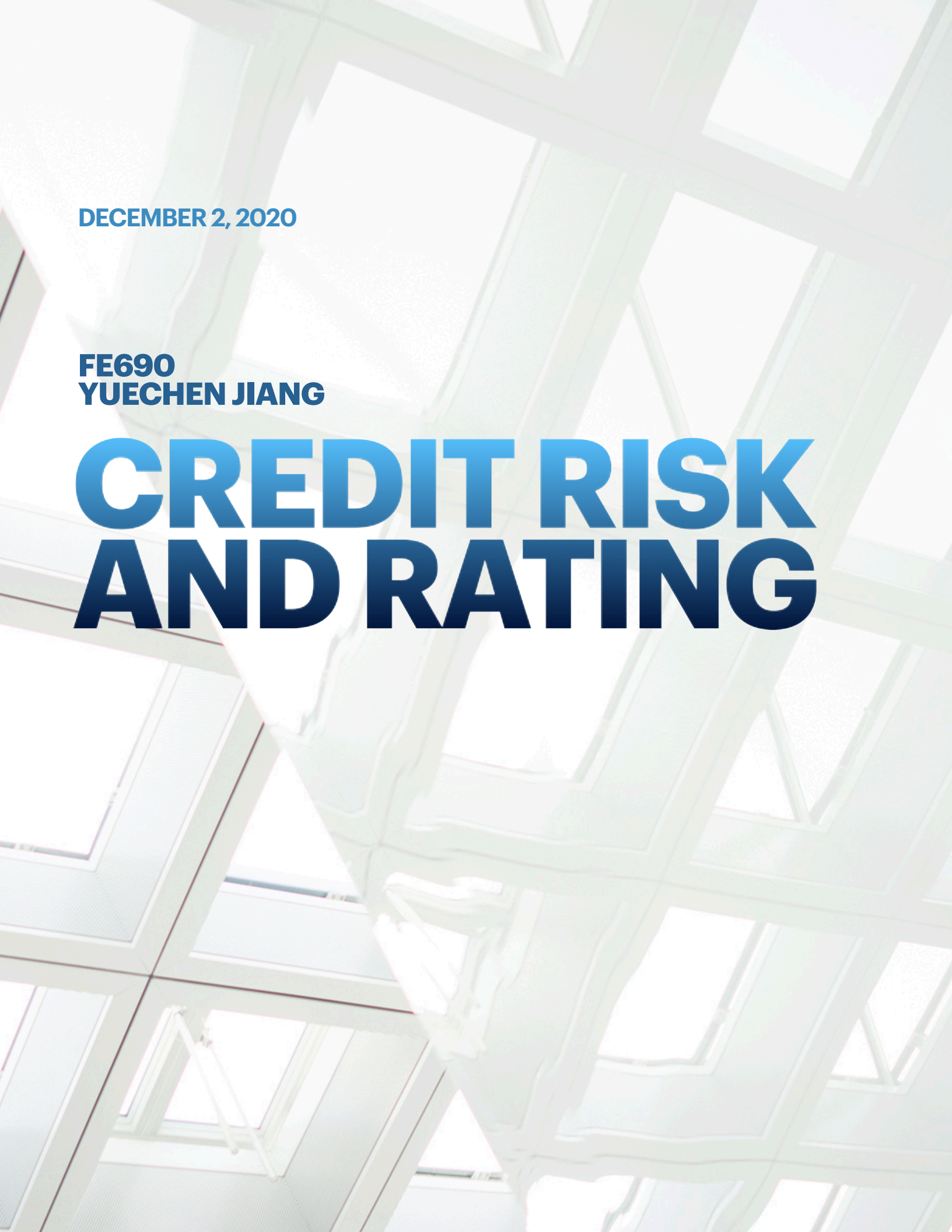
DECEMBER 2, 2020

FE690
YUECHEN JIANG

CREDIT RISK AND RATING



1. INTRODUCTION

Credit rating assessment is a complicated process in which many parameters describing a company are taken into consideration and a grade is assigned, which represents the reliability of a potential client. Such an assessment is expensive because domain experts have to be employed to perform the rating. One way of lowering the costs of performing the rating is to use an automated rating procedure.

There are many instances when the credibility of a company needs to be measured. Bond investors, debt issuers, governmental officers, and companies that provide credit use credit ratings to as sess the investment risk. These ratings are a basis for important decisions, and therefore there is a need for them to be as accurate as possible. Unfortunately, this usually means that many details of a company's profile have to be taken into consideration. Such a detailed analysis can be performed by experts, but this is often costly and time-consuming. Because manual analysis of a company's profile is slow and costly, currently significant emphasis is placed on computational methods of credit rating assessment.

2. CREDIT RATING PREDICTION

Corporate credit rating is a process in which a grade x 2 X from a predefined rating scale X is assigned to a company. Rating agencies, such as Standard & Poor's (S&P's), Moody's, and Fitch have their own rating scales. For example, the rating scale of the S&P's is X = {AAA,AA,A,BBB,BB,B,CCC,CC,C,D} – a total of 10 grades (rating classes) that are ordered from AAA, the most promising for investors, to D, the most risky one.

To use statistical methods, one has to first choose a model with a predefined structure to represent observations. Then, the parameters of the model are estimated to fit the model to the observational data. The advantage of such an approach is that the models are relatively easy to explain. However, statistical models require various assumptions to be theoretically valid.

Another approach is to use AI methods. The AI methods differ from traditional statistical methods in that they allow learning the model from data. The advantage of such an approach is that AI methods usually do not require specific assumptions on the distribution of data.

Using concepts from the machine learning paradigm, the problem of credit rating prediction can be formulated as a classification problem in which rating classes used by a particular rating agency are known in advance. A typical classification procedure is performed as supervised learning. This learning type requires a sample of companies that were initially assigned proper ratings. A classifier of a chosen type is first trained using this sample. Then, the trained classifier can be used to predict the ratings of previously unseen companies.

Neural networks (NNs) are commonly used in the literature for credit rating prediction. NNs were found to be significantly more accurate than traditional statistical methods in previous studies .

Other AI methods used for credit rating prediction include artificial immune systems (AISs) , case-based reasoning (CBR) , evolutionary algorithms, and ant colony optimization.

3. DATASETS

The parameters used to describe the company in our research can be divided into two categories: business conditions and financial indicators.

Factors such as industry risk, scale, characteristics and management skills can be used to describe the company's business conditions. Company size can be expressed by measuring market value, assets, equity, cash flow, etc. The company's ability to pay off loans depends on factors such as company size. Company size is also related to diversification and market power. The company's characteristics (reputation) are difficult to measure. To some extent, this factor can be inferred from information about internal personnel and institutional equity. Industry risk indicates the sensitivity of companies in a particular industry or market to external business factors (such as macroeconomic changes).

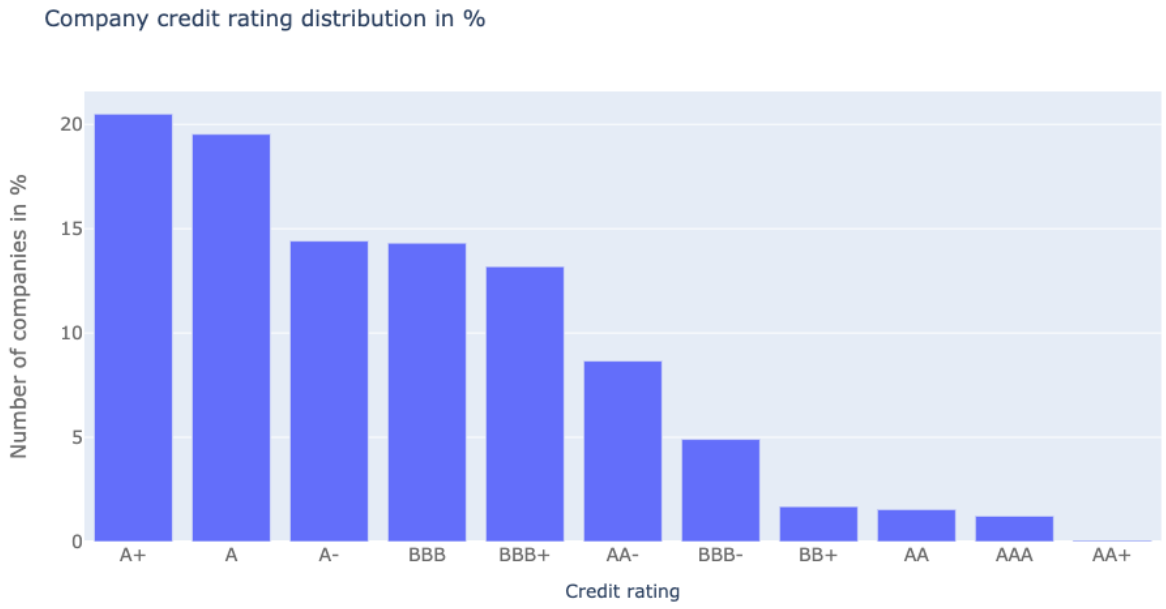
The Y train of this project is: ’S&P Domestic Long-term Issuer Credit Rating'

