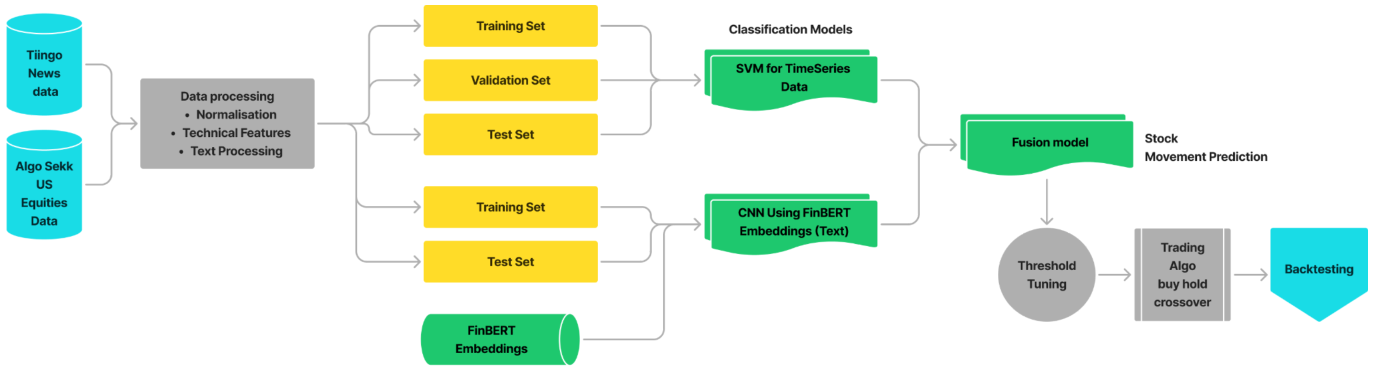
SQA ALPHATHON 2024

Multimodal Model Approach to Generate Alpha Using FinBERT

Question 2: “Use LLM to outperform S&P 500”
Team Name: RamQuant
Team Members: Karandeep Sonewane
Email: kcs@fordham.edu
Strategy for Question 2: We are trying to leverage a multimodal model using FinBERT embeddings to predict extreme price movements in Tech Stocks.

This is a simple demonstration of a strategy without much finetuning of the models. Additionally, the data for CNN Tiingo data have been cut to 6 months (2019-06-01 - 2019-12-31), which can be further improved to train the model effectively on minority class.



1. MultiModal Dataset

As Quantconnect provides a multitude of datasets including price data and news data, our objective is to utilize a multimodal dataset to predict the extreme price movements is S&P 500 constituents. For simplicity we are only interested in Tesla (“TSLA”) as it has shown significant price movements in the past.

