Backtesting Random Forest II

MGMT 638: Data-Driven Investments: Equity

Kerry Back, Rice University

Outline

Follow same procedure as in 06a-random_forest_backtest except

- Multiply all features by mktvol
    - So twice as many features: original plus original*mktvol
- Use percentile (rank from 0 to 100) of return instead of (return - median) as target
- Use maxdepth=4 instead of maxdepth=3

Read data

In [49]:

```
import pandas as pd

url = "https://www.dropbox.com/scl/fi/hjpebns5qv0nzh1ucl4tr/data-2023-11-13.c
df = pd.read_csv(url)
df.head()
```

Out[49]:

	ticker	date	marketcap	pb	ret	mom	volume	volatility
0	AACC	2011-01-14	188.3	1.4	-0.014634	-0.184615	2.078000e+04	0.071498
1	AAI	2011-01-14	1012.1	2.0	0.002677	0.438224	2.775580e+06	0.128450
2	AAIC	2011-01-14	189.3	1.0	-0.010119	0.684547	3.466000e+04	0.048505
3	AAON	2011-01-14	479.4	4.2	0.007778	0.528685	2.817291e+05	0.044912
4	AATC	2011-01-14	63.3	1.4	-0.013960	0.008216	6.800000e+03	0.049756

Define model and target

In [50]:

```
from sklearn.ensemble import RandomForestRegressor
forest = RandomForestRegressor(max_depth=4)

df["target"] = df.groupby("date", group_keys=False).ret.apply(
    lambda x: x.rank(pct=True)
)
```

Define predictors (features)

In [51]:

```python
features = [
    "marketcap",
    "pb",
    "mom",
    "volume",
    "volatility",
    "roe",
    "accruals",
    "agr"
]
features.sort()
```

In [52]:

```
for x in features:
    df[x+"_vol"] = df[x]*df.mktvol

features += [x+"_vol" for x in features]
```

Define training dates and training windows

- For this example, I am going to train once per year using the prior three years of data.
- Obviously, other choices are possible.
- The reason for not using all past data is to capture any changes in the market.

In [53]:

```
dates = list(df.date.unique())
dates.sort()
train_dates = dates[156::52]  # once per year starting after three years

past_dates = {}               # dates on which to train for each training dat
future_dates = {}             # dates for which to predict for each training 
for date in train_dates:
    past_dates[date] = dates[(dates.index(date)-156):dates.index(date)]
    if date < train_dates[-1]:
        future_dates[date] = dates[dates.index(date):(dates.index(date)+52)]
    else:
        future_dates[date] = dates[dates.index(date):]
```

Run the loop

In [54]:

```
new_data = None
for date in train_dates:
    past = past_dates[date]
    past = df[df.date.isin(past)]
    future = future_dates[date]
    future = df[df.date.isin(future)]
    forest.fit(X=past[features], y=past.target)
    predictions = forest.predict(X=future[features])
    predictions = pd.DataFrame(predictions)
    predictions.columns = ["predict"]
    for col in ["ticker", "date"]:
        predictions[col] = future[col].to_list()
    new_data = pd.concat((new_data, predictions))

df = df.merge(new_data, on=["ticker", "date"], how="inner")
```

In [55]:
```
df.tail()
```

Out[55]:

	ticker	date	marketcap	pb	ret	mom	volume	volatility
1010370	ZNTL	2023-11-10	1262.5	2.4	-0.449664	-0.174662	743655.8	0.08655:
1010371	ZUMZ	2023-11-10	335.0	0.9	-0.069190	-0.245402	201904.4	0.05363:
1010372	ZUO	2023-11-10	1088.9	9.7	0.018065	0.080163	662494.2	0.07031i
1010373	ZYME	2023-11-10	504.0	1.1	-0.018843	-0.215539	435386.8	0.06276ɛ
1010374	ZYXI	2023-11-10	310.4	5.3	-0.023484	-0.356304	379338.0	0.06636:

5 rows × 23 columns

Form portfolios and compute returns

In [56]:

```
df["rnk_long"] = df.groupby("date", group_keys=False).predict.rank(
    ascending=False,
    method="first"
)
df["rnk_short"] = df.groupby("date", group_keys=False).predict.rank(
    ascending=True,
    method="first"
)
longs = df[df.rnk_long<=44]
shorts = df[df.rnk_short<=44]
```

In [57]:

```
long_ret = longs.groupby("date").ret.mean()
short_ret = shorts.groupby("date").ret.mean()
print(f"mean annualized long return is {52*long_ret.mean():.2%}")
print(f"mean annualized short return is {52*short_ret.mean():.2%}")
```

```
mean annualized long return is 21.60%
mean annualized short return is -27.99%
```

Try sector-neutral strategy

In [58]:

```
df["rnk_long"] = df.groupby(["date", "sector"], group_keys=False).predict.ran
    ascending=False,
    method="first"
)
df["rnk_short"] = df.groupby(["date", "sector"], group_keys=False).predict.ra
    ascending=True,
    method="first"
)
longs = df[df.rnk_long<=4]
shorts = df[df.rnk_short<=4]
```

In [59]:

```
long_neutral_ret = longs.groupby("date").ret.mean()
short_neutral_ret = shorts.groupby("date").ret.mean()
print(f"mean annualized long sector-neutral return is {52*long_neutral_ret.me
print(f"mean annualized short sector-neutral return is {52*short_neutral_ret.
```

```
mean annualized long sector-neutral return is 20.04%
mean annualized short sector-neutral return is -17.89%
```

Plot long-minus-short returns

In [60]:

```
lms = long_ret - short_ret
lms_neutral = long_neutral_ret - short_neutral_ret

lms.index = pd.to_datetime(lms.index)
lms_neutral.index = pd.to_datetime(lms_neutral.index)

import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style("whitegrid")

(1+lms).cumprod().plot(logy=True, label="long minus short")
(1+lms_neutral).cumprod().plot(logy=True, label="neutral long minus short")
plt.legend()
plt.show()
```