

Unsupervised Learning project - New York Stock Exchange

January 28, 2021

1 Summary

The dataset for this project was collected from [kaggle](#) and originates from [Nasdaq Financials](#). `fundamentals.csv` contains New York Stock Exchange historical metrics extracted from annual SEC 10K filings (2012-2016), should be enough to derive most of popular fundamental indicators.

In this project, we will focus on **clustering** and apply unsupervised learning techniques to find the best candidate algorithm that accurately predicts whether a company has net profit or net loss. To do that, we will transform **Net Income** column into a binary representation of whether or not a company made profit, where **0** represents **loss** and **1** represents **profit**.

Why do we use net income?

Net income indicates a company's profit after all of its expenses have been deducted from revenues. This number appears on a company's income statement and is also an indicator of a company's profitability.

2 Exploratory Data Analysis

```
[1]: import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import AgglomerativeClustering
from sklearn.cluster import KMeans
from scipy.cluster import hierarchy

# Mute the sklearn and IPython warnings
import warnings
warnings.filterwarnings('ignore', module='sklearn')
pd.options.display.float_format = '{:,.2f}'.format
```

```
[2]: data = pd.DataFrame(pd.read_csv('./fundamentals.csv', sep=','))
data.head()
```

```
[2]: Unnamed: 0  Ticker Symbol  Period Ending  Accounts Payable  \
0          0          AAL      2012-12-31      3068000000.00
1          1          AAL      2013-12-31      4975000000.00
```

2	2	AAL	2014-12-31	4668000000.00
3	3	AAL	2015-12-31	5102000000.00
4	4	AAP	2012-12-29	2409453000.00

	Accounts Receivable	Add'l income/expense items	After Tax ROE	\
0	-222000000.00	-1961000000.00	23.00	
1	-93000000.00	-2723000000.00	67.00	
2	-160000000.00	-150000000.00	143.00	
3	352000000.00	-708000000.00	135.00	
4	-89482000.00	600000.00	32.00	

	Capital Expenditures	Capital Surplus	Cash Ratio	...	\
0	-1888000000.00	4695000000.00	53.00	...	
1	-3114000000.00	10592000000.00	75.00	...	
2	-5311000000.00	15135000000.00	60.00	...	
3	-6151000000.00	11591000000.00	51.00	...	
4	-271182000.00	520215000.00	23.00	...	

	Total Current Assets	Total Current Liabilities	Total Equity	\
0	7072000000.00	9011000000.00	-7987000000.00	
1	14323000000.00	13806000000.00	-2731000000.00	
2	11750000000.00	13404000000.00	2021000000.00	
3	9985000000.00	13605000000.00	5635000000.00	
4	3184200000.00	2559638000.00	1210694000.00	

	Total Liabilities	Total Liabilities & Equity	Total Revenue	\
0	24891000000.00	16904000000.00	24855000000.00	
1	45009000000.00	42278000000.00	26743000000.00	
2	41204000000.00	43225000000.00	42650000000.00	
3	42780000000.00	48415000000.00	40990000000.00	
4	3403120000.00	4613814000.00	6205003000.00	

	Treasury Stock	For Year	Earnings Per Share	Estimated Shares Outstanding
0	-367000000.00	2012.00	-5.60	335000000.00
1	0.00	2013.00	-11.25	163022222.22
2	0.00	2014.00	4.02	716915422.89
3	0.00	2015.00	11.39	668129938.54
4	-27095000.00	2012.00	5.29	73283553.88

[5 rows x 79 columns]

```
[3]: data.isnull().sum()
```

```
[3]: Unnamed: 0      0
      Ticker Symbol    0
      Period Ending    0
      Accounts Payable  0
```

```

Accounts Receivable      0
...
Total Revenue            0
Treasury Stock           0
For Year                 173
Earnings Per Share       219
Estimated Shares Outstanding 219
Length: 79, dtype: int64

```

```

[4]: plt.figure(figsize = (15, 3))
dt = data.sort_values(by = 'Net Income', ascending=False).head(50)
sns.set_context("notebook")

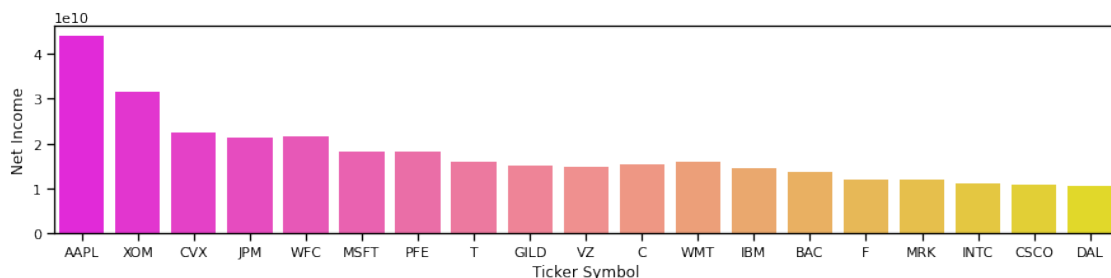
sns.barplot(x = dt['Ticker Symbol'], y =data['Net Income'], palette=("spring"),
            ci=None)

