How I used a Random Forest Classifier to Day Trade for 2 Months — Part 2

An end-to-end machine learning project.

Michael
Jan 4 · 6 min read ★

Photo by Joshua Hoehne on Unsplash

Part 1 — Model Summary (Article)

Part 2 — Model Deployment & Trading Results

TL;DR

👾 See the code: GitHub

💻 Try the web app: Demo

Get in touch: LinkedIn

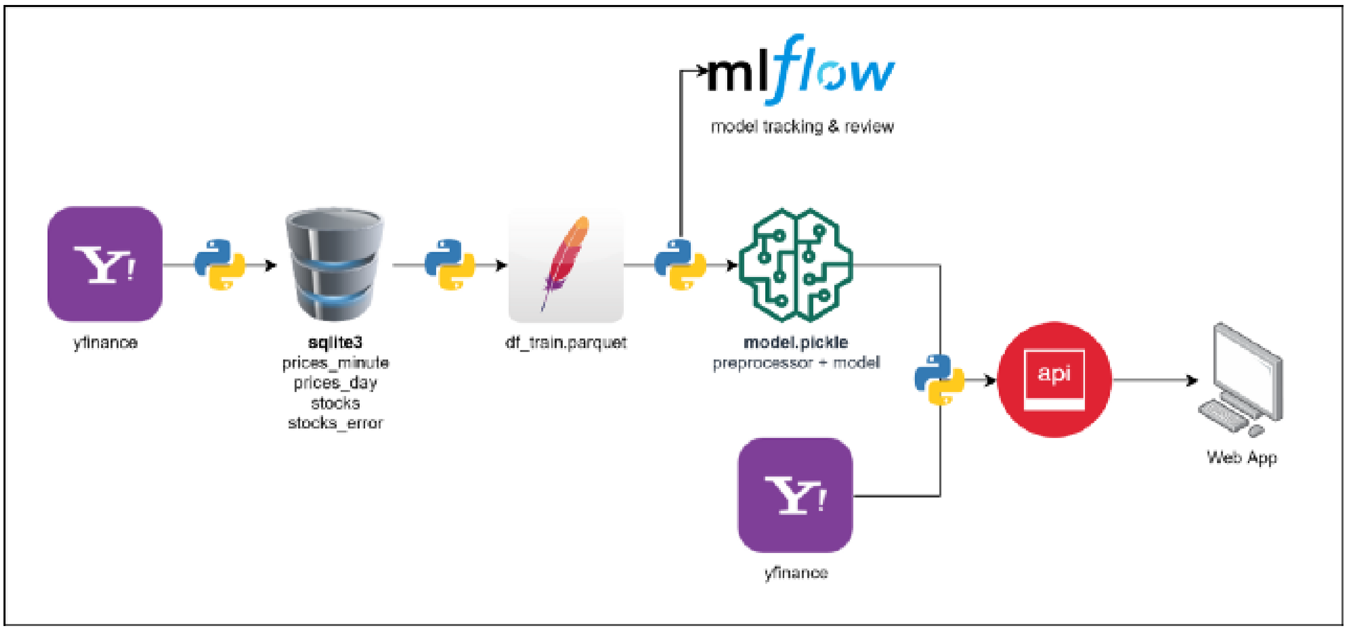
Data Pipeline

Good deployment should start with a well-designed data pipeline, which has to support:

- Storage of raw and processed data
- Model training (and performance tracking)
- Model serving

SQLite3 was used to store the bulky raw price data and other stock-related information. **Parquet** was chosen for intermediate training data (much faster than CSV). Lastly, the trained model was stored using **Pickle** for loading later.

Here is the entire pipeline:



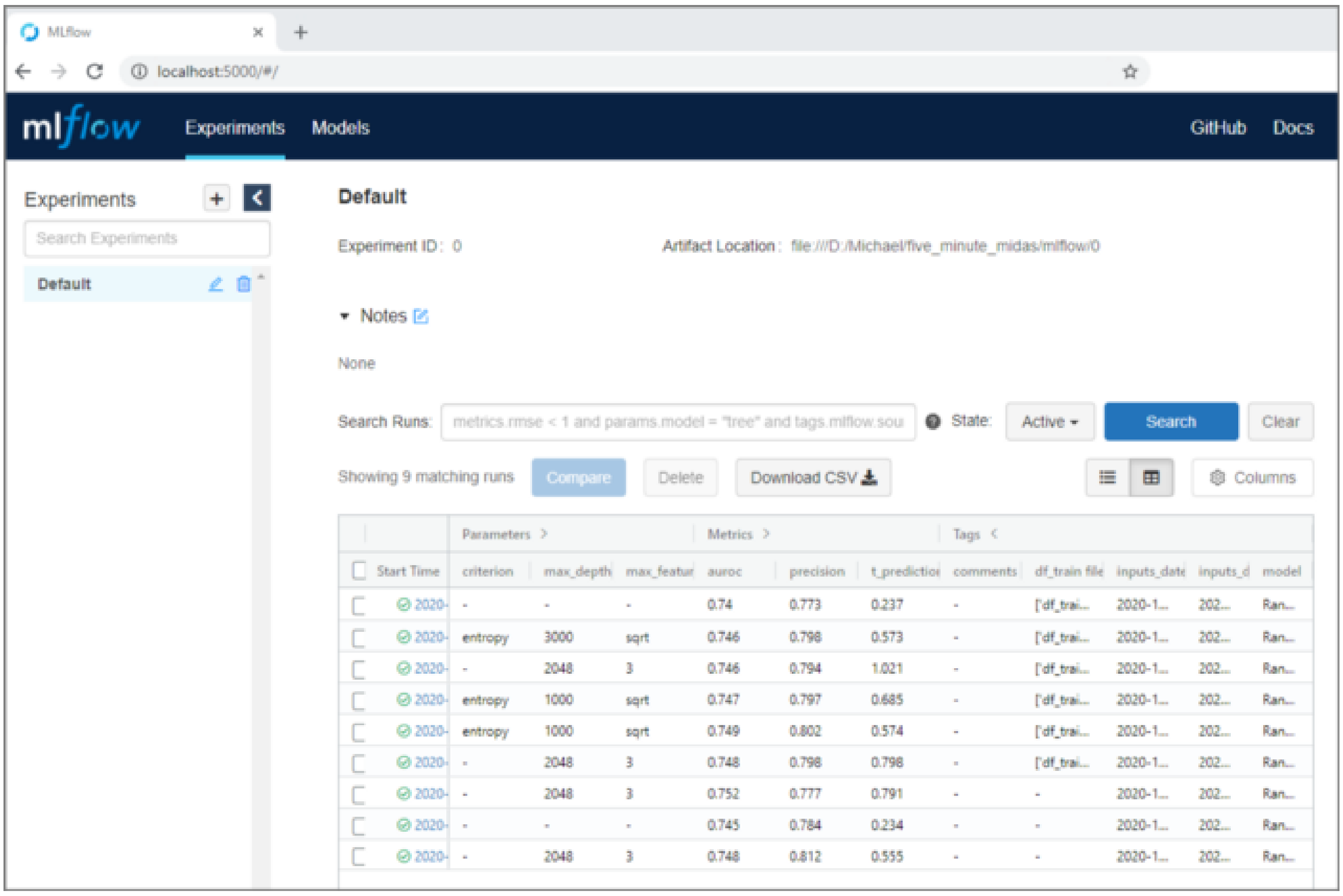
Pipe dreams. — Made using draw.io — Image by author

MLflow

> MLflow is an open source platform to manage the ML lifecycle, including experimentation, reproducibility, deployment, and a central model

registry.
- MLflow.org

I used **MLflow Tracking** to track model performance, hyperparameters, as well as other model metadata. I followed this tutorial by Databricks. MLflow also contains useful deployment frameworks which I may explore in the future.


Default

Experiment ID: 0 Artifact Location: file:///D:/Michael/five_minute_midas/mlflow/0

Notes

None

Search Runs: metrics.rmse < 1 and params.model = "tree" and tags.mlflow.sou State: Active

