Case Study 2: Decision Tree in Credit Risk Management

——Ref: Explainable Machine Learning in Credit Risk Management

Reference:
[1] Bussmann, N., Giudici, P., Marinelli, D., & Papenbrock, J. (2021) Explainable machine learning in credit risk management. *Computational Economics*, **57**, 203-216.

Obejctives of the analysis

In this case study, we apply decision tree method to predict the credit risk of a large sample of small and medium enterprises.

The reason for making this case study is that the Artificial Intelligence model can be used in credit risk management, particularly in measuring the risks that arise when credit is borrowed by employing peer-to-peer lending platforms.

The empirical analysis of about 15,000 small and medium companies asking for credit reveals that both risky and not risky borrowers (Y) can be grouped according to a set of similar financial characteristics (X), which can be employed to explain their credit score and, therefore, to predict their future behaviour.

Desciption of the data

There are two sheets in our xlsx file.

First, let's load the first sheet of the xlsx file.

In [1]:
```
import pandas as pd
```

In [2]:
```
#Load the first sheet of xlsx file
data = pd.read_excel('/Users/ranjiang/Desktop/caseStudy2_DT/data/fi
```

- Tips: seprate operation functions and showing functions (data.head()) in different cells, since we may implement our operation functions more than one time by accident.

In [3]:
```
data.head()
```

Out[3]:

	Unnamed: 0	status	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	...	V_15	V_16
0	2	0	3.30	0.62	1.30	0.96	0.70	0.94	-21.75	-0.12	...	-0.01	1
1	3	0	-7.93	-0.19	0.87	1.35	1.34	225.95	-42.44	-0.59	...	-0.10	1
2	7	0	1.24	0.41	1.81	1.14	0.65	1.10	-10.84	-0.08	...	0.00	1
3	10	0	-1.16	-0.68	0.14	1.60	0.87	2.23	-16.29	-0.27	...	-0.07	1
4	12	0	-1.90	-0.86	0.47	0.42	0.42	-1.78	-13.83	-0.24	...	-0.17	1

5 rows × 26 columns

In [4]:
```
# See the dimension of our data
data.shape
```

Out[4]:
```
(15045, 26)
```

Sheet 1 is our main data.

- The sample size: 15045
- The number of columns: 25
- The first column: since it's cleaned data, the first column is a useless ID.
- V_1 to V_24: variables/features X, float
- Status (label Y): only 2 states: 0 and 1. 1 is defaulted, 0 is active, int

Then let's see the second sheet.

In [5]:
```
# Load the second sheet of the xlsx file
name = pd.read_excel('/Users/ranjiang/Desktop/caseStudy2_DT/data/fi
                     sheet_name = "sheet2")
df_name = pd.DataFrame(name)
```

In [6]:
```
df_name.head()
```

Out[6]:

	ID	FORMULA
0	V_1	(Total assets - Shareholders Funds)/Sharehold...
1	V_2	(Long term debt + Loans)/Shareholders Funds
2	V_3	Total assets/Total liabilties
3	V_4	Current assets/Current liabilties
4	V_5	(Current assets - Current assets: stocks)/Cur...

The second sheet shows the name of variablese V_1 to V_24

Exploratorty data analysis

In [7]:
```
# Remove the first column (Name column)
data = data.drop(data.columns[0], axis = 1)
df = pd.DataFrame(data)
```

In [8]:
```
df.head()
```

Out[8]:

	status	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9	...	V_15	V_16	V_17
0	0	3.30	0.62	1.30	0.96	0.70	0.94	-21.75	-0.12	-0.53	...	-0.01	1	1
1	0	-7.93	-0.19	0.87	1.35	1.34	225.95	-42.44	-0.59	-8.12	...	-0.10	1	1
2	0	1.24	0.41	1.81	1.14	0.65	1.10	-10.84	-0.08	-0.21	...	0.00	1	1
3	0	-1.16	-0.68	0.14	1.60	0.87	2.23	-16.29	-0.27	-0.05	...	-0.07	1	1
4	0	-1.90	-0.86	0.47	0.42	0.42	-1.78	-13.83	-0.24	-0.23	...	-0.17	1	1

5 rows × 25 columns

In [9]:
```
# Convert binary numbers to class
categories = {
    0: 'active',
    1: 'defaulted'
} #categories is created in data structure of dictionary so as to u
```

In [10]:
```
#df["status"] = sta.apply(lambda x:categories [str (sta[x])])

df['status'] = df['status'].map(categories)  # a = a+1, overwrite
```

- Tips: mapping function is useful if we want to convert numebrs/names to names/numbers.