```

```

[4]: <AxesSubplot:xlabel='Ticker Symbol', ylabel='Net Income'>

```



3 Feature Transformation

- Drop **Unnamed: 0**, **Ticker Symbol** and **Period Ending** column as they don't carry any information.
- Drop columns with missing values.
- Make sure all the columns are continuous which is what we need for K-means clustering.
- Transform **Net Income** into a binary column
- Ensure the data is scaled and normally distributed

```

[5]: data.drop(['Unnamed: 0', 'Ticker Symbol', 'Period Ending'],axis = 1,
            inplace=True)
data.dropna(axis=1,inplace=True)

```

```

[6]: data.isnull().sum().all() == 0

```

```

[6]: True

```

```

[7]: data.dtypes.all() == 'float64' # all floats except Ticker Symbol

```

```
[7]: True
```

```
[8]: data['Net Income'] = data['Net Income'].apply(lambda x : 1 if x > 0 else 0)
```

```
[9]: data['Net Income'].value_counts()
```

```
[9]: 1    1679  
     0     102  
     Name: Net Income, dtype: int64
```

```
[10]: log_columns = data.skew().sort_values(ascending=False)  
      log_columns = log_columns.loc[log_columns > 0.75]  
  
      log_columns
```

```
[10]: Pre-Tax ROE                18.00  
      After Tax ROE            15.98  
      Other Operating Activities 15.83  
      Minority Interest         15.77  
      Equity Earnings/Loss Unconsolidated Subsidiary 14.91  
      Accounts Receivable       14.46  
      Common Stocks             12.15  
      Short-Term Debt / Current Portion of Long-Term Debt 11.88  
      Non-Recurring Items       11.80  
      Long-Term Debt            11.36  
      Interest Expense          11.28  
      Other Liabilities         11.07  
      Short-Term Investments     10.87  
      Cash and Cash Equivalents 10.11  
      Intangible Assets         10.03  
      Add'l income/expense items 9.98  
      Other Current Liabilities 9.89  
      Operating Margin          9.52  
      Other Current Assets       9.46  
      Retained Earnings         9.44  
      Long-Term Investments      9.25  
      Pre-Tax Margin            9.24  
      Total Liabilities         9.01  
      Other Assets              8.94  
      Deferred Asset Charges     8.86  
      Total Assets              8.82  
      Total Liabilities & Equity 8.82  
      Profit Margin             8.79  
      Other Operating Items      8.78  
      Accounts Payable          8.73  
      Misc. Stocks              8.56  
      Inventory                 7.91
```

Income Tax	7.41
Other Financing Activities	6.97
Fixed Assets	6.83
Net Cash Flow-Operating	6.82
Deferred Liability Charges	6.44
Cost of Revenue	6.25
Net Income-Cont. Operations	6.20
Earnings Before Tax	6.15
Total Revenue	6.14
Research and Development	5.93
Depreciation	5.83
Total Equity	5.82
Sales, General and Admin.	5.75
Net Receivables	5.75
Operating Income	5.66
Earnings Before Interest and Tax	5.66
Capital Surplus	5.57
Net Income Applicable to Common Shareholders	5.53
Gross Profit	5.17
Goodwill	5.11
Total Current Assets	4.90
Total Current Liabilities	4.67
Net Cash Flow	2.80
Liabilities	1.35
dtype: float64	

```
[11]: # The log transformations
      for col in log_columns.index:
          data[col] = np.log1p(data[col])
```

```
C:\Users\veres01.CRWIN\Anaconda3\lib\site-packages\pandas\core\series.py:726:
RuntimeWarning: invalid value encountered in log1p
      result = getattr(ufunc, method)(*inputs, **kwargs)
```

```
[12]: data.dropna(axis=1,inplace=True)
```

```
[13]: sc = StandardScaler()
      feature_columns = [x for x in data.columns if x not in 'Net Income']
      for col in feature_columns:
          data[col] = sc.fit_transform(data[[col]])

      data.head(4)
```

```
[13]: Accounts Payable  After Tax ROE  Capital Expenditures  \
0                0.35         0.30             -0.21
1                0.48         1.39             -0.63
2                0.47         2.17             -1.36
3                0.49         2.11             -1.64
```

	Cash and Cash Equivalents	Changes in Inventories	Common Stocks \
0	0.24	0.17	0.53
1	0.51	0.17	-0.11
2	0.40	0.17	-0.04
3	0.13	0.17	-0.07

	Cost of Revenue	Deferred Asset Charges	Deferred Liability Charges \
0	0.42	-0.84	0.58
1	0.43	-0.84	0.73
2	0.49	-0.84	0.71
3	0.43	1.41	0.69

	Effect of Exchange Rate ...	Sale and Purchase of Stock \
0	0.24 ...	0.28
1	0.24 ...	0.28
2	0.24 ...	-0.10
3	0.24 ...	-1.12

	Short-Term Debt / Current Portion of Long-Term Debt \
0	0.68
1	0.68
2	0.70
3	0.74

	Short-Term Investments	Total Assets	Total Current Assets \
0	1.48	0.26	0.52
1	1.57	0.70	0.60
2	1.54	0.71	0.58
3	1.54	0.80	0.56