Showing 9 matching runs

	Parameters			Metrics			Tags				
Start Time	criterion	max_depth	max_featur	auroc	precision	t_predictio	comments	df_train file	inputs_date	inputs_d	model
2020-	-	-	-	0.74	0.773	0.237	-	['df_trai...	2020-1...	202...	Ran...
2020-	entropy	3000	sqrt	0.746	0.798	0.573	-	['df_trai...	2020-1...	202...	Ran...
2020-	-	2048	3	0.746	0.794	1.021	-	['df_trai...	2020-1...	202...	Ran...
2020-	entropy	1000	sqrt	0.747	0.797	0.685	-	['df_trai...	2020-1...	202...	Ran...
2020-	entropy	1000	sqrt	0.749	0.802	0.574	-	['df_trai...	2020-1...	202...	Ran...
2020-	-	2048	3	0.748	0.798	0.798	-	['df_trai...	2020-1...	202...	Ran...
2020-	-	2048	3	0.752	0.777	0.791	-	-	2020-1...	202...	Ran...
2020-	-	-	-	0.745	0.784	0.234	-	-	2020-1...	202...	Ran...
2020-	-	2048	3	0.748	0.812	0.555	-	-	2020-1...	202...	Ran...

MLflow tracking server UI. — Image by author

Flask API

To serve the model, I used **Flask** to implement a simple Web API.

```python
# This is just an idea of how the flask app operates
from flask import Flask

# load model
full_pipeline = pickle.load(open('model.pickle', 'rb'))

