03 · Backtest & Visualization

Validate the SPX-focused ML signals using a vectorized long/short simulator and visualize equity curves.

Goals

- Load cached features, prices, and trained model
- Generate probability-driven signals and flip into positions
- Run a fast vectorized backtest with transaction costs
- Visualize performance metrics and sanity-check exposures

In [1]:
```
from pathlib import Path
import sys

import joblib
import pandas as pd
import plotly.express as px
import seaborn as sns
import matplotlib.pyplot as plt

PROJECT_ROOT = Path("..").resolve()
if str(PROJECT_ROOT) not in sys.path:
    sys.path.append(str(PROJECT_ROOT))

from momentum_lib import (
    bootstrap_env,
    generate_signals,
    backtest_signals,
)

sns.set_theme(style="whitegrid")
bootstrap_env(Path("../.env"))
data_dir = Path("../data")
features = pd.read_csv(data_dir / "features.csv", parse_dates=[0], index_col=0)
prices = pd.read_csv(data_dir / "prices.csv", parse_dates=[0], index_col=0)
model = joblib.load(data_dir / "uprx_model.joblib")

overlap = features.index.intersection(prices.index)
X = features.loc[overlap]
prices = prices.loc[overlap]
print(f"Aligned frame shape: {X.shape}")
```

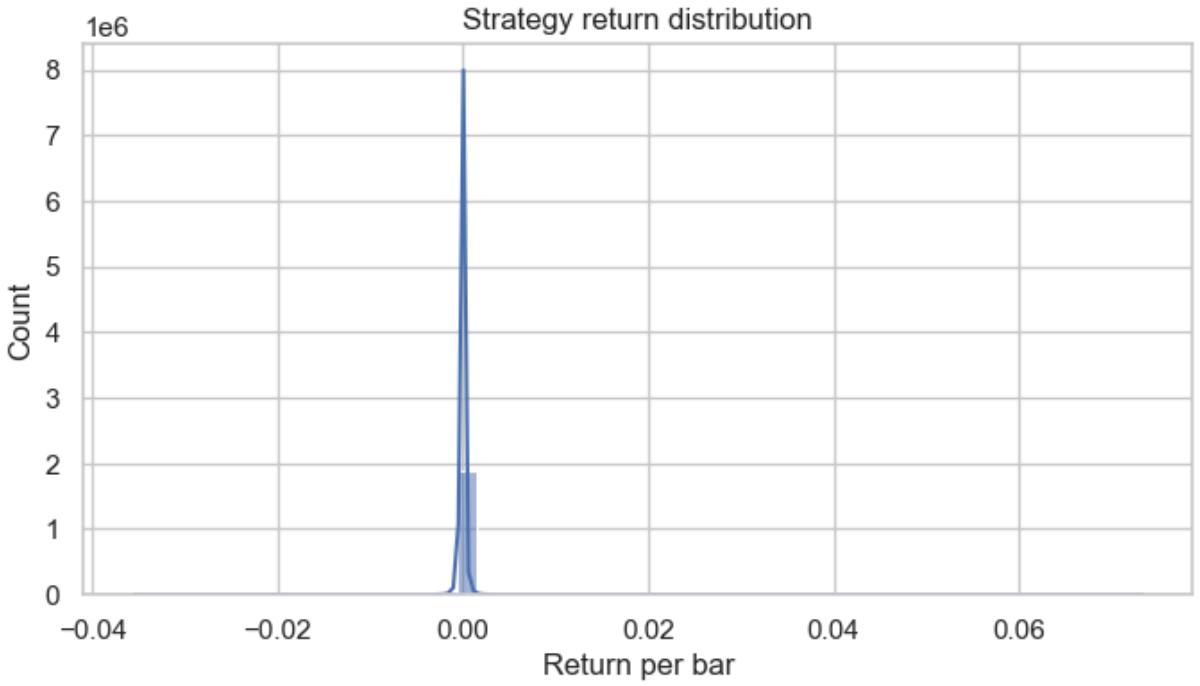
```
Aligned frame shape: (1943340, 6)
```

In [2]:
```
signals = generate_signals(model, X, threshold=0.55)
results = backtest_signals(prices, signals)
results.head()
```

