ASSIGNMENT 3

Working with Polish Company Bankruptcy datasets:
Pre-processing and Exploratory data analysis,
Repo2Docker, Flask,

COURSE: INFO7390
Advance Data Science & Architecture

PROFESSOR:
Srikanth Krishnamurthy

SUBMITTED BY:
TEAM 9
Amit Pingale - 001898697
Himani Solanki - 001899580
Shubham Patel - 001899476

Objective:

The Report summarizes the design and implementation of machine learning performed on the Appliance Energy Consumption dataset. The report is divided into:

Part 1: Model design and building
Part 2: Model development

Part 1: Model design and building

Step 1: Convert the artff file to csv

```
In [491]: from arff2pandas import a2p
          with open('1year.arff') as f:
              df = a2p.load(f)
              print(df)
```

```
     Attr1@NUMERIC  Attr2@NUMERIC  Attr3@NUMERIC  Attr4@NUMERIC  \
0         0.200550       0.379510       0.396410        2.04720
1         0.209120       0.499880       0.472250        1.94470
2         0.248660       0.695920       0.267130        1.55480
3         0.081483       0.307340       0.458790        2.49280
4         0.187320       0.613230       0.229600        1.40630
5         0.228220       0.497940       0.359690        1.75020
6         0.111090       0.647440       0.289710        1.47050
7         0.532320       0.027059       0.705540       53.95400
8         0.009020       0.632020       0.053735        1.12630
9         0.124080       0.838370       0.142040        1.16940
10        0.240010       0.443550       0.188350        1.44000
11       -0.027117       0.111480       0.119890        2.07540
12        0.266690       0.349940       0.611470        3.02430
13        0.067731       0.198850       0.081562        2.95760
```

Step 2:Setting up the column names

In [492]:
```
df.columns = ['1 net profit / total assets',
'2 total liabilities / total assets',
'3 working capital / total assets',
'4 current assets / short-term liabilities',
'5 [(cash + short-term securities + receivables - short-term liabilities) / (operating expenses - depreciation)] * 365',
'6 retained earnings / total assets',
'7 EBIT / total assets',
'8 book value of equity / total liabilities',
'9 sales / total assets',
'10 equity / total assets',
'11 (gross profit + extraordinary items + financial expenses) / total assets',
'12 gross profit / short-term liabilities',
'13 (gross profit + depreciation) / sales',
'14 (gross profit + interest) / total assets',
'15 (total liabilities * 365) / (gross profit + depreciation)',
'16 (gross profit + depreciation) / total liabilities',
'17 total assets / total liabilities',
'18 gross profit / total assets',
'19 gross profit / sales',
'20 (inventory * 365) / sales',
'21 sales (n) / sales (n-1)',
'22 profit on operating activities / total assets',
'23 net profit / sales',
'24 gross profit (in 3 years) / total assets',
'25 (equity - share capital) / total assets',
'26 (net profit + depreciation) / total liabilities',
'27 profit on operating activities / financial expenses',
```

Step 3: Reading the file

In [4]:
```
df.head()
```

Out[4]:

