

Building logical solutions for automated trading

WINNERS OR LOSERS

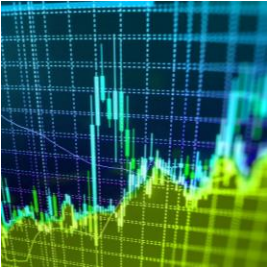
MACHINE LEARNING MODEL STRATEGIES FOR DAY TRADING

FABIO RUBINO ARVID JOHANSSON JOEL SÖDERBERG

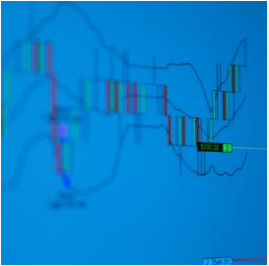
PROJECT GOAL



Building logical solutions for automated trading



Using ML to predict Win-rate from historical data



Focus: daily trading (buy open, sell close)

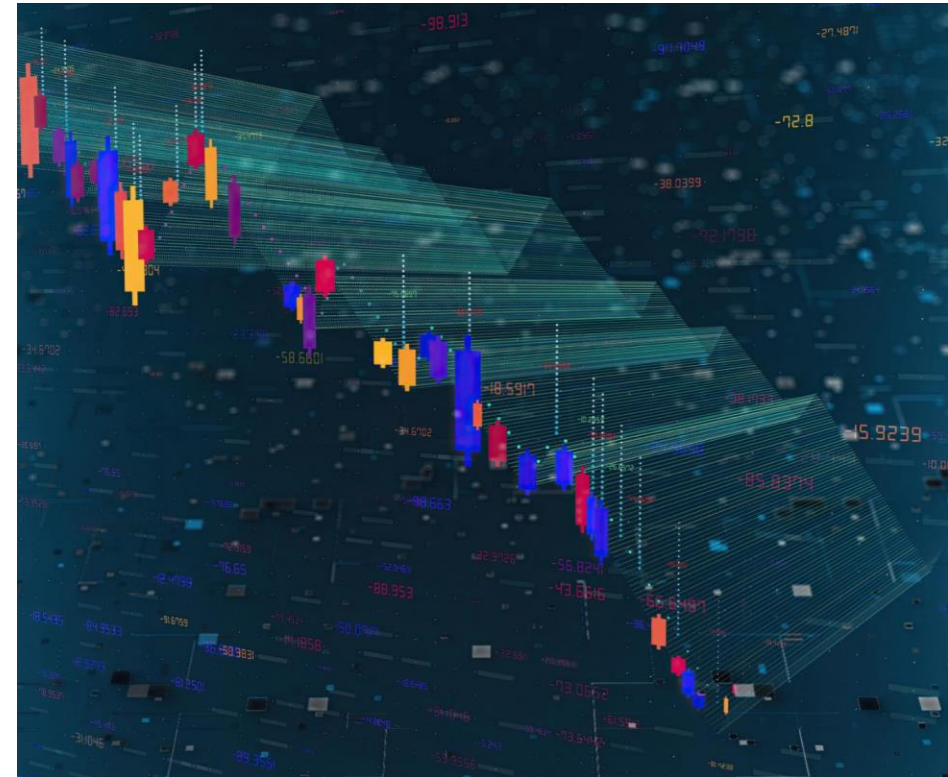
“If you want to beat the market, start by understanding it.”

Our Asset: SPY ETF

- Widely traded ETF, listed on NYSE
- Pays dividends every 3 months
- Chosen for liquidity & availability

Day Trading Strategy

- Buy SPY at market opening
- Sell SPY at market closing
- Train ML model on historical performance with different features
- Predict daily win-rates



STEP 1: GOOGLE CLOUD SETUP

**Creating accounts, permissions,
roles**



**Updating permissions = boring but
necessary 🤖**

**Then moved to GitHub for
collaboration**



DATA SOURCE

- Chosen API: Polygon (free & reliable)
- Delivered all market data we needed
- Alternative APIs were costly or limited

```
Losers-or-Winners > data >  stock_data.csv >  data
```

