Baruch ML HW 4

Annie Yi, Daniel Tuzes, group 9

March 3, 2025

1 exercise

In a Fama-French 5-factor model, the expected return of a stock is given by the equation:

$$\begin{aligned}\mathbf{R}_i - \mathbf{R}_f = \alpha_i \mathbf{1} + \beta_{i,R_M}(\mathbf{R}_M - \mathbf{R}_f) &+ \beta_{i,\mathrm{SMB}} \cdot \mathbf{SMB} + \\ &+ \beta_{i,\mathrm{HML}} \cdot \mathbf{HML} + \\ &+ \beta_{i,\mathrm{RMW}} \cdot \mathbf{RMW} + \\ &+ \beta_{i,\mathrm{CMA}} \cdot \mathbf{CMA} + \epsilon_i\end{aligned}$$

where:

- $\mathbf{R}_i$ is the return of stock i. We will use Apple's return using Yahoo's financial services.
- $\mathbf{R}_f$ is the risk-free rate, $\mathbf{R}_M$ is the market return (measured by all companies appearing in CRSP, weighted by their market capitalization)
- α_i is the intercept of the regression, representing the idiosyncratic return of stock i.
- β_{i,R_M} is the sensitivity of stock i to the market return.
- $\beta_{i,\mathrm{SMB}}$, $\beta_{i,\mathrm{HML}}$, $\beta_{i,\mathrm{HML}}$ and $\beta_{i,\mathrm{CMA}}$ are the sensitivity of stock i to the SMB (Small Minus Big), HML (High Minus Low), RMW (Robust Minus Weak) and CMA (Conservative Minus Aggressive) factors.

The bold symbols are vectors, having values for different time points. We want to minimise it in the norm of ϵ_i by doing an OLS fitting to every stock i.

From a webpage [1] we can download the data [2] containing the factors Fama-French 5-factor model.

Using Yahoo finance, we can get the returns for a selected stock. As Apple was not available in the Yahoo finance API, we used the stock of Microsoft. Some of the columns for the last few days are shown in table 2.

[1] `http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html`

[2] `http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/ftp/F-F_Research_Data_5_Factors_2x3_daily_CSV.zip`

Table 1: Fama-French 5-factor data tail

	Mkt-RF	SMB	HML	RMW	CMA	RF
20241224	1.11	-0.12	-0.05	-0.13	-0.37	0.017
20241226	0.02	1.09	-0.19	-0.44	0.35	0.017
20241227	-1.17	-0.44	0.56	0.41	0.03	0.017
20241230	-1.09	0.24	0.74	0.55	0.14	0.017
20241231	-0.46	0.31	0.71	0.33	0.0	0.017

Table 2: Microsoft stock data tail

	Open	High	Low	Close	return
Date					
20250225	401.100	401.920	396.700	397.900	-0.015
20250226	398.010	403.600	394.250	399.730	0.005
20250227	401.270	405.740	392.170	392.530	-0.018
20250228	392.660	397.630	386.570	396.990	0.011
20250303	398.820	398.820	386.160	388.490	-0.021

From here, we can calculate the excess return of the stock by subtracting the risk-free rate from the stock's return. Note that the risk-free rate is given in percentage, so we need to divide it by 100. As X, we will use the market's excess return, the SMB, HML, RMW and CMA factors. As y, we will use the excess return of the stock. By doing an OLS fitting, we can get α_i and the betas, which are the coefficients of the regression SMB, HML, RMW and CMA.

The output of the regression is

```
Intercept: 0.00020720675127399466
Coefficients: [ 0.01128495 -0.00369999 -0.00378986  0.00249309 -0.00161599]
R-squared: 0.6079755982101198
```

Using the coefficients, we can calculate the predicted return of the stock as a function of the actual return as shown in figure 1.

We can also show the cumulative excess return of the stock and the predicted cumulative excess return of the stock as a function of time as shown in figure 2. Note that the cumulative excess returns may look very similar, even for models with low R-squared values.