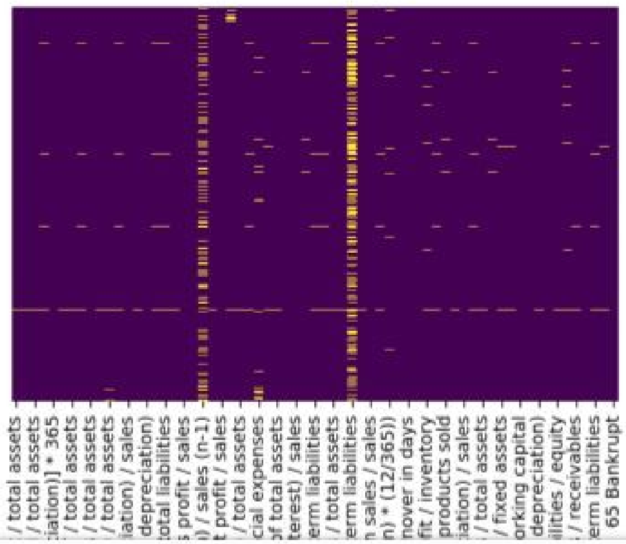
	1 net profit / total assets	2 total liabilities / total assets	3 working capital / total assets	4 current assets / short-term liabilities	5 [(cash + short-term securities + receivables - short-term liabilities) / (operating expenses - depreciation)] * 365	6 retained earnings / total assets	7 EBIT / total assets	8 book value of equity / total liabilities	9 sales / total assets	10 equity / total assets	...	56 (sales - cost of products sold) / sales	57 (current assets - inventory - short-term liabilities) / (sales - gross profit - depreciation)	58 total costs /total sales	59 long-term liabilities / equity	60 sa inven
0	0.200550	0.37951	0.39641	2.0472	32.3510	0.38825	0.249760	1.33050	1.1389	0.50494	...	0.121960	0.39718	0.87804	0.001924	8.4
1	0.209120	0.49988	0.47225	1.9447	14.7860	0.00000	0.258340	0.99601	1.6996	0.49788	...	0.121300	0.42002	0.85300	0.000000	4.1
2	0.248660	0.69592	0.26713	1.5548	-1.1523	0.00000	0.309060	0.43695	1.3090	0.30408	...	0.241140	0.81774	0.76599	0.694840	4.9
3	0.081483	0.30734	0.45879	2.4928	51.9520	0.14988	0.092704	1.86610	1.0571	0.57353	...	0.054015	0.14207	0.94598	0.000000	4.5
4	0.187320	0.61323	0.22960	1.4063	-7.3128	0.18732	0.187320	0.63070	1.1559	0.38677	...	0.134850	0.48431	0.86515	0.124440	6.3

5 rows × 65 columns

Step 4: Missing value analysis

In [5]:
```
sns.heatmap(df.isnull(), yticklabels=False, cbar=False, cmap='viridis')
```

Out[5]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1cc065e11d0>
```



Step 5: Finding the sum of null values in each column

In [6]:
```
null_counts = df.isnull().sum()
null_counts[null_counts > 0].sort_values(ascending=False)
```

Out[6]:
```
37 (current assets - inventories) / long-term liabilities                          2740
21 sales (n) / sales (n-1)                                                         1622
27 profit on operating activities / financial expenses                              311
60 sales / inventory                                                                135
45 net profit / inventory                                                           134
24 gross profit (in 3 years) / total assets                                         124
41 total liabilities / ((profit on operating activities + depreciation) * (12/365))  84
11 (gross profit + extraordinary items + financial expenses) / total assets          39
32 (current liabilities * 365) / cost of products sold                               38
28 working capital / fixed assets                                                    34
64 sales / fixed assets                                                              34
53 equity / fixed assets                                                             34
54 constant capital / fixed assets                                                   34
46 (current assets - inventory) / short-term liabilities                             31
4 current assets / short-term liabilities                                            30
40 (current assets - inventory - receivables) / short-term liabilities               30
12 gross profit / short-term liabilities                                             30
63 sales / short-term liabilities                                                    30
33 operating expenses / short-term liabilities                                       30
47 (inventory * 365) / cost of products sold                                         29
52 (short-term liabilities * 365) / cost of products sold)                           29
26 (net profit + depreciation) / total liabilities                                   25
17 total assets / total liabilities                                                  25
16 (gross profit + depreciation) / total liabilities                                 25
34 operating expenses / total liabilities                                            25
```

Step 6: Analysis for replacing the missing values

```
In [9]: df['37 (current assets - inventories) / long-term liabilities'].describe()

Out[9]: count      4287.000000
        mean        173.453694
        std        6339.491580
        min        -525.520000
        25%           1.296500
        50%           3.438300
        75%          11.393500
        max      398920.000000
        Name: 37 (current assets - inventories) / long-term liabilities, dtype: float64
```

```
In [12]: df['59 long-term liabilities / equity'].describe()

Out[12]: count    7026.000000
         mean        0.277829
         std         6.339149
         min      -327.970000
         25%         0.000000
         50%         0.028438
         75%         0.273867
         max       119.580000
         Name: 59 long-term liabilities / equity, dtype: float64
```

```
In [15]: df['57 (current assets - inventory - short-term liabilities) / (sales - gross profit - depreciation)'].describe()

Out[15]: count    7026.000000
         mean        0.193243
         std         4.344046
         min      -315.370000
         25%         0.056772
         50%         0.175745
         75%         0.351922
         max       126.670000
         Name: 57 (current assets - inventory - short-term liabilities) / (sales - gross profit - depreciation), dtype: float64
```

```
In [16]: df['21 sales (n) / sales (n-1)'].describe()

