Market Sentiment Classifier

Group: Bimal Rajbhandari, Kathleen Nie, Nick Pandolfi

ADSP 32021 ON01 Machine Learning Operations

Term: Fall 2025

Agenda

Predicting Market Sentiment

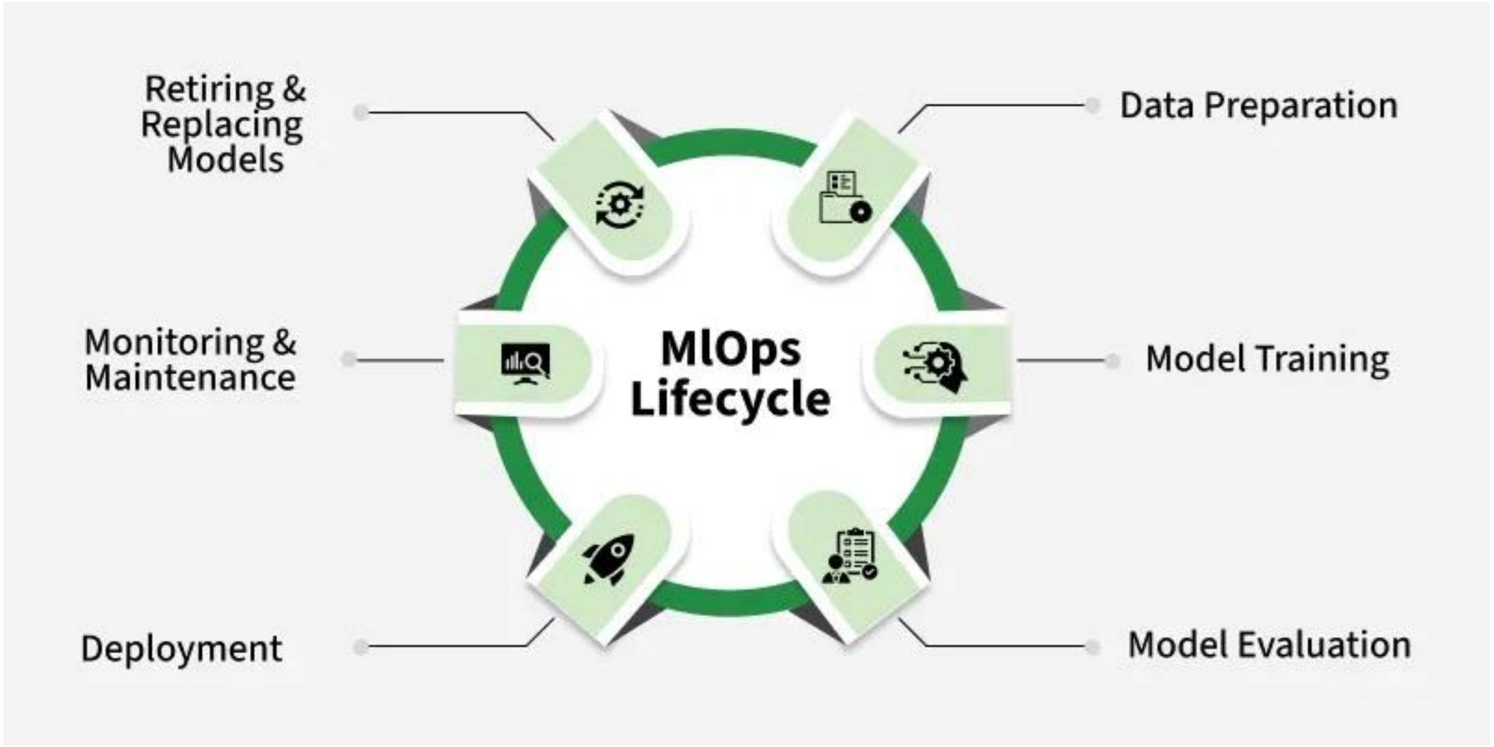
Why It Matters

- **Market sentiment (fear/neutral/greed)** drives short-term trading and risk exposure.
- **Early prediction** enables better portfolio adjustments and risk management.