1	ticker,date,open,close,volume
2	SPY,2024-09-03,560.47,552.08,60600113.0
3	SPY,2024-09-04,550.2,550.95,47224939.0
4	SPY,2024-09-05,550.89,549.61,44264258.0
5	SPY,2024-09-06,549.94,540.36,68493805.0
6	SPY,2024-09-09,544.65,546.41,40445822.0
7	SPY,2024-09-10,548.36,548.79,36394579.0
8	SPY,2024-09-11,548.7,554.42,75248608.0
9	SPY,2024-09-12,555.01,559.09,51892735.0
10	SPY,2024-09-13,559.71,562.01,39310501.0
11	SPY,2024-09-16,561.74,562.84,36656122.0

PIPELINE STRUCTURE

Our Orchestrator: Composer

Dag 1 : Ingest

Fetch data from API

Upload to GCS

Store Metadata

BigQuery Load

Dag 2: Predict

Read from BigQuery

Load model from GCS

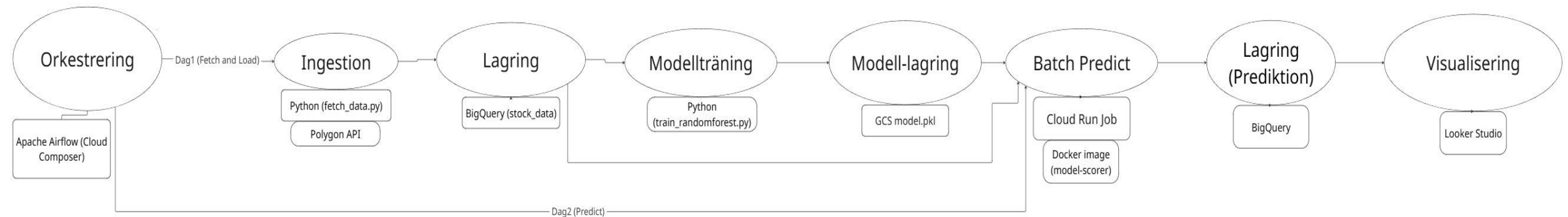
Feature Engineering

Run predictions

Save predictions CSV

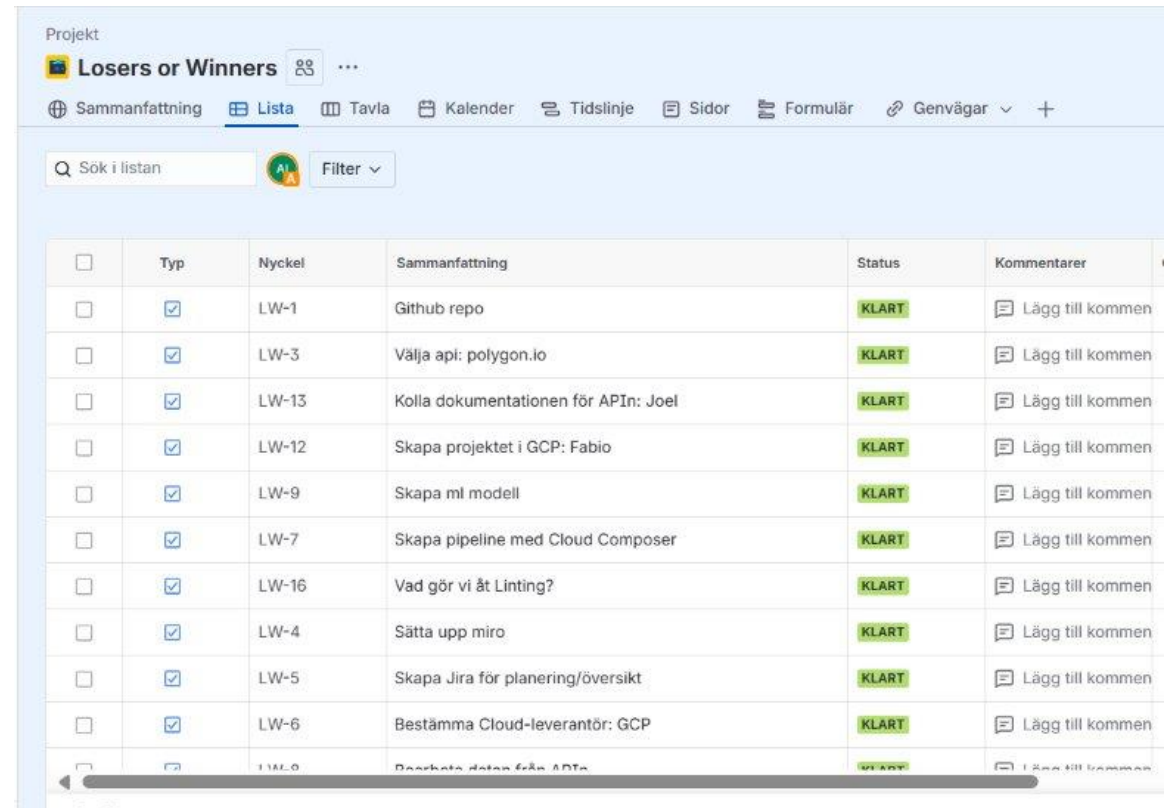
Store in BigQuery

Winners or Losers: Pipeline



AGILE FRAMEWORKS

- ✓ Weekly sprints illustrated on KanBan Board via Jira.
- ✓ Shared Word doc for daily progress
- ✓ Continuous Stand-Ups to inform about progress and problems.
- ✓ Dividing tasks between each other each week.
- ✓ One Project Owner and rotating Scrum Master.
- ✓ Using Miro for pipeline visualization
- ✓ **Documentation = survival !!**

