Integrating Multi-Frequency Trading Signals for Optimal Market Timing: A Machine Learning Approach

1 Introduction

In the rapidly evolving domain of financial markets, the ability to accurately predict stock movements significantly enhances trading strategies and financial outcomes. This project is focused on leveraging advanced machine learning techniques to tackle a specific classification problem: predicting whether the closing price of a stock will be higher than its opening price on the following day. We plan to utilize sophisticated algorithms such as Long Short-Term Memory (LSTM) networks and XGBoost to address this binary classification challenge, integrating the predictive power of sequential data processing and ensemble learning.

Accurately classifying stock movements has profound implications for portfolio management and trading strategies. Traditional quantitative models, often linear in nature, have struggled to capture the complexities and nonlinearities inherent in financial markets [4]. The adoption of machine learning in financial prediction has introduced new dynamics to modeling strategies, offering improved prediction accuracy and better handling of large datasets.

Machine learning models, particularly those involving deep learning and ensemble methods, have been shown to effectively handle the complexities of financial time series data. LSTM networks are well-suited for this task due to their ability to capture long-term dependencies in time series data, an essential feature for understanding stock price movements [**?**]. On the other hand, XGBoost has gained popularity in financial applications for its efficiency and performance in classification tasks, providing robust predictive capabilities with an emphasis on scalability and speed [2].

Recent studies have emphasized the integration of machine learning with financial analytics, showing that models like LSTM and XGBoost can outperform traditional methods in predicting market trends and movements [3]. These models not only accommodate the volume and variety of financial data but also adapt to its evolving nature, allowing for more dynamic and responsive financial decision-making.

Integrating LSTM and XGBoost for this classification task embodies the cutting-edge of current financial machine learning research. By applying these models, we aim to harness their respective strengths in handling time-series data and structured, tabular data, thus providing a holistic approach to predicting stock price movements. This project not only contributes to the academic and practical understanding of financial prediction models but also explores the potential of machine learning to redefine market analytics.

2 Problem Setup

$$y_{t+1} = \begin{cases} 1 & \text{if } p_{t+1}^{close} > p_{t+1}^{open}, \\ 0 & \text{if } p_{t+1}^{close} \leq p_{t+1}^{open}. \end{cases}$$

Where:

- p_{t+1}^{close} and p_{t+1}^{open} represent the closing and opening prices of the stock on day $t+1$ respectively.

Assumptions

1. **Market Efficiency**: The semi-strong form of the Efficient Market Hypothesis (EMH) is assumed, where all publicly available information is presumed to be reflected in the stock prices at any given time.
2. **Data Independence**: It is assumed that the features derived from the market data used for model training are sufficiently independent, minimizing multicollinearity and enhancing model robustness.
3. **Static Model Parameters**: Over the short prediction horizon, the model parameters are considered static, suggesting the relationships captured by the model remain constant.

Definitions

- **Feature Set** (X_t): Includes a variety of technical indicators calculated from historical price data, such as moving averages, relative strength index (RSI), volume, and potentially others, which serve as input features for the model.
- **Model**: The function f mapping the input features X_t to the predicted category y_{t+1}. This will be implemented using classification algorithms such as Logistic Regression, Support Vector Machines (SVM), or advanced neural networks like LSTM or GRU adapted for classification tasks.
- **Performance Metrics**: To evaluate the model's efficacy, metrics such as accuracy, precision, recall, F1-score, and the area under the ROC curve (AUC) will be utilized.

3 Dataset

3.1 Raw data

This research utilizes a comprehensive dataset sourced from Bloomberg and Barchart, covering an array of stocks and their respective options. The data spans from October 10, 2023, to April 26, 2024, encompassing a detailed look at market dynamics through high-frequency and daily financial metrics.

Equity Price Data

The equity price dataset obtained from Bloomberg comprises high-frequency intraday data for several key technology and semiconductor stocks, including Nvidia (NVDA), Advanced Micro Devices (AMD), ARM Holdings, Super Micro Computer (SMCI), Sonos (SONO), the Semiconductor ETF (SOXX), the NASDAQ 100 Index Tracking Stock (QQQ), the SP 500 ETF (SPY), Tesla (TSLA), and Taiwan Semiconductor Manufacturing Company (TSM). The dataset features 8-minute frequency recordings of the following variables for each stock:

- **Open Price:** The opening price at the beginning of each 8-minute interval.
- **High Price:** The highest price reached during the interval.
- **Low Price:** The lowest price during the interval.
- **Close Price:** The closing price at the end of the interval.
- **Volume:** The total volume of stocks traded during the interval.