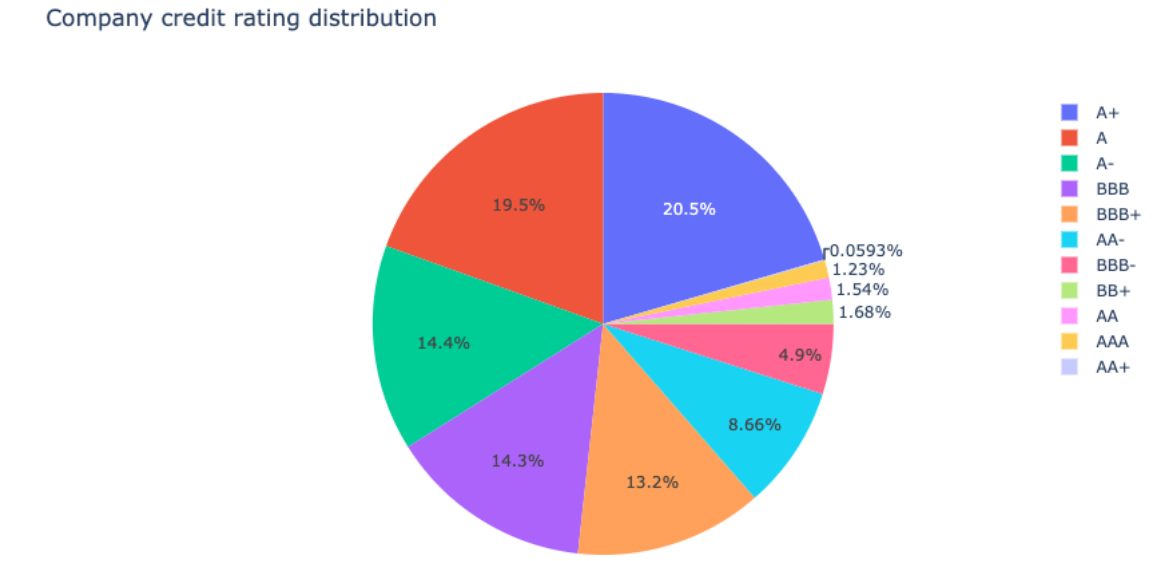
X train is selected as follows

"Schiller's cyclically adjusted P/E ratio”, "Books/Markets","Enterprise Value Multiple", "Price/ operating income (basic, excluding EI)”, "Price/operating income (after dilution, excluding EI)”, "P/E

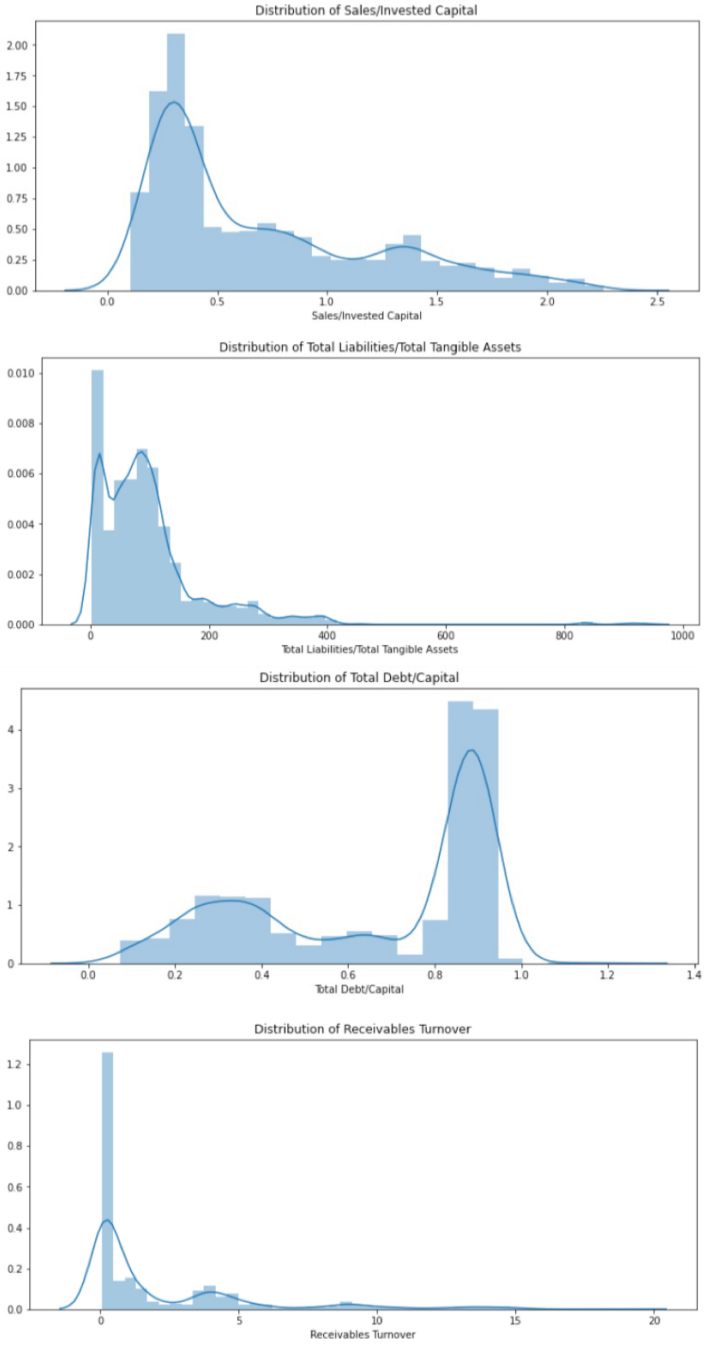
(dilution, excluding EI)", "P/E (dilution, excluding EI)”, “Price/sale", "Price/Cash Flow”, "Dividend Payout Rate”, "Net profit rate”, "Operating profit rate before depreciation”, "Operating profit margin after depreciation”, 'Gross profit margin’, "Pre-tax profit rate”, "Cash Flow Margin”, 'Return on Assets’, "Return on Equity”, "Return on Capital Used”, "Effective tax rate”, "Average after-tax income of common stock”, "Investment capital after-tax income”, "After-tax income of total shareholder equity”, "Gross Profit/Total Assets”, "Common Stock/Investment Capital”, "Long-term debt/investment capital”, "Total debt/invested capital”, "Capitalization Ratio”, "Cash Balance/Total Liabilities”, "Total debt/total assets”, "Total Debt/EBITDA”, "Short-term debt/total debt”, "Long-term debt/total debt”, "Cash Flow/ Total Debt”, "Free cash flow/operating cash flow”, "Total liabilities/total tangible assets”, "Long-term debt/book equity”, "Total debt/total assets. 1”, "Total Debt/Capital”, "Total debt/equity”, "Asset Turnover Rate”, "Accounts Receivable Turnover Rate”, "Accounts Payable Turnover Rate”, "Sales/ Investment Capital”, "Sales/Shareholders' Equity”, "Research and Development/Sales”, "Advertising Expenses/Sales”, "Labor Expenses/Sales”, "Cumulative/average assets”, “Price/Booking", "Dividend Yield”

First, I visualized the data. From this bar chart and pie chart, we can see the number and proportion of companies at each level.





Next, let us take a look at the distribution of several importance features obtained in the next section. It can be seen from the figure that although the distributions of the four indicators are somewhat similar, they do not obey the known distributions, so the AI method is a good choice.



Since the data set is relatively large, we select 20% as the training set and the remaining 80% as the test set.

4. FEATURE SELECTION AND MODEL CONSTRUCTION

In this project, a total of nine models were selected to predict the credit rating, and random forest is mainly selected for illustration.

