FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of master's thesis

Title:	Automated Data Pre-processing via Meta-learning
Student:	Bc. Maxim Sachok
Supervisor:	Ing. Stanislav Kuznetsov
Study program:	Informatics
Branch / specialization:	Software Engineering
Department:	Department of Software Engineering
Validity:	until the end of summer semester 2022/2023

Instructions

The main goal of this thesis is to use meta-learning methods for automated dataset pre-processing. We show various techniques to evaluate the quality of time-series datasets from a trading domain. We present specific scoring, filter and optimization methods and describe the benefits of each of them. The final solution could automatically decide which dataset is more proper for machine learning purposes.

1. Prepare state-of-the-art of meta-learning methods and automated data pre-processing solutions
2. Describe the principles and choose the set of methods
3. Choose different time-series datasets and prepare the data pool.
4. Implement a scoring, filtering, and optimization algorithm that gives a dataset's fitness score.
5. Presents the result of the solution.

Electronically approved by Ing. Michal Valenta, Ph.D. on 17 August 2022 in Prague.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master's thesis

Automated Data Pre-processing via Meta-learning

Bc. Maxim Sachok

Department of Software Engineering
Supervisor: Ing. Stanislav Kuznetsov

June 28, 2023

Acknowledgements

I wish to thank my family for supporting me in studying software engineering, my supervisor for helping me write this thesis and my friends for cheering me up.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act.

In Prague on June 28, 2023 . . . . . .

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Sachok, Maxim. *Automated Data Pre-processing via Meta-learning.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2023.

Abstrakt

V této práci budeme zkoumat možnosti strojového učení, vybírat a testovat různé metody předzpracování dat. Implementujeme skórovací algoritmus, abychom porovnali, jak dobře tyto metody fungují na různých úrovních znečištěných dat.

Klíčová slova Meta-learningový algoritmus, optimalizace dat, mazání dat, strojové učení

Abstract

In this thesis we will research machine learning capabilities, choose and test different data pre-processing methods. We weill implement a scoring algorithm to compare how well those methods perform on different levels of polluted data.

Keywords Meta-learning algorithm, data optimisation, data clearing, machine learning

Contents

List of Figures

List of Tables

Introduction

In recent years, the use of large amounts of data and utilization of machine learning, has underscored the significance of efficient data processing techniques. High-quality, well-structured datasets are pivotal to the successful application of machine learning models and the derivation of insightful conclusions. However, the manual pre-processing of these datasets, such as data cleaning, transformation, and integration, is labor-intensive and may introduce human bias. Thus, automated data pre-processing solutions are a compelling alternative, especially when coupled with meta-learning methods to tailor the pre-processing to the features of each dataset.

This thesis aims to explore and utilize meta-learning methods for automated dataset pre-processing in the trading domain. The focus will be on time-series datasets, given their ubiquitous presence and importance in this sector. By harnessing meta-learning strategies, we seek to evaluate the quality of these datasets, consequently providing an automatic, unbiased decision regarding the suitability of each dataset for machine learning applications.

The study will embark on a comprehensive review of the state-of-the-art meta-learning methods and automated data pre-processing solutions. A deep understanding of the principles of these methodologies will guide the selection of a set of techniques most apt for the task at hand. Simultaneously, a selection of diverse time-series datasets will be collected and prepared to provide a comprehensive data pool for the study.

Building on this, we will implement a scoring algorithm that will choose most optimal algorithm for the presented dataset. This algorithm will assess the 'fitness' of a dataset, giving an indication of its suitability for machine learning endeavors. By outlining the benefits of each method within this algorithm, we aim to provide a nuanced and detailed understanding of the dataset evaluation process.

Finally, we will present the results of our automated solution. In doing so, we aspire not only to demonstrate the validity and effectiveness of our approach but also to contribute to the broader discourse on the automation of data pre-processing. Through this research, we aim to pave the way for more accurate, efficient, and automated data handling, thereby maximizing the potential benefits and insights derived from machine learning in the trading domain.

Chapter 1

Machine learning

Machine learning is a branch of artificial intelligence that involves a set of algorithms that can ”learn” from data and make predictions or choices based on what they find. These algorithms work by building a mathematical model from sample data, called ”training data,” to make predictions or decisions without being explicitly programmed to do the job.

Machine learning is based on the idea that algorithms can learn from data and make choices based on what they have learned. This learning can happen with guidance, with only some supervision, or without any supervision at all. In supervised learning, a set of input-output pairs is used to map a function that can predict future outputs. In unsupervised learning, input data is used to find patterns without any answers being labeled. A mix of these two, semi-supervised learning uses a small amount of labeled data and a large amount of unlabeled data to teach a machine.

Machine learning is used more and more in a wide range of businesses because it can handle large and complex datasets and improve and change as it learns. In health care, it helps make decisions about diagnosis, predict how a disease will progress, and customize treatment plans. In business, it is a key part of predicting how stocks will move, finding fraud, and making the best use of a portfolio.

In the world of e-commerce, machine learning algorithms customize the user experience by recommending goods based on the user's browsing history and by looking at the patterns of what people buy. In the transportation and logistics business, machine learning helps with things like figuring out the best routes, predicting demand, and allocating resources. In the energy industry, it is used to predict demand, optimize energy use, and improve grid control.

In essence, machine learning is a way to learn and draw conclusions from data. It includes a wide range of tasks, such as classification, regression, grouping, anomaly detection, and reinforcement learning, but is not limited to them. As the world becomes more and more data-driven, machine learning becomes more and more important for getting useful information from data. This puts it at the forefront of modern technology and study.

1.1 Why machine learning

Traditional algorithms and machine learning models work and solve problems in very different ways. Traditional algorithms solve a job by following steps and rules that have already been set up. This means that explicit programming is needed for every problem domain. Machine learning models, on the other hand, are made to ”learn” from data and apply what they’ve learned to similar tasks. They adapt their behavior based on the data they are taught on.

Machine learning models work best in situations where it’s hard to program clear rules. This includes environments with a lot of dimensions, a lot of complexity, and situations where humans can’t figure out how to solve problems with their instincts. For example, tasks like speech recognition, image recognition, and natural language processing, which are easy for people but hard to describe with explicit programming rules, can be done better with machine learning models.

Machine learning models also have the benefit of being able to change over time, as they can get better as they see more data. This trait is especially useful in dynamic settings where data patterns can change, like predicting the stock market or analyzing how users act in digital marketing.

Machine learning models are also good because they can deal with noisy and imperfect data. In the real world, data is often noisy, irregular, or missing. Machine learning algorithms are made to handle these kinds of flaws by being able to find useful patterns and avoid noise that isn’t useful.

Also, one of the best things about machine learning models is that they can find hidden trends and correlations in data. For example they are very effective against viruses and malware which are evolving every single day and are hard to detect [2]. In big and complicated datasets, these patterns and correlations may not be easy to see or understand, but machine learning algorithms can find them. Such hidden information can be very helpful in many fields, from medicine to business and climate science.

Conventional algorithms, also called rule-based algorithms, work by following a set of clear instructions to do a job. They use a method called ”determinism,” which means that if we give them the same input, they’ll always give the same result. This makes them reliable and easy to
understand, but it also makes it hard to handle jobs that involve complexity, variation, and uncertainty, which are common in real-world data.