How the Model Works

- **Inputs:**
  - SPY/VIX: Price, volatility, moving averages
  - News headlines: Sentiment scores (VADER, TextBlob)
- **Output:** Predicts **tomorrow's sentiment** (fear/neutral/greed)

MLOPS Component and Lifecycle



Data Management & Preparation

Data Preparation

Data Source

- Fetched historical SPY (S&P 500 proxy) and VIX (Volatility Index) data from **Yahoo Finance API (yfinance).**
- Financial news headlines from Kaggle
- *Time Horizon:* 2008-01-02 to 2024-03-04.

Feature Engineering & Target Creation

We transformed raw inputs into 11 predictive features and the target variable

Time-Series Lagging: All predictor variables were explicitly **lagged by one day** to ensure the model uses only t-1 information for predicting.

Target Labeling: The dependent variable, Sentiment_Label, was engineered by applying rules-based thresholds to the VIX Close price (VIX > 30 = Fear, VIX < 20 = Greed, else Neutral).

Data Preparation

Processed and aggregated news headline sentiment

- Applied VADER compound scores
- Applied TextBlob polarity and subjectivity

Engineered 11 Lagged Features (using only t-1 data)

– VIX-derived features:

- Lag_1_VIX_Close
- Lag_1_VIX_Daily_Change
- Lag_1_VIX_7D_MA
- Lag_1_VIX_30D_MA

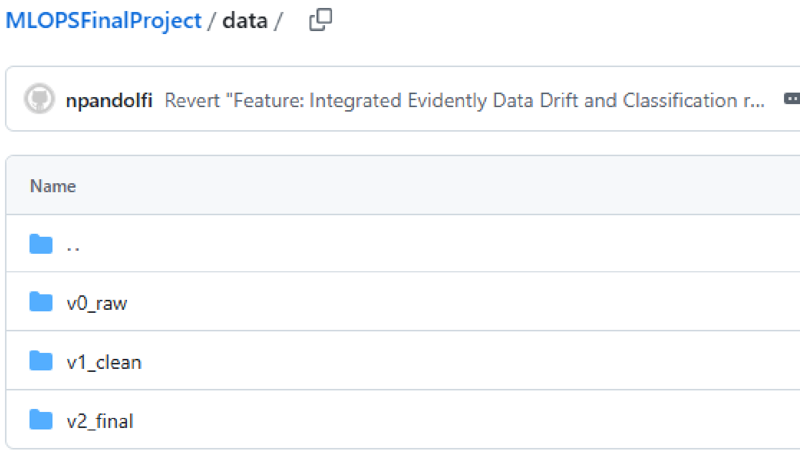
– SPY-derived features:

- Lag_1_SPY_Close
- Lag_1_SPY_Daily_Return
- Lag_1_SPY_7D_Return
- Lag_1_SPY_30D_Return

Data Management

Data Versioning & Structure

- Raw market and news un-processed data stored as Version 0
- Merged Feature engineered SPY, VIX and news dataset as Version 1
- Feature-engineered news,SPY, VIX modeling-ready dataset as Version 2
- **Git Integration**: All versions tracked using GitHub for reproducibility and data management



MLOPSFinalProject / data /

npandolfi Revert "Feature: Integrated Evidently Data Drift and Classification r...

