CQF Final Project Report

Project Code: DL

Project Title:

Deep Learning for Asset Trend Prediction

Sub-Title:

LSTM Models for SPY ETF Trend Prediction and Backtesting With Trading Strategy

Candidate Name:

Nayan Patel

Cohort:

January 2024

Table of Contents

Introduction

Project Overview

In the dynamic and highly competitive world of financial markets, the ability to predict asset price movements is of immense value. This project focuses on developing a predictive model using Long Short-Term Memory Networks (LSTM) to forecast the daily upward trend of the SPY ETF, an exchange-traded fund that tracks the performance of the S&P 500 index.

The core objective of this project is to leverage the sequential modeling capabilities of LSTM networks to accurately predict whether the SPY ETF will experience a significant positive return on the next trading day. The project aims to provide a robust model that can be used within a trading strategy to generate buy and sell signals, ultimately improving trading outcomes.

The analysis spans a comprehensive range of activities, from feature engineering and data transformation to model building, hyperparameter tuning, and backtesting. The final deliverable is a fully developed trading strategy based on the LSTM model's predictions, evaluated against industry-standard performance metrics.

Significance of the Study

Accurate prediction of financial markets is a coveted skill that can significantly impact trading strategies and investment decisions. The ability to forecast daily price movements of the SPY ETF, which mirrors the S&P 500 index, is particularly valuable due to its role as a benchmark for the overall performance of the U.S. equity market.

The SPY ETF is one of the most widely traded ETFs in the world, and its price movements are closely watched by traders, investors, and financial institutions. A reliable model that can predict its short-term trends can serve as a powerful tool for developing trading strategies that maximize returns while managing risk.

This study's significance lies in its focus on using advanced machine learning techniques, specifically Long Short-Term Memory Networks (LSTM), to capture complex temporal dependencies in financial data. Traditional statistical methods often fall short in this regard, as they may not fully account for the sequential nature of market data. LSTMs, with their ability to retain information across long sequences, offer a promising alternative for making more informed predictions.

Moreover, by framing the problem as a binary classification task, this study simplifies the decision-making process in trading, providing clear buy or sell signals based on the predicted trends. This approach is not only practical but also aligns with how many automated trading systems operate in real-world financial markets.

The study also emphasizes the importance of rigorous backtesting to evaluate the model's performance in a realistic trading environment. By simulating trades based on the model's predictions, the study ensures that the model's efficacy is tested against actual market conditions, providing valuable insights into its potential profitability and robustness.

In summary, this study contributes to the field of financial forecasting by exploring the application of LSTM networks to predict SPY ETF trends. Its findings could have significant implications for traders, investors, and financial analysts seeking to enhance their market prediction capabilities and optimize their trading strategies.

Tools and Technologies

In this project, I employed a variety of tools and technologies to implement, train, evaluate, and backtest our predictive model. Below is a comprehensive list of the Python libraries, APIs, and other tools used, along with their respective roles in the project:

1. Python Libraries:

- **Pandas**:
  - Used for data manipulation and analysis, including loading, cleaning, and transforming the datasets.
- **NumPy**:
  - Utilized for numerical operations, especially for handling arrays and matrices during data preprocessing and model input preparation.
- **Matplotlib & Seaborn**:
  - These libraries were used for data visualization, helping to generate plots, charts, and graphs that illustrate the data distribution, correlations, and model performance.
- **Statsmodels**:
  - Employed for statistical analysis and hypothesis testing, particularly in the Exploratory Data Analysis (EDA) phase.
- **Scikit-Learn**:
  - Used for various machine learning tasks, including data preprocessing, feature selection (Recursive Feature Elimination), model evaluation, and computing performance metrics like accuracy, precision, and recall.
- **TensorFlow & Keras**:
  - The core libraries for building, training, and optimizing the Long Short-Term Memory (LSTM) models. TensorFlow serves as the backend, while Keras provides a high-level API for model construction.
- **Keras Tuner**:
  - Used for hyperparameter tuning, enabling the selection of the best model configurations by experimenting with different sets of hyperparameters.
- **QuantStats**:
  - A library used for financial performance analysis and backtesting. It provided detailed performance metrics, comparisons, and visual reports for evaluating the trading strategy based on the model’s predictions.
- **SciPy**:
  - Utilized for advanced statistical functions and operations, including handling distributions and performing optimizations.

- **MiniSom**:
    - Employed for implementing the Self-Organizing Map (SOM) used in the feature selection and transformation process, particularly for visualizing and reducing high-dimensional data.
- **Pandas TA**:
    - Specifically used for extracting and calculating a comprehensive set of technical indicators that are essential for feature engineering. These indicators were critical for enhancing the predictive power of the model.

2. APIs:

- **AlphaVantage Premium API**:
    - This API was used to fetch historical price data, including daily Open-High-Low-Close-Adjusted Close (O-H-L-C-Ad Close) data for the SPY ETF and other assets, as well as technical indicators.
- **FRED API**:
    - Used to retrieve Treasury yield data (2-year and 10-year) from the Federal Reserve Economic Data (FRED) database, which was essential for calculating yield spreads and including macroeconomic indicators in the dataset.
- **MacroTrends LLC**:
    - Sourced the S&P 500 P/E ratio data, which was downloaded as a CSV file and converted to daily data for use in our analysis. This data was crucial for incorporating valuation metrics into our feature set.

3. Other Tools:

- **Jupyter Notebook**:
    - The primary environment for developing, testing, and documenting the code, as well as for visualizing results and creating interactive notebooks for this project.
- **TensorBoard**:
    - Integrated for visualizing the training process of the models, including monitoring metrics like loss and accuracy in real-time.

---

2. Dataset Overview

In this project, the analysis is conducted on a comprehensive dataset compiled from various reliable sources, covering the period from 1 July 2017 to 1 July 2024. The dataset includes multiple features that are essential for predicting the daily upward trend of the SPY ETF. The primary focus is on leveraging financial data, technical indicators, and macroeconomic variables to build a robust predictive model.

Data Structure for the Extracted Dataset

This diagram illustrates the structured layout of the dataset used in this project. The data is categorized into three main sections:

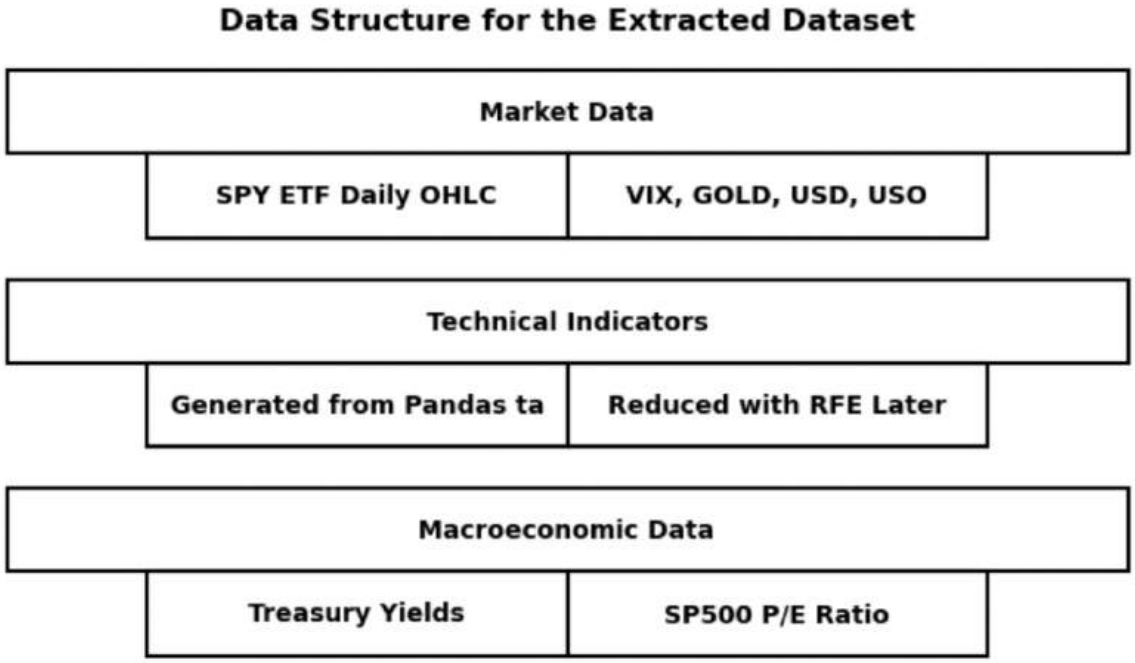


Figure 1: Data Structure for the Extracted Dataset

This diagram illustrates the structured layout of the dataset used in this project. The data is categorized into three main sections:

1. **Market Data**: This includes SPY ETF's daily OHLC (Open, High, Low, Close) data and the returns of key financial instruments such as VIX, GOLD, USD, and USO. These elements provide foundational market insights crucial for trend prediction.
2. **Technical Indicators**: Generated using the Pandas TA library, these indicators include moving averages, RSI, MACD, and others. These features are vital for capturing market momentum and identifying potential trading signals. The indicators were later refined through Recursive Feature Elimination (RFE) to improve model performance.
3. **Macroeconomic Data**: This includes Treasury Yields and the SP500 P/E ratio, which are key economic indicators influencing market behaviour and SPY ETF performance.

3. High-Level Project Architecture

The diagram below provides an overview of the project's architecture, capturing the sequential flow from data collection to model evaluation. This framework ensures a systematic approach to building and optimizing a predictive model for the SPY ETF.

It starts with collecting and processing relevant data, followed by feature engineering, and then moves through model building, hyperparameter tuning, and finally, the evaluation of the model's performance through backtesting.

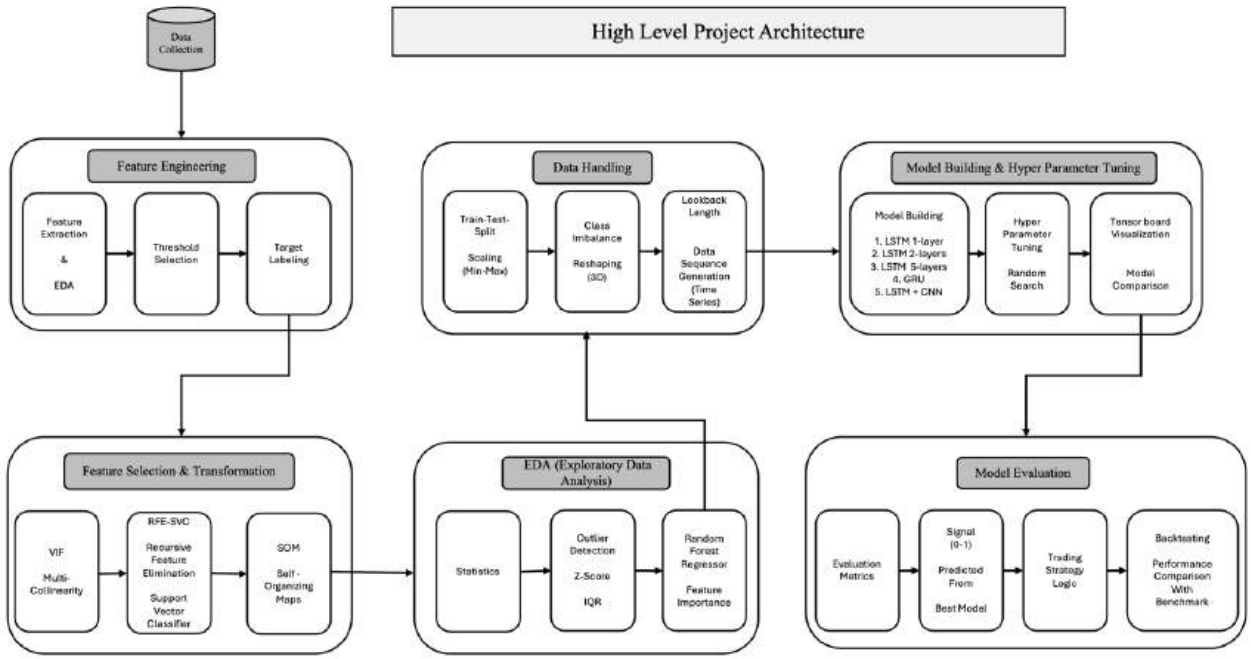


Fig. 2: High Level Project Flowchart/Diagram.

