PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

Ministry of Higher Education and Scientific Research

UNIVERSITY ECHAHID HAMMA LAKHDAR - EL OUED FACULTY OF EXACT SCIENCES

Computer Science department

End of study memory Presented for the Diploma of

ACADEMIC MASTER

Field: **Mathematics and Computer Science**
Option: **Computer Science**
Specialty: **Distributed Systems and Artificial Intelligence**

Presented by :

- ATALLAH Amor
- SAIED Smail

Title

Machine learning for market volatility prediction

Defended on June 15th 2022

Examination Commitee :

M. Chourouk Guettas	**MCA**	**Chair**
M. Ben Barika Mohammed Kamel	**MCA**	**Examinator**
M. Naoui Mohammed Anouar	**MCA**	**Supervisor**

Academic year: 2021-2022

Thanks

We would like to thank everyone who has contributed to the success of our thesis and which helped us when writing this thesis.

First of all, we would like to thank, our supervisor **" Naoui Mohammed Anouar"**, for his patience, his availability and above all his sound advice.

We would also like to thank the entire teaching staff and teachers from the Computer Science department.

Abstract

Anticipating market volatility is one of the most common terms in today's trading market. Price movements, market volatility and trading risk are represented by realized volatility. A small change in volatility affects the expected return for all assets.

In this research to forecast volatility, we used the data set provided by Kaggle Optiver, a leading global electronic market maker committed to continuous improvement of financial markets and improving access and pricing of options, exchange-traded funds (ETFs), on several exchanges around the world, where the Cash Stocks, Bonds and Foreign Exchange. In our study we used the following forecast models: LightGBM, XGBoost, CatBoost and Linear Regression, and we concluded some related works on forecasting volatility. Then we ran our models. The results show that the LightGBM model is the best among these models, as it achieved the lowest root mean square error percentage (RMSPE) score of: 0.286, And the highest score in the coefficient of determination R2 is: 0.817, and the RMSPE for other models: XGBoost, CatBoost and Linear regression is, respectively: 0.303, 0.302 and 0.347, and the score of r2 is also: 0.791, 0.784, 0.766. Then we computed our models using feature geometry, the results improved, and the best improved result was in the LightGBM model and CatBoost model. Then we also applied K-Fold Cross-Validation to the four models, and noticed that the LightGBM result is the best result compared to the other models.

Through our study and after various experiments, we concluded that the lightGBM model is the best model compared to the XGBoost, CatBoost and Linear Regression models in predicting stock market fluctuations.

Keywords: Machine learning, prediction market volatility.

ملخص

يعد توقع تقلبات السوق أحد أكثر المصطلحات شيوعًا في سوق التداول اليوم. يتم تمثيل تحركات الأسعار وتقلبات السوق ومخاطر التداول من خلال التقلبات المحققة. و يؤثر تغيير طفيف في التقلب على العائد المتوقع لجميع الأصول.

في هذا البحث للتنبؤ بالتقلبات ، استخدمنا مجموعة البيانات التي توفرها منصة Kaggle Optiver ،وهو صانع سوق إلكتروني عالمي رائد وملتزم بالتحسين المستمر للأسواق المالية و تحسين الوصول وأسعار الخيارات ، والصناديق المتداولة في البورصة (ETFs) ، في العديد من البورصات حول العالم ، حيث يتم تداول الأسهم النقدية والسندات والعملات الأجنبية.في دراستنا استخدمنا نماذج التنبؤ التالية : LightGBM ، XGBoost ، CatBoost و Linear Regression، واستنتجنا بعض الأعمال ذات الصلة حول التنبؤ بالتقلب. وبعد ذلك قمنا بتشغيل نماذجنا . تظهر النتائج أن نموذج LightGBM هو الأفضل من بين هاته النماذج ،حيث حقق أدنى درجة جذر متوسط مربع النسبة المئوية لفقدان الخطأ (RMSPE) وهي : 0.286 ، وأعلى درجة في معامل التحديد R2 هي : 0.817 ، وأن RMSPE للنماذج الأخرى : XGBoost، CatBoost و Linear Regression هي على التوالي : 0.303 ، 0.302 ، 0.347 ، ودرجة r2 لها هي على التوالي ايضا : 0.791 ، 0.784 ، 0.766.ثم قمنا بحساب نماذجنا باستخدام هندسة الميزات ، وتحسنت النتائج ، ولكن كانت أفضل نتيجة محسّنة في نموذج LightGBM ونموذج CatBoost .ثم طبقنا أيضًا K-Fold Cross-Validation على النماذج الأربعة ، فلاحظنا أن نتيجة lightGBM هي أفضل نتيجة مقارنة بالنماذج الأخرى.

من خلال الدراسة التي قمنا بها وبعد التجارب المختلفة استخلصنا أن نموذج lightGBM هو النموذج الأفضل مقارنة بالنماذج XGBoost، CatBoost و Linear Regression في توقع تقلبات سوق الأسهم.

TABLES OF CONTENTS

List of Figures

List of Tables

General Introduction

Market Volatility is a very central and important topic in the financial literature and is of paramount importance in its many applications. It is an essential component of many investment decisions and is often the starting point for optimal portfolio allocations.

In 1952, Harry Markowitz laid the foundation for modern portfolio theory in which investors avoid risk and have an interest function that increases with expected return and decreases with volatility. Volatility is also a major input to the pricing of many derivatives. In fact, it is necessary to know the volatility of the underlying asset until the expiration date, to assess the fair value of the options. However, in recent years even derivatives with the same volatility have been presented as a basis, in these cases the definition and measurement of volatility must be specified in the derivative contracts themselves.

In risk management, volatility is associated with calculating the value at risk (VaR), the measurement of which at financial institutions has become a standard practice. Since the entry into force of the first Basel Convention in 1996, banks and trading venues are required to set aside a reserve capital of at least three times the capital at risk. Volatility also has broad consequences for the whole economy. Therefore, policy makers consider volatility as an indicator of uncertainty in the financial market. Besides, it negatively affects market liquidity, as when volatility increases, market liquidity usually decreases. Volatility is also crucial to hedging strategies. During tense market conditions, volatility increases, and this also leads to correlations between different securities. In these circumstances, derivative instruments can act as insurance against a sudden market downturn.

The importance of predicting volatility comes as a natural result of all of these previously mentioned reasons, and the literature on this topic has been extensive and many predictive models have been proposed in the past years.

We have seen a significant increase in the applications of machine learning techniques in many different sectors in recent years. We therefore presented this thesis Which aims to predict market fluctuations using machine learning.
This thesis is structured as follows:

In the first chapter, we presented an overview of market volatility, and reviewed definitions of volatility and their measures. The chapter also provides historical volatility.

in The second chapter, we provided an overview of machine learning (history of machine learning, definition of machine learning, machining learning application, machine learning approaches), and also provides some models of machine learning , and features engineering.

In the third chapter, we provided an overview of the data used. And we described the methodology, the programming of machine learning models ,then summarised the results of the different models of machine learning and compared them to each other. Finally , we improved the models by applying feature engineering and method K-Fold Cross-Validation.

Chapter 01:

Marcket Volatility

Chapter 1

Market volatility

1.1 Introduction

Market volatility has several definitions. In this chapter, we will present a set of definitions of market volatility and also show how important it is to the financial literature. We will also focus in this chapter on realized volatility and implied volatility mainly.

1.2 Defining and measuring volatility:

Volatility is a statistical measure of the dispersion of returns for a specific security or index. The higher the volatility, the riskier the security because the price is less predictable.

Volatility is defined as the standard deviation or variance of returns from the same security or market index [1].There are several methods for determining or calculating the volatility of a stock or index.

Many aspects of volatility have been studied, including exchange rate volatility, oil price volatility, and community price volatility. Volatility is frequently measured by standard deviation, which refers to the daily, weekly, or monthly change in the value of a financial asset.

Volatility is a measure of the unexpected variability of asset returns over time. [2]. A security's or a market index's volatility is a measure of the spread of its returns across their mean. Typically denoted by σ, it is defined as the standard deviation of logarithmic

returns observed over fixed time intervals [3].

The spread of all possible outcomes of an uncertain variable is referred to as volatility. In financial markets, we are frequently concerned with the spread of asset return. Statistically, volatility is often measured as sample standard deviation [4]:

$$\sigma = \sqrt{\frac{1}{T-1}\sum_{t=1}^{T}(r_t - \mu)^2} \tag{1.1}$$

where [4]:

- $r_t = log(\frac{P_t}{P_{t-1}}) \cong \frac{(P_t - P_{t-1})}{P_{t-1}}$ is the return on day t.
- P_t is the price at time t.
- μ is the mean of r_t.
- T is the period of time.

Variance σ^2 is sometimes used as a volatility metric. Because variance is simply the square of standard deviation, it doesn't matter which measure we use to compare the volatility of two assets; however, variance is less stable and desirable as an object for computer estimation and volatility forecast evaluation than standard deviation. Furthermore, standard deviation has the same unit of measure as the mean; if the mean is in dollars, the standard deviation is also expressed in dollars, whereas variance will be expressed in dollar square. For this reason, standard deviation is more convenient and intuitive when we think about volatility.

Volatility is expressed in annual terms. Since most of the variation in securities returns come from trading activity, practitioners consider a year as formed by 252 days,We annu-

alize the standard deviation by multiplying it by 252 [5], which is typically the amount of days stocks are traded within a year. Assuming independent returns and constant volatility, After that, the annual volatility can be calculated as [6]:

$$\sigma_{annual} = \frac{\sigma_\tau}{\sqrt{\tau}} \tag{1.2}$$

where τ is the length of time interval in years. For example, if we had to express daily volatility per annum:

$$\sigma_{annual} = \sigma_{daily}\sqrt{252} \tag{1.3}$$

The volatility mentioned above is commonly referred to as historical volatility. There are two kinds of volatility: realized volatility and implied volatility.

