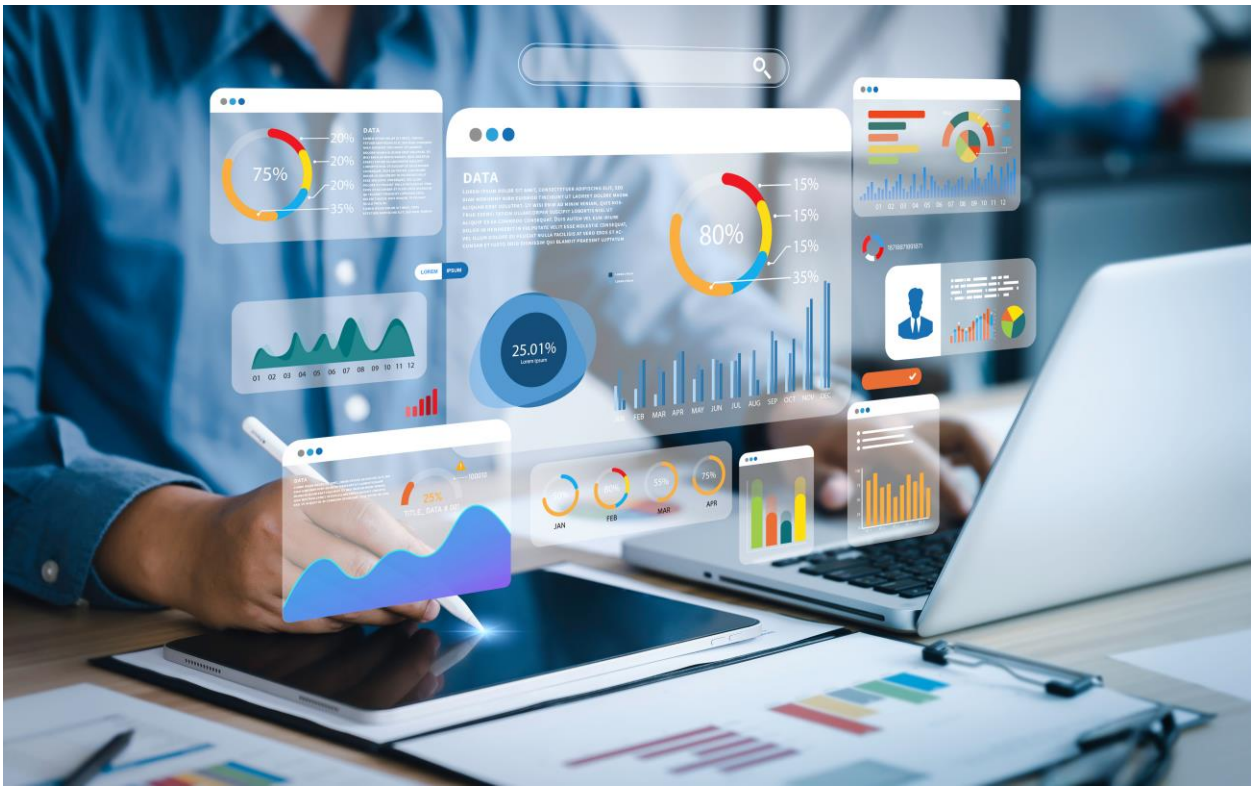
ACC 424 Accounting Information System

Financial Statement Analysis with AI-Assisted Data Visualization
Python notes for 4.1-4.4



Section 001 MW 9:30 AM – 10:45 AM at Rm. 257 over Jan 13 – May 07
Section 002 MW 11:00 AM – 12:15 AM at Rm. 127 over Jan 13 – May 07

4.1. Financial statement analysis

Financial statement analysis involves evaluating a company's financial statements (such as the income statement, balance sheet, and cash flow statement) to assess its **financial health**, **performance**, and **future prospects**. It helps stakeholders (investors, managers, creditors) make informed decisions.

Financial statements are prepared periodically (e.g., **quarterly** or **annually**) and provide a **snapshot** of the company's financial position. The three main types of financial statements are:

1. **Income Statement**: This statement shows the company's revenues, expenses, and profits over a specific period. It answers the question, "How much money did the company make or lose during this time?" Key components include revenue, cost of goods sold (COGS), gross profit, operating expenses, and net income.
2. **Balance Sheet**: The balance sheet provides a snapshot of the company's financial position at a specific point in time. It lists the company's **assets** (what it owns), ***liabilities*** (what it owes), and **shareholders' equity** (the owners' claim after liabilities). **Assets = Liabilities + Shareholders' Equity**.
3. **Cash Flow Statement**: This statement tracks the flow of **cash** into and out of the company over a period. It is divided into three sections: **operating** activities (day-to-day business), **investing** activities (buying/selling assets), and **financing** activities (raising capital or paying dividends). It answers the question, "How did the company generate and use its cash?"