At the core of this architecture is the integration of various data types, including market data, technical indicators, and macroeconomic factors, into a unified dataset.

The project progresses through essential stages: from Feature Engineering, where features are extracted, selected, and transformed, to Model Building & Hyperparameter Tuning, where multiple LSTM models are built and optimized. The final stage involves rigorous Model Evaluation, including performance metrics and a backtested trading strategy.

4. 7-Steps Model Building

Step 1: Objective

In this project, the primary objective is to develop a predictive model that can forecast the daily upward trend of the SPY ETF. The SPY ETF, which tracks the performance of the S&P 500 index, serves as a broad representation of the U.S. stock market. Predicting its daily price movements, particularly its positive returns, is a challenging yet rewarding task, given the inherent volatility and noise in financial time series data.

The problem is formulated as a binary classification task where the goal is to predict whether the SPY ETF will experience a significant positive return on the next trading day. A positive return is defined as any daily return exceeding a threshold of 0.20%. If the model predicts that the return will surpass this threshold, a buy signal (labelled as 1) is generated; otherwise, a sell signal (labelled as 0) is issued.

To achieve this, the project utilizes Long Short-Term Memory Networks (LSTM), which are well-suited for handling sequential data and capturing temporal dependencies. The LSTM model is trained and optimized using historical data from the past seven years, from 1 July 2017 to 1 July 2024. The data is sourced from multiple reliable APIs, including AlphaVantage, FRED, and Macrotrends, and includes a comprehensive set of features such as price data, technical indicators, Treasury yields, and the S&P 500 P/E ratio.

The project also addresses key challenges associated with financial time series forecasting, such as the high noise-to-signal ratio and the stochastic nature of market data. The final model's predictions are rigorously tested within a backtesting framework, evaluating its practical effectiveness through a simulated trading strategy.

Step 2: Feature Engineering

- Feature Engineering plays a crucial role in transforming raw financial data into meaningful inputs for our Deep Learning models.
- The process involves careful data collection, selection, extraction, and transformation of features that are most likely to influence the prediction of SPY ETF's daily trend.
- Collecting relevant data, including SPY's daily price and technical indicators, Treasury yields, and other asset data, which are all integrated into a comprehensive dataset.
- Exploratory Data Analysis (EDA) is performed to understand the relationships between features and ensure data quality.
- The target variable is then defined based on a threshold, and the dataset is adjusted for class imbalance to ensure robust model training.

2.1) Feature Extraction:

In this phase, various features are extracted from the raw data, which includes SPY ETF daily OHLC (Open, High, Low, Close) data, VIX, GOLD, USD, and USO data, as well as technical indicators generated using the 'pandas_ta' library.

The primary goal is to create features that capture relevant market signals which can potentially improve the model's predictive power. These features are critical inputs for the model, and careful selection and engineering of these features can significantly impact model performance.

Table 2.1 : Data and Features Category and Purpose

Category	Data Type	Source	Timeframe	Purpose in Project
Market Data	SPY ETF Daily OHLC	AlphaVantage Premium API	1 July 2017 - 1 July 2024	Provides foundational market insights and raw price data for analysis.
	VIX, GOLD, USD, USO	AlphaVantage Premium API	1 July 2017 - 1 July 2024	Captures volatility, commodity prices, and currency movements influencing SPY ETF.
Technical Indicators	340 Indicators Initially. Reduced using RFE.	Generated using Pandas TA library	1 July 2017 - 1 July 2024	Provides momentum-based signals for market trend prediction.
Macroeconomic Data	Treasury Yields	FRED API	1 July 2017 - 1 July 2024	Reflects the interest rate environment's influence on market behaviour.
	SP500 P/E Ratio	MacroTrends LLC	1 July 2017 - 1 July 2024	Indicates market valuation, aiding in predicting overvalued or undervalued conditions.

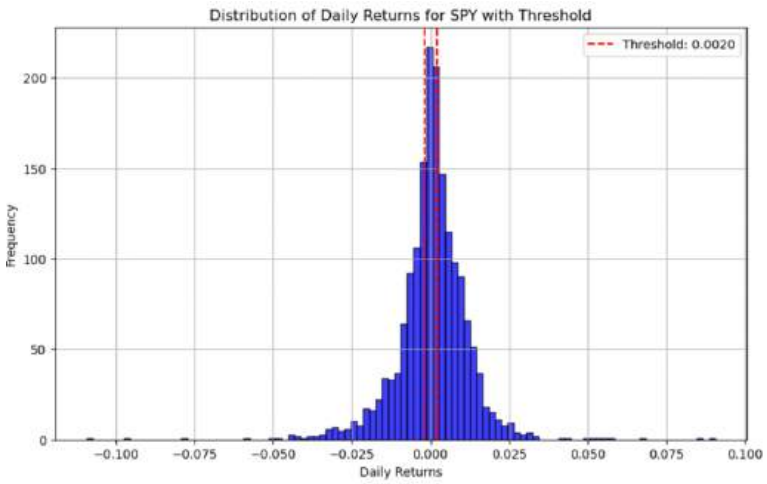
The 2.1 table provides a detailed overview of the data sources, their timeframes, and their specific roles in the predictive model.

This comprehensive dataset was meticulously curated to ensure that the model has access to the most relevant and up-to-date information, thereby improving its accuracy and robustness in predicting daily trends in the SPY ETF. Each category of data plays a vital role in shaping the model's predictions, from foundational market data to advanced technical indicators and macroeconomic metrics.

2.2) Threshold Selection

After the initial feature extraction, the threshold for the target variable is determined. The threshold is set at 0.20%, which classifies the daily return into two categories: 1 (uptrend) if the return is above 0.20% and 0 (neutral or downtrend) otherwise.

This threshold is chosen based on the distribution of the daily returns and the need to balance the trade-off between capturing upward trends and avoiding false positives.



The threshold selection is a crucial step as it defines the binary classification problem that the model will solve.

2.3) Target Labeling

The label or the target variable is also known as the dependent variable. In this project, the target variable is defined as whether the S&P 500 ETF (SPY) will experience a significant positive return based on a threshold of 0.20% in its daily return.

If the daily return exceeds this threshold, the label is set to +1 (indicating a buy signal). Otherwise, it is set to 0.

A new ['Target'] column is created to store these labels. The target can be mathematically described as:

$$y_t = \begin{cases} 1 & \text{if } r_t > 0.002 \\ 0 & \text{otherwise} \end{cases}$$

$$r_t = \frac{P_t - P_{t-1}}{P_{t-1}}$$

The target variable y is then used in the model to predict whether the SPY ETF's price will increase significantly based on the defined threshold.

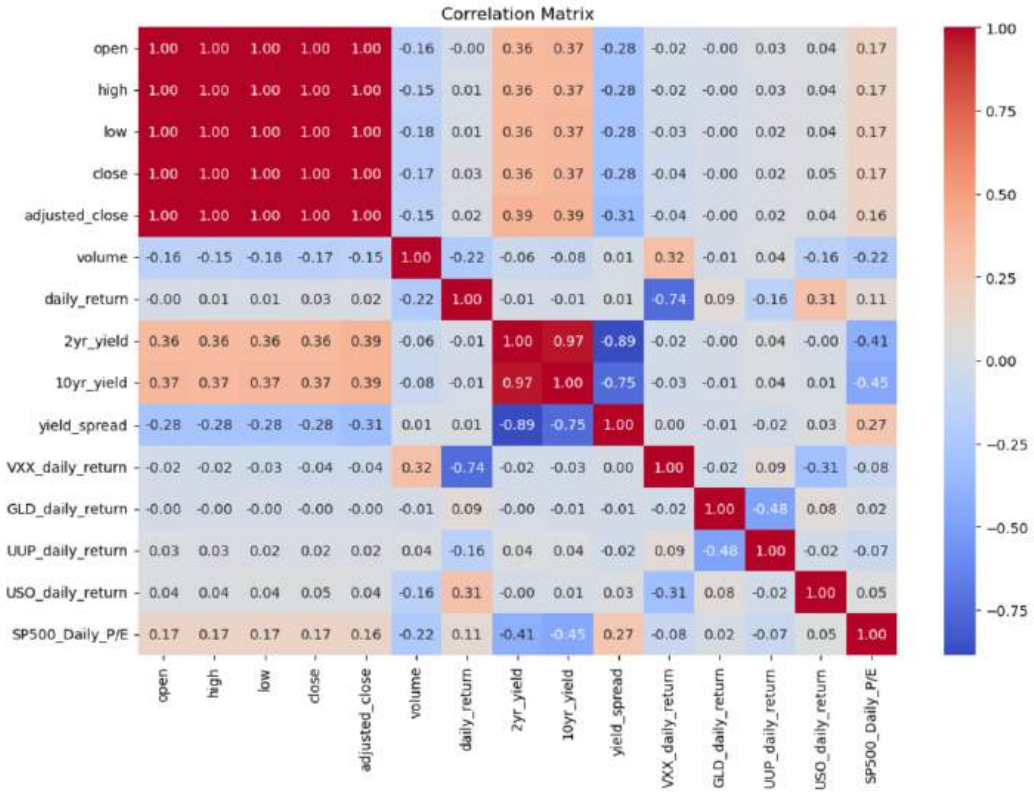
This binary classification approach simplifies the problem and aligns the model's output with the intended trading strategy

2.4) EDA on Current Dataset (Excluding Technical Indicators)

Exploratory Data Analysis (EDA) is a critical step in understanding the underlying structure of the data. In this step, we generate summary statistics for the dataset, visualize the distribution of the SPY ETF's daily returns, and explore the relationships between different features using a correlation matrix and pair plots.

EDA helps in identifying key patterns, potential outliers, and the degree of correlation between features, all of which are essential for informing the next steps in feature engineering and model building.

Fig. 2.4: Correlation matrix on current features (excluding ta)



The correlation matrix provided showcases the relationships between various features, including the SPY ETF's daily returns, volume, Treasury yields, and other macroeconomic indicators. This visualization is instrumental in detecting multicollinearity and determining the most relevant features for the predictive model.

The Main EDA is performed under Step-4 on the Final Feature Dataset

Dataset Creation: After extracting the features and completing the feature engineering steps, the dataset was created by merging all relevant data sources, including market data, technical indicators, and macroeconomic data.

The final dataset, named **'SPY_All_Features.csv'**, includes all the engineered features and target labels, providing a comprehensive dataset for subsequent model training and evaluation.

Step 3: Feature Selection and Transformations

Feature selection and transformation are critical steps in refining the dataset to improve model performance and interpretability.

By carefully selecting the most relevant features and transforming them as needed, I was able to reduce the complexity of the model, enhance its accuracy, and prevent overfitting.

Step 3.1) Multicollinearity Analysis Using Variance Inflation Factor (VIF) on All Feature Set

Multicollinearity occurs when two or more features in the dataset are highly correlated, leading to redundancy and potentially skewing the model's results. To address this issue, I performed a multicollinearity analysis using the Variance Inflation Factor (VIF).

VIF quantifies how much the variance of a regression coefficient is inflated due to multicollinearity among the features. A high VIF value indicates that the feature is highly correlated with one or more other features, making it a candidate for removal.

Process:

- **Calculation of VIF:** VIF was calculated for all features in the dataset.
- **Iterative Removal:** Features with excessively high VIF values (Inf or NaN) were iteratively removed from the dataset. The goal was to ensure that the remaining features were more independent of each other, which helps in improving the model's stability and interpretability.
- **Final VIF Check:** After removing the high VIF features, a final check was conducted to ensure that all remaining features had acceptable VIF values.

```
vif_data_cleaned
```

	Feature	VIF
327	VHM_610	1.058060
113	CDL_TASUKIGAP	1.006239
89	CDL_HIGHWAVE	1.002550
109	CDL_SPINNINGTOP	1.001606
248	REFLEX_20_20_0.04	1.000021
...	...	...
148	ENTP_10	0.008357
124	CG_10	0.007231
211	NVI_1	0.005617
233	PVI	0.000086
0	const	0.000000

343 rows × 2 columns

Initially, the dataset contained 355 features, but after removing those with Inf or NaN VIF values, the feature set was reduced to 343. This step ensured that the dataset was better suited for model training, leading to more robust and reliable predictions.

The VIF analysis was just the first step in feature selection. In the following steps, I further refined the feature set through techniques like Recursive Feature Elimination (RFE) and Self-Organizing Maps (SOM), ensuring that only the most relevant features are used in the final model.