# create flask object
app = Flask(__name__)

# create api endpoint
```

```python
@app.route('/proba', methods=['POST'])
def api_get_df_proba():
    global full_pipeline
    df_cooked = get_df_cooked()
    arr_proba = full_pipeline.predict_proba(df_cooked)
    json_df_proba = get_json_df_proba(df_cooked, arr_proba)
    return json_df_proba

# run flask
if __name__ == '__main__':
    app.run(debug=False, host='0.0.0.0')
```

flask1.py hosted with ♡ by GitHub view raw

Serve the API.

With the API running, we can make a POST request to get a dataframe that contains the latest stock predictions.

```python
import json
import requests

# make API call
url = 'http://localhost:5000/proba'
headers = {'content-type': 'application/json', 'Accept-Charset': 'UTF-8'}
r = requests.post(url, data='', headers=headers)

# convert from JSON to pandas DataFrame
data = json.loads(r.text)
df = pd.DataFrame(**data)

# format datetime columns correctly
for col in ['datetime', 'datetime_update']:
    df[col] = pd.to_datetime(df[col]).dt.round('min')
```

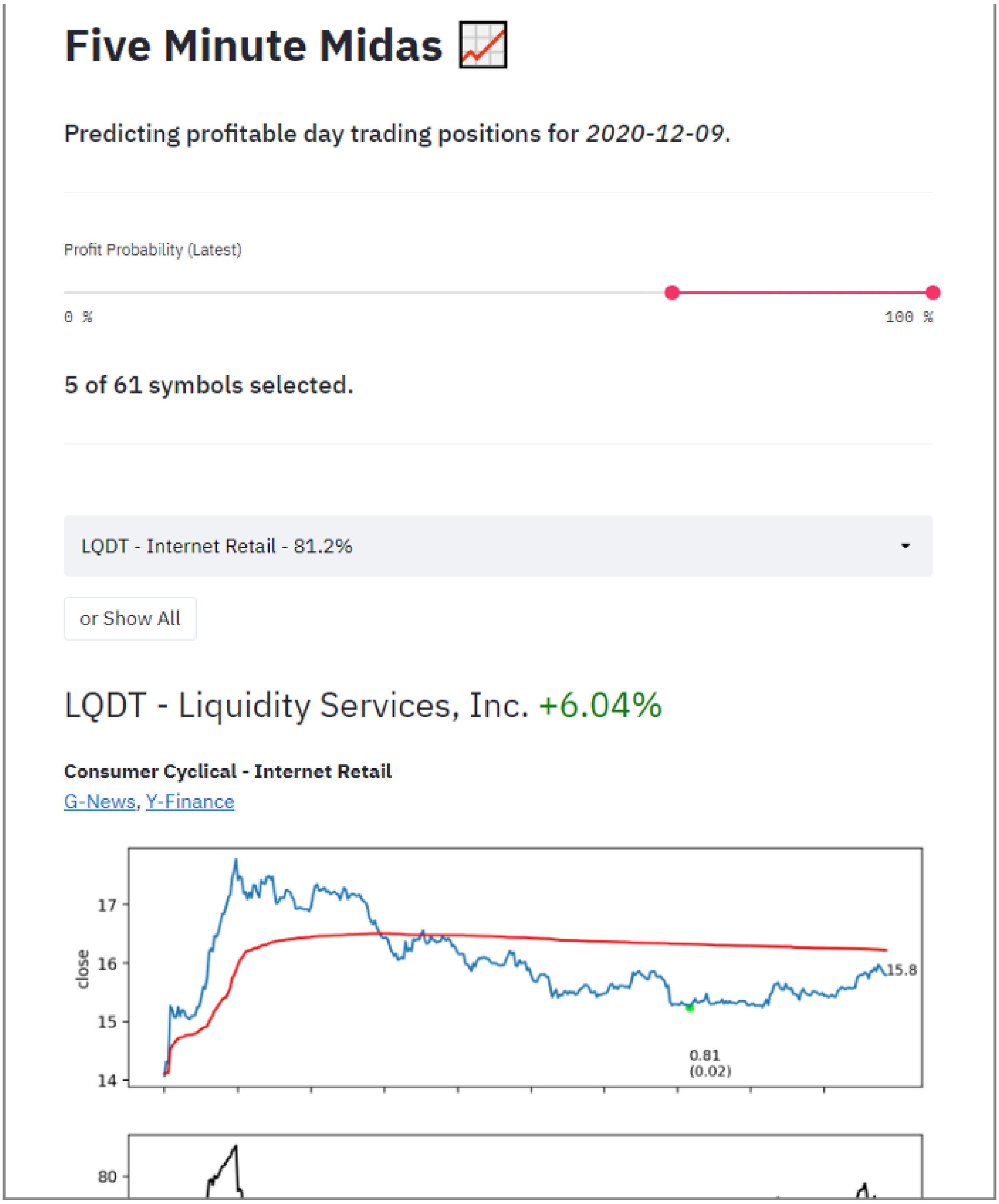
flask2.py hosted with ♡ by GitHub view raw

