

Technical University of Košice
Faculty of Mining, Ecology, Process Control
and Geotechnologies

Advanced prediction models for sales
forecasting
Master thesis

2023

Bc. Aleš Jandera

Technical University of Košice
Faculty of Mining, Ecology, Process Control
and Geotechnologies

Advanced prediction models for sales
forecasting

Master thesis

Study Programme: Process control of raw materials and material extraction
and processing
Field of Study: Cybernetics
Department: Faculty of Mining, Ecology, Process Control and
Geotechnologies (FBERG)
Supervisor: doc. Ing. Tomáš Škovránek, PhD.

Košice 2023

Bc. Aleš Jandera

Abstract in English

Sales forecasting can be divided into two main categories: short-term and long-term forecasting. Short-term forecasting is generally done on a weekly or monthly basis. Long-term forecasting is done on a quarterly or annual basis. There are many different methods that can be used for sales forecasting. The most common method is trend analysis. Trend analysis looks at past sales data to identify patterns and trends that can be used to predict future sales. Other methods include regression analysis and time series analysis. Advanced prediction modelling is a type of long-term forecasting. It uses historical data and statistical techniques to predict future sales. Advanced prediction modelling is often used by companies to make strategic decisions about inventory, pricing, and marketing.

Keywords in English

Mathematic modeling, forecasting, linear prediction

Abstract in Slovak

Prognózu predaja možno rozdeliť do dvoch hlavných kategórií: krátkodobá a dlhodobá prognóza. Krátkodobé prognózy sa vo všeobecnosti vykonávajú na týždennej alebo mesačnej báze. Dlhodobé prognózy sa robia na štvrťročnej alebo ročnej báze. Existuje mnoho rôznych metód, ktoré možno použiť na predpovedanie predaja. Najbežnejšou metódou je analýza trendov. Analýza trendov sa zameriava na údaje o minulých predajoch, aby identifikovala vzory a trendy, ktoré možno použiť na predpovedanie budúceho predaja. Ďalšie metódy zahŕňajú regresnú analýzu a analýzu časových radov. Pokročilé predikčné modelovanie je typ dlhodobého predikovania. Na predpovedanie budúceho predaja využíva historické údaje a štatistické techniky. Pokročilé predikčné modelovanie často používajú spoločnosti na strategické rozhodnutia o zásobách, cenách a marketingu.

Keywords in Slovak

Matematické modelovanie, predpoved, lineárna predikcia

Bibliographic Citation

JANDERA, Aleš. *Advanced prediction models for sales forecasting*. Košice: Technical University of Košice, Faculty of Mining, Ecology, Process Control and Geotechnologies, 2023. 66s. Supervisor: doc. Ing. Tomáš Škovránek, PhD.

71130

TECHNICKÁ UNIVERZITA V KOŠICIACH
FAKULTA BANÍCTVA, EKOLÓGIE, RIADENIA A GEOTECHNOLÓGIÍ
Ústav riadenia a informatizácie výrobných procesov

ZADANIE DIPLOMOVEJ PRÁCE

Študijný odbor: **Kybernetika**
Študijný program: **Riadenie procesov získavania a spracovania surovín**

Názov práce:

Pokročilé predikčné modely pre prognózu predaja
Advanced prediction models for sales forecasting

Študent: **Bc. Aleš Jandera**
Školiteľ: **doc. Ing. Tomáš Škovránek, PhD.**
Školiace pracovisko: **Ústav riadenia a informatizácie výrobných procesov**
Konzultant práce:
Pracovisko konzultanta:


Pokyny na vypracovanie diplomovej práce:

1. Úvod
2. Analýza súčasného stavu
3. Návrhová časť
4. Zhodnotenie prínosu práce a návrh odporúčaní pre ďalší postup
5. Záver

Odporúčaný rozsah práce: 50 - 70 strán

Jazyk, v ktorom sa práca vypracuje: **anglický**
Termín pre odovzdanie práce: **21.04.2023**
Dátum zadania diplomovej práce: **31.10.2022**




Dr. h. c. prof. Ing. Michal Cehlár, PhD.
dekan fakulty

Declaration

I hereby declare that this thesis is my own work and effort. Where other sources of information have been used, they have been acknowledged.

Košice, 21.4.2023

.....