Out[16]: count     5405.000000
         mean        10.367516
         std        417.358298
         min      -1325.000000
         25%          1.024400
         50%          1.137400
         75%          1.287600
         max      27900.000000
         Name: 21 sales (n) / sales (n-1), dtype: float64
```

Step 7: Replacing the missing values

```
In [27]: df['21 sales (n) / sales (n-1)'].replace(0, 1.156960)

Out[27]: 0       1.24790
         1       1.42930
         2       1.42830
         3       1.50690
         4       1.15696
         5       1.72780
         6       0.56811
         7       1.15696
         8       1.07520
         9       1.58660
         10      1.68560
         11      1.16250
         12      1.05810
         13      1.18480
         14      0.99083
         15      1.10490
         16      1.12810
         17      0.78628
         18      1.74740
```

```
In [25]: df['21 sales (n) / sales (n-1)'].fillna(0, inplace=True)
```

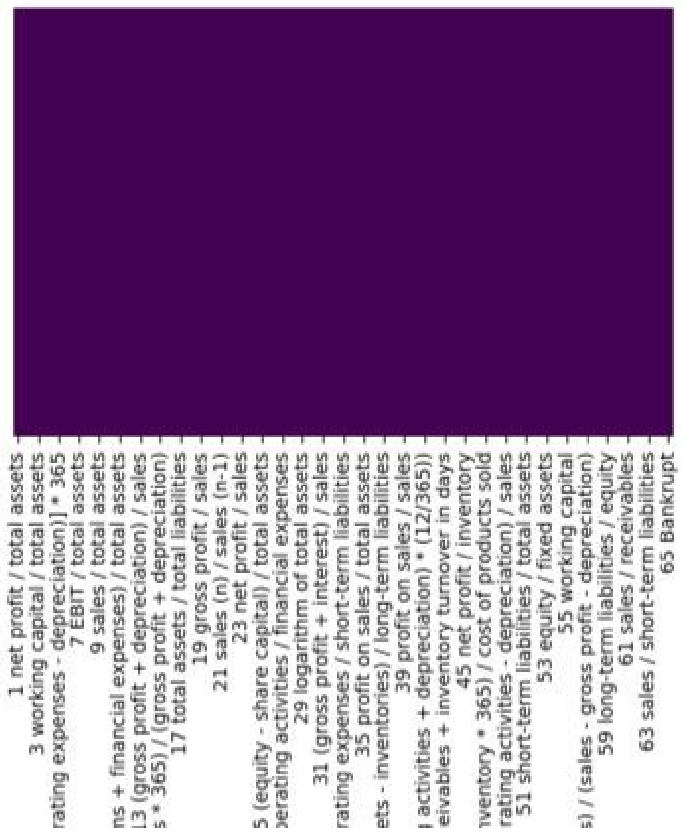
```
In [26]: df['21 sales (n) / sales (n-1)']

Out[26]: 0       1.24790
         1       1.42930
         2       1.42830
         3       1.50690
         4       0.00000
         5       1.72780
         6       0.56811
         7       0.00000
         8       1.07520
         9       1.58660
         10      1.68560
         11      1.16250
         12      1.05810
         13      1.18480
         14      0.99083
         15      1.10490
         16      1.12810
         17      0.78628
         18      1.74740
```

Step 8:Missing value analysis

In [57]:
```
sns.heatmap(df.isnull(), yticklabels=False, cbar=False, cmap='viridis')
```

Out[57]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1cc0c6b9b00>
```



Step 9:Feature selection using Recursive Feature Elimination and Ranking

In [79]:
```
X = df.drop(['65 Bankrupt'], axis =1)
y = df['65 Bankrupt'].astype('int')
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

In [80]:
```
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
rfe = RFE(model, 7)
rfe = rfe.fit(X_train, y_train)
print(rfe.support_)
print(rfe.ranking_)
print(rfe.n_features_)
rfe.score(X_train, y_train)
```