Name
..
v0_raw
v1_clean
v2_final

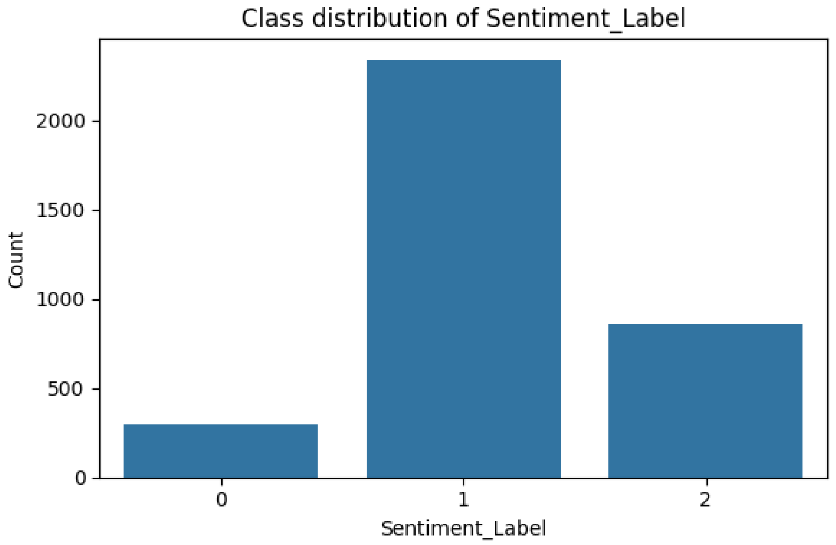
Exploratory Data Analysis

Exploratory Data Analysis

We use a mix of Quantitative & Qualitative Data to build our model

- **Yahoo Finance:** API that allows deep, long-term access to Stock Market Data for quantitative purposes. We pull VIX and S&P500 daily returns
- **Financial News:** S&P 500 with Financial News Headlines (2008–2024) dataset from Kaggle to extract qualitative data

These datasets are combined in the Data Engineering Phase to build a final Dataset

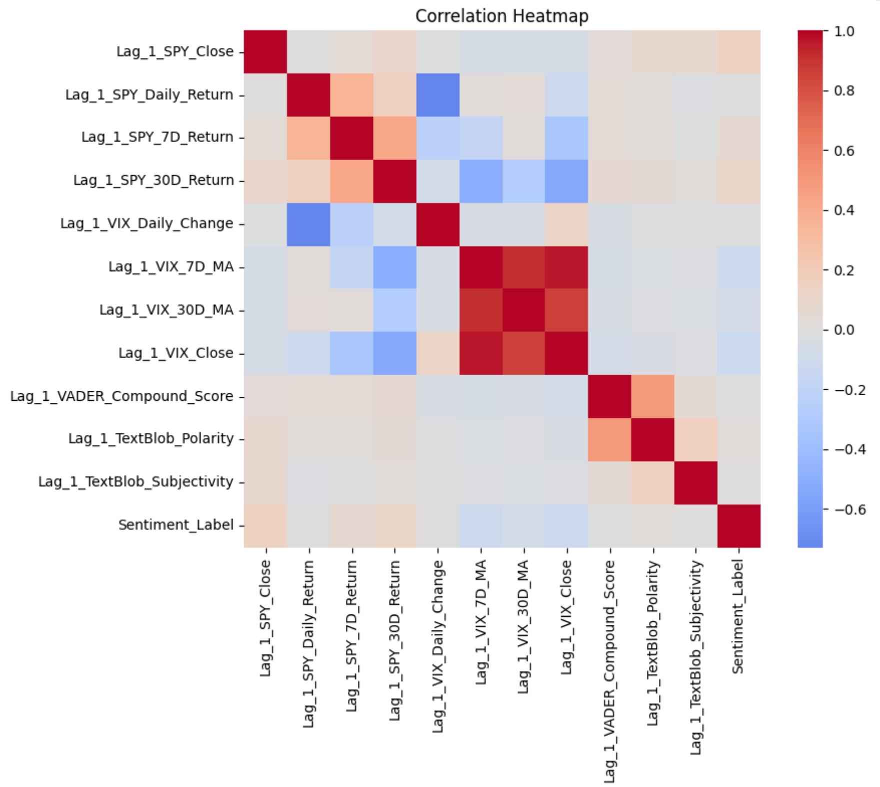


Exploratory Data Analysis

Correlation Structure of Market & Sentiment Features

- VIX group is strongly intercorrelated
- SPY-related features form their own block
- Text sentiment is weakly correlated but non-zero

Sentiment_Label shows meaningful relationships to SPY/VIX changes but weaker ties to text scores.