Below the jupyter notebook used to generate this content, as pdf, can be found. To generate the pdf from the jupyter notebook, we used the nbconvert tool. The commands used were:

```
jupyter nbconvert --to latex fitting.ipynb
pdflatex fitting.tex
```

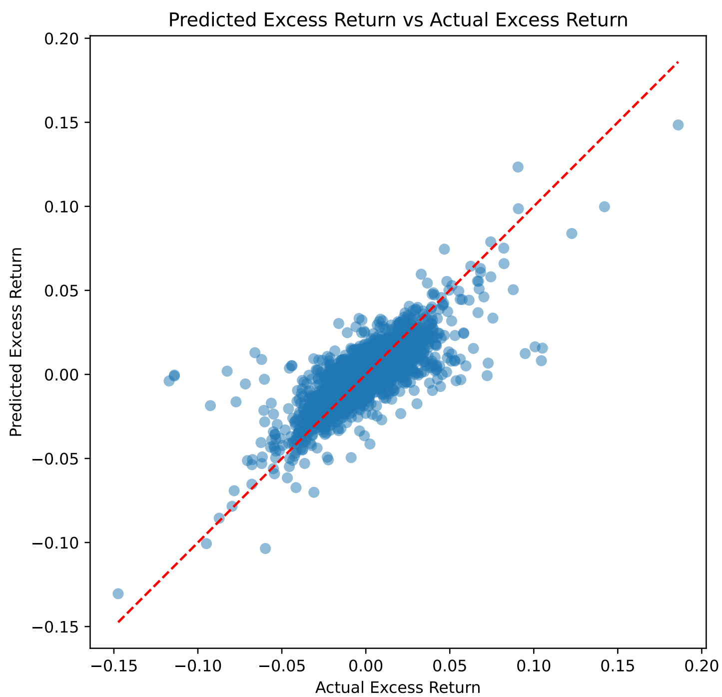


Figure 1: Predicted return of the stock as a function of the actual return

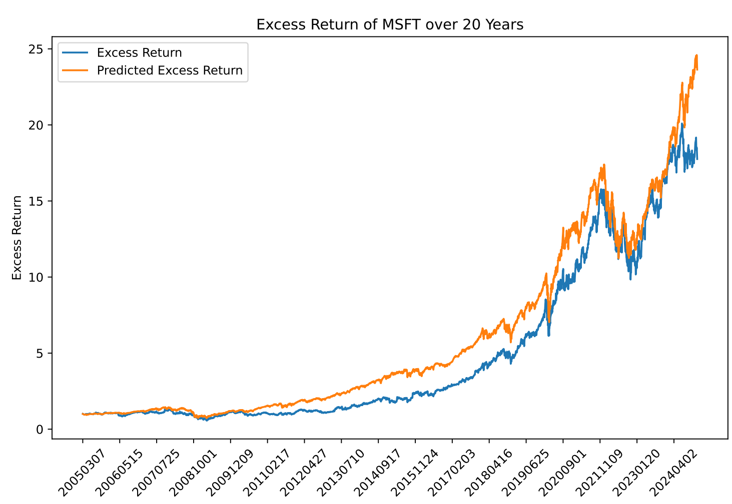


Figure 2: Cumulative excess return of the stock and the predicted excesscumulative return of the stock

fitting

March 3, 2025

1 Fama-French 5 Factor Model

[ ]:
```python
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
import yfinance as yf
import matplotlib.pyplot as plt
```

1.1 get data and preview it

1.1.1 factors for the model

[23]:
```python
fama_fr = pd.read_csv(r"F-F_Research_Data_5_Factors_2x3_daily.CSV", skiprows=3,␣
↪index_col=0)
fama_fr.index =fama_fr.index.astype(str)
fama_fr.tail()
```

[23]:

	Mkt-RF	SMB	HML	RMW	CMA	RF
20241224	1.11	-0.12	-0.05	-0.13	-0.37	0.017
20241226	0.02	1.09	-0.19	-0.44	0.35	0.017
20241227	-1.17	-0.44	0.56	0.41	0.03	0.017
20241230	-1.09	0.24	0.74	0.55	0.14	0.017
20241231	-0.46	0.31	0.71	0.33	0.00	0.017

[73]:
```python
fama_fr_latex = fama_fr.tail().map(lambda x: "{:.2f}".format(x))

# Generate LaTeX table
fama_latex_table = fama_fr_latex.to_latex(escape=False)  # escape=False prevents␣
↪double escaping
print(fama_latex_table)
```

```
\begin{tabular}{lllllll}
\toprule
 & Mkt-RF & SMB & HML & RMW & CMA & RF \\
\midrule
20241224 & 1.11 & -0.12 & -0.05 & -0.13 & -0.37 & 0.02 \\
20241226 & 0.02 & 1.09 & -0.19 & -0.44 & 0.35 & 0.02 \\
20241227 & -1.17 & -0.44 & 0.56 & 0.41 & 0.03 & 0.02 \\
20241230 & -1.09 & 0.24 & 0.74 & 0.55 & 0.14 & 0.02 \\
```

```
20241231 & -0.46 & 0.31 & 0.71 & 0.33 & 0.00 & 0.02 \\
\bottomrule
\end{tabular}
```