Here is a preview of some data:

	Unnamed: 0	Global Company Key	S&P Domestic Long Term Issuer Credit Rating	Data Date	Company Name	Ticker Symbol	Shillers Cyclically Adjusted P/E Ratio	Book/Market	Enterprise Value Multiple	Price/Operating Earnings (Basic, Excl. EI)	...	Receivables Turnover	Payables Turnover	Sales/Invest Cap
0	0	1447	A+	20040229	AMERICAN EXPRESS CO	AXP	27.958	0.247	20.164	22.829	...	0.414	0.639	0.8
1	1	1447	A+	20040331	AMERICAN EXPRESS CO	AXP	27.264	0.247	20.164	22.158	...	0.414	0.639	0.8
2	2	1447	A+	20040430	AMERICAN EXPRESS CO	AXP	25.740	0.247	20.164	20.919	...	0.414	0.639	0.8
3	3	1447	A+	20040531	AMERICAN EXPRESS CO	AXP	26.623	0.237	20.164	20.444	...	0.433	0.666	0.7
4	4	1447	A+	20040630	AMERICAN EXPRESS CO	AXP	26.980	0.237	20.164	20.718	...	0.433	0.666	0.7

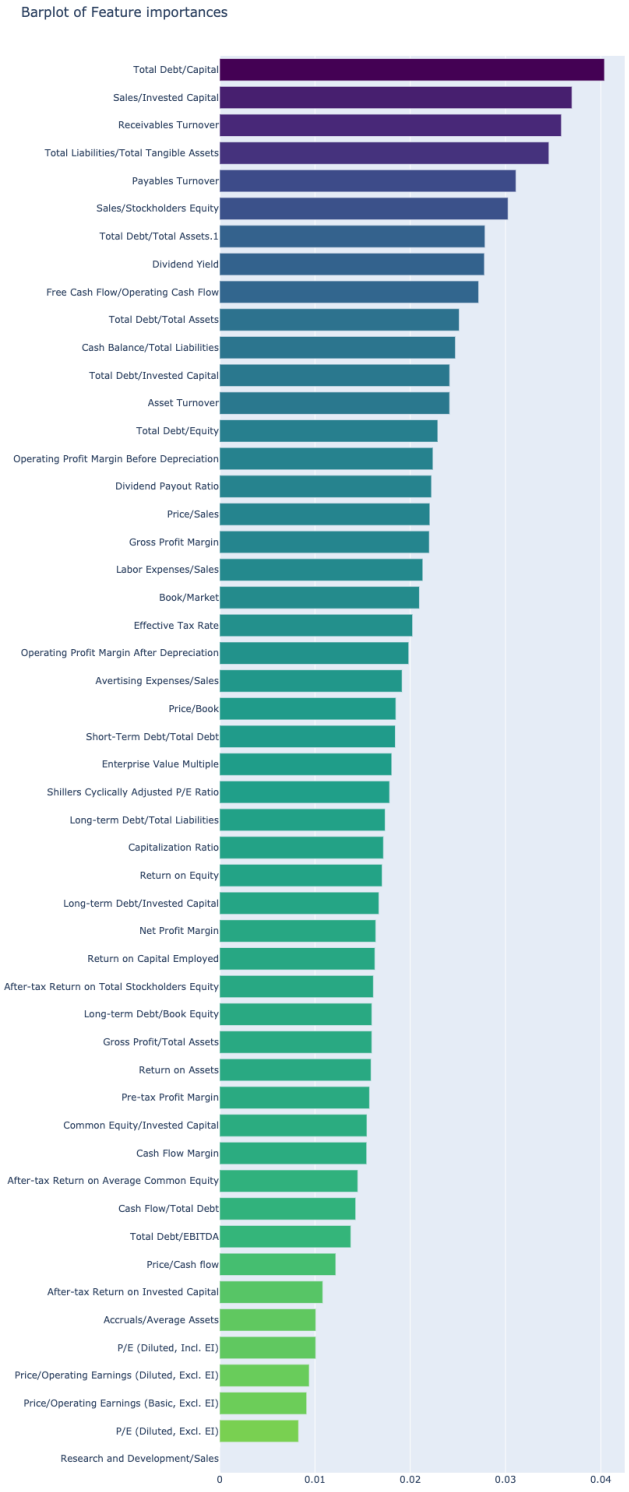
Since there is a column in the data as a string, it is now encoded as a number. In the data shown above, delete irrelevant columns, and the processed data below.

	S&P Domestic Long Term Issuer Credit Rating	Shillers Cyclically Adjusted P/E Ratio	Book/Market	Enterprise Value Multiple	Price/Operating Earnings (Basic, Excl. EI)	Price/Operating Earnings (Diluted, Excl. EI)	P/E (Diluted, Excl. EI)	P/E (Diluted, Incl. EI)	Price/Sales	Price/Cash flow	...	Receivables Turnover	Payables Turnover	Sale
0	1	27.958	0.247	20.164	22.829	23.126	23.126	23.226	2.581	27.026	...	0.414	0.639	
1	1	27.264	0.247	20.164	22.158	22.446	22.446	22.543	2.517	26.356	...	0.414	0.639	
2	1	25.740	0.247	20.164	20.919	21.190	21.190	21.283	2.376	24.882	...	0.414	0.639	
3	1	26.623	0.237	20.164	20.444	20.779	20.779	21.303	2.360	11.186	...	0.433	0.666	
4	1	26.980	0.237	20.164	20.718	21.057	21.057	21.588	2.392	11.336	...	0.433	0.666	

Extract training feature data and target values. In the data set as the "Standard & Poor's domestic long-term issuer credit rating" column. In order to test the performance of the prediction model, the data set is divided into training data set and test data set. Since the data set is very large, only 20% of the data is used as the training set. Build a predictive model. When we complete all the above operations, what we get is the training set and the test set. The model can now be built. Since this is a classification task, we choose random forest to complete it. The construction and training of the model have been completed above, now let us test the accuracy of the model. We can use the classification report method provided by sklearn for comprehensive evaluation. Here are the results of Random Forest:

	precision	recall	f1-score	support
0	0.95	0.92	0.93	809
1	0.94	0.94	0.94	849
2	0.93	0.92	0.93	593
3	0.98	0.81	0.89	78
4	0.00	0.00	0.00	0
5	0.88	0.95	0.91	334
6	1.00	1.00	1.00	48
7	1.00	0.95	0.98	66
8	0.98	0.92	0.95	615
9	0.89	0.98	0.93	475
10	0.90	0.93	0.92	181
accuracy			0.93	4048
macro avg	0.86	0.85	0.85	4048
weighted avg	0.94	0.93	0.94	4048

The following figure shows the importance of the features through the results of the model.



In the previous step, we have printed the distributions of the four variables with the highest correlation.

5. EXPERIMENTAL RESULTS

Next, we used the following nine models to build the prediction model:

Logistic regression

Random forest

Decision tree

Multilayer perceptron

Adaptive gradient boost

Bagging algorithm

Gradient Boosting Algorithm

Support Vector Machines

Naive Bayes

Here are the results of 9 models:

LogisticRegression()

	precision	recall	f1-score	support
0	0.49	0.28	0.36	1366
1	0.36	0.32	0.34	943
2	0.19	0.29	0.23	386
3	0.30	0.66	0.41	29
4	0.00	0.00	0.00	0
5	0.33	0.50	0.40	243
6	0.71	0.79	0.75	43
7	0.00	0.00	0.00	7
8	0.30	0.42	0.35	413
9	0.38	0.33	0.35	606
10	0.02	0.33	0.04	12
accuracy			0.33	4048
macro avg	0.28	0.36	0.29	4048
weighted avg	0.38	0.33	0.34	4048

RandomForestClassifier()

	precision	recall	f1-score	support
0	0.95	0.90	0.92	831
1	0.94	0.95	0.95	841
2	0.92	0.92	0.92	585
3	1.00	0.83	0.91	77
4	0.00	0.00	0.00	0
5	0.88	0.96	0.92	331
6	1.00	0.96	0.98	50
7	1.00	0.95	0.98	66
8	0.98	0.92	0.95	615
9	0.87	0.97	0.92	467
10	0.94	0.94	0.94	185
accuracy			0.93	4048
macro avg	0.86	0.85	0.85	4048
weighted avg	0.94	0.93	0.93	4048

