Financial_Data_Module_5_Lesson_1

April 7, 2023

0.1 FINANCIAL DATA

MODULE 5 | LESSON 1

---

1 DOWNLOADING, CLEANING, AND TRANSFORMING DATA

Reading Time	30 minutes
Prior Knowledge	Basic Python, Probability of default (PD)
Keywords	df.dropna(), inplace, df.replace(), .astype(), pd.to_datetime(), df['X'].dt.strftime, masking / filtering, df.concat(), .append, df.values.tolist(), datetime.strptime, datetime.datetime

---

In this module, we perform the several steps required to calculate the market-implied probability of default (PD), which we introduced in Module 5. In this lesson, we will indicate the company (bond issuer) and determine which of the company's bonds to use after cleaning, transforming, and filtering the data.

In the lessons that follow in this module, we will estimate the expected cashflows and the risk-adjusted discount rate (lesson 2). Finally we estimate the market-implied probability of default (lesson 3) and compare it to the PD that Standard and Poor's (S&P) associates with the rating (lesson 4).

1.1 1. Re-introduction to Probability of Default (PD)

As we discussed in the Financial Markets course, the probability of default is the probability that a bond issuer will not meet its contractual obligations on schedule. Although the most common event of default is nonpayment leading to bankruptcy proceedings, the bond prospectus might identify other events of default, such as the failure to meet a different obligation or the violation of a financial covenant.

In the following example, we will determine the probability of default given corporate bond prices. The default probabilities that are reached in this exercise are called market-implied default probabilities. Historically, practitioners have focused on the one-year probability of default calculation.

short term: business cycle effect (shorter 1,2 years)
long term: less cyclical and more stable
→ use corp bond 1,2 year to calculate market-implied default prob.

Over shorter horizons of one or two years, firms are exposed to the business cycle effect, while over longer horizons, the business cycle effect tends to have a lesser impact and the company's capital structure becomes more important. This effect has made long-run risk levels less cyclical and more stable. Intuitively, default risk over a longer time period is less sensitive to the instantaneous default rates in the economy (Beygi 3). For this reason, we will focus on corporate bonds with one or two years until maturity to calculate the market-implied default probabilities.

We will verify the accuracy of the market-implied default probabilities with the Standard & Poor's "Average One-Year Transition Rates For Global Corporates" table, which uses historical data from 1981-2019. This transition matrix shows the observed historical probabilities of a particular rating transitioning to another rating, including default, over the course of one year.

In order to calculate the market-implied default probabilities, we must first acquire the company's current bond prices. Using a short Selenium script that emulates a user's keystrokes and clicks in a browser as a means of navigating to Trade Reporting and Compliance Engine (TRACE) bond data provided by the Financial Industry Regulatory Authority (FINRA), we can access the data needed to calculate the market-implied default probabilities.

The following is an example script. In case you do not have Selenium installed, you can visit their respective links and download them using pip in your terminal. We will also need a chromedriver (the simulated chrome browser Selenium controls). To download it using Python, you can use the WebDriver-manager package also found in PyPi.

You will need to insert your own path to your chromedriver in the code block below.

Note: The code in this lesson is adapted to work correctly in Selenium package version 4.2.0. Thus we advise to install specific version by running `!pip install selenium==4.2.0 --force-reinstall` command before proceeding with the rest of this module.

1.2 2. Selenium and WebDrivers

As discussed by van Gulik, Selenium is tool we can use to automate browser activity that would be done by a user, for example loading a web page and filling out a form on that web page. It requires a WebDriver specific to one's web browser.

We will use SM Energy (SM) in the subsequent code; however, this analysis works with the ticker of any large publicly traded company.

We download information about the company's bonds from the database TRACE (Trade Reporting and Compliance Engine), which is maintained by FINRA (Financial Industry Regulatory Authority).

[1]:
```
# Python libraries to install
import time
from datetime import date
from datetime import datetime as dt

import numpy as np
import pandas as pd
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
```

```
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.ui import Select, WebDriverWait
from webdriver_manager.chrome import ChromeDriverManager
```

[2]:
```
# Required
company_ticker = "HES"  # or try: 'F', 'KHC', 'DVN'

# Optional
company_name = "Hess"  # or try: 'Ford Motor', 'Kraft Heinz Co', 'Devon Energy'

# Optional Input Choices:
# ALL, Annual, Anytime, Bi-Monthly, Monthly, N/A, None,
# Pays At Maturity, Quarterly, Semi-Annual, Variable
coupon_frequency = "Semi-Annual"
```