Exploratory Data Analysis

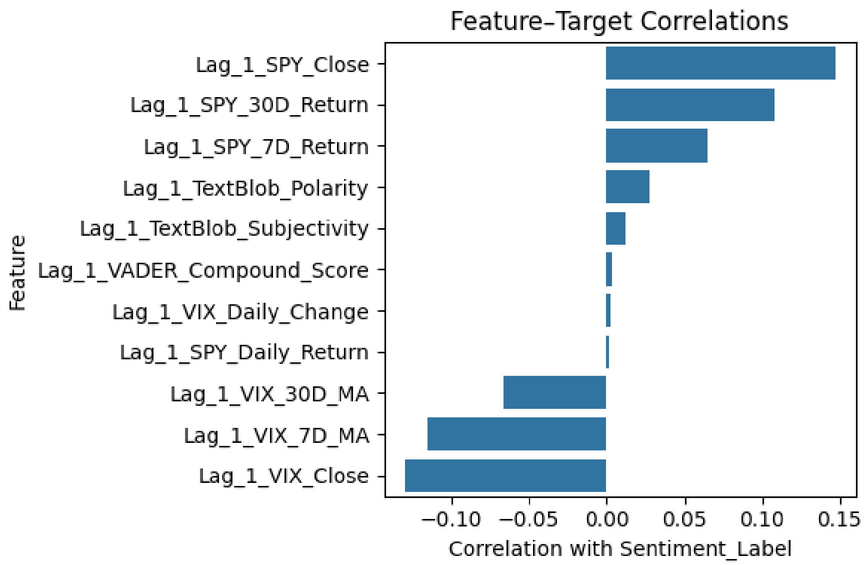
Key Predictive Features for Market Sentiment

Top positive-leverage features:

- Lag_1_SPY_Close
- Lag_1_SPY_7D_Return
- Lag_1_SPY_30D_Return

Top negative-leverage features:

- Lag_1_VIX_Close
- Lag_1_VIX_7D_MA
- Lag_1_VIX_30D_MA



Pipeline & Modeling

Scikit-Learn Pipeline

Handling Class Imbalance

Pipeline 1 – FLAML with Class Weighting

- Used FLAML’s AutoML with class_weight='balanced' (via custom function).
- Class weighting forces the model to pay more attention to minority classes.
- Models: Random Forest, XGBoost, LightGBM (tree-based, support class weights).
- Metric: F1-score (better for imbalanced data than accuracy).

Pipeline 2 – FLAML with SMOTE

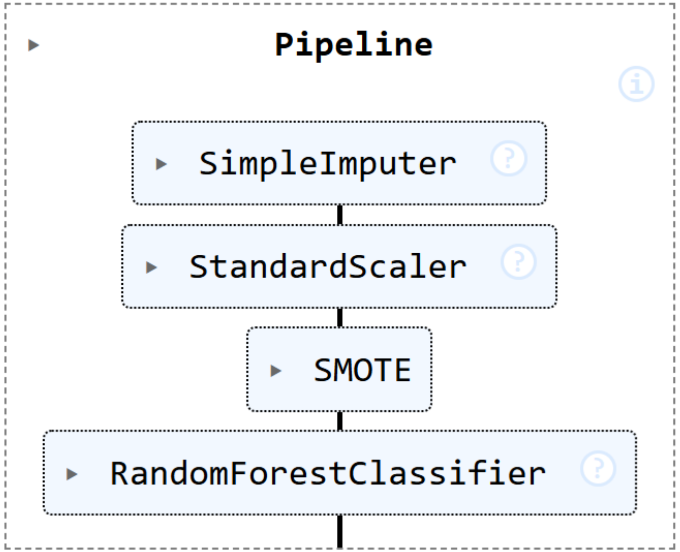
- Used ImbPipeline to integrate SMOTE into the workflow.
- Steps: Standard scaling → SMOTE → FLAML AutoML.
- SMOTE creates synthetic samples, balancing class distribution.

