FIN 3080 assignment 2 Qiu Junwen 122090436

Before question:

Data preprocessing: download the corresponding data on CSMAR database separately. Download monthly stock clothing price, return (without cash dividend reinvested) from individual stock trading table; quarterly return on equity – TTM and net Assets per share from financial indicator table; daily stock volatility at 2010/12/31 from stock market derivative index table. Load them into python using pandas:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

data1 = pd.read_excel('TRD_Mnth.xlsx')
data2 = pd.read_excel('FI_T5(Merge Query).xlsx')
data3 = pd.read_excel('STK_MKT_STKBTAL.xlsx')
data1 = data1.drop(data1.index[:2])
data2 = data2.drop(data2.index[:2])
data3 = data3.drop(data3.index[:2])
```

After change name and types of the columns, preprocessing and clearing data, below are the row data:

```
[27] data1.info()
0.1s

<class 'pandas.core.frame.DataFrame'>
Int64Index: 551377 entries, 1 to 553038
Data columns (total 5 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   Code                   551377 non-null  object
 1   Trading month          551377 non-null  datetime64[ns]
 2   Monthly closing price  551377 non-null  float64
 3   Monthly return         551377 non-null  float64
 4   Ending date            551377 non-null  datetime64[ns]
dtypes: datetime64[ns](2), float64(2), object(1)
memory usage: 25.2+ MB
```

```
[28] data2.info()
0.0s

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 208985 entries, 0 to 208984
Data columns (total 6 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   Code                 208985 non-null  object
 1   Short name           208985 non-null  object
 2   Ending date          208985 non-null  datetime64[ns]
 3   Statement type       208985 non-null  object
 4   Return on equity     208985 non-null  float64
 5   Net assets per share 208985 non-null  float64
dtypes: datetime64[ns](1), float64(2), object(3)
memory usage: 9.6+ MB
```

```python
data3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1692 entries, 0 to 1691
Data columns (total 3 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Code               1692 non-null   object
 1   Trading date       1692 non-null   datetime64[ns]
 2   Return volatility  1692 non-null   float64
dtypes: datetime64[ns](1), float64(1), object(1)
memory usage: 39.8+ KB
```

Then, construct lagged variable and merge data1 and data2 together and thus calculate the monthly PB ratio for further analysis in problem 1 & 2:

```python
# define an adjusting function to create lagged varibles
def add_lagged(trading_time):
    year = trading_time.year
    month = trading_time.month
    day = trading_time.day
    if month in [1, 2, 3]:
        return f"{year-1}-12-31"
    if month in [4, 5, 6]:
        return f"{year}-03-31"
    if month in [7, 8, 9]:
        return f"{year}-06-30"
    if month in [10, 11, 12]:
        return f"{year}-09-30"

data1['Ending date'] = data1['Trading month'].apply(add_lagged)
data1['Ending date'] = pd.to_datetime(data1['Ending date'])
data1 = data1[data1['Trading month'] >= '2009-12-31']
# merge the necessary data into a single dataframe
data = pd.merge(left=data1, right=data2, on=['Code', 'Ending date'], how='inner')
data.head()
```

	Code	Trading month	Monthly closing price	Monthly return	Ending date	Short name	Statement type	Return on equity	Net assets per share
0	000001	2010-01-01	21.70	-10.9561	2009-12-31	SFZA	A	0.272887	6.591545
1	000001	2010-02-01	22.45	3.4562	2009-12-31	SFZA	A	0.272887	6.591545
2	000001	2010-03-01	23.20	3.3408	2009-12-31	SFZA	A	0.272887	6.591545
3	000001	2010-04-01	20.56	-11.3793	2010-03-31	SFZA	A	0.280066	7.119722
4	000001	2010-05-01	17.51	-14.8346	2010-03-31	SFZA	A	0.280066	7.119722

Define an additional regression set for regression :