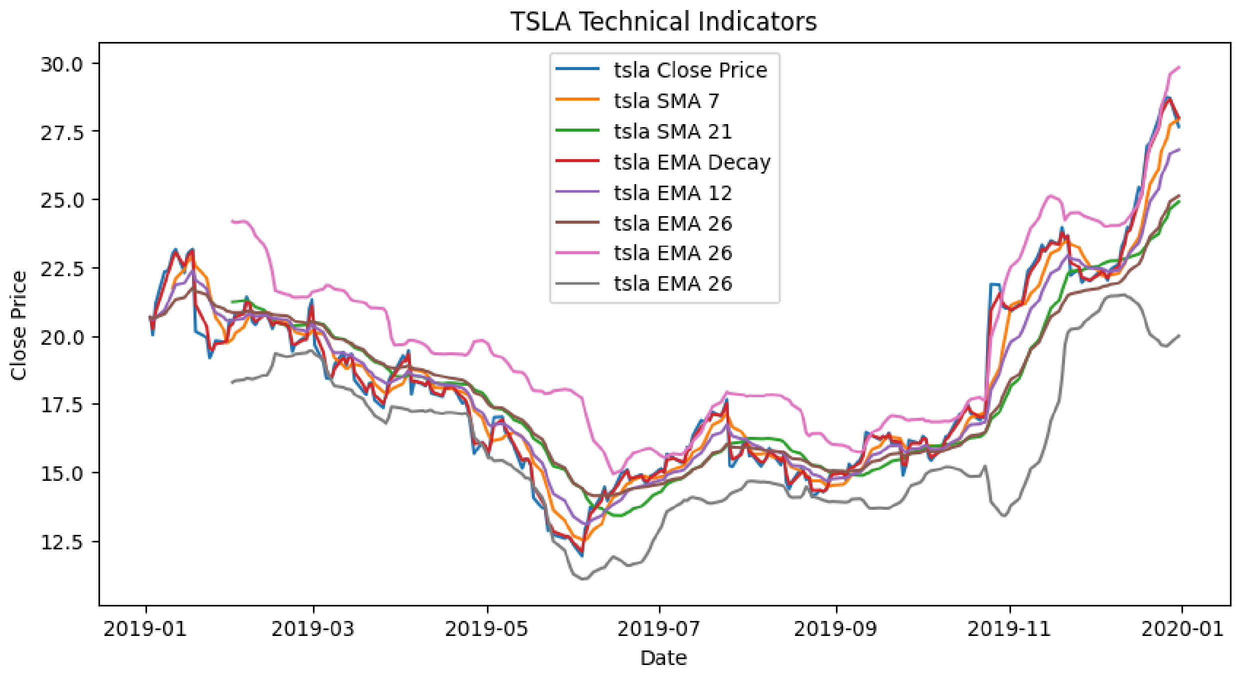
Data from Elon Musk's Twitter activity in 2021, comprising 563 tweets, is analyzed to understand its impact on Tesla's stock returns. The research uses the Capital Asset Pricing Model (CAPM) to calculate Tesla's abnormal returns, comparing Tesla's daily stock returns with expected market returns. The findings show that the correlation between the frequency of all tweets and Tesla's abnormal returns is weak at -0.02, while the absolute magnitude of abnormal returns correlates positively at 0.11. Interestingly, Tesla-related tweets are found to have a small positive correlation with abnormal returns (0.062). Sentiment analysis reveals that negative sentiment in social media and fewer CEO tweets coincide with negative abnormal returns. For example, in May 2021, during periods of higher pessimism, Tesla experienced significant negative abnormal returns *(Jauron Dam, 2023).*

For this purpose we are using Tiingo News data and AlgoSeek US Equites dataset focusing only on TSLA. We are collecting the time series data and News data from 2019-01-01 to 2019-12-31 to train our models.

