LESSON

Data Pipelines Guided Lesson

Lesson Goals

In this guided lesson, we will complete a full data project and construct a data pipeline from start to finish.

Introduction

When working as a data analyst or data scientist, it is important to organize the code you write so that it is understandable and repeatable. Data pipelines can help you do this. By wrapping your code in functions and structuring your project according to the stages you learned about in the Data Pipelines lesson, you can produce more professional analyses.

In this lesson, we will assume that we are working as an analyst at an investment firm. We have been tasked with pulling stock market data, analyzing it, and exporting our analysis so that the firm's portfolio managers can review it and use it to make important investment decisions.

More specifically, they have provided us with a list of 30 companies they are interested in, and they would like us to analyze the stocks of these companies to determine which are the 10 most likely to provide the best expected return per unit of risk.

They would also like to ensure that the correlation of returns between the top 10 stocks is not too high as they would like to further reduce their risk by

days of available stock prices.

To calculate these metrics and produce the necessary analysis for them, we will need to pull historical stock price data for each company. From this stock price data, we will need to calculate the average return for a specific period of time. We will also need to quantify how risky each company's stock is, which can be measured by the variability of the returns and how correlated the returns of stocks are using the correlation coefficient.

The Data

We will be obtaining our stock data from Quandl which has a Python API that we can access via the `quandl` library. We can pip install the library as follows, and once it is installed, we should be able to import it and access financial data.

```
$ pip install quandl
```

Note: The Quandl API is free to use and, as of this writing, will give you 50 free API calls per day if you don't have a Quandl account. If you need more than that, you can get up to 50,000 API calls per day if you sign up for a free Quandl account.

Data Acquisition

The first data set we will need is the list of 30 companies the portfolio managers would like us to analyze. They have placed it in a data directory, and we can import it as follows.

Copy

```
    companies = pd.read_csv(path)
    ticker_list = list(companies['Ticker'])
    print('Retrieved', str(len(ticker_list)), 'ticker
symbols.')
    return ticker_list
```

Once we have this list of companies, we will need to extract the ticker symbol for each company and make a call to the Quandl API in order to retrieve the historical prices for that company's stock.

Note: `companies.csv` can be downloaded here.

Copy

```
import quandl

def get_prices(ticker):
    print('Retrieving data for', ticker)
    prices = quandl.get('WIKI/' + ticker)['Adj.
Close'].reset_index()
    prices['Ticker'] = ticker
    return prices

data = []

ticker_list = get_tickers('companies.csv')

for ticker in ticker_list:
    prices = get_prices(ticker)
    data.append(prices)
```

When this is run, you should see the following output.

```
Retrieving data for AMZN
Retrieving data for MSFT
Retrieving data for GOOG
Retrieving data for TSLA
Retrieving data for FB
Retrieving data for AAP
Retrieving data for DIS
Retrieving data for NKE
Retrieving data for UA
Retrieving data for BAC
Retrieving data for CCL
Retrieving data for CI
Retrieving data for AAL
Retrieving data for LUV
Retrieving data for WFC
Retrieving data for WMT
Retrieving data for HD
Retrieving data for ORCL
Retrieving data for IBM
Retrieving data for NCLH
Retrieving data for RCL
Retrieving data for TWX
Retrieving data for FOX
Retrieving data for F
Retrieving data for GCI
Retrieving data for XOM
Retrieving data for WWE
Retrieving data for WM
```

The result is a list where each element is a data frame containing the Date, Adj. Close price, and Ticker for each company.

Data Wrangling

the required analysis for the portfolio managers.

The first thing we are going to do is concatenate the list of data frames using the `concat` method so that they are all in a single data frame. Then, we are going to pivot the data using the `pivot_table` method so that the rows each represent a single date, each column represents a company, and the values in the pivot table are the company's stock price on a specific date.

```
def concat_pivot(data, rows, columns, values):
    df = pd.concat(data, sort=True)
    pivot = df.pivot_table(values=values, columns=columns, index=rows)
    return pivot
```

Copy

Now we have company stock prices per day, but the metrics we need for our analysis need to be calculated from the stocks' *returns* or the percentage change in the stock's price from day to day. We can create another pivot table containing these returns pretty easily by using the `pct_change` method.

```
def compute_returns(df):
    returns = df.pct_change()
    return returns
```

Copy

This will go down each column and calculate the percentage change in value between each row and the row preceding it.