```
[False False  True False False False False False False False False False
  True False False False False False False False  True False False False
 False  True False False False False False False False False False False
 False False False False False False False False False  True False  True
 False False False  True False False False False False False False False
 False False False False]
[24 39  1  7 55 30 35 14 26 40 13 22  1 36 57 32  3 34 12 46  1 54  2 25 31
  1 56 17 20 19 23 51 10  8 21 15 52 42  4  5 27 38 48 50 33  1 45  1 11 28
 18  1 37 16 58 29 43  6 44 53 41 49  9 47]
7
```

Out[80]:
```
0.98106646058732616
```

In [81]:
```
ranking = pd.DataFrame({'Index':data_1.columns, 'Ranking': rfe.ranking_})
ranking
```

Out[81]:

	Index	Ranking
0	1 net profit / total assets	24
1	2 total liabilities / total assets	39
2	3 working capital / total assets	1
3	4 current assets / short-term liabilities	7
4	5 [(cash + short-term securities + receivables...	55
5	6 retained earnings / total assets	30
6	7 EBIT / total assets	35
7	8 book value of equity / total liabilities	14
8	9 sales / total assets	26
9	10 equity / total assets	40
10	11 (gross profit + extraordinary items + finan...	13
11	12 gross profit / short-term liabilities	22
12	13 (gross profit + depreciation) / sales	1
13	14 (gross profit + interest) / total assets	36
14	15 (total liabilities * 365) / (gross profit +...	57
15	16 (gross profit + depreciation) / total liabi...	32
16	17 total assets / total liabilities	3

Step 10: Feature selection using BorutaPy and Ranking

In [76]:

```
from boruta import BorutaPy
feat_selector = BorutaPy(rf1, n_estimators='auto', verbose=5, random_state=1)
feat_selector.fit(X_boruta,y_boruta)
```

```
Tentative:      64
Rejected:       0
Iteration:      4 / 100
Confirmed:      0
Tentative:      64
Rejected:       0
Iteration:      5 / 100
Confirmed:      0
Tentative:      64
Rejected:       0
Iteration:      6 / 100
Confirmed:      0
Tentative:      64
Rejected:       0
Iteration:      7 / 100
Confirmed:      0
Tentative:      64
Rejected:       0
Iteration:      8 / 100
Confirmed:      23
```

In [77]:

```
feat_selector.support_
print(feat_selector.ranking_)
```

```
[ 1  1  3  6  1  1  6  1  1  1 16  1  1  6 14  1  1  6  1 29  1 23  1  1  1
  1  1  6  1 22 11 33 29  1 18 20  9  1 13 31 22 24 35 37  2  1 26 25 35 12
  1 39 17 11  1 15 19  3 38 37 33 29 27  1]
```

```
In [78]: ranking = pd.DataFrame({'Index':data_1.columns, 'Ranking': feat_selector.ranking_})
         ranking
```

Out[78]:

	Index	Ranking
0	1 net profit / total assets	1
1	2 total liabilities / total assets	1
2	3 working capital / total assets	3
3	4 current assets / short-term liabilities	6
4	5 [(cash + short-term securities + receivables...	1
5	6 retained earnings / total assets	1
6	7 EBIT / total assets	6
7	8 book value of equity / total liabilities	1
8	9 sales / total assets	1
9	10 equity / total assets	1
10	11 (gross profit + extraordinary items + finan...	16
11	12 gross profit / short-term liabilities	1
12	13 (gross profit + depreciation) / sales	1
13	14 (gross profit + interest) / total assets	6
14	15 (total liabilities * 365) / (gross profit +...	14
15	16 (gross profit + depreciation) / total liabi...	1
16	17 total assets / total liabilities	1

Step 11: Finding correlation matrix on Boruta selected features