Signature

Acknowledgement

Very strong thanks to whole teaching staff at the Institute of Control and Informationization of Production Processes for their patience, leadership and knowledge which helped me to write this thesis. I would like to express my deepest appreciation to my supervisor Tomas Skovranek for his time and advices.

Contents

Introduction	1
1 Analytical part	2
1.1 Models used for sales data forecasting	3
1.1.1 Time-series models	3
1.1.2 Regression models	4
1.1.3 Decision tree models	5
1.1.4 Machine learning models	6
1.2 Neural networks	7
1.2.1 Classification	12
1.2.2 Activation functions	12
1.2.3 Application of neural networks in prediction	13
1.3 Linear prediction	15
1.3.1 Short-term linear prediction	15
1.3.2 Long-term linear prediction	16
1.3.3 Estimation of optimal prediction order	19
1.3.4 Levinson-Durbin scheme	20
1.4 Detection of repeatable patterns	22
1.4.1 Hidden Markov models	23
1.4.2 Fourier transform	24
1.4.3 Seasonal decomposition	25
1.4.4 Pattern Masking for Dictionary Matching	26
2 Goal of the thesis	28
3 Methodology	29
3.1 Characteristics of the research object	29
3.2 Methods	30
3.2.1 Linear prediction	30
3.2.2 Backpropagation	31

3.3	Datasets	31
3.4	Comparison criteria	32
3.5	Statistical measures	34
4	Syntactic part	36
4.1	Novel approach to forecast sales data	36
4.2	Extended long-term prediction	37
4.3	Prediction order estimation	38
4.4	Long-term prediction shift estimation	40
4.4.1	Autocorrelation approach	40
4.4.2	Neural network approach for shift estimation	41
4.5	Estimation of the seasonal weights	44
5	Evaluation	46
5.1	Preprocessing of input data	47
5.2	Mathematical model for sales forecasting	48
5.3	Results of the experiment	49
6	Summary	54
7	Resume	56
7.1	Analýza	56
7.2	Syntéza	59
7.3	Experiment	60
7.4	Záver	62
	Bibliography	64

List of Figures

1.1	Example of decision tree flow.	6
1.2	Perceptron preview. [7]	8
1.3	Typical feed-forward neural network composed of three layers. [8]	9
1.4	Illustration of (a) Convolution, (b) Tiled Convolution, (c) Dilated Convolution, and (d) Deconvolution. [9]	10
1.5	Typical recurrent network. [10]	10
1.6	Long short-term memory network. [11]	10
1.7	An autoencoder neural network. [12]	11
1.8	The conditional GAN schema. [13]	11
1.9	Neural network activation functions.	14
1.10	Shift explanation [18].	18
5.1	Results of shot-term linear prediction.	50
5.2	Results of long-term linear prediction.	51
5.3	Comparison of MSE for standard and extended LP.	52
5.4	Results of prediction performance.	53

List of Tables

5.1	Comparison of linear prediction models	53
-----	--	----

Introduction

Linear prediction [1] is a method used in signal processing to predict future values of a time series based on past observations. The technique is based on the assumption that the signal can be modeled as a linear combination of past values and a noise term. Long-term linear prediction refers to the application of this method to predict values over a longer period of time, such as months or years. It requires a iteater amount of data and is more complex than short-term prediction, but can be useful in areas such as stock market forecasting and weather prediction. The goal of this master thesis is to develop new algorithms and mathematical models to improve the accuracy of long-term predictions and use signal principles in sales forecasting. Matlab¹ [2] will be used as development environment to validate developed models and machine learning developed approaches.

Task formulation

To propose a mathematical model and algorithms for sales forecasting based on long-term prediction with improved Levinson - Durbin scheme which should have better performance and accuracy than known linear prediction mechanisms in the sales forecasting.

¹MATLAB is a fourth-generation programming language and numerical analysis environment. Uses for MATLAB include matrix calculations, developing and running algorithms, creating user interfaces (UI) and data visualization.