```python
# select the required columns for regression
regression_data = (data[(data['Ending date'] == '2010-09-30') & (data['Trading month'] == '2010-12-01')]).reset_index(drop = True)
regression_data = pd.merge(left=data3, right=regression_data, on='Code', how='inner')

# regression_data['Code', 'Monthly PB', 'Return on equity', 'Return volatility']
regression_data = regression_data[['Code', 'Monthly PB', 'Return on equity', 'Return volatility']]

regression_data
```

	Code	Monthly PB	Return on equity	Return volatility
0	000001	1.711926	0.239194	0.375077
1	000002	1.831914	0.143634	0.360978
2	000004	7.999336	0.229803	0.382118
3	000005	4.969425	-0.090036	0.409691
4	000006	2.436701	0.141314	0.502408
...	...	...	...	...
1384	601958	5.873981	0.049726	0.517640
1385	601988	1.371196	0.183965	0.201127
1386	601991	2.026694	0.093324	0.285140
1387	601998	1.693021	0.183411	0.357737
1388	601999	3.828095	0.088044	0.419746

1389 rows × 4 columns

Problem 1: import statsmodel.api to do the regression for Monthly PB based on ROE and risk volatility at the end of 2010

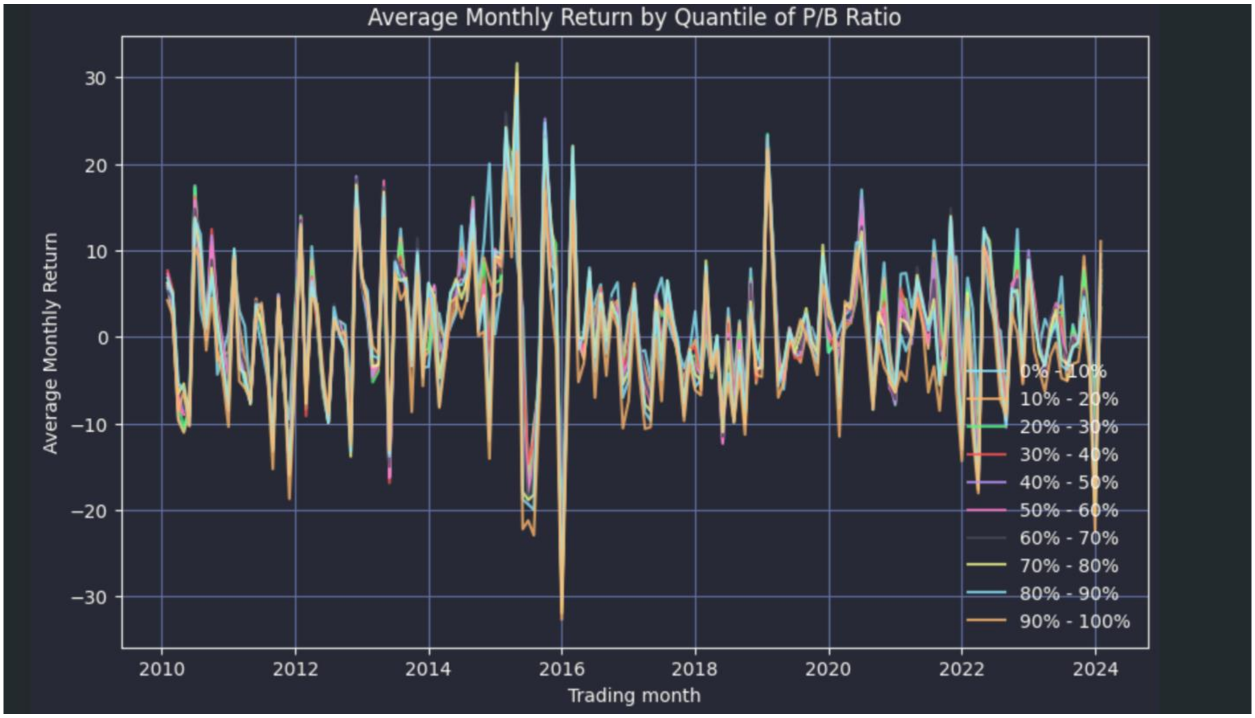
```python
# start the regression analysis
import statsmodels.api as sm
X = regression_data[['Return on equity', 'Return volatility']]
X = sm.add_constant(X)  # Add a constant term
y = regression_data['Monthly PB']

model = sm.OLS(y, X)
results = model.fit()