```
In [87]: corr_data = df_rfe.corr()
         corr_data
```

Out[87]:

	3 working capital / total assets	13 (gross profit + depreciation) / sales	21 sales (n) / sales (n-1)	26 (net profit + depreciation) / total liabilities	46 (current assets - inventory) / short-term liabilities	48 EBITDA (profit on operating activities - depreciation) / total assets	52 (short-term liabilities * 365) / cost of products sold)	65 Bankrupt
3 working capital / total assets	1.000000	0.218361	0.044245	0.350022	0.515240	0.270987	-0.114361	-0.055780
13 (gross profit + depreciation) / sales	0.218361	1.000000	0.080892	0.500384	0.271155	0.214673	0.031264	-0.083554
21 sales (n) / sales (n-1)	0.044245	0.080892	1.000000	0.009921	-0.015235	0.025392	-0.006535	-0.139571
26 (net profit + depreciation) / total liabilities	0.350022	0.500384	0.009921	1.000000	0.540767	0.260474	0.015682	-0.057043
46 (current assets - inventory) / short-term liabilities	0.515240	0.271155	-0.015235	0.540767	1.000000	0.200337	-0.060869	-0.054590
48 EBITDA (profit on operating activities - depreciation) / total assets	0.270987	0.214673	0.025392	0.260474	0.200337	1.000000	-0.267372	-0.024126
52 (short-term liabilities * 365) / cost of products sold)	-0.114361	0.031264	-0.006535	0.015682	-0.060869	-0.267372	1.000000	-0.000013
65 Bankrupt	-0.055780	-0.083554	-0.139571	-0.057043	-0.054590	-0.024126	-0.000013	1.000000

Step 12: Heatmap of correlation

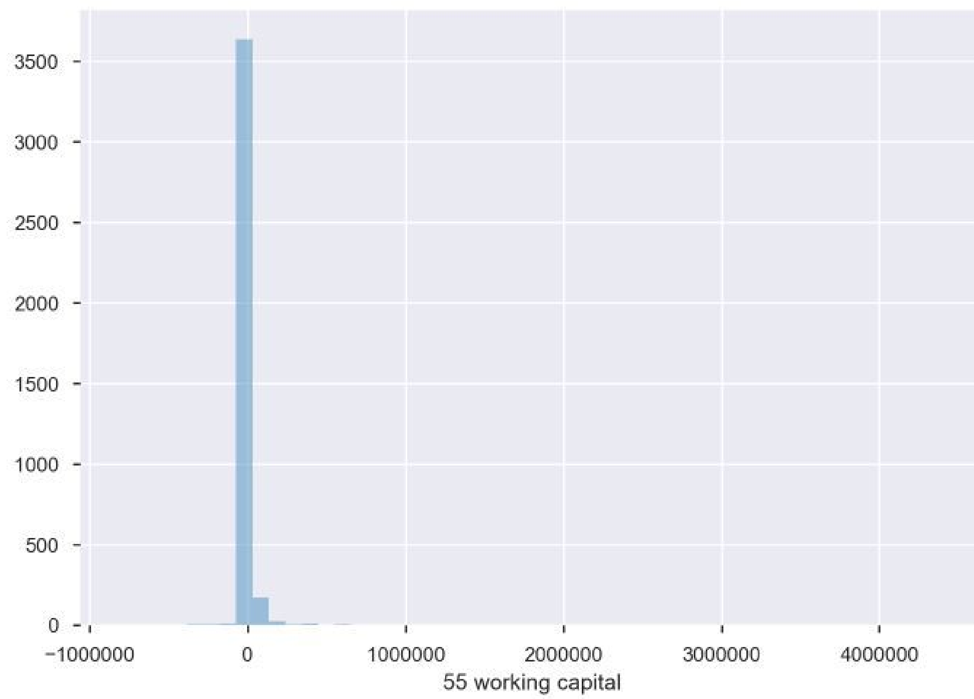
```
In [88]: f, ax = plt.subplots(figsize=(10, 8))
         corr = df_rfe.corr()
         sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool), cmap=sns.diverging_palette(220, 10, as_cmap=True),
                     square=True, ax=ax)

Out[88]: <matplotlib.axes._subplots.AxesSubplot at 0x1cc329e01d0>
```

Step 13: Plots to see the frequency of the features selected by BorutaPy

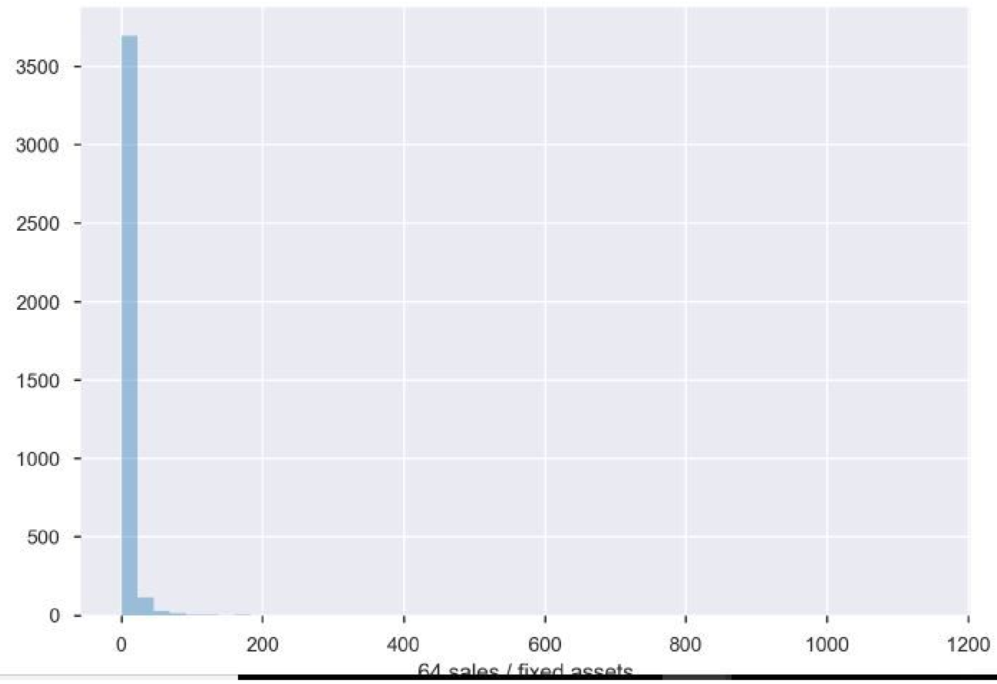
```
In [111]: sns.distplot(df['55 working capital'].astype('int'), kde=False)