1.3 Historical volatility:

1.3.1 Classical estimator of volatility:

In practice, it is not possible to obtain σ_t at a specific time t. However, it is feasible to estimate the values of σ_t within a period $[t\text{ -}\triangle t, t]$ for $\triangle t \geq 0$. Often, Volatility is estimated using historical returns collected over a given time period. We consider the following estimation of v(t) such that [7]:

$$\nu_t = \frac{1}{\triangle t}\int_{t-\triangle t}^{t} \sigma^2(\tau)d\tau \tag{1.4}$$

Consider putting together an estimate $\nu(t)$ from observed prices $P(t_k)$ with $t_k \in [t\text{ -}\triangle t, t]$ $k = n_0, n_0+1, \ldots, n$, where the time t_k are observed at equal intervals (equispaced), $\delta = t_k - t_{k-1}, t_{n_0} = t - \triangle t$, and $t_n = t$ (or $\triangle t = (n-n_0)\delta$) . The sample variance in discrete

$$\hat{\nu}(t_n) = \frac{1}{\triangle t} \sum_{k=n_0+1}^{n} (E_n - R_k)^2, \tag{1.5}$$

where

$$E_n = \tfrac{1}{n-n_0} \textstyle\sum_{k=n_0+1}^{n} R_k,$$

$$R_k = logP(t_k) - logP(t_{k-1}).$$

The square-root value of equation (1.5) is commonly referred to as classical historical volatility. It is interesting to note that the assumption of collecting data at equal interval of unit of time is crucial, as trades occur at discrete points in time in practice.

1.3.2 Realised volatility:

With the availability of high-frequency data, Andersen et al. [8] suggest an an alternative measure of volatility using all available high-frequency intraday data. This measure is so-called realised volatility. Comparable to the classical historical volatility, the realised volatility is a nonparametric expost estimate of return realisations over a fixed time interval.Particularly, this measure was built on the theory of continuous-time and arbitrage-free processes with the theory of quadratic variation.Let us expand the assumptions for the classical historical volatility estimator to include the following.
To lay out the fundamental concept and intuition, assume that the logarithmic $N \times 1$ vector price process, $logP_t$ follows a multivariate continuous-time stochastic volatility diffusion [9]:

$$dlogP_t = \mu_{(t)} d_t + \sigma_t dw_t \tag{1.6}$$

where, w_t denotes a standard N-dimensional Brownian motion, the process for the $N \times N$ positive definite diffusion matrix, μ_t is the drift term (continuous function) [10] and σ_t is the instantaneous volatility that exaggerates the price change relative to dW_t.

In this time t $-$ 1 to t. The total return over the time period from $t - \triangle t$ to t $0 \leq \triangle t \leq t$, is written [6]:

$$r_{(t,\triangle t)} = P_t - P_{(t-\triangle t)} = \int_{t-\triangle t}^{t} \mu_\tau d\tau + \int_{t-\triangle t}^{t} \sigma_\tau w_t \tag{1.7}$$

and the quadratic variation $QV(t-\triangle t)$ is:

$$QV_{(t-\triangle t)} = \int_{t-\triangle t}^{t} \sigma_\tau^2 d\tau \tag{1.8}$$

Equation (1.9) clearly shows that innovations to the mean component μ_t The return's sample path variation is unaffected. Intuitively, this is due the mean term,$\mu_t dt$,In terms of second order properties, is of lower order than the diffusive innovations., $\sigma_t dw_t$ [11]. As a result, the cumulated returns from many high-frequency values over a short time interval can be neglected. Therefore, the variance in equation (1.5) coincides with the quadratic variation $QV(t, \triangle t)$. Now, considering a discrete partition $(t - \triangle t + \frac{j}{n}, j = 1, \ldots, n.\triangle t)$ of that the time interval $[t - \triangle t, t]$ is observed is observed equally at $\frac{j}{n}$.,so, the realized volatility (the realized variance),RV, is defined like:

$$RV(t, \triangle t; n) = \sum_{j=1}^{n.\triangle t} r(t - \triangle t + \frac{j}{n}, \frac{1}{n})^2 \tag{1.9}$$

Under semi-martingale theory [12], As the sampling frequency n increases, the realized variance measure converges in probability to the quadratic variation, i.e.

$$RV(t, \triangle t) \longrightarrow QV(t, \triangle t) \quad \text{as} \quad n \longrightarrow \infty \tag{1.10}$$

So, RV approximates QV as the sampling frequency n increases. Furthermore, the formula number (1.10) can be rewritten for a one-day period t volatility estimate as:

$$RV(t - \triangle t, t) = \sqrt{\sum_{j=0}^{N-1} r_{t-j\delta}} \tag{1.11}$$

where, $[t-\triangle t, t]$ is the length of one day, $\delta = \frac{1}{N}, r_{j-t\delta} = P_{(j-t\delta)} - P_{(j-(j+1)\delta)}$, and returns are assumed to be sampled evenly at δ time step with N observations in that time period. Since we assume that the data is collected at equal-space, can be assembled at second, minute and hour frequencies.

1.4 Implied volatility:

Implied volatility (IV), We find this expression definition in "investopedia". The expression implied volatility indicates to a metric that captures the market's perception of the likelihood of changes in a given security's price. Investors can use implied volatility to plan future moves and supply and demand, and often appoint it to price options contracts.

Implied volatility (IV) in the market indicates to the predicted magnitude, or one standard deviation (SD) scope, of potential movement away from the underlying price in a year's time.

Implied volatility ($\sigma_{CIV,t}$ and $\sigma_{PIV,t}$) is a stock's current volatility as reflected by its option price. Because option pricing models cannot be inverted very easily, so implied volatility is calculated numerically. Implied volatility is estimated using the BS(Black-Scholes) selection pricing model employing the bisection method as Displayed as follows. [13]:

$$\text{Volatility Estimate } = \sigma_l + \frac{C - C_H}{C_L - C_H}(\sigma_H - \sigma_L) \tag{1.12}$$

where σ_L and σ_H are the low and high volatility values respectively, C_L and C_h are the corresponding options values and C is the market price of the option.

An option pricing model can be used as a tool, to obtain the features of option prices, such as implied volatility, by relating the option price with the price of the underlying asset under the arbitrage free assumption to keep a fair market [14]. The main thrusts behind generating the Black-Scholes model is to find a way to transform the market prices into an expression in terms of implied volatility [15].

Implied volatility is known to be a forecasting made by investors of the future movement of the underlying asset, i.e. it is their belief of how variable the stock is going to

be between today and the maturity date. Since the accurate forecast is crucial, reliable methods for approximating it are required. Given the observed option value V_{mkt}, implied volatility can be derived from [16]

$$V_{mkt} = BS(\sigma^*, S, K, T, r) \tag{1.13}$$

Since the Black-Scholes equation is monotonic with respect to σ^*, implied volatility exists and can be written implicitly by means of the inverse Black-Scholes function as

$$\sigma^* = BS^{-1}(V_{mkt}, S, K, T, r) \tag{1.14}$$

which does not have an analytic solution and thus, needs to be solved numerically [17].

Usually, in order to approximate the solution of (1.15), iterative approaches, such as Newton-Raphson method, are used. Those approaches take five input variables which are shown in Equation (1.15).

However, when the price of the option is unknown, the calculation of the option by (1.15) becomes impossible. Therefore, another approach could be to look at the historical volatility and try to predict the implied volatility from there.

1.5 Conclusion

We have seen during this chapter the principle notions of the domain of market volatility, the market volatility has a role in the stock market and international oil markets, etc. In this chapter, we talked about general concepts about market volatility and realized and implied volatility.

Chapter 02:

Machine learning

Chapter 2

Machine Learning

2.1 Introduction

Machine learning has become one of the most important pillars of information technology over the past years, and thus it has become an essential part of our daily lives, and an urgent necessity that contributes to facilitating and simplifying various functions and activities, even without us realizing it. With the availability of this large number of data and information that is constantly increasing and developing, we believe that smart data analysis will become widespread and will be one of the necessary components of technological progress and development.

2.2 History of machine learning :

The term machine learning is a term coined in 1959 by Arthur Samuel, one of the American IBMer companies and a pioneer in the field of computer games and artificial intelligence. Nelson's book on machine learning also addressed machine learning research during the 1960s, focusing mostly on machine learning to classify patterns. Interest in pattern recognition continued into the 1970s, and in 1981 a report was presented on the use of teaching strategies to teach a neural network to recognize 40 characters from a computer terminal.

Tom M. Mitchell has given a formal definition: "A computer program is said to learn from experience with respect to some task category and measure performance if its performance on tasks at task category, as measured by performance, improves with experience." This definition provides for tasks which machine learning relates to a basic operational

definition rather than defining the domain in cognitive terms. This comes after Alan Turing suggested in his paper "Machine Computing and Intelligence," where the question "Can machines think?" With the question "Can machines do what we (as thinking entities) can do?" [18].

Modern machine learning has two primary goals, the first is to classify data based on a set of developed models, and the second is to predict future outcomes based on those models.

2.3 Definition of machine learning:

Machine Learning (ML): It is a branch of Artificial Intelligence. Where machine learning algorithms build models based on data samples, called training data, to use them in order to make predictions or make decisions without directly programming them. "It is the study of computer algorithms that can improve automatically through experience and with data" [19].