Options Data

Complementary to the equity price data, the options dataset from Barchart provides daily insights into the options market for the same set of stocks. This dataset includes the following features:

- **Implied Volatility:** A measure of the market's forecast of a likely movement in a security's price.
- **Put/Call Volume:** The total volume of put and call options traded.
- **Option Volume:** The aggregate volume of all options traded.
- **Put/Call Open Interest:** The total open contracts that have not been settled for puts and calls.
- **Total Open Interest:** The sum of all open contracts that have not been settled.

Data Sources

The data for this study was sourced from:

- Bloomberg: Provides high-frequency intraday equity price data[**?**].
- Barchart: Supplies daily options data including implied volatility and open interest metrics[1].

3.2 Data Preprocessing

Data preprocessing and feature engineering are critical steps in transforming raw data into a format suitable for advanced predictive models, particularly in financial market analysis. This section details the comprehensive preprocessing applied to datasets sourced from Bloomberg and Barchart, optimized for machine learning applications aimed at predicting market movements.

Preprocessing Steps

Handling Missing Values: Missing values were addressed through linear interpolation for stock data and forward filling for options metrics to maintain the continuity and integrity of the time series.

Outlier Detection and Removal: Outliers were identified using the Z-score method and removed from the dataset to ensure robust analysis, defining outliers as data points beyond three standard deviations from the mean.

Normalization and Scaling: All numeric data, including prices and volumes, were normalized using min-max scaling to ensure that all features have equal influence during the model training phase.

Feature Engineering

Enhancing the dataset's predictive power involved engineering several key features from both stock and options data:

- **Technical Indicators:** Calculated Moving Averages, Relative Strength Index (RSI), and Bollinger Bands to capture market trends and volatility.
- **Options Metrics:** Derived metrics such as the put-call ratio to gauge market sentiment.
- **Percentage Changes:** Computed the percentage change in implied volatility and stock prices to capture market dynamics.
- **Normalized Prices:** Prices were normalized against moving averages to reduce volatility effects and highlight longer-term trends.
- **Volume Weighted Features:** Created volume-weighted by normalized moving average to emphasize significant trading activities aligned with price changes.

Insights from Processed Data

The preprocessing and feature engineering significantly enriched the dataset, providing a robust foundation for predictive modeling:

- Enhanced data quality and consistency ensure that the input to machine learning models is reliable and representative of underlying market behaviors.
- Engineered features like normalized prices and volume-weighted indicators allow models to discern more complex patterns, crucial for capturing both short-term fluctuations and longer-term movements.
- Market sentiment and volatility indicators from options data provide valuable insights into investor behavior, enhancing the model's ability to predict market movements accurately.

The systematic approach to data preprocessing ensures that the resulting dataset is not only clean and consistent but also augmented with meaningful attributes that enhance the analytical capabilities of machine learning models. This structured preparation is essential for developing reliable and effective predictive models in the financial domain.

4 Methodology

Training Objective

The following five models will be used to predict whether a stock's closing price will be higher or lower than its opening price, based on factors generated from high-frequency data.

1. XGBoost

Architecture: The model takes dozens of factors as input features, builds a sequence of decision trees, and outputs a binary prediction indicating whether the closing price is higher or lower than the opening price. Each tree consists of nodes and branches, defined by decision splits. Nodes split the input data based on specific features and thresholds, leading to different branches. Each leaf node represents a prediction and is assigned a weight, contributing to the final output. The combined prediction from all trees is processed through a logistic function to output a probability indicating whether the closing price is higher or lower than the opening price.