DecisionTreeClassifier()

	precision	recall	f1-score	support
0	0.80	0.85	0.82	743
1	0.86	0.89	0.87	824
2	0.88	0.81	0.84	643
3	0.95	0.69	0.80	89
4	0.00	0.00	0.00	0
5	0.83	0.89	0.86	336
6	0.81	1.00	0.90	39
7	0.73	0.74	0.74	62
8	0.85	0.82	0.84	594
9	0.84	0.83	0.83	522
10	0.83	0.79	0.81	196
accuracy			0.84	4048
macro avg	0.76	0.75	0.75	4048
weighted avg	0.84	0.84	0.84	4048

MLPClassifier(max_iter=100)

	precision	recall	f1-score	support
0	0.52	0.61	0.56	675
1	0.54	0.58	0.56	789
2	0.43	0.46	0.44	557
3	0.55	0.36	0.43	98
4	0.00	0.00	0.00	0
5	0.54	0.55	0.54	350
6	0.71	0.56	0.62	61
7	0.68	0.63	0.66	68
8	0.64	0.61	0.63	603
9	0.50	0.38	0.43	688
10	0.49	0.57	0.53	159
accuracy			0.53	4048
macro avg	0.51	0.48	0.49	4048
weighted avg	0.53	0.53	0.53	4048

AdaBoostClassifier()

	precision	recall	f1-score	support
0	0.17	0.35	0.23	395
1	0.30	0.35	0.32	711
2	0.00	0.00	0.00	0
3	0.00	0.00	0.00	0
4	0.00	0.00	0.00	0
5	0.00	0.00	0.00	0
6	0.65	0.97	0.78	32
7	0.67	0.18	0.28	237
8	0.05	0.19	0.08	147
9	0.92	0.19	0.31	2526
10	0.00	0.00	0.00	0
accuracy			0.24	4048
macro avg	0.25	0.20	0.18	4048
weighted avg	0.69	0.24	0.30	4048

BaggingClassifier()

	precision	recall	f1-score	support
0	0.93	0.88	0.90	828
1	0.93	0.92	0.93	853
2	0.88	0.89	0.89	580
3	1.00	0.80	0.89	80
4	0.00	0.00	0.00	0
5	0.86	0.93	0.90	335
6	1.00	1.00	1.00	48
7	0.97	0.98	0.98	62
8	0.94	0.91	0.92	594
9	0.85	0.93	0.88	474
10	0.92	0.89	0.91	194
accuracy			0.91	4048
macro avg	0.84	0.83	0.84	4048
weighted avg	0.91	0.91	0.91	4048

GradientBoostingClassifier()

	precision	recall	f1-score	support
0	0.89	0.93	0.91	760
1	0.95	0.89	0.92	909
2	0.87	0.92	0.89	556
3	0.92	0.80	0.86	74
4	0.00	0.00	0.00	0
5	0.85	0.96	0.90	320
6	0.81	0.93	0.87	42
7	0.68	0.70	0.69	61
8	0.97	0.90	0.93	619
9	0.89	0.89	0.89	519
10	0.91	0.90	0.90	188
accuracy			0.90	4048
macro avg	0.79	0.80	0.80	4048
weighted avg	0.91	0.90	0.90	4048

SVC(kernel='linear')

	precision	recall	f1-score	support
0	0.69	0.56	0.62	969
1	0.67	0.66	0.66	864
2	0.44	0.54	0.48	475
3	0.91	0.67	0.77	86
4	0.00	0.00	0.00	0
5	0.63	0.78	0.69	292
6	0.77	0.90	0.83	41
7	0.98	0.70	0.82	89
8	0.74	0.69	0.72	616
9	0.52	0.68	0.59	396
10	0.62	0.53	0.57	220
accuracy			0.63	4048
macro avg	0.63	0.61	0.61	4048
weighted avg	0.65	0.63	0.64	4048

LogisticRegression()

	precision	recall	f1-score	support
0	0.49	0.28	0.36	1366
1	0.36	0.32	0.34	943
2	0.19	0.29	0.23	386
3	0.30	0.66	0.41	29
4	0.00	0.00	0.00	0
5	0.33	0.50	0.40	243
6	0.71	0.79	0.75	43
7	0.00	0.00	0.00	7
8	0.30	0.42	0.35	413
9	0.38	0.33	0.35	606
10	0.02	0.33	0.04	12
accuracy			0.33	4048
macro avg	0.28	0.36	0.29	4048
weighted avg	0.38	0.33	0.34	4048

RandomForestClassifier()

	precision	recall	f1-score	support
0	0.95	0.90	0.92	831
1	0.94	0.95	0.95	841
2	0.92	0.92	0.92	585
3	1.00	0.83	0.91	77
4	0.00	0.00	0.00	0
5	0.88	0.96	0.92	331
6	1.00	0.96	0.98	50
7	1.00	0.95	0.98	66
8	0.98	0.92	0.95	615
9	0.87	0.97	0.92	467
10	0.94	0.94	0.94	185
accuracy			0.93	4048
macro avg	0.86	0.85	0.85	4048
weighted avg	0.94	0.93	0.93	4048

GaussianNB()

	precision	recall	f1-score	support
0	0.55	0.42	0.48	1033
1	0.27	0.67	0.39	344
2	0.40	0.32	0.36	737
3	0.62	0.67	0.65	60
4	0.00	0.00	0.00	0
5	0.73	0.37	0.49	717
6	0.94	0.71	0.81	63
7	0.97	0.52	0.67	118
8	0.16	0.73	0.26	127
9	0.31	0.65	0.42	244
10	0.60	0.18	0.28	605
accuracy			0.41	4048
macro avg	0.51	0.48	0.44	4048
weighted avg	0.53	0.41	0.42	4048

The following is a comparison table of the accuracy of the nine models

	name	acc
0	LogisticRegression	0.333251
1	RandomForestClassifier	0.935524
2	DecisionTreeClassifier	0.836215
3	MLPClassifier	0.522480
4	AdaBoostClassifier	0.239130
5	BaggingClassifier	0.896245
6	GradientBoostingClassifier	0.902174
7	SVMClassifier	0.633646
8	NaiveBayesClassifier	0.414773

It can be seen from the table that the best model is Random Forest, followed by Gradient Boosting. The results of decision tree and Bagging algorithm are also considered ideal,

while SVM results are average. The results of the remaining four models are very unsatisfactory.

My project currently only does this. In addition to the inappropriateness of the model itself, it may also be due to the selection of too many low-relevance features. Next, I will select five to ten high-relevance features for each model for training and prediction again, and try to update Multiple models to find the most suitable model.

FE690HW3

December 2, 2020

[1]:
```
import pandas as pd
df = pd.read_csv('/Users/yuechenjiang/Downloads/SPRatings.csv')
df.head()
```

[1]:

	Unnamed: 0	Global Company Key	S&P Domestic Long Term Issuer Credit Rating	\
0	0	1447	A+	
1	1	1447	A+	
2	2	1447	A+	
3	3	1447	A+	
4	4	1447	A+	

	Data Date	Company Name	Ticker Symbol	\
0	20040229	AMERICAN EXPRESS CO	AXP	
1	20040331	AMERICAN EXPRESS CO	AXP	
2	20040430	AMERICAN EXPRESS CO	AXP	
3	20040531	AMERICAN EXPRESS CO	AXP	
4	20040630	AMERICAN EXPRESS CO	AXP	

	Shillers Cyclically Adjusted P/E Ratio	Book/Market	\
0	27.958	0.247	
1	27.264	0.247	
2	25.740	0.247	
3	26.623	0.237	
4	26.980	0.237	

	Enterprise Value Multiple	Price/Operating Earnings (Basic, Excl. EI)	…	\
0	20.164	22.829	…	
1	20.164	22.158	…	
2	20.164	20.919	…	
3	20.164	20.444	…	
4	20.164	20.718	…	

	Receivables Turnover	Payables Turnover	Sales/Invested Capital	\
0	0.414	0.639	0.817	
1	0.414	0.639	0.817	
2	0.414	0.639	0.817	
3	0.433	0.666	0.798	

```
4                 0.433              0.666                    0.798

   Sales/Stockholders Equity  Research and Development/Sales  \
0                      1.735                             0.0
1                      1.735                             0.0
2                      1.735                             0.0
3                      1.819                             0.0
4                      1.819                             0.0

   Avertising Expenses/Sales  Labor Expenses/Sales  Accruals/Average Assets  \
0                        0.0                 0.238                   -0.003
1                        0.0                 0.238                   -0.003
2                        0.0                 0.238                   -0.003
3                        0.0                 0.238                    0.016
4                        0.0                 0.238                    0.016

   Price/Book  Dividend Yield
0       4.476         0.00749
1       4.365         0.00771
2       4.121         0.00817
3       4.118         0.00789
4       4.173         0.00779

[5 rows x 57 columns]
```

