Manager -->
  ▶<noscript>…</noscript>
  ▶<script type="text/javascript">…</script>
    <!-- End Google Tag Manager -->
    <div id="global-nav-bg-div" class="hide-for-medium-down"></div>
  ▶<div id="global-wrapper-bg-1" class="clearfix">…</div>
  ▶<script type="text/javascript" id>…</script>
    <script type="text/javascript" id="_fed_an_ua_tag" src="https://
    dap.digitalgov.gov/UniversalFederatedAnalyticsMin.js?agency=SEC&pua=ua-
    33523145-2&sdor=sec.gov"></script>
  </body>
</html>
```

Comments:
CIK——CIK code;
TCA——Total Current Assets;
TCL——Total Current Liabilities;
TSE——Total Stockholders’ Equity;
NI——Net Income



```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
▶<head>…</head>
▼<body style="margin: 0">
    <!-- SEC Web Analytics - For information please visit:
    http://www.sec.gov/privacy.htm#collectedinfo -->
  ▶<noscript>…</noscript>
    <script src="//www.youtube.com/iframe_api" async></script>
    <script type="text/javascript" async src="https://www.google-analytics.com/
    analytics.js"></script>
    <script async src="//www.googletagmanager.com/gtm.js?id=GTM-
    TD3BKV"></script>
  ▶<script>…</script>
    <!-- End SEC Web Analytics -->
  ▶<noscript>…</noscript>
    <!-- BEGIN BANNER -->
  ▶<div id="headerTop">…</div>
```

3.2

Part 2

part 2

read the data.

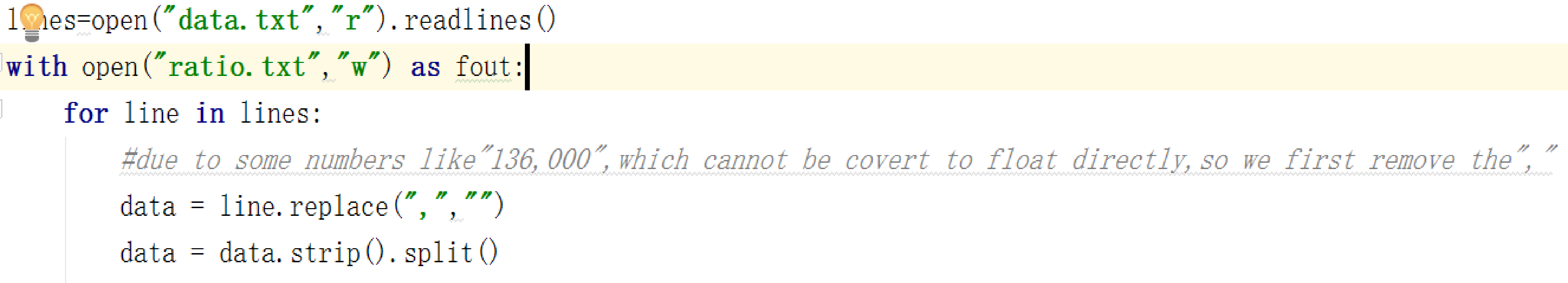
process the original data.

compute rational

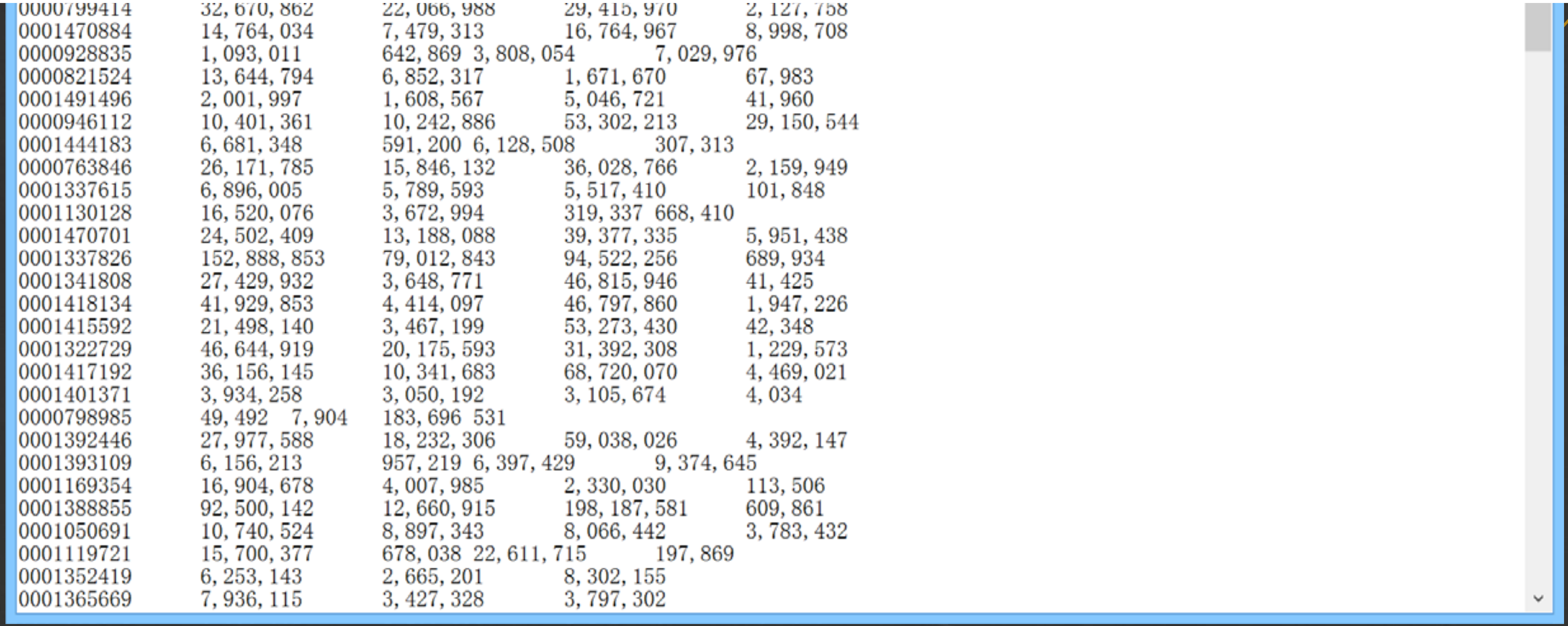
financial ratios

sort the F-score

read the data

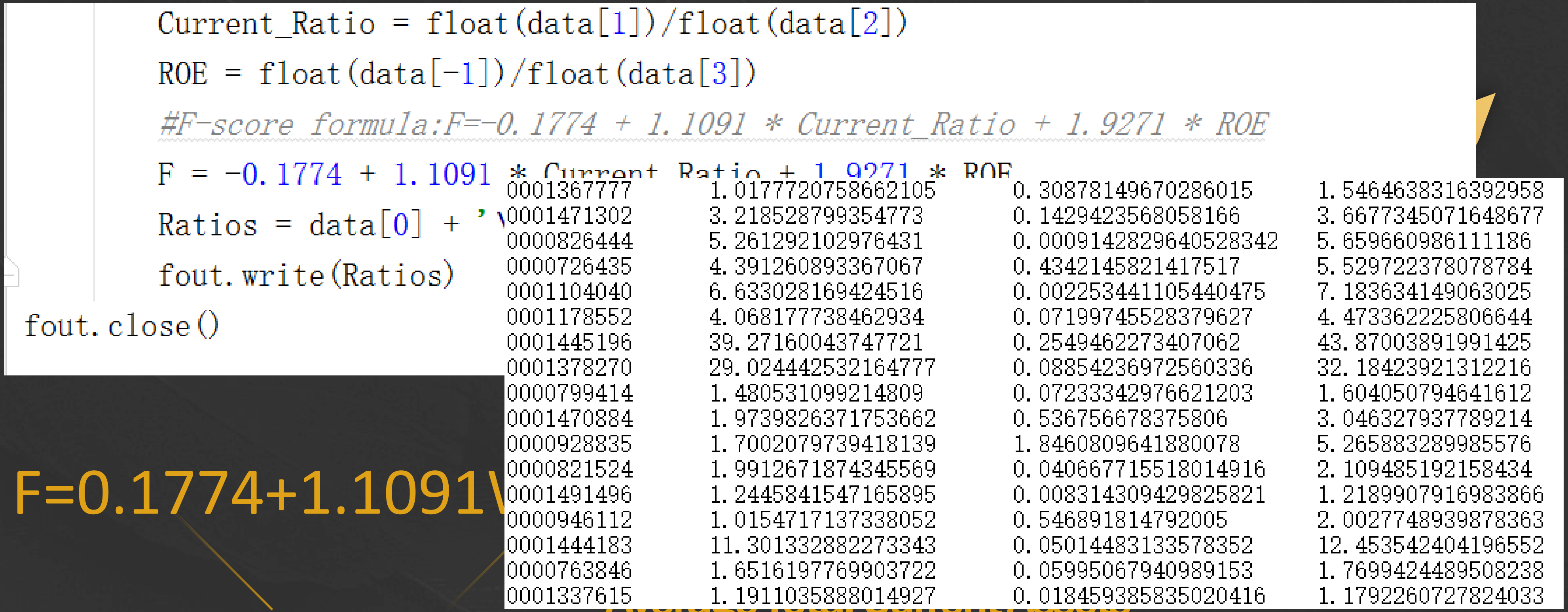