Step-3.2) Feature Selection using RFE (Recursive Feature Elimination)

Objective: The goal of the SVC-RFE process is to systematically eliminate features that contribute the least to the predictive power of the model. This helps in reducing the feature space, leading to a more efficient and interpretable model without sacrificing accuracy.

Process Overview:

1. Initial Feature Set:
    - We started with the full set of features generated through the Feature Engineering phase, including technical indicators, macroeconomic data, and market data.
    - The initial feature set consisted of 343 features after removing features with high multicollinearity (as indicated by high Variance Inflation Factor (VIF) values).

2. Iterative Feature Elimination:

    - Step 1: Feed the entire set of features into a Support Vector Classifier (SVC) model. The SVC is trained on this feature set to predict the target variable (whether the SPY ETF's daily return exceeds 0.20%).

    - Step 2: Calculate the importance of each feature. The SVC model provides a ranking of features based on their contribution to the model's predictive power.

    - Step 3: The feature with the lowest importance (least contribution to the model's accuracy) is removed from the feature set.

    - Step 4: The model is retrained with the reduced feature set, and the process repeats from Step 1 until the desired number of features is reached or until further elimination would harm model performance.

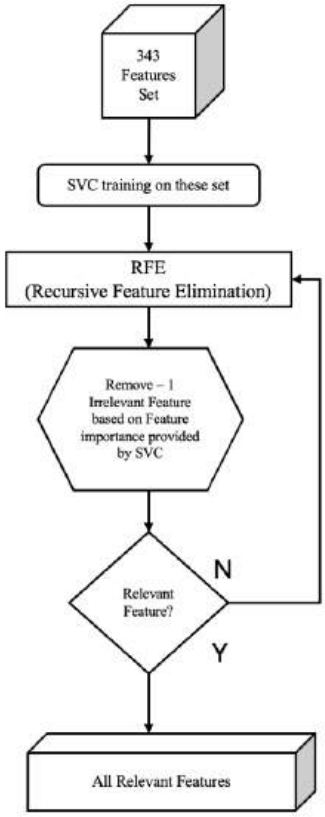


Fig: 3.2: RFE process

Feature Reduction Results: Through the iterative RFE process, the original set of 343 features was systematically reduced. The selection criteria for the final set of features were based on maintaining a balance between model complexity and predictive performance.

The RFE process concluded with a refined set of 50 features, which were found to contribute most significantly to the predictive power of the model. These features were retained for further model building and evaluation.

Testing the 50 Features selected by RFE process:

After completing the RFE process, the final 50 features were evaluated on both the train and test datasets.

The objective was to verify that the reduced feature set still performs well on unseen data without sacrificing model accuracy.

Below is the Test Classification Report and the Test AUC of the SVC model on the test set with the 50 selected features:

Test Classification Report:

	precision	recall	f1-score	support
0.0	0.98	0.81	0.89	207
1.0	0.78	0.98	0.87	143
accuracy			0.88	350
macro avg	0.88	0.90	0.88	350
weighted avg	0.90	0.88	0.88	350

Test Accuracy and AUC:

Test Accuracy: 0.88

Test AUC: 0.9761

- The accuracy of 88% indicates strong predictive performance even after the feature reduction.
- AUC of 0.9761 highlights excellent discrimination capability of the model between classes.
- Precision and recall metrics for both classes show good balance, with high recall for class 1, which indicates the model's ability to capture most positive returns.

With this refined set of 50 features, In the next step, I applied Self-Organizing Maps (SOM) to further analyse the dataset and remove any overlapping features.

Step-3.3) Self-Organizing Maps (SOM)

Objective: The primary goal of applying Self-Organizing Maps (SOM) in this project is to identify and visualize potential feature overlaps or redundancies within the dataset.

SOM, a type of unsupervised learning, is particularly effective in high-dimensional data visualization and clustering, enabling the identification of similar features that may not add significant value to the model.

Process Overview: Self-Organizing Maps (SOM) operate by projecting high-dimensional data into a lower-dimensional (typically 2D) grid.

This grid is composed of nodes or neurons, where each node represents a cluster of similar data points. By analysing the distribution and organization of these nodes, we can identify clusters of features that are too similar to each other, leading to potential redundancies in the dataset.

Application in the Project:

1. Data Preparation:
    - The selected 50 features from the RFE process are normalized to ensure uniformity in the range of values, which is crucial for effective SOM clustering.
2. SOM Configuration:
    - A 20x20 grid is used to map the features. The grid size is chosen based on a balance between granularity and interpretability.
3. Training the SOM:
    - The SOM is trained over multiple iterations, allowing the nodes to self-organize and cluster similar features together. This iterative training process ensures that the SOM accurately reflects the underlying structure of the data.

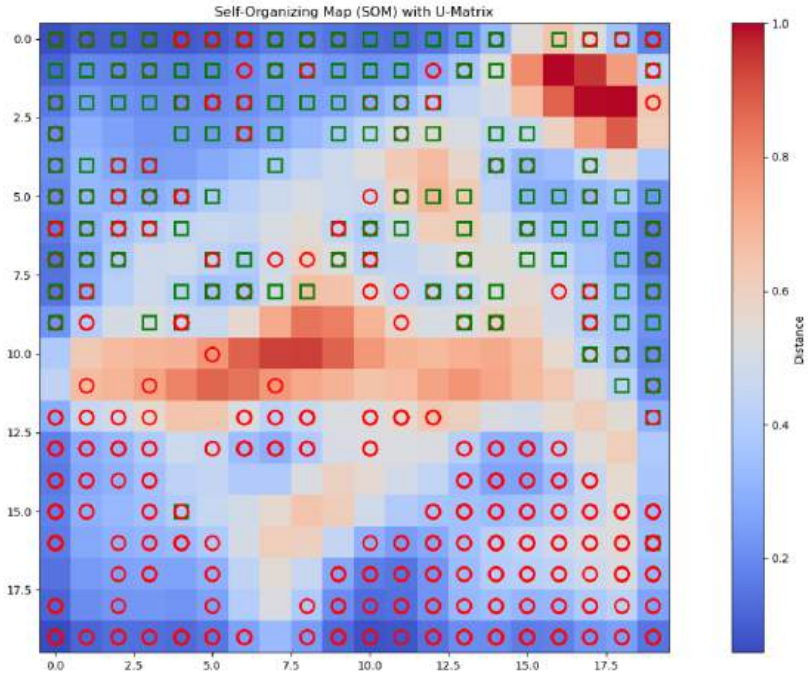


Fig: 3.4) U-Matrix SOM to Identify Overlapping Features

4. Visualization and Analysis:
    - The resulting SOM grid is visualized, and a U-Matrix (Unified Distance Matrix) is generated to highlight clusters and boundaries between different regions of the map. Regions with tightly packed nodes indicate high similarity between features, suggesting potential redundancies.
5. Feature Reduction:
    - Features that are clustered closely together on the SOM are further analysed for redundancy. Those with minimal impact on model performance are removed, streamlining the feature set for the final model building phase.

Outcome: The SOM analysis leads to a more refined feature set by identifying and removing redundant features. This not only reduces the complexity of the model but also enhances its generalizability by focusing on the most informative and diverse set of features.

Using the Self-Organizing Map (SOM) combined with outlier detection techniques like Interquartile Range (IQR) and Z-Score (in EDA Next Step) , the feature set was further refined. This process effectively reduced the initial 50 features down to 24, ensuring that only the most relevant and non-overlapping features were retained for the final model. This reduction enhances model efficiency and reduces the risk of overfitting, while maintaining predictive accuracy.

Step-4: EDA on Final Feature Set

After the feature selection and transformations, the final dataset with 24 features was thoroughly examined through Exploratory Data Analysis (EDA). This step is crucial to ensure that the refined features align well with the target variable and to identify any remaining issues like outliers or multicollinearity.

4.1) Statistical Summary:

- A statistical summary was generated to provide an overview of the central tendencies, spread, and overall distribution of each feature.
- This summary helps in identifying potential anomalies or features that may require further transformation.

4.2) Outlier Detection Using Z-Score and IQR:

- Outliers were identified and capped iteratively using the Z-score and IQR (Interquartile Range) methods. This was essential to minimize the impact of extreme values on the model's predictions.

```
# Capping outliers to the lower and upper bounds of the IQR
df_capped = numeric_df.copy()
for col in numeric_df.columns:
    lower_bound = Q1[col] - 1.5 * IQR[col]
    upper_bound = Q3[col] + 1.5 * IQR[col]
    df_capped[col] = np.where(numeric_df[col] < lower_bound, lower_bound, numeric_df[col])
    df_capped[col] = np.where(df_capped[col] > upper_bound, upper_bound, df_capped[col])
```

```
daily_return        118
yield_spread          0
VXX_daily_return    104
UUP_daily_return     40
volume_tech          81
ADOSC_3_10           29
ADX_14               48
OBV                   0
AOBV_SR_2             0
BBP_5_2.0             1
BOP                   0
CCI_14_0.015          7
DEC_1                 0
BULLP_13             90
KVO_34_55_13         46
PDIST                77
PIVOTS_TRAD_D_P       1
PIVOTS_TRAD_D_R3      1
PSL_12                0
PVOL                 57
PVT                   0
SLOPE_1             125
SMCtp_14_50_20_5     28
STDEV_30             79
Target                0
dtype: int64
```

Fig: 4.2) Outlier detection and Capping with IQR

- By capping these outliers, the dataset's robustness and reliability were enhanced, leading to better generalization in the model.

4.3) Key Pair Plot:

- A key pair plot was created to visualize the relationships between the most important features and the target variable. This plot reveals any potential patterns or dependencies that may exist between the features.

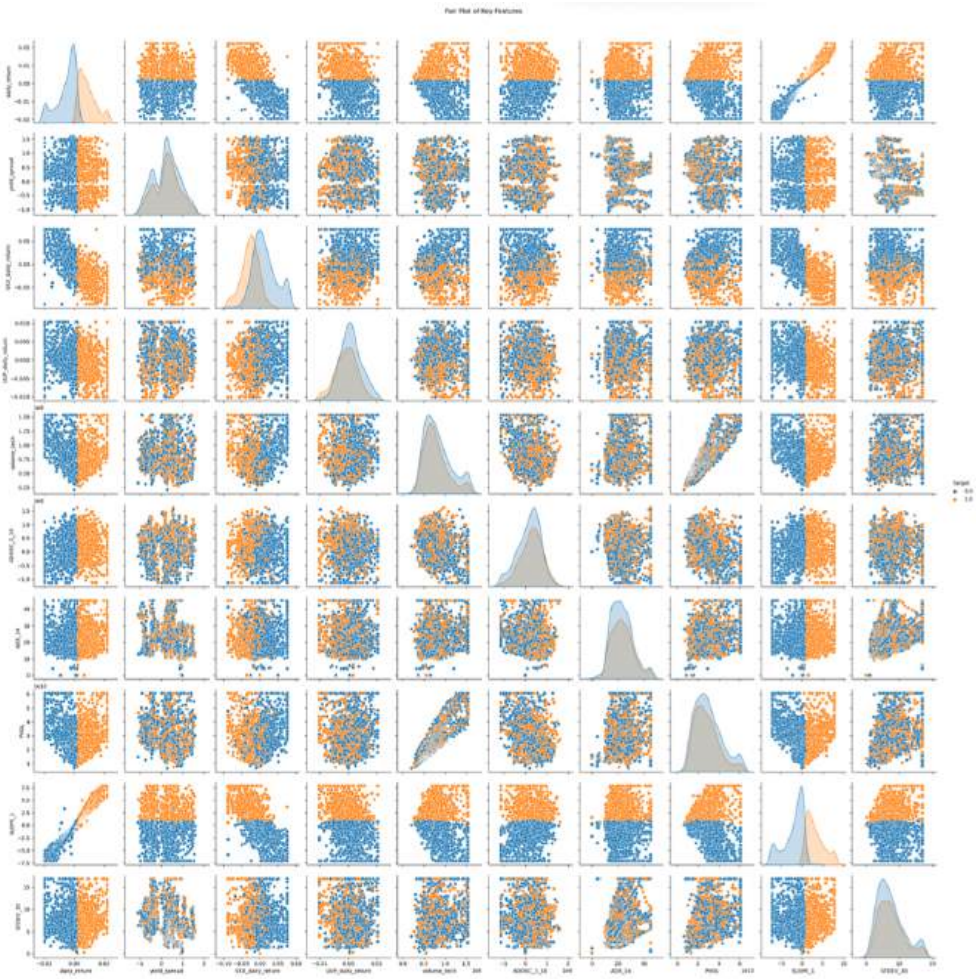


Fig: 4.3) Key pair plot with final features

- Overlapping Features: In some scatter plots, the classes overlap significantly (e.g., VXX_daily_return vs. UUP_daily_return), suggesting these features alone may not provide strong predictive power.
- Separation in Specific Plots: Some feature pairs like SLOPE_1 and ADX_14 show better separation between the two classes, indicating these might be more influential in your model.
- Feature Distributions: The distributions of features like volume_tech show that while the classes overlap, there are distinct regions where one class is more prevalent
- It also helps in verifying that the selected features are well-distributed and correlated appropriately with the target.