[2]:
```
df.describe()
```

[2]:
```
         Unnamed: 0  Global Company Key     Data Date  \
count  5059.000000         5059.000000  5.059000e+03
mean   2793.627001        30018.449694  2.010032e+07
std    1640.274394        49170.150035  4.530048e+04
min       0.000000         1447.000000  2.000013e+07
25%    1274.500000         4699.000000  2.006113e+07
50%    2910.000000         9783.000000  2.011043e+07
75%    4222.500000        16245.000000  2.014053e+07
max    5543.000000       177376.000000  2.017023e+07

       Shillers Cyclically Adjusted P/E Ratio  Book/Market  \
count                             5059.000000  5059.000000
mean                                19.423288     0.738549
std                                 60.348755     0.445294
min                              -2882.600000     0.005000
25%                                 13.000000     0.416000
50%                                 17.473000     0.651000
75%                                 24.720500     0.971000
max                               1183.140000     4.879000
```

	Enterprise Value Multiple	Price/Operating Earnings (Basic, Excl. EI)	\
count	5059.000000	5059.000000	
mean	12.242197	18.921327	
std	6.973954	47.890697	
min	-69.251000	-495.670000	
25%	8.624000	9.964000	
50%	10.855000	13.079000	
75%	13.593000	17.631000	
max	103.895000	557.831000	

	Price/Operating Earnings (Diluted, Excl. EI)	P/E (Diluted, Excl. EI)	\
count	5059.000000	5059.000000	
mean	19.202093	20.998846	
std	48.843184	49.982233	
min	-495.670000	-370.880000	
25%	10.129000	10.259000	
50%	13.235000	13.541000	
75%	17.826500	18.743000	
max	575.273000	569.767000	

	P/E (Diluted, Incl. EI)	…	Receivables Turnover	Payables Turnover	\
count	5059.000000	…	5059.000000	5059.000000	
mean	20.732894	…	2.201683	4.080838	
std	50.647402	…	3.360186	8.262851	
min	-585.000000	…	0.051000	-0.287000	
25%	10.287500	…	0.119000	0.024000	
50%	13.518000	…	0.380000	0.297000	
75%	18.534500	…	3.659500	5.036000	
max	556.323000	…	18.925000	59.530000	

	Sales/Invested Capital	Sales/Stockholders Equity	\
count	5059.000000	5059.000000	
mean	0.698236	1.357206	
std	0.517942	5.607073	
min	0.106000	0.132000	
25%	0.293000	0.526000	
50%	0.465000	0.858000	
75%	1.015000	1.607000	
max	2.255000	131.505000	

	Research and Development/Sales	Avertising Expenses/Sales	\
count	5059.000000	5059.000000	
mean	0.000034	0.009615	
std	0.000519	0.014875	
min	0.000000	0.000000	
25%	0.000000	0.000000	
50%	0.000000	0.000000	

```
75%                          0.000000                   0.015000
max                          0.012000                   0.098000

       Labor Expenses/Sales  Accruals/Average Assets   Price/Book  \
count           5059.000000              5059.000000  5059.000000
mean               0.200659                 0.014907     2.260501
std                0.161612                 0.023853     3.934515
min                0.000000                -0.213000     0.121000
25%                0.000000                 0.002000     1.030500
50%                0.215000                 0.010000     1.550000
75%                0.322000                 0.024000     2.452000
max                0.696000                 0.191000    71.952000

       Dividend Yield
count     5059.000000
mean         0.020176
std          0.014652
min          0.000860
25%          0.010600
50%          0.018500
75%          0.025200
max          0.205000

[8 rows x 54 columns]
```

[3]:
```
df.shape
```

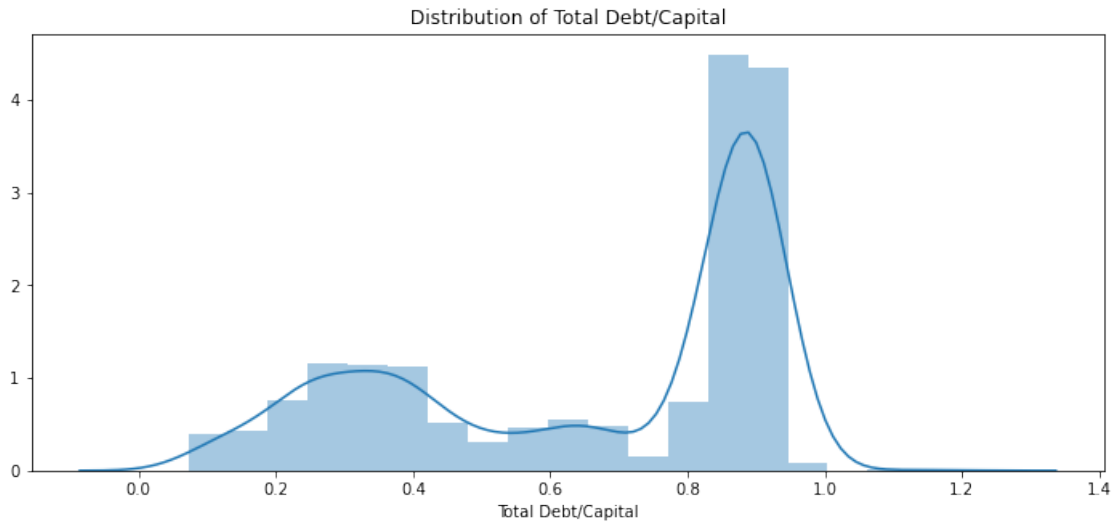
[3]:
```
(5059, 57)
```

[4]:
```
df.columns
```

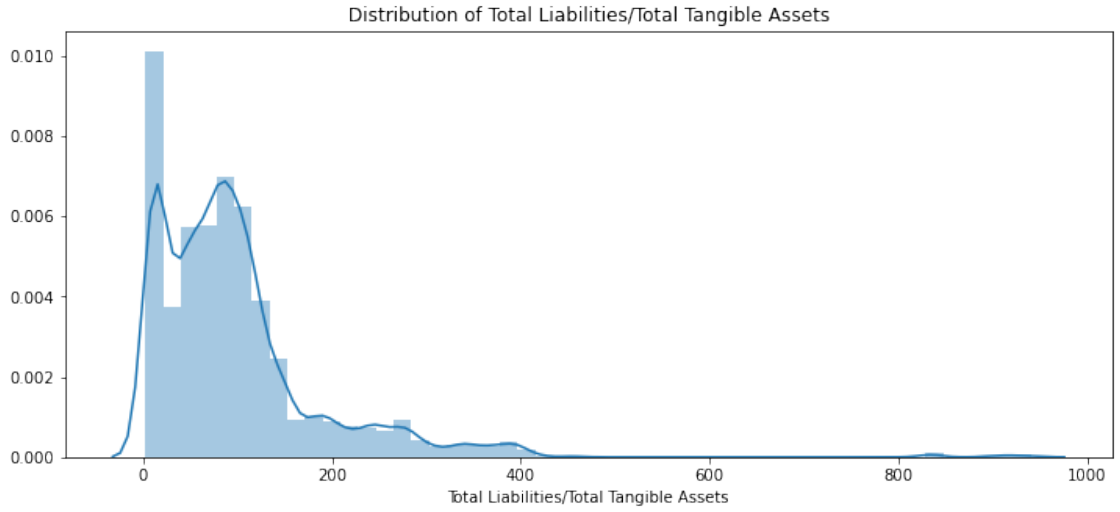
[4]:
```
Index(['Unnamed: 0', 'Global Company Key',
       'S&P Domestic Long Term Issuer Credit Rating', 'Data Date',
       'Company Name', 'Ticker Symbol',
       'Shillers Cyclically Adjusted P/E Ratio', 'Book/Market',
       'Enterprise Value Multiple',
       'Price/Operating Earnings (Basic, Excl. EI)',
       'Price/Operating Earnings (Diluted, Excl. EI)',
       'P/E (Diluted, Excl. EI)', 'P/E (Diluted, Incl. EI)', 'Price/Sales',
       'Price/Cash flow', 'Dividend Payout Ratio', 'Net Profit Margin',
       'Operating Profit Margin Before Depreciation',
       'Operating Profit Margin After Depreciation', 'Gross Profit Margin',
       'Pre-tax Profit Margin', 'Cash Flow Margin', 'Return on Assets',
       'Return on Equity', 'Return on Capital Employed', 'Effective Tax Rate',
       'After-tax Return on Average Common Equity',
       'After-tax Return on Invested Capital',
       'After-tax Return on Total Stockholders Equity',
```

```
       'Gross Profit/Total Assets', 'Common Equity/Invested Capital',
       'Long-term Debt/Invested Capital', 'Total Debt/Invested Capital',
       'Capitalization Ratio', 'Cash Balance/Total Liabilities',
       'Total Debt/Total Assets', 'Total Debt/EBITDA',
       'Short-Term Debt/Total Debt', 'Long-term Debt/Total Liabilities',
       'Cash Flow/Total Debt', 'Free Cash Flow/Operating Cash Flow',
       'Total Liabilities/Total Tangible Assets', 'Long-term Debt/Book Equity',
       'Total Debt/Total Assets.1', 'Total Debt/Capital', 'Total Debt/Equity',
       'Asset Turnover', 'Receivables Turnover', 'Payables Turnover',
       'Sales/Invested Capital', 'Sales/Stockholders Equity',
       'Research and Development/Sales', 'Avertising Expenses/Sales',
       'Labor Expenses/Sales', 'Accruals/Average Assets', 'Price/Book',
       'Dividend Yield'],
      dtype='object')
```