1.1 Technical Data

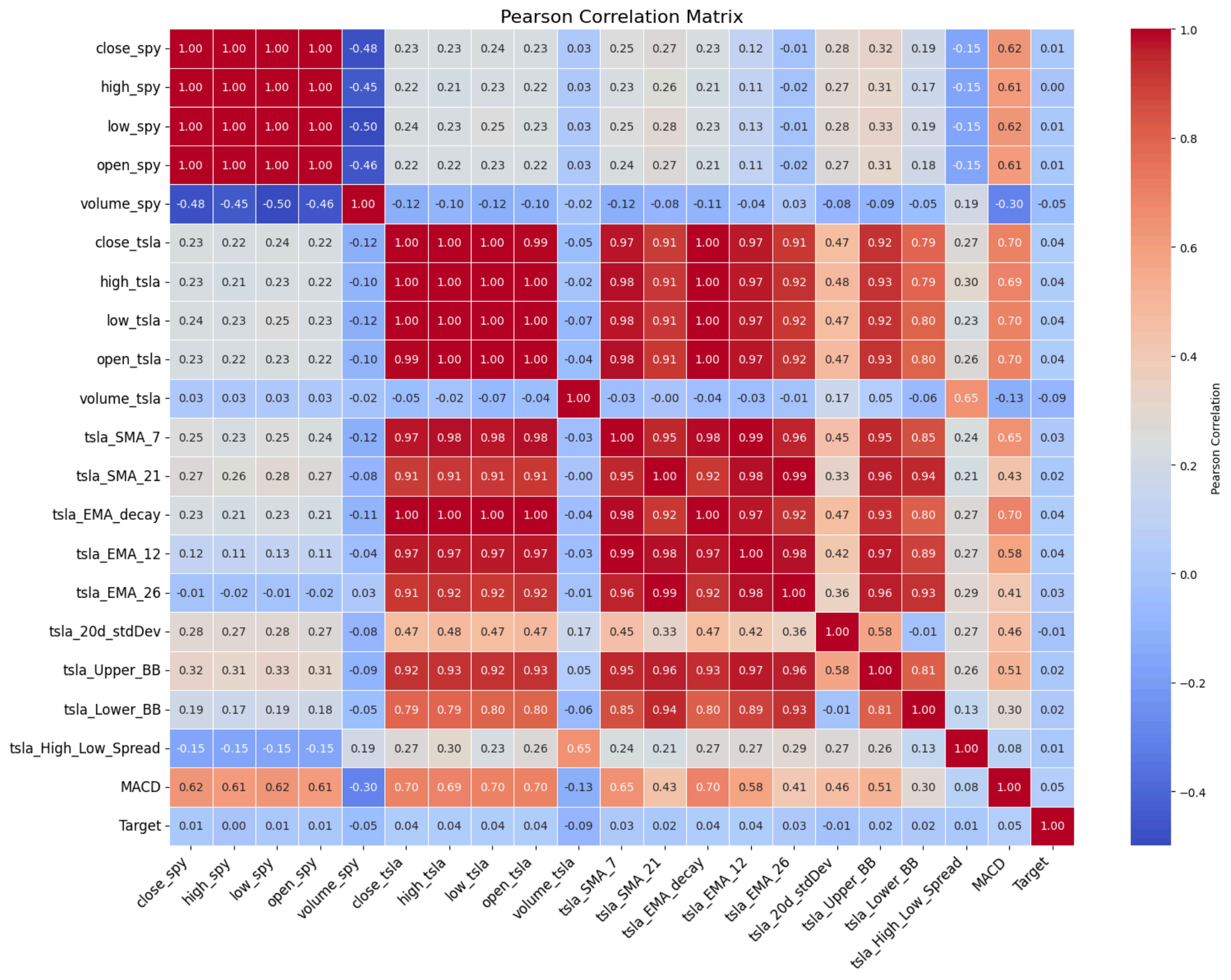
We will refer to price related data as Technical Data. It consists of OHLCV data of TSLA, SPY close and volume data, along with technical features. The data range from 2019-01-01 to 2019-12-31 on Daily resolution.

1.2 Technical Indicators



1. **close_tsla**: The closing price of Tesla, which reflects the market's final valuation of the stock for the day, crucial for price trend analysis.
2. **high_tsla**: The highest price of Tesla during the day, indicating the stock's peak performance and volatility.
3. **low_tsla**: The lowest price of Tesla during the day, helping to understand support levels and market sentiment.

4. **open_tsla**: Tesla's opening price, providing insight into initial market reactions and early trading activity.
5. **volume_tsla**: The total number of Tesla shares traded, which shows the strength of market interest and liquidity.
6. **tsla_SMA_7**: The 7-day Simple Moving Average, used to smooth out short-term fluctuations and identify trends.
7. **tsla_SMA_21**: The 21-day Simple Moving Average, providing a broader view of medium-term price trends for Tesla.
8. **tsla_EMA_decay**: Exponential Moving Average with a decay factor, giving more weight to recent prices for capturing momentum.
9. **tsla_EMA_12**: The 12-day Exponential Moving Average, often used in combination with longer EMAs to generate buy/sell signals.
10. **tsla_EMA_26**: The 26-day Exponential Moving Average, commonly used in MACD calculations to indicate long-term trends.
11. **MACD**: Moving Average Convergence Divergence, used to detect changes in momentum, often signaling trend reversals.
12. **tsla_20d_stdDev**: The 20-day standard deviation of Tesla prices, capturing price volatility and risk.
13. **tsla_Upper_BB**: The upper Bollinger Band, indicating overbought conditions when prices touch or exceed this level.
14. **tsla_Lower_BB**: The lower Bollinger Band, signaling oversold conditions and potential reversal points when prices drop below it.
15. **tsla_High_Low_Spread**: The difference between the daily high and low prices, representing intraday volatility and price swings.
16. **close_spy**: The closing price of the S&P 500 ETF (SPY), used to compare Tesla's performance with the broader market.
17. **volume_spy**: The trading volume of SPY, reflecting overall market activity and liquidity, helping to contextualize Tesla's movements.