Out[111]: <matplotlib.axes._subplots.AxesSubplot at 0x1cc1007a9b0>
```



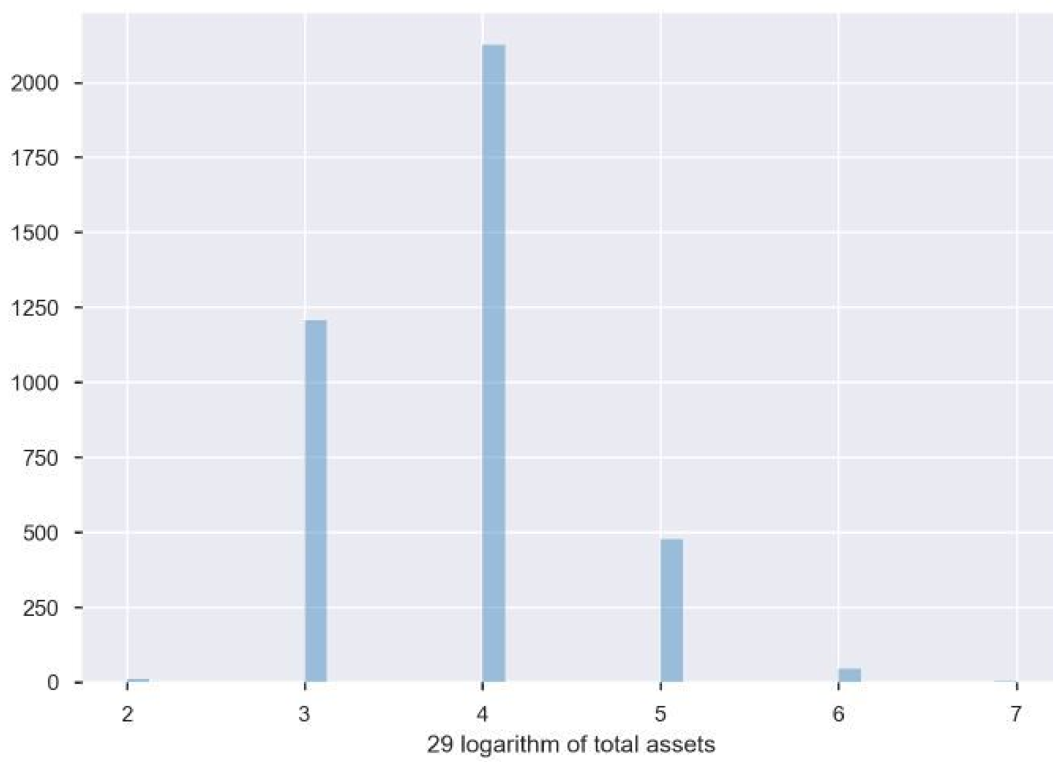
```
In [112]: sns.distplot(df['64 sales / fixed assets'].astype('int'), kde=False)
```

```
Out[112]: <matplotlib.axes._subplots.AxesSubplot at 0x1cc0ff9b470>
```




```
In [106]: sns.distplot(df['29 logarithm of total assets'].astype('int'), kde=False)
```