Out[2]:

	equity	strategy_return
2019-01-01 18:14:00-05:00	100000.000000	0.000000
2019-01-01 18:15:00-05:00	99992.051727	-0.000079
2019-01-01 18:16:00-05:00	99980.129318	-0.000119
2019-01-01 18:17:00-05:00	99999.960259	0.000198
2019-01-01 18:18:00-05:00	99972.141304	-0.000278

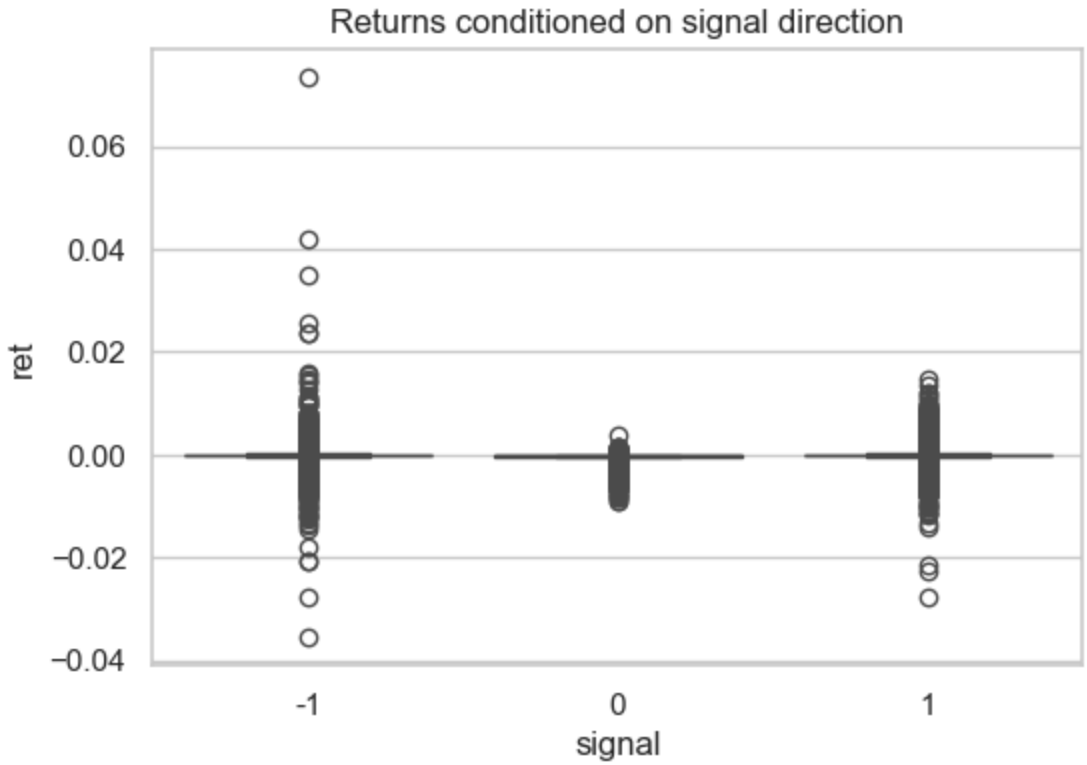
In [3]:
```
plt.figure(figsize=(8, 4))
sns.histplot(results["strategy_return"], bins=50, kde=True)
plt.title("Strategy return distribution")
plt.xlabel("Return per bar")
plt.show()
```



In [4]:
```
results.attrs["stats"]
```

Out[4]:
```
{'sharpe': np.float64(-43.67730542268217),
 'cagr': np.float64(-0.9939658091681602),
 'max_drawdown': -1.0}
```

In [5]:
```
signal_frame = pd.DataFrame({"signal": signals, "ret": results["strategy_return"]})
plt.figure(figsize=(6, 4))
sns.boxplot(data=signal_frame, x="signal", y="ret")
plt.title("Returns conditioned on signal direction")
plt.show()
```



In [6]:
```
fig = px.line(results.reset_index(), x="index", y="equity", title="Strategy Equity
fig.show()
```

In [7]:
```
signals.value_counts(normalize=True).rename("signal_share")
```

Out[7]:
```
 1    0.451760
-1    0.440682
 0    0.107558
Name: signal_share, dtype: float64
```