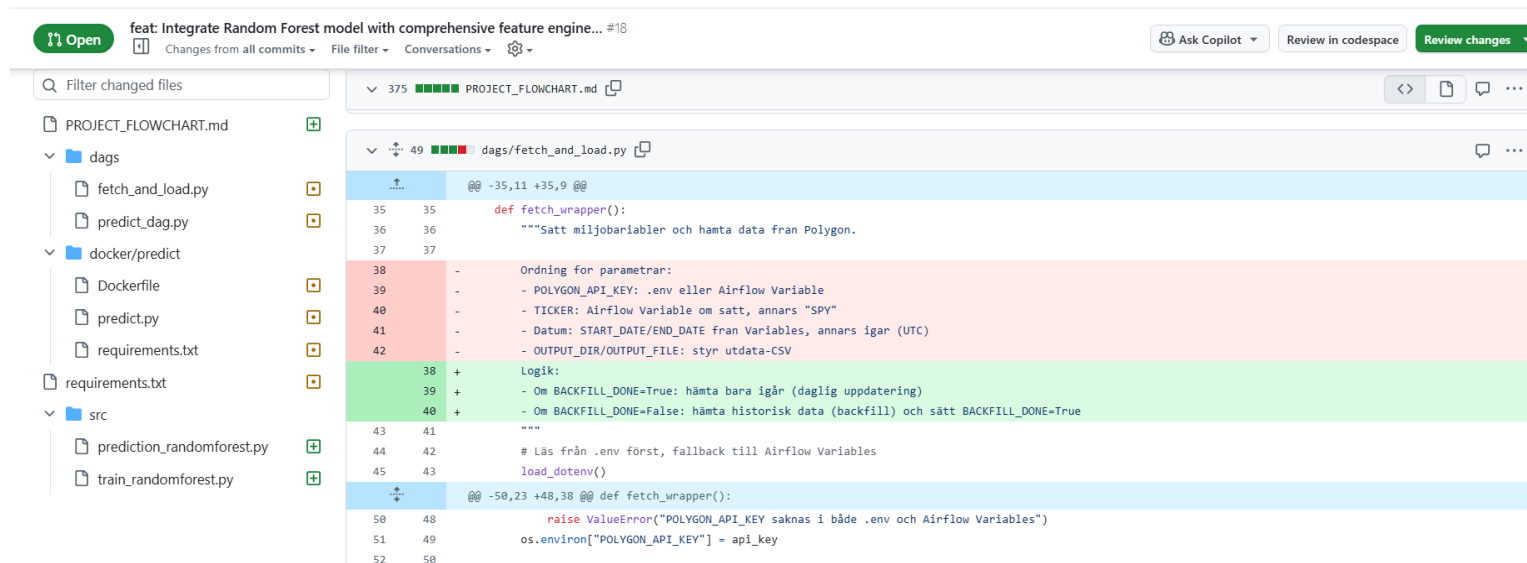


The screenshot shows a Jira project page for 'Losers or Winners'. The interface includes a navigation bar with options like 'Sammanfattning', 'Lista', 'Tavla', 'Kalender', 'Tidslinje', 'Sidor', 'Formulär', and 'Genvägar'. Below the navigation bar is a search bar and a filter dropdown. The main content is a table with columns: 'Typ', 'Nyckel', 'Sammanfattning', 'Status', and 'Kommentarer'. The table lists several tasks, all with a status of 'KLART' (Done).

Typ	Nyckel	Sammanfattning	Status	Kommentarer
<input checked="" type="checkbox"/>	LW-1	Github repo	KLART	Lägg till kommentar
<input checked="" type="checkbox"/>	LW-3	Välja api: polygon.io	KLART	Lägg till kommentar
<input checked="" type="checkbox"/>	LW-13	Kolla dokumentationen för API:n: Joel	KLART	Lägg till kommentar
<input checked="" type="checkbox"/>	LW-12	Skapa projektet i GCP: Fabio	KLART	Lägg till kommentar
<input checked="" type="checkbox"/>	LW-9	Skapa ml modell	KLART	Lägg till kommentar
<input checked="" type="checkbox"/>	LW-7	Skapa pipeline med Cloud Composer	KLART	Lägg till kommentar
<input checked="" type="checkbox"/>	LW-16	Vad gör vi åt Linting?	KLART	Lägg till kommentar
<input checked="" type="checkbox"/>	LW-4	Sätta upp miro	KLART	Lägg till kommentar
<input checked="" type="checkbox"/>	LW-5	Skapa Jira för planering/översikt	KLART	Lägg till kommentar
<input checked="" type="checkbox"/>	LW-6	Bestämma Cloud-leverantör: GCP	KLART	Lägg till kommentar
<input checked="" type="checkbox"/>	LW-8	Bestämma data från API:n	KLART	Lägg till kommentar

GITHUB & COLLABORATION:

- All commits with comments before merge