1.3 Normalizing Features

We have tried using MinMax and Standard Normalization, however find the custom percentage based scalar function more appropriate in this case.

1. TSLA price and price related features are normalized using the percentage change of the closing price of the previous day.
2. Volume and SPY price is normalized as their own value of the previous day
3. MACD, 20_std_dev and high_low_spread are normalized as the previous day's closing price.

1.4 Class Labels

In this study, we address stock price movements by defining two threshold categories: a **±2% threshold** to flag smaller price movements and a **±4% threshold** for large price movements.

However, due to the skewed nature of the data, we face an imbalanced classification problem, which is common when working with financial datasets.

Imbalanced classifications present a significant challenge in predictive modeling as most machine learning algorithms are designed under the assumption of equal class distributions. This often results in models that perform poorly, particularly for the **minority class**, which in many cases, is the most critical to predict. In our case, the minority class represents extreme price movements (beyond ±4%), which are key for higher gains and managing downside risk. These movements, while rare, are highly important for effective risk management and trading strategy optimization.

To address this, we propose the use of a **class distribution ratio** tailored to emphasize the importance of the minority class. By adjusting the model to focus on this class, we can mitigate the impact of class imbalance and improve prediction accuracy where it matters most. This method can also be extended to **anomaly detection**, identifying unusual or extreme price movements that could signal significant market events or risks.

By focusing on the minority class, we ensure that our predictive model aligns with real-world trading needs, where correctly predicting large movements (both positive and negative) is essential for maximizing gains and minimizing risk.

1.5 SVM Model Architecture

Support Vector Machine (SVM) Model Architecture

In this study, we use a **Support Vector Machine (SVM)** model with a **Radial Basis Function (RBF)** kernel to predict stock price movements. The SVM is well-suited for this task because it can handle complex patterns in the data, especially when the relationship between features and the target is non-linear, which is often the case with financial data.

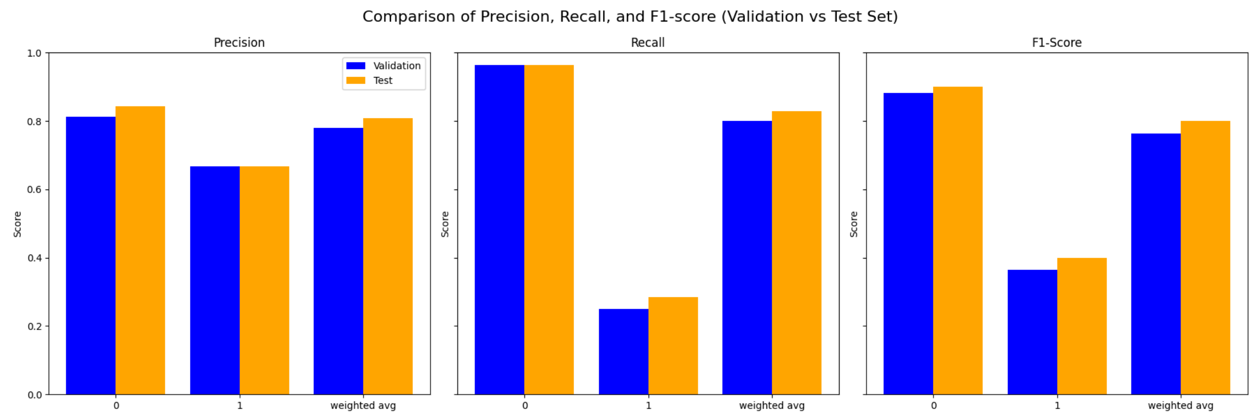
Key Features of the Model:

- **RBF Kernel**: We chose the **RBF kernel** because it helps the model capture non-linear patterns in the data by transforming the feature space.
- **Handling Imbalanced Data**: Since our data is skewed (with far fewer examples of large price movements), we applied **class weights** (`{0: 0.57, 1: 4.04}`) to give more importance to predicting these rare but significant events.
- **C Parameter**: The **C=1** parameter controls how strictly the model fits the training data.
- **Gamma**: The **gamma='scale'** setting adjusts how much influence each data point has on the decision boundary.
- **Probability Estimation**: We enabled **probability estimates** in the SVM, which allows the model to not only predict the class (small or large movement) but also give a probability for each prediction.