Call the API.

User Interface

To allow users to interact with the prediction results easily, I designed a web app using the powerful **Streamlit** library. I highly recommend it for beginners as no HTML/CSS knowledge is required, and coding is minimal.

Try the demo! (May take a while to load, be patient!)


Five Minute Midas 📈

Predicting profitable day trading positions for *2020-12-09*.

Profit Probability (Latest)

0 % 100 %

5 of 61 symbols selected.

LQDT - Internet Retail - 81.2%

or Show All

LQDT - Liquidity Services, Inc. +6.04%

Consumer Cyclical - Internet Retail
G-News, Y-Finance

UI! — Image by author

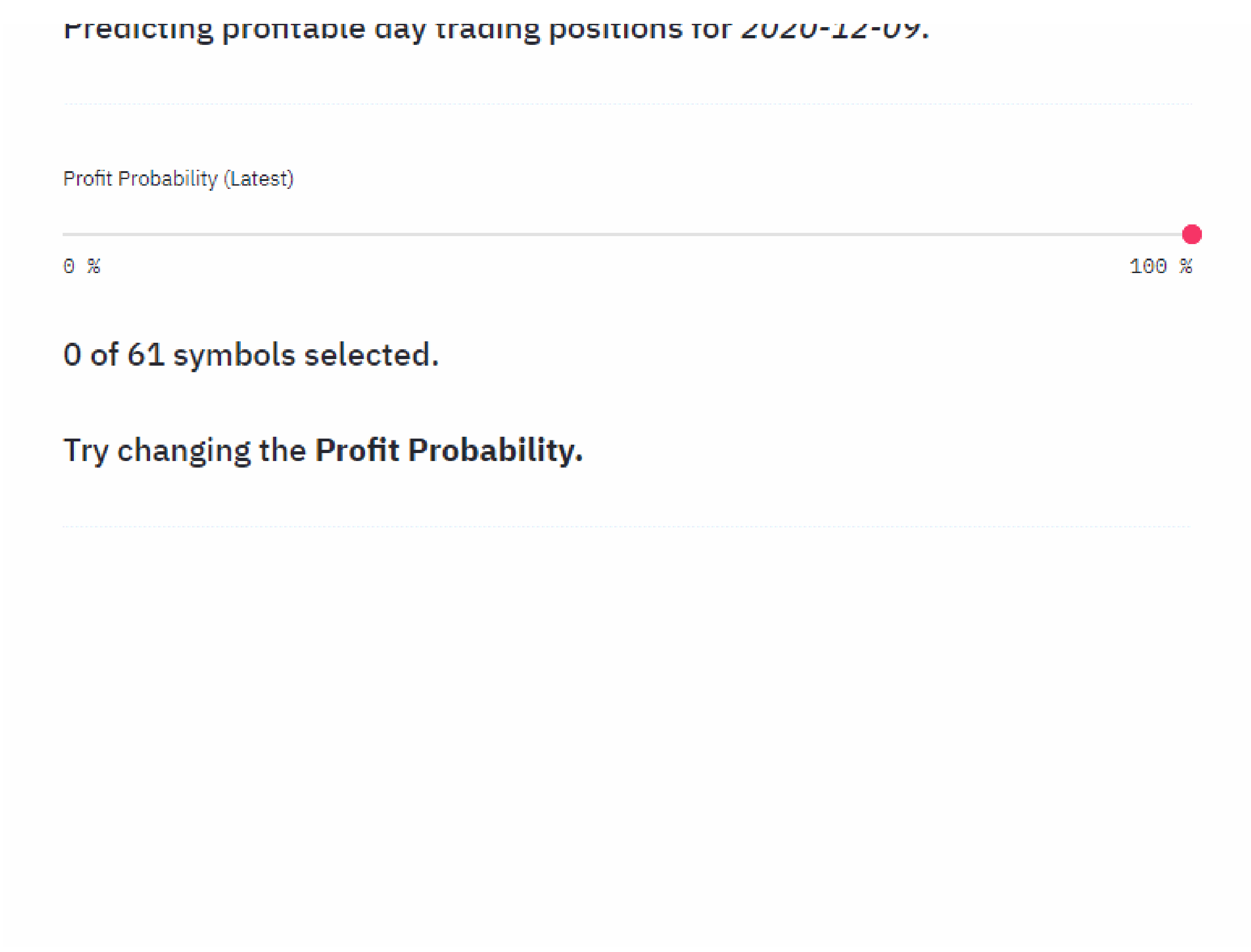
How I Used the App to Trade

1. Start the Flask server

The Flask server periodically grabs the latest minute-level price data and feeds it into the ML model. When I was trading, I only looked at stocks from the *Technology,* *Utilities,* and *Communications Services* sectors. The model then outputs a dataframe with the profitability predictions of the entry positions.

2. Filter stock symbols

The web app will make an API call to retrieve the latest predictions and allow me to interact with the data. I filter to symbols above a certain profit probability (more than 70%). I also select only predictions within a certain period (usually 10 minutes). This is because prices tend to change very quickly after an RSI divergence, reducing the usefulness of old predictions.


Predicting profitable day trading positions for 2020-12-09.

Profit Probability (Latest)

0 % 100 %

0 of 61 symbols selected.