- Linting & clean code enforced
- Full transparency across team members





The screenshot displays a GitHub pull request interface. At the top, the pull request title is "feat: Integrate Random Forest model with comprehensive feature engine... #18". The interface shows a list of files on the left, including "PROJECT_FLOWCHART.md", "dags", "docker/predict", and "src". The main area shows a diff for the file "PROJECT_FLOWCHART.md". The diff highlights changes in the "dags" directory, specifically in the "fetch_and_load.py" file. The changes include a new function "def fetch_wrapper()" and a new variable "api_key". The diff is color-coded: green for additions, red for deletions, and blue for context. The interface also includes a search bar for changed files, a file filter, and a "Review changes" button.

```
35 35 def fetch_wrapper():
36 36     """Sätt miljövariabler och hämta data från Polygon.
37 37
38 38 - Ordning för parametrar:
39 39 - POLYGON_API_KEY: .env eller Airflow Variable
40 40 - TICKER: Airflow Variable om satt, annars "SPY"
41 41 - Datum: START_DATE/END_DATE från Variables, annars igår (UTC)
42 42 - OUTPUT_DIR/OUTPUT_FILE: styr utdata-CSV
43 43 + Logik:
44 44 + - Om BACKFILL_DONE=True: hämta bara igår (daglig uppdatering)
45 45 + - Om BACKFILL_DONE=False: hämta historisk data (backfill) och sätt BACKFILL_DONE=True
46 46
47 47 + """
48 48 +
49 49 + # Läs från .env först, fallback till Airflow Variables
50 50 + load_dotenv()
51 51 +
52 52 + def fetch_wrapper():
53 53 +     raise ValueError("POLYGON_API_KEY saknas i både .env och Airflow Variables")
54 54 +     os.environ["POLYGON_API_KEY"] = api_key
```