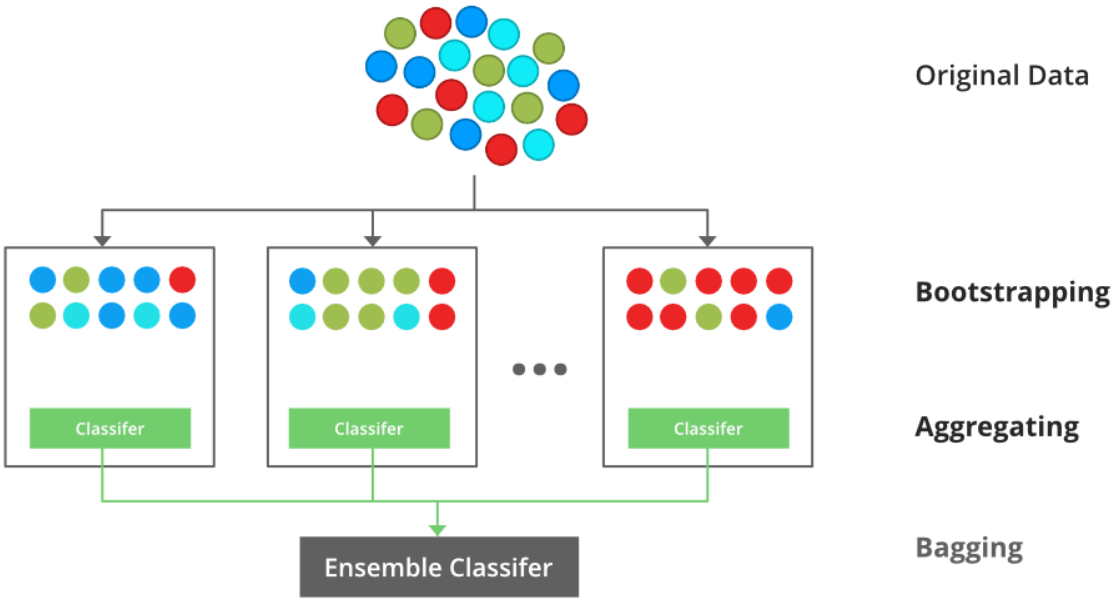

Figure 1: XGBoost

Parameterization:

- **Trainable Parameters:**
  - *Tree Structure:* Nodes and branches defined by decision splits.
  - *Leaf Weights:* Weights assigned to leaf nodes, representing the contribution to the final prediction.
- **Non-Trainable Parameters:**

– *Number of Trees:* Defined by the user or determined through early stopping.

Hyperparameters:

- *Learning Rate:* Controls how much each new tree contributes to the ensemble.
- *Max Depth:* Maximum depth of each decision tree.
- *Subsample:* Proportion of training data used for each tree.
- *Colsample_bytree:* Fraction of features used for each tree.
- *N_estimators:* Total number of trees to train.
- *Objective Function:* Binary logistic regression loss function.

2. Random Forest

Architecture: The model builds decision trees from the generated factors and combines their predictions by majority vote to produce a binary classification. Each tree consists of nodes and branches, defined by decision splits based on specific features and thresholds. Each leaf node represents a prediction (binary class indicating higher or lower closing price). Predictions from all trees are combined by majority vote for classification tasks. The final prediction is a binary classification indicating whether the closing price is higher or lower than the opening price.