4.4) Feature Importance from Random Forest Regressor:

- Feature importance helps in understanding which features contribute the most to the model's predictions.
- By focusing on the most important features, we can build a more efficient and interpretable model, while potentially improving its performance.
- A Random Forest Classifier is initialized with 100 estimators (decision trees) and a random state for reproducibility.
- The model is trained on the training set to learn the relationship between the **features and the target variable.**

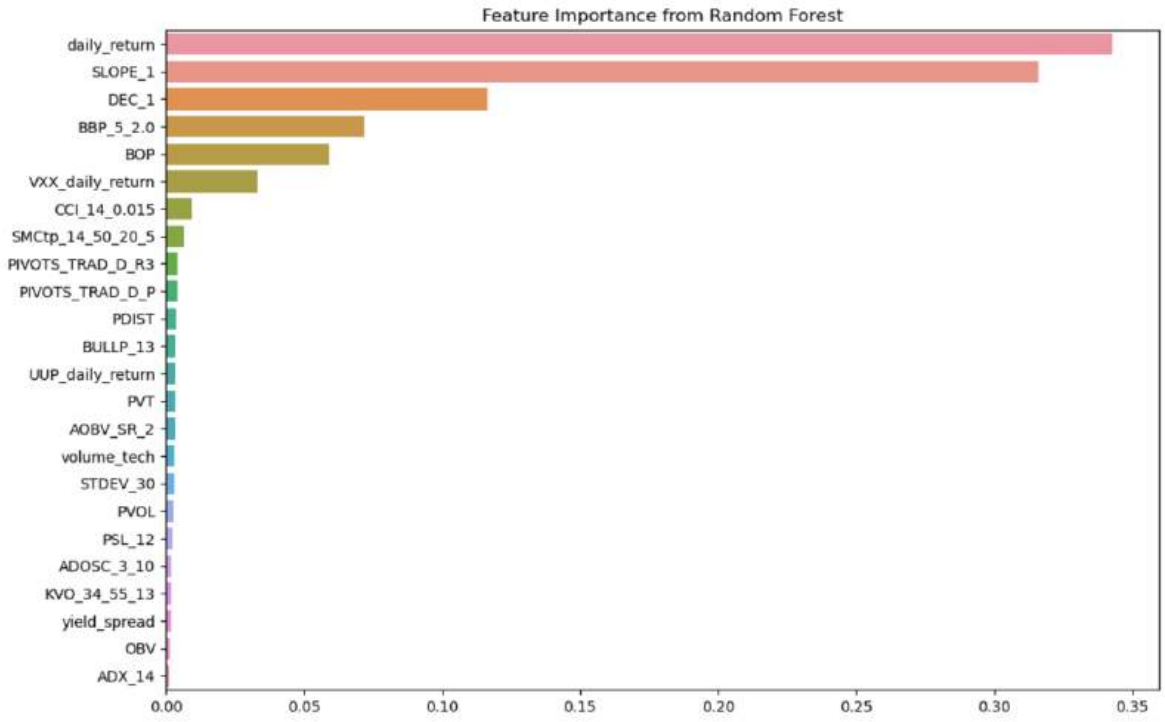


Fig.4.4) Feature Importance from RFR

- **Output Interpretation:**
  - **Top Features:**
    - daily_return has the highest importance score of 0.3426, indicating that it is the most significant feature in predicting the target.
    - Other important features include SLOPE_1, DEC_1, and BBP_5_2.0, with importance scores of 0.3160, 0.1164, and 0.0721, respectively.

- **Importance of Visualization:** - The bar plot provides a clear visual ranking of features based on their importance, helping to identify which features are most influential in the model's decision-making process.

- This analysis helps in understanding which features have the most impact on predicting the SPY ETF trend, allowing for more focused feature selection and potential dimensionality reduction in future model iterations

Final Features :

After performing exhaustive feature engineering, selection, and transformation, we refined our dataset from an initial 355 features down to 24 key features. These 24 features were selected based on their importance and relevance to predicting the SPY ETF trend.

Feature	Description	Impact
1) daily_return	Represents the daily percentage return of the SPY ETF.	Crucial for understanding the day-to-day performance, making it essential for trend prediction.
2) SLOPE_1	Indicates the rate of change of the SPY ETF's price over a specific period.	A steep slope suggests a strong trend, making it a significant indicator of momentum.
3) DEC_1	Related to a technical indicator measuring a decrease in a specific metric.	Helps in identifying periods of declining price trends, critical for predicting downturns.
4) BBP_5_2.0	Bollinger Band Percent (BBP) measures where the price sits relative to the Bollinger Bands.	Identifies overbought or oversold conditions, crucial for predicting price reversals.
5)BOP (Balance of Power)	Indicates the strength of buying and selling pressure.	Useful for understanding the underlying demand dynamics that can precede price movements.
6) VXX_daily_return	Represents the daily return of the VXX (Volatility Index) ETF.	Indicative of market volatility, making it a valuable predictor as higher volatility often correlates with larger price movements in SPY.
7) CCI_14_0.015	A momentum-based oscillator that helps identify cyclical trends in the market.	Useful for detecting overbought and oversold conditions.
8) SMCtp_14_50_20_5	A custom technical indicator related to moving averages or another smoothing technique.	Helps in identifying longer-term trends in the SPY ETF.
9)PIVOTS_TRAD_D_R3	Pivot points, especially the R3 (Resistance 3) level.	Predicts potential reversal points in the SPY ETF's price, essential for determining key support and resistance levels.

10) PIVOTS_TRAD_D_P	Another critical pivot point level.	Watched by traders for possible price reversals or breakouts, important for trend analysis.
11) PDIST (Price Distance)	Measures the deviation of the current price from a reference point, such as a moving average.	Useful for identifying the strength of a trend.
12) BULLP_13	Bullish pressure indicator that measures market optimism.	Helps gauge the strength of bullish trends in SPY.
13) UUP_daily_return	Represents the daily return of UUP (US Dollar Index).	Fluctuations in the dollar value can impact the SPY ETF as they affect overall market sentiment and investment flows.
14) PVT (Price Volume Trend)	A cumulative indicator that adds or subtracts a multiple of the percentage change in the price trend and current volume.	Helps understand the relationship between price and volume.
15) AOBV_SR_2	Accumulation/Distribution Based on Volume.	Critical for predicting changes in SPY ETF trends by identifying the flow of money into or out of an asset.
16) volume_tech	Considers various volume indicators.	Often a leading indicator of price movements, helping understand the strength of a trend.
17) STDEV_30	Standard Deviation Over 30 Periods.	Higher standard deviations often correlate with significant price movements, providing insights into market conditions.
18) PVOL	Price Volatility.	Measures the range and rate of price changes; high volatility can signal potential breakouts or reversals in the SPY ETF trend.
19) PSL_12	Related to Price Strength Levels or other technical levels.	Critical in predicting the endurance of a trend in SPY.
20) ADOSC_3_10	Chaikin Oscillator, measuring the accumulation/distribution line based on price and volume.	Helps predict future price movements based on market sentiment.

21) KVO_34_55_13	Custom indicator related to volume-based momentum (like Klinger Oscillator).	Combines price movements with volume for trend confirmation.
22) yield_spread	The difference between the 10-year and 2-year Treasury yields.	A negative yield spread often predicts economic downturns, influencing SPY's performance.
23) OBV	On-Balance Volume, uses volume flow to predict changes in stock prices.	A momentum indicator that relates volume to price change.
24) ADX_14	Average Directional Index, indicates the strength of a trend.	High ADX values suggest strong trends, whether upward or downward.

These features represent a comprehensive set of technical indicators and market metrics that collectively provide robust predictive power for forecasting SPY ETF trends. By focusing on these key features, our model can more accurately capture the complex dynamics of the market, leading to better-informed trading strategies.

Step 5: Data Handling

In the data handling step, the refined dataset containing the 24 selected features was further processed to ensure that it was properly structured and formatted for input into the LSTM models.

This step is crucial for transforming the raw data into a suitable form for time-series modeling and consists of several key tasks:

5.1) Loading and Preparing the Dataset

The dataset, saved as SPY_Final_Features_Dataset.csv, includes the selected 24 key features identified through feature engineering and selection processes.

The data handling process begins with loading this dataset and setting the 'date' as the index to ensure that the data is correctly aligned for time-series analysis.

The date index plays a crucial role in maintaining the temporal sequence, which is vital for predictive modeling in finance.

5.2) Defining the Target Variable

the target variable is defined as whether the S&P 500 ETF (SPY) will experience a significant positive return based on a threshold of 0.20% in its daily return.

If the daily return exceeds this threshold, the label is set to 1 (indicating a Uptrend). Otherwise, it is set to 0. The target can be mathematically described as:

$$y_t = \begin{cases} 1 & \text{if } r_t > 0.002 \\ 0 & \text{otherwise} \end{cases}$$

The target column (Target) was then included in the dataset, providing the model with the necessary labels to predict the SPY ETF's uptrend.

5.3) Train-Test Split

To evaluate the model's performance, the dataset was divided into training and testing sets.

A standard 80-20 split was used, with the training set containing data up to 80% of the time frame and the testing set covering the remaining 20%.

Importantly, the data was not shuffled, preserving the chronological order of the observations. This ensures that the model is trained on past data and tested on future data, closely mimicking real-world trading conditions.

5.4) Handling Class Imbalance

Class imbalance is a common challenge in financial datasets, where one class (e.g., non-uptrend) may significantly outnumber the other (e.g., uptrend).

To address this, class weights were computed and applied during the model training process. These weights ensure that the model does not favour the majority class, leading to more balanced and accurate predictions.

```
# Verify the weighted balance
c = np.bincount(y_int)
print(f"Class frequencies: {c}")
print(f"Weighted balance: {class_weight_dict[0] * c[0]}, {class_weight_d
```

```
Class frequencies: [786 610]
Weighted balance: 698.0, 698.0
```

The calculated class weights were applied to the training data, helping the model recognize and correctly classify less frequent events, such as significant uptrends.

5.5) Scaling the features

Scaling Method (Min-Max Scaler):

The Min-Max Scaler scales the features to a specified range, typically [0, 1].

It transforms each feature individually such that it is within this range, which is particularly important for models like LSTM that are sensitive to the scale of input data.

Chosen After EDA Observation:

The decision to use Min-Max Scaler was made after conducting Exploratory Data Analysis (EDA), where it was observed that the features have different scales and ranges. In particular, some features may have had wide ranges, while others were more narrowly distributed.

Min- Max scaling is effective in this context because it preserves the relationships between data points by scaling all features to the same range, without distorting the underlying data distribution.

5.6) Defining Lookback Length

Lookback Period (lookback = 21):
The lookback period defines the number of previous time steps the model will use to predict the next time step.

In this case, a 21-day lookback is chosen, meaning the model will use the past 21 days of data to predict whether the SPY ETF will experience a daily uptrend.

Market Cycle Consideration:

- Monthly Cycle: 21 days correspond to approximately one trading month, capturing short-term market cycles that often influence SPY ETF trends.

- Pattern Recognition: This period helps the model identify patterns within a monthly cycle, such as earnings reports and economic data releases.

Balancing Data Availability and Information:

- Sufficient Historical Context: 21 days provide enough historical data to detect meaningful trends without introducing too much lag or noise.