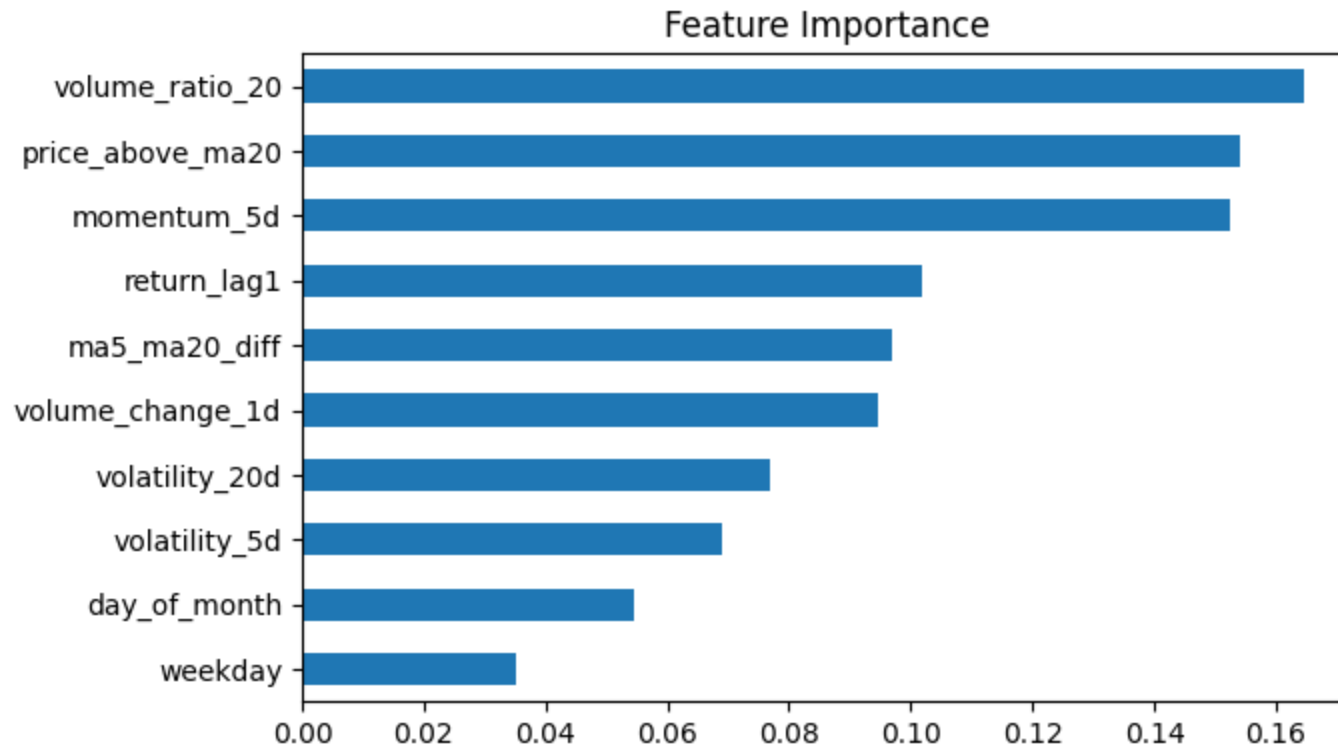
FIRST ML MODEL: LOGISTIC REGRESSION

- Simple, quick baseline
- Predict win-rate from historical data
- Problems: low accuracy, overfitting risk. Worked only on tot. daily win-rate
- **“Not perfect, but a starting point.”**

```
 Baslinje (köp varje dag, sälj vid stängning):  
Totalt antal dagar: 251  
Vinstdagar: 135  
Förlustdagar: 116  
Winrate: 53.78%
```

```
 Winrate per veckodag (köp vid öppning, sälj vid stängning):  
Monday      : 58.33% (28 vinstdagar, 20 förlustdagar)  
Tuesday      : 56.6% (30 vinstdagar, 23 förlustdagar)  
Wednesday    : 54.9% (28 vinstdagar, 23 förlustdagar)  
Thursday     : 42.86% (21 vinstdagar, 28 förlustdagar)  
Friday       : 56.0% (28 vinstdagar, 22 förlustdagar)
```