1 Analytical part

Linear prediction is a statistical method used to predict future values based on historical data. The Durbin-Levinson algorithm is a method for solving the linear prediction problem for autoregressive (AR) models¹, which are models where the current output depends on previous outputs. The algorithm solves the linear prediction problem by finding the coefficients of the AR model that minimize the prediction error. The resulting AR coefficients can be used to make predictions about future values based on past observations. This method should be used with using the pattern of linear relationship between the independent and dependent variables. Here's a basic outline of the steps involved in using linear prediction to forecast sales data:

1. Collect sales data: Gather the historical sales data for the product or service that we want to forecast.
2. Plot the data: Plot the sales data over time to visually inspect the trend and identify any patterns.
3. Choose a model: Select an appropriate linear model to represent the relationship between the independent and dependent variables in the data. For example, we might choose a simple linear regression model.
4. Train the model: Train the selected model on the historical sales data using a method such as least squares regression.

¹Autoregressive (AR) models are time series models that describe the the relationship between the current value of a variable and its past values. In an autoregressive model, each observation is modeled as a linear combination of past observations, with weights called AR coefficients. AR models are widely used in various fields such as economics, engineering, and finance for modelling and forecasting time series data. The order of the AR model, denoted as "p", refers to the number of past values used to predict the current value. For example, an AR(1) model uses only the previous observation to predict the current value, while an AR(2) model uses the previous two observations.

5. Make predictions: Use the trained model to make predictions on future sales data. We may want to generate predictions for several months or years in advance.
6. Evaluate the model: Assess the accuracy of the predictions by comparing them to the actual sales data. Use metrics such as mean absolute error or root mean squared error to quantify the model's performance.
7. Refine the model: If necessary, refine the model by adding additional independent variables or transforming the existing variables. Repeat the training and evaluation steps until we have a model that provides accurate forecasts.

For shift calculation in longterm prediction can be used autocorrelation method, in this thesis neural network for identification shift will be developed and similar mechanism for optimal order detection.

1.1 Models used for sales data forecasting

There are several mathematical models used for sales prediction, including Time series models 1.1.1 which are used to analyse and forecast sales data over time, such as seasonal patterns, trends, and fluctuations. Regression models 1.1.2 to use historical data to determine the relationship between sales and one or more independent variables, such as price, promotion, and advertising. Decision tree models 1.1.3 use a tree-like structure to make decisions based on the relationship between sales and multiple independent variables and machine learning models 1.1.4 to use algorithms such as neural networks and support vector machines to make predictions based on patterns in the data.

The choice of mathematical model depends on the characteristics of the data, the desired level of accuracy, and the computational resources available.

1.1.1 Time-series models

Time-series models are mathematical models used to analyse and forecast data that are collected over time [3]. These models are used to study and make predictions about the trends, patterns, and behaviour of the data over time, taking into account historical values and their relationship with the present. Time-series

models are widely used in areas such as economics, finance, and weather forecasting, among others. The models are based on various statistical techniques, including ARIMA (AutoRegressive Integrated Moving Average), SARIMA (Seasonal ARIMA), and exponential smoothing, among others. The goal of time-series modelling is to build a mathematical representation of the underlying process that generates the time-series data, allowing for accurate prediction of future values. Time-series models are statistical models used to analyse and make predictions about time-dependent data. They are widely used in various fields, including finance, economics, engineering, and social sciences.

Time-series models make use of past values of a variable to predict future values. They assume that there is a pattern or trend in the data that can be used to forecast future behaviour. Some commonly used time-series models include:

- Autoregressive Integrated Moving Average (ARIMA). This model is used to analyse and forecast stationary time-series data. It consists of three components: autoregression, differencing, and moving average.
- Seasonal Autoregressive Integrated Moving Average (SARIMA) is an extension of ARIMA that takes into account seasonal patterns in the data.
- Exponential Smoothing (ETS) is used to forecast time-series data that have a trend and/or seasonality. It uses a smoothing parameter to assign more or less weight to past observations based on their recency.
- Vector Autoregression (VAR) is used when there are multiple time-series variables that influence each other. It can be used to analyse the relationships between these variables and to make predictions about their future behavior.
- These models are valuable tools for analyzing and predicting time-series data, but they require careful consideration of the specific characteristics of the data being analysed and the appropriate model to use.

