Quantitative Investing: Factor Timing

Quant Case

Systematic Equity Strategies

PGGM Business Course

Maart 2024

13:15 - 16:30

Contents

1 Introductie

PGGM is een pensioenuitvoerder die aan haar klanten, de pensioenfondsen, verschillende diensten aanbiedt. Eén van deze diensten is het beheer van het vermogen van de desbetreffende klant. PGGM Vermogensbeheer verzorgt het pensioen van zo'n 2,9 miljoen deelnemers. Waarvan PFZW ("Pensioenfonds Zorg en Welzijn") de grootste klant is en het meeste vermogen inbrengt. In een gespreide beleggingsportefeuille beheert Vermogensbeheer een bedrag van ruim €240,- miljard aan pensioenvermogen. Binnen deze unit Vermogensbeheer heb je verschillende beleggingsteams verspreid over de afdelingen Public Markets (liquide markten) en Private Markets (illiquide markten).

Eén van de beleggingsteams binnen Public Markets is de afdeling Systematic Equity Strategies (SES). Het team belegt wereldwijd circa 4 miljard euro in beursgenoteerde bedrijven uit ontwikkelde landen. Dit gebeurt op een systematische wijze doormiddel van een factor model. Factor beleggen is een beleggingsstrategie waarbij beleggers zich richten op specifieke kenmerken (of 'factoren') van aandelen. Beleggers kunnen vervolgens portefeuilles construeren die zich concentreren op bepaalde factoren, met als doel het behalen van een hoger rendement dan het marktgemiddelde.

De volgende vijf factoren spelen een belangrijke rol in de huidige academische literatuur en worden ook vaak gebruikt door beleggers in de praktijk:

- **Value** is gebaseerd op de overtuiging dat aandelen die goedkoop geprijsd zijn in verhouding tot een bepaalde fundamentele waarde, beter presteren dan relatief dure aandelen.
- **Momentum** maakt gebruik van het fenomeen dat bedrijven die goed hebben gepresteerd over de afgelopen periodes in het algemeen op de korte termijn ook goed blijven presteren.
- **Quality** refereert naar de neiging van aandelen met doorgaans stabielere inkomsten, sterkere balansen en hogere marges om op de lange termijn beter te presteren dan aandelen van lage kwaliteit.
- **Size** verwijst naar het idee dat bedrijven met een kleine marktkapitalisatie doorgaans beter presteren dan bedrijven met een grote marktkapitalisatie.
- **Low Volatility** richt zicht op aandelen met een lager risico dan de markt, sinds deze historisch gezien hogere risico-gecorrigeerde rendementen hebben opgeleverd.

Het probleem is echter dat portefeuilles gebaseerd op deze individuele factoren erg verschillende rendementen kunnen opleveren in specifieke periodes. Om tot een betere constructie te komen voor de doel portefeuille vraagt SES jullie, als Quant Researchers, de performance van factoren te voorspellen. Aan de hand van de voorspellingen moeten jullie advies kunnen geven over hoeveel jullie in welke factor willen beleggen.

2 Opdrachtomschrijving

Het investment committee (IC) van SES heeft een nieuw beleggingsmandaat voorgesteld dat focust op het implementeren van een long-short Momentum en Value strategie die maandelijks wordt geherbalanceert. De keuze voor deze combinatie van stijl factoren is gekomen uit de waarneming dat zij de neiging hebben om goed te presteren in verschillende fases van de marktcyclus. De hypothese is dat het combineren zal helpen om de lange-termijn prestaties te verbeteren en het volatiliteitsrisico te beheersen.

Het team heeft echter nog één vraag die zij graag willen beantwoorden. Wat is de beste wegingsstrategie voor de twee factoren? In plaats van een gelijke weging wilt het team een dynamisch wegingsschema dat anticipeert op voorspellingen over de komende maand. Als men namelijk zou weten welke factor de hoogste rendementen oplevert in de komende maand, dan zou het winstgevend zijn om die factor portefeuille meer gewicht te geven gedurende die periode.

Er zijn echter twee **regels** vanuit het **nieuwe beleggingsmandaat** waar de strategie zich aan moet houden:

1. Bij afwijking van een gelijk gewogen portefeuille aanpak mag het gegeven over- of ondergewicht naar een individuele factor portefeuille niet een constante zijn.
    - Voorbeeld: niet toegestaan de Momentum factor altijd 60% gewicht te geven.
2. Het verschil in gewicht tussen de Momentum en Value portefeuille voor een maand mag niet groter zijn dan 30 procent punten.
    - Het gewicht moet dus minimaal 35% zijn en het mag maximaal 65% zijn.