[3]:
```
# Selenium script
options = Options()
options.add_argument("--headless")
options.add_argument("--no-sandbox")
options.add_argument("--disable-dev-shm-usage")
driver = webdriver.Chrome(
    service=Service(ChromeDriverManager().install()), options=options
)

# store starting time
begin = time.time()

# FINRA's TRACE Bond Center
driver.get("http://finra-markets.morningstar.com/BondCenter/Results.jsp")

# click agree
WebDriverWait(driver, 10).until(
    EC.element_to_be_clickable((By.CSS_SELECTOR, ".button_blue.agree"))
).click()

# click edit search
WebDriverWait(driver, 10).until(
    EC.element_to_be_clickable((By.CSS_SELECTOR, "a.qs-ui-btn.blue"))
).click()

# input Issuer Name
WebDriverWait(driver, 10).until(
    EC.presence_of_element_located((By.CSS_SELECTOR,␣
  ↪"input[id=firscreener-issuer]"))
)
```

```
inputElement = driver.find_element_by_id("firscreener-issuer")
inputElement.send_keys(company_name)

# input Symbol
WebDriverWait(driver, 10).until(
    EC.presence_of_element_located((By.CSS_SELECTOR,␣
 ↪"input[id=firscreener-cusip]"))
)
inputElement = driver.find_element_by_id("firscreener-cusip")
inputElement.send_keys(company_ticker)

# click advanced search
WebDriverWait(driver, 10).until(
    EC.element_to_be_clickable((By.CSS_SELECTOR, "a.ms-display-switcher.hide"))
).click()

# input Coupon Frequency
WebDriverWait(driver, 10).until(
    EC.presence_of_element_located((By.CSS_SELECTOR,␣
 ↪"select[name=interestFrequency]"))
)
Select(
    (driver.find_elements_by_css_selector("select[name=interestFrequency]"))[0]
).select_by_visible_text(coupon_frequency)

# click show results
WebDriverWait(driver, 10).until(
    EC.element_to_be_clickable((By.CSS_SELECTOR, "input.
 ↪button_blue[type=submit]"))
).click()

# wait for results
WebDriverWait(driver, 10).until(
    EC.presence_of_element_located(
        (By.CSS_SELECTOR, ".rtq-grid-row.rtq-grid-rzrow .rtq-grid-cell-ctn")
    )
)

# create DataFrame from scrape
frames = []
for page in range(1, 11):
    bonds = []
    WebDriverWait(driver, 10).until(
        EC.presence_of_element_located(
            (By.CSS_SELECTOR, (f"a.qs-pageutil-btn[value='{str(page)}']"))
        )
    )  # wait for page marker to be on expected page
```

```
    time.sleep(2)

    headers = [
        title.text
        for title in driver.find_elements_by_css_selector(
            ".rtq-grid-row.rtq-grid-rzrow .rtq-grid-cell-ctn"
        )[1:]
    ]

    tablerows = driver.find_elements_by_css_selector(
        "div.rtq-grid-bd > div.rtq-grid-row"
    )
    for tablerow in tablerows:
        tablerowdata = tablerow.find_elements_by_css_selector("div.
 ↪rtq-grid-cell")
        bond = [item.text for item in tablerowdata[1:]]
        bonds.append(bond)

        # Convert to DataFrame
        df = pd.DataFrame(bonds, columns=headers)

    frames.append(df)

    try:
        driver.find_element_by_css_selector("a.qs-pageutil-next").click()
    except:  # noqa E722
        break

bond_prices_df = pd.concat(frames)

# store end time
end = time.time()

# total time taken
print(f"Total runtime of the program is {end - begin} seconds")

bond_prices_df
```

```
[WDM] - Downloading: 100%|      | 6.83M/6.83M [00:00<00:00, 101MB/s]
/tmp/ipykernel_148/2651632054.py:30: DeprecationWarning: find_element_by_*
commands are deprecated. Please use find_element() instead
  inputElement = driver.find_element_by_id("firscreener-issuer")
/tmp/ipykernel_148/2651632054.py:37: DeprecationWarning: find_element_by_*
commands are deprecated. Please use find_element() instead
  inputElement = driver.find_element_by_id("firscreener-cusip")
/tmp/ipykernel_148/2651632054.py:50: DeprecationWarning:
find_elements_by_css_selector is deprecated. Please use
```

```
find_elements(by=By.CSS_SELECTOR, value=css_selector) instead
  (driver.find_elements_by_css_selector("select[name=interestFrequency]"))[0]
/tmp/ipykernel_148/2651632054.py:78: DeprecationWarning:
find_elements_by_css_selector is deprecated. Please use
find_elements(by=By.CSS_SELECTOR, value=css_selector) instead
  for title in driver.find_elements_by_css_selector(
/tmp/ipykernel_148/2651632054.py:83: DeprecationWarning:
find_elements_by_css_selector is deprecated. Please use
find_elements(by=By.CSS_SELECTOR, value=css_selector) instead
  tablerows = driver.find_elements_by_css_selector(
/tmp/ipykernel_148/2651632054.py:87: DeprecationWarning:
find_elements_by_css_selector is deprecated. Please use
find_elements(by=By.CSS_SELECTOR, value=css_selector) instead
  tablerowdata = tablerow.find_elements_by_css_selector("div.rtq-grid-cell")
/tmp/ipykernel_148/2651632054.py:87: DeprecationWarning:
find_elements_by_css_selector is deprecated. Please use
find_elements(by=By.CSS_SELECTOR, value=css_selector) instead
  tablerowdata = tablerow.find_elements_by_css_selector("div.rtq-grid-cell")
/tmp/ipykernel_148/2651632054.py:87: DeprecationWarning:
find_elements_by_css_selector is deprecated. Please use
find_elements(by=By.CSS_SELECTOR, value=css_selector) instead
  tablerowdata = tablerow.find_elements_by_css_selector("div.rtq-grid-cell")
/tmp/ipykernel_148/2651632054.py:87: DeprecationWarning:
find_elements_by_css_selector is deprecated. Please use
find_elements(by=By.CSS_SELECTOR, value=css_selector) instead
  tablerowdata = tablerow.find_elements_by_css_selector("div.rtq-grid-cell")
/tmp/ipykernel_148/2651632054.py:87: DeprecationWarning:
find_elements_by_css_selector is deprecated. Please use
find_elements(by=By.CSS_SELECTOR, value=css_selector) instead
  tablerowdata = tablerow.find_elements_by_css_selector("div.rtq-grid-cell")
/tmp/ipykernel_148/2651632054.py:87: DeprecationWarning:
find_elements_by_css_selector is deprecated. Please use
find_elements(by=By.CSS_SELECTOR, value=css_selector) instead
  tablerowdata = tablerow.find_elements_by_css_selector("div.rtq-grid-cell")
/tmp/ipykernel_148/2651632054.py:87: DeprecationWarning:
find_elements_by_css_selector is deprecated. Please use
find_elements(by=By.CSS_SELECTOR, value=css_selector) instead
  tablerowdata = tablerow.find_elements_by_css_selector("div.rtq-grid-cell")
/tmp/ipykernel_148/2651632054.py:87: DeprecationWarning:
find_elements_by_css_selector is deprecated. Please use
find_elements(by=By.CSS_SELECTOR, value=css_selector) instead
  tablerowdata = tablerow.find_elements_by_css_selector("div.rtq-grid-cell")
/tmp/ipykernel_148/2651632054.py:87: DeprecationWarning:
find_elements_by_css_selector is deprecated. Please use
find_elements(by=By.CSS_SELECTOR, value=css_selector) instead
  tablerowdata = tablerow.find_elements_by_css_selector("div.rtq-grid-cell")
/tmp/ipykernel_148/2651632054.py:87: DeprecationWarning:
find_elements_by_css_selector is deprecated. Please use
```

```
find_elements(by=By.CSS_SELECTOR, value=css_selector) instead
  tablerowdata = tablerow.find_elements_by_css_selector("div.rtq-grid-cell")

Total runtime of the program is 18.554232597351074 seconds

/tmp/ipykernel_148/2651632054.py:97: DeprecationWarning:
find_element_by_css_selector is deprecated. Please use
find_element(by=By.CSS_SELECTOR, value=css_selector) instead
  driver.find_element_by_css_selector("a.qs-pageutil-next").click()
```