1.1.2 Regression models

Regression models are a type of statistical models used to examine the relationship between a dependent variable and one or more independent variables [4]. The goal of regression analysis is to model the relationship between these variables and make predictions about the dependent variable based on the values of the independent variables. Regression models are widely used in many fields, including economics, finance, marketing, and social sciences, to make predictions

and understand the relationship between variables. There are several types of regression models, including:

- Linear regression is a simple regression model where the relationship between the dependent and independent variables is modeled using a linear equation.
- Logistic regression is used for binary classification problems where the dependent variable is binary and the goal is to model the relationship between the independent variables and the probability of the dependent variable being either 0 or 1.
- Multiple regression is used when there are multiple independent variables and the goal is to model the relationship between all of these variables and the dependent variable.
- Polynomial regression is used when the relationship between the dependent and independent variables is non-linear and can be modeled using a polynomial equation.

The choice of regression model depends on the nature of the data and the research question being asked.

1.1.3 Decision tree models

Decision tree models are a type of machine learning models used for both regression and classification tasks [5]. They are tree-like structures that make predictions by breaking down a dataset into smaller and smaller subsets, based on the values of the input variables. At each internal node of the tree, a decision rule is used to split the data based on the value of a feature, and the process continues until the data are separated into homogeneous groups, or leaves. The predictions are then made based on the average or majority class in each leaf node. On figure 1.1 we can see example schema how the decision tree works.

Decision trees have several advantages, including ease of interpretability, handling of non-linear relationships, and ability to handle both categorical and numerical data. Some examples of decision tree algorithms are CART (Classification and Regression Tree) and Random Forest.

The decision tree model is trained using a dataset, and the tree structure is built using a greedy algorithm that seeks to maximize the reduction in impurity of

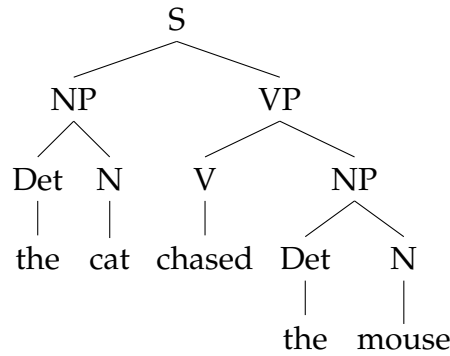


Figure 1.1: Example of decision tree flow.

the target variable at each split. The model can then be used to make predictions on new data by following the decision rules in the tree.

1.1.4 Machine learning models

Machine learning models are a subset of artificial intelligence that allows computers to learn and make predictions or decisions without being explicitly programmed. Machine learning models are based on algorithms that use statistical methods to find patterns in data and make predictions about new, unseen data. There are several types of machine learning models, including **Supervised learning** where the model is trained on labeled data, with the goal of learning the relationship between the input features and the target variable, and making predictions about the target variable for new, unseen data. **Unsupervised learning** where the model is trained on unlabeled data, with the goal of finding patterns or structure in the data, such as clustering or dimensionality reduction. **Reinforcement learning** where the model learns by receiving rewards or penalties for its actions in an environment, with the goal of maximizing the reward over time. **Deep learning** a subset of machine learning that uses artificial neural networks with multiple hidden layers to model complex relationships in the data.

The choice of machine learning model depends on the problem being solved and the type of data being used. Machine learning models have been applied to a wide range of tasks, including image and speech recognition, natural language processing, and predictive modelling. Advances in machine learning (ML), faster processors and the availability of digitized healthcare data have contributed to a growing number of papers describing ML applications in healthcare [6].

K-Nearest Neighbors (KNN)

Machine learning algorithms, K-Nearest Neighbors (KNN), is a type of artificial intelligence that allows computers to learn and make decisions based on data. KNN is a simple yet effective algorithm used for classification and regression tasks in supervised learning.

KNN is a non-parametric algorithm, which means it doesn't make any assumptions about the underlying distribution of the data. Instead, it uses the similarity between data points to classify or predict the target variable. The algorithm works by finding the k-nearest neighbors to a new data point and then assigning the class label of the majority of those neighbors to the new data point.

For example, if we have a dataset of images with labels indicating whether each image contains a cat or a dog, we can use KNN to classify a new image by finding the k-nearest neighbors to the new image and assigning the label of the majority of those neighbors to the new image.

KNN is a relatively simple algorithm, but it can be very effective when applied to the right problems. It is particularly useful in cases where the decision boundary is nonlinear or where there is no clear separation between classes. However, KNN can be computationally expensive when working with large datasets, and it may not perform well in high-dimensional spaces.