1.1.2 pick MSFT as stock

[76]:
```python
stock_yahoo = yf.Ticker("MSFT")
```

[80]:
```python
last20y = stock_yahoo.history(period="20y")
last20y.index = last20y.index.strftime("%Y%m%d")
last20y.drop(columns=["Dividends", "Stock Splits", "Volume"], inplace=True)
last20y["return"] = last20y["Close"].pct_change()
last20y.dropna(inplace=True)
last20y.tail()
```

[80]:

Date	Open	High	Low	Close	return
20250225	401.100006	401.920013	396.700012	397.899994	-0.015099
20250226	398.010010	403.600006	394.250000	399.730011	0.004599
20250227	401.269989	405.739990	392.170013	392.529999	-0.018012
20250228	392.660004	397.630005	386.570007	396.989990	0.011362
20250303	398.820007	398.820007	386.160004	388.489990	-0.021411

[82]:
```python
# use 2 digits after the decimal point
stock_latex = last20y.tail().map(lambda x: "{:.3f}".format(x))

# Generate LaTeX table
stock_latex_table = stock_latex.to_latex(escape=False)  # escape=False prevents␣
↪double escaping
print(stock_latex_table)
```

```
\begin{tabular}{llllll}
\toprule
 & Open & High & Low & Close & return \\
Date &  &  &  &  &  \\
\midrule
20250225 & 401.100 & 401.920 & 396.700 & 397.900 & -0.015 \\
20250226 & 398.010 & 403.600 & 394.250 & 399.730 & 0.005 \\
20250227 & 401.270 & 405.740 & 392.170 & 392.530 & -0.018 \\
20250228 & 392.660 & 397.630 & 386.570 & 396.990 & 0.011 \\
20250303 & 398.820 & 398.820 & 386.160 & 388.490 & -0.021 \\
\bottomrule
\end{tabular}
```