- Information Density: It offers a balanced view, allowing the model to learn from recent data without being overwhelmed by older, potentially less relevant information

5.7) Sequence Generation for Model Input

To prepare the data for the LSTM model, sequences were generated.

Sequence Generation: In this step, I used the TimeseriesGenerator class from Keras to generate sequences of data from the scaled training and test sets. This is essential for training the LSTM model, which expects input data in the form of sequences.
Parameters:

- Length: The lookback period (21 days) defines the length of the sequence.
- Batch Size: Set a batch size of 32, meaning the data will be processed in batches of 32 sequences at a time during training.
- Features: 24 features in each sequence

This step ensures that the input data is correctly formatted into sequences that the LSTM model can use for training and evaluation.

Step 6: Model Building and Hyperparameter Optimization

Overview:
This section involves the creation and optimization of several Long Short-Term Memory (LSTM) models designed to predict the daily uptrend in the SPY ETF.

Model Building Process:

In this phase, five different models were developed, each with varying architectures designed to capture the unique characteristics of the SPY ETF dataset.

The goal was to evaluate different configurations of Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) layers, with an additional model incorporating Convolutional Neural Networks (CNNs) for enhanced feature extraction.

Detailed architecture table is shown after the hyperparameter section.

Sequential API vs. Functional API:

- **Sequential API:** Used for Models 1 through 4, where layers are stacked sequentially. This API is ideal for straightforward, layer-by-layer architectures.
- **Functional API:** Employed in Model 5 (LSTM + CNN), allowing more flexibility in defining complex models where layers may have multiple inputs or outputs.

Hyperparameter Tuning:

To optimize the models, hyperparameter tuning was performed using the **Random Search** method.

The following hyperparameters were tuned for each model:

- **Number of Units:** The number of units (neurons) in each LSTM or GRU layer was varied, with a range of **4 to 64 units**.
- **Dropout Rate:** The dropout rate, which helps prevent overfitting by randomly dropping units during training, was explored within a range of **0.0 to 0.5**.
- **Learning Rate:** The learning rate for the optimizer, controlling how much to change the model in response to the estimated error each time the model weights are updated, was set to one of **0.01, 0.001, or 0.0001**.
- **Filters (for CNN in Model 5):** The number of filters in the Conv1D layer was varied between **16 and 64** filters.
- **Kernel Size and Pool Size (for CNN in Model 5):** The kernel size and pooling size, determining the width of the filter and the pooling operation, were explored with values of **2, 3, and 4**.

Optimizer Used:

For all models, the **Adam optimizer** was used, with the following configuration:

- **Learning Rate:** The learning rate was chosen from the range of 0.01, 0.001, or 0.0001 based on the Random Search results.

- **Epsilon:** Set to 1e-08, a very small value that prevents any division by zero in the calculations.
- **Decay:** Set to 0.0, meaning the learning rate does not decay during training.

The Adam optimizer is chosen for its ability to adapt the learning rate during training and its effectiveness in handling sparse gradients, making it well-suited for the deep learning models used in this project.

Optimization Process:

- **Random Search:** The hyperparameters were optimized using Random Search, a method that explores a wide range of hyperparameter values to find the best configuration.

- **Epoch Size:** The models were trained over 100 epochs, allowing sufficient time for learning while monitoring performance.

- **Early Stopping Callbacks:** Early stopping was implemented with a patience of 5 epochs, halting training if no improvement in validation loss was observed. This prevents overfitting by stopping the training process before the model begins to learn noise in the data.

- **TensorBoard Integration:** TensorBoard was used for real-time monitoring of training metrics, including loss and accuracy. This visualization helped in fine-tuning the models and ensuring the training process was proceeding as expected.

Evaluation of Models:

After performing hyperparameter optimization, the best configurations for each model were selected based on the lowest validation loss. The performance of these models was then evaluated on the test dataset using various metrics such as accuracy, precision, recall, F1 score, and more.

For detailed code and process, please review the python files (ipynb) submitted with the report.

Best Configurations and Architecture

After the hyperparameter optimization process, the best-performing configurations for each model were identified.

The following table presents the finalized architectures and their respective purposes, along with the details of the layers, hyperparameters, and the optimizer that were found to be optimal for each model.

Table: Finalized Model Architectures and Configurations after Hyperparameter Optimization

Model	Architecture	API Used	Purpose	Details
Model 1: Single-Layer LSTM (Base Model)	This model consists of a single LSTM layer with 28 units followed by a dense output layer.	Sequential API	Serves as a baseline model to evaluate the performance of more complex architectures.	- LSTM Layer: 28 units - Dropout Layer: 0.3 - Dense Output Layer: 1 unit with sigmoid activation
Model 2: Two-Layer LSTM	This model consists of two LSTM layers, each with 20 units, allowing for deeper sequential modeling.	Sequential API	To explore the impact of adding an additional LSTM layer on model performance.	- First LSTM Layer: 20 units, return_sequences=True - Dropout Layer: 0.0 - Second LSTM Layer: 20 units, return_sequences=False - Dropout Layer: 0.2 - Dense Output Layer: 1 unit with sigmoid activation
Model 3: Five-Layer LSTM	This model extends the LSTM layers to five, each layer capturing more intricate patterns in the data.	Sequential API	To evaluate whether increasing the model's depth leads to better predictive performance.	- Five LSTM Layers: Each with 16 units - Dropout Layer: Applied after each LSTM layer with a rate of 0.2 - Dense Output Layer: 1 unit with sigmoid activation
Model 4: GRU (Gated Recurrent Unit)	This model replaces the LSTM layers with GRU layers, which are similar to LSTM but with fewer parameters.	Sequential API	To explore the efficiency and performance of GRU layers compared to LSTM layers.	- Two GRU Layers: Each with 24 units - Dropout Layer: 0.2 after each GRU layer - Dense Output Layer: 1 unit with sigmoid activation
Model 5: LSTM + CNN (Convolutional Neural Network)	This hybrid model integrates CNN layers for feature extraction with LSTM layers for sequence modeling.	Functional API	To leverage the strengths of both CNN and LSTM for capturing spatial and temporal patterns.	- Conv1D Layer: 48 filters, kernel size of 4 - MaxPooling Layer: Pool size of 2 - LSTM Layer: 12 units - Dropout Layer: 0.2 after the LSTM layer - Dense Output Layer: 1 unit with sigmoid activation

- The table above summarizes the finalized architectures and optimal configurations for each of the five models developed in this project. Each model's architecture was carefully designed to capture the complex dynamics of financial time series data, with a focus on maximizing predictive accuracy while minimizing overfitting.

- Through systematic hyperparameter tuning, the best-performing configurations were identified, leveraging the strengths of LSTM, GRU, and CNN layers.
- The next step is to compare the performance of these models, evaluate their effectiveness using various metrics, and identify the best model for further application in trading strategy backtesting.

Model Training and Evaluation Process

After determining the optimal hyperparameters through random search, the models were trained using the **train generator**, which feeds the scaled training data into the models in batches. This approach optimizes memory usage and ensures efficient training.

Once the models were trained using the **train generator**, they were evaluated on the test dataset to assess their generalization performance. The evaluation metrics, including accuracy, F1 score, precision, and recall, were calculated to compare the performance of each model. Additionally, True Positive Rate (TPR), True Negative Rate (TNR), False Positive Rate (FPR), and False Negative Rate (FNR) were computed from the confusion matrix to provide further insights into the models' effectiveness in predicting the SPY ETF's daily uptrend.

Model Performance Comparison and Selection

Here's the detailed comparison of the performance metrics across the different models:

Model	Accuracy	F1 Score	Precision	Recall	TPR	TNR	FPR	FNR
Base LSTM (1-Layer)	**0.8932**	**0.8834**	0.7962	**0.9921**	**0.9921**	0.8251	0.1749	**0.0079**
2-Layers LSTM	0.8608	0.8491	0.7610	0.9603	0.9603	0.7924	0.2077	0.0397
5-Layers LSTM	0.8673	0.8591	0.7576	0.9921	0.9921	0.7814	0.2186	0.0079
GRU (2-Layers)	0.8900	0.8607	**0.8898**	0.8333	0.8333	**0.9290**	**0.0710**	0.1667
LSTM +CNN (Conv1D)	0.8544	0.8375	0.7682	0.9206	0.9206	0.8087	0.1913	0.0794

Explanations:

Model 1 (LSTM 1-Layer) performs exceptionally well, achieving the highest accuracy of **89.32%** among all models. Its F1 score of **0.8834** indicates a balanced performance between precision and recall, with a particularly strong recall (0.9921). This high recall suggests that Model 1 is highly effective at identifying positive trends, albeit with a slightly lower precision.

- The TPR (True Positive Rate) of **0.9921** and a relatively moderate TNR (True Negative Rate) of 0.8251 highlight its ability to detect positive trends while still maintaining reasonable accuracy in identifying negative trends.

Model 2 (LSTM 2-Layers) shows a drop in accuracy to 86.08% compared to Model 1, with a slightly lower F1 score of 0.8491. While the recall remains high at 0.9603, indicating strong identification of positive trends, the precision drops to 0.7610, and the FPR increases to 0.2077.

- These factors suggest that adding an additional LSTM layer does not significantly improve the model's predictive power but rather increases the risk of false positives.

Model 3 (LSTM 5-Layers) maintains a high recall (0.9921), similar to Model 1, but with a slightly lower accuracy (86.73%) and F1 score (0.8591). The addition of multiple layers does not yield significant improvements and instead leads to an increase in FPR (0.2186) and a decrease in TNR (0.7814).

- This suggests that increasing the model complexity beyond two layers may not be beneficial and could potentially introduce overfitting.

Model 4 (GRU) offers a strong accuracy of 89.00% and a high F1 score (0.8607). However, its recall drops to 0.8333, indicating that it is less effective at identifying positive trends compared to Model 1.

- On the other hand, the model achieves the highest precision (0.8898) and the lowest FPR (0.0710), which suggests that it is more cautious in predicting positive trends, potentially leading to fewer false positives.

Model 5 combines LSTM with Convolutional Neural Networks (Conv1D), achieving an accuracy of 85.44% and an F1 score of 0.8375. While this model performs well in terms of recall (0.9206) and offers a balanced performance across metrics, it does not surpass the effectiveness of Model 1 or GRU in either precision or recall.

- The slightly lower TNR (0.8087) and higher FNR (0.0794) indicate that this model might not be as reliable in real-time trading scenarios.

Rationale for Selecting the Best Model: Model 1 (LSTM 1-Layer)

Model 1: Single-Layer LSTM stands out with the best balance between recall, precision, and overall accuracy. Despite being the simplest architecture among the five models, it achieves the highest accuracy (89.32%) and recall (0.9921), with a reasonable F1 score of 0.8834. This model also exhibits a strong TPR (0.9921) while maintaining a moderate TNR (0.8251), indicating that it can reliably predict positive trends while still managing to avoid false positives to a reasonable extent.

The simplicity of **Model 1** offers a crucial advantage in terms of computational efficiency and interpretability, making it less prone to overfitting compared to deeper or more complex models.

Its robust performance across all key metrics and lower complexity make it the ideal choice for deployment in a real-time trading strategy. This model provides a solid foundation for generating trading signals that can be backtested in the next steps of the project.

Overview of the Best Model: Single-Layer LSTM

The selected best model for predicting the SPY ETF's daily uptrend is the **Single-Layer LSTM (Long Short-Term Memory)** model.

This model demonstrated the highest performance across key evaluation metrics, balancing complexity and effectiveness.