[3]:

	Issuer Name	Symbol	Callable	Sub-Product Type	Coupon	Maturity	Moody's®	S&P	Price	Yield
0	HESS CORP	HES.GH	Yes	Corporate Bond	6.000	01/15/2040	Baa3	BBB-	103.168	5.703
1	HESS CORP	HES.GI	Yes	Corporate Bond	5.600	02/15/2041	Baa3	BBB-	98.762	5.711
2	HESS CORP	HES4136877	Yes	Corporate Bond	3.500	07/15/2024	Baa3	BBB-	98.100	5.070
3	HESS CORP	HES4405829	Yes	Corporate Bond	4.300	04/01/2027	Baa3	BBB-	99.055	4.562
4	HESS CORP	HES4405830	Yes	Corporate Bond	5.800	04/01/2047	Baa3	BBB-	100.271	5.779
5	HESS MIDSTREAM OPERATIONS LP	HES5392919	Yes	Corporate Bond	5.500	10/15/2030	Ba2	BB+	95.088	6.332
6	HESS MIDSTREAM OPERATIONS LP	HES4567499	Yes	Corporate Bond	5.625	02/15/2026	WR	BB+	104.500	4.431
7	HESS MIDSTREAM OPERATIONS LP	HES4927355	Yes	Corporate Bond	5.625	02/15/2026		BB+	98.250	6.303
8	HESS MIDSTREAM OPERATIONS LP	HES5233164	Yes	Corporate Bond	4.250	02/15/2030	Ba2	BB+	90.015	6.052
9	HESS MIDSTREAM PARTNERS LP	HES4918686	Yes	Corporate Bond	5.125	06/15/2028	Ba2	BB+	96.143	6.001

1.3 3. Cleaning, Transforming, and Filtering

We will now filter the corporate bond prices DataFrame to align with the purpose of this example using the code below.