```python
lines=open("data.txt","r").readlines()
with open("ratio.txt","w") as fout:
    for line in lines:
        #due to some numbers like"136,000",which cannot be covert to float directly,so we first remove the","
        data = line.replace(",","")
        data = data.strip().split()
```



```
0000799414	32,670,862	22,066,988	29,415,970	2,127,758
0001470884	14,764,034	7,479,313	16,764,967	8,998,708
0000928835	1,093,011	642,869	3,808,054	7,029,976
0000821524	13,644,794	6,852,317	1,671,670	67,983
0001491496	2,001,997	1,608,567	5,046,721	41,960
0000946112	10,401,361	10,242,886	53,302,213	29,150,544
0001444183	6,681,348	591,200	6,128,508	307,313
0000763846	26,171,785	15,846,132	36,028,766	2,159,949
0001337615	6,896,005	5,789,593	5,517,410	101,848
0001130128	16,520,076	3,672,994	319,337	668,410
0001470701	24,502,409	13,188,088	39,377,335	5,951,438
0001337826	152,888,853	79,012,843	94,522,256	689,934
0001341808	27,429,932	3,648,771	46,815,946	41,425
0001418134	41,929,853	4,414,097	46,797,860	1,947,226
0001415592	21,498,140	3,467,199	53,273,430	42,348
0001322729	46,644,919	20,175,593	31,392,308	1,229,573
0001417192	36,156,145	10,341,683	68,720,070	4,469,021
0001401371	3,934,258	3,050,192	3,105,674	4,034
0000798985	49,492	7,904	183,696	531
0001392446	27,977,588	18,232,306	59,038,026	4,392,147
0001393109	6,156,213	957,219	6,397,429	9,374,645
0001169354	16,904,678	4,007,985	2,330,030	113,506
0001388855	92,500,142	12,660,915	198,187,581	609,861
0001050691	10,740,524	8,897,343	8,066,442	3,783,432
0001119721	15,700,377	678,038	22,611,715	197,869
0001352419	6,253,143	2,665,201	8,302,155
0001365669	7,936,115	3,427,328	3,797,302
```

compute financial ratios



```
        Current_Ratio = float(data[1])/float(data[2])
        ROE = float(data[-1])/float(data[3])
        #F-score formula:F=-0.1774 + 1.1091 * Current_Ratio + 1.9271 * ROE
        F = -0.1774 + 1.1091 * Current_Ratio + 1.9271 * ROE
        Ratios = data[0] + '\
        fout.write(Ratios)
fout.close()
```

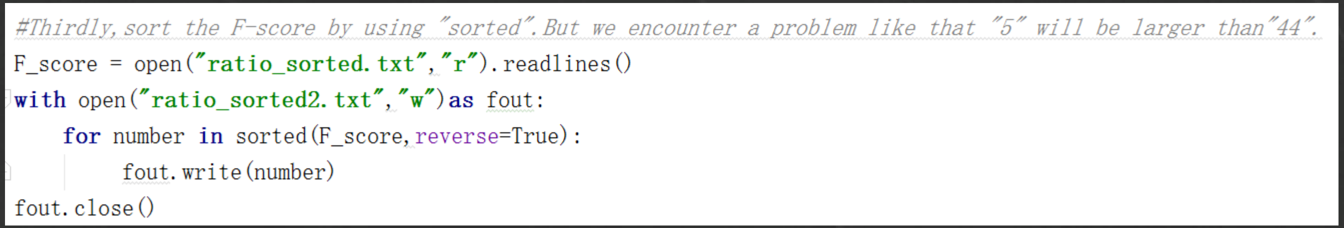
```
0001367777      1.0177720758662105      0.30878149670286015     1.5464638316392958
0001471302      3.218528799354773       0.1429423568058166      3.6677345071648677
0000826444      5.261292102976431       0.00091428296405283 42  5.659660986111186
0000726435      4.391260893367067       0.4342145821417517      5.529722378078784
0001104040      6.633028169424516       0.0022534411054404 75   7.183634149063025
0001178552      4.068177738462934       0.07199745528379627     4.473362225806644
0001445196      39.2716004374772 1      0.2549462273407062      43.87003891991425
0001378270      29.024442532164777      0.08854236972560336     32.18423921312216
0000799414      1.480531099214809       0.07233342976621203     1.604050794641612
0001470884      1.9739826371753662      0.536756678375806       3.046327937789214
0000928835      1.7002079739418139      1.846080964188007 8     5.265883289985576
0000821524      1.9912671874345569      0.04066771551801491 6   2.109485192158434
0001491496      1.2445841547165895      0.0083143094298258 21   1.2189907916983866
0000946112      1.0154717137338052      0.546891814792005       2.0027748939878363
0001444183      11.301332882273343      0.05014483133578352     12.453542404196552
0000763846      1.6516197769903722      0.05995067940989153     1.7699424489508238
0001337615      1.1911035888014927      0.018459385835020416    1.1792260727824033
```

F=0.1774+1.1091\

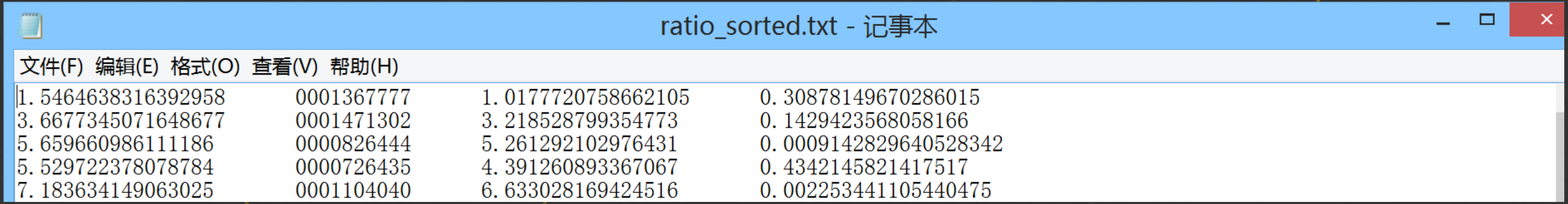
Current Ratio= $\dfrac{\text{Average Total Current Assets}}{\text{Average Total Current Liabilities}}$

Return On Equity= $\dfrac{\text{Net Income}}{\text{Average Total Equity}}$

Try to sort the F in the descending order

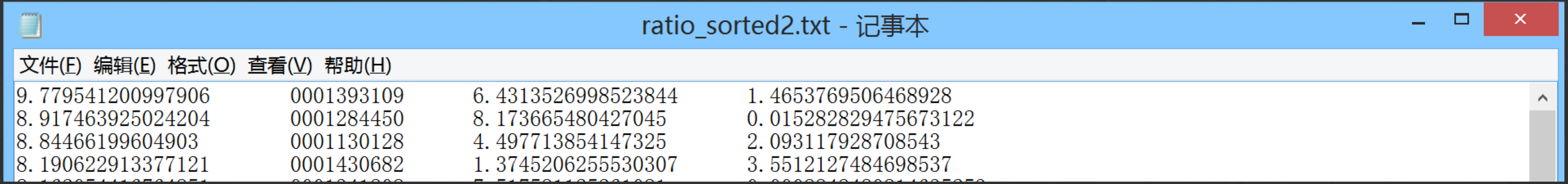