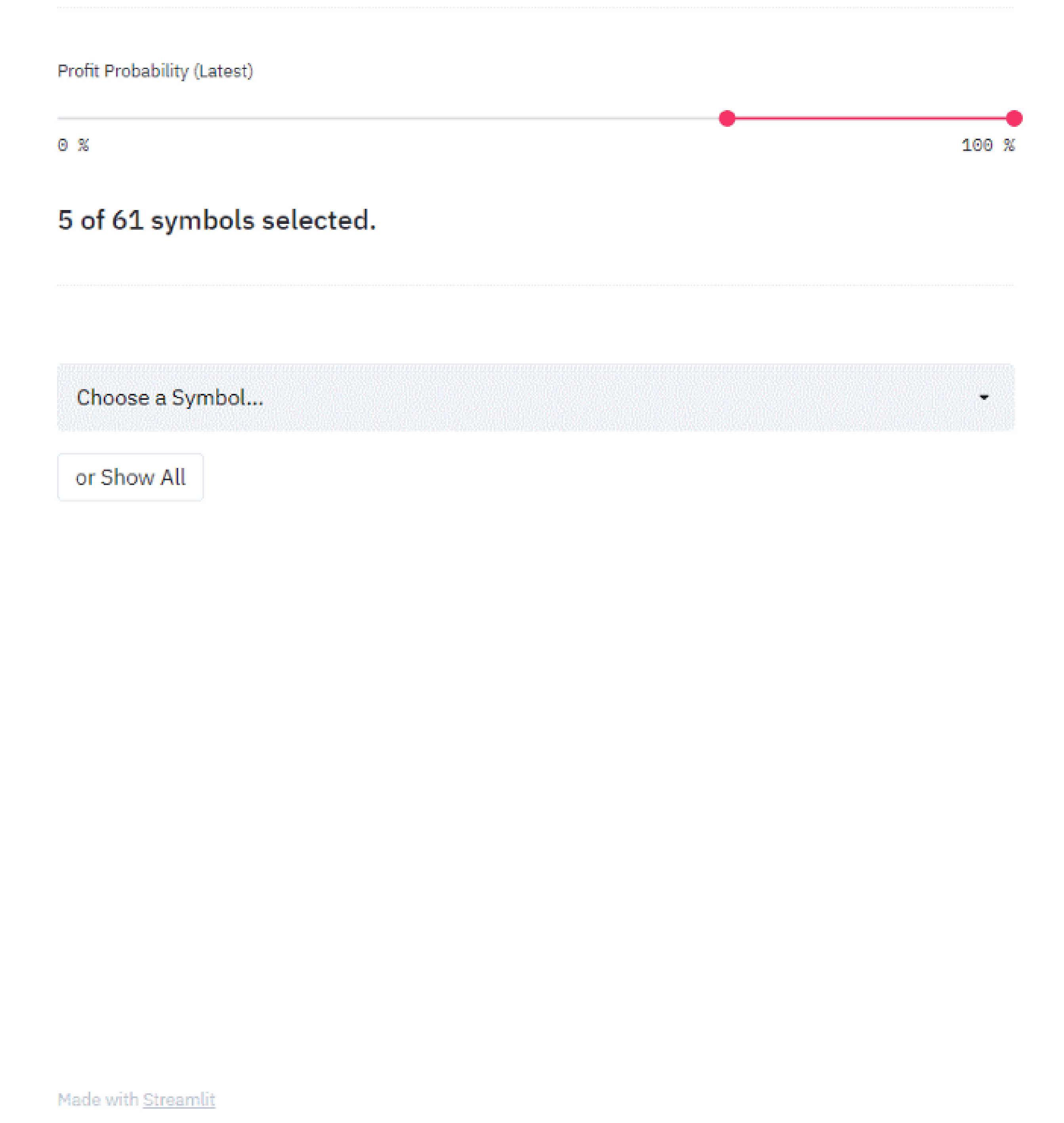
Try changing the **Profit Probability.**

3. Chart analysis

With the initial symbols chosen, I do a deep dive into each stock, usually starting with the highest profit probability. I may also choose well-known symbols first (e.g. Tesla, Zoom, Spotify) or others that are trending.

The web app makes an API call to the Flask server, grabbing the latest predictions for the selected stock and plots it for analysis.

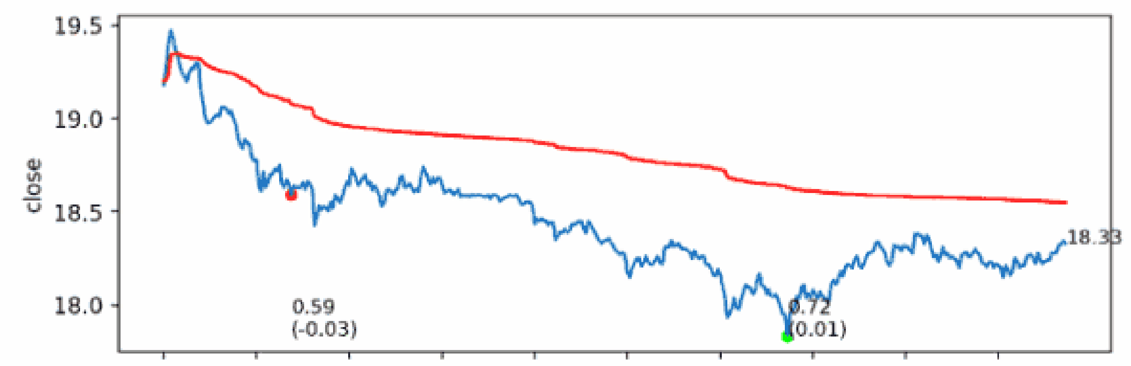
Predicting profitable day trading positions for 2020-12-09.



5. Other analysis

On top of the web app chart analysis, I will do additional research using Google News, scanning through the headlines for any positive/negative sentiment. A convenient link to Google News is provided.