Er zijn twee datasets belangrijk voor het beantwoorden van dit vraagstuk.[1] De eerste dataset bestaat uit twee maandelijkse tijdsreeksen, die voor elke maand de rendementen vertegenwoordigen die de Momentum en Value portefeuille gaan maken over de komende maand. Deze data kan als start punt worden gebruikt voor de afhankelijke variabele in jullie voorspellingsmodel. De tweede dataset bestaat uit dagelijkse tijdsreeksen, namelijk historische rendementen van factor portefeuilles en exogene markt data. Deze data kan als start punt worden gebruikt voor de onafhankelijke variabelen in jullie voorspellingsmodel. Tenslotte is er nog één dataset die het wegingsschema representeert, dit zijn twee maandelijkse tijdsreeksen, waarbij als standaard optie beide factoren elke maand een gewicht van 50% zijn toegewezen.

[1] Voor meer informatie zie Appendix A voor een overzicht van de verschillende tijdsreeksen.

Het uiteindelijke doel van de case is dat jullie een dynamische wegingsstrategie bedenken voor het nieuwe beleggingsmandaat (zie het stappenplan voor meer details). Vervolgens geven jullie in een korte presentatie advies over het gebruik van een dynamisch wegingsschema. De presentatie moet inzicht geven over de keuzes die jullie hebben gemaakt met betrekking tot data en modellering, en het gevolg op de prestatie van de portefeuille.

3 Stappenplan

Om jullie op weg te helpen hebben wij een stappenplan opgesteld met daarin concrete handvatten voor het uitvoeren van deze opdracht. Neem het gehele stappenplan goed door voordat je begint, om latere onduidelijkheid en tijdsnoot te voorkomen. Jullie hebben, exclusief het presenteren, immers minder dan drie uur om aan deze case te werken. Verder bevat Appendix B een basis overzicht, voor Python gebruikers, van de scikit-learn package die jullie mogelijk kunnen gebruiken voor het modelleren.

3.1 Data Voorbereiding (45 min)

1. **Exploratory Data Analysis & Cleaning**: Start met het analyseren van de gegeven datasets en doorloop de gebruikelijke data cleaning stappen.

2. **Feature Engineering**: Denk vervolgens na over welke onafhankelijke variabelen jullie mee willen nemen in jullie analyse. Jullie zijn niet beperkt tot de aangeleverde data, wees creatief. Wat voor nieuwe informatie kunnen jullie uit de gegeven data halen? Jullie mogen echter **geen** externe data gebruiken voor jullie analyse.

3. **Resampling**: De eerste dataset (*dependent_variables.csv*) bestaat uit maandelijkse data, terwijl de tweede dataset (*independent_variables.csv*) uit dagelijkse data bestaat. Denk goed na over welke informatie jullie aan het eind van de maand willen hebben om een voorspelling te maken voor de komende maand en welke transformaties hierbij komen kijken.

Om jullie op weg te helpen ontvangen jullie op de dag van de case een notebook met daarin de eerste stappen van Explanatory Data Analysis en Resampling. Enkele manieren van resampling zijn hierin opgenomen, waarvan jullie de code kunnen overnemen. Het staat jullie vrij om deze manieren te gebruiken. Een aanpak anders dan opgenomen is uiteraard ook toegestaan, let hierbij wel op de geringe tijd die beschikbaar is voor het uitvoeren van de gehele opdracht. Uiteindelijk blijft de motivatie voor een specifieke keuze het belangrijkst, ongeacht de manier van resampling die wordt gekozen.

3.1.1 Resampling methoden

Om te resamplen in Python kan je een Pandas DataFrame methode gebruiken die resample heet. De resample methode werkt alleen op time-series data en maakt het mogelijk om een time-series van de ene frequentie om te zetten naar de andere frequentie. De meest gebruikte frequenties zijn **W**: wekelijkse frequentie, **M**: einde van de maand frequentie, **SM**: 15e en einde van de maand en **Q**: einde kwartaal frequentie.