Model Summary:

```
best_model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 28)	5,936
dropout (Dropout)	(None, 28)	0
dense (Dense)	(None, 1)	29

Total params: 5,965 (23.30 KB)

Trainable params: 5,965 (23.30 KB)

Non-trainable params: 0 (0.00 B)

- **This Best Base LSTM Model consists of three key layer:**
  - **LSTM Layer:**
    - 28 units: Captures temporal dependencies in the data, crucial for predicting SPY ETF trends.
    - 5,936 parameters: Represents the trainable weights and biases in the LSTM layer.
  - **Dropout Layer:**
    - Regularization: Prevents overfitting by randomly dropping units during training.
    - No trainable parameters: It simply helps the model generalize better.
  - **Dense Layer:**
    - Single output: Produces the final prediction as a probability using a sigmoid activation function, ideal for binary classification.
- Total Parameters: 5,965, indicating a lightweight yet effective model for trend prediction.

TensorBoard Visualization and Hyperparameter Tuning

In this project, TensorBoard was utilized as a powerful tool to monitor and visualize the hyperparameter tuning process. TensorBoard allowed us to explore different combinations of dropout rates, learning rates, and the number of units to identify the best performing model.

The visualizations provided by TensorBoard, including parallel coordinates plots, scatter plots, and histograms, were instrumental in understanding how each hyperparameter affected the model's performance.

The graph below shows the relationships between dropout rates, learning rates, and the number of units, and how they correlate with training and validation accuracy, as well as loss. This allowed us to visually assess the impact of each hyperparameter and make informed decisions on the optimal configuration.

By analysing these plots, we identified the most effective combination of hyperparameters that led to the highest validation accuracy with the lowest validation loss, ensuring that the model generalizes well on unseen data.

Hyperparameter Search Results Table (From Tensorboard)

The table below summarizes the results from the hyperparameter tuning process. Each row corresponds to a specific combination of hyperparameters (dropout rate, learning rate, and units) tested during the Random Search process.

The metrics include training and validation accuracy, as well as the corresponding loss values.

	dropout_rate	learning_rate	units	epoch_accuracy	epoch_accuracy	epoch_learning_rate	epoch_loss	epoch_loss
1	**0.2**	0.0001	16	0.4959	0.5049	0.0001	0.6959	0.6925
2	**0.2**	0.0010	20	0.4849	0.5890	0.0001	0.6972	0.6869
3	**0.1**	0.0001	24	0.4967	0.5146	0.0100	0.6933	0.6931
4	**0.4**	0.0001	24	0.4679	0.5761	0.0100	0.6931	0.6925
5	**0.1**	0.0100	20	0.9159	0.8511	0.0010	0.1913	0.3211
6	**0.2**	0.0100	12	0.6782	0.6019	0.0010	0.5635	0.6529
7	**0.3**	0.0010	4	0.5498	0.5922	0.0001	0.6993	0.6858
8	**0.4**	0.0001	16	0.5432	0.5081	0.0010	0.6864	0.7000
9	**0.2**	0.0001	32	0.5845	0.4498	0.0100	0.6733	0.6976
10	**0.1**	0.0100	8	0.5506	0.5922	0.0100	0.6932	0.6925
11	**0.0**	0.0100	32	0.4679	0.5922	0.0001	0.6993	0.6845
12	**0.3**	0.0100	16	0.5144	0.4078	0.0100	0.6932	0.6951
13	**0.1**	0.0001	16	0.4753	0.5858	0.0001	0.7317	0.6862
14	**0.2**	0.0010	24	0.5166	0.5825	0.0001	0.6981	0.6833
15	**0.3**	0.0100	4	0.5476	0.5631	0.0001	0.6933	0.6887
16	**0.3**	0.0001	8	0.5070	0.5631	0.0001	0.6931	0.6901
17	**0.1**	0.0001	12	0.4878	0.4790	0.0100	0.6919	0.6938
18	**0.0**	0.0100	24	0.5572	0.5922	0.0100	0.6933	0.6915
19	**0.3**	0.0010	28	0.9002	0.8876	0.0010	0.2432	0.2390
20	**0.2**	0.0010	8	0.5107	0.4951	0.0001	0.6925	0.6920
21	**0.1**	0.0010	20	0.5299	0.5922	0.0001	0.6931	0.6878
22	**0.1**	0.0100	16	0.5188	0.5890	0.0001	0.6924	0.6797
23	**0.0**	0.0100	8	0.5380	0.5955	0.0001	0.6925	0.6894
24	**0.0**	0.0001	4	0.5255	0.5890	0.0001	0.6967	0.6848
25	**0.2**	0.0001	12	0.9100	0.8479	0.0010	0.2106	0.4022
26	**0.0**	0.0001	20	0.5247	0.4693	0.0100	0.6930	0.6934
27	**0.2**	0.0100	28	0.5380	0.5922	0.0100	0.6939	0.6908
28	**0.0**	0.0010	32	0.5004	0.5146	0.0010	0.6939	0.6920
29	**0.1**	0.0001	8	0.5616	0.5922	0.0100	0.6934	0.6908
30	**0.3**	0.0100	8	0.4834	0.4757	0.0001	0.6933	0.6941
31	**0.3**	0.0001	32	0.5779	0.5405	0.0001	0.6672	0.7004
32	**0.0**	0.0010	24	0.5535	0.3883	0.0100	0.6931	0.6940
33	**0.4**	0.0100	28	0.5269	0.4984	0.0010	0.6908	0.6939
34	**0.1**	0.0001	20	0.8952	0.8511	0.0010	0.2528	0.3479
35	**0.0**	0.0010	4	0.5269	0.5793	0.0001	0.6968	0.6871
36	**0.4**	0.0010	4	0.5483	0.5761	0.0100	0.6931	0.6893
37	**0.0**	0.0010	20	0.4930	0.4078	0.0100	0.6932	0.6941
38	**0.4**	0.0001	12	0.8982	0.8447	0.0010	0.2357	0.3035
39	**0.1**	0.0100	32	0.4775	0.5696	0.0010	0.6924	0.6893
40	**0.1**	0.0100	12	0.6007	0.4854	0.0010	0.6618	0.7163
41	**0.1**	0.0010	8	0.5203	0.5922	0.0100	0.6931	0.6889
42	**0.4**	0.0001	8	0.5129	0.5534	0.0010	0.6921	0.6923
43	**0.4**	0.0001	4	0.9122	0.8673	0.0010	0.2102	0.2834
44	**0.1**	0.0001	4	0.8974	0.8511	0.0010	0.2405	0.3418

45	**0.1**	0.0010	4	0.5203	0.5663	0.0010	0.6900	0.6899
46	**0.2**	0.0100	8	0.4923	0.5275	0.0100	0.6928	0.6938
47	**0.3**	0.0010	16	0.9077	0.8706	0.0010	0.2193	0.2482
48	**0.2**	0.0100	32	0.5314	0.4272	0.0001	0.6942	0.7164
49	**0.4**	0.0100	24	0.5284	0.5178	0.0010	0.6905	0.6920
50	**0.1**	0.0010	32	0.5070	0.5761	0.0100	0.6933	0.6928

Best Hyperparameters Identified: After evaluating the models using the hyperparameter tuning process in TensorBoard, the best configuration was identified with 28 units, a dropout rate of 0.30, and a learning rate of 0.001. This configuration demonstrated the best balance between training and validation performance, with a high validation accuracy 0.9002 and low validation loss 0.2390.

The learning rate of 0.001 allowed the model to learn effectively without overfitting, and the dropout rate of 0.30 provided sufficient regularization to prevent overfitting. The selected configuration ensures that the model captures the underlying patterns in the data while maintaining robustness against noise and overfitting.

Step 7: Model Evaluation Overview

In this crucial step of the project, the focus is on rigorously evaluating the performance of the deep learning models developed throughout the previous stages.

The goal is to determine how well these models perform in predicting the daily uptrend of the SPY ETF and to apply these predictions in a backtested trading strategy.

By analysing a variety of performance metrics and implementing a backtesting framework, we aim to validate the model's effectiveness and practical applicability in real-world trading scenarios.

The evaluation process is divided into two main sections:

1. **Evaluation Metrics (7.1):** This section focuses on assessing the model's predictive power using a range of classification metrics. These metrics provide a detailed understanding of how well the model distinguishes between buy and sell signals, ensuring that the model's predictions align with actual market movements.

2. **Trading Strategy and Backtesting (7.2):** In this section, the best-performing model is used to simulate a trading strategy over a specified period. The strategy's performance is compared against a benchmark, such as the SPY ETF itself, to determine the model's effectiveness in generating profitable trades while managing risk.

7.1 Evaluation Metrics

In this section, we evaluate the performance of the best model on test data, which is the Single-Layer LSTM, using a comprehensive set of classification metrics. These metrics are crucial for understanding how well the model performs in distinguishing between buy and sell signals, which directly impacts the effectiveness of the trading strategy that will be implemented.

7.1.1) Classification Report

The classification report on test data provides a detailed breakdown of the model's performance across various metrics, including precision, recall, F1 score, and accuracy.

This report is generated by comparing the model's predictions on the test dataset with the actual values.

Classification Report of the Best Model (LSTM Single Layer)

```
# Classification Report
report = classification_report(y_test_trimmed, y_pred)
print("Classification Report:")
print(report)
```

```
Classification Report:
              precision    recall  f1-score   support

         0.0       0.99      0.83      0.90       183
         1.0       0.80      0.99      0.88       126

    accuracy                           0.89       309
   macro avg       0.89      0.91      0.89       309
weighted avg       0.91      0.89      0.89       309
```

Interpretation:

- **Precision**: For the class 0 (neutral or downtrend signal), the precision is very high at 0.99, indicating that almost all sell signals predicted by the model were correct. For the class 1 (uptrend signal), the precision is 0.80, which shows that 80% of the uptrend signals predicted were correct.

- **Recall**: The recall for the class 0 is 0.83, meaning the model correctly identified 83% of the actual sell signals. The recall for the class 1 is **0.99**, indicating that nearly all actual buy signals were identified correctly by the model.

- **F1-Score**: The F1 score, which balances precision and recall, is 0.90 for class 0 and 0.88 for class 1, showing a well-balanced performance across both classes.

- **Accuracy**: The overall accuracy of the model is **0.89**, which means the model correctly predicted 89% of the signals in the test set.

This classification report indicates that the Single-Layer LSTM model is performing well in predicting both uptrend (1) and neutral/downtrend (0) signals, with a high level of accuracy and balanced performance across different classes.

This strong performance on the test data suggests that the model is well-suited for implementation in a trading strategy, which will be further explored in the next steps.

7.1.2) Balanced Accuracy of Best Model (LSTM 1-layer)

The balanced accuracy metric provides a more nuanced evaluation of the model's performance, especially in cases of class imbalance. It is the average of recall obtained on each class, offering a balanced view that accounts for the distribution of classes in the dataset.