NKLA - Nikola Corporation
Consumer Cyclical - Auto Manufacturers
G-News, Y-Finance



6. Buy the stock

Once I have completed my analysis and feel comfortable with the stock, I use a brokerage application (desktop/mobile) to execute the purchase. Instead of pouring all my capital into one symbol, I usually spread out my purchases, around $1000 USD per symbol. If all goes well, I exit my positions on the same day once they hit the 1% profit target.

Lessons learnt

Photo by Scott Graham on Unsplash

Real-world data is dirty

Many data anomalies were found throughout the project, whether it be errors during extraction, stock market holidays, or stocks with sparse/missing price data. These anomalies had to be accounted for, to provide clean training data for the model.

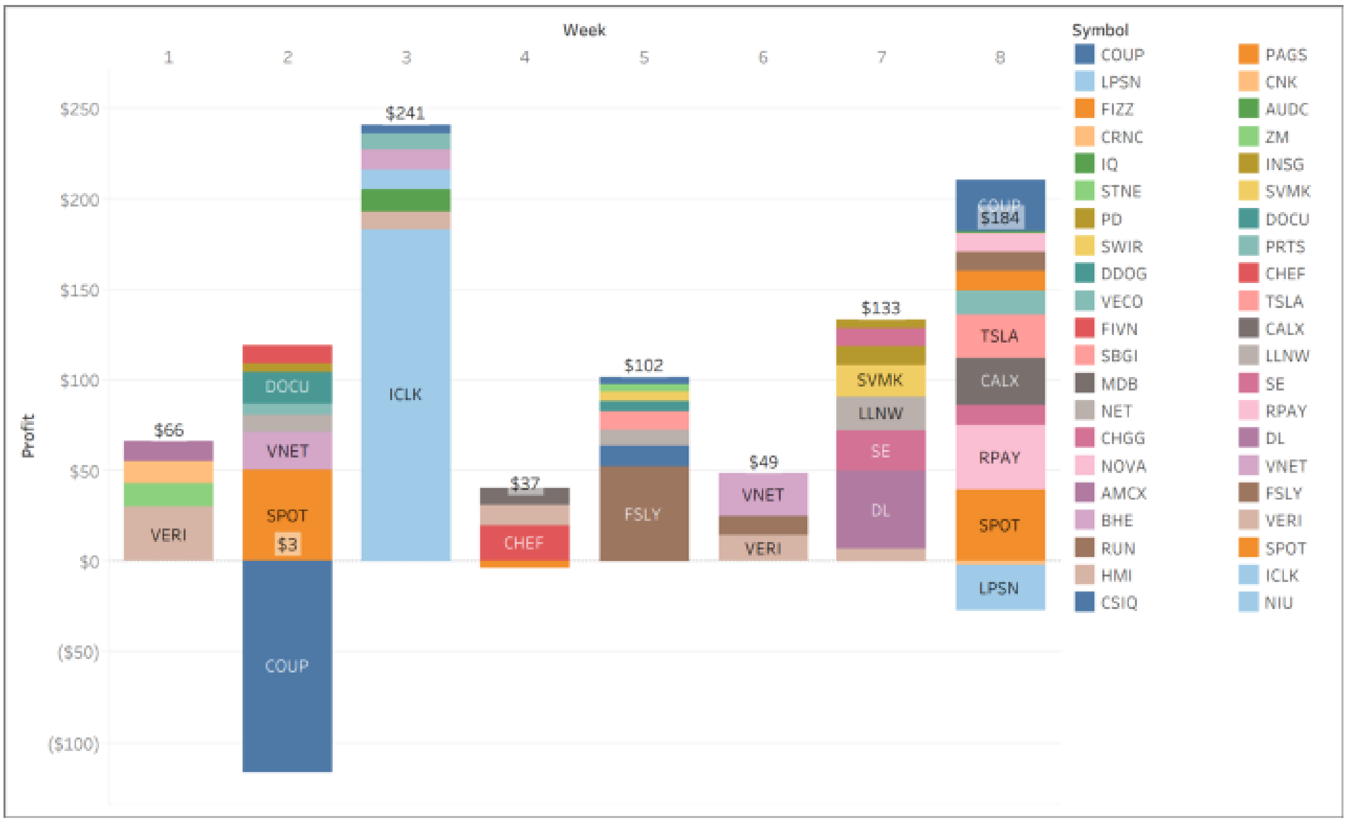
Real Trading vs Paper Trading

Trading real money is way harder than doing paper trades (simulated trading). It is a rollercoaster of emotions; the anxiety before a purchase, heartache over losses, and delight when you lock in profits. The good news is that with experience, you will get used to the literal ups and downs of the process.