Scikit-Learn Pipeline

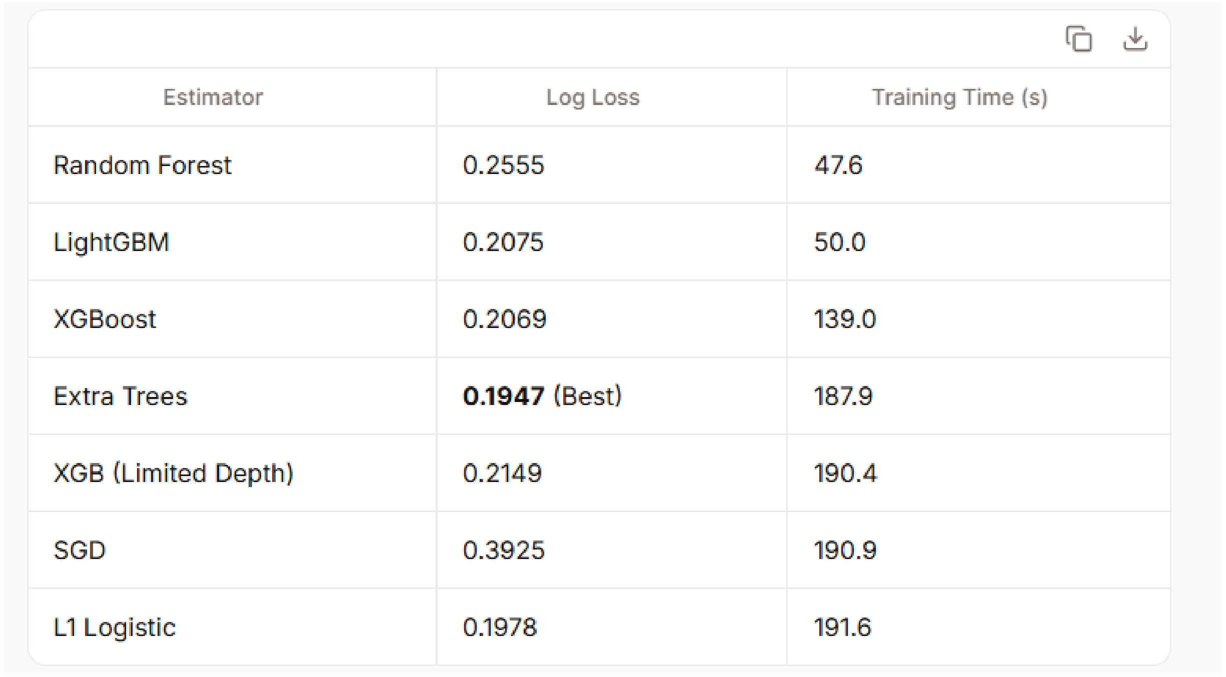
Handling Class Imbalance

Pipeline 3 - FLAML Fear-Weighted Random Forest with SMOTE

- Built on FLAML AutoML's best RandomForest hyperparameters
- Incorporated Fear-focused class weights
- Combination of pipeline 1 and 2