At this point, we have daily historical stock price returns for each company of interest. Our data is now at a point where we can be analyzed, so let's jump right into it.

We are going to create a function that filters our returns to just the last X number of days and computes the mean return for each company and the standard deviation of returns. From there, we divide each company's mean by its standard deviation to get the average return per unit of risk metric the portfolio managers wanted to see.

```
def return_risk_ratio(df, days=30):
    means = pd.DataFrame(df.tail(days).mean())
    std = pd.DataFrame(df.tail(days).std())
    ratios = pd.concat([means, std], axis=1).reset_index()
    ratios.columns = ['Company', 'Mean', 'Std']
    ratios['Ratio'] = ratios['Mean']/ratios['Std']
    return ratios
```

Copy

Once we have this, we can sort in descending order and filter for just the top 10 companies that had the highest ratios.

```
top10 = ratios.sort_values('Ratio', ascending=False).head(10)
```

Copy

And once we have this top 10 list, we can then compare the correlations of their returns over varying periods of time so that the portfolio managers can limit their risk by not investing in stocks whose returns are too highly correlated with

← Go back | Copy URL for Students | Edit in Studio

```
    corr_matrix = df.tail(days).corr()
    return corr_matrix

target_list = returns[list(top10['Company'])]
correlation = corr_matrix(target_list)
```

Copy

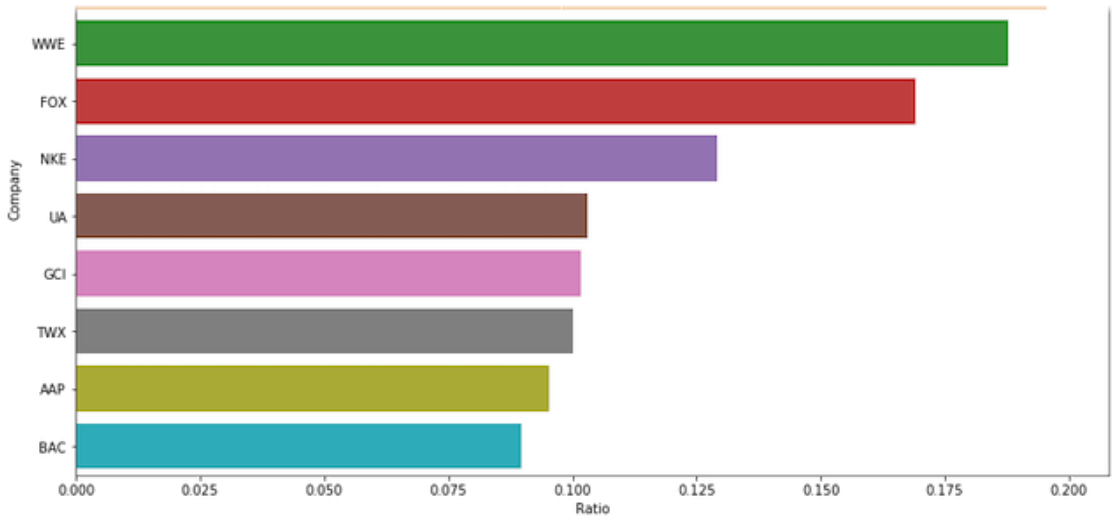
Reporting and Distribution

Now that we have completed the necessary steps to analyze the data, the next step is to produce and distribute the reports. First, we are going to put together a horizontal bar chart that shows the top 10 companies and their return vs. risk ratios.

```
import matplotlib.pyplot as plt
import seaborn as sns

def barchart(df, x, y, length=8, width=14, title=""):
    df = df.sort_values(x, ascending=False)
    plt.figure(figsize=(width,length))
    chart = sns.barplot(data=df, x=x, y=y)
    plt.title(title + "\n", fontsize=16)
    return chart

bar_plot = barchart(top10, 'Ratio', 'Company',
title='Stock Return vs. Risk Ratios')
```