1.6 Model Performance

After training the model on our dataset, we evaluated its performance on both the validation and test sets using metrics like **precision, recall, and f1-score**. These metrics are especially important in imbalanced datasets because they help us understand how well the model is identifying the minority class (large price movements).

```
Validation Set Performance:

              precision    recall  f1-score   support

           0       0.81      0.96      0.88        27

           1       0.67      0.25      0.36         8

    accuracy                           0.80        35

   macro avg       0.74      0.61      0.62        35

weighted avg       0.78      0.80      0.76        35

Test Set Performance:

              precision    recall  f1-score   support

           0       0.84      0.96      0.90        28

           1       0.67      0.29      0.40         7

    accuracy                           0.83        35

   macro avg       0.76      0.62      0.65        35

weighted avg       0.81      0.83      0.80        35
```



2. CNN Model with FinBERT Embeddings

2.1 Data Cleaning

We perform basic data cleaning on the Tiingo News dataset to standardize the text data, specifically for the description and title fields, which are critical for sentiment analysis and prediction tasks. The cleaning process involves the following steps:

1. **Lowercasing:** All text is converted to lowercase to ensure uniformity and remove case sensitivity issues.
2. **Removing Non-Alphanumeric Characters:** Any characters that are not letters, numbers, or spaces (e.g., punctuation, special symbols) are removed.
3. **Removing Extra Spaces:** Multiple spaces are condensed into a single space, and leading/trailing whitespace is removed.

2.2 FinBERT Embeddings

We apply **FinBERT**, a transformer model pre-trained on financial text, to perform sentiment analysis and generate **embeddings** from financial news data. The goal is to use these embeddings as input to a **Convolutional Neural Network (CNN)** for further predictive tasks.

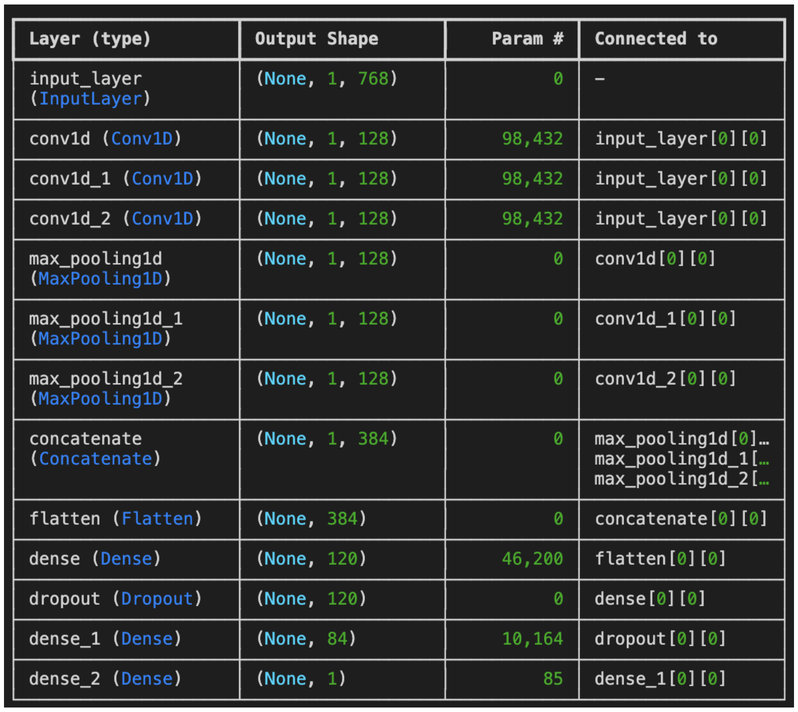
We first employ FinBERT to classify the sentiment of news descriptions as **positive** or **negative**. The **BERT tokenizer** processes each text input by truncating or padding it to a maximum length of 512 tokens. FinBERT then predicts sentiment, and the results are stored in the dataset as numerical labels (1 for positive, 0 for negative). This process ensures accurate financial sentiment classification tailored to market language.