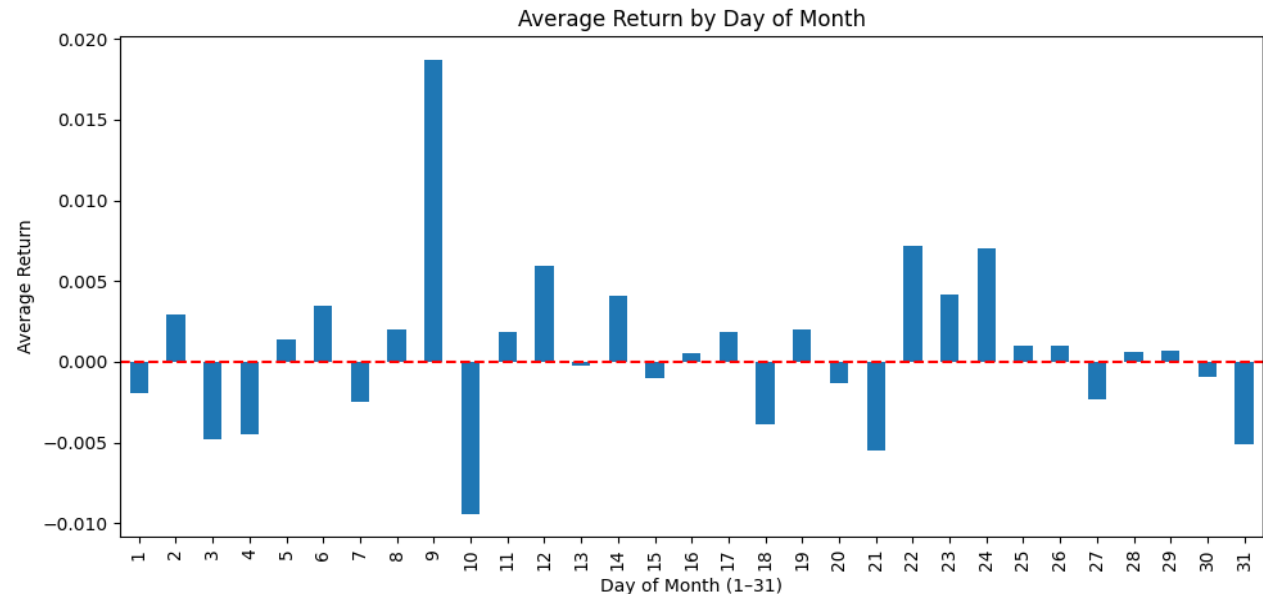
ADVANCED ML MODEL: RANDOM FOREST



- We implemented a Random Forest classifier to predict daily SPY win-rate.
- It uses ~10 engineered features (momentum, volatility, moving averages, volume,etc etc).
- The model reduces overfitting compared to linear regression and captures non-linear relationships.
- Achieved stable predictions on historical intraday data, forming the base for future improvements.

FEATURES RANDOM FOREST

- weekday – day-of-week effect (Monday to Friday)
- day_of_month – calendar patterns (Calendar day)
- return_lag1 – momentum/mean-reversion
- Volatility 20d – risk regimes
- momentum_5d – short-term trend
- price_above_ma20 – medium-term trend
- ma5_ma20_diff – moving average crossover
- volume_ratio_20 – market participation
- volume_change_1d – shocks/novelties



RESULTS

- Working model with ~10 features
- Predictive capability on historical data
- Scalable with more APIs, features, assets
- Feature importance chart as visualization



WHAT WE LEARNED

Routine Tasks



Checklists

- ✓ Agile teamwork = progress
- ✓ Documentation is mandatory, already from the very beginning
- ✓ Daily check-ins solve problems faster
- ✓ **Composer is awesome but costly** 🤪

Complex Processes



Flowcharts & SOPs

FINAL THOUGHTS

- ✓ Achieved all goals (and even more, since we introduced one more model that wasn't part of the plan)
- ✓ Pipeline fully functional
- ✓ Future: Docker + Workflow to replace Composer
- ✓ “This was just a school project, but we built it like a startup.” 