Machine learning algorithms are used in several applications in different fields, such as medicine, voice and image recognition, email follow-up... These tasks are difficult to develop traditional algorithms to perform, or even useless for them. One of the most important areas of machine learning is making predictions using a computer that is related to computational statistics, in addition to data mining and exploratory data analysis through unsupervised learning. Some machine learning applications use data and neural networks in a way that mimics the functioning of the human brain.

2.4 Machine learning applications:

Machine learning algorithms are used in several applications in different fields, including:

2.4.1 Recognize images:

It is one of the most famous applications for machine learning, and image recognition applications have become known to most people, and they are used to identify people, places, and others.

2.4.2 Speech recognition:

It is one of the modern applications of machine learning and it has become popular and has several uses such as converting voice instructions into written text, for example, the search application on the Google search engine with the option of voice search.

2.4.3 Self-driving cars:

Today, these cars have become one of the most popular applications for machine learning, as major companies have become competing to manufacture cars that rely on this highly efficient technology, the most famous of which is Tesla, the leader in this field.

2.4.4 Medical diagnosis:

Machine learning and artificial intelligence are becoming increasingly used in the medical field to help doctors and specialists detect and diagnose diseases, such as finding tumors easily and quickly.

2.4.5 Language machine translation:

It is one of the applications that has become well known and widely used, as by using the applications available for machine translation, we do not have to have the language of the place we want to visit, for example the machine translation provided by Google.

2.4.6 Stock market prediction:

It is one of the most recent applications of machine learning at all, as it has become widely used in the financial and stock markets, to predict the rise and fall of financial stocks and determine the trends of these markets. This is the subject of our research.

2.5 Machine learning approaches:

Broadly, machine learning is split into four parts – supervised learning, unsupervised learning, and semi-supervised learning, Reinforcement learning as shown in the following figure:

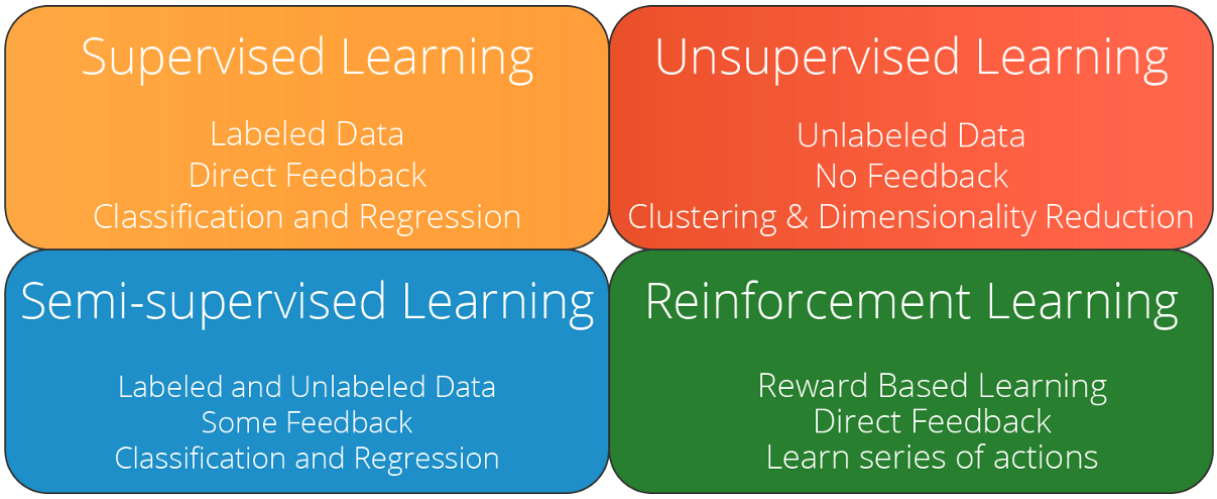

Figure 2.1: Machine learning approaches

2.5.1 Supervised learning:

In order to learn a general rule that defines inputs for outputs, the ”teacher” gives the computer examples of the desired inputs and outputs [20]. The support vector machine is a supervised learning model that splits data with a linear boundary into regions separated from each other, separating white circles from black circles. Supervised learning algorithms build a mathematical model of a set of data that contains the desired inputs and outputs. This data, called training data, is made up of a set of training examples. Each training example contains one or more desired inputs and outputs, also known as a supervisory signal. In the mathematical model, each training example is represented by a vector or matrix, called the feature vector, and the training data is represented by a matrix. Through iterative optimization of an objective function, supervised learning algorithms based on iterative optimization of an objective function learn a function that you can use to predict multiple outputs associated with new inputs [21]. The optimization functionality of the algorithm allows the identification of new input outputs. We say that the algorithm has learned to perform the task entrusted to it if it works on improving the accuracy of its outputs or predictions over time. Supervised learning is further divided into two types based on the kind of problem. Regression and Classification.

2.5.1.1 Regression:

Regression is used where continuous values are being predicted. market volatility prediction is typically a regression problem. In this study, we have continuous data that

changes every minute of every day. Regression analysis is most commonly used to estimate a relationship between two or more variables. Hence it is used in forecasting or prediction based applications. Regression analysis can not only indicate a significant relationship between explained or explanatory variables, but it can also indicate a strength of the relationship between multiple explanatory variables on a single explained variable. This is a very important part of the regression analysis that is used in this study.

Regression analysis is further classified into several types. Linear regression, logistic regression, polynomial regression, and stepwise regression are some of the more common types of regression. These types are mainly divided based on the number of explanatory variables, the number of explained variables and the shape of the regression line. The most commonly used regression analysis technique is linear regression, has a linear regression line and it establishes a relationship between a explained variable and one or more explanatory variables. For binary explained variables, logistic regression is used.

2.5.1.2 Classification:

Classification is used to predict discrete values. It is applied when there is input data that is categorized into separate categories. Such as classification of images or speech recognition, for example.But classification may not always be a good approach in many real world cases, we may have inputs that can belong to boundary states with features of both classes.Also sometimes the entries may not belong to any of the categories.So even though categorization seems like a more natural approach on paper, it isn't always a good approach

2.5.2 Unsupervised learning:

The machine learning algorithm is not given any labels. Unsupervised learning in itself can be a goal or an end (discovering hidden patterns in the data) and it can also be a means to an end (distinguished learning). Unsupervised learning algorithms take data that contains only the input, and find themselves a structure in this data, as aggregation or aggregation of data points. Therefore, these algorithms learn from test data that has not been labeled or ranked. Instead of responding to feedback, unsupervised learning algorithms identify commonalities in the data to react based on whether or not they are

present (commonality) in each new piece of data [21]. The field of density estimation in statistics is one of the central applications of unsupervised learning. Although unsupervised learning includes other areas that include summarizing and explaining features of data. Unsupervised learning can be divided into two kinds:

2.5.2.1 Clustering:

Clustering, It is the process of organizing and arranging objects into several groups whose members are similar in some way [21]. In this algorithm, the data is divided into distinct groups so that each group is similar in some way.One of the most popular clustering algorithms is k-means.

2.5.2.2 Dimensionality Reduction:

Dimensional reduction is the process of reducing the dimensions of the feature set, by which the number of random variables under study is reduced by obtaining a set of key variables. It can be called as "feature count". Most dimensionality reduction techniques are feature removal or extraction. A common technique for dimensionality reduction is Principal Component Analysis (PCA). Which involves changing higher dimensional data (eg, 3D) into a smaller space. It results in a smaller dimension of the data (2D instead of 3D), all without changing the data with all the original variables in the model. The manifold hypothesis suggests that high-dimensional data sets lie along low-dimensional manifolds, and this assumption is carried out by many dimensionality reduction techniques, and this leads to the fields of diverse learning and diverse organization [22].

2.5.3 Semi-Supervised Learning:

Semi-supervised learning lies between unsupervised learning (without any labeled training data) and supervised learning (with fully labeled training data). A computer program interacts with a dynamic environment in which it performs a specific goal as it navigates a problem space, and that program is provided with feedback similar to rewards, which it is trying to magnify. Some training examples lack training labels, though machine learning researchers have found that unlabeled data can lead to a significant improvement in learning accuracy, if used in conjunction with a small amount of labeled data [23].

2.5.4 Reinforcement learning:

Reinforcement learning is an area concerned with how software agents take actions in an environment to maximize the idea of cumulative reward. Due to its generality, this field is studied in several different disciplines, including: information theory, operations research, genetic algorithms, statistics, and swarm intelligence. In machine learning, the environment is usually represented as a Markov decision process (MDP). Many reinforcement learning algorithms use dynamic programming techniques. They do not assume knowledge of the exact mathematical model of MDP, and these algorithms are used when the exact models are not applicable. One of the areas of use of these algorithms in self-driving vehicles, and in learning to play against humans [24].

2.6 Machine Learning Models:

There are many machine learning models that cannot be mentioned all, and we will provide a definition of only the four models on which our study is based.

2.6.1 XGBoost:

XGBoost is a scalable machine learning system. This model was developed by Chen and Guestrin (2015) who describe it in their paper as a scalable machine learning system based on the concept of gradient tree reinforcement. It gives advanced results in a wide range of problems, especially classification problems. Its scalability in all scenarios is the most important factor behind its success. The system runs 10 times faster than common solutions found on a single machine and can accommodate billions of examples in distributed or memory-limited settings. It has also recently gained in popularity as the best algorithm for many teams that have won machine learning competitions [25].

Chen and Guestrin listed some of the new innovative features that contribute to XGBoost's scalability, including:

- A new tree learning algorithm to deal with sparse data sets.
- A theoretically justified weighted graph procedure, allowing handling of instance weights in approximate tree learning.