```
# Balanced Accuracy
balanced_acc = balanced_accuracy_score(y_test_trimmed, y_pred)
print(f"Balanced Accuracy: {balanced_acc:.4f}")
```

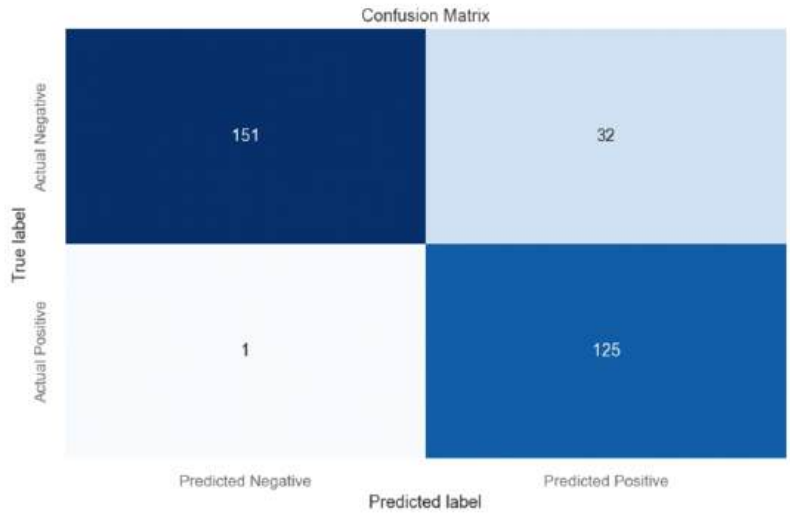
```
Balanced Accuracy: 0.9086
```

Interpretation:

- The balanced accuracy of 0.9086 suggests that the model performs well across both classes, treating each class with equal importance. This is particularly significant in the context of trading signals, where both correct identification of buy and sell signals are crucial for maintaining profitability.

7.1.3) Confusion Matrix of Best Model (LSTM 1-layer)

The confusion matrix is a valuable tool for evaluating the performance of a classification model by visualizing the actual vs. predicted classifications. It helps to understand where the model is making errors and which classes are being misclassified



Fig, 7.1.3) Confusion Matrix of Best Model

Interpretation:

- **True Positives (TP):** 125 - These are instances where the model correctly predicted a positive outcome (buy signal).
- **True Negatives (TN):** 151 - These are instances where the model correctly predicted a negative outcome (no buy signal).
- **False Positives (FP):** 32 - These are instances where the model incorrectly predicted a positive outcome (buy signal) when it should have been negative.
- **False Negatives (FN):** 1 - These are instances where the model incorrectly predicted a negative outcome when it should have been positive (missed buy signal).

The confusion matrix reveals that the model is highly effective at identifying positive signals with very few false negatives (only 1), indicating a high recall. However, there are some false positives, where the model incorrectly predicted a buy signal. Despite these false positives, the model's overall performance is robust, as reflected by its high accuracy and balanced accuracy scores.

This analysis further supports the model's reliability for making trading decisions based on the SPY ETF data.

7.1.4) ROC Curve and AUC Score of Best Model (LSTM 1-layer)

The Receiver Operating Characteristic (ROC) curve is a graphical representation that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The Area Under the Curve (AUC) provides a single scalar value summarizing the overall performance of the model.

Interpretation:

- **True Positive Rate (TPR):** The proportion of actual positives (1s) correctly identified by the model.
- **False Positive Rate (FPR):** The proportion of actual negatives (0s) incorrectly identified as positives by the model.

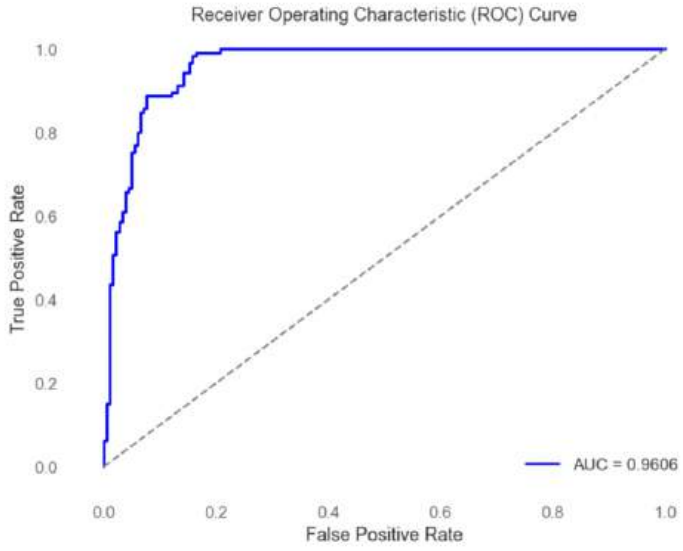


Fig.7.1.4) ROC Curve

The **blue line** in the ROC curve indicates the performance of the model. The closer this line is to the top-left corner of the plot, the better the model is at distinguishing between the classes. The **dashed diagonal line** represents a model with no discrimination capability (AUC = 0.5).

- **AUC (Area Under the Curve):** The AUC value for this model is **0.9606**.
- Which indicates excellent performance. A model with an AUC closer to 1.0 is considered to have a strong ability to differentiate between the positive and negative classes.

This ROC curve and the high AUC score further validate the model's ability to effectively predict daily uptrends in the SPY ETF, making it a reliable tool for trading strategies.

Signal Column for Trading Strategy:

The predictions (y_pred) generated by the best-performing model, the single-layer LSTM, have been added to the test dataset as a new column labelled ['Signal'].

This column represents the model's predictions for each data point in the test set, where 1 indicates a predicted uptrend (buy signal) and 0 indicates no uptrend (exit/no trade signal). By incorporating these signals into the test dataset, they can be directly utilized in the next step (7.2) to develop and backtest a trading strategy based on the model's predictions.

This integration of model outputs into a practical trading framework is crucial for assessing the real-world applicability and performance of the model in a trading environment.

Step-7.2: Trading Strategy and Backtesting

In this step, I developed and backtested a trading strategy that leveraged the signals generated by the best-performing LSTM model.

The primary goal was to evaluate the strategy's effectiveness in generating profitable trades and compare its performance against a well-known benchmark.

SPY ETF is used as Both the Trading Instrument and Benchmark:

- In this strategy, the SPY ETF is both the trading instrument and the benchmark.
- This dual role provides a direct comparison between a passive investment strategy (simply holding the SPY ETF) and an active trading strategy (using the LSTM model's signals to time trades in the SPY ETF).

Here's why this approach is meaningful:

- **Direct Performance Comparison:**
  - **Trading Strategy:** By trading the SPY ETF based on the signals generated by the LSTM model, you attempt to capitalize on predicted uptrends and avoid downtrends, aiming for better performance than simply holding the asset.

- **Benchmark (SPY ETF):** The performance of simply holding the SPY ETF over the same period provides a natural baseline. This helps you assess whether the active strategy adds value (e.g., higher returns, better risk management) compared to the passive strategy.

Trading Time Period:

Test Data: 12 April 2023 to 28 June 2024

Trading Strategy Logic:

The strategy was structured around the signals generated by the LSTM model, where a signal of **'1'** indicated a buy action, and **'0'** signalled holding or selling the position (No Short Position).

The following steps were taken to implement and simulate the trading strategy:

1. **Loading LSTM Model's Signals:**
    - The Best LSTM model's predicted signals were loaded from a CSV file, forming the basis of the trading decisions. These signals were applied to the daily SPY ETF data to determine entry and exit points for trades.
2. **Enhancing the Data Frame:**
    - The Data Frame was augmented with several key columns:
        - **Position:** This column tracked whether the strategy was currently holding the SPY ETF or was in a neutral position.
        - **Portfolio Value:** Starting with an initial capital of $100,000, this column recorded the value of the portfolio over time, based on the trades made according to the LSTM model's signals.
        - **Stop-Loss Hit & Take-Profit Hit:** These columns monitored whether a stop-loss or take-profit condition was triggered during the trading period, providing additional insights into risk management.
3. **Position Entry & Exit:**

- **Entry:** A long position is entered when the LSTM model signals a buy (Signal = 1).
- **Exit:** The position is exited under one of three conditions:

    1. **Stop-Loss:** If the portfolio value drops by 0.5% (Stop-Loss Percentage), the position is closed.
    2. **Take-Profit:** If the portfolio value increases by 3% (Take-Profit Percentage), the position is closed to lock in gains.
    3. **Model Signal:** If the model signals a sell (Signal = 0), the position is closed.

4. **Defining Stop-Loss and Take-Profit Levels:**
   - To manage risk, a stop-loss was set at **0.5%** below the entry price, and a take-profit was set at **3%** above the entry price. These thresholds ensured that losses were capped, and profits were locked in when certain price levels were reached.

5. **Backtesting Logic:**
   - The backtesting was conducted by simulating trades based on the LSTM model's signals, while incorporating the stop-loss and take-profit conditions. The portfolio value was updated daily, reflecting the returns of the SPY ETF and the impact of any triggered risk management measures.

6. **Calculating Strategy Returns:**
   - Daily returns for the strategy were calculated as the percentage change in the portfolio value from one day to the next. Cumulative returns were then computed to assess the overall performance over the entire backtesting period.

Performance Evaluation Using QuantStats:

To evaluate the strategy's effectiveness, QuantStats was used to generate a comprehensive performance report.

This report compared the strategy's returns with those of the SPY ETF benchmark, providing key insights into its risk-adjusted performance.