```
#Thirdly, sort the F-score by using "sorted". But we encounter a problem like that "5" will be larger than"44".
F_score = open("ratio_sorted.txt","r").readlines()
with open("ratio_sorted2.txt","w")as fout:
    for number in sorted(F_score,reverse=True):
        fout.write(number)
fout.close()
```



ratio_sorted.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
1. 5464638316392958    0001367777    1. 0177720758662105    0. 30878149670286015
3. 6677345071648677    0001471302    3. 218528799354773     0. 1429423568058166
5. 659660986111186     0000826444    5. 261292102976431     0. 0009142829640528342
5. 529722378078784     0000726435    4. 391260893367067     0. 4342145821417517
7. 183634149063025     0001104040    6. 633028169424516     0. 002253441105440475
```



ratio_sorted2.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
9. 779541200997906     0001393109    6. 4313526998523844    1. 4653769506468928
8. 917463925024204     0001284450    8. 173665480427045     0. 015282829475673122
8. 84466199604903      0001130128    4. 497713854147325     2. 093117928708543
8. 190622913377121     0001430682    1. 3745206255530307    3. 5512127484698537
```

3.3

Part 3

part 3

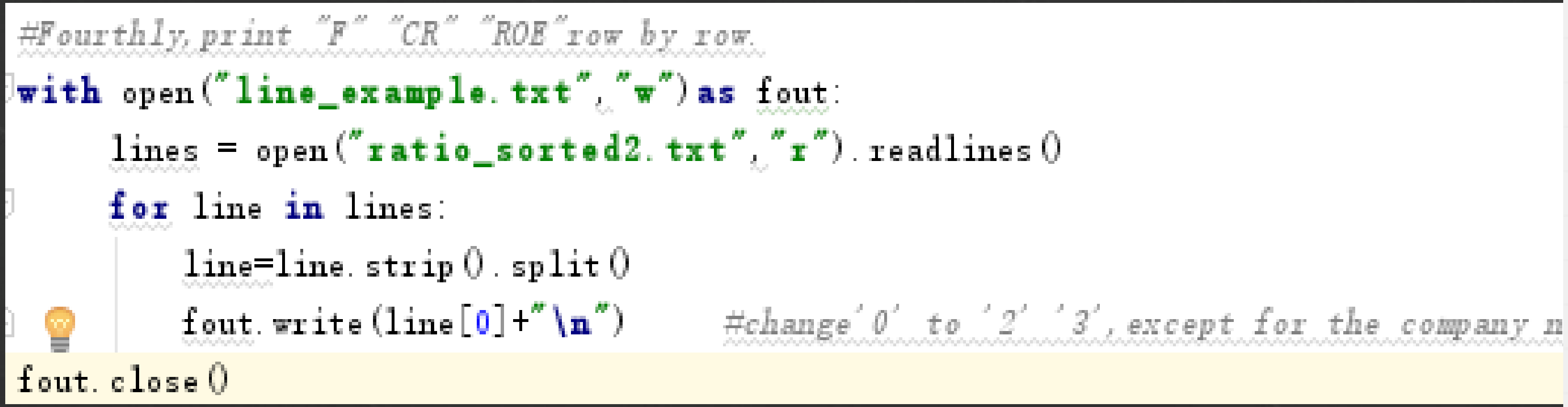
Step One:covert the format “txt.” into “csv.”

Step two: Utilize machine learning

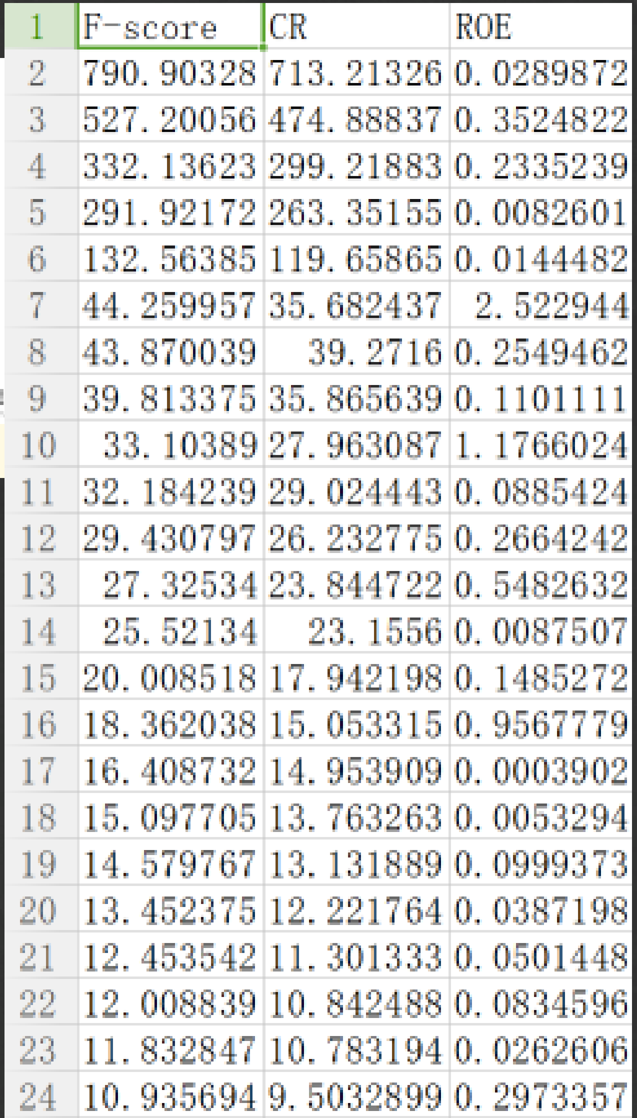
Step three: evaluate KNN-Regressor mode's accuracy

Step four: Export graphics

covert “txt.” into “csv.”

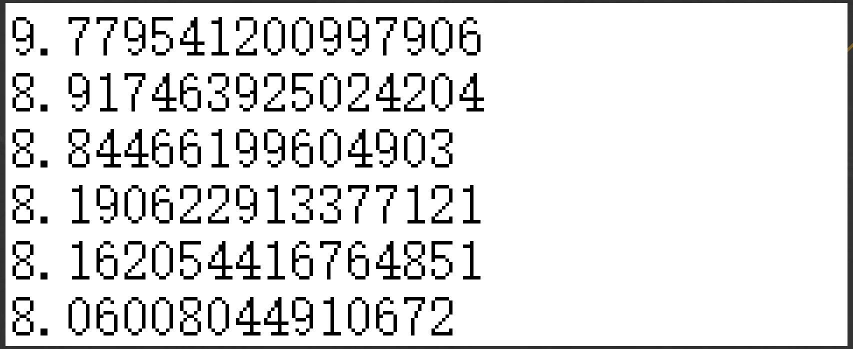