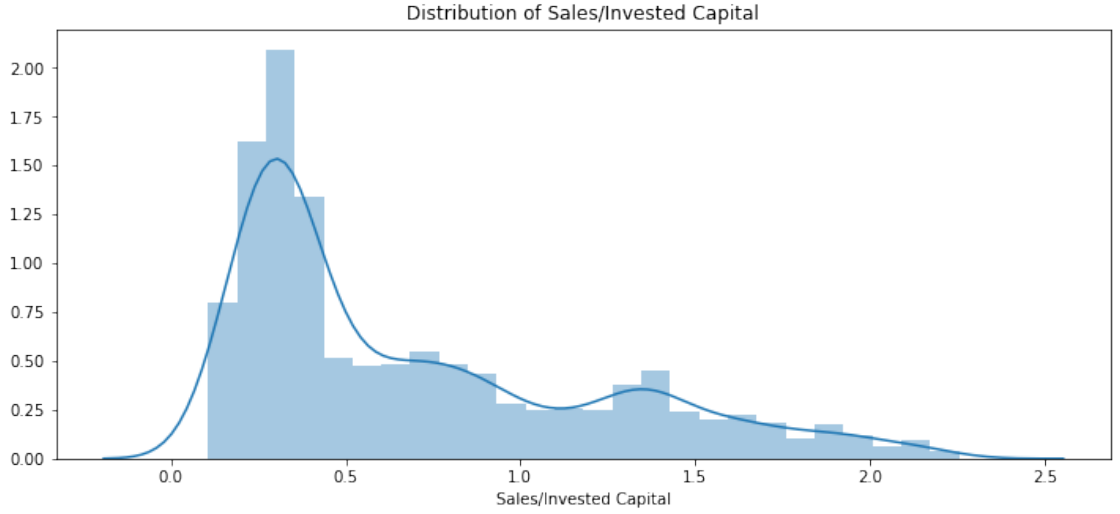
[5]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline

plt.figure(figsize=(12, 5))
plt.title("Distribution of Total Debt/Capital")
ax = sns.distplot(df["Total Debt/Capital"])
```



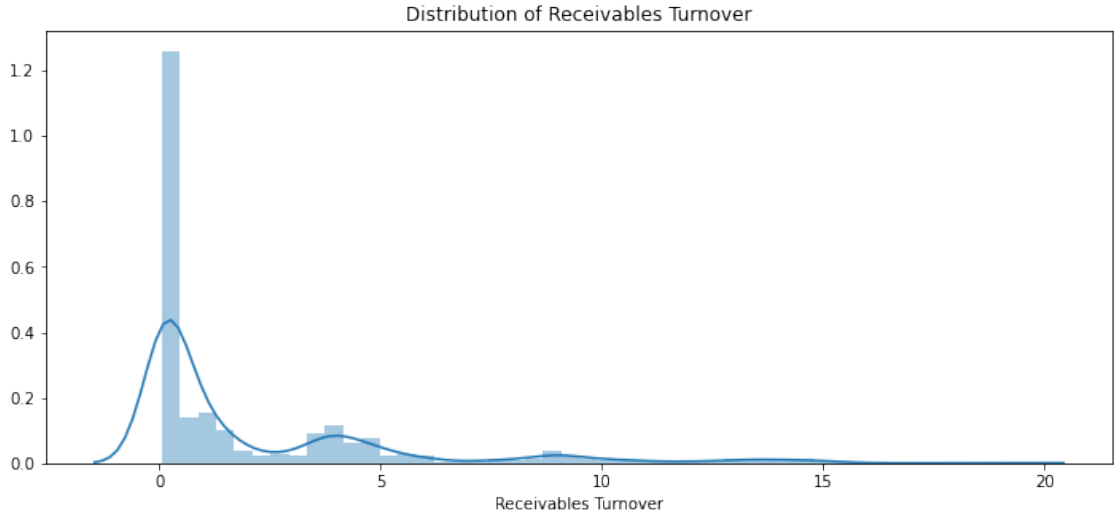
[6]:
```
plt.figure(figsize=(12, 5))
plt.title("Distribution of Total Liabilities/Total Tangible Assets")
ax = sns.distplot(df["Total Liabilities/Total Tangible Assets"])
```



[7]:
```
plt.figure(figsize=(12, 5))
plt.title("Distribution of Sales/Invested Capital")
ax = sns.distplot(df["Sales/Invested Capital"])
```



[8]:
```
plt.figure(figsize=(12, 5))
plt.title("Distribution of Receivables Turnover")
ax = sns.distplot(df["Receivables Turnover"])
```



[9]:
```
import plotly.offline as offline
import plotly.graph_objs as go
import plotly.offline as py
from plotly.offline import init_notebook_mode, iplot
init_notebook_mode(connected=True)
offline.init_notebook_mode()
```

[10]:
```
temp = df["S&P Domestic Long Term Issuer Credit Rating"].value_counts()

trace = [go.Bar(x=temp.index, y=(temp / temp.sum())*100,)]

layout = go.Layout(
    title="Who accompanied client when applying for the  application in % ",
    xaxis=dict(title='Name of type of the Suite',
               tickfont=dict(size=14, color='rgb(107, 107, 107)')),
    yaxis=dict(title='Count of Name of type of the Suite in %',
               titlefont=dict(size=16, color='rgb(107, 107, 107)'),
               tickfont=dict(size=14, color='rgb(107, 107, 107)')))

fig = go.Figure(data=trace, layout=layout)
iplot(fig, filename='schoolStateNames')
```

[11]:
```
temp = df["S&P Domestic Long Term Issuer Credit Rating"].value_counts()

trace = [go.Pie(labels=temp.index, values=temp.values)]

layout = go.Layout(title='Company credit rating distribution')

fig = go.Figure(data=trace, layout=layout)
```

```
iplot(fig)
```

[12]:
```
df_drop = df.dropna(axis=1)
df_drop.head()
```

[12]:

	Unnamed: 0	Global Company Key	S&P Domestic Long Term Issuer Credit Rating	\
0	0	1447	A+	
1	1	1447	A+	
2	2	1447	A+	
3	3	1447	A+	
4	4	1447	A+	

	Data Date	Company Name	Ticker Symbol	\
0	20040229	AMERICAN EXPRESS CO	AXP	
1	20040331	AMERICAN EXPRESS CO	AXP	
2	20040430	AMERICAN EXPRESS CO	AXP	
3	20040531	AMERICAN EXPRESS CO	AXP	
4	20040630	AMERICAN EXPRESS CO	AXP	

	Shillers Cyclically Adjusted P/E Ratio	Book/Market	\
0	27.958	0.247	
1	27.264	0.247	
2	25.740	0.247	
3	26.623	0.237	
4	26.980	0.237	

	Enterprise Value Multiple	Price/Operating Earnings (Basic, Excl. EI)	…	\
0	20.164	22.829	…	
1	20.164	22.158	…	
2	20.164	20.919	…	
3	20.164	20.444	…	
4	20.164	20.718	…	

	Receivables Turnover	Payables Turnover	Sales/Invested Capital	\
0	0.414	0.639	0.817	
1	0.414	0.639	0.817	
2	0.414	0.639	0.817	
3	0.433	0.666	0.798	
4	0.433	0.666	0.798	

	Sales/Stockholders Equity	Research and Development/Sales	\
0	1.735	0.0	
1	1.735	0.0	
2	1.735	0.0	
3	1.819	0.0	
4	1.819	0.0	