Als je gaat resamplen krijg je een Pandas object. Dit object kan je uitlezen door met een forloop elk item in het object te printen (zie de notebook). De echte kracht zit hem in het doen van berekeningen over die items. Zie de voorbeeld opties hieronder. Voor meer uitleg over de resample functie zie: `https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.resample.html`

1. **Cumulatief product per maand**: De eerste optie die jullie mogelijk zouden willen gebruiken is het omzetten van het dagelijkse rendement naar maandelijks rendement. De onderstaande functie zet de dagelijkse variabelen in dataframe df om naar maandelijkse cumulatieve observaties.

```
# Voorbeeld resample() method:
df.resample('M').apply(lambda x: (1+x).prod()-1)
```

   Voor het verkrijgen van het cumulatieve product per maand wordt een lambda functie gebruikt op de resampled data. In het bovenstaande geval zien we `.apply(lambda x: (1+x).prod()-1)`. De lambda functie zorgt ervoor dat op ieder element x de bewerking `(1+x).prod()-1` wordt uitgevoerd. In dit geval is ieder element x een vector met daarin de dagelijkse observaties over een specifieke maand. De dagelijkse observaties worden eerst verhoogd met 1 om de returns als indexcijfer weer te geven. Vervolgens worden de dagelijkse returns met `.prod()` met elkaar vermenigvuldigd. Als laatste stap wordt het indexcijfer weer omgezet tot procentueel return.

2. **Gemiddelde per maand**: De tweede manier van aggregatie is het nemen van het gemiddelde. Dit kan met:

```
# Voorbeeld resample() method:
df.resample('M').mean()
```

3. **Laatste waarde van de maand**: De derde optie is het gebruiken van de laatste observatie iedere maand. Dit kan met:

```
# Voorbeeld resample() method:
df.resample('M').apply('last')
```

4. **Combinatie van resampling**: Als vierde optie kan een combinatie van de eerdergenoemde resampling methode worden gebruikt. Een voorbeeld hiervan is opgenomen in de code die met jullie wordt gedeeld.
5. **Eigen idee**: Een eigen manier van resampling is natuurlijk toegestaan.
6. **Geen resampling**: Ook is het mogelijk om geen resampling te gebruiken.

Het is belangrijk dat jullie kunnen onderbouwen waarom jullie wel of niet geresampled hebben naar maandelijkse data (er is geen goed of fout).

3.2 Modellering (45 min)

In deze fase gaan jullie je eigen voorspellingsmodel(len) bouwen. Het is belangrijk dat jullie duidelijk hebben wat jullie gaan voorspellen. De eerste dataset bestaat namelijk uit toekomstige rendements-data van de Momentum portefeuille en Value portefeuille. Het einddoel is bepalen hoeveel van het totale vermogen gaat worden belegd in de Momentum factor en hoeveel in de Value factor.

Een mogelijke optie is om twee aparte modellen te bouwen waar één Momentum rendementen voorspelt en de ander Value rendementen. Een andere optie is om er een classificatie probleem van te maken en dus jullie eigen target variable te creëren. Dit zijn echter maar twee voorbeelden, wees creatief, maar houdt het einddoel in gedachten (vertaalslag naar portefeuille gewichten).

De data beslaat meerdere decennia, waarin verschillende economische regimes hebben plaatsgevonden. Wij raden het daarom aan om gebruik te maken van een rolling window. Het is hierbij belangrijk dat jullie voor een individuele voorspelling alleen data gebruiken die op dat moment in de tijd beschikbaar was, om zo genoemde *look-ahead bias* te voorkomen.

De afbeelding hieronder is een voorbeeld van de data opgeslagen in *dependent_variables.csv*, dit zijn de maandelijkse *forward returns* van de individuele factor portefeuilles. Dat wil zeggen, de Momentum observatie op 1997-02-28 geeft aan wat het cumulatieve rendement van de Momentum portefeuille is over 1997-03-01 tot en met 1997-03-31. Stel jullie gaan exact de *forward returns* voorspellen, dan zouden jullie voor deze observatie alleen data tot en met 1997-02-28 moeten gebruiken.

Date	1997-02-28	1997-03-31	1997-04-30	1997-05-31	1997-06-30	1997-07-31	1997-08-31	1997-09-30	1997-10-31	1997-11-30
Momentum	-0.002132	0.042500	-0.037883	0.025762	0.034029	-0.037878	0.011715	0.007300	-0.033398	0.014490
Value	0.030542	-0.026738	-0.002677	-0.016978	0.006114	0.050161	0.012717	-0.020454	-0.017455	0.016265

De data in *independent_variables.csv* bevat ook rendementen van factor portefeuilles, echter stelt hier de Momentum observatie op 1997-02-28 het rendement voor op die dag. Om verwarring te

voorkomen is het eventueel handig om de namen van de kolommen in de DataFrames aan te passen. Dus de factor rendementen in *dependent_variable.csv* zijn forward returns over de komende maand en de factor rendementen in *independent_variable.csv* zijn rendementen van die dag.