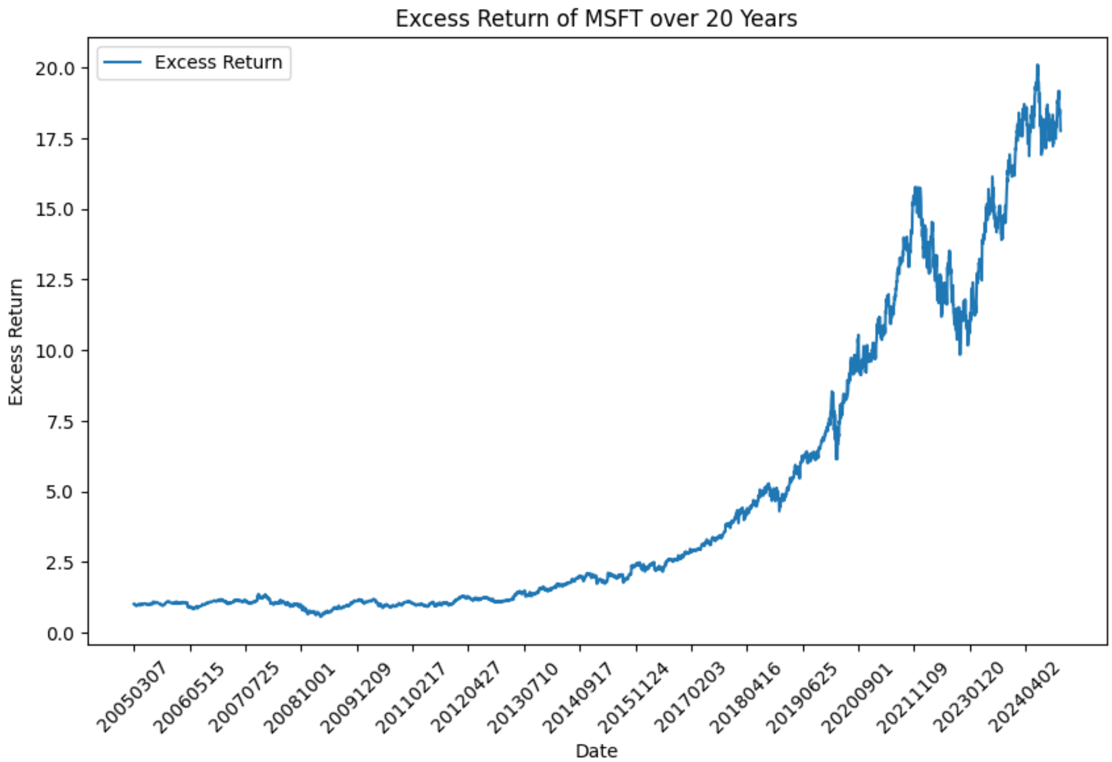
1.1.3 merge and check excess return

[50]:
```python
fama_fr_msft_last20y = pd.merge(
    last20y["return"],
    fama_fr,
    left_index=True,
    right_index=True,
    how="inner"
)
fama_fr_msft_last20y["excess_return"] = fama_fr_msft_last20y["return"] -␣
↪fama_fr_msft_last20y["RF"]/100
fama_fr_msft_last20y
```

[50]:
```
            return  Mkt-RF   SMB   HML   RMW   CMA     RF  excess_return
20050307  0.011920    0.27 -0.43  0.14 -0.33  0.16  0.010       0.011820
20050308 -0.002748   -0.54 -0.38  0.02  0.23 -0.17  0.010      -0.002848
20050309 -0.003543   -0.96  0.00 -0.09 -0.18  0.56  0.010      -0.003643
20050310  0.004741    0.01 -0.84 -0.13  0.01  0.14  0.010       0.004641
20050311 -0.013370   -0.55  0.61  0.42  0.17  0.11  0.010      -0.013470
...            ...     ...   ...   ...   ...   ...    ...            ...
20241224  0.009374    1.11 -0.12 -0.05 -0.13 -0.37  0.017       0.009204
20241226 -0.002777    0.02  1.09 -0.19 -0.44  0.35  0.017      -0.002947
20241227 -0.017302   -1.17 -0.44  0.56  0.41  0.03  0.017      -0.017472
20241230 -0.013240   -1.09  0.24  0.74  0.55  0.14  0.017      -0.013410
20241231 -0.007838   -0.46  0.31  0.71  0.33  0.00  0.017      -0.008008

[4990 rows x 8 columns]
```

[53]:
```python
# plot excess return as function of time
plt.figure(figsize=(10, 6))
plt.plot(fama_fr_msft_last20y.index, (1 + fama_fr_msft_last20y["excess_return"]).
↪cumprod(), label="Excess Return")
plt.title("Excess Return of MSFT over 20 Years")
xtick_indices = np.arange(0, len(fama_fr_msft_last20y.index), 300)
plt.xticks(fama_fr_msft_last20y.index[xtick_indices], rotation=45)
plt.xlabel("Date")
plt.ylabel("Excess Return")
plt.legend()
plt.show()
```



1.2 Do regression with sklearn

[ ]:
```
X = fama_fr_msft_last20y[["Mkt-RF", "SMB", "HML", "RMW", "CMA"]]
y = fama_fr_msft_last20y["excess_return"]
model = LinearRegression(fit_intercept=True)
model.fit(X, y)
print("Intercept:", model.intercept_)
print("Coefficients:", model.coef_)
print("R-squared:", model.score(X, y))
```

```
Intercept: 0.00020720675127399466
Coefficients: [ 0.01128495 -0.00369999 -0.00378986  0.00249309 -0.00161599]
R-squared: 0.6079755982101198
```