One big problem is that it’s hard to define rules clearly for complicated jobs. In areas like natural language processing, image recognition, and voice recognition, it is

hard, if not impossible, to make a complete set of rules that can cover the variety and complexity of human language, different visual scenes, and different accents and speech patterns.

Rule-based systems also have a problem because they are fixed. Once a standard algorithm has been created and put into action, it doesn't change or adapt to new data unless it is reprogrammed. Machine learning algorithms, on the other hand, can learn from new data and get better on their own. This makes them more adaptable to changing surroundings and trends.

Also, traditional algorithms have trouble with high-dimensional data, which has a large number of properties. Such situations often lead to the problem of scalability, which is when the cost of computing becomes too high or an algorithm can't find useful patterns because high-dimensional data is sparse and spread out. Machine learning algorithms, especially those that are based on neural networks, are great at handling high-dimensional data and finding trends in it.

Also, traditional algorithms don't know how to deal with confusing or imperfect data. They are often sensitive to small changes or mistakes in the data they receive, which can cause them to stop working. Machine learning systems, on the other hand, are usually more resistant to noise and can learn to ignore changes in the data that don't matter.

But it's important to remember that machine learning isn't a magic bullet. It has its own problems, like being hard to understand and not being fair. Whether to use a rule-based method or a machine learning model should depend on the type of problem, the quality and amount of data available, the need for interpretability, and the amount of computing power available. As our knowledge and technology grow, mixed methods that take the best parts of both rule-based and learning-based methods could also be a good way forward.

Even though machine learning models have the benefits listed above, it's important to note that they are not always better than standard algorithms. Whether to use a machine learning model or a traditional algorithm relies on the problem at hand, the quality and amount of data available, how easy it needs to be to understand, and how much computing power is available. Still, the unique features of machine learning models make them a powerful tool in the data-driven decision-making model used in modern study and business.

1.2 Dangers of machine learning

Machine learning is a powerful set of tools for getting insights and making predictions from big, complex datasets. However, it is not without its own problems and risks. To use machine learning in a responsible and well-informed way, we need to know about these risks. This will help us avoid getting wrong results and make sure that decisions are made in an ethical and reliable way.

The problem of bias is one of the most important things to think about when using machine learning models. Models learn from data, and if the data used to train them is biased, it's possible that the models will keep or even increase these biases. Machine

learning can be biased in many ways, such as through sampling bias, latent social bias in the data, or algorithmic bias. These biases can lead to unfair or unfairly discriminatory results, which is especially worrying in sensitive areas like jobs, lending, criminal justice, and healthcare.

Another big problem is that machine learning models are often hard to understand and explain, especially when they are complicated like deep neural networks. Because these models are hard to see into, they are often called "black boxes." This lack of openness can hurt trust and adoption, especially in high-stakes areas where it's important to know why a choice was made.

Data protection is also a big worry. Machine learning models can accidentally show private information in their training data, which could lead to a privacy breach. Also, models can be attacked by adversarial attacks, which are small, often undetectable changes to the input that cause the model to give wrong results.

Lastly, people worry about how reliable and strong machine learning models are. Small changes to the input data or the working environment can have a big effect on the model's output, which could lead to decisions that are wrong or harmful. This is especially important for systems that need to be safe, like self-driving cars or health care apps.

Machine learning has the ability to change the world, but for it to be used in a responsible way, these risks need to be understood and dealt with. Using techniques like reducing bias, making models easy to understand, different levels of privacy, robustness checks, and careful validation can help make sure that machine learning is a reliable and responsible tool for making decisions.

1.2.1 Bias

Bias in machine learning is a critical issue that can significantly impact the performance and fairness of predictive models. In machine learning, bias refers to the tendency of a model to make systematic errors based on certain attributes within the dataset, leading to unfair outcomes or predictions.

> In October 2019, researchers found that an algorithm used on more than 200 million people in US hospitals to predict which patients would likely need extra medical care heavily favored white patients over black patients. While race itself wasn't a variable used in this algorithm, another variable highly correlated to race was, which was healthcare cost history. The rationale was that cost summarizes how many healthcare needs a particular person has. For various reasons, black patients incurred lower health-care costs than white patients with the same conditions on average[3].

The sources of bias in machine learning are manifold. They can stem from the data, the way models are designed and trained, or the way they are deployed in the real world. Data bias is perhaps the most prevalent and can arise from imbalanced representation, historical prejudice, or non-random sampling, among other factors. For example, if a

facial recognition system is trained primarily on images of people with light skin tones, it will likely perform poorly when attempting to recognize individuals with darker skin tones.

These biases can have far-reaching implications when machine learning models are used in real-world, high-stakes decision-making scenarios. In hiring, if a predictive model is trained on historical hiring data that is biased against certain gender or racial groups, the model may perpetuate or even amplify this discrimination, unfairly disadvantaging individuals from these groups. Similarly, in criminal justice, if a predictive policing model is trained on historically biased arrest data, it could result in over-policing of certain communities.

Biased algorithms can also propagate social inequality. For instance, in credit scoring, if the data used to train a model contains bias against certain socio-economic classes, individuals from these classes could be unfairly denied loans or charged higher interest rates. In healthcare, biased algorithms could lead to misdiagnosis or inadequate treatment for certain demographic groups.

Mitigating bias in machine learning is a complex task that involves careful data collection and pre-processing, thoughtful feature selection, and appropriate model design. Various techniques such as re-balancing the dataset, using fairness-aware algorithms, and post-hoc fairness correction methods can be used to reduce bias.

Furthermore, ensuring diversity among those who design and deploy these models can also contribute to reducing bias. Regular audits and continuous monitoring of deployed models to assess their fairness and impact on different demographic groups can help detect and rectify bias.

It's crucial to recognize and address the potential biases in these systems to ensure they deliver fair outcomes. As the use of machine learning in decision-making expands, so too does the responsibility to ensure these tools promote equity and do not reinforce existing inequalities.

1.2.2 Black box

In the field of machine learning, a "black box" is a model that makes predictions based on the data it is given, but the process behind how it makes these predictions is hard to understand. This trait is often found in complex models, like deep neural networks, where thousands or millions of parameters are learned and changes within the model are highly non-linear and intertwined.

It's very important to understand how machine learning methods work, which is called "model interpretability," for a number of reasons. First, the models must be able to be understood in order to be validated and checked. Before we can believe a model's predictions, we need to know and make sure that the model is making decisions based on good criteria and not just on random correlations in the data.

Second, we need to know how a model makes decisions in order to find mistakes or biases in its estimates and fix them. If a model gives skewed or unfair results, we need to figure out why so that we can fix the problem.

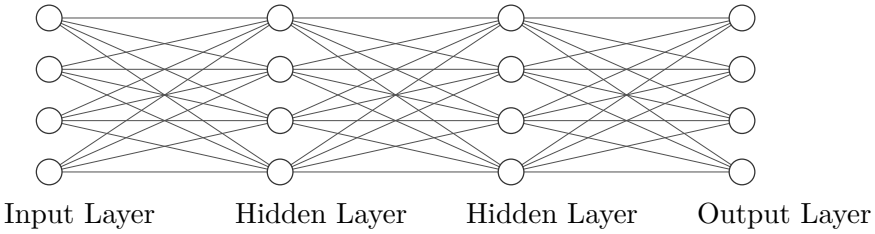

Figure 1.1: Diagram of neural network with hidden layers.