- Parallel and distributed computing, to make the training period much shorter, to enable us to make faster model explorations.
- Exploiting out-of-core computations allows data scientists to process large amounts of data.

Through the growth of level-wise trees the XGBoost tree expands.The corresponding figure shows an example of the way the XGBoost tree expands [26].

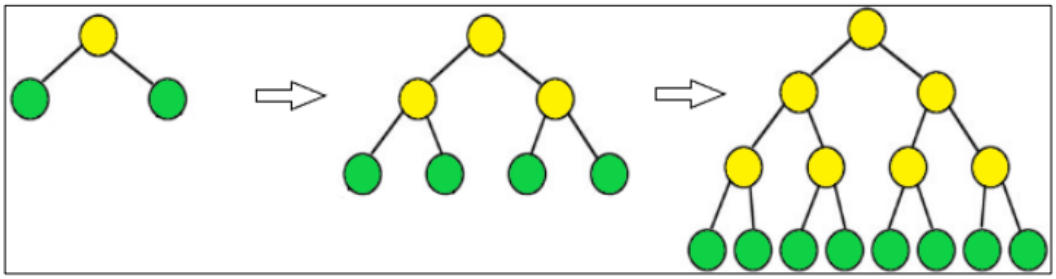

Figure 2.2: XGBoost tree expansion method

Yellow circles denote an older level of leaves, which have already been calculated and green circles denote the level of new leaves that are added to the tree.

2.6.2 LightGBM:

LightGBM is an acronym for Light Gradient Boosting Machine developed by Ke et al. (2017), another GBDT model, can significantly outperform XGBoost in terms of computing speed and memory consumption. This algorithm contains two new technologies: gradient-based one-sided sampling and exclusive feature aggregation to handle a large number of instances Data and a large number of features respectively [27]. In scenarios involving large amounts of data and a number of features. The LightGBM model uses XGBoost as a baseline, while approaching the classification problem in a completely different way, through the use of the two new technologies.

In scenarios involving large amounts of data and a number of features. The LightGBM model uses XGBoost as a baseline, while approaching the classification problem in a completely different way, through the use of the two new technologies.

Gradient-based one-sided sampling necessitates that the model omits most examples where we expect the gradient weight to be smaller. This may help us avoid going down the branches, which may be less important. Data states with different gradient values have a different effect on the expected information acquisition. This is how trees are expanded in LightGBM, sheet by sheet. The following figure illustrates this [26]:

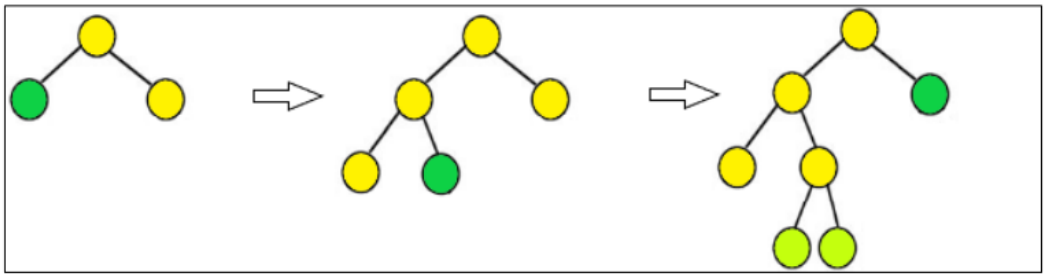

Figure 2.3: LightGBM method

The yellow circles denote older leaves, which have already been explored, the green circles denote the current leaf that is being considered.

LightGBM uses another method where it aggregates exclusive features, which helps reduce feature variance, which makes it possible to reduce model training and deployment time much faster than other GBDTs [26].

2.6.3 CatBoost:

CatBoost is a model proposed by Prokhorenkova et al. (2019) Focusing more on categorical features, they argued that the CatBoost algorithm introduces two new core functions, a rotation-driven rank-boosting algorithm and another algorithmic approach specifically for dealing with categorical features. They show that CatBoost combats the problem of exponential feature set growth by using a greedy method At each new split of the current tree. It is also capable of handling scenarios where the number of classes is too large for other GBDT models. CatBoost addresses this issue with these three steps:

- It divides the data into random subgroups as a first stage.
- Converts labels to integers.
- Finally, it converts the remaining categorical attributes to numerical values.

The resulting data set should then be ready for use by the CatBoost model. The exact technical specifications of each step largely depend on what the developers specify. CatBoost is best when we are dealing with non-numeric data as it can convert categorical values to integers in an optimal way. Performance may vary greatly depending on the types of features implemented because CatBoost was developed to handle categorical features. It's a good idea to implement two versions of the CatBoost model to quantify the difference in performance, with one being trained on a categorical feature dataset and the other on a hot-coded single feature dataset.[26].

2.6.4 Linear Regression:

Linear regression is a model that has been around for a long time and is one of the most important tools in the field of statistics. This linear regression model can be represented by the following mathematical expression:

$$\begin{aligned} \mathbf{X} &= (X_1, X_2, \ldots, X_p) \\ \beta &= (\beta_1, \beta_2, \ldots, \beta_p) \\ \hat{Y} &= \hat{\beta}_0 + \sum_{j=1}^{p} X_j \beta_j \end{aligned} \tag{2.1}$$

X_t represents the model input variables, and Y represents the model output variable. $\beta_i, i = 1, 2, \ldots, p$, are the model parameters to be estimated.The term β_0 is the intersection, or the so-called bias in machine learning. It is often more convenient to include the constant variable 1 in X , include β_0 in the vector of coefficients β , and then write the linear model in vector form as an internal product:

$$\hat{Y} = X^T \hat{\beta} \tag{2.2}$$

Estimation using the least squares method.In this approach the most common method for estimating the model parameters is, and an estimate is made in order to reduce the remaining sum of squares:

$$RSS(\beta) = \sum_{j=1}^{N} (y_i - x^T \hat{\beta})^2 \tag{2.3}$$

writing the formula in matrix notation we have:

$$RSS(\beta) = (y - X\beta)^T (y - X\beta) \tag{2.4}$$

where X is an $N * P$ matrix with each row and input vector, and y is an N vector of the outputs in training set. differentiating in order of β and equal to zero we get to the equations:

$$\begin{aligned} X^T(y - X\beta) &= 0 \\ \beta &= (X^T X)^{-1} X^T \end{aligned} \tag{2.5}$$

2.7 Feature engineering for machine learning:

Feature engineering is the set of preprocessing steps by which raw data is converted into features that can be used in machine learning algorithms and models. Predictive models consist of outcome variable and predictor variables, and using the feature engineering process the most useful predictor variables are created and selected for the predictive model. Since 2016, automated feature engineering has been made available in some machine learning programs. Feature engineering in ML consists of four main steps [28]:

2.7.1 Create features:

It involves identifying the variables that will be most useful in the predictive model. It is a subjective process that takes place with human intervention using his creativity. In it, existing features are mixed through several operations such as addition, subtraction, multiplication, and ratio in order to create new derived features with greater predictive power.

2.7.2 Transformations:

It includes the treatment of prediction variables to improve the performance of the model. As we work to ensure the flexibility of the model in the diversity of data that it can accommodate; ensuring that all variables are on the same scale to facilitate understanding of the model; improve accuracy; Ensure that all features are within the acceptable range of the model to avoid arithmetic errors.

2.7.3 Feature extraction:

it is the extraction and automatic creation of new variables from the raw data. The purpose is to automatically reduce the size of the data to a more manageable set of modeling. Some feature extraction methods include block analysis, text analytics, edge detection algorithms, and principal component analysis.

2.7.4 Feature selection:

Feature selection algorithms basically analyze and judge different features. Then you rank them to determine which are not relevant, which redundant features should be removed, and which features are most useful to the model and which should be prioritized.

2.8 Conclusion

In this chapter, we provide an introduction, history, and definition to machine learning, and applications, approaches of machine learning, and Feature engineering, than an overview about different types of algorithms exist.

The next chapter describes implementation methods of machine learning (LightGBM(Light Gradient Boosting Machine) and XGBoost and CatBoostRegressor and Linear Regression), and shows discuss experimental results.

Chapter 03:

Experimental results

Chapter 3

Experimental results

3.1 Introduction

After we have seen the theoretical concept for our project, and to understand this project of market volatility, we need to explain the conception using the tools and methods that allow the realization of this kind of system.

In this chapter we are going to design our intelligent regression system, by detailing how it works, for simplify the comprehension of our project.

3.2 Dataset :

The regression applies to several types of data from several fields (finance, biology,chemistry, marketing etc.).

We started collecting the data by using the information provided on the kaggle.com site. Our study contains stock market data relevant to the practical execution of trades in the financial markets. In particular, it includes order book snapshots and executed trades. With one second resolution, it provides a uniquely fine grained look at the micro-structure of modern financial markets.

3.2.1 Book parquet:

A parquet file partitioned by stock_id. Provides order book data on the most com-

petitive buy and sell orders entered into the market. The top two levels of the book are shared. The first level of the book will be more competitive in price terms, it will then receive execution priority over the second level.