```
#Fourthly,print “F” “CR” “ROE”row by row.
with open("line_example.txt","w") as fout:
    lines = open("ratio_sorted2.txt","r").readlines()
    for line in lines:
        line=line.strip().split()
        fout.write(line[0]+"\n")      #change‘0’ to ‘2’‘3’, except for the company n
fout.close()
```



	F-score	CR	ROE
2	790.90328	713.21326	0.0289872
3	527.20056	474.88837	0.3524822
4	332.13623	299.21883	0.2335239
5	291.92172	263.35155	0.0082601
6	132.56385	119.65865	0.0144482
7	44.259957	35.682437	2.522944
8	43.870039	39.2716	0.2549462
9	39.813375	35.865639	0.1101111
10	33.10389	27.963087	1.1766024
11	32.184239	29.024443	0.0885424
12	29.430797	26.232775	0.2664242
13	27.32534	23.844722	0.5482632
14	25.52134	23.1556	0.0087507
15	20.008518	17.942198	0.1485272
16	18.362038	15.053315	0.9567779
17	16.408732	14.953909	0.0003902
18	15.097705	13.763263	0.0053294
19	14.579767	13.131889	0.0999373
20	13.452375	12.221764	0.0387198
21	12.453542	11.301333	0.0501448
22	12.008839	10.842488	0.0834596
23	11.832847	10.783194	0.0262606
24	10.935694	9.5032899	0.2973357

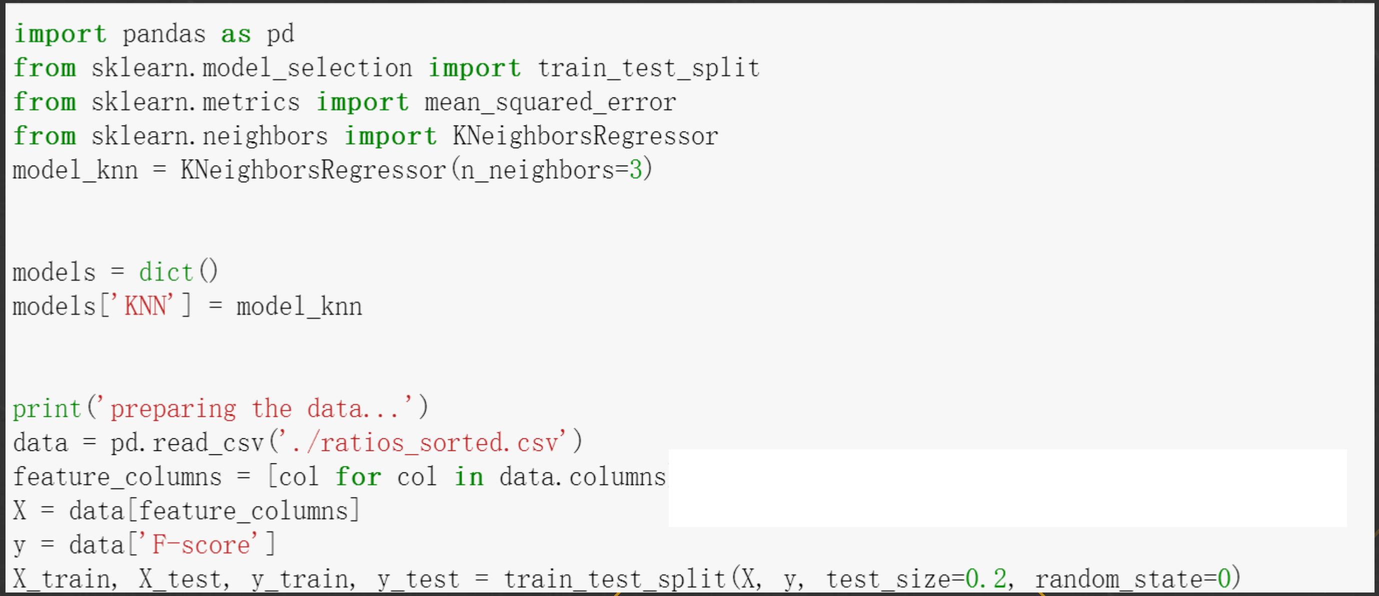
Print“F-score”“CR””ROE”seperately row by row



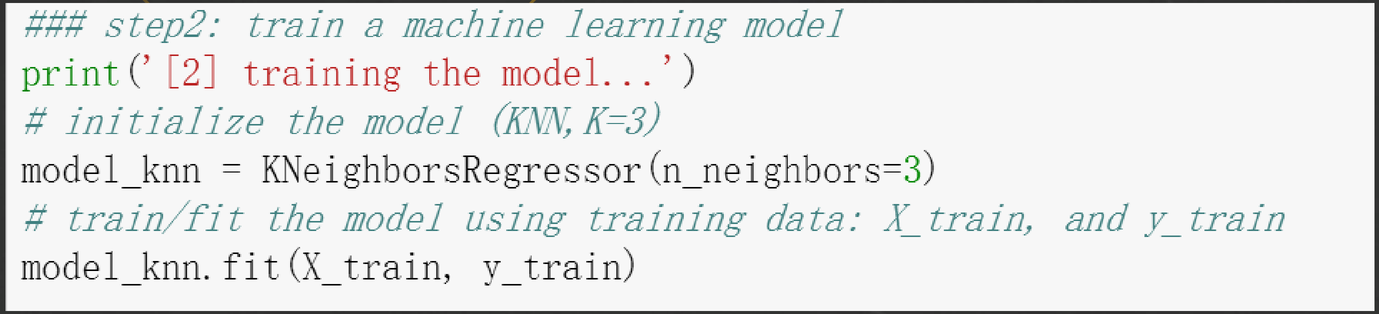
```
9.779541200997906
8.917463925024204
8.84466199604903
8.190622913377121
8.162054416764851
8.06008044910672
```

（Sample output）

Machine learning

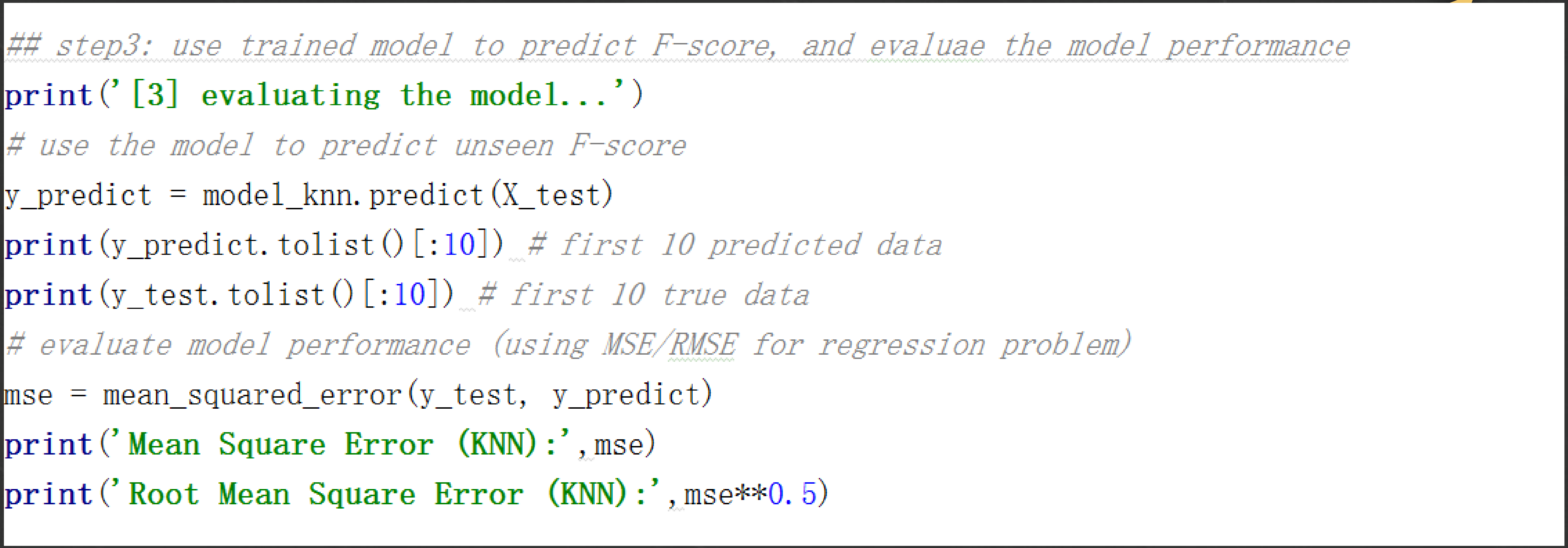