```
Out[106]: <matplotlib.axes._subplots.AxesSubplot at 0x1cc0f414a20>
```



Step 14: Machine learning algorithms and pickling them

Random Forest Regression

In [91]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [92]:
```
# splitting data into training and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=2)

scaler = preprocessing.MinMaxScaler()
X = scaler.fit_transform(X)
```

In [93]:
```
# build RabdomForestClassifier model with SMOTE imblearn
rfc_pipeline = make_pipeline_imb(SMOTE(random_state=4), RandomForestClassifier(n_estimators=50))
smote_model = rfc_pipeline.fit(X_train, y_train)
smote_prediction = smote_model.predict(X_test)
filename = 'rfc_model.pckl'
pickle.dump(rfc_pipeline,open(filename,'wb'))

print()
print_results("RandomForest classification", y_test, smote_prediction)
print()
```

```
Model Name: RandomForest classification
accuracy: 0.9648033126293996
precision: 0.23076923076923078
recall: 0.3
f1: 0.2608695652173913
```

Logistic Regression

In [94]:
```
from sklearn.linear_model import LogisticRegression
```

In [95]:
```
# build Logistic Rrgression Classifier model with SMOTE imblearn
lr_pipeline = make_pipeline_imb(SMOTE(random_state=4), LogisticRegression())
smote_model = lr_pipeline.fit(X_train, y_train)
smote_prediction = smote_model.predict(X_test)
filename = 'lr_model.pckl'
pickle.dump(lr_pipeline,open(filename,'wb'))

print()
print_results("Logistic Regression classification", y_test, smote_prediction)
print()
```

```
Model Name: Logistic Regression classification
accuracy: 0.7712215320910973
precision: 0.0759493670886076
recall: 0.9
f1: 0.14007782101167315
```

Neural Nets

In [96]:
```
from sklearn.neural_network import MLPClassifier
```

In [97]:
```
# build Neural Nets Classifier model with SMOTE imblearn
nn_pipeline = make_pipeline_imb(SMOTE(random_state=4), MLPClassifier())
smote_model = nn_pipeline.fit(X_train, y_train)
smote_prediction = smote_model.predict(X_test)
filename = 'nn_model.pckl'
pickle.dump(nn_pipeline,open(filename,'wb'))

print()
print_results("Neural Nets", y_test, smote_prediction)
print()
```

```
Model Name: Neural Nets
accuracy: 0.6749482401656315
precision: 0.040625
recall: 0.65
f1: 0.07647058823529412
```

BernoulliNB

In [103]:
```
from sklearn.naive_bayes import BernoulliNB
```

In [104]:
```
# build SVC model with SMOTE imblearn
svc_pipeline = make_pipeline_imb(SMOTE(random_state=4), BernoulliNB())
smote_model = svc_pipeline.fit(X_train, y_train)
smote_prediction = smote_model.predict(X_test)
filename = 'BernoulliNB_model.pckl'
pickle.dump(svc_pipeline,open(filename,'wb'))

print()
print_results("BernoulliNB", y_test, smote_prediction)
print()
```

```
Model Name: BernoulliNB
accuracy: 0.855072463768116
precision: 0.03125
recall: 0.2
f1: 0.05405405405405406
```

Step 15:Getting accuracy-error metrics

In [100]:
```
info = model_name,accuracy,precision,recall,f1score
```

In [105]:
```
describe1 = pd.DataFrame(info[0],columns = {"Model_Name"})
describe2 = pd.DataFrame(info[1], columns ={"Accuracy_score"})
describe3 = pd.DataFrame(info[2],columns = {"Precision_score"})
describe4 = pd.DataFrame(info[3],columns = {"Recall_score"})
describe5 = pd.DataFrame(info[4],columns = {"F1_score"})



des = describe1.merge(describe2, left_index=True, right_index=True, how='inner')
des = des.merge(describe3,left_index=True, right_index=True, how='inner')
des = des.merge(describe4,left_index=True, right_index=True, how='inner')
des = des.merge(describe5,left_index=True, right_index=True, how='inner')

#des = des.merge(describe9,left_index=True, right_index=True, how='inner')
final_csv = des.sort_values(ascending=False,by="Accuracy_score").reset_index(drop = True)
```

In [106]:
```
final_csv.to_csv(str(os.getcwd()) + "/accuracy_error_metrics.csv")
```

Part2: Model Deployment

Step 1:Web application