Problem description

You are to predict whether a company will go bankrupt in the following year, based on financial attributes of the company.

Perhaps you are contemplating lending money to a company, and need to know whether the company is in near-term danger of not being able to repay.

Goal

Learning objectives

- Demonstrate mastery on solving a classification problem and presenting the entire Recipe for Machine Learning process in a notebook.
- We will make suggestions for ways to approach the problem
    - But there will be little explicit direction for this task.
- It is meant to be analogous to a pre-interview task that a potential employer might assign to verify your skill

Import modules

In [1]:

```
## Standard imports
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import sklearn

import os
import math

%matplotlib inline
```

API for students

In [2]:

```
## Load the bankruptcy_helper module

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

# Reload all modules imported with %aimport
%load_ext autoreload
%autoreload 1

# Import bankruptcy_helper module
import bankruptcy_helper
%aimport bankruptcy_helper

helper = bankruptcy_helper.Helper()
```

Get the data

The first step in our Recipe is Get the Data.

- Each example is a row of data corresponding to a single company
- There are 64 attributes, described in the section below
- The column `Bankrupt` is 1 if the company subsequently went bankrupt; 0 if it did not go bankrupt
- The column `Id` is a Company Identifier

In [3]:
```python
# Data directory
DATA_DIR = "./Data"

if not os.path.isdir(DATA_DIR):
    DATA_DIR = "../resource/asnlib/publicdata/bankruptcy/data"

data_file = "5th_yr.csv"
data = pd.read_csv( os.path.join(DATA_DIR, "train", data_file) )

target_attr = "Bankrupt"

n_samples, n_attrs = data.shape
print("Date shape: ", data.shape)
```

```
Date shape:  (4818, 66)
```

Have a look at the data

We will not go through all steps in the Recipe, nor in depth.

But here's a peek

In [4]:

```
data.head()
```

Out[4]:

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	...	X57	X58	
0	0.025417	0.41769	0.0568	1.1605	-126.39	0.41355	0.025417	1.2395	1.16500	0.51773	...	0.049094	0.85835	0.12
1	-0.023834	0.2101	0.50839	4.2374	22.034	0.058412	-0.027621	3.6579	0.98183	0.76855	...	-0.031011	1.01850	0.06
2	0.030515	0.44606	0.19569	1.565	35.766	0.28196	0.039264	0.88456	1.05260	0.39457	...	0.077337	0.95006	0.25
3	0.052318	0.056366	0.54562	10.68	438.2	0.13649	0.058164	10.853	1.02790	0.61173	...	0.085524	0.97282	0
4	0.000992	0.49712	0.12316	1.3036	-71.398	0	0.001007	1.0116	1.29210	0.50288	...	0.001974	0.99925	0.01

5 rows × 66 columns

Pretty *unhelpful* !

What are these mysteriously named features ?

Description of attributes

Attribute Information: Id Company Identifier X1 net profit / total assets X2 total liabilities / total assets X3 working capital / total assets X4 current assets / short-term liabilities X5 [(cash + short-term securities + receivables - short-term liabilities) / (operating expenses - depreciation)] * 365 X6 retained earnings / total assets X7 EBIT / total assets X8 book value of equity / total liabilities X9 sales / total assets X10 equity / total assets X11 (gross profit + extraordinary items + financial expenses) / total assets X12 gross profit / short-term liabilities X13 (gross profit + depreciation) / sales X14 (gross profit + interest) / total assets X15 (total liabilities * 365) / (gross profit + depreciation) X16 (gross profit + depreciation) / total liabilities X17 total assets / total liabilities X18 gross profit / total assets X19 gross profit / sales X20 (inventory * 365) /