In [11]:
```
df.head()
```

Out[11]:

	status	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9	...	V_15	V_16	V_17
0	active	3.30	0.62	1.30	0.96	0.70	0.94	-21.75	-0.12	-0.53	...	-0.01	1	1
1	active	-7.93	-0.19	0.87	1.35	1.34	225.95	-42.44	-0.59	-8.12	...	-0.10	1	1
2	active	1.24	0.41	1.81	1.14	0.65	1.10	-10.84	-0.08	-0.21	...	0.00	1	1
3	active	-1.16	-0.68	0.14	1.60	0.87	2.23	-16.29	-0.27	-0.05	...	-0.07	1	1
4	active	-1.90	-0.86	0.47	0.42	0.42	-1.78	-13.83	-0.24	-0.23	...	-0.17	1	1

5 rows × 25 columns

In [12]:
```
# Add readableColNames to the df_name
df_name['readableColNames'] = ["total assets divided by shareholder
                      "total Assets to total Liabilities", "Current
                      "Shareholders Funds plus Non current liabiliti
                      "profit or loss after tax/divided by sharehold
                      "sales divided by total assets", "interest pai
                      "EBITDA to Interest Coverage Ratio", "EBITDA t
                      "constraint EBIT", "constraint profit and loss
                      "constraint profit and losses for period in th
                      "trade receivables divided by operating revenu
                      "Industry classification on NACE code, 4 digit
```

In [13]:
```
df_name.head()
```

Out[13]:

	ID	FORMULA	readableColNames
0	V_1	(Total assets - Shareholders Funds)/Sharehold...	total assets divided by shareholder funds minus 1
1	V_2	(Long term debt + Loans)/Shareholders Funds	long term debt plus loans then divided by shar...
2	V_3	Total assets/Total liabilties	total Assets to total Liabilities
3	V_4	Current assets/Current liabilties	Current Ratio

3	V_4	Current assets/Current liabilties	Current Ratio
4	V_5	(Current assets - Current assets: stocks)/Cur...	current assets without stocks divided by curre...

In [14]:
```
pd.set_option('expand_frame_repr', False)
df_name
```

Out[14]:

	ID	FORMULA	readableColNames
0	V_1	(Total assets - Shareholders Funds)/Sharehold...	total assets divided by shareholder funds minus 1
1	V_2	(Long term debt + Loans)/Shareholders Funds	long term debt plus loans then divided by shar...
2	V_3	Total assets/Total liabilties	total Assets to total Liabilities
3	V_4	Current assets/Current liabilties	Current Ratio
4	V_5	(Current assets - Current assets: stocks)/Cur...	current assets without stocks divided by curre...
5	V_6	(Shareholders Funds + Non current liabilities...	Shareholders Funds plus Non current liabilitie...
6	V_7	EBIT/interest paid	EBIT to interest paid
7	V_8	(Profit (loss) before tax + Interest paid)/To...	profit before taxes plus interest paid then di...
8	V_9	P/L after tax/Shareholders Funds	profit or loss after tax/divided by shareholde...
9	V_10	Operating revenues/Total assets	operating revenue divided by total assets
10	V_11	Sales/Total assets	sales divided by total assets
11	V_12	Interest paid/(Profit before taxes + Interest...	interest paid divided by profit before tax plu...
12	V_13	EBITDA/interest paid	EBITDA to Interest Coverage Ratio
13	V_14	EBITDA/Operating revenues	EBITDA to operating revenue ratio
14	V_15	EBITDA/Sales	EBITDA to sales
15	V_16	Constraint EBIT	constraint EBIT
16	V_17	Constraint PL before tax	constraint profit and losses before tax
17	V_18	Constraint Financial PL	constraint financial profit and losses
18	V_19	Constraint P/L for period th EUR	constraint profit and losses for period in th EUR
19	V_20	Trade Payables/Operating revenues	trade payables divided by operating revenues
20	V_21	Trade Receivables/Operating revenues	trade receivables divided by operating revenues
21	V_22	Inventories/Operating revenues	inventories divided by operating revenues
22	V_23	Total Revenue	total revenue
23	V_24	Industry classification on NACE code, 4 digits...	Industry classification on NACE code, 4 digits...