Scikit-Learn Pipeline

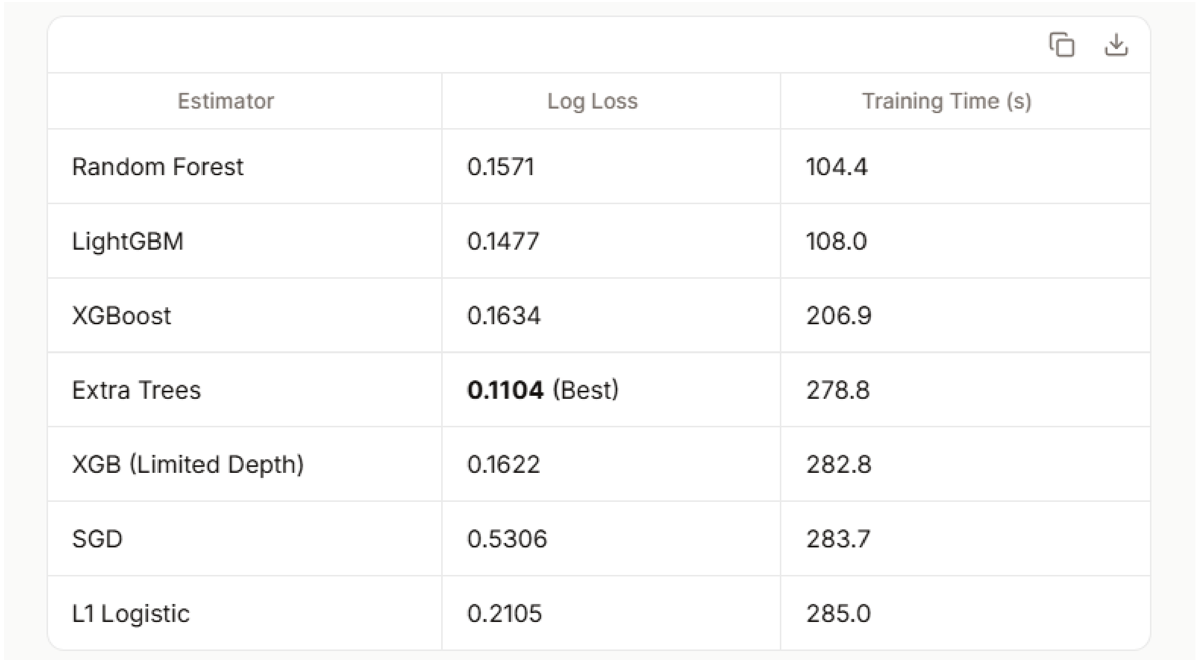


Estimator	Log Loss	Training Time (s)
Random Forest	0.2555	47.6
LightGBM	0.2075	50.0
XGBoost	0.2069	139.0
Extra Trees	**0.1947** (Best)	187.9
XGB (Limited Depth)	0.2149	190.4
SGD	0.3925	190.9
L1 Logistic	0.1978	191.6

Pipeline 1 – FLAML with Class Weighting: Training Summary

- **Best Model: Extra Trees Classifier** (log loss = 0.1947).
- **Final Model Parameters:**
  - `criterion='entropy'`
  - `max_features=1.0`
  - `max_leaf_nodes=18344`
  - `n_estimators=2047`
  - `n_jobs=-1`

Scikit-Learn Pipeline



Estimator	Log Loss	Training Time (s)
Random Forest	0.1571	104.4
LightGBM	0.1477	108.0
XGBoost	0.1634	206.9
Extra Trees	**0.1104** (Best)	278.8
XGB (Limited Depth)	0.1622	282.8
SGD	0.5306	283.7
L1 Logistic	0.2105	285.0