	Total Current Liabilities	Total Liabilities	Total Liabilities & Equity \
0	0.60	0.61	0.01
1	0.65	1.00	0.70
2	0.65	0.94	0.71
3	0.65	0.97	0.80

	Total Revenue	Treasury Stock
0	0.84	0.25
1	0.90	0.28
2	1.28	0.28
3	1.25	0.28

[4 rows x 46 columns]

4 Train models

- Fit a **K-means clustering** model with two clusters and
- Fit 2 **Agglomerative clustering** models with two clusters (ward-link and complete-link clustering)
- Compare the results to those obtained by K-means with regards to wine color by reporting the number of red and white observations in each cluster for both K-means and agglomerative clustering.
- Visualize the **dendrogram** produced by agglomerative clustering

4.1 K-means

```
[14]: km = KMeans(n_clusters=2, random_state=42)
      km = km.fit(data[feature_columns])

      data['kmeans'] = km.predict(data[feature_columns])
      (data[['Net Income', 'kmeans']]
       .groupby(['kmeans', 'Net Income'])
       .size()
       .to_frame()
       .rename(columns={0: 'number'}))
```

```
[14]:
```

		number
kmeans	Net Income	
0	0	8
	1	295
1	0	94
	1	1384

4.2 Agglomerative Clustering

```
[15]: for linkage in ['complete', 'ward']:
      ag = AgglomerativeClustering(n_clusters=2, linkage=linkage,
      ↪compute_full_tree=True)
      ag = ag.fit(data[feature_columns])
      data[str('agglom_'+linkage)] = ag.fit_predict(data[feature_columns])
```

```
[16]: (data[['Net Income', 'agglom_ward']]
      .groupby(['Net Income', 'agglom_ward'])
      .size()
      .to_frame()
      .rename(columns={0: 'number'}))
```

```
[16]:
```

		number
Net Income	agglom_ward	
0	0	13
	1	89
1	0	323

1 1356

```
[17]: (data[['Net Income', 'agglom_complete']]
      .groupby(['Net Income', 'agglom_complete'])
      .size()
      .to_frame()
      .rename(columns={0: 'number'}))
```

```
[17]:
```

		number
Net Income	agglom_complete	
0	0	102
1	0	1671
	1	8

```
[18]: # Comparing AgglomerativeClustering with KMeans
      (data[['Net Income', 'agglom_complete', 'agglom_ward', 'kmeans']]
      .groupby(['Net Income', 'agglom_complete', 'agglom_ward', 'kmeans'])
      .size()
      .to_frame()
      .rename(columns={0: 'number'}))
```

```
[18]:
```

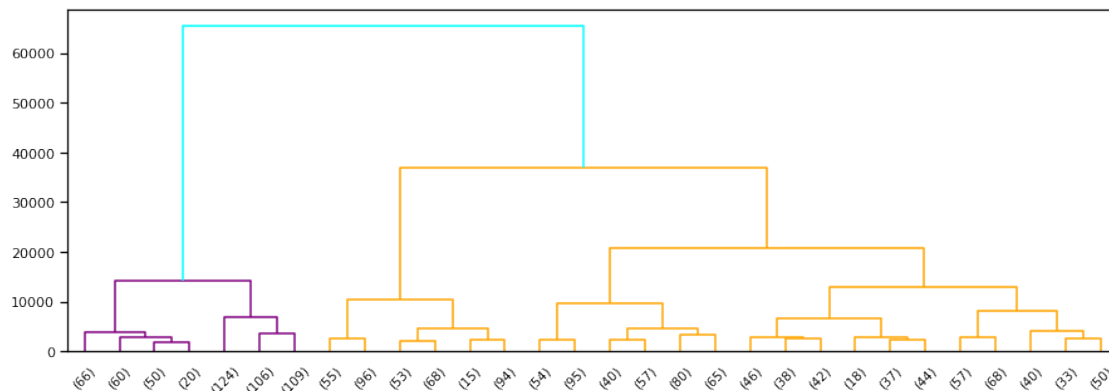
				number
Net Income	agglom_complete	agglom_ward	kmeans	
0	0	0	0	8
			1	5
		1	1	89
1	0	0	0	287
			1	28
		1	1	1356
	1	0	0	8

```
[19]: Z = hierarchy.linkage(ag.children_, method='ward')

fig, ax = plt.subplots(figsize=(15,5))

hierarchy.set_link_color_palette(['purple', 'orange'])

den = hierarchy.dendrogram(Z, orientation='top',
                           p=30, truncate_mode='lastp',
                           show_leaf_counts=True, ax=ax,
                           above_threshold_color='cyan')
```

5 Results

Comparing the results shows that I am able to predict profit better than loss which is what I expected given that we have more data for companies with profit(1: 1679 vs 0: 102). The best algorithm for predicting loss is the **Complete-link Agglomerative Clustering** model and for predicting profit **KMeans Clustering** seems to be the best candidate although **Ward-link Agglomerative Clustering** achieved nearly the same result.

Better result could be achieved by performing PCA or hyperparameter tuning.