```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.neighbors import KNeighborsRegressor
model_knn = KNeighborsRegressor(n_neighbors=3)


models = dict()
models[’KNN’] = model_knn


print(’preparing the data...’)
data = pd.read_csv(’./ratios_sorted.csv’)
feature_columns = [col for col in data.columns
X = data[feature_columns]
y = data[’F-score’]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```



```
### step2: train a machine learning model
print(’[2] training the model...’)
# initialize the model (KNN, K=3)
model_knn = KNeighborsRegressor(n_neighbors=3)
# train/fit the model using training data: X_train, and y_train
model_knn.fit(X_train, y_train)
```

Evaluate the accuracy of KNN

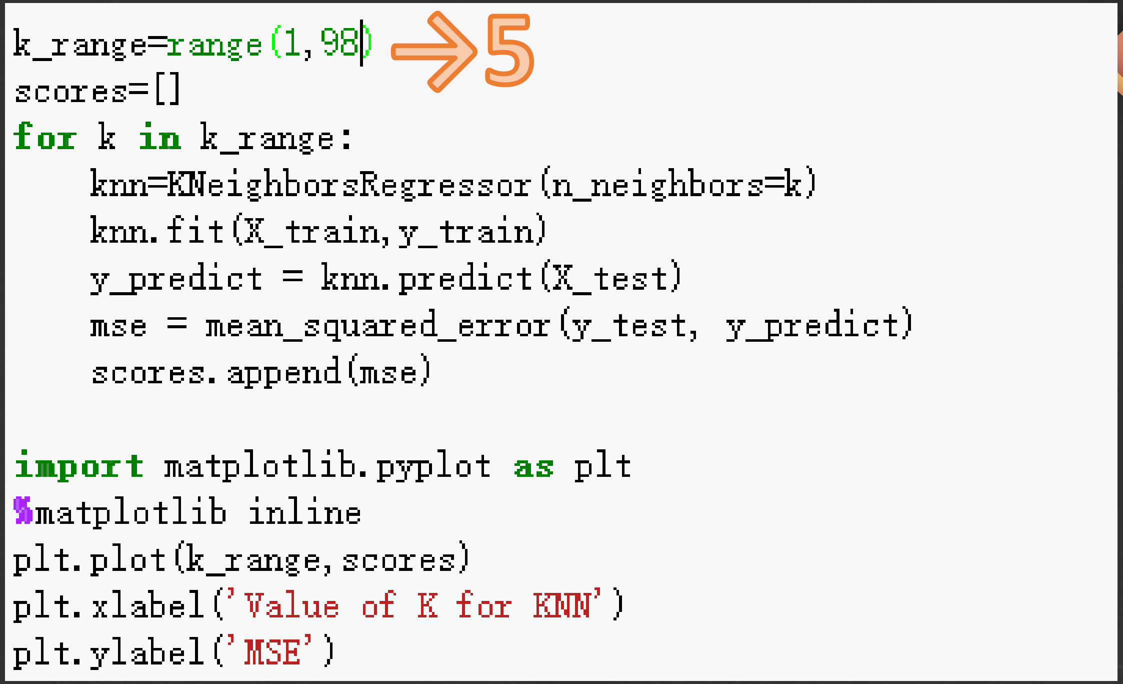


```
## step3: use trained model to predict F-score, and evaluae the model performance
print(’[3] evaluating the model...’)
# use the model to predict unseen F-score
y_predict = model_knn.predict(X_test)
print(y_predict.tolist()[:10]) # first 10 predicted data
print(y_test.tolist()[:10]) # first 10 true data
# evaluate model performance (using MSE/RMSE for regression problem)
mse = mean_squared_error(y_test, y_predict)
print(’Mean Square Error (KNN):’,mse)
print(’Root Mean Square Error (KNN):’,mse**0.5)
```

Mean Square Error (KNN): 9.92373352601, Root Mean Square Error (KNN): 3.15019579169