- stock_id - ID code for the stock. Not all stock IDs exist in every time bucket. Parquet coerces this column to the categorical data type when loaded; you may wish to convert it to int8.
- time_id - ID code for the time bucket. Time IDs are not necessarily sequential but are consistent across all stocks.
- seconds_in_bucket - Number of seconds from the start of the bucket, always starting from 0.
- bid_price[1/2] - Normalized prices of the most/second most competitive buy level.
- ask_price[1/2] - Normalized prices of the most/second most competitive sell level.
- bid_size[1/2] - The number of shares on the most/second most competitive buy level.
- ask_size[1/2] - The number of shares on the most/second most competitive sell level.

	time_id	seconds_in_bucket	bid_price1	ask_price1	bid_price2	ask_price2	bid_size1	ask_size1	bid_size2	ask_size2	stock_id
0	5	0	1.001422	1.002301	1.00137	1.002353	3	226	2	100	0
1	5	1	1.001422	1.002301	1.00137	1.002353	3	100	2	100	0
2	5	5	1.001422	1.002301	1.00137	1.002405	3	100	2	100	0
3	5	6	1.001422	1.002301	1.00137	1.002405	3	126	2	100	0
4	5	7	1.001422	1.002301	1.00137	1.002405	3	126	2	100	0
5	5	11	1.001422	1.002301	1.00137	1.002405	3	100	2	100	0
6	5	12	1.001422	1.002301	1.00137	1.002405	3	126	2	100	0
7	5	14	1.001422	1.002301	1.00137	1.002405	3	126	2	100	0
8	5	15	1.001422	1.002301	1.00137	1.002405	3	126	2	100	0
9	5	16	1.001422	1.002301	1.00137	1.002405	3	126	2	100	0

Figure 3.1: DataFrame of Book_[train/test].parquet

3.2.2 Trade parquet:

A parquet file partitioned by stock_id. Contains data on trades that actually executed. Usually, in the market, there are more passive buy/sell intention updates (book updates) than actual trades, therefore one may expect this file to be more sparse than the order book.

- stock_id - Same as above.
- time_id - Same as above.
- seconds_in_bucket - Same as above. Note that since trade and book data are taken from the same time window and trade data is more sparse in general, this field is not necessarily starting from 0.
- price - The average price of executed transactions happening in one second. Prices have been normalized and the average has been weighted by the number of shares traded in each transaction.
- size - The sum number of shares traded.
- order_count - The number of unique trade orders taking place.

	time_id	seconds_in_bucket	price	size	order_count	stock_id
0	5	21	1.002301	326	12	0
1	5	46	1.002778	128	4	0
2	5	50	1.002818	55	1	0
3	5	57	1.003155	121	5	0
4	5	68	1.003646	4	1	0
5	5	78	1.003762	134	5	0
6	5	122	1.004207	102	3	0
7	5	127	1.004577	1	1	0
8	5	144	1.004370	6	1	0
9	5	147	1.003964	233	4	0

Figure 3.2: DataFrame of Trade_[train/test].parque

3.2.3 Train:

Provides the mapping between the other data files and the submission file.

- stock_id - Same as above, but since this is a csv the column will load as an integer instead of categorical.
- time_id - Same as above.
- target - The realized volatility computed over the 10 minute window following the feature data under the same stock/time_id. There is no overlap between feature and target data.

	stock_id	time_id	target
0	0	5	0.004136
1	0	11	0.001445
2	0	16	0.002168
3	0	31	0.002195
4	0	62	0.001747
5	0	72	0.004912
6	0	97	0.009388
7	0	103	0.004120
8	0	109	0.002182
9	0	123	0.002669

Figure 3.3: DataFrame of Train.csv

There are 112 types of stock_id, 3830 types of time_id, and 414287 types of target.

3.3 Data processing :

The dataset includes the Weighted averaged price and Log returns and Realized volatility.

3.3.1 Weighted averaged price:

The order book is also one of the primary source for stock valuation. A fair book-based valuation must take two factors into account: the level and the size of orders. In this competition we used weighted averaged price, or WAP, to calculate the instantaneous stock valuation and calculate realized volatility as our target.

To calculate realized volatility first we calculate the weighted average price using the

formula of WAP can be written as below, which takes the top level price and volume information into account:

$$WAP = \frac{(\text{BibPrice1} * \text{AskSize1} * \text{AskPrice1} * \text{BidSize1})}{(\text{BidSize1} * \text{AskSize1})} \tag{3.1}$$

As you can see, if two books have both bid and ask offers on the same price level respectively, the one with more offers in place will generate a lower stock valuation, as there are more intended seller in the book, and more seller implies a fact of more supply on the market resulting in a lower stock valuation.

Note that in most of cases, during the continuous trading hours, an order book should not have the scenario when bid order is higher than the offer, or ask, order. In another word, most likely, the bid and ask should never be in cross.

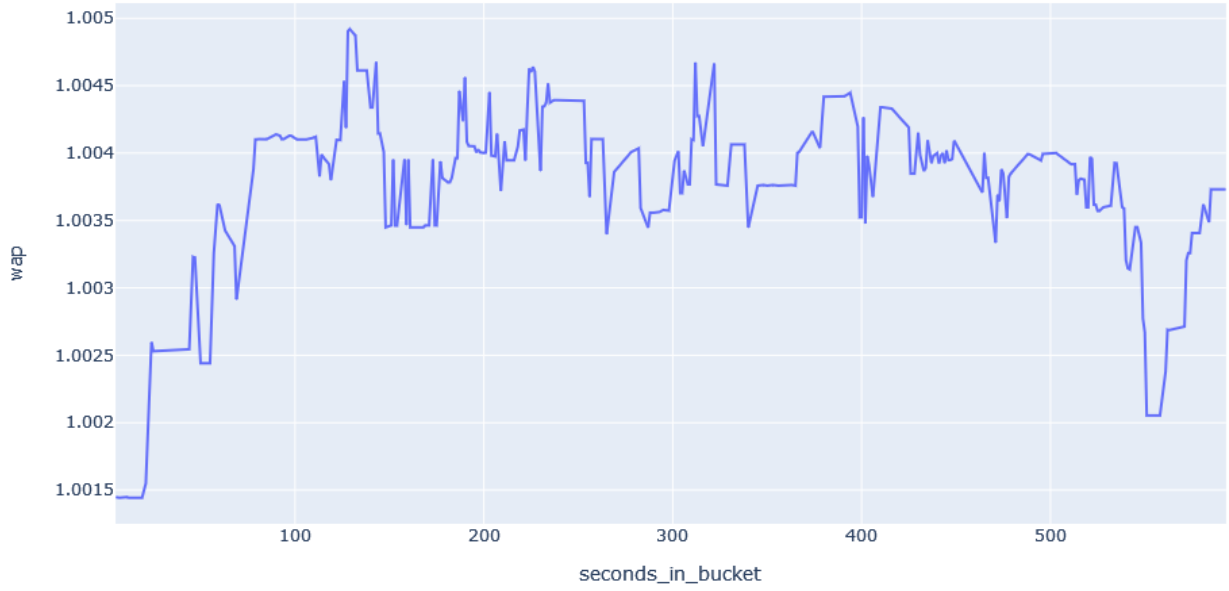

Figure 3.4: WAP stock_id_0,time_id_5

The code below calculates functions of Weighted averaged price:

```
1
2   def WAP1(df):
3   WAP = (df['bid_price1'] * df['ask_size1'] + df['ask_price1'] *
```

```
    df['bid_size1'])/(df['bid_size1'] + df['ask_size1'])
    return WAP

    def WAP2(df):
    wap = (df['bid_price2'] * df['ask_size2'] + df['ask_price2'] *
    df['bid_size2'])/(df['bid_size2'] + df['ask_size2'])
    return wap

```

Listing 3.1: Function of Weighted averaged price

3.3.2 Calculate Log returns:

Returns are widely used in finance, however log returns are preferred whenever some mathematical modelling is required. Calling P_t the price of the stock S at time t, we can define the log return between t_1 and t_2 as:

$$r_{t_1,t_2} = \log(\frac{P_{t_1}}{P_{t_2}}) \tag{3.2}$$

Usually, we look at log returns over fixed time intervals, so with 10-minute log return we mean $r_t = r_{t-10min,t}$. Where we use WAP as price of the stock to compute log returns.

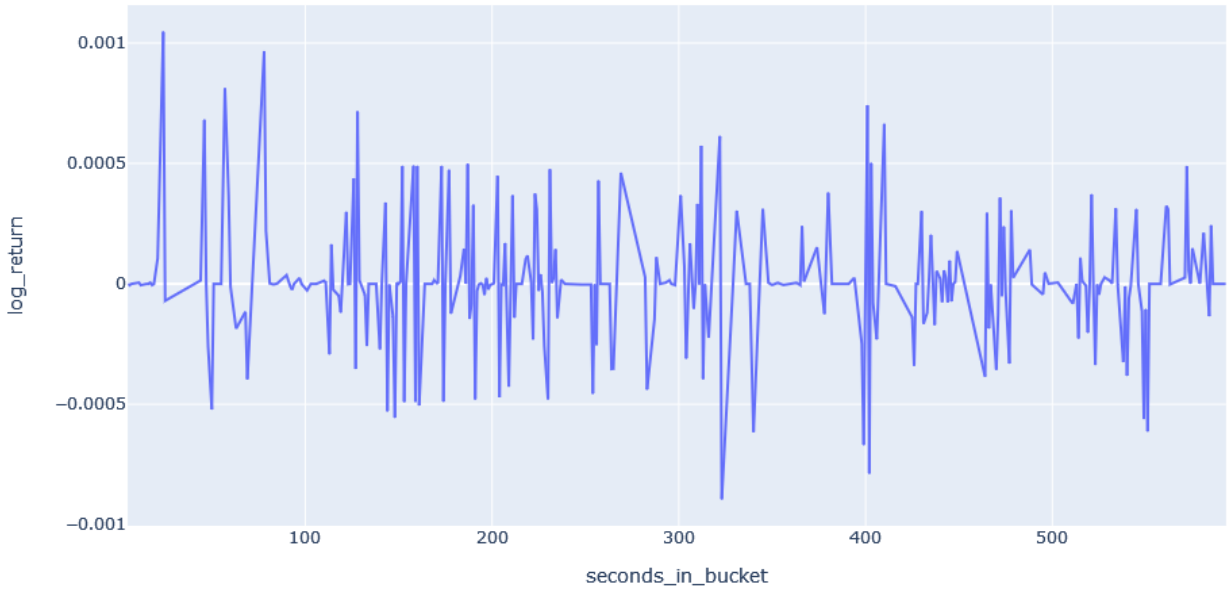

Figure 3.5: Log return of stock_id_0,time_id_5

The code below calculates function of Weighted averaged price:

```
def log_return(WAP):
return np.log(WAP).diff()

```

Listing 3.2: Function of Log returns

3.3.3 Calculate Realized volatility:

When we trade options, a valuable input to our models is the standard deviation of the stock log returns. The standard deviation will be different for log returns computed over longer or shorter intervals, for this reason it is usually normalized to a 1-year period and the annualized standard deviation is called volatility Which we talked about it in the first chapter.