We then generate **FinBERT embeddings** for each cleaned news description, which will be used as input features for our CNN. The **generate_embeddings** function processes the text in batches, tokenizing the input, and uses FinBERT to extract the **pooled output representations** at the **CLS token** (representing the entire sentence). These embeddings are reshaped with a dummy sequential dimension (1 x 768) to be compatible with CNN input layers.

The generated FinBERT embeddings serve as **input features** for a **Convolutional Neural Network (CNN)**. CNNs are effective at capturing patterns in sequential data, making them suitable for detecting intricate relationships in the financial news embeddings, which can then be used for tasks such as anomaly detection or predicting stock movements.

2.3 CNN Model Architecture

This model is a **Parallel Convolutional Neural Network (CNN)** designed to process **FinBERT embeddings** of financial news data. The architecture leverages multiple convolution layers to extract meaningful patterns from the input embeddings.

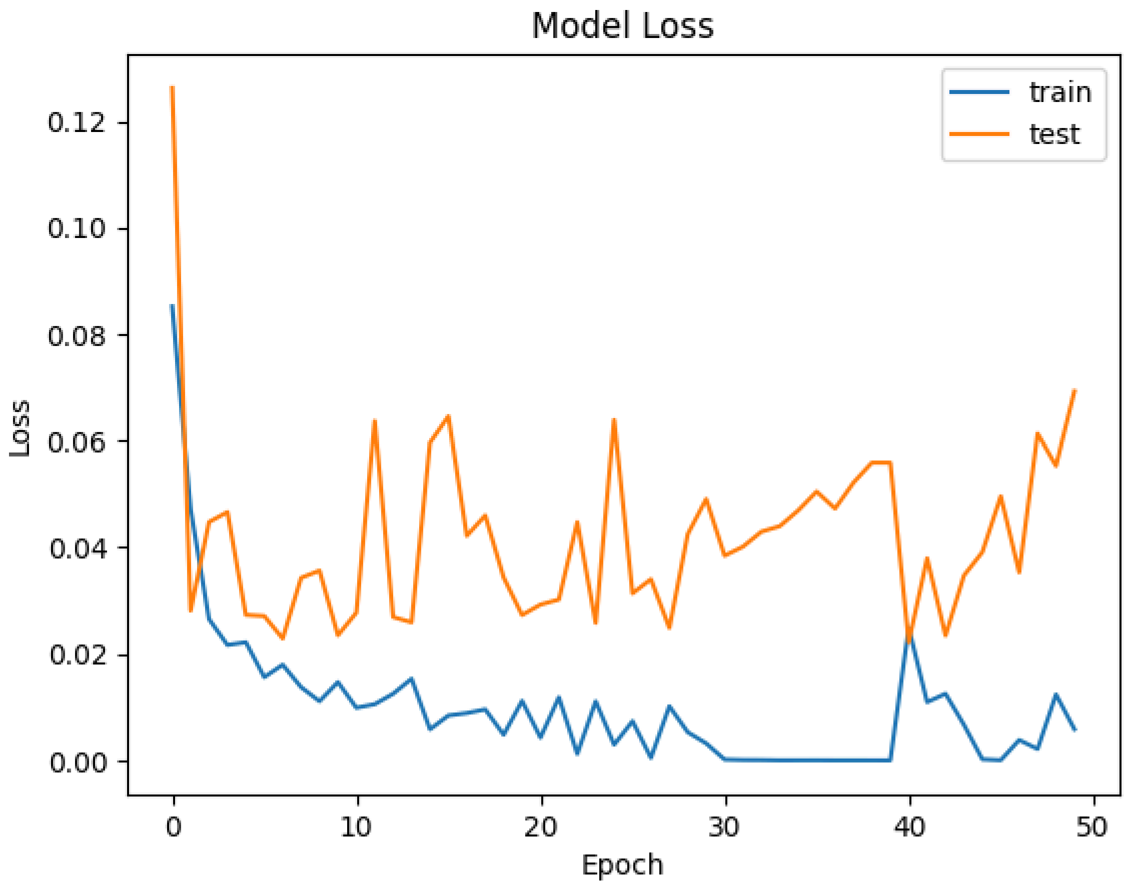


```
┏━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━┓
┃ Layer (type)        ┃ Output Shape      ┃    Param # ┃ Connected to      ┃
┡━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━┩
│ input_layer         │ (None, 1, 768)    │          0 │ -                 │
│ (InputLayer)        │                   │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ conv1d (Conv1D)     │ (None, 1, 128)    │     98,432 │ input_layer[0][0] │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ conv1d_1 (Conv1D)   │ (None, 1, 128)    │     98,432 │ input_layer[0][0] │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ conv1d_2 (Conv1D)   │ (None, 1, 128)    │     98,432 │ input_layer[0][0] │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ max_pooling1d       │ (None, 1, 128)    │          0 │ conv1d[0][0]      │
│ (MaxPooling1D)      │                   │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ max_pooling1d_1     │ (None, 1, 128)    │          0 │ conv1d_1[0][0]    │
│ (MaxPooling1D)      │                   │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ max_pooling1d_2     │ (None, 1, 128)    │          0 │ conv1d_2[0][0]    │
│ (MaxPooling1D)      │                   │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ concatenate         │ (None, 1, 384)    │          0 │ max_pooling1d[0]… │
│ (Concatenate)       │                   │            │ max_pooling1d_1[… │
│                     │                   │            │ max_pooling1d_2[… │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ flatten (Flatten)   │ (None, 384)       │          0 │ concatenate[0][0] │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ dense (Dense)       │ (None, 120)       │     46,200 │ flatten[0][0]     │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ dropout (Dropout)   │ (None, 120)       │          0 │ dense[0][0]       │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ dense_1 (Dense)     │ (None, 84)        │     10,164 │ dropout[0][0]     │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ dense_2 (Dense)     │ (None, 1)         │         85 │ dense_1[0][0]     │
└─────────────────────┴───────────────────┴────────────┴───────────────────┘
```