The Role of **AI** in Financial Statement Analysis

In recent years, AI-assisted tools have revolutionized financial statement analysis. These tools can **automate** data processing, perform complex calculations, and generate **interactive visualizations**, making the analysis faster, more accurate, and more accessible. For example:

AI can automatically extract data from financial statements and calculate key ratios.

Machine learning algorithms can identify trends and **patterns** in financial data that might be missed by human analysts.

Data visualization tools (e.g., Python's Plotly) can create **interactive dashboards** that make it easier for stakeholders to understand complex financial data.

4.2. Key Financial Ratios and Calculations

Financial ratios serve as a bridge between raw financial data and actionable insights. These ratios are useful because they **standardize** financial data, making it easier to compare companies of different sizes or industries. For example, a small company and a large corporation can be compared using the **net profit margin**, which expresses profitability as a percentage of revenue.

Common Financial Ratios:

1. **Profitability Ratios**: Profitability ratios measure a company's ability to generate **profit** relative to its revenue, assets, or equity. They are crucial for assessing how efficiently a company is operating.
    - **Gross Profit Margin** = (Gross Profit / Revenue) × 100

      This ratio shows the percentage of revenue that remains after deducting the **cost of goods sold** (COGS). A high gross profit margin indicates that the company is effectively controlling its **production costs**.

- **Net Profit Margin** = (Net Income / Revenue) × 100

  This ratio measures the percentage of revenue that remains as **net income** after all **expenses**, including taxes and interest. A high net profit margin suggests that the company is managing its **expenses** well.

- **Return on Equity (ROE)** = (Net Income / Shareholder's Equity) × 100

  This ratio evaluates how effectively a company is using shareholders' equity to generate profit. A high ROE indicates that the company is generating strong returns for its **shareholders**.

2. **Liquidity Ratios**: Liquidity ratios assess a company's ability to meet its **short-term obligations** using its current assets. They are important for evaluating **financial stability**.

   - **Current Ratio** = Current Assets / Current Liabilities

     This ratio measures the company's ability to pay off its short-term liabilities with its current assets. A current ratio greater than 1 indicates that the company has enough **assets** to cover its short-term **liabilities**.

   - **Quick Ratio** = (Current Assets - Inventory) / Current Liabilities

     Also known as the **acid-test** ratio, this ratio is a more stringent measure of liquidity because it excludes inventory (inventory is not always easily convertible to **cash** in the short term) from current assets. A quick ratio greater than 1 suggests that the company can meet its short-term obligations without relying on **selling inventory**.

3. **Solvency Ratios**: Solvency ratios evaluate a company's **long-term** financial stability and its ability to meet **long-term obligations**. These ratios are particularly important for creditors and investors.

   - **Debt-to-Equity Ratio** = Total Liabilities / Shareholder's Equity

This ratio measures the proportion of debt to shareholders' equity. A high debt-to-equity ratio indicates that the company is heavily reliant on **debt** financing, which may increase **financial risk**.

- **Interest Coverage Ratio** = EBIT / Interest Expense

  This ratio assesses the company's ability to pay **interest** on its **debt** using its earnings before interest and taxes (EBIT). A high interest coverage ratio suggests that the company can **comfortably** meet its interest obligations.

4. **Efficiency Ratios**: Efficiency ratios measure how effectively a company is using its assets and **resources** to generate **revenue**. These ratios are important for evaluating **operational performance**.

   - **Asset Turnover Ratio** = Revenue / Total Assets

     This ratio evaluates how efficiently a company is using its **assets** to generate **revenue**. A high asset turnover ratio indicates that the company is using its assets **effectively** to generate sales.

   - **Inventory Turnover Ratio** = Cost of Goods Sold / Average Inventory

     This ratio measures how quickly a company sells and replaces its **inventory**. A high **inventory turnover** ratio suggests that the company is managing its inventory efficiently.

Motivation for Using Ratios:

- Ratios **simplify** complex financial data into meaningful metrics.
- They allow for **comparisons** across companies and industries.
- AI tools can **automate** ratio calculations and **visualize** trends over time.

4.3. AI-Assisted Data Visualization

What is AI-Assisted Data Visualization?

AI-assisted data visualization uses machine learning and artificial intelligence tools to **automatically** process financial data and generate **interactive** charts, graphs, and dashboards. This helps quickly identify trends, **outliers**, and **patterns**.