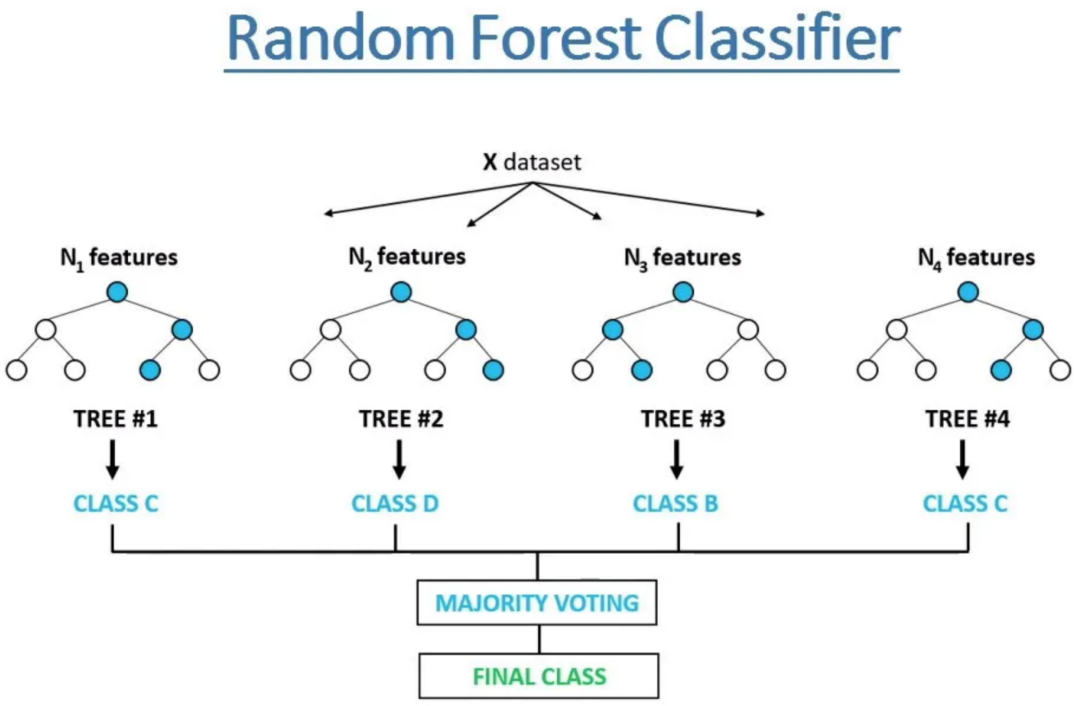

Figure 2: Random Forest

Parameterization:

- **Trainable Parameters:**
  - *Tree Structure:* Nodes and branches defined by decision splits.
  - *Leaf Weights:* Define the class predicted by each leaf node.
- **Non-Trainable Parameters:**
  - *Number of Trees:* Defined by the user.

Hyperparameters:

- *N_estimators:* Total number of trees to train.
- *Max Depth:* Maximum depth for each tree.
- *Min Samples Split:* Minimum number of samples to split an internal node.
- *Min Samples Leaf:* Minimum number of samples at a leaf node.
- *Bootstrap:* Whether to use bootstrap sampling.

3. LSTM

Architecture: The model takes time series data derived from 8-minute high-frequency data, including prices, volume, etc., and converts them into vectors. These vectors pass through a series of LSTM layers, which retain information over time and process each time step through input, forget, and output gates. The LSTM layers maintain a hidden state and cell state, enabling them to capture temporal dependencies. After the LSTM layers, the output is fed into a dense (fully connected) layer. The final output layer provides a binary classification, indicating whether the closing price is higher or lower than the opening price.

Parameterization:

- **Trainable Parameters:**
  - *Weights and Biases:* For the input, forget, and output gates, along with cell state transformation.
- **Non-Trainable Parameters:**
  - *Sequence Length:* Defines the number of time steps processed at once.
  - *Input Dimension:* Defines the input vector's dimension.

Hyperparameters:

- *Hidden Size:* Number of units in each LSTM cell.
- *Number of Layers:* Defines the number of stacked LSTM layers.
- *Dropout:* Regularization to set some outputs to zero.
- *Learning Rate:* Controls the step size for weight updates.