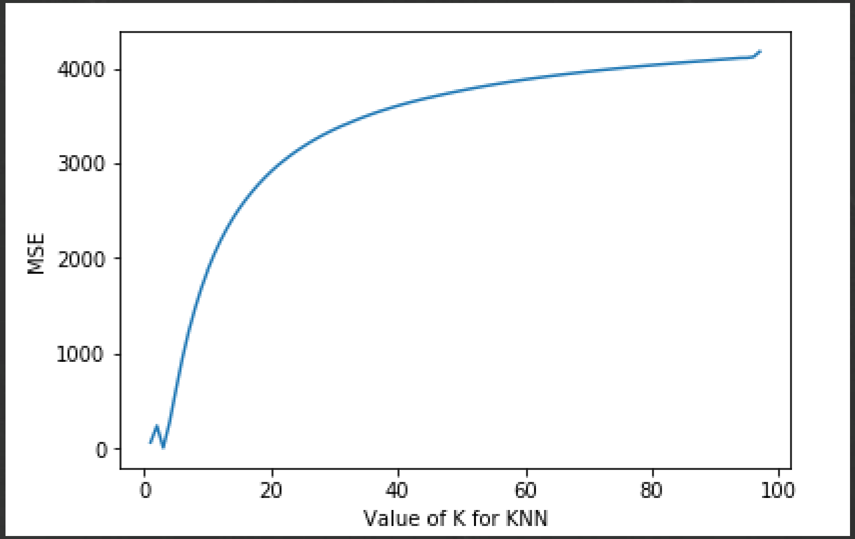
Accuracy

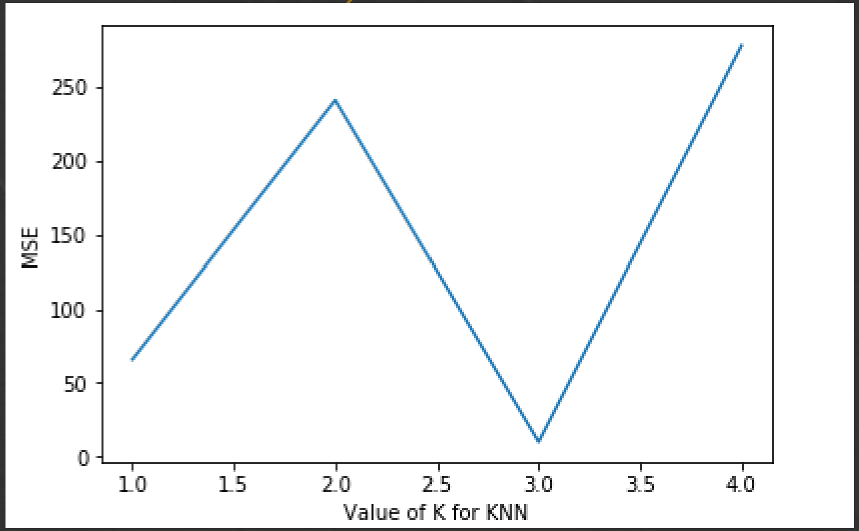
look for the highest degree of accuracy



```
k_range=range(1,98)
scores=[]
for k in k_range:
    knn=KNeighborsRegressor(n_neighbors=k)
    knn.fit(X_train,y_train)
    y_predict = knn.predict(X_test)
    mse = mean_squared_error(y_test, y_predict)
    scores.append(mse)

import matplotlib.pyplot as plt
%matplotlib inline
plt.plot(k_range,scores)
plt.xlabel(’Value of K for KNN’)
plt.ylabel(’MSE’)
```

→5





k=3 !

OUTCOME

790.9032840708696	0000042136
527.2005619089598	0001380706
332.1362294007186	0001527675
291.9217215654966	0001104904
132.56385340352097	0000029952
44.25995652844528	0001346352
43.87003891991425	0001445196
39.813375206461075	0001451264
33.10389048404004	0001650101
32.18423921312216	0001378270

Companies ranking among the top ten

04

Optimization

1.Efficiency optimization

In the first step and the third step, first go through all <a> or <font>, to find out what we need. On average, each page needs to loop for 8000-25000 times. Later changing the object to the <TR>, it is reduced to 400-800 times, greatly improving operation efficiency. By going through strings, searching and skipping the inevitable failure of the cycle in time, we reduce the times of looping. By using linear search instead of the regular expression,the program improves the efficiency of operation.

2. Debugging process optimization

In the third step, there are some failures. In the following improvements, the failed URL is saved in "errlog.txt"for following processing, which is to use multiple-nested "try-except-block" to judge the documents, use if-else block to choose corresponding processing method and meanwhile add similarities when finding <TR>.It improves the rate of success of matching. The success rate increases from 12/150 to 123/376.

3. Output optimization

By using formatted strings,it generates the unified format strings. By saving the document, we pass the CIK in the first step to the third step and output the txt document.Then we can reduce the following workload. We choose \t as separator so that we can use string.split () to make them into LIST.

OUTCOME

790.9032840708696	0000042136
527.2005619089598	0001380706
332.1362294007186	0001527675
291.9217215654966	0001104904
132.56385340352097	0000029952
44.25995652844528	0001346352
43.87003891991425	0001445196
39.813375206461075	0001451264
33.10389048404004	0001650101
32.18423921312216	0001378270

Companies ranking among the top ten

THE "WINNER":0000042136 742,455 1,041 553,00 16,030

↑

Deficiency

F-score isn't always the bigger the better. The F of small scale companies sometimes is high.

We finally choose excel to solve sequencing problem. Because the “sort”function in python may come out the results that 5 is bigger than 44.When we turn string to float for sorting,“CIK”can't move with F-score automatically,which makes it difficult to figure out which company's performance is better.

F-score is a very simple reference.Different companies of different kinds have different potential,which can't reflect in the F-score.When making investment decisions, we should also consider other factors.

MANY THANKS