Third, being able to understand how machine learning models work can make it easier for people to learn from them. If we can figure out how a model makes choices, we might be able to learn from it and use what we've learned in other situations.

But the 'black box' structure of some machine learning models can make it hard to use them in some situations. This is especially true in areas where being easy to understand is needed by law or ethics. For example, if a model is used to identify a disease or suggest a treatment, doctors and patients need to know how the model came to its conclusions before they can trust it and follow its advice.

In the same way, regulations often require that choices that affect customers in finance or credit scoring be able to be explained. If a buyer is turned down for a loan, the lending company should be able to tell them why. In this situation, it wouldn't make sense to use a "black box" model.

Also, in safety-critical applications like self-driving cars, not being able to understand and interpret the choices made by "black box" models is a big risk. If an accident happens, it's important to figure out why and how the model made the choices that led to it so that similar accidents don't happen again.

So, even though "black box" machine learning models have done well at making predictions in a wide range of tasks, their inability to be understood is a big problem. When designing and deploying machine learning systems, it's important to find a balance between how well the model works and how easy it is to understand.

1.2.3 Training Data

Machine learning algorithms get their ability to predict from the data they are trained on. This means that the quality of the data has a huge effect on how well the model works. When machine learning models are trained on data that is missing, broken, or noisy, it can cause big problems with how they work and how useful they are.

Bias can be introduced into the learning process by missing or incomplete data, which changes how the model sees the spread of the underlying data. The bias can have different effects depending on the pattern of the missingness, which could be totally random, random, or not random. In the worst cases, the model might not learn the necessary relationships between the input variables and the goal variable, which means it won't be able to make good predictions. There are many ways to deal with missing

data, such as imputation, but all of them have their own limits and assumptions that need to be carefully thought through.

Data that is corrupted, mislabeled, or displayed in an inconsistent way can also hurt the model's ability to learn. For example, in supervised learning, mislabeled examples can lead the model astray, especially if the model is complicated and can account for noise in the data. This can cause the model to learn wrong links between the input variables and the goal variable, which can lead to bad predictions.

Noisy data, which is data that has random changes or mistakes, is another big problem. Machine learning algorithms are usually made to be able to handle a certain amount of noise, but too much noise can make it harder to find trends in the data, which can lower the accuracy of the model's predictions. Also, if the noise in the data is consistent or linked to certain factors, it can cause the model to learn false correlations that don't exist in the real data.

These problems with the quality of the data can also cause overfitting, which is when the model learns the noise or oddities in the data instead of the underlying trends. Overfitted models don't work well with new, untested data, which limits their use in the real world.

Also, models that are based on bad data can lead to unfair or unfairly biased results. For example, if some groups are underrepresented in the data, the model might not work well for people from these groups, which could lead to choices that aren't fair.

So, it is very important to make sure that the data used to train machine learning models is accurate. Data preprocessing and cleaning, exploratory data analysis, feature selection, robust learning methods, model validation, and performance tracking can help find and reduce the effects of incomplete, damaged, or noisy data. "Garbage in, garbage out," as the saying goes, shows how important good quality data is for machine learning systems to work.

Chapter 2

Data preparation

Data pre-processing is an important step in the machine learning process, and its importance is especially clear when time-series data are involved. Time-series data is a list of data pieces that are ordered by time and often depend on each other in time. Because of its unique structure and the possibility of trends, patterns, and noise that changes over time, time-series data needs special pre-processing methods before it can be used for machine learning tasks.

The workflow for pre-processing time-series data include several steps, such as:

- Data Collection: Gather data related to the trading domain. This could include historical prices, volume data, or other financial indicators such as P/E ratios or debt/equity ratios. Without data, there could be no pre-processing. We will be using dataset from yahoo stocks[4].

- Handling Missing Values: Financial datasets often have missing values due to market closures on weekends and holidays, or sometimes because of data recording errors. We'll need to decide how to handle these. Common strategies include forward fill, backward fill, linear interpolation, or using a statistical method to estimate missing values.

  Benefits of Handling Missing Values in Time Series Dataset:

  - **Preservation of Data Integrity:** By handling missing values appropriately, we can ensure the integrity of the time series dataset. Filling in or imputing missing values allows for a more complete representation of the underlying data and helps maintain the overall structure and patterns present in the time series.
  - **Accurate Analysis and Modeling:** Handling missing values enables more accurate analysis and modeling of the time series data. Missing values can introduce bias and distort statistical measures, making it difficult to derive meaningful insights. By addressing missing values, we can improve the ac-

curacy and reliability of statistical analyses, forecasting, and other modeling techniques.

- **Better Decision-Making:** A time series dataset with missing values can lead to incomplete and unreliable information for decision-making. By properly handling missing values, we can obtain a more comprehensive view of the data, facilitating better-informed decisions and actions.
- **Increased Data Usability:** Handling missing values improves the usability of the time series dataset. With a reduced number of missing values, the dataset becomes more suitable for various data-driven applications, including machine learning, predictive analytics, and data visualization.
- **Improved Performance of Models:** When missing values are handled appropriately, the performance of models trained on the time series data can improve. Filling in missing values or using advanced imputation techniques can enhance the accuracy and robustness of models, leading to better predictions and insights.

Disadvantages of Handling Missing Values in Time Series Dataset:

- **Imputation Uncertainty:** Handling missing values involves making assumptions and imputing values in place of the missing data. This introduces uncertainty as the imputed values may not accurately represent the true values that were missing. The imputation process can potentially introduce noise and impact the reliability of subsequent analyses and models.
- **Potential Bias:** The handling of missing values, if not performed carefully, can introduce bias into the time series dataset. The choice of imputation method and the assumptions made during the imputation process can inadvertently influence the results and conclusions drawn from the data, leading to biased interpretations.
- **Increased Complexity:** Dealing with missing values in a time series dataset adds complexity to the data preprocessing and analysis tasks. Different imputation methods, such as mean imputation, regression imputation, or time series-specific imputation techniques, may require additional computational resources, specialized knowledge, and careful implementation.
- **Loss of Information:** In some cases, the missing values themselves may carry valuable information or represent meaningful patterns in the time series. By imputing or filling in missing values, there is a potential risk of losing important information or obscuring underlying patterns that may be relevant for analysis or decision-making.
- **Assumption of Missingness Mechanism:** Handling missing values often assumes a certain mechanism behind the missingness, such as missing completely at random (MCAR), missing at random (MAR), or missing not at random (MNAR). Making assumptions about the missingness mechanism

may not always hold true and can introduce bias or inaccuracies in the imputation process.

- Time-based Ordering: For time series data, it is crucial to maintain chronological order because previous events can have an impact on future events.
- Outliers Detection & Handling: Outliers, or extreme values that deviate from other observations, can be caused by extraordinary events such as market crashes or financial crises. These outliers could potentially skew the model's predictions, and we need to decide how to handle them, either by capping, transforming, or removing them.