sales X21 sales (n) / sales (n-1) X22 profit on operating activities / total assets X23 net profit / sales X24 gross profit (in 3 years) / total assets X25 (equity - share capital) / total assets X26 (net profit + depreciation) / total liabilities X27 profit on operating activities / financial expenses X28 working capital / fixed assets X29 logarithm of total assets X30 (total liabilities - cash) / sales X31 (gross profit + interest) / sales X32 (current liabilities * 365) / cost of products sold X33 operating expenses / short-term liabilities X34 operating expenses / total liabilities X35 profit on sales / total assets X36 total sales / total assets X37 (current assets - inventories) / long-term liabilities X38 constant capital / total assets X39 profit on sales / sales X40 (current assets - inventory - receivables) / short-term liabilities X41 total liabilities / ((profit on operating activities + depreciation) * (12/365)) X42 profit on operating activities / sales X43 rotation receivables + inventory turnover in days X44 (receivables * 365) / sales X45 net profit / inventory X46 (current assets - inventory) / short-term liabilities X47 (inventory * 365) / cost of products sold X48 EBITDA (profit on operating activities - depreciation) / total assets X49 EBITDA (profit on operating activities - depreciation) / sales X50 current assets / total liabilities X51 short-term liabilities / total assets X52 (short-term liabilities * 365) / cost of products sold) X53 equity / fixed assets X54 constant capital / fixed assets X55 working capital X56 (sales - cost of products sold) / sales X57 (current assets - inventory - short-term liabilities) / (sales - gross profit - depreciation) X58 total costs /total sales X59 long-term liabilities / equity X60 sales / inventory X61 sales / receivables X62 (short-term liabilities *365) / sales X63 sales / short-term liabilities X64 sales / fixed assets

This may still be somewhat unhelpful for those of you not used to reading Financial Statements.

But that's partially the point of the exercise

- You can *still* perform Machine Learning *even if* you are not an expert in the problem domain
    - That's what makes this a good interview exercise: you can demonstrate your thought process even if you don't know the exact meaning of the terms
- Of course: becoming an expert in the domain *will improve* your ability to create better models
    - Feature engineering is easier if you understand the features, their inter-relationships, and the relationship to the target

Let's get a feel for the data

- What is the type of each attribute ?

In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4818 entries, 0 to 4817
Data columns (total 66 columns):
X1            4818 non-null object
X2            4818 non-null object
X3            4818 non-null object
X4            4818 non-null object
X5            4818 non-null object
X6            4818 non-null object
X7            4818 non-null object
X8            4818 non-null object
X9            4818 non-null float64
X10           4818 non-null object
X11           4818 non-null object
X12           4818 non-null object
X13           4818 non-null float64
X14           4818 non-null object
X15           4818 non-null object
X16           4818 non-null object
X17           4818 non-null object
X18           4818 non-null object
X19           4818 non-null float64
X20           4818 non-null float64
X21           4818 non-null object
X22           4818 non-null object
X23           4818 non-null float64
X24           4818 non-null object
X25           4818 non-null object
X26           4818 non-null object
X27           4818 non-null object
X28           4818 non-null object
X29           4818 non-null object
X30           4818 non-null float64
X31           4818 non-null float64
X32           4818 non-null object
```