In [15]:
```
df.columns = ['status']+ df_name.loc[:,'readableColNames'].tolist()
# list operation ['a'] + ['b'] -> ['a', 'b']
```

In [16]:
```
df.head()
```

Out[16]:

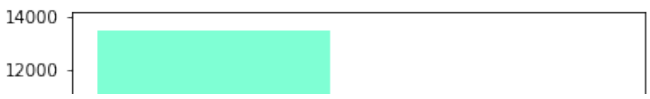
	status	total assets divided by shareholder funds minus 1	long term debt plus loans then divided by shareholder funds	total Assets to total Liabilities	Current Ratio	current assets without stocks divided by current liabilities	Shareholders Funds plus Non current liabilities then divided by Fixed assets	EBIT to interest paid	i
0	active	3.30	0.62	1.30	0.96	0.70	0.94	-21.75	
1	active	-7.93	-0.19	0.87	1.35	1.34	225.95	-42.44	
2	active	1.24	0.41	1.81	1.14	0.65	1.10	-10.84	
3	active	-1.16	-0.68	0.14	1.60	0.87	2.23	-16.29	
4	active	-1.90	-0.86	0.47	0.42	0.42	-1.78	-13.83	

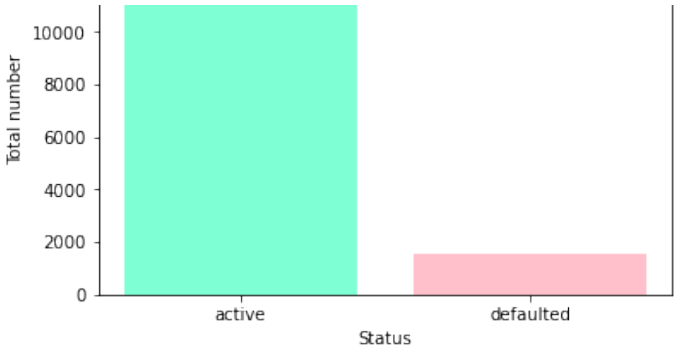
5 rows × 25 columns

In [17]:
```
#pip install matplotlib
import numpy as np
import matplotlib.pyplot as plt
```

In [18]:
```
#plt.bar()
df['status']
x = ['active', 'defaulted']
y = [df[df['status']=='active'].shape[0], df[df['status']=='default
plt.bar(x,y,color=['aquamarine', 'pink'])
plt.xlabel('Status')
plt.ylabel('Total number')
```

Out[18]:
```
Text(0, 0.5, 'Total number')
```





We can find that the status 'defaulted' is much less than 'active'.

Then, we see the proportion of defaulted companies within this dataset.

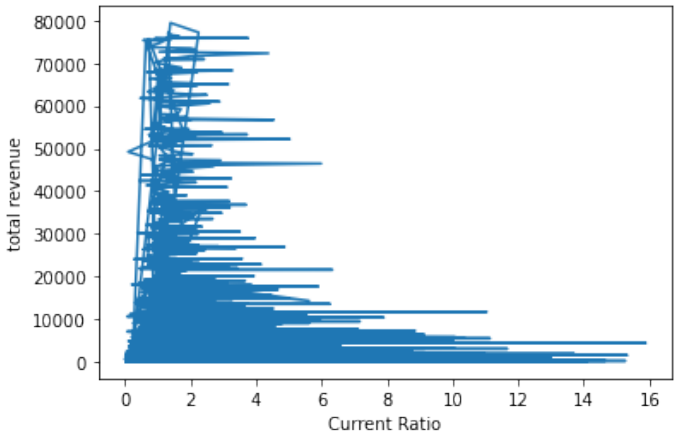
In [19]:
```
# The proportion of defaulted companies within this dataset
default = df [ df["status"] == "defaulted" ].shape[0]  / df.shape[0
default
```

Out[19]:
```
0.10269192422731804
```

Choose two variables in vector X, and see their relationships.