Copy

← Go back Copy URL for Students Edit in Studio

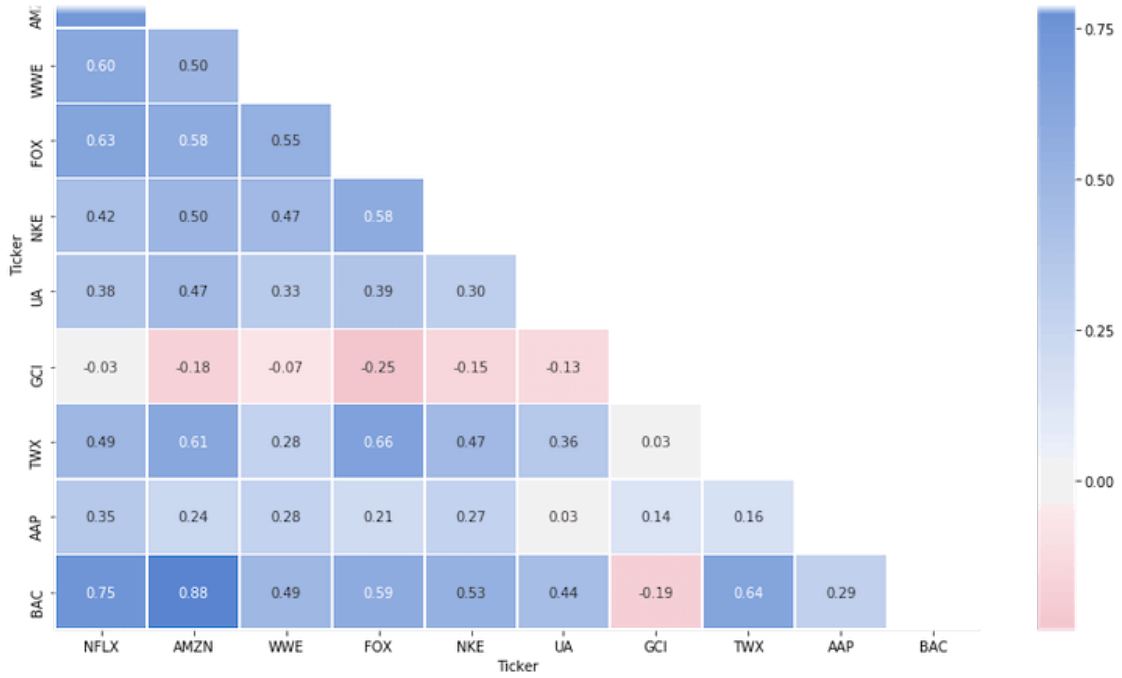


Next, we will produce a correlation matrix heatmap that visually shows the correlations between each company's returns.

Copy

```
import numpy as np

def correlation_plot(corr, title=""):
    mask = np.zeros_like(corr, dtype=np.bool)
    mask[np.triu_indices_from(mask)] = True

    plt.subplots(figsize=(15, 10))
    cmap = sns.diverging_palette(6, 255, as_cmap=True)

    chart = sns.heatmap(corr, mask=mask, cmap=cmap,
center=0, linewidths=.5, annot=True, fmt='.2f')
    plt.title(title, fontsize=16)
    return chart

corr_plot = correlation_plot(correlation, title='Stock
Return Correlation')
```

← Go back Copy URL for Students Edit in Studio



We will save each chart we create as a .PNG image file so that the portfolio managers can view them and make their investment decisions based on the information presented.

```
1   def save_viz(chart, title):
2       fig = chart.get_figure()
3       fig.savefig(title + '.png')
```

Copy

Putting the Entire Pipeline Together

Now that we have most of the code base, it only takes a little bit of modification and reorganization to structure this so that it is repeatable for 90, 180, and 360 day time periods. In a Python (.py) file, we will perform all our imports first, then include each the functions we wrote above, create new functions for each

The complete code should look something like the following and should produce a total of six charts - a bar chart and a correlation heatmap for 90, 180, and 360 days.