# Print the regression results
print(results.summary())
```

22] ✓ 0.0s

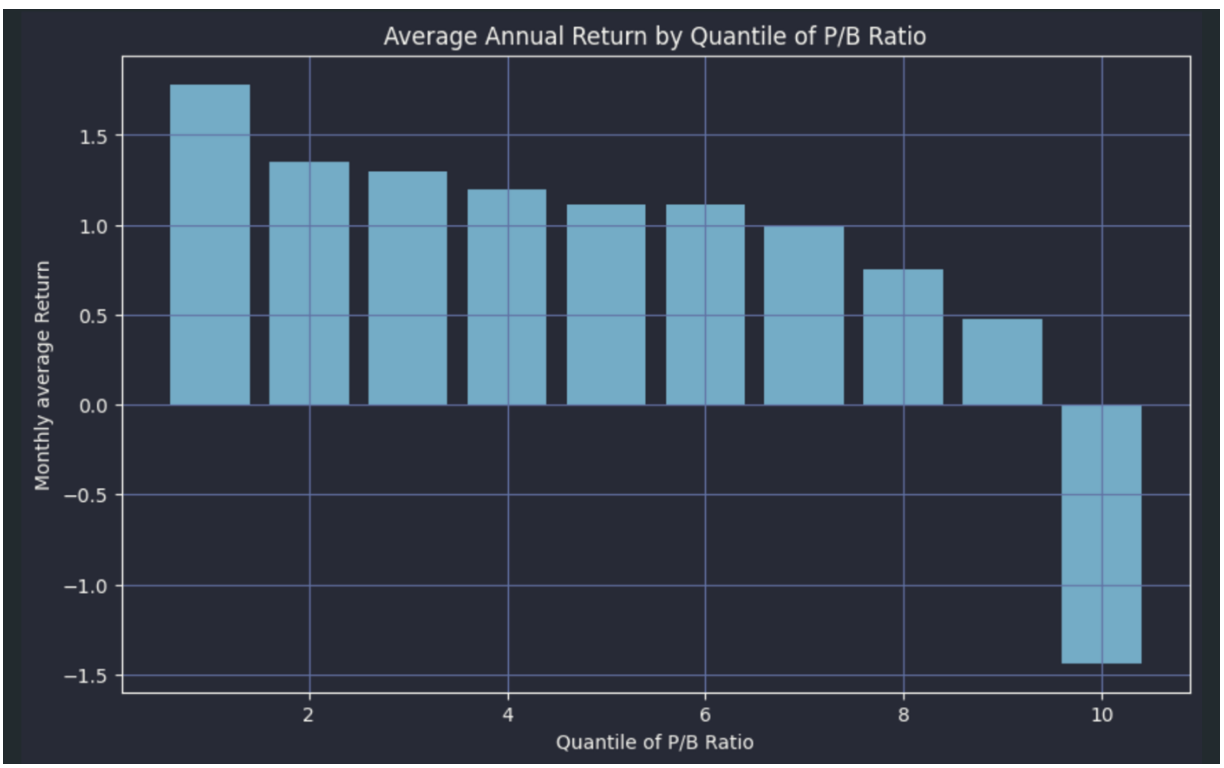
```
                            OLS Regression Results
==============================================================================
Dep. Variable:             Monthly PB   R-squared:                       0.134
Model:                            OLS   Adj. R-squared:                  0.133
Method:                 Least Squares   F-statistic:                     107.5
Date:                Tue, 26 Mar 2024   Prob (F-statistic):           3.82e-44
Time:                        15:51:32   Log-Likelihood:                -2885.9
No. Observations:                1389   AIC:                             5778.
Df Residuals:                    1386   BIC:                             5794.
Df Model:                           2
Covariance Type:            nonrobust
=====================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
const                 0.0966      0.289      0.334      0.738      -0.471       0.664
Return on equity      1.7659      0.418      4.220      0.000       0.945       2.587
Return volatility     8.8194      0.628     14.052      0.000       7.588      10.051
==============================================================================
Omnibus:                      129.823   Durbin-Watson:                   1.786
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              180.588
Skew:                           0.733   Prob(JB):                     6.11e-40
Kurtosis:                       3.987   Cond. No.                         14.6
==============================================================================
```

From the regression results, we found that the monthly PB ratio of stocks have positive correlation with the return on equity and return volatility, and their p value are both cloth to 0, implying that there exist statictic significance of return on equity and return volatility at $p < 0.05$. The Prob(F-statistic) = 3.82e-44, indicates the linear regression model *P/Bi = a + b1ROE + b2StockVolatility + residual* is statistically significant. R^2 here implies the independent variables here explain 13.4% percentage of the volatility of stocks' monthly PB ratio.

Problem 2: Below is the graph capturing monthly P/B ratio of ten PB ratio percentiles for over a decade:



Then calculate each percentiles' average and plot the bar chart:



As we can see from above bar chart, portfolios with lower PB ratio at last month will tend to generate higher monthly return while the portfolios with high PB ratio at last month will generate less return even lead to loss. This phenomenon can be interpreted by the pricing model. If the PB ratio is low, implying the price of stocks may be undervalued, while PB ratio be overvalued. The price of stocks will tend to regress to their reasonable pricing in the long term. Then, stocks with low PB will increase its price and those with high PB will decrease, which explain the relation illustraed from the above plot.

Below is the implementation code:

```
# first create lagged PB for each stock, implying the last month PB ratio
data = data.sort_values(by=['Code', 'Trading month']).reset_index(drop=True)
data['Lagged PB'] = data.groupby('Code')['Monthly PB'].shift(1)
data.dropna(subset='Lagged PB', inplace=True)

#reorder the data for furthter separation based on PB ratio in each month
# data = data.sort_values(by=['Trading month', 'Monthly PB']).reset_index(drop=True)

# Calculate quantiles separately through time
data['Quantile'] = data.groupby('Trading month')['Lagged PB'].transform(lambda x: pd.qcut(x, q=10, labels=False))

# Group the data by Trading month and Quantile, and calculate the average monthly return
# (suppose we equally weight the stocks in each quantile)
average_return = data.groupby(['Trading month', 'Quantile'])['Monthly return'].mean().reset_index()

# Plot the average monthly return for each quantile
import matplotx
with plt.style.context(matplotx.styles.dracula):
    plt.figure(figsize=(10, 6))
    for quantile in range(10):
        quantile_data = average_return[average_return['Quantile'] == quantile]
        label = f'{quantile * 10}% - {(quantile + 1) * 10}%'
        plt.plot(quantile_data['Trading month'], quantile_data['Monthly return'], label=label, alpha=0.8)
    plt.xlabel('Trading month')
    plt.ylabel('Average Monthly Return')
    plt.title('Average Monthly Return by Quantile of P/B Ratio')
    plt.legend()
    plt.grid(True)
    plt.show()
```

```
# count the average return of each quantile from the panal data
quantile_average_return = average_return.groupby('Quantile')['Monthly return'].mean()
import matplotx
with plt.style.context(matplotx.styles.dracula):
    plt.figure(figsize=(10, 6))
    plt.bar(range(1, 11), quantile_average_return, alpha=0.8, color='skyblue')
    plt.xlabel('Quantile of P/B Ratio')
    plt.ylabel('Monthly average Return')
    plt.title('Average Annual Return by Quantile of P/B Ratio')
    plt.legend()
    plt.grid()
    plt.show()

```