  Benefits of Outlier Detection:

  - **Anomaly Detection:** Detecting outliers can help identify anomalies in the data, which could be indicative of a problem or a special condition. For example, in a production process, an outlier could be a sign of a defect or problem.
  - **Improved Accuracy:** Removing outliers can improve the accuracy of our data models. Outliers can skew the model and lead to inaccurate predictions.
  - **Fraud Detection:** Outlier detection can be used to detect fraudulent transactions in financial data. Fraudulent transactions are typically significantly different from normal transactions.

  Downsides of Outlier Detection:

  - **False Positives:** One of the main challenges with outlier detection is dealing with false positives. An algorithm may incorrectly identify a data point as an outlier when it is not.
  - **Complexity:** Time-series data often exhibits complex patterns and trends, which can make it difficult to correctly identify outliers. Some outlier detection methods may not be able to handle these complexities.
  - **Resource Intensive:** Depending on the method used, outlier detection can be computationally expensive and time-consuming, particularly for large datasets.
  - **Subjectivity:** Defining what constitutes an outlier can be subjective and may depend on the specific context and domain knowledge. This can make it challenging to define a one-size-fits-all solution for outlier detection.

- Feature Engineering: Create new variables that can help in making better predictions. For instance, technical indicators like moving averages, RSI (Relative Strength Index), MACD (Moving Average Convergence Divergence), etc., can be generated from price and volume data.

  Benefits of Feature Engineering:

- **Improved Predictive Power:** Feature engineering allows the creation of new variables that capture important patterns and relationships in the data. By incorporating domain knowledge and creating informative features such as technical indicators (moving averages, RSI, MACD), the predictive models can better capture the underlying dynamics of the price and volume data, leading to improved prediction accuracy.
- **Enhanced Model Interpretability:** Feature engineering can introduce new variables that have intuitive interpretations. For example, technical indicators like moving averages or RSI provide insights into market trends and momentum. By including such features, the resulting models become more interpretable, allowing stakeholders to understand the factors driving the predictions.
- **Capturing Nonlinear Relationships:** Creating new variables based on price and volume data enables the modeling of complex, nonlinear relationships. Technical indicators can capture nonlinear patterns and behaviors that may not be apparent in the raw data. This can help the predictive models uncover hidden patterns and improve their ability to capture nonlinear relationships between variables.
- **Handling Missing or Noisy Data:** Feature engineering techniques can help mitigate the impact of missing or noisy data. By generating new features from multiple data sources or by applying smoothing or filtering techniques, it is possible to create more robust features that are less sensitive to data quality issues. This can lead to more reliable and stable predictions.

Disadvantages of Feature Engineering:

- **Increased Dimensionality:** Feature engineering can lead to an increase in the number of variables in the dataset. If not handled carefully, this can result in the curse of dimensionality, where the model's performance deteriorates due to a higher ratio of variables to observations. It may require additional computational resources and increase the complexity of the modeling process.
- **Information Leakage:** In feature engineering, there is a risk of unintentionally introducing information leakage, where the created features incorporate knowledge of the target variable that is not available in real-world scenarios. This can lead to overly optimistic model performance during training but poor generalization to new, unseen data.
- **Overfitting:** Introducing a large number of engineered features increases the risk of overfitting, where the model becomes overly specialized to the training data and fails to generalize well to unseen data. Careful feature selection and regularization techniques are necessary to mitigate the risk of overfitting and ensure the model's robustness.
- **Domain Expertise Requirement:** Effective feature engineering often relies on domain expertise and a deep understanding of the data and problem

domain. It may require subject matter experts or experienced data scientists to identify relevant features, select appropriate transformations, and design effective feature extraction techniques. This can pose challenges if domain expertise is limited or not readily available.

- **Increased Computational Complexity:** Feature engineering can significantly increase the computational complexity and time required for model training. Generating new variables, especially complex ones like technical indicators, may involve iterative calculations or complex algorithms. This can impact the scalability and efficiency of the modeling process, particularly when dealing with large datasets.

- Normalization/Standardization: Machine learning algorithms perform better when input numerical variables fall within a similar scale.

  Benefits of Normalization/Standardization of Data for Time Series Dataset:

  - **Comparable Scale:** Normalization/standardization rescales the values in the time series dataset, making them comparable across different variables or features. This is particularly useful when the variables have different units or scales, allowing for fair comparisons and analysis.
  - **Improved Model Performance:** Normalizing/standardizing the data can improve the performance of certain machine learning algorithms, such as those based on distance calculations or gradient-based optimization. By bringing the data to a similar scale, it can prevent certain variables from dominating the learning process, leading to more balanced and accurate models.
  - **Stabilized Variance:** Standardization scales the data to have zero mean and unit variance, which can help stabilize the variance across variables. This is particularly beneficial when working with algorithms that assume the data to be normally distributed or have equal variances, leading to more reliable and valid statistical analysis.
  - **Interpretability and Visualization:** Normalization/standardization can improve the interpretability and visualization of the time series data. By bringing the variables to a common scale, it becomes easier to compare their relative magnitudes and understand their contributions to the overall pattern. This aids in data exploration, pattern recognition, and communicating insights to stakeholders.

  Disadvantages of Normalization/Standardization of Data for Time Series Dataset:

  - **Loss of Original Scale and Interpretation:** Normalization/standardization transforms the data to a standardized scale, which can result in the loss of the original scale and interpretation. This can make it challenging to relate the transformed values back to the original units, limiting the direct interpretation of the transformed variables.

  - **Altered Distribution and Outlier Impact:** Normalization/standardization can alter the distribution of the variables, particularly when extreme outliers are present. The transformation may amplify or suppress the impact of outliers, affecting the statistical properties and assumptions of the data. It is essential to handle outliers appropriately before normalization/standardization to avoid distorting the data.
  - **Dependency on Entire Dataset:** The normalization/standardization process typically relies on summary statistics calculated from the entire dataset, such as mean and standard deviation. This means that any changes or updates to the dataset may require recalculating the summary statistics and reapplying the normalization/standardization, which can be cumbersome for large or continuously growing datasets.
  - **Impact on Time-dependent Patterns:** Normalization/standardization can potentially impact time-dependent patterns present in the time series data. Depending on the specific normalization method applied, it may introduce artificial patterns or modify the temporal relationships between variables, which can affect the accuracy and reliability of subsequent time-dependent analyses or forecasts.
  - **Algorithm Sensitivity:** While normalization/standardization can improve the performance of certain algorithms, it can also make them more sensitive to the specific scaling of the data. Some algorithms may require data to be strictly normalized or standardized to a specific range for optimal performance, requiring careful consideration and experimentation to identify the most suitable approach.

- Train/Test Split: Unlike random splitting in conventional ML tasks, time-series data requires careful handling. A common method is to use a chronological split, where the train set is from the earlier period and the test set is from the later period.

- Sequence Data Preparation: Because we are using LSTM to test our pre-processing, we will be splitting data into sequences, but this will not be present in the final dataset because, since there are a lot of machine learning algorithms with a lot of requirements, where some of them can not work with dataset split into sequences.