```
   Avertising Expenses/Sales  Labor Expenses/Sales  Accruals/Average Assets  \
0                        0.0                 0.238                   -0.003
1                        0.0                 0.238                   -0.003
2                        0.0                 0.238                   -0.003
3                        0.0                 0.238                    0.016
4                        0.0                 0.238                    0.016

   Price/Book  Dividend Yield
0       4.476         0.00749
1       4.365         0.00771
2       4.121         0.00817
3       4.118         0.00789
4       4.173         0.00779

[5 rows x 57 columns]
```

[13]:
```
from sklearn import preprocessing
# Remove non-numeric columns
categorical_feats = [
    f for f in df_drop.columns if df_drop[f].dtype == 'object'
]
# Encode non-numeric columns
for col in categorical_feats:
    lb = preprocessing.LabelEncoder()
    lb.fit(list(df_drop[col].values.astype('str')))
    df_drop[col] = lb.transform(list(df_drop[col].values.astype('str')))

# View coding results
df_drop.head()
```

[13]:
```
   Unnamed: 0  Global Company Key  \
0           0                1447
1           1                1447
2           2                1447
3           3                1447
4           4                1447

   S&P Domestic Long Term Issuer Credit Rating  Data Date  Company Name  \
0                                            1   20040229             3
1                                            1   20040331             3
2                                            1   20040430             3
3                                            1   20040531             3
4                                            1   20040630             3

   Ticker Symbol  Shillers Cyclically Adjusted P/E Ratio  Book/Market  \
0              6                                  27.958        0.247
1              6                                  27.264        0.247
```

```
2              6                                  25.740        0.247
3              6                                  26.623        0.237
4              6                                  26.980        0.237

   Enterprise Value Multiple  Price/Operating Earnings (Basic, Excl. EI)  …  \
0                     20.164                                      22.829  …
1                     20.164                                      22.158  …
2                     20.164                                      20.919  …
3                     20.164                                      20.444  …
4                     20.164                                      20.718  …

   Receivables Turnover  Payables Turnover  Sales/Invested Capital  \
0                 0.414              0.639                   0.817
1                 0.414              0.639                   0.817
2                 0.414              0.639                   0.817
3                 0.433              0.666                   0.798
4                 0.433              0.666                   0.798

   Sales/Stockholders Equity  Research and Development/Sales  \
0                      1.735                             0.0
1                      1.735                             0.0
2                      1.735                             0.0
3                      1.819                             0.0
4                      1.819                             0.0

   Avertising Expenses/Sales  Labor Expenses/Sales  Accruals/Average Assets  \
0                        0.0                 0.238                   -0.003
1                        0.0                 0.238                   -0.003
2                        0.0                 0.238                   -0.003
3                        0.0                 0.238                    0.016
4                        0.0                 0.238                    0.016

   Price/Book  Dividend Yield
0       4.476         0.00749
1       4.365         0.00771
2       4.121         0.00817
3       4.118         0.00789
4       4.173         0.00779

[5 rows x 57 columns]
```

[14]:
```python
# Partition data
# Remove irrelevant columns
df_drop1 = df_drop.drop("Unnamed: 0", axis=1)
df_drop1 = df_drop1.drop("Global Company Key", axis=1)
df_drop1 = df_drop1.drop("Data Date", axis=1)
df_drop1 = df_drop1.drop("Company Name", axis=1)
```

```
df_drop1 = df_drop1.drop("Ticker Symbol", axis=1)
df_drop1.head()
```

[14]:

```
   S&P Domestic Long Term Issuer Credit Rating  \
0                                            1
1                                            1
2                                            1
3                                            1
4                                            1

   Shillers Cyclically Adjusted P/E Ratio  Book/Market  \
0                                  27.958        0.247
1                                  27.264        0.247
2                                  25.740        0.247
3                                  26.623        0.237
4                                  26.980        0.237

   Enterprise Value Multiple  Price/Operating Earnings (Basic, Excl. EI)  \
0                     20.164                                      22.829
1                     20.164                                      22.158
2                     20.164                                      20.919
3                     20.164                                      20.444
4                     20.164                                      20.718

   Price/Operating Earnings (Diluted, Excl. EI)  P/E (Diluted, Excl. EI)  \
0                                        23.126                   23.126
1                                        22.446                   22.446
2                                        21.190                   21.190
3                                        20.779                   20.779
4                                        21.057                   21.057

   P/E (Diluted, Incl. EI)  Price/Sales  Price/Cash flow  …  \
0                   23.226        2.581           27.026  …
1                   22.543        2.517           26.356  …
2                   21.283        2.376           24.882  …
3                   21.303        2.360           11.186  …
4                   21.588        2.392           11.336  …

   Receivables Turnover  Payables Turnover  Sales/Invested Capital  \
0                 0.414              0.639                   0.817
1                 0.414              0.639                   0.817
2                 0.414              0.639                   0.817
3                 0.433              0.666                   0.798
4                 0.433              0.666                   0.798

   Sales/Stockholders Equity  Research and Development/Sales  \
0                      1.735                             0.0
```

```
1                      1.735                                0.0
2                      1.735                                0.0
3                      1.819                                0.0
4                      1.819                                0.0

   Avertising Expenses/Sales  Labor Expenses/Sales  Accruals/Average Assets  \
0                        0.0                 0.238                   -0.003
1                        0.0                 0.238                   -0.003
2                        0.0                 0.238                   -0.003
3                        0.0                 0.238                    0.016
4                        0.0                 0.238                    0.016

   Price/Book  Dividend Yield
0       4.476         0.00749
1       4.365         0.00771
2       4.121         0.00817
3       4.118         0.00789
4       4.173         0.00779

[5 rows x 52 columns]
```

[15]:
```
# Extract training feature data and target values. The target value here is the␣
 ↪applicant's ability to repay,
# in the data set as the "S&P Domestic Long Term Issuer Credit Rating" column.
data_X = df_drop1.drop("S&P Domestic Long Term Issuer Credit Rating", axis=1)
data_y = df_drop1['S&P Domestic Long Term Issuer Credit Rating']

# In order to test the performance of the prediction model,
# the data set is divided into training data set and test data set.
# Because the data set is large, only 20% of the data is taken as the training␣
 ↪set.
from sklearn import model_selection

train_x, test_x, train_y, test_y = model_selection.train_test_split(data_X.
 ↪values,
                                                                    data_y.
 ↪values,
                                                                    test_size=0.
 ↪8,
                                                                    ␣
 ↪random_state=0)
```

[16]:
```
# Build a predictive model
# When we complete all the above operations,
# what we get is a training set and a test set. The model can now be built.
# Since this is a classification task, we choose random forest to complete it.
from sklearn.ensemble import RandomForestClassifier
```

```
model = RandomForestClassifier()  # Build model
model.fit(train_x, train_y)  # Training model
# The construction and training of the model have been completed above,
# now let's test the accuracy of the model.
from sklearn import metrics
y_pred = model.predict(test_x)  # Forecast test set
metrics.accuracy_score(y_pred, test_y)  # Evaluation of forecast results
```

[16]:
```
0.9347826086956522
```

[17]:
```
# We can use the classification report method provided by sklaern to get a␣
 ↪comprehensive assessment.

print(metrics.classification_report(y_pred, test_y))
```

```
              precision    recall  f1-score   support

           0       0.95      0.92      0.93       809
           1       0.94      0.94      0.94       849
           2       0.93      0.92      0.93       593
           3       0.98      0.81      0.89        78
           4       0.00      0.00      0.00         0
           5       0.88      0.95      0.91       334
           6       1.00      1.00      1.00        48
           7       1.00      0.95      0.98        66
           8       0.98      0.92      0.95       615
           9       0.89      0.98      0.93       475
          10       0.90      0.93      0.92       181

    accuracy                           0.93      4048
   macro avg       0.86      0.85      0.85      4048
weighted avg       0.94      0.93      0.94      4048
```

[18]:
```
# Obtain the importance of features through the results of the model.
features = data_X.columns.values  # Take out the column name in the data set,␣
 ↪that is, the feature name
```

[19]:
```
# Get features and their importance
x, y = (list(x) for x in zip(*sorted(zip(model.feature_importances_, features),␣
 ↪reverse=False)))
```

[20]:
```
# Draw a histogram
trace2 = go.Bar(x=x, y=y, marker=dict(color=x, colorscale='Viridis',␣
 ↪reversescale=True),
                name='Random Forest Feature importance', orientation='h',)
# Set figure title, font, etc.
```

```
layout = dict(title='Barplot of Feature importances', width=900, height=2000,
              yaxis=dict(showgrid=False, showline=False, showticklabels=True,),␣
 ↪margin=dict(l=300,))
# Display graphics
fig1 = go.Figure(data=[trace2])
fig1['layout'].update(layout)
iplot(fig1, filename='plots')
```

[21]:
```
# As can be seen from the above results, different features have different␣
 ↪importance.