Key Components:

1. **Input Layer**: The model takes an input shape of **(1, 768)**, representing the FinBERT embedding of a news article with a dummy sequential dimension.
2. **Parallel Convolutional Layers**: Three parallel **1D Convolution layers** are used, each with **128 filters** and a kernel size of **1**. These layers extract features from the embedding. Each convolution layer is followed by **MaxPooling**, where the pool size is set to the length of the input sequence to reduce dimensionality and focus on the most important features.
3. **Concatenation**: The outputs from the three parallel convolutional layers are **concatenated** and then **flattened** to create a combined feature representation.
4. **Fully Connected Layers**: The concatenated features are passed through two **fully connected (Dense) layers**: the first with **120 units** and the second with **84 units**, both using the **ReLU** activation function. A **Dropout** layer (with a dropout rate of 0.5) is added to reduce overfitting by randomly dropping neurons during training.
5. **Output Layer**: The final layer has **1 output unit** with a **softmax** activation function, indicating a binary classification task (e.g., sentiment prediction as positive or negative).

6. **Compilation**: The model is compiled using the **Adam optimizer** and **binary cross-entropy loss**, optimized for binary classification tasks, with **accuracy** as the evaluation metric.



3. FUSION MODEL

We developed a **Fusion Model** combining outputs from a **Technical Analysis (TA) SVM** and a **CNN-based sentiment model** for predicting significant price movements in TSLA stock. The model was designed to predict whether the next day's stock price would increase by **more than 4%**, using the probabilities from both models as input features.

Input Features:

1. **Positive_Class (TA SVM Probability)**: Represents the probability of the price increase based on technical indicators.
2. **Predictions (CNN Probability)**: Represents the probability based on the sentiment analysis derived from news articles.

Target Variable:

The target variable is binary, indicating whether the stock price increased by **more than 4%** the next day (1) or not (0), based on the `pct_change` in the stock price.

Modeling Approach:

- **Support Vector Machine (SVC)**: A **radial basis function (RBF) kernel** was used for classification.
- **Feature Scaling**: Input features were scaled using **StandardScaler** to normalize

4. Backtesting and Trading Strategy Overview

The algorithm employs a hybrid approach, combining the Fama-French **4-factor model** with a **prediction-based trading strategy** for tech stocks. The trading strategy is focused on tech stocks for which a model generates price movement predictions, while the **Fama-French factors** are used to diversify the portfolio across non-tech stocks.