It's important to keep in mind that the steps and methods used in pre-processing can change based on the type of time-series data and the needs of the machine learning algorithm being used. Properly pre-processed time-series data can make machine learning models much better at predicting the future. This is why it is an important part of any time-series research pipeline.

2.1 Missing values

2.1.1 Mean

The way this method works is that it fills in the missing numbers with the mean (average) value of the feature or variable in question. To find the mean, add up all of the feature's values and divide by the number of values.
Benefits: It's a simple and quick way to deal with missing info. Also, using the mean doesn't change how spread out the data is.
Downsides: It doesn't work well with categorical data. Also, because the mean is sensitive to high values, it can lead to wrong conclusions if the data is skewed or has "outliers." Not very good for time series data, where closeness in time may be more important than the overall mean.

2.1.2 Middle

The way this method works is that it fills in missing numbers with the median (middle) value of the feature or variable in question.
Benefits: It can handle outliers well, so it's a better choice if our data has a lot of them.
Downsides: Like mean imputation, it isn't very good for time series data where there may be autocorrelation and the value is probably linked to the observations around it.

2.1.3 Most frequent

How it works: This method fills in missing numbers with the most common (mode) value in the feature or variable in question.
Benefits: It's helpful for putting in numbers for categorical features.
Downsides: If the mode doesn't match the missing values, it can cause bias. Like the mean and the median, it doesn't take into account how data in a time series changes over time.

2.1.4 KNN

This method fills in empty values by looking at the values of the k-nearest neighbors. The missing values of an observation are filled in using the mean value of the 'k' nearest neighbors found in the training set. Neighbors are found based on Euclidean distance.
Benefits: It can be more accurate than mean, median, or mode imputation because it doesn't just look at the central tendency but also at other observations that are related. It can work with both numbers and words, as long as they can be coded in the right way.
Downsides: It can be expensive to run, especially on big datasets. Also, picking the right 'k' value is not easy and may take more testing.

2.1.5 Iterative

How it works: This method describes each missing value feature as a function of other missing value features in a round-robin way. It is a multivariate imputer that estimates each trait from all the others.
Benefits:When there is a link between traits, it can help get better results.
Downsides: It takes more time to calculate than other ways. It can also make our data handling pipeline more complicated, which can make it harder to keep up with or fix bugs. It still doesn't take the time part of time series data into account.

2.2 Time-based Ordering

When working with time series data, it is very important to make sure that the data is in order based on time. The order of the data points can have a big effect on what we can learn and how well we can guess from it. We usually do this with a method called "time-based ordering."

In time-based ordering, the sort function is a key tool that lets us change the order of any items in the dataset that aren't in the right order based on when they happened. As the name suggests, time-based sorting involves putting the data points in the dataset in order by the timestamps that are attached to each point. This code looks at each data point in the dataset, compares the timestamps, and puts them in an order that makes sense in terms of time, either going up or down.

But it's important to keep in mind that the method used for time-based ordering doesn't change much, no matter what kind of data we're working with. The same rules apply, which are to compare the "values" (in this case, the timestamps) of each data point and put them in a certain order. The main goal is to keep the data accurate and consistent so that they can be analyzed later.

But time-based ordering could be a problem when it comes to processing very big datasets. Even though the sort function is effective, it could take a long time when there are a lot of data points. In these situations, the amount of computing work goes up, which could slow down the whole data processing and research pipeline. It could even slow down the computer if there aren't enough tools. So, when working with big data sets, it's important to think of efficient techniques or other ways to do the job more effectively.

2.3 Outliers Detection & Handling

The Isolation Forest is a method for machine learning that was made to find unusual things. It is based on the idea of finding outliers instead of putting together a profile of regular data points.

The Isolation Forest works on the idea that there aren't many anomalies and that they are different, so it's easy to find them and put them in a separate place. The algorithm separates data by picking a feature at random and then picking a split value

between the feature's highest and lowest values at random. This random partitioning makes the tracks for anomalies noticeably shorter, which leads to the basic idea that anomalies are more likely to be isolated.

Normality is measured by the path length, or the number of lines an observation must travel from the root of the tree to the last node. Anomalies usually have shorter paths because their data instances have attribute values that are very different from those of regular instances. This leads to earlier isolation. On average, the length of the path should be shorter for material that doesn't fit the norm. So, when a forest of random trees gives shorter path lengths for some samples as a whole, those samples are very likely to be outliers.

The Isolation Forest method has the following advantages:

Efficiency: The Isolation Forest algorithm takes a linear amount of time and has a low constant, which makes it very good for big datasets.

Scalability: The method is very scalable because the size of the subsample does not depend on the size of the dataset.

Performance: Compared to other famous methods, it does a great job of finding outliers in high-dimensional datasets.

No need to tune parameters ahead of time: It doesn't need to calculate distance or density, so there's no need to set an anomaly level ahead of time.

Even with all of these great benefits, the Isolation Forest method has some problems:

Binary Split: The Isolation Forest algorithm uses a method called "binary splitting," which may not work well with categorical factors.

Instability: It can be quite unstable because small changes can greatly change the length of the path, which can lead to different outcomes.

Because the criteria for splitting are picked at random, the algorithm's results are not always the same.

Lack of Explanation: Unlike some other methods for finding outliers, Isolation Forests don't make it clear why a certain point was picked as an outlier.

2.4 Feature Engineering

2.4.1 Moving Averages

: A moving average is an easy way to smooth out data that helps find underlying trends by taking the average of short-term changes. The Simple Moving Average (SMA) and the Exponential Moving Average (EMA) are two popular types. The first one figures out the average price of a set of prices over a certain number of time periods. The second one gives more weight to the most recent prices.

- **Advantages**: Moving averages smooth out the direction of a time series, making patterns easier to see and reducing noise. They are easy to figure out and understand, and they can be used as a starting point for more complicated signs.

- **Disadvantages**: But moving averages are always behind the real data, which can cause trading models to send late signals to enter or leave a trade. They also don't work in markets that don't have trends and stay in a certain area.

2.4.2 (RSI) Relative Strength Index

: The RSI is a momentum oscillator that uses a scale from 0 to 100 to measure how fast and how much prices change. The average gains and loses over a certain time period are used to figure it out.

- **Benefits**: RSI can be a very useful tool for spotting overbought or oversold situations, which can be signs that a trend is about to change. Its ability to show how momentum changes can add a useful and unique dimension to models.
- **Disadvantages**: However, in a market with a strong trend, the RSI may stay in "overbought" or "oversold" area for a long time, which could send false signals. It also tends to send out false signs that need to be checked again.

2.4.3 Moving Average Convergence and Divergence (MACD)

: MACD is a momentum indicator that follows trends. It is made by taking the difference between the 26-period EMA and the 12-period EMA. The 9-day exponential moving average of the MACD is also drawn as a warning line. This line tells traders when to buy or sell.

- **Benefits**: The MACD helps find changes in a stock's price trend's power, direction, momentum, and length. It can also let us know when it might be a good time to buy or sell.
- **Disadvantages**: However, MACD may not work well in volatile markets because its signs change often, which could lead to too much trading. Also, because it is a lagging sign, it might not be able to respond quickly enough to changes in the market.