Copy

```
 4  import matplotlib.pyplot as plt
 5  import seaborn as sns
 6
 7  def get_tickers(path):
 8      companies = pd.read_csv(path)
 9      ticker_list = list(companies['Ticker'])
10      print('Retrieved', str(len(ticker_list)), 'ticker
    symbols.')
11      return ticker_list
12
13  def get_prices(ticker):
14      print('Retrieving data for', ticker)
15      prices = quandl.get('WIKI/' + ticker)['Adj.
    Close'].reset_index()
16      prices['Ticker'] = ticker
17      return prices
18
19  def concat_pivot(data, rows, columns, values):
20      df = pd.concat(data, sort=True)
21      pivot = df.pivot_table(values=values,
    columns=columns, index=rows)
22      return pivot
23
24  def compute_returns(df):
25      returns = df.pct_change()
26      return returns
27
28  def return_risk_ratio(df, days=30):
29      means = pd.DataFrame(df.tail(days).mean())
30      std = pd.DataFrame(df.tail(days).std())
31      ratios = pd.concat([means, std],
    axis=1).reset_index()
32      ratios.columns = ['Company', 'Mean', 'Std']
33      ratios['Ratio'] = ratios['Mean']/ratios['Std']
34      return ratios
35
36  def corr_matrix(df, days=30):
37      corr_matrix = df.tail(days).corr()
38      return corr_matrix
39
40  def barchart(df, x, y, length=8, width=14, title=""):
```

```
45      return chart
46
47  def correlation_plot(corr, title=""):
48      mask = np.zeros_like(corr, dtype=np.bool)
49      mask[np.triu_indices_from(mask)] = True
50
51      plt.subplots(figsize=(15, 10))
52      cmap = sns.diverging_palette(6, 255, as_cmap=True)
53
54      chart = sns.heatmap(corr, mask=mask, cmap=cmap,
    center=0, linewidths=.5, annot=True, fmt='.2f')
55      plt.title(title, fontsize=16)
56      return chart
57
58  def save_viz(chart, title):
59      fig = chart.get_figure()
60      fig.savefig(title + '.png')
61
62  def acquire():
63      data = []
64
65      ticker_list = get_tickers('companies.csv')
66
67      for ticker in ticker_list:
68          prices = get_prices(ticker)
69          data.append(prices)
70      return data
71
72  def wrangle(data):
73      pivot = concat_pivot(data, 'Date', 'Ticker', 'Adj.
    Close')
74      returns = compute_returns(pivot)
75      return returns
76
77  def analyze(returns, days=30):
78      ratios = return_risk_ratio(returns, days=days)
79      top10 = ratios.sort_values('Ratio',
    ascending=False).head(10)
80
81      target_list = returns[list(top10['Company'])]
82      correlation = corr_matrix(target_list)
```

```
    title='Stock Return vs. Risk Ratios - ' + str(day) + '
    Days')
87      save_viz(bar_plot, 'Return vs. Risk Top 10 - ' +
    str(day) + ' Days')
88
89      corr_plot = correlation_plot(correlation,
    title='Stock Return Correlation - ' + str(day) + '
    Days')
90      save_viz(corr_plot, 'Correlation Plot - ' + str(day)
    + ' Days')
91
92  if __name__ == "__main__":
93      data = acquire()
94      returns = wrangle(data)
95
96      num_days = [90,180,360]
97
98      for day in num_days:
99          top10, correlation = analyze(returns, days=day)
100         report(top10, correlation)
```

We can save the Python file as `stock_analysis.py` and run it from the command line as follows.

```
$ python stock_analysis.py
```

Summary

In this guided lesson, we walked through a full data analysis project and the construction of a data pipeline that performed all the steps in the analysis - from acquiring the data to wrangling it, analyzing it, and generating and distributing visualizations. We tried to make the example contain a sufficient amount of complexity that you would get a taste of what building a real data

way. Doing so will ensure that you produce higher quality data analytics work.

Mark lesson as completed

PREVIOUS LESSON
← Data Pipelines

NEXT LESSON
PROJECT | Data Pipel