```
X33         4818 non-null object
X34         4818 non-null object
X35         4818 non-null object
X36         4818 non-null object
X37         4818 non-null object
X38         4818 non-null object
X39         4818 non-null float64
X40         4818 non-null object
X41         4818 non-null object
X42         4818 non-null float64
X43         4818 non-null float64
X44         4818 non-null float64
X45         4818 non-null object
X46         4818 non-null object
X47         4818 non-null object
X48         4818 non-null object
X49         4818 non-null float64
X50         4818 non-null object
X51         4818 non-null object
X52         4818 non-null object
X53         4818 non-null object
X54         4818 non-null object
X55         4818 non-null float64
X56         4818 non-null float64
X57         4818 non-null object
X58         4818 non-null float64
X59         4818 non-null object
X60         4818 non-null object
X61         4818 non-null object
X62         4818 non-null float64
X63         4818 non-null object
X64         4818 non-null object
Bankrupt    4818 non-null int64
Id          4818 non-null int64
dtypes: float64(16), int64(2), object(48)
memory usage: 2.4+ MB
```

You may be puzzled:

- Most attributes are `object` and *not* numeric (`float64`)
- But looking at the data via `data.head()` certainly gives the impression that all attributes are numeric

Welcome to the world of messy data ! The dataset has represented numbers as strings.

- These little unexpected challenges are common in the real-word
- Data is rarely perfect and clean

So you might want to first convert all attributes to numeric

Hint

- Look up the Pandas method `to_numeric`
    - We suggest you use the option `errors='coerce'`

Evaluating your project

We will evaluate your submission on a test dataset that we provide

- It has no labels, so **you** can't use it to evaluate your model, but **we** have the labels
- We will call this evaluation dataset the "holdout" data

Let's get it

In [6]:
```
holdout_data = pd.read_csv( os.path.join(DATA_DIR, "holdout", '5th_yr.csv') )

print("Data shape: ", holdout_data.shape)
```

```
Data shape:  (1092, 65)
```

We will evaluate your model on the holdout examples using metrics

- Accuracy
- Recall
- Precision

From our lecture: we may have to make a trade-off between Recall and Precision.

Our evaluation of your submission will be partially based on how you made (and described) the trade-off.

You may assume that it is 5 times worse to *fail to identify a company that will go bankrupt* than it is to fail to identify a company that won't go bankrupt.

Your model

Time for you to continue the Recipe for Machine Learning on your own.

Submission guidelines

Although your notebook may contain many models (e.g., due to your iterative development) we will only evaluate a single model. So choose one (explain why !) and do the following.

- You will implement the body of a subroutine `MyModel`
    - That takes as argument a Pandas DataFrame
        - Each row is an example on which to predict
        - The features of the example are elements of the row
    - Performs predictions on each example
    - Returns an array or predictions with a one-to-one correspondence with the examples in the test set

We will evaluate your model against the holdout data

- By reading the holdout examples `X_hold` (as above)
- Calling `y_hold_pred = MyModel(X_hold)` to get the predictions
- Comparing the predicted values `y_hold_pred` against the true labels `y_hold` which are known only to the instructors

See the following cell as an illustration

Give the model a name (will appear in the print statement) name = "This is the name I've given to my model" X_hold = pd.read_csv(os.path.join(DATA_DIR, "holdout", '5th_yr.csv')) # Predict using MyModel y_hold_pred = MyModel(X_hold) # Compute metrics # accuracy accuracy_hold = accuracy_score(y_hold, y_hold_pred) # recall_ recall_hold = recall_score(y_hold, y_hold_pred, pos_label=1, average="binary") # precision precision_hold = precision_score(y_hold, y_hold_pred, pos_label=1, average="binary") print("\t{m:s} Accuracy: {a:3.1%}, Recall {r:3.1%}, Precision {p:3.1%}".format(m=name, a=accuracy_hold, r=recall_hold, p=precision_hold))

Remember

The holdout data is in the same format as the one we used for training

- Except that it has no attribute for the target
- So you will need to perform all the transformations on the holdout data
    - As you did on the training data
    - Including turning the string representation of numbers into actual numeric data types

All of this work *must* be performed within the body of the `MyModel` routine you will write

We will grade you by comparing the predictions array you create to the answers known to us.

In [7]:
```
import pandas as pd
import os

def MyModel(X):
    # It should create an array of predictions; we initialize it to the empty ar
ray for convenience
    predictions = []

    # YOUR CODE GOES HERE


    return predictions
```

Check your work: predict and evaluate metrics on *your* test examples

Although only the instructors have the correct labels for the holdout dataset, you may want to create your own test dataset on which to evaluate your out of sample metrics.

If you choose to do so, you can evaluate your models using the same metrics that the instructors will use.