In [20]:
```
plt.plot(df['Current Ratio'], df['total revenue'])
#plt.show()
plt.xlabel('Current Ratio')
plt.ylabel('total revenue')
```

Out[20]:
```
Text(0, 0.5, 'total revenue')
```



We also can choose other variables to find their relationships.

Model Specific

Modelling

Preprocessing

In [21]:
```
# extract the variables which will not be used based on the paper
# out = [16,17,18,19,24]
# df.iloc[:,out]
df_droped = df.drop(df.columns[[16,17,18,19,24]], axis=1)
```

In [22]:
```
df_droped.head()
```

Out[22]:

	status	total assets divided by shareholder funds minus 1	long term debt plus loans then divided by shareholder funds	total Assets to total Liabilities	Current Ratio	current assets without stocks divided by current liabilities	Shareholders Funds plus Non current liabilities then divided by Fixed assets	EBIT to interest paid	i
0	active	3.30	0.62	1.30	0.96	0.70	0.94	-21.75	
1	active	-7.93	-0.19	0.87	1.35	1.34	225.95	-42.44	
2	active	1.24	0.41	1.81	1.14	0.65	1.10	-10.84	
3	active	-1.16	-0.68	0.14	1.60	0.87	2.23	-16.29	
4	active	-1.90	-0.86	0.47	0.42	0.42	-1.78	-13.83	

In [23]:
```
# Split dataset into train and test (80% train and 20% validation)
# import sklearn
from sklearn.model_selection import train_test_split
```

In [24]:
```
train, validation = train_test_split(df_droped, train_size = 0.8, r
#shuffle=False        ## one table split, validation; test, new tab
```

See the sizes of the train set and the validation set respecitvely.

In [25]:
```
[train.size, validation.size]
```

Out[25]: [240720, 60180]

In [26]:
```
train.head()
```

Out[26]:

	status	total assets divided by shareholder funds minus 1	long term debt plus loans then divided by shareholder funds	total Assets to total Liabilities	Current Ratio	current assets without stocks divided by current liabilities	Shareholders Funds plus Non current liabilities then divided by Fixed assets	EBI inte
425	defaulted	26.05	1.38	1.04	0.84	0.77	0.21	3
12375	active	19.51	0.00	1.05	1.25	1.25	19.22	6
13083	active	8.59	0.00	1.12	0.65	0.64	0.26	-1
6991	active	0.92	0.59	2.08	1.60	1.19	1.22	
10411	active	0.48	0.11	3.09	1.93	1.46	1.72	3

In [27]:
```
validation.head()
```

Out[27]:

	status	total assets divided by shareholder funds minus 1	long term debt plus loans then divided by shareholder funds	total Assets to total Liabilities	Current Ratio	current assets without stocks divided by current liabilities	Shareholders Funds plus Non current liabilities then divided by Fixed assets	EBIT t interes pai
4424	active	2.10	0.74	1.48	1.43	0.50	3.16	1.1
10011	active	10.07	3.25	1.10	0.80	0.80	0.76	2.3
10142	active	9.13	0.00	1.11	1.11	1.11	243.41	286.8
9389	active	0.89	0.19	2.13	2.78	2.19	21.12	4.4
8551	active	8.82	5.06	1.11	1.85	1.68	5.61	2.1

In [28]:
```
# The proportion of defaulted companies within this dataset
default_train = train [ train["status"] == "defaulted" ].shape[0]
default_validation = validation [ validation["status"] == "defaulte
[default_train, default_validation]
```

Out[28]:
```
[0.10335659687603856, 0.10003323363243602]
```

Construct a decision tree

In [29]:
```
from sklearn import tree
classifier = tree.DecisionTreeClassifier(criterion='entropy', max_d
```

In this case, we choose entropy. There are some explanation of the entropy.