# Above we mainly use logistic regression to build predictive models.
# Of course, there are many Chinese methods. Now try other methods.

from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB

# Build 7 algorithms
models = [LogisticRegression(solver='lbfgs'),         # Logistic regression
          RandomForestClassifier(n_estimators=100),  # Random forest
          DecisionTreeClassifier(),                   # Decision tree
          MLPClassifier(max_iter=100),                # Multilayer perceptron
          AdaBoostClassifier(),                       # Adaptive gradient boost
          BaggingClassifier(),                        # Bagging algorithm
          GradientBoostingClassifier(),               # Gradient Boosting␣
 ↪Algorithm
          SVC(kernel = 'linear'),
          GaussianNB()]

model_name = ['LogisticRegression',
              'RandomForestClassifier',
              'DecisionTreeClassifier',
              'MLPClassifier',
              'AdaBoostClassifier',
              'BaggingClassifier',
              'GradientBoostingClassifier',
              'SVMClassifier',
              'NaiveBayesClassifier']

acc = []        #
```

```python
# f1 = []          #     f1
# recall = []      #

for model in models:  #
    model.fit(train_x, train_y)
    acc.append(model.score(test_x, test_y))
    y_pred = model.predict(test_x)
    #f = metrics.f1_score(y_pred, test_y)
    #f1.append(f)
    #recall.append(metrics.recall_score(y_pred, test_y))
    print(metrics.classification_report(y_pred, test_y))
```

```
              precision    recall  f1-score   support

           0       0.49      0.28      0.36      1366
           1       0.36      0.32      0.34       943
           2       0.19      0.29      0.23       386
           3       0.30      0.66      0.41        29
           4       0.00      0.00      0.00         0
           5       0.33      0.50      0.40       243
           6       0.71      0.79      0.75        43
           7       0.00      0.00      0.00         7
           8       0.30      0.42      0.35       413
           9       0.38      0.33      0.35       606
          10       0.02      0.33      0.04        12

    accuracy                           0.33      4048
   macro avg       0.28      0.36      0.29      4048
weighted avg       0.38      0.33      0.34      4048

              precision    recall  f1-score   support

           0       0.94      0.93      0.93       801
           1       0.96      0.95      0.96       859
           2       0.92      0.94      0.93       577
           3       1.00      0.80      0.89        80
           4       0.00      0.00      0.00         0
           5       0.87      0.96      0.91       327
           6       1.00      0.96      0.98        50
           7       1.00      0.95      0.98        66
           8       0.98      0.91      0.94       621
           9       0.89      0.95      0.92       484
          10       0.91      0.92      0.92       183

    accuracy                           0.94      4048
   macro avg       0.86      0.84      0.85      4048
```

weighted avg	0.94	0.94	0.94	4048

	precision	recall	f1-score	support
0	0.80	0.84	0.82	751
1	0.86	0.93	0.89	789
2	0.86	0.78	0.82	643
3	0.98	0.74	0.85	85
4	0.00	0.00	0.00	0
5	0.83	0.84	0.84	358
6	0.81	0.93	0.87	42
7	0.71	0.85	0.78	53
8	0.86	0.84	0.85	594
9	0.81	0.79	0.80	530
10	0.83	0.76	0.79	203
accuracy			0.84	4048
macro avg	0.76	0.75	0.75	4048
weighted avg	0.84	0.84	0.84	4048

	precision	recall	f1-score	support
0	0.74	0.42	0.54	1378
1	0.50	0.62	0.55	690
2	0.21	0.52	0.29	234
3	0.33	0.60	0.42	35
4	0.00	0.00	0.00	0
5	0.43	0.63	0.51	251
6	0.65	0.57	0.61	54
7	0.73	0.70	0.71	66
8	0.70	0.56	0.62	731
9	0.47	0.52	0.49	475
10	0.42	0.58	0.49	134
accuracy			0.52	4048
macro avg	0.47	0.52	0.48	4048
weighted avg	0.60	0.52	0.54	4048

	precision	recall	f1-score	support
0	0.17	0.35	0.23	395
1	0.30	0.35	0.32	711
2	0.00	0.00	0.00	0
3	0.00	0.00	0.00	0
4	0.00	0.00	0.00	0
5	0.00	0.00	0.00	0
6	0.65	0.97	0.78	32
7	0.67	0.18	0.28	237

8	0.05	0.19	0.08	147
9	0.92	0.19	0.31	2526
10	0.00	0.00	0.00	0
accuracy			0.24	4048
macro avg	0.25	0.20	0.18	4048
weighted avg	0.69	0.24	0.30	4048

	precision	recall	f1-score	support
0	0.87	0.87	0.87	789
1	0.92	0.89	0.91	873
2	0.87	0.90	0.89	566
3	0.95	0.79	0.87	77
4	0.00	0.00	0.00	0
5	0.82	0.93	0.87	317
6	1.00	1.00	1.00	48
7	1.00	0.95	0.98	66
8	0.96	0.89	0.92	618
9	0.89	0.90	0.89	518
10	0.88	0.93	0.91	176
accuracy			0.90	4048
macro avg	0.83	0.82	0.83	4048
weighted avg	0.90	0.90	0.90	4048

	precision	recall	f1-score	support
0	0.90	0.93	0.91	760
1	0.95	0.88	0.91	916
2	0.86	0.92	0.89	551
3	0.92	0.80	0.86	74
4	0.00	0.00	0.00	0
5	0.85	0.96	0.90	320
6	0.81	0.93	0.87	42
7	0.68	0.70	0.69	61
8	0.97	0.90	0.93	623
9	0.88	0.89	0.89	516
10	0.89	0.90	0.89	185
accuracy			0.90	4048
macro avg	0.79	0.80	0.79	4048
weighted avg	0.91	0.90	0.90	4048

	precision	recall	f1-score	support
0	0.69	0.56	0.62	969
1	0.67	0.66	0.66	864

```
              2       0.44      0.54      0.48       475
              3       0.91      0.67      0.77        86
              4       0.00      0.00      0.00         0
              5       0.63      0.78      0.69       292
              6       0.77      0.90      0.83        41
              7       0.98      0.70      0.82        89
              8       0.74      0.69      0.72       616
              9       0.52      0.68      0.59       396
             10       0.62      0.53      0.57       220

    accuracy                           0.63      4048
   macro avg       0.63      0.61      0.61      4048
weighted avg       0.65      0.63      0.64      4048

              precision    recall  f1-score   support

              0       0.55      0.42      0.48      1033
              1       0.27      0.67      0.39       344
              2       0.40      0.32      0.36       737
              3       0.62      0.67      0.65        60
              4       0.00      0.00      0.00         0
              5       0.73      0.37      0.49       717
              6       0.94      0.71      0.81        63
              7       0.97      0.52      0.67       118
              8       0.16      0.73      0.26       127
              9       0.31      0.65      0.42       244
             10       0.60      0.18      0.28       605

    accuracy                           0.41      4048
   macro avg       0.51      0.48      0.44      4048
weighted avg       0.53      0.41      0.42      4048
```

[22]:
```
#
pd.DataFrame({"name": model_name, "acc": acc})
```

[22]:

	name	acc
0	LogisticRegression	0.333251
1	RandomForestClassifier	0.935524
2	DecisionTreeClassifier	0.836215
3	MLPClassifier	0.522480
4	AdaBoostClassifier	0.239130
5	BaggingClassifier	0.896245
6	GradientBoostingClassifier	0.902174
7	SVMClassifier	0.633646
8	NaiveBayesClassifier	0.414773