We will compute the log returns over all consecutive book updates and we define the realized volatility, σ, as the squared root of the sum of squared log returns.

$$\sigma = \sqrt{\sum_{t} r_{t-1,t}^2} \tag{3.3}$$

We want to keep definitions as simple and clear as possible, So we are not annualizing the volatility and we are assuming that log returns have 0 mean.

The code below calculates function of Realized volatility:

```
def realized_volatility(log_r):
return np.sqrt((log_r**2).sum())


```

Listing 3.3: Function of Realized volatility

3.4 Suggested action steps:

In this work, we relied on the prediction of achieved volatility, using four models (LightGBM, XGBoost, CatBoost and linear regression). The suggested steps are illustrated in the following figure:

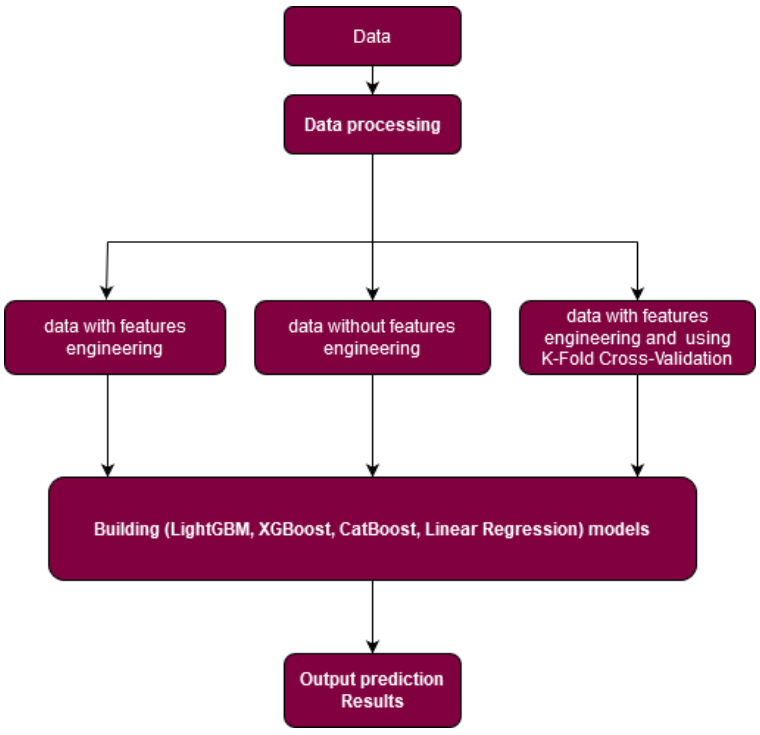

Figure 3.6: Suggested action steps

3.5 Methodology:

In order to conduct a fair comparison between the four different models machine learning procedures to predict to predict the market volatility of Optiver. LightGBM(Light Gradient Boosting Machine) and XGBoost and CatBoostRegressor and Linear Regression methods are used. The performance of the models is measured by two metrics:

RMSPE: Root Mean Square Error, it is a measure of the deviation between the observed value and the real value and it is often used as a standard for the prediction results of machine learning models [29].

$$RMSPE(X,) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}((X_i) - Y_i)^2} \tag{3.4}$$

Where (X_i) is the prediction value at the time i, (Y_i) is the and ground truth value at time t and m is the total number of the train or test data.

R2 score: is a metric that tells the performance of your model, not the loss in an absolute sense that how many wells did your model perform.

$$R2 \quad score = 1 - \frac{SSr}{SSm} \tag{3.5}$$

where SSr is squared sum error of regression line and SSm is squared sum error of mean line.

The machine learning library sklearn for Python is used to create the addressed machine learning models.

For all four learning methods(LightGBM and XGBoost and CatBoost and Linear Regression), multiple models are created, and two tests are carried out. In many cases, the dataset is first shuffled and then split into training and testing, and this is after calculating (WAP,log(return),realized volatility), we used the machine learning library sklearn for Python to split dataset for the best results. Afterwards, fitting the model is done, and then the fitted model is used to predict the realized volatility. The predictions

are then compared with the actual values, and the errors are calculated. To do this work we used Google Colab and google drive. The following code allows to transfer data from google drive to google colab:

```
from google.colab import drive
drive.mount('/content/drive')
from zipfile import ZipFile
with ZipFile('/content/drive/MyDrive/optiver-realized-volatility-prediction.zip','r') as objet:
objet.extractall('data_optiver')

```

Listing 3.4: Code import dataset

3.6 Features of model:

Features are a vital component for machine learning models. They are the foundation on what the predictions are made. In this case, We used three files (book-train, trade-train, train.cvc) to prepare the features as shown in figure as follows:

	stock_id	time_id	real_vol_1	real_vol_2	real_vol_trade	target
0	32	5	0.002370	0.002919	0.001805	0.002649
1	32	11	0.001249	0.001784	0.000805	0.000688
2	32	16	0.002012	0.002388	0.001762	0.002431
3	32	31	0.001421	0.001984	0.001236	0.001881
4	32	62	0.001308	0.001663	0.000990	0.001463

Figure 3.7: Dataframe after preparing data

In this dataframe, we have six features(variables) stock-id, time-id, target, and (real-vol1, real-vol2, real-vol-trade) that we obtained from calculate function (WAP and log return). and the code below explains that:

```
def consol_book_df(path):
df = read_data(path)
df['stock_id'] = int(path.split("=")[1])
df['WAP1'] = WAP1(df)
df['WAP2'] = WAP2(df)
```

```
    df['book_log_ret1'] = df.groupby('time_id')['WAP1'].apply(log_return
).fillna(0)
    df['book_log_ret2'] = df.groupby('time_id')['WAP2'].apply(log_return
).fillna(0)
    final_book = df.groupby(['stock_id', 'time_id']).agg(
    real_vol_1 =('book_log_ret1', realized_volatility),
    real_vol_2 = ('book_log_ret2', realized_volatility)
    ).reset_index()
    return final_book

    def consol_trade_df(path):
    df = read_data(path)
    df['stock_id'] = int(path.split("=")[1])
    df['trade_log_ret'] =df.groupby('time_id')['price'].apply(log_return
).fillna(0)
    final_trade = df.groupby(['time_id', 'stock_id']).agg(
    real_vol_trade=('trade_log_ret', realized_volatility)).reset_index()
    return final_trade

```

Listing 3.5: Code Correlation between the 6 features

The linear correlation matrix presented on the below of figure shows a linear correlation between , time-id, target, real-vol1, real-vol2, real-vol-trade as follows:

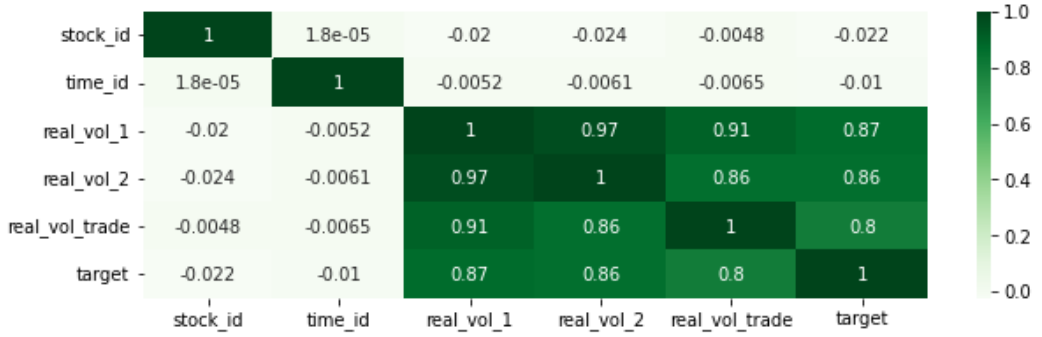

Figure 3.8: : Correlation between the 6 features

The code below is Correlation between the 6 features:

```
    from matplotlib import pyplot as plt
    corr = train_set.corr()
    plt.figure(figsize=(10,3))
```

```
    sns.heatmap(corr, cmap="Greens",annot=True)
    plt.show()
```

Listing 3.6: Code Correlation between the 6 features

3.7 Results:

3.7.1 Models Comparison:

In this section,the four models can now be compared.The comparison of the models are displayed in Table 3.1 as follows:

Table 3.1: Best performing Machine learning models comparison

model/metric	r2-score	RMSPE
LightGBM	**0.817**	**0.286**
XGBoost	0.791	0.303
CatBoost	0.784	0.302
Linear Regression	0.766	0.347

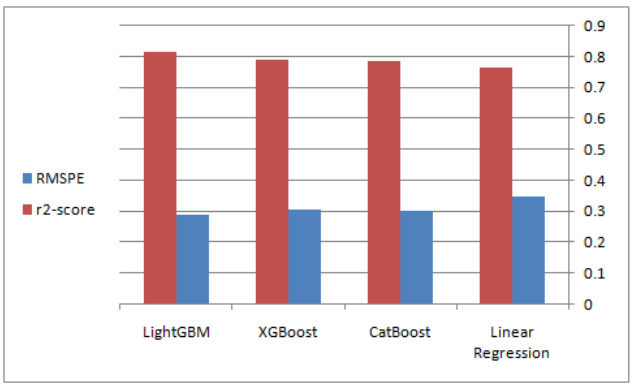

Figure 3.9: Histogram Comparison between models

We note through the table 3.1, the best result of each metric is colored gray. As seen, each model excels in a specific metric. the result shows that model LightGBM has a lowest RMSPE score that is 0.286 and a highest r2-score is 0.817, and the RMSPE of other models such as XGBoost, CatBoost, Linear Regression are respectively 0.303, 0.302, 0.347, and r2-score are 0.791, 0.784, 0.766. the best model is LightGBM.