Model Limitations

The determination of RSI divergence is highly dependent on exactly which points are labeled as max/min. This may lead to changes in the location of RSI divergence points as more price data comes in. Because model training and validation are performed on data with "finalized" RSI divergences, the predictions made during live trading may be less accurate until the RSI divergences solidify. This issue can be mitigated by giving 1–2 minutes for the signal to stabilize before using the prediction.

Results



Profit/Loss Breakdown. Cross my heart...— Chart made in Tableau — Image by author

All in all, I made around $800 USD in net profits.

This consisted of roughly 60 day trades, averaging at $14 of profit per trade. The chart above summarizes the profits and losses made per symbol each week. A vast majority of my entries hit the 1% profit target and were sold within the same day (sometimes within minutes). There were, however, a few losses as well as seen in the chart.

Mind you, I did not follow the recommendations of the ML model blindly and often spent a good amount of time doing additional research. This included my own price chart analysis and looking at other information sources (e.g. Google News, Yahoo Finance). The model functioned as a stock screener, highlighting stocks I should focus my manual analysis on.

These results are not a measure of the predictive power of the model but serve as a good proof-of-concept regarding the effectiveness of the model's deployment and usage.

> Step 1: Collect stock market data
> Step 2: Train ML model
> Step 3: ???
> Step 4: Profit.
> - Anonymous

Conclusion

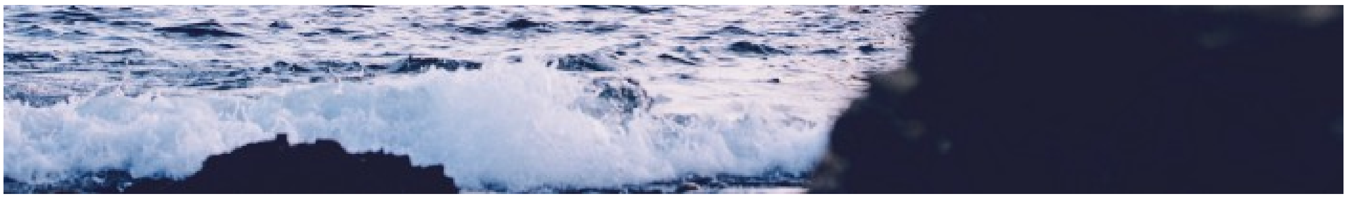


Photo by Austin Neill on Unsplash

This project really stretched me beyond my existing capabilities:

New things I picked up:

- End-to-end machine learning (Data collection, training, deployment)
- API, web app cloud deployment (Flask, Streamlit, Heroku)
- Model tracking (MLflow)

Skills I honed:

- ML Classifiers, hyperparameter tuning (scikit-learn)
- ETL, Database query/design (pandas, SQLite3)
- Software version control (GitHub)

Future Work

Photo by Andreas Selter on Unsplash

1. Cloud Hosting

In this project, I hosted the API and web app locally for personal use. Moving forward, I could containerize the entire application with Docker and host it on a cloud provider (Heroku, AWS, Azure, GCP).

2. Data and Model Enhancements

Advanced feature engineering can be implemented (PCA, advanced outlier detection). Creating new features based on other trading indicators can also be explored. Other ML models such as Support Vector Machines and Neural Networks can be experimented with.

3. Sentiment Analysis

The training data used in this project is entirely based on price action. Sentiment analysis could bring additional insight to the model and improve prediction accuracy.

Thank you for reading to the end!

👨‍💻 See the code: GitHub
💻 Try the web app: Demo
✍️ Get in touch: LinkedIn

> All courses of action are risky, so prudence is not in avoiding danger (it's impossible), but calculating risk and acting decisively. Make mistakes of ambition and not mistakes of sloth. Develop the strength to do bold things, not the strength to suffer.
> - Niccolo Machiavelli

Machine Learning Data Science Python Random Forest Trading