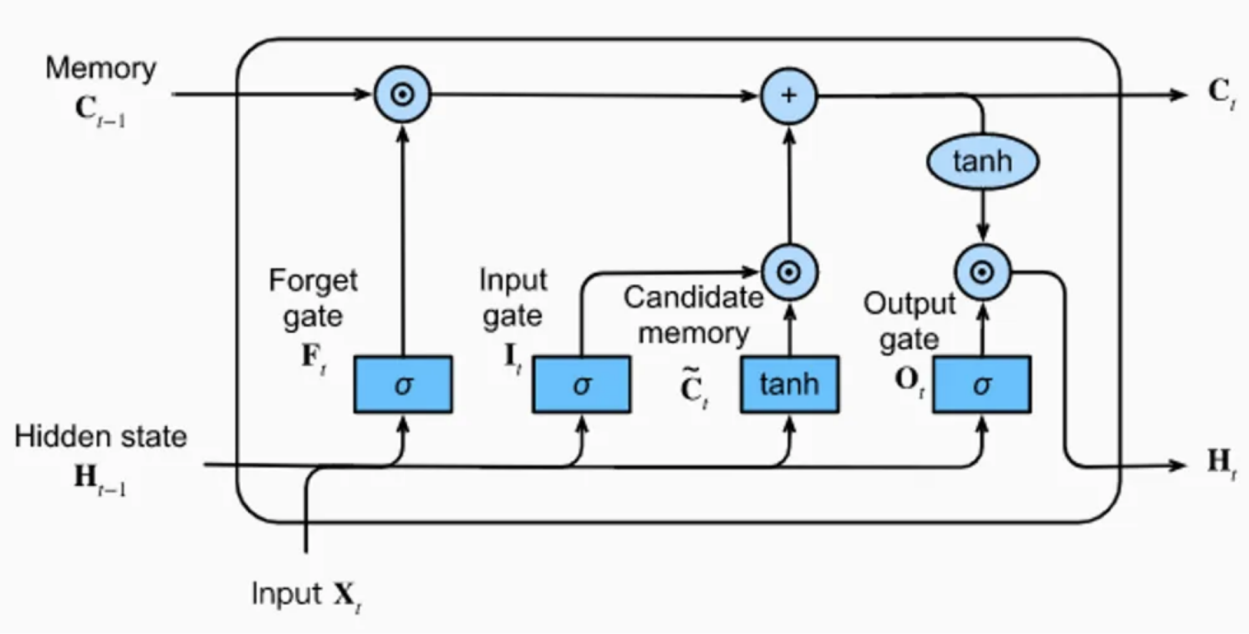

Figure 3: LSTM

4. FFN

Architecture: The model uses generated factors as input, processes them through hidden layers, and outputs a binary prediction. The input is passed through a series of hidden layers, with each layer transforming the input using a set of weights and biases. After each hidden layer, an activation function (e.g., ReLU, sigmoid, or tanh) introduces non-linearity. The final layer produces a binary classification, indicating whether the closing price is higher or lower than the opening price.

Parameterization:

- **Trainable Parameters:**
  - *Weights and Biases:* Define the transformation between layers.
- **Non-Trainable Parameters:**
  - *Input and Output Sizes:* Define input and output vector sizes.

Hyperparameters:

- *Number of Layers:* Defines the number of hidden layers.
- *Layer Sizes:* Number of units in each hidden layer.
- *Activation Functions:* Functions after each layer (ReLU, sigmoid, tanh).
- *Learning Rate:* Controls the step size for weight updates.