[4]:
```python
def bond_dataframe_filter(df):
    # Drop bonds with missing yields and missing credit ratings
    df["Yield"].replace("", np.nan, inplace=True)
    df["Moody's®"].replace({"WR": np.nan, "": np.nan}, inplace=True)
    df["S&P"].replace({"NR": np.nan, "": np.nan}, inplace=True)
    df = df.dropna(subset=["Yield"])
    df = df.dropna(subset=["Moody's®"])
    df = df.dropna(subset=["S&P"])
```

```
    # Create Maturity Years column that aligns with Semi-Annual Payments from␣
↪corporate bonds
    df["Yield"] = df["Yield"].astype(float)
    df["Coupon"] = df["Coupon"].astype(float)
    df["Price"] = df["Price"].astype(float)
    now = dt.strptime(date.today().strftime("%m/%d/%Y"), "%m/%d/%Y")
    df["Maturity"] = pd.to_datetime(df["Maturity"]).dt.strftime("%m/%d/%Y")
    daystillmaturity = []
    yearstillmaturity = []
    for maturity in df["Maturity"]:
        daystillmaturity.append((dt.strptime(maturity, "%m/%d/%Y") - now).days)
        yearstillmaturity.append((dt.strptime(maturity, "%m/%d/%Y") - now).days␣
↪/ 360)
    df = df.reset_index(drop=True)
    df["Maturity"] = pd.Series(daystillmaturity)
    #         `df['Maturity Years'] = pd.Series(yearstillmaturity).round()` #␣
↪Better for Annual Payments
    df["Maturity Years"] = (
        round(pd.Series(yearstillmaturity) / 0.5) * 0.5
    )  # Better for Semi-Annual Payments

    # Target bonds with short-term maturities
    df["Maturity"] = df["Maturity"].astype(float)
    years_mask = (df["Maturity Years"] > 0) & (df["Maturity Years"] <= 5)
    df = df.loc[years_mask]
    return df
```

[5]:
```
bond_df_result = bond_dataframe_filter(bond_prices_df)
bond_df_result
```

[5]:
```
  Issuer Name      Symbol Callable Sub-Product Type  Coupon  Maturity  \
2   HESS CORP  HES4136877      Yes   Corporate Bond     3.5     465.0
3   HESS CORP  HES4405829      Yes   Corporate Bond     4.3    1455.0

  Moody's®   S&P   Price  Yield  Maturity Years
2     Baa3  BBB-  98.100  5.070             1.5
3     Baa3  BBB-  99.055  4.562             4.0
```

Make sure that you review the documentation for the relevant code and are familiar with how the code above works; in particular, understand the following (and their parameters), as these will all serve you well when you clean, transform, and filter your data in the future (the related documentation is also required reading):

`df.dropna()` `inplace` `df.replace()` `df['X'].astype()` `pd.to_datetime()` `df['X']).dt.strftime` and the code lines that involve the variable `years_mask` Also, be sure to understand how `df.values.tolist()` works.

1.4 4. Conclusion

In this lesson, we revisited the calculation for market-implied probability of default but in much more detail. We also downloaded price and rating information for bonds issued by a particular issuer from a well-regarded bond database, FINRA's TRACE. Once we downloaded it, we cleaned it, transformed some of the data related to bond maturity, and filtered the data so that we could focus our analysis on the bonds with shorter maturities. In the following lesson, we will take the next step toward calculating the market-implied probability of default by estimating the expected cashflows. Then, we can estimate the risk-adjusted discount rate, which equates the bond price to those expected cash flows.

References

- Donnelly, Hugh. "Calculating a Company's Probability of Default with Python." AlphaWave Data. https://github.com/AlphaWaveData/Jupyter-Notebooks/blob/master/AlphaWave%20Market-Implied%20Probability%20of%20Default%20Example.ipynl
- van Gulik, Eltjo. "Basic website performance testing with Python and Selenium." 19 Mar 2021. https://www.go-euc.com/basic-website-performance-testing/
- Beygi, Sajjad et al. "Features of a Lifetime PD Model: Evidence from Public, Private, and Rated Firms." Moody's Analytics, May 2018.
- The code and related documentation used in this lesson is adapted from: **Hugh Donnelly, CFA***AlphaWave Data* **March 2021** under the following MIT License:

  Copyright (c) 2020 HDVI Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Note: The above MIT license notice is copied here to comply with its requirements, but it does **not** apply to the content in these lesson notes.