2.5 Normalization

2.5.1 Z-score Normalization

: The Z-score normalization technique implements a type of standardization that assumes data follows a Gaussian distribution, and it scales them such that the distribution is centered around 0 with a standard deviation of 1. It is calculated as:

$$z = \frac{(x - \mu)}{\sigma} \tag{2.1}$$

where x is the feature, μ is its mean, and σ is its standard deviation.

Advantages: Z-score normalization maintains the shape of the original distribution and is less affected by small changes in the data. It is useful in optimization algorithms that require data with zero mean and unit variance, such as support vector machines (SVMs) and principal component analysis (PCA).

Disadvantages: Z-score normalization does not ensure a specific minimum and maximum for the data, making it less suitable for algorithms that require data within a certain range. Moreover, the presence of significant outliers can distort the scaling due to the influence of outliers on the calculation of the mean and standard deviation.

2.5.2 Min-Max Normalization

:

Min-Max normalization rescales the data to a fixed range, usually 0 to 1, or -1 to 1 if there are negative values in the input data, preserving the shape of the original distribution. The transformation is given by:

$$X_{std} = \frac{(X - X_{min})}{(X_{max} - X_{min})} \tag{2.2}$$

$$X_{scaled} = X_{std} \times (max - min) + min \tag{2.3}$$

where min, max are the feature range of the output data, and X_{min}, X_{max} are the minimum and maximum feature values of the input data respectively.

Advantages: Min-Max normalization ensures a specific range for the scaled data, which might be necessary for some algorithms. It is useful when the algorithm demands non-negative inputs, like in the case of neural networks.

Disadvantages: Min-Max normalization is sensitive to outliers. A single observation with an extreme value can affect the minimum or maximum, skewing the range for all other values.

2.5.3 Robust Normalization (RobustScaler)

:

The Robust normalization technique uses a similar method to the Min-Max normalization but uses interquartile range, instead of the min-max, to normalize data, making it robust to outliers. It centers and scales the data according to the median and the interquartile range:

$$X_{scaled} = \frac{(X - Q1(X))}{(Q3(X) - Q1(X))} \tag{2.4}$$

where $Q1(X)$ and $Q3(X)$ are the first and third quartiles of the input data, respectively.

Advantages: Robust normalization is robust to outliers, and it's useful when dealing with data where outliers carry important information.

Disadvantages: Similar to Z-score normalization, it does not ensure a particular range for the output data and might not be suitable for certain algorithms requiring data within a specific range.

Chapter 3

Realisation

3.1 Language

Creating an auto pre-processing application can be done in more than one computer language. R, Java, C++, and Python are just a few of these languages. Each has its own pros and cons that need to be carefully weighed before deciding on a specific language for app development.

- R: R was made just for statistical computing and making graphs. It has a lot of different tools for pre-processing data and machine learning.
  - **Advantages**: R makes it easy to work with data and see how it is organized. It has a large set of statistical functions, which makes it a great tool for analyzing data.
  - **Disadvantages**: R is not a general-purpose programming language; it is mostly used for statistical research. It's hard to learn, and applications built with R may have trouble running when they have to deal with very big datasets.
- **Java**: Java is a general-purpose language that is class-based, object-oriented, and made to have as few implementation dependencies as possible.
  - **Advantages**: Java has strong tools for data preprocessing, machine learning, and artificial intelligence. It also works well with big datasets and has good support for distributed computing.
  - **Disadvantages**: Because Java is so wordy, it can be harder to write and fix code than in other languages. It doesn't have as many tools for statistical analysis and data analysis as R or Python.
- **C++**: C++ is a general-purpose language that lets us manipulate memory at both a low level and a high level.

  - **Advantages**: C++ gives us a lot of power over how system resources are used. It can be faster and better than other languages, especially when the speed of computation is important.
  - **Disadvantages**: Because C++ is complicated and has a lower level of abstraction, it can be hard to build and manage applications. It doesn't have any built-in packages or tools for data analysis, which are very important for data pre-processing.

- **Python**: Python is a high-level language that can be used for many things. It has a simple syntax and a large number of tools like Pandas, NumPy, and Scikit-learn for pre-processing data.
  - **Advantages**: The simple structure of Python makes it simple to write and understand code. It has many tools for pre-processing data, machine learning, and displaying data. Python also has a strong community, which helps solve problems and learn new ways to do things.
  - **Disadvantages**: Even though Python has become faster over time, it is still slower than languages like C++ and Java. But this isn't usually a problem when pre-processing data because the job is IO-bound and the speed of the language isn't the bottleneck.

Because of these things, Python has been picked as the language for the auto pre-processing application. Python is easy to use, has a lot of tool support, and has a lot of community resources. Because Python's syntax and meaning are easy to understand, developers can focus more on fixing problems and less on the technical details of the language. Also, Python's many tools for data manipulation, analysis, and machine learning make it a great choice for making an application for automatic pre-processing.

3.2 Libraries

Scikit-learn, TensorFlow.Keras, and TA-Lib (Technical Analysis Library) are integral Python libraries for building an auto pre-processing program. They provide a wide range of functions and tools for data preprocessing, machine learning, and technical analysis respectively.

- **Scikit-learn**:

  Scikit-learn is a machine learning library in Python. It provides a selection of efficient tools for machine learning and statistical modeling, including classification, regression, clustering, and dimensionality reduction via a consistent interface in Python [5].

  The library is built upon the SciPy (Scientific Python) that must be installed before we can use scikit-learn. This stack includes NumPy, SciPy, Matplotlib, pandas, and SymPy.

For the auto pre-processing program, Scikit-learn is used primarily for its powerful tools for data pre-processing. It offers functions for normalization, handling missing data, feature extraction, and much more, thereby providing comprehensive capabilities required for pre-processing in a machine learning pipeline.

- **TensorFlow.Keras**:

  Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow. It was developed with a focus on enabling fast experimentation and being highly modular and composable [6].

  For the auto pre-processing program, TensorFlow.Keras is primarily used for its extensive suite of tools for data pre-processing for deep learning models. It provides functionality for one-hot encoding, image pre-processing, text pre-processing, and sequence pre-processing, thus enabling the transformation of raw input into a form suitable for neural network training.

- **TA-Lib (Technical Analysis Library)**:

  TA-Lib is widely used by trading software developers requiring to perform technical analysis of financial markets. It includes over 150 functions such as moving averages, oscillators, and candlestick pattern recognition [7].

  In the context of the auto pre-processing program, TA-Lib is utilized to provide technical analysis on time-series data. This library can help generate new features from existing time-based data, enhance the decision-making capabilities of subsequent machine learning models, and is particularly useful when the program is used for financial data analysis.

  Scikit-learn provides comprehensive data pre-processing tools, TensorFlow.Keras enables pre-processing specifically designed for deep learning models, and TA-Lib offers technical analysis on time-series data, enriching feature generation from existing data.

3.3 Algorithm for testing

Long Short-Term Memory (LSTM) is a type of architecture for Recurrent Neural Networks (RNNs) that was made to model temporal sequences and their long-range dependencies more closely than traditional RNNs.