Motivation for AI-Assisted Visualization:

- **Efficiency**: Automates **repetitive** tasks like data cleaning and chart creation. AI tools can process **large** datasets and generate visualizations much **faster** than humans. This saves time to focus on **interpreting** the results rather than creating charts.
- **Accuracy**: By automating data processing and visualization, AI reduces the **risk** of human error. This ensures that the insights derived from the data are reliable and accurate.
- **Insight Discovery**: AI algorithms can identify patterns, trends, and correlations in the data that might be **missed** by human analysts.
- **User-Friendly**: AI-assisted visualization tools often come with **interactive** features, such as zooming, filtering, and drilling down into specific data points. This makes it easier for **non-technical** stakeholders (e.g., executives, investors) to explore the data and understand the insights.

Common Visualization Tools:

There are many tools available for AI-assisted data visualization. In the hands-on section (4.4), we will use the following Python libraries and tools to perform financial analysis and create visualizations:

1. **Pandas** is a powerful library for data manipulation and analysis. While Pandas is primarily used for **data manipulation**, it also includes basic visualization capabilities, such as plotting line charts and bar graphs directly from **DataFrames**.
2. Matplotlib and Seaborn: **Matplotlib** is a widely-used library for creating **static** visualizations, such as line charts, bar graphs, and scatter plots. **Seaborn** builds on Matplotlib and provides a **higher-level** interface for creating more attractive and informative statistical graphics.
3. **Plotly** is a library that specializes in creating **interactive** visualizations. Unlike static charts, Plotly's visualizations allow users to **hover** over data points, **zoom in** on specific areas, and **filter** data dynamically. This makes it an excellent tool for exploring financial data and presenting insights in an **engaging** way.
4. **Scikit-learn** is a **machine learning** library that can be used for advanced financial analysis, such as **predicting** future trends based on **historical** data. While we will only touch on its basic capabilities in this section, Scikit-learn is a powerful tool for building predictive models and performing statistical analysis.

4.4. Hands-On Python: Financial Analysis and Visualization

https://anaconda.cloud/share/notebooks/05f70fbd-ef5c-4aee-80bb-fde64484c485/overview

Please use the above weblink to download the Python coding of **Financial Analysis and Visualization**.

In this hands-on section, we:

1. **Imported financial data** using Pandas.
2. **Calculated key financial ratios** such as Gross Profit Margin and Current Ratio.

3. **Created static visualizations** using Matplotlib and Seaborn to analyze trends.
4. **Built interactive visualizations** using Plotly for a more engaging experience.
5. **Performed predictive analysis** using Scikit-learn to forecast future performance.

Step 1: Import Libraries

First, we need to import the necessary Python libraries. These libraries provide the tools we need for data manipulation, analysis, and visualization.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from sklearn.linear_model import LinearRegression  # For predictive analysis
```

- **Pandas**: Used for data manipulation and analysis. It allows us to work with **DataFrames**.
- **NumPy**: Provides support for **numerical** operations, such as creating arrays for predictive analysis.
- **Matplotlib** and **Seaborn**: Used for creating **static** visualizations like line charts and bar graphs.
- **Plotly**: Used for creating **interactive** visualizations.
- **Scikit-learn**: A machine learning library used for **predictive** analysis.

Step 2: Create Financial Data

We create a **Pandas** DataFrame containing financial data for a company over four years. The data includes:

- **Year**: The year of the financial data.

- **Revenue**: Total revenue for the year.
- **Gross Profit**: Gross profit for the year.
- **Current Assets**: Total current assets for the year.
- **Current Liabilities**: Total current liabilities for the year.
- **Net Income**: Net profit for the year.

```
# Create financial data
data = {
    'Year': [2020, 2021, 2022, 2023],
    'Revenue': [100000, 120000, 150000, 180000],
    'Gross Profit': [40000, 48000, 60000, 72000],
    'Current Assets': [50000, 55000, 60000, 65000],
    'Current Liabilities': [25000, 30000, 35000, 40000],
    'Net Income': [10000, 14400, 22500, 32400]
}

# Convert the data into a Pandas DataFrame
df = pd.DataFrame(data)

# Display the DataFrame
print(df)
```

- We create a **dictionary** (data) containing financial data.
- The pd.DataFrame() function converts the dictionary into a Pandas DataFrame, which is a **tabular** data structure.
- The print(df) statement displays the DataFrame so we can **verify** the data.

Step 3: Calculate Financial Ratios

We calculate the following key financial ratios:

- **Gross Profit Margin**: Measures the percentage of revenue that remains after deducting the **cost of goods sold**.

- **Current Ratio**: Measures the company's ability to pay **short-term** liabilities with its current assets.
- **Net Profit Margin**: Measures the percentage of revenue that remains as **net income** after all expenses.