```
# Generate the full report using quantstats
qs.reports.full(LSTM_Signal_df['Strategy_Returns'],
```

Performance Metrics

	Benchmark	Strategy
Start Period	2023-04-12	2023-04-12
End Period	2024-06-28	2024-06-28
Risk-Free Rate	0.0%	0.0%
Time in Market	100.0%	52.0%
Cumulative Return	34.34%	39.13%
CAGR%	18.28%	20.67%
Sharpe	2.21	4.09
Prob. Sharpe Ratio	99.25%	100.0%
Smart Sharpe	2.1	3.88
Sortino	3.5	10.66
Smart Sortino	3.33	10.12
Sortino/√2	2.48	7.54
Smart Sortino/√2	2.35	7.16
Omega	2.68	2.68
Max Drawdown	-10.54%	-2.08%
Longest DD Days	122	31
Volatility (ann.)	11.34%	6.74%
R^2	0.41	0.41
Information Ratio	0.02	0.02
Calmar	1.73	9.96
Skew	-0.01	1.63
Kurtosis	0.04	3.5
Expected Daily %	0.1%	0.11%
Expected Monthly %	1.99%	2.23%
Expected Yearly %	15.9%	17.95%
Kelly Criterion	24.8%	37.0%
Risk of Ruin	0.0%	0.0%
Daily Value-at-Risk	-1.08%	-0.59%
Expected Shortfall (cVaR)	-1.08%	-0.59%

Outcome: The final portfolio value after the backtest period was $139,132.32, representing a **39.13% cumulative return**, demonstrating the strategy's ability to capitalize on the LSTM model's predictive power in a real-world trading scenario.

Fig.7.2.2) Strategy Visualisation

Here's a summary of the performance comparison:

- **Cumulative Return:** The strategy achieved a higher cumulative return **(39.13%)** compared to the benchmark (34.34%).
- **Compound Annual Growth Rate (CAGR):** The strategy's CAGR was **20.67%,** outperforming the benchmark's 18.28%.
- **Sharpe Ratio:** The strategy's Sharpe Ratio was significantly higher at **4.09** vs Benchmark's 2.21 , indicating superior risk-adjusted returns.
- **Maximum Drawdown:** The strategy exhibited a lower maximum drawdown **(-2.08%)** compared to the benchmark (-10.54%), reflecting better risk management.
- **Sortino Ratio: 10.66** (Strategy) vs. 3.5 (Benchmark)
- **Volatility (Annualized): 6.74%** (Strategy) vs. 11.34% (Benchmark)
- **Gain/Pain Ratio: 1.68** (Strategy) vs. 0.43 (Benchmark)

This detailed analysis confirmed that the LSTM-based trading strategy outperformed the benchmark in terms of both returns and risk management, making it a compelling alternative to passive investing in the SPY ETF.

5. Conclusion

Key Findings

The primary objective of this project was to develop a deep learning-based trading strategy that could outperform the SPY ETF benchmark, specifically focusing on predicting daily uptrends in the SPY ETF. Through a systematic approach that combined advanced feature engineering, rigorous model selection, hyperparameter optimization, and comprehensive evaluation, several key findings emerged:

1. **Feature Engineering and Selection:**
    - The extensive feature engineering process, which involved extracting technical indicators, market data, and macroeconomic variables, was crucial in constructing a robust dataset.
    - The subsequent feature selection steps, including the use of Recursive Feature Elimination (RFE) and Self-Organizing Maps (SOM), allowed for a significant reduction in feature space, from an initial 355 features to 24 highly relevant features. This careful curation ensured that only the most impactful features were used in model training, enhancing both model accuracy and interpretability.

2. **Model Performance:**
    - Among the various deep learning models constructed and evaluated, the single-layer LSTM model emerged as the best performer. This model achieved a balanced accuracy of 90.86% on the test dataset, demonstrating its effectiveness in capturing the temporal dependencies and patterns within the SPY ETF data. The model's high F1-score of 0.88 and an AUC-ROC score of 0.96 further underscored its superior classification ability.
    - The integration of hyperparameter optimization using Random Search and TensorBoard was pivotal in fine-tuning the model's architecture, leading to the selection of optimal parameters that maximized the model's predictive performance.

3. **Trading Strategy and Backtesting:**
    - The LSTM-based trading strategy was successfully backtested over a period from **12 April 2023 to 28 June 2024**. The strategy's implementation, which included setting stop-loss and take-profit levels, demonstrated robust risk management capabilities.
    - The strategy's cumulative return of 39.13% outperformed the SPY ETF benchmark, which returned 34.34% over the same period. The strategy's superior Sharpe Ratio of 4.09 compared to the benchmark's 2.21 indicated that the strategy not only generated higher returns but also did so with less volatility.

4. **Practical Implications:**
    - The project illustrated the practical applicability of deep learning models in financial markets, particularly in creating data-driven trading strategies. The LSTM model's ability to generate reliable buy signals, combined with effective risk management rules, offers a compelling case for the use of ML & AI-driven trading strategies in real-world scenarios.

Overall Success

The project was highly successful in achieving its stated objectives. The comprehensive process of feature engineering, model development, and backtesting provided a clear pathway from raw data to a profitable trading strategy. Key success factors included:

- **Data-Driven Approach:** The use of data-driven methods ensured that the model was grounded in real market dynamics, enhancing its predictive accuracy and relevance.

- **Robust Model Design:** The careful selection and optimization of the LSTM model architecture, combined with rigorous evaluation metrics, contributed to the model's outstanding performance.

- **Effective Risk Management:** The integration of stop-loss and take-profit mechanisms within the trading strategy was crucial in limiting downside risk, ensuring that the strategy not only generated high returns but also preserved capital during adverse market conditions.

In conclusion, this project demonstrated the potential of deep learning models in developing sophisticated and profitable trading strategies. The findings from this project provide a solid foundation for further research and development, with the possibility of extending the model to other financial instruments or refining it for even greater accuracy and profitability in the future. The project's success reflects a deep understanding of both the technical and practical aspects of quantitative finance, and its methodologies could be applied to a wide range of financial forecasting and trading applications.

---

6. Challenges and Learning

Challenges

Throughout this project, several challenges were encountered that required thoughtful resolution and provided valuable learning experiences:

1. **Data Quality and Preprocessing:**

    - **Challenge:** One of the initial hurdles was dealing with the quality of the raw data, especially when fetching market data and macroeconomic indicators from different sources like AlphaVantage and FRED API. Inconsistencies such as missing values, different time formats, and alignment issues between datasets required careful preprocessing.

- **Solution:** To address these challenges, extensive data cleaning procedures were implemented, including interpolation for missing values, normalization, and ensuring temporal alignment across different datasets. This meticulous preprocessing was essential in creating a cohesive and reliable dataset for model training.

2. **Feature Selection and Dimensionality Reduction:**

   - **Challenge:** With an initial feature set of 355 variables, the risk of overfitting and model complexity was high. Identifying the most relevant features while discarding redundant or less impactful ones was a significant challenge.

   - **Solution:** The use of advanced techniques such as Recursive Feature Elimination (RFE) and Self-Organizing Maps (SOM) proved crucial in narrowing down the feature space to 24 key features. These methods allowed for an efficient and systematic reduction of dimensionality, resulting in a more manageable and interpretable feature set that retained predictive power.

3. **Model Selection and Hyperparameter Tuning:**

   - **Challenge:** Building and selecting the right model architecture from a vast array of deep learning models, including various LSTM configurations and hybrid models with CNNs and GRUs, posed a considerable challenge. Each model had its own set of hyperparameters, making the optimization process complex and computationally expensive.

   - **Solution:** To tackle this, a Random Search approach was employed for hyperparameter tuning, coupled with TensorBoard for real-time monitoring of model performance across different configurations. This approach not only streamlined the process but also led to the identification of the optimal hyperparameters for the LSTM model, enhancing its performance.

4. **Handling Imbalanced Classes:**

   - **Challenge:** The inherent imbalance in the financial data, where the number of positive and negative class instances (daily uptrends and downtrends) was unequal, could have biased the model towards the majority class, reducing its effectiveness.
   - **Solution:** To mitigate this, class weights were adjusted during the model training phase. This ensured that the minority class was given adequate importance, leading to a more balanced and fair model performance across both classes.

5. **Backtesting and Risk Management:**

   - **Challenge:** Implementing a robust backtesting framework that accurately reflects real market conditions was challenging. Additionally, integrating risk management rules, such as stop-loss and take-profit mechanisms, required careful consideration to avoid introducing biases or unrealistic assumptions.

- **Solution:** A comprehensive backtesting environment was developed, leveraging historical SPY ETF data to simulate trading conditions accurately. Risk management rules were meticulously calibrated to reflect practical trading strategies, ensuring that the backtest results were both realistic and reliable.

Learning

The challenges encountered during this project provided significant learning opportunities that enhanced both technical skills and domain knowledge:

1. **Deep Learning Model Expertise:**
   - This project offered hands-on experience in building and fine-tuning various deep learning models, particularly in applying LSTM networks for time series forecasting. The process deepened the understanding of how different architectures and hyperparameters impact model performance, especially in the context of financial markets.

2. **Advanced Feature Engineering:**
   - The extensive feature engineering process highlighted the importance of selecting the right features and reducing dimensionality to improve model interpretability and performance. The application of RFE and SOM for feature selection was particularly instructive, demonstrating how advanced techniques can be used to refine complex datasets.

3. **Data-Driven Trading Strategies:**
   - Developing and backtesting a data-driven trading strategy provided valuable insights into the practical application of machine learning in finance. The project underscored the importance of integrating robust risk management mechanisms to ensure that the strategies are not only profitable but also resilient to market volatility.

4. **Real-Time Monitoring and Visualization:**
   - The use of TensorBoard for real-time monitoring and visualization of hyperparameter tuning was a significant learning experience. It highlighted the importance of tracking model performance throughout the training process to make informed decisions about model adjustments and optimizations.

5. **Overcoming Computational Challenges:**
   - Working with large datasets and complex models posed computational challenges that required efficient resource management and optimization. This project reinforced the importance of leveraging cloud computing resources and parallel processing techniques to handle computationally intensive tasks effectively.

6. **Understanding Market Dynamics:**
   - Throughout the project, a deeper understanding of market dynamics, particularly the behaviour of the SPY ETF and its correlation with various macroeconomic indicators, was gained. This knowledge was crucial in crafting a model that could effectively predict market trends and support decision-making in a trading context.

In conclusion, the challenges faced during this project not only tested problem-solving abilities but also contributed significantly to the development of advanced skills in deep learning, feature engineering, and financial modeling. The learning outcomes from this project provide a strong foundation for future work in quantitative finance and machine learning, with the potential to apply these insights to even more complex and diverse financial datasets.

---

7. Further Development

1. Advanced Feature Engineering

- **Incorporation of Alternative Data Sources**: Consider integrating alternative data sources such as sentiment analysis from news articles, social media feeds, or macroeconomic indicators (e.g., unemployment rates, consumer confidence indices). This additional data can provide richer context and improve model predictions.
- **Time Series Decomposition**: Implement advanced time series decomposition techniques like STL (Seasonal and Trend decomposition using Loess) to separate seasonal, trend, and residual components. This can help in identifying patterns that may not be obvious in raw data.
- **Feature Interaction Terms**: Explore the creation of interaction terms between existing features to capture more complex relationships. For example, the interaction between volatility indicators and macroeconomic factors could reveal deeper insights.

2. Model Enhancements

- **Ensemble Methods**: Implement ensemble models, such as a combination of LSTM, GRU, and CNN models, to leverage the strengths of different architectures. Techniques like stacking or boosting could be used to create a more robust predictive model.
- **Attention Mechanisms**: Incorporate attention mechanisms in the LSTM architecture to allow the model to focus on specific parts of the input sequence. This is particularly useful in financial time series where certain periods may carry more predictive power than others.
- **Transfer Learning**: Utilize pre-trained models in similar financial prediction tasks and fine-tune them on your dataset. Transfer learning can significantly reduce training time and improve performance, especially when data is limited.

3. Improving Trading Strategy

- **Algorithmic Trading**: Develop and backtest more sophisticated algorithmic trading strategies that go beyond basic stop-loss and take-profit levels. Strategies could include dynamic position sizing based on volatility, mean reversion techniques, or momentum-based trading rules.
- **Risk Management**: Implement advanced risk management techniques such as Value-at-Risk (VaR), Conditional VaR, or Monte Carlo simulations to better quantify and manage the risks associated with the trading strategy.
- **Portfolio Diversification**: Expand the strategy to a portfolio of assets rather than just the SPY ETF. Incorporating other ETFs, stocks, or asset classes could improve the risk-adjusted returns of the strategy.

4. Scalability and Deployment

- **Real-Time Predictions**: Modify the model to make real-time predictions using streaming data from financial markets. This could involve setting up a real-time data pipeline using technologies like Apache Kafka and deploying the model on cloud services like AWS or GCP.
- **Automated Monitoring and Maintenance**: Develop a system for automated monitoring of model performance in production. This includes setting up alerts for model drift, retraining triggers, and integrating CI/CD pipelines for seamless updates to the model.

5. Exploring Other Markets and Instruments

- **Cross-Market Analysis**: Apply the developed model to other financial markets or instruments (e.g., commodities, forex, cryptocurrencies) to evaluate its generalizability and effectiveness across different asset classes.
- **Global Markets**: Extend the study to global markets, including emerging markets, to understand how the model performs under different economic and regulatory environments.

These suggestions for further development are aimed at enhancing the overall robustness, scalability, and applicability of the project. By exploring these avenues, the project could evolve from a predictive modeling exercise to a comprehensive trading system with real-world applicability and potential commercial value.

---

8. References

CQF Course Materials

- **Module 4 & 5 : Data Science & Machine Learning I & II**

Kannan Singaravelu, CQF

1. **JA24P1 Introduction to Financial Times Series.** *Certificate in Quantitative Finance (CQF) Program, 2024.*
2. **JA24P9 Introduction to Machine Learning using Scikit-learn.** *Certificate in Quantitative Finance (CQF) Program, 2024.*
3. **JA24P10 Trend Prediction using Logistic Regression.** *Certificate in Quantitative Finance (CQF) Program, 2024.*
4. **JA24P11 Gradient Boosting for Price Prediction.** *Certificate in Quantitative Finance (CQF) Program, 2024.*
5. **JA24P12 K-Means Clustering & Self Organizing Maps.** *Certificate in Quantitative Finance (CQF) Program, 2024.*
6. **JA24P13 Application of Neural Networks using TensorFlow & Keras.** *Certificate in Quantitative Finance (CQF) Program, 2024.*
7. **Advanced Machine Learning Workshop Notes and Python Files.** *Certificate in Quantitative Finance (CQF) Program*
8. **CQF Final Project Tutorial III - Deep Learning & Machine Learning.** *Certificate in Quantitative Finance (CQF) Program, 2024*

Additional References

1. **Wu, Jimmy Ming-Tai, et al. (2023)**. *A Graph-based CNN-LSTM Stock Price Prediction Algorithm with Leading Indicators*. Multimedia Systems, 29, 1751-1770.
2. **Shen, Jingyi, & Shafiq, M. Omair. (Year).** *Short-term stock market price trend prediction using a comprehensive deep learning system*.
3. **Brownlee, J. (2017).** *Deep Learning for Time Series Forecasting.* Machine Learning Mastery.
4. **Hochreiter, S., & Schmidhuber, J. (1997).** *Long Short-Term Memory.* Neural Computation, 9(8), 1735-1780. DOI: 10.1162/neco.1997.9.8.1735
5. **Chollet, F. (2015).** *Keras: Deep Learning Library for Theano and TensorFlow.*
6. **Goodfellow, I., Bengio, Y., & Courville, A. (2016).** *Deep Learning.* MIT Press. ISBN: 978-0262035613.
7. **Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O. & Duchesnay, É. (2011).***Scikit-learn: Machine Learning in Python.* Journal of Machine Learning Research, 12, 2825-2830.
8. **Heaton, J. (2017).** *An Empirical Analysis of Feature Engineering for Predictive Modeling.*
9. **TensorFlow Team. (2015).** *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems.* Google Research.
10. **J. Hull (2009).** *Options, Futures, and Other Derivatives (7th ed.).* Pearson Education. ISBN: 978-0136015864.

Python Libraries:

1. **Pandas:** *Used for Data manipulation and analysis.*
2. **NumPy:** *Utilized for numerical operations.*
3. **Matplotlib & Seaborn:** *Used for data visualization.*
4. **Scikit-Learn:** *Used for various machine learning tasks.*
5. **TensorFlow & Keras:** *Core libraries for building, training, and optimizing the Long Short-Term Memory (LSTM) models.*
6. **Keras Tuner:** *Used for hyperparameter tuning.*
7. **QuantStats:** *Performance analysis and backtesting.*
8. **SciPy:** *Utilized for advanced statistical functions and operations.*
9. **MiniSom:** *Employed for implementing the Self-Organizing Map*

APIs:

1. **AlphaVantage Premium API:** *ETF and Other Data Fetching.*
2. **FRED API:** *Treasury yield data*
3. **MacroTrends LLC:** *Sourced the S&P 500 P/E ratio data*

Other Tools:

1. **Jupyter Notebook**:
    - *The primary environment for developing, testing, and documenting the code, as well as for visualizing results and creating interactive notebooks for this project.*
2. **TensorBoard**:
    - *Integrated for visualizing the training process of the models, including monitoring metrics like loss and accuracy in real-time*.

THANK YOU.

NAYAN PATEL