Pipeline 2 – FLAML with SMOTE

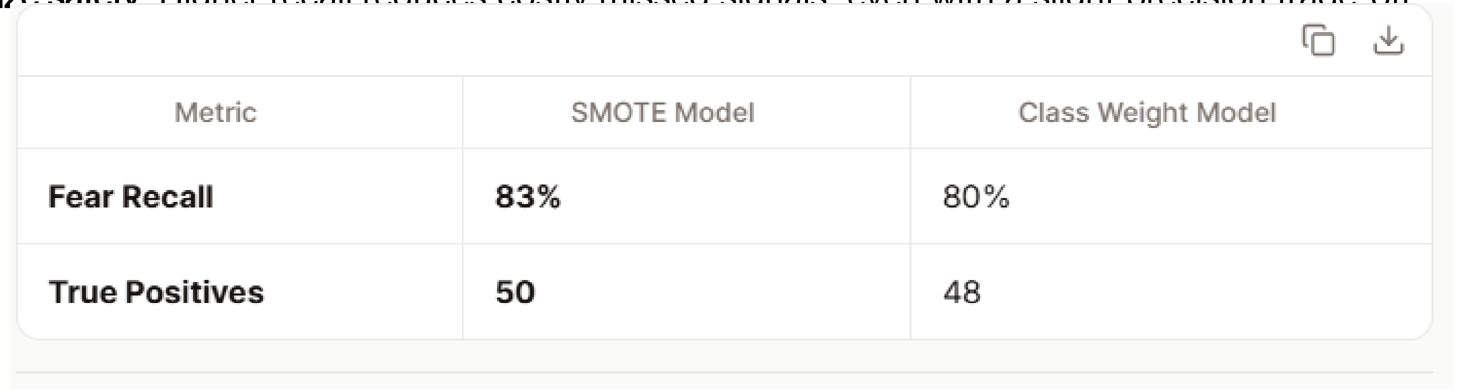
- **Best Model: Extra Trees Classifier** (log loss = 0.1104).
- **Final Model Parameters:**
  - `criterion='entropy'`
  - `max_features=1.0`
  - `max_leaf_nodes=18344`
  - `n_estimators=2047`
  - `n_jobs=-1`

Scikit-Learn Pipeline

Maximizing Fear Detection for Portfolio Safety

SMOTE Model – Better for Risk Management

- **Fear (Class 0)** = Critical, rare signal to **"go defensive"** and avoid downside risk.
- **False Negatives (missing Fear)** = Staying aggressively invested during market downturns → **high risk.**
- **SMOTE captures 3% more Fear instances** (83% vs. 80% recall).
- **Prioritize safety:** Higher recall reduces costly missed signals, even with a slight precision trade-off



Metric	SMOTE Model	Class Weight Model
Fear Recall	**83%**	80%
True Positives	**50**	48

Deployment

Dockerization for Reproducible Model Deployment

Objective: To package the trained model and its dependencies into a single, portable, and reproducible unit for easy deployment.

Dockerization Process:

1. **Dependencies:** A requirements.txt file listed all necessary Python libraries (FLAML, Scikit-learn, FastAPI, uvicorn, etc.).
2. **Docker Structure:** The Dockerfile contained instructions to:
   - Set up a base Python environment.
   - Install the required dependencies.
   - Copy the saved model pipeline file.
   - Copy the FastAPI application code (api.py).

3. **FastAPI Endpoint** (lightweight, high-performance).

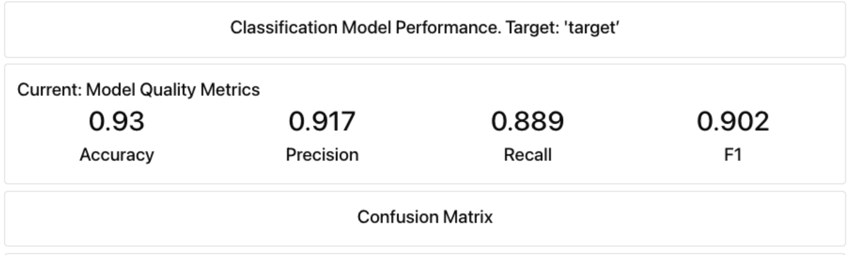
   - Accepts market and sentiment features as input.
   - Returns predicted sentiment class (0, 1, or 2).

Model Monitoring With Evidently

Model Monitoring with Evidently

Classification Evaluation:

- **Evidently Performance Tracking** shows the accuracy of our classification model at 93% accuracy
- **Multiclass Classification:** Our model performs extremely well at predicting **Greed** and the most poorly at predicting **Neutral**
- Evidently also provides a data summary by evaluating distributions, counts, NAs, etc