3.7.2 Feature engineering for models:

In this section. Features will be constructed using book-train and trade-train files. Besides realized-volatility-1, realized-volatility-2, features encompass statistical features as mean, max, min, std and median for each of the variable (price-spread, bid-spread, ask-spread, total-volume volume-imbalance, trade-log-ret, position-size, average-ord-size). and the code below explains that:

```
def consol_book_df(path):
df = read_data(path)
df['price_spread'] = (df['ask_price1'] - df['bid_price1']) / ((df['ask_price1'] + df['bid_price1'])/2)
df['bid_spread'] = abs(df['bid_price1'] - df['bid_price2'])
df['ask_spread'] = abs(df['ask_price1'] - df['ask_price2'])
df['total_volume'] = (df['ask_size1'] + df['ask_size2']) + (df['bid_size1'] + df['bid_size2'])
df['volume_imbalance'] = abs((df['ask_size1'] + df['ask_size2']) - (df['bid_size1'] + df['bid_size2']))
df['WAP1'] = WAP1(df)
df['WAP2'] = WAP2(df)
df['book_log_ret1'] = df.groupby('time_id')['WAP1'].apply(log_return).fillna(0)
df['book_log_ret2'] = df.groupby('time_id')['WAP2'].apply(log_return).fillna(0)
df_book=df.groupby(['stock_id','time_id'])[ 'price_spread', 'bid_spread', 'ask_spread', 'total_volume','volume_imbalance'].agg(['min', 'max', 'mean','std', 'median'])
final_book = df.groupby(['stock_id', 'time_id']).agg(
real_vol_1 =('book_log_ret1', realized_volatility),
real_vol_2 = ('book_log_ret2', realized_volatility)).reset_index()
final_book = final_book.merge(df_book,on=['stock_id','time_id'], how='left')
return final_book

def consol_trade_df(path):
df = read_data(path)
df['trade_log_ret'] = df.groupby('time_id')['price'].apply(log_return).fillna(0)
df['position_size'] = df['price']*df['size']
```

```
    df['average_ord_size'] = df['size']/df['order_count']
    df_trade=df.groupby(['stock_id','time_id'])[ 'size', 'position_size'
, 'average_ord_size', 'order_count',
    'seconds_in_bucket'].agg(['min', 'max', 'mean','std', 'median'])
    final_trade = df.groupby(['time_id', 'stock_id']).agg(
    real_vol_trade=('trade_log_ret', realized_volatility)).reset_index()
    final_trade = final_trade.merge(df_trade,on=['stock_id','time_id'],
how='left')
    return final_trade

```

Listing 3.7: Code Correlation between the 6 features

We will apply to the models (LightGBM and XGBoost and CatBoost and Linear Regression) as shown in table 3.2 as follows:

Table 3.2: results with optimised parameters for models

model/metric	r2-score	RMSPE
LightGBM	**0.832**	**0.273**
XGBoost	0.8	0.297
CatBoost	0.832	0.273
Linear Regression	0.785	0.313

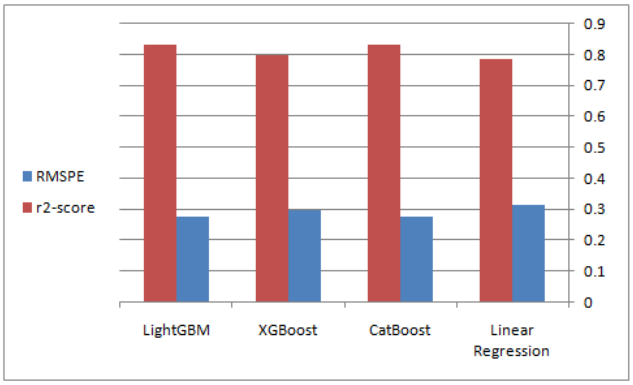

Figure 3.10: Histogram Comparison optimised parameters for models

We note through the table 3.2, that all models improved, but the best improved score was in the LightGBM model and the CatBoost model, where the RMSPE score of 0.273 and the highest score in r2 was 0.832.

3.7.3 Feature importance analysis:

The result of feature importance analysis are shown in bellow Figure 3.11 . Firstly, we apply feature engineering data with 56 features on LightGBM. which has RMSPE score that is 0.273 and a highest r2-score is 0.832. Then, we perform feature importance analysis. As shown in bellow figure, the five most important features are time-id, real-vol-trade,real-vol-1,stock-id,real-vol-2, Further, we observe that almost all the five features are the same features is shown in figure (3.9). The following figure shows feature importance:

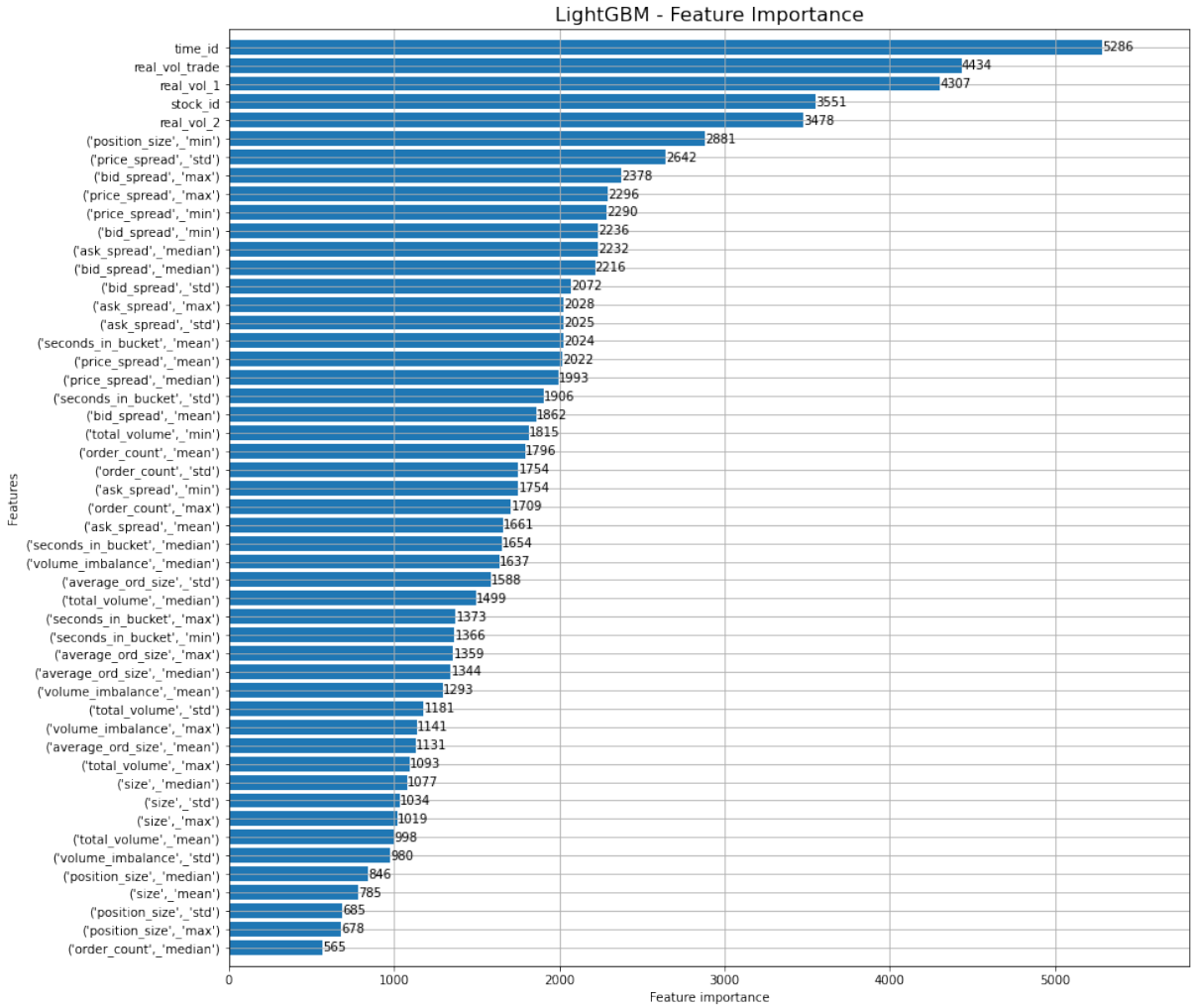

Figure 3.11: The feature importance ranking of LightGBM (56 features)

The code below is LightGBM - Feature Importance:

```
fig, ax = plt.subplots(figsize=(14,14))
lgb.plot_importance(model, max_num_features=50, height=0.8, ax=ax)
#ax.grid(False) Prints in grid format
plt.title("LightGBM - Feature Importance", fontsize=16)
```

```
    plt.grid()