RNNs have a problem with gradients that disappear or explode, which makes it hard for them to learn long-term relationships. LSTM units have something called a ”memory cell” that can store information for a long time. So, LSTM networks are better able to deal with problems like gradients that go away or explode and can learn from data where long-term relationships exist.

An LSTM cell is made up of three parts: an input gate, a forget gate, and an output gate. The input gate controls how much new data flows into the memory cell, the forget gate controls how much memory is cleared, and the output gate controls how much

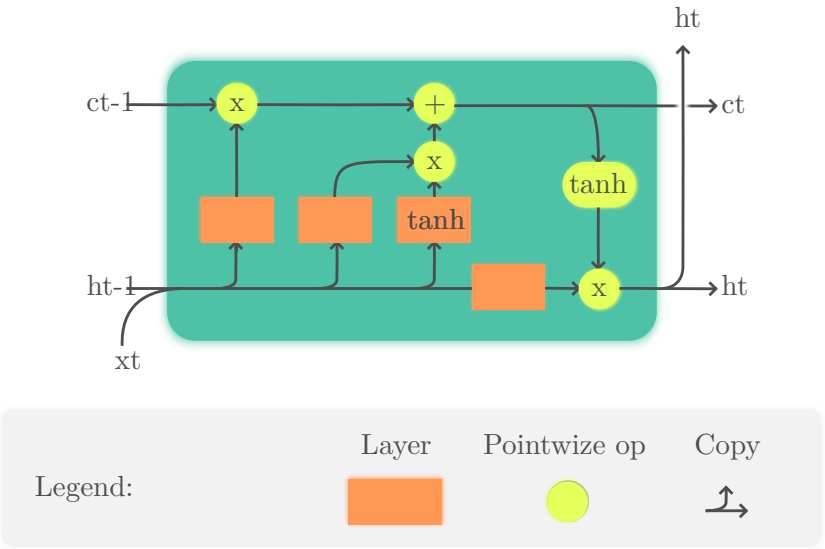

Figure 3.1: Long Short-term memory cell diagram [1]

information about the state of the cell flows out of the cell and into the rest of the network. When these gates work together, they control the flow of information into and out of the memory cell. This makes it possible for LSTMs to learn long-range relationships.

The *benefits* of LSTMs include their ability to find long-term dependencies in sequence data, handle big sequences of data, and model complex patterns. They have worked well in a wide range of situations, such as understanding natural language, recognizing speech, and predicting time series.

But there are also some *disadvantages* to LSTMs. They can be hard to train and take a long time, especially with long patterns. They also tend to fit the data too well, especially when the data set is small. Also, even though LSTMs can learn long-term dependencies, it can be hard to figure out how much information to include.

In conclusion, LSTM networks have their flaws, but they are a powerful way to model temporal data and have worked well in many uses that deal with sequential data.

3.4 Scoring

The scoring of a dataset can be performed by comparing the loss between a non-processed dataset and a processed dataset. In this context, the term ”loss” refers to a measure of how well a machine learning model (such as an LSTM algorithm) performs on a given task.

Typically, the non-processed dataset represents the original, raw data without any pre-processing steps applied. The processed dataset, on the other hand, refers to the data

that has undergone specific transformations or feature engineering techniques, which could include steps like normalization, scaling, or encoding categorical variables.

To obtain a score, the LSTM algorithm is trained and evaluated on both the non-processed and processed datasets. The loss values are then compared between the two. The lower the loss value, the better the performance of the model on the dataset. Thus, if the processed dataset results in a lower loss compared to the non-processed dataset, it can be considered as an improvement in the model's performance.

However, it's important to note that the results may vary due to the nature of LSTM algorithms. LSTM (Long Short-Term Memory) is a type of recurrent neural network (RNN) that can effectively handle sequential data and capture long-term dependencies. The LSTM algorithm's performance can be influenced by factors such as the architecture of the network, the quality and quantity of the training data.

Therefore, when using LSTM algorithms to acquire scores by comparing loss values between non-processed and processed datasets, it's essential to consider the inherent variability of the results.

Chapter 4

Testing

To get the most out of the Long Short-term Memory (LSTM) method, it is important to compare scores on scaled data. If we don't scale our data, the loss metric for unprocessed and processed datasets could be very different, which would make it hard to figure out how well other pre-processing methods work.

Data scaling is a key part of normalizing the characteristics of a dataset, especially when the factors have different scales or units. By using scaling methods like standardization or normalization, the data is changed so that it has a consistent and standardized range. This makes it easier to make fair comparisons and measure performance in a reliable way.

If we don't use data scaling techniques, we might come to wrong conclusions and give wrong estimates of how well different pre-processing methods work. The big differences in loss between the unprocessed and processed datasets would make it impossible to compare other pre-processing methods.

So, data scaling has to be thought of as an important part of the LSTM algorithm process. By using the right scaling methods, the data can be properly prepared to ensure fair comparisons, valid performance evaluations, and reliable conclusions. This is in line with the strict standards of scientific and research paper language.

4.1 Data pollution

The provided Python code showcases a function called `introduce_pollution` that processes a given CSV file to introduce pollution into the data. The pollution is manifested as missing values, spikes, and noise.

The code begins by reading the input CSV file into a DataFrame. It assumes the presence of a "Date" column, which is temporarily removed for pollution manipulation. Missing values are then introduced by randomly masking a fraction of the data using the `missing_prob` parameter.

Next, the code introduces spikes by randomly selecting data points using the `spike_prob` probability. The spike values are generated from a normal distribution with a specified

mean and standard deviation.

To introduce noise, the code calculates the noise magnitude based on the absolute value of each data point. The noise is then generated from a normal distribution with mean zero and a magnitude defined as a percentage of the absolute value.

Finally, the code adds back the date column to the DataFrame and saves two separate CSV files. One file contains the polluted data with less pollution, while the other contains the polluted data with more pollution. These files are specified by the `output_file_less_pollution` and `output_file_more_pollution` arguments, respectively.

```
import pandas as pd
import numpy as np

def introduce_pollution(csv_file, output_file_less_pollution,
output_file_more_pollution, missing_prob=0.1, spike_prob=0.1):
    # Read the CSV file into a DataFrame
    df = pd.read_csv(csv_file)

    # Remove the date column
    date_column = df['Date']
    df = df.drop(columns=['Date'])

    # Introduce missing values
    mask = np.random.choice([False, True],
    size=df.shape, p=[1-missing_prob, missing_prob])
    df_missing = df.mask(mask)

    # Introduce spikes
    spikes = np.random.choice([False, True],
    size=df.shape, p=[1-spike_prob, spike_prob])
    spike_values = np.random.normal(10, 5, size=df.shape)
    # introducing spikes with a mean of 10 and std of 5
    df_spikes = df_missing.copy()
    df_spikes = df_spikes.mask(spikes, spike_values)

    # Introduce noise based on value
    noise_std = df_spikes.abs() * 0.1
    # noise magnitude is 10% of the absolute value
    noise = np.random.normal(0, noise_std, size=df.shape)
    df_noisy = df_spikes + noise

    # Add back the date column
    df_noisy['Date'] = date_column
    df_spikes['Date'] = date_column
```

```
    # Save the DataFrame with less pollution to a CSV file
    df_spikes.to_csv(output_file_less_pollution, index=False)

    # Save the DataFrame with more pollution to a CSV file
    df_noisy.to_csv(output_file_more_pollution, index=False)

introduce_pollution('input_data.csv', 'less_pollution.csv',
'more_pollution.csv', missing_prob=0.2, spike_prob=0.2)
```

4.2 Base dataset

As we dig deeper into our data analysis process, it's important to remember that the results from our base information aren't always the same. Most of this changeability comes from the review model, which is called the Long Short-Term Memory (LSTM) model. LSTMs, which are a type of Recurrent Neural Network (RNN), are great at dealing with sequential data because they can find and learn trends over time. This gives us a deeper understanding of the information contained in our dataset.