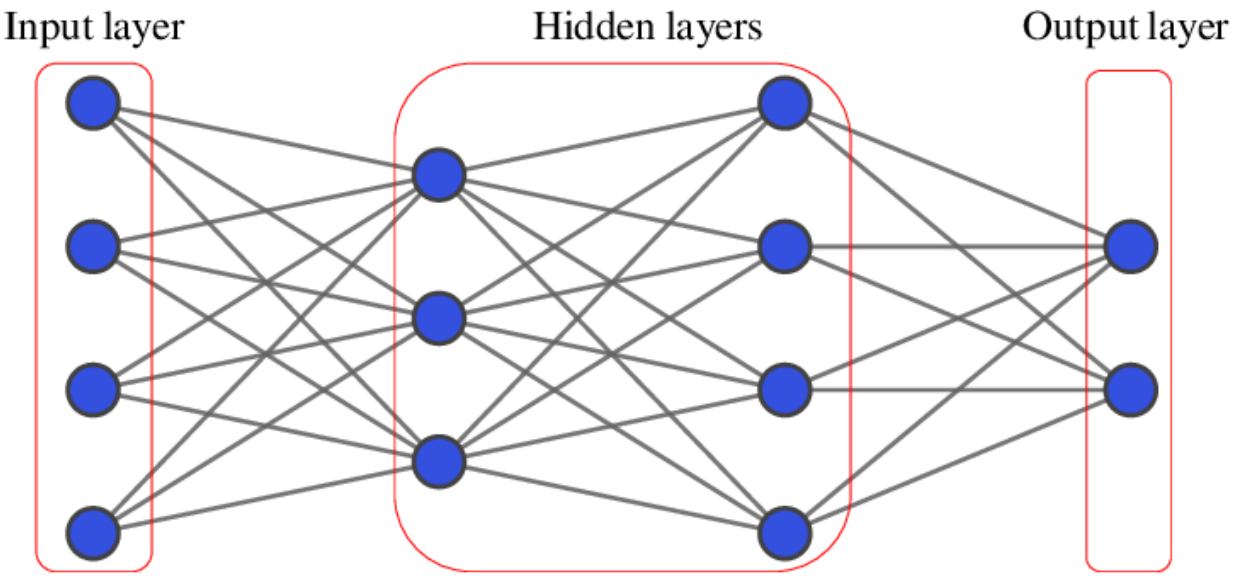

Figure 4: FFN

5. CNN

Architecture: The model processes patterns from 8-minute data through convolutional layers, producing a binary prediction. These patterns are processed through a series of convolutional layers, with each layer applying filters to detect different features. The output of convolutional layers is subsampled using pooling layers, reducing dimensionality. After convolutional and pooling layers, the output is passed to a dense layer. The final output layer provides a binary classification, indicating whether the closing price is higher or lower than the opening price.

Parameterization:

- **Trainable Parameters:**
  - *Convolutional Filters:* Weights of each filter detecting features.
  - *Fully Connected Layers:* Connects convolutional layers to output.
- **Non-Trainable Parameters:**
  - *Input Shape:* Defines data dimensions.
  - *Filter Sizes:* Defines filter sizes.
  - *Stride and Padding:* Control how filters move across the input.

Hyperparameters:

- *Number of Filters:* Number of convolutional filters per layer.
- *Kernel Size:* Defines each filter's size.
- *Pooling Layers:* Subsamples convolutional layer outputs.
- *Activation Functions:* Functions after each layer (ReLU, sigmoid).
- *Learning Rate:* Controls weight updates.

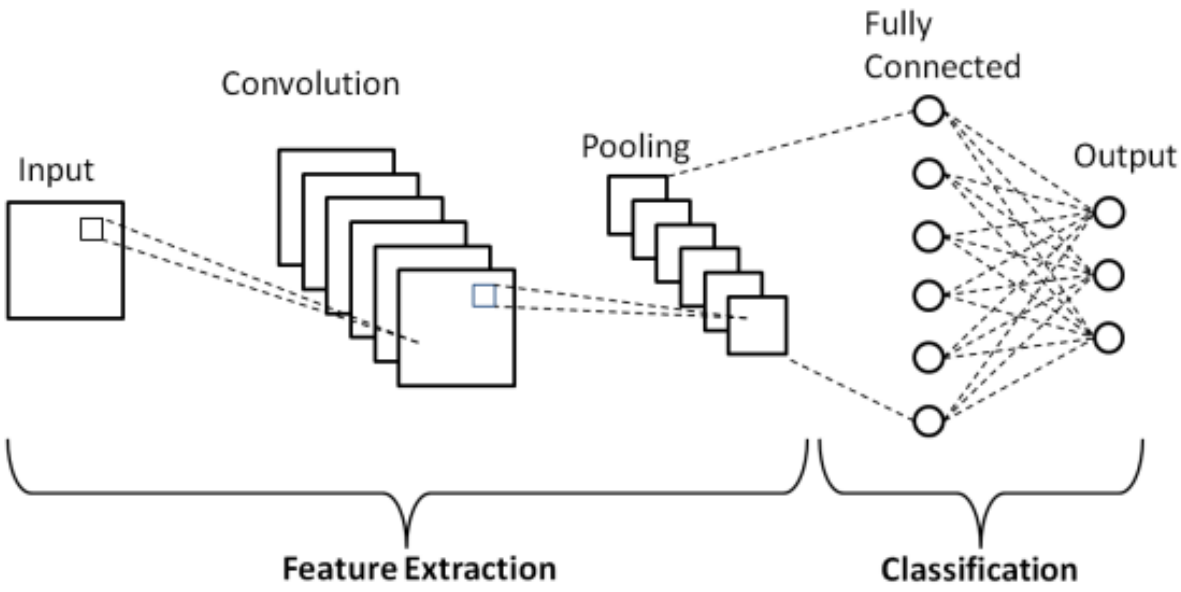

Figure 5: CNN

References

[1] Barchart. Daily Options Data. `https://www.barchart.com`, 2023-2024. [Accessed: 2024-04-30].

[2] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016.

[3] Marcos Lopez De Prado. *Advances in financial machine learning*. John Wiley & Sons, 2018.

[4] Shihao Gu, Bryan T Kelly, and Dacheng Xiu. Empirical asset pricing via machine learning. *Chicago Booth Research Paper*, (18-04):2018–09, 2019.