3.3 Wegingsschema (30 min)

Nu is het tijd om de model output te vertalen naar een praktische implementatie, namelijk jullie strategie met betrekking tot het (dynamische) wegingsschema. De afbeelding hieronder is een voorbeeld van de data opgeslagen in *weights_default.csv*. Deze heeft dezelfde datum index als de data in *dependent_variable.csv*.

Date	1997-02-28	1997-03-31	1997-04-30	1997-05-31	1997-06-30	1997-07-31	1997-08-31	1997-09-30	1997-10-31	1997-11-30
Momentum	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
Value	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5

Aan jullie simpel weg de taak om deze gewichten aan te passen op basis van jullie model inzichten/voorspelling(en) voor de volgende maand. Dat wil zeggen, het gewicht op eind februari (1997-02-28) wordt aangehouden gedurende de komende maand. Verder laat het volgende voorbeeld zien hoe jullie op basis van de eerder genoemde forward returns en het wegingsschema tot de rendementen van de uiteindelijke doel portefeuille kunnen komen.

```
# Voorbeeld constructie doel portefeuille:
strategy = returns.mul(weights).sum(axis=1) # DataFrames hebben dezelfde kolomnamen.

# Voorbeeld visualisatie van rendementen:
strategy.cumsum().plot(grid=True)
```

Het is niet een verplichting dat jullie wegingsstrategie begint op 1997-02-28. Gegeven jullie voorgestelde model raamwerk is het ook toegestaan om een latere startdatum te gebruiken. Wees er wel van bewust dat de robuustheid van de strategie minder goed te beoordelen is met minder observaties.

3.3.1 Inleveren wegingsschema

Graag willen wij alle groepen vragen om het wegingsschema op te slaan als csv in hetzelfde format als *weights_default.csv* met als naam: *pggm_quant_case_groep_x.csv* waarbij x het groepsnummer is. Dit bestand moeten jullie via de mail delen met de begeleider van de case alvorens we aan de presentaties beginnen.

3.4 Presentatie (15 min)

Bereid een korte presentatie voor van rond de **5 minuten**. Het is belangrijk dat jullie uitleggen welke keuzes jullie hebben gemaakt in (1) Data voorbereiding, (2) Modellering en (3) Wegingsschema. De toelichting van jullie keuzes is belangrijker dan het uiteindelijke resultaat.

A Data Overzicht

Niet beschikbaar: de data wordt pas op de dag van de case gedeeld.

B Scikit-learn Syntax

De standaard modellering syntax van de scikit-learn package is zeer intuïtief en consistent (zie voorbeeld). Start door het gewenste model van de scikit-learn library te importeren uit de bijbehorende module. Daarna definieer je een object van het model en specifieer je de gewenste parameters. Raadpleeg de bijbehorende documentatie van het model om meer inzicht te krijgen over de mogelijke parameters. Vervolgens train je het model doormiddel van de **fit()** functie, op basis van de gekozen observaties **X** en de bijbehorende target variabele **y**. Als het model klaar is met trainen, kun je voorspellingen doen met gebruik van de **predict()** functie.

```
# General Scikit-learn Modelling Syntax:
from sklearn.MODULE import MODEL
mdl = MODEL()                       # Initialize model and set hyperparameters.
mdl.fit(X_train, y_train)           # Train model using both X and y.
pred = model.predict(X_test)        # Predict new y observations.
```

Hieronder is een basis voorbeeld te zien van zowel een *OLS* model als een Random Forest Regressor.

```
# Scikit-learn - Linear Regression Example:
from sklearn.linear_model import LinearRegression
mdl = LinearRegression(fit_intercept=True, normalize=True)
mdl.fit(X_train, y_train)          # mdl.coef_ returns betas
pred = model.predict(X_test)

# Scikit-learn - Random Forest Example:
from sklearn.ensemble import RandomForestRegressor
mdl = RandomForestRegressor(n_estimators=50, max_depth=10, random_state=42)
mdl.fit(X_train, y_train)          # mdl.feature_importance returns ranking
pred = model.predict(X_test)
```

De **model_selection** module van scikit-learn bevat verschillende manieren om de data te verdelen, het model te valideren en de parameters te fine-tunen. In de bovenstaande voorbeelden is gebruik gemaakt van een relatief simpele split doormiddel van de **train_test_split** functie:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33)
```

Tenslotte, is er ook de statsmodels package die veel wordt gebruikt door data-scientists. Het heeft een soortgelijke syntax als scikit-learn, maar is vooral gericht op meer traditionele econometrische modellen en niet op machine learning. Het bied echter wel een visualisatie van de model resultaten die te vergelijken is met output uit programma's zoals Eviews en Stata.