```

Listing 3.8: Code LightGBM - Feature Importance

3.7.4 Method Using K-Fold Cross-Validation for models:

k-Fold introduces a new way of splitting the dataset which helps to overcome the "test only once". We will apply K-Fold Cross-Validation at four models. Pick a number of folds – k. Usually, k is 5 or 10 but you can choose any number which is less than the dataset's length. we will choose number os folds-k is 5, after applying this method, we obtained the following results, as shown in the following 3.3 table:

Table 3.3: results of K-Fold Cross-Validation for models

model/metric	RMSPE
LightGBM	**0.229**
XGBoost	0.287
CatBoost	0.272
Linear Regression	0.307

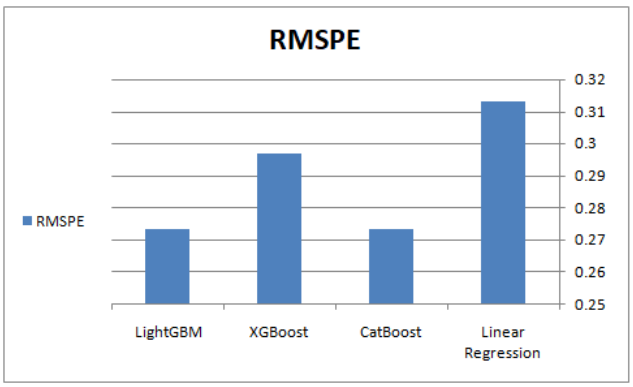

Figure 3.12: Histogram Comparison K-Fold Cross-Validation for models

We note through the table 3.3, that is the best result remained at lightGBM model, where, RMSPE score that is 0.229, much improvement compared to other models. We conclude that the best model is LightGBM. The code below explains LightGBM model:

```
    n_splits = 5
    import lightgbm as lgb
    from sklearn.model_selection import KFold
    kf = KFold(n_splits=n_splits, random_state=42, shuffle=True)
```

```
    for fold, (trn_idx, val_idx) in enumerate(kf.split(X)):
    print(f"Fold : {fold + 1}")
    X_train, y_train,X_test, y_test = X.iloc[trn_idx], y.iloc[trn_idx],
X.iloc[val_idx], y.iloc[val_idx]
    lgb_params = {'task': 'train',
      'boosting_type': 'gbdt',
      'learning_rate': 0.01,
      'objective': 'regression',
      'metric': 'rmse',
      'max_depth': -1,
      'n_jobs': -1,
      'feature_fraction': 0.8,
      'bagging_fraction': 0.8,
      'lambda_l2': 1,
      'verbose': -1
      #'bagging_freq': 5
    }
    cats=["stock_id"]
    lgb_train=lgb.Dataset(X_train,label=y_train, categorical_feature=
cats, weight=1/np.power(y_train,2))
    lgb_val=lgb.Dataset(X_test,label=y_test, categorical_feature=cats,
weight=1/np.power(y_test,2))
    # training
    model = lgb.train(lgb_params,
    train_set=lgb_train,
    valid_sets=lgb_val,
    num_boost_round=5000,
    verbose_eval=250,
    early_stopping_rounds=500,
    )
    # validation
    y_pred=model.predict(X_test)
    def rmspe(y_test, y_pred):
    return  (np.sqrt(np.mean(np.square((y_test - y_pred) / y_test))))
    RMSPE = round(rmspe(y_test , y_pred =y_pred),3)
    print(f'Performance of the lightGBM prediction: RMSPE: {RMSPE}')

```

Listing 3.9: Code K-Fold Cross-Validation for LightGBM

3.8 Conclusion

In this chapter, we have applied four models of machine learning about market volatility which we talked about in Chapter Two, we used dataset provided by the Kaggle platform. To predict market volatility we used four models we found that a model of lightGBM is a best model.

General Conclusion

Market volatility prediction is one of the most commonly used terms in the trading market today. Price movements, market volatility, and trading risks are all represented by realized volatility. A small change in volatility affects the expected return on all assets.

The goal of this thesis was to predict market volatility with machine learning models, compare the results and determine which approach is more efficient. In particular, our analysis covered the LightGBM , XGBoost , CatBoost and Linear Regression.

A preliminary theoretical discussion allowed us to focus our empirical research on the the LightGBM , XGBoost , CatBoost and Linear Regression. These were compared based on their forecasting accuracy of the realized volatility.

The prediction models that we used in our study are LightGBM, XGBoost, CatBoost and Linear Regression, and we have drawn some relevant work on volatility prediction. Then we ran our models, and the result showed that the LightGBM model has the lowest RM-SPE score of 0.286 and the highest r2 score of 0.817. The RMSPE for other models such as XGBoost, CatBoost, and Linear Regression are respectively 0.303, 0.302, and 0.347, and the r2-score is 0.791, 0.784, and 0.766. And we calculated our models using feature engineering, we noticed that the results improved. The best results were that the LightGBM model and the CatBoost model were the best. Next, we also applied K-Fold Cross-Validation in the four models. The LightGBM result was the best compared to the other models.

Through this study, we find that LightGBM is the best prediction model to predict realized volatility.

References

[1] A. Nõu, D. Lapitskaya, M. H. Eratalay, and R. Sharma, "Predicting stock return and volatility with machine learning and econometric models: a comparative case study of the baltic stock market," *Available at SSRN 3974770*, 2021.

[2] A. Bucci *et al.*, "Forecasting realized volatility: a review," *Journal of Advanced Studies in Finance (JASF)*, vol. 8, no. 16, pp. 94–138, 2017.

[3] R. J. Shiller, *Market volatility*. MIT press, 1992.

[4] S.-H. Poon, *A practical guide to forecasting financial market volatility*. John Wiley & Sons, 2005.

[5] S. Muzzioli, "The relation between implied and realised volatility: are call options more informative than put options? evidence from the dax index options market," 2007.

[6] S. Romano, "Machine learning for volatility forecasting," 2021.

[7] P. A. C. Luong, *Measuring and Modelling the Volatility of Financial Time Series*. PhD thesis, Curtin University, 2016.

[8] T. G. Andersen, T. Bollerslev, F. X. Diebold, and P. Labys, "Modeling and forecasting realized volatility," *Econometrica*, vol. 71, no. 2, pp. 579–625, 2003.

[9] T. G. Andersen, T. Bollerslev, F. X. Diebold, and H. Ebens, "The distribution of realized stock return volatility," *Journal of financial economics*, vol. 61, no. 1, pp. 43–76, 2001.

[10] E. Rahimikia, S. Zohren, and S.-H. Poon, "Realised volatility forecasting: Machine learning via financial word embedding," *Available at SSRN 3895272*, 2021.

[11] T. G. Andersen and L. Benzoni, "Realized volatility, working paper 2008-14," 2008.

[12] S.-w. He, J.-g. Wang, and J.-a. Yan, *Semimartingale theory and stochastic calculus*. Routledge, 2019.

[13] P. Padhi and I. Shaikh, "On the relationship of implied, realized and historical volatility: evidence from nse equity index options," *Journal of Business Economics and Management*, vol. 15, no. 5, pp. 915–934, 2014.

[14] P. Tankov, *Financial modelling with jump processes*. Chapman and Hall/CRC, 2003.

[15] J. D. MacBeth and L. J. Merville, "An empirical examination of the black-scholes call option pricing model," *The journal of finance*, vol. 34, no. 5, pp. 1173–1186, 1979.

[16] S. Arziyev, "Simulations of implied volatility and option pricing using neural networks and finite difference methods for heston model," 2020.

[17] S. Liu, C. W. Oosterlee, and S. M. Bohte, "Pricing options and computing implied volatilities using neural networks," *Risks*, vol. 7, no. 1, p. 16, 2019.

[18] T. M. Mitchell, "Artificial neural networks," *Machine learning*, vol. 45, pp. 81–127, 1997.

[19] T. M. Mitchell, "Does machine learning really work?," *AI magazine*, vol. 18, no. 3, pp. 11–11, 1997.

[20] B. He, K.-i. Shu, and H. Zhang, "Machine learning and data mining in diabetes diagnosis and treatment," in *IOP Conference Series: Materials Science and Engineering*, vol. 490, p. 042049, IOP Publishing, 2019.

[21] A. B. Tucker, *Computer science handbook*. Chapman and Hall/CRC, 2004.

[22] M. Islam, Z. Jannat, *et al.*, "Prediction of facebook addiction using machine learning," 2020.

[23] A. Ratner, S. Bach, P. Varma, and C. Ré, "Weak supervision: the new programming paradigm for machine learning," *Hazy Research. Available via https://dawn. cs. stanford. edu//2017/07/16/weak-supervision/. Accessed*, vol. 5, 2019.

[24] M. v. Otterlo and M. Wiering, "Reinforcement learning and markov decision processes," in *Reinforcement learning*, pp. 3–42, Springer, 2012.

[25] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

[26] M. Kinnander, "Predicting profitability of new customers using gradient boosting tree models: Evaluating the predictive capabilities of the xgboost, lightgbm and catboost algorithms," 2020.

[27] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in neural information processing systems*, vol. 30, 2017.

[28] "Feature engineering." `https://www.heavy.ai/technical-glossary/feature-engineering`. Accessed:05 04 2022 13:36:26 GMT.

[29] Y. Zhao and M. Khushi, "Wavelet denoised-resnet cnn and lightgbm method to predict forex rate of change," in *2020 International Conference on Data Mining Workshops (ICDMW)*, pp. 385–391, IEEE, 2020.