The aim of entropy is to find a way to encode information efficiently and without loss, where efficency means the short average length, no loss means encode all the information.

Suppose that a message has N possible states and each state is equiprobable, i.e. $P(i) = \frac{1}{N}$

Then we need at least $log_2 N$ bits to encode it.

Thus, the minimum length to encode the information:

$$log_2 N = -log_2 \frac{1}{N} = -log_2 P$$

Entropy is the minimum average encoding length of losslessly encoded event information that obeys a particular probability distribution of events, so we can calculate entropy (discrete) by

$$Entropy = -\sum_i P(i) log_2 P(i)$$

Where $P(i)$ is the probability of the i^{th} information state.

In [30]:
```
train_x = train.drop(train.columns[0], axis = 1)
train_y = train['status']
```

In [31]:
```
classifier = classifier.fit(train_x, train_y)
```

Draw the decision tree.

In [32]:
```
pip install pydotplus
```

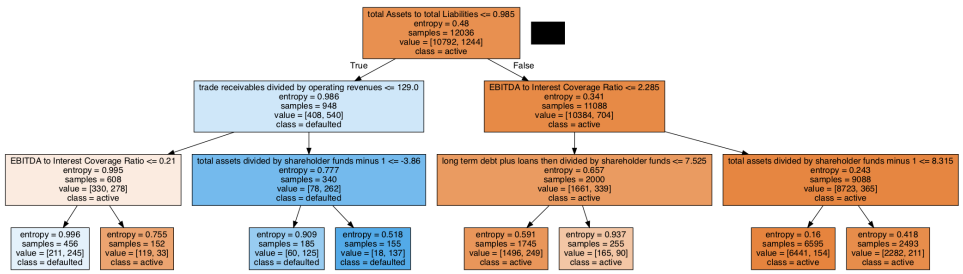
```
Requirement already satisfied: pydotplus in /Users/ranjiang/opt/an
aconda3/lib/python3.9/site-packages (2.0.2)
Requirement already satisfied: pyparsing>=2.0.1 in /Users/ranjiang
/opt/anaconda3/lib/python3.9/site-packages (from pydotplus) (3.0.4
)
Note: you may need to restart the kernel to use updated packages.
```

In [33]:
```
import pydotplus
from IPython.display import Image
```

In [34]:
```
dot_data = tree.export_graphviz(classifier, feature_names=train_x.c
                                out_file=None)
```