Because these LSTM models have a built-in memory function that takes into account the history of the data, their results can sometimes be different from one another. The value and accuracy of the model's forecasts are not hurt by this change, though. It just shows how well the model can change and learn from the constantly changing patterns and trends in the sequential data it looks at.

On average, we've seen that the most significant changes to our base dataset don't come from random changes, but rather from adding new features. This fact shows how important feature engineering is for improving the success of machine learning models. Feature engineering is a key part of machine learning. It includes making new input variables out of existing ones. This lets the model find complex patterns that would have stayed hidden otherwise.

At the same time, it's important to stress how important data growth is to our method. Without scaling, the model's loss value tends to be very high, which hurts its ability to predict and its performance as a whole. Compared to the original data, the adjusted data leads to a much smaller loss, which makes the model work better.

Scaling is a basic step in the preprocessing process that includes changing the data so that it fits into a certain range, usually between 0 and 1. This is very important to make sure that the different parts of the model add about the same amount to the prediction. For example, if one feature runs from 0 to 1000 and another from 0 to 1, the model might give too much weight to the larger feature, which would lead to an unfair and unbalanced prediction. Scaling prevents this problem by making sure that all traits are given the same weight.

So, as we've already said, scale is not a nice-to-have extra in our method; it's an essential part of the process. The way it changes the LSTM model's results shows how

important good data preprocessing and feature engineering are in the field of machine learning.

Dataset is the last 5 years of Netflix stock history from yahoo.

After running the program on the base dataset:

Metric	After Processing	Before Processing	Loss Decrease
Mean Squared Error	0.00022	0.00014	59.75%

Table 4.1: Loss comparison, both datasets are scaled

Loss increased when comparing raw dataset to the processed one, and it is because of pre-processing performed on the dataset and because the unprocessed dataset is already mostly the best dataset you can ask for. The only way to enhance it is to add more features, but in this case generating features from existing ones didn't help. Used methods: MinMaxScaler SimpleImputer (most frequent) , IsolationForest (contamination=0.05), RSI 15.

4.3 Low pollution dataset

In Testing chapter, we carefully looked over the code that dealt with the Yahoo-obtained raw dataset of Netflix stock history. Our goal was to see how well the code could make two new datasets based on the first one. We added missing values, noise, and different spikes on purpose to the dataset to mimic real-world situations and make sure that it was robust. This intentional mixing up and messing up of the data let us figure out how well the pre-processing methods worked.

By making these controlled changes to the dataset, we tried to simulate the problems that often come up when analyzing data in the real world. For example, missing numbers can be caused by technical problems, mistakes in collecting data, or other things that can't be predicted. Noise, on the other hand, can come from outside sources like market changes or events that are hard to predict, while spikes can be caused by sudden changes in how investors feel or by news that was not anticipated.

Through our testing, we tried to find out if the code could handle such complexities and correctly pre-process the data, making sure that the datasets that were made were always reliable and consistent. By looking at how well the code made new datasets out of the contaminated and polluted data, we were able to figure out how reliable the pre-processing methods were. Used methods: MinMaxScaler,SimpleImputer (mean), IsolationForest (contamination=0.05), RSI with window size 25.

Metric	After Processing	Before Processing	Loss Decrease
Mean Squared Error	0.03515	0.07673	54.18%

Table 4.2: Loss comparison, both datasets are scaled.

4.4 High pollution dataset

Used methods: MinMaxScaler SimpleImputer (median) , IsolationForest (contamination=0.05) RSI 30 ordering by date column Again, as in datasets before, without scaling,

Metric	After Processing	Before Processing	Loss Decrease
Mean Squared Error	0.02271	0.04326	47.51%

Table 4.3: Loss comparison, both datasets are scaled.

removing null values, running a scoring algorithm would be impossible, because of the nature of the algorithm used.

We can see that with high polution we can clearly see that data pre-processing makes a huge difference on the accuracy.

Conclusion

After careful research into machine learning, problems connected with it, biases, importance of clear data, low probability of it in the real world, we identified methods used to enhance data, make it more suitable for the machine learning algorithms. We compared advantages and disadvantages of different algorithms for data imputance, outlier handling, feature generation, normalization-scaling. After this we identified an algorithm that can help us test our chosen methods for data pre-processing. We implemented a scoring mechanism based on the mean square root error and compared the results of different stages of data with different pollution levels: base dataset with no pollution, low pollution, high pollution. After reviewing results we can say that data pre-processing is very important for machine learning and directly affects the end result. Data with high pollution improved by 47.51%, data with low pollution improved by 54.18%, base data was not improved, and this is mostly because of the stochastic nature of lstm algorithm and that initial non-polluted dataset didn't have any noise nor missing values. For future work we could implement another algorithm that would try different combinations of hyperparameters of data pre-processing algorithms that would further enhance the quality of the dataset being produced.

Bibliography

[1] Wikimedia Commons. The LSTM Cell. Available from: https://commons.wikimedia.org/wiki/File:The_LSTM_Cell.svg

[2] Ijaz, M.; Durad, M. H.; et al. Static and Dynamic Malware Analysis Using Machine Learning. In *2019 16th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, 2019, pp. 687–691, doi:10.1109/IBCAST.2019.8667136.

[3] Shin, T. Real-life Examples of Discriminating Artificial Intelligence. 2020. Available from: https://towardsdatascience.com/real-life-examples-of-discriminating-artificial-intelligence-cae395a90070

[4] Finance yahoo historical data. 2023. Available from: https://finance.yahoo.com/quote/NFLX/history?p=NFLX

[5] Pedregosa, F.; Varoquaux, G.; et al. Scikit-learn: Machine Learning in Python. 2011, 1201.0490.

[6] Chollet, F.; et al. Keras: The Python Deep Learning API. https://keras.io, 2015.

[7] TA-Lib.org. TA-Lib : Technical Analysis Library. https://www.ta-lib.org, 2007.

Appendix A

Acronyms

LSTM Long Short-term memory

ML Machine learning

KNN K nearest neighbours

MCAR Missing completely at random

MAR Missing at random

MNAR Missing not at random

RSI elative Strength Index

MACD Moving Average Convergence Divergence

Appendix B

Contents of enclosed CD

- `readme.txt` ..... the file with CD contents description
- `src` ..... the directory of source codes
  - `wbdcm` ..... implementation sources
  - `thesis` ..... the directory of LaTeX source codes of the thesis
- `text` ..... the thesis text directory
  - `DP_Sachok_Maxim_2023.pdf` ..... the thesis text in PDF format