1.2.1 show replicated results

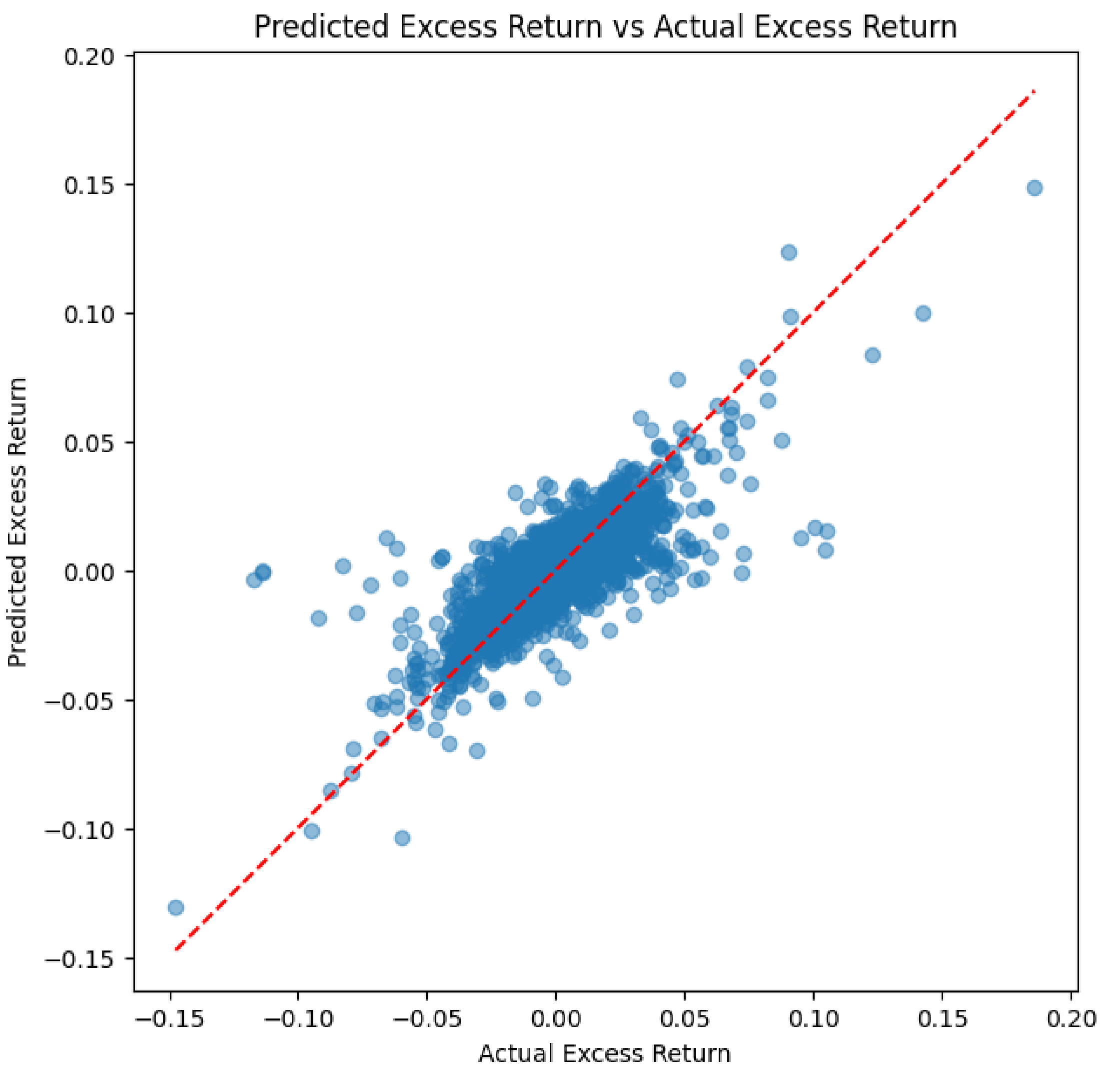
[57]:
```
predicted_Y = model.predict(X)
```

[ ]:
```
# plot the predicted excess return as function of the actual excess return
plt.figure(figsize=(7, 7))  # Ensure the figure itself is square
plt.scatter(fama_fr_msft_last20y["excess_return"], predicted_Y, alpha=0.5)
plt.title("Predicted Excess Return vs Actual Excess Return")
plt.xlabel("Actual Excess Return")
```

```python
plt.ylabel("Predicted Excess Return")

min_val = min(fama_fr_msft_last20y["excess_return"].min(), predicted_Y.min())
max_val = max(fama_fr_msft_last20y["excess_return"].max(), predicted_Y.max())
plt.plot([min_val, max_val], [min_val, max_val], 'r--')

plt.xlim(min_val, max_val)
plt.ylim(min_val, max_val)
plt.axis("equal")  # Ensures the aspect ratio is 1:1
plt.savefig("predicted_vs_actual_excess_return.pdf", bbox_inches='tight')
plt.show()
```



[92]:
```python
# show comulative returns
plt.figure(figsize=(10, 6))
plt.plot(fama_fr_msft_last20y.index, (1 + fama_fr_msft_last20y["excess_return"]).
 ↪cumprod(), label="Excess Return")
plt.plot(fama_fr_msft_last20y.index, (1 + predicted_Y).cumprod(),␣
 ↪label="Predicted Excess Return")
plt.title("Excess Return of MSFT over 20 Years")
xtick_indices = np.arange(0, len(fama_fr_msft_last20y.index), 300)
plt.xticks(fama_fr_msft_last20y.index[xtick_indices], rotation=45)
plt.xlabel("Date")
plt.ylabel("Excess Return")
plt.legend()
plt.savefig("cumulative_excess_return.pdf", bbox_inches='tight')
plt.show()
```