This approach allows for **systematic exposure to key market factors** while capitalizing on specific **price predictions** in the tech sector. The strategy also includes **risk management features** such as diversification, rebalancing, slippage, and transaction fees.

4.1 Key Components of the Trading Strategy

1. **Tech Stock Predictions (5th Factor)**:

A **machine learning model** (e.g., CNN) predicts price movements for key tech stocks. If the **predicted price movement** exceeds a threshold (e.g., **0.95 probability of upward movement**), the stock is selected for investment. Each tech stock with a strong prediction signal is allocated a **fixed weight (10%)** in the portfolio.

2. **Fama-French 4-Factor Model**:

The **Fama-French model** is a well-established factor model used to explain stock returns. Factors include:

1. MKT (Market Risk Premium): Exposure to general market movements.
2. SMB (Size - Small Minus Big): Exposure to smaller companies versus larger ones.
3. HML (Value - High Minus Low): Exposure to value stocks versus growth stocks.
4. MOM (Momentum): Exposure to stocks with positive momentum.

These factors guide the allocation of the **non-tech stocks** in the portfolio. Non-tech stocks are selected based on these factors and receive the **remaining portfolio weight**.

3. **Equal Weighting and Daily Rebalancing**. The algorithm rebalances the portfolio daily based on the predictions for tech stocks and the Fama-French factors for diversification.

The allocation is recalculated every day, ensuring that tech stocks with strong prediction signals are given priority (up to 10% allocation per stock).

4.2 Risk Management Features

1. **Diversification**: The portfolio is diversified across both **tech stocks** (using predictions) and **non-tech stocks** (using Fama-French factors). This limits exposure to individual stock risk. The strategy allocates up to **40%** of the portfolio to the **Fama-French market risk factor** and spreads the remaining across the **size, value, and momentum factors**. This provides balanced exposure to different market risks.
2. **Dynamic Rebalancing**: The strategy rebalances daily based on **predictions** and **market factors**. This ensures that the portfolio adapts to new information and market conditions. **Tech stocks** are reweighted every day based on the **latest predictions,** allowing the strategy to reduce exposure when predictions are not favorable.
3. **Slippage and Transaction Fees**: The strategy incorporates **constant slippage (0.02%)** to account for the cost of trading execution slippage. **Interactive Brokers' fee model** is applied to simulate real-world transaction costs, ensuring that the strategy considers the impact of trading fees on returns.
4. **Position Sizing and Thresholds**: The algorithm uses **position sizing** by limiting the tech stock allocation to 10% per stock if the **prediction probability** exceeds a defined threshold (0.95). This prevents over-concentration in any single stock and ensures that **position sizing** is aligned with the strength of the predictions.
5. **Exposure to Market Factors**: Non-tech stock allocations are determined by the **Fama-French factors**, ensuring that the strategy is exposed to **systematic risk factors** that have been shown to explain stock returns historically.

4.3 Trading Strategy Execution

1. **Portfolio Construction**: **Tech stocks** are selected based on **predictions**, while **non-tech stocks** are selected based on the **Fama-French factors**. Each tech stock with a positive signal is allocated up to 10%, while the rest of the portfolio weight is spread across non-tech stocks using the factor-based strategy.
2. **Daily Rebalancing**: The portfolio is rebalanced every day based on the predictions and factor weights. This allows the strategy to quickly adapt to changes in market conditions and stock-specific news or predictions.
3. **Risk Management in Rebalancing**: If the prediction signal is weak or absent, tech stocks do not receive any allocation, and the portfolio remains allocated to **safer, more diversified holdings** (non-tech stocks). This reduces the risk of holding tech stocks during periods of uncertainty or low-confidence predictions.

Our Strategy outperformed the benchmark for 2019-01-01 to 2019-12-31 however, did not perform well for 2 Years and 3 Years period. Please check the backtest report for complete details.

Credits & Acknowledgement

1. This research is inspired by **Yanzhao Zou, Dorien Herremans Prebit - A multimodal model with Twitter FinBERT embeddings for extreme price movement prediction of Bitcoin.**
2. I would like to express my sincere gratitude to **Professor Dimitry Udler** and **Professor N. K. Chidambaran, Fordham University** for their continuous guidance and support.