```
# Calculate Gross Profit Margin
df['Gross Profit Margin'] = (df['Gross Profit'] / df['Revenue']) * 100

# Calculate Current Ratio
df['Current Ratio'] = df['Current Assets'] / df['Current Liabilities']

# Calculate Net Profit Margin
df['Net Profit Margin'] = (df['Net Income'] / df['Revenue']) * 100

# Display the updated DataFrame with calculated ratios
print(df[['Year', 'Gross Profit Margin', 'Current Ratio', 'Net Profit Margin']])
```

- We use **arithmetic** operations to calculate the ratios and store them in new columns in the **DataFrame**.
- The **print()** statement displays the calculated ratios for each year.

Step 4: Create Static Visualizations

Let's create **static** visualizations using **Matplotlib** and **Seaborn** to analyze trends in the financial ratios.

Plot Gross Profit Margin Over Time

```
# Plot Gross Profit Margin over time
plt.figure(figsize=(10, 6))
sns.lineplot(x='Year', y='Gross Profit Margin', data=df, marker='o')
plt.title('Gross Profit Margin Over Time')
plt.xlabel('Year')
plt.ylabel('Gross Profit Margin (%)')
plt.grid(True)
plt.show()
```

- plt.figure(figsize=(10, 6)): Sets the **size** of the plot.

- sns.lineplot(): Creates a **line** plot using **Seaborn**.
- plt.title(), plt.xlabel(), and plt.ylabel(): Add a title and axis labels to the plot.
- plt.grid(True): Adds a **grid** to the plot for better readability.
- plt.show(): Displays the plot.

Plot Current Ratio Over Time

```
# Plot Current Ratio over time
plt.figure(figsize=(10, 6))
sns.lineplot(x='Year', y='Current Ratio', data=df, marker='o', color='red')
plt.title('Current Ratio Over Time')
plt.xlabel('Year')
plt.ylabel('Current Ratio')
plt.grid(True)
plt.show()
```

- Like the previous plot, but this time we're visualizing the **Current Ratio**.
- The color='red' argument changes the **line** color to red.

Step 5: Create Interactive Visualizations

For more engaging visualizations, we'll use **Plotly** to create **interactive** charts. We can **hover over** any **data point** to see its **exact** value.

Interactive Line Plot for Gross Profit Margin

```
# Interactive line plot for Gross Profit Margin
fig = px.line(df, x='Year', y='Gross Profit Margin',
              title='Gross Profit Margin Over Time',
              labels={'value': 'Gross Profit Margin (%)', 'variable': 'Category'})
fig.show()
```

- px.line(): Creates an **interactive** **line** plot using Plotly.

- The title and labels arguments **customize** the plot's title and axis labels.
- fig.show(): Displays the interactive plot.

Interactive Bar Plot for Current Ratio

```
# Interactive bar plot for Current Ratio
fig = px.bar(df, x='Year', y='Current Ratio',
             title='Current Ratio Over Time',
             labels={'value': 'Current Ratio', 'variable': 'Category'})
fig.show()
```

- px.bar(): Creates an **interactive bar** plot using Plotly.
- The title and labels arguments customize the plot's title and axis labels.
- fig.show(): Displays the interactive plot.

Step 6: Predictive Analysis

Using **Scikit-learn**, we can perform **predictive** analysis to **forecast** future financial performance based on **historical** data.

Predict Gross Profit Margin for Future Years

```
# Prepare data for prediction
X = df[['Year']]  # Independent variable (Year)
y = df['Gross Profit Margin']  # Dependent variable (Gross Profit Margin)

# Train a linear regression model
model = LinearRegression()
model.fit(X, y)

# Predict Gross Profit Margin for the next 3 years
future_years = np.array([[2024], [2025], [2026]])
predictions = model.predict(future_years)

# Display predictions
print("Predicted Gross Profit Margins for 2024-2026:")
for year, prediction in zip(future_years, predictions):
    print(f"{year[0]}: {prediction:.2f}%")
```

- X and Y: Define the independent (**Year**) and dependent (**Gross Profit Margin**) variables.
- LinearRegression(): Creates a **linear** regression model.
- model.fit(X, y): Trains the model using the **historical** data.
- model.predict(): Predicts the Gross Profit Margin for **future** years.
- The print() statement displays the predictions.