In [35]:
```
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```

Out[35]:



Prediction

In [36]:
```
validation_x = validation.drop(validation.columns[0], axis = 1)
validation_y = validation['status']
validation_x.head()
```

Out[36]:

	total assets divided by shareholder funds minus 1	long term debt plus loans then divided by shareholder funds	total Assets to total Liabilities	Current Ratio	current assets without stocks divided by current liabilities	Shareholders Funds plus Non current liabilities then divided by Fixed assets	EBIT to interest paid	pr bef ta p inter p th divic by tc ass
4424	2.10	0.74	1.48	1.43	0.50	3.16	1.12	0
10011	10.07	3.25	1.10	0.80	0.80	0.76	2.36	0
10142	9.13	0.00	1.11	1.11	1.11	243.41	286.87	0
9389	0.89	0.19	2.13	2.78	2.19	21.12	4.44	0
8551	8.82	5.06	1.11	1.85	1.68	5.61	2.18	0

In [37]:
```
[validation_x.shape, validation_y.shape]
```

Out[37]:
```
[(3009, 19), (3009,)]
```

In [38]:
```
# Predict
predict_y = classifier.predict(validation_x)
predict_y
```

Out[38]:
```
array(['active', 'active', 'active', ..., 'active', 'active', 'act
ive'],
      dtype=object)
```

In [39]:
```
# Test our results
predict_y[3008]
# pd.Series(predict_y,name='Real Status')
```

Out[39]:
```
'active'
```

In [40]:
```
results = pd.concat([pd.Series(validation_y.tolist(),name='Truth'),
results
```

Out[40]:

	Truth	Predicted
0	active	active
1	active	active
2	active	active
3	active	active
4	active	active
...	...	...
3004	active	active
3005	active	active
3006	active	active
3007	active	active

3007	active	active
3008	defaulted	active

3009 rows × 2 columns

In [41]:
```
features = pd.DataFrame(validation_x)
features = features.reset_index(drop = True)
```

- Tips: Note that the index is different, so we need to drop it to maintain consistency of rows.

In [42]:
```
totalTable = pd.concat([features,pd.DataFrame(results)], axis=1)
totalTable
```

Out[42]:

	total assets divided by shareholder funds minus 1	long term debt plus loans then divided by shareholder funds	total Assets to total Liabilities	Current Ratio	current assets without stocks divided by current liabilities	Shareholders Funds plus Non current liabilities then divided by Fixed assets	EBIT to interest paid	prof befo taxe plu intere pa the divide by tot asse
0	2.10	0.74	1.48	1.43	0.50	3.16	1.12	0.(
1	10.07	3.25	1.10	0.80	0.80	0.76	2.36	0.(
2	9.13	0.00	1.11	1.11	1.11	243.41	286.87	0.1
3	0.89	0.19	2.13	2.78	2.19	21.12	4.44	0.(
4	8.82	5.06	1.11	1.85	1.68	5.61	2.18	0.(
...	...	...	...	...	...	...	...	
3004	13.96	0.00	1.07	1.10	1.04	1.21	1.34	0.(
3005	19.77	3.78	1.05	0.94	0.72	0.75	2.10	0.(
3006	6.57	0.00	1.15	1.13	1.13	6.14	-94.08	-0.1
3007	5.46	0.52	1.18	1.17	1.04	2.42	2.88	0.(
3008	131.56	20.26	1.01	1.04	0.31	1.50	-2.17	-0.(

3009 rows × 21 columns

Misclassification

Recall the confusion matrix.

In [43]:
```
# Method 1
from sklearn.metrics import confusion_matrix
confusion = confusion_matrix(totalTable['Truth'],totalTable['Predic
confusion
```

Out[43]:
```
array([[2623,   85],
       [ 188,  113]])
```

In [44]:
```
# Method 2
pd.crosstab(results['Truth'],results['Predicted'],
            rownames = ['Truth'], colnames = ['Pediction'])
```

Out[44]:

Pediction	active	defaulted
Truth		
active	2623	85
defaulted	188	113

$f_{10} = 188$ $f_{01} = 85$, $f_{00} = 2623$, $f_{11} = 113$

$$ErrorRate = \frac{\text{Number of wrong predictions}}{\text{Total Number of predictions}}$$

Since this is a binary classification problem, we have:

$$ErrorRate = \frac{f_{10} + f_{01}}{f_{11} + f_{10} + f_{01} + f_{00}}$$

In [45]:
```
# Error Rate
errorRate = (confusion[0][1] + confusion[1][0]) / (confusion[0][0]
errorRate
```

Out[45]:
```
0.09072781655034895
```

Then we find that Error Rate = 9.07%.

Supplements

We also can implement the prediction by the XGboost model.

Extreme Gradient Boosting (XGBoost) is a supervised model based on the combination of tree models with Gradient Boosting, and it's faster than tree model algorithms.

More information can see
https://github.com/danpele/FINTECH_HO_2020/blob/main/3.%20Use%20Cases/2.%20Ar
%20Explainable%20Machine%20Learning%20in%20credit%20risk%20management/Rep
(https://github.com/danpele/FINTECH_HO_2020/blob/main/3.%20Use%20Cases/2.%20A
%20Explainable%20Machine%20Learning%20in%20credit%20risk%20management/Rep

Summary

- **Content**: In this case study we analyse credit risk management by the decision tree.

- **Objective**: Predict the default status.

- **Model specific**: We first construct a decision tree and predict the default status. Then, we calculate the missclassification.