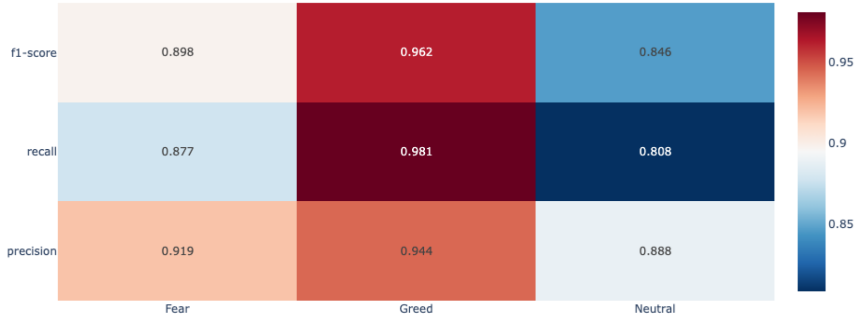


Classification Model Performance. Target: 'target'

Current: Model Quality Metrics

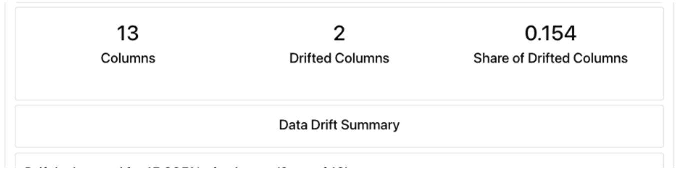
0.93	0.917	0.889	0.902
Accuracy	Precision	Recall	F1

Confusion Matrix



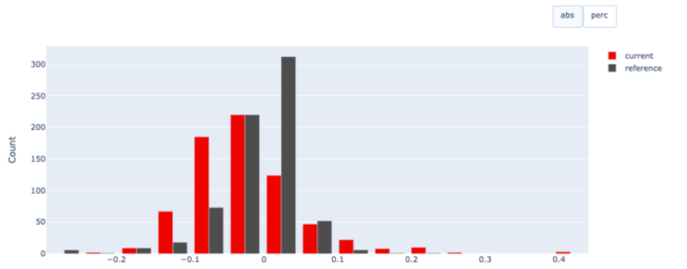
	Fear	Greed	Neutral
f1-score	0.898	0.962	0.846
recall	0.877	0.981	0.808
precision	0.919	0.944	0.888

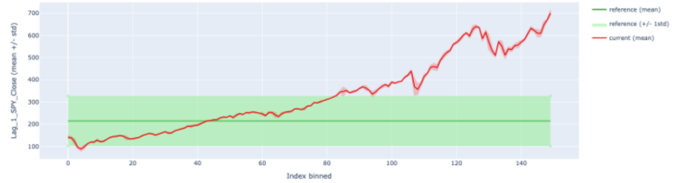
Data Drift



13	2	0.154
Columns	Drifted Columns	Share of Drifted Columns

Data Drift Summary





Real-World Data Drift Scenarios:

- **Regime Change:** We simulate a regime change by multiplying the daily S&P500 by 120% (organic stock market rise of 20% in 2 years). This highlights the continuous need to re-train our model as time goes on
- **Feature Swap:** We also simulate an error of 2 columns being mixed up in the data ingestion process (Lag_1_SPY_30D_Return and Lag_1_VIX_Daily_Change)

Our Model Monitoring is able to detect both of these scenarios via the K-S p_value (test for difference in distributions), highlighting 2 drifted columns and 15% share of drifted columns

Summary

- **Recap**
  Models are often developed and proven effective in a sterile research environment (the laptop), but they fail to be deployed, scaled, or reliably maintained in the real-world production environment without MLOps

- **Pipeline & Engineering**
  Our data versioning provides reproducible snapshots of data and features used for training, and pipelines automate the end-to-end workflow using our versioned assets.

- **Model Overview**
  AutoML can help identify the best performing model for our given dataset to redistribute human resources to other areas

- **Model Monitoring**
  Evidently can be great for automated performance tracking and data drift monitoring to ensure our model is consistently being refreshed as data shifts for various reasons

Thank you!

References

Check it out on GitHub!

https://github.com/rajbhanb/MLOPSFinalProject

https://www.kaggle.com/datasets/notlucasp/financial-news-headlines