1.2 Neural networks

A neural network is a type of machine learning algorithm inspired by the structure and function of biological neurons in the human brain. It is composed of interconnected nodes, called neurons, that are organised into layers. The input layer receives raw data, such as images or text, and passes it on to the hidden layers, which perform calculations and apply weights to the input data to create a prediction. Finally, the output layer produces the final prediction or classification.

As we can see on image 1.2 each input X_n should be properly weighted by a certain weight W_n before all the signals enter the summation stage. Afterwards, the weighted summation is forwarded into the activation unit producing the neuron's output signal.

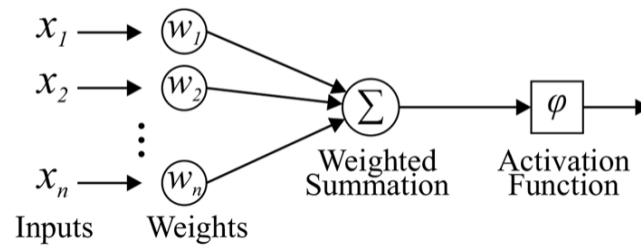


Figure 1.2: Perceptron preview. [7]

Neural networks are trained on large datasets using a process called backpropagation, which adjusts the weights and biases of the neurons to minimize the error between the predicted output and the actual output. Once a neural network has been trained, it can be used to make predictions on new data.

A neuron is a basic building block of a neural network, also known as an artificial neuron or a perceptron. It is modeled based on the biological neuron in the human brain, which receives input signals from other neurons, processes them, and sends output signals to other neurons.

In a neural network, a neuron receives input from other neurons or directly from the input data, applies a mathematical function to the input, and produces an output that is sent to other neurons in the network. The input to a neuron is usually a vector of numbers, and each input is multiplied by a corresponding weight. The neuron then sums up the weighted inputs, adds a bias term, and applies an activation function to the result.

The purpose of the activation function is to introduce nonlinearity into the neuron, which allows the neural network to learn complex patterns and relationships in the data. There are several different types of activation functions that can be used, such as the sigmoid function, ReLU (Rectified Linear Unit) function, and tanh (hyperbolic tangent) function.

The output of a neuron is typically fed into other neurons in the next layer of the neural network. The weights and biases of the neurons are adjusted during the training process using a technique called backpropagation, which involves computing the gradient of the error with respect to the weights and updating

them using an optimization algorithm such as stochastic gradient descent.

Overall, the neurons in a neural network work together to learn patterns and relationships in the input data and produce output that can be used for a variety of tasks, such as classification, regression, and prediction.

Neural networks have been successfully applied in a wide range of fields, including image and speech recognition, natural language processing, and autonomous vehicles, among others. Neural networks can be broadly classified into the following types:

Feedforward Neural Networks

These are the most basic type of neural networks, where the information flows only in one direction, from input to output. These networks can have one or more hidden layers and are often used for classification or regression tasks. The schema of a basic feedforward NN is on Fig. 1.3

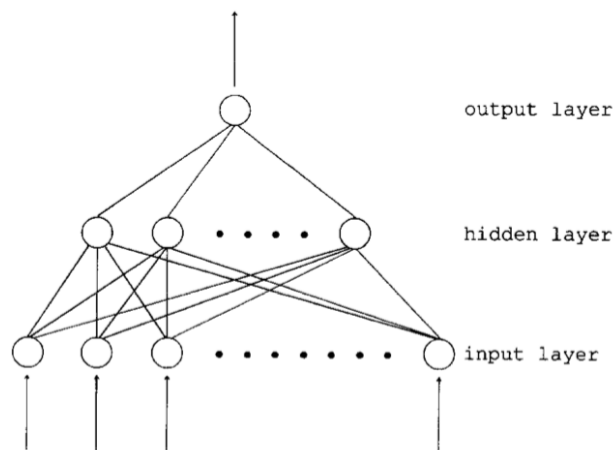


Figure 1.3: Typical feed-forward neural network composed of three layers. [8]

Convolutional Neural Networks (CNNs)

These networks are specialized for processing images and are commonly used in computer vision tasks. They use convolutional layers to extract features from images and can learn to recognise patterns and objects in images see in Fig. 1.4

Recurrent Neural Networks (RNNs)

These networks are designed to work with sequential data, such as time-series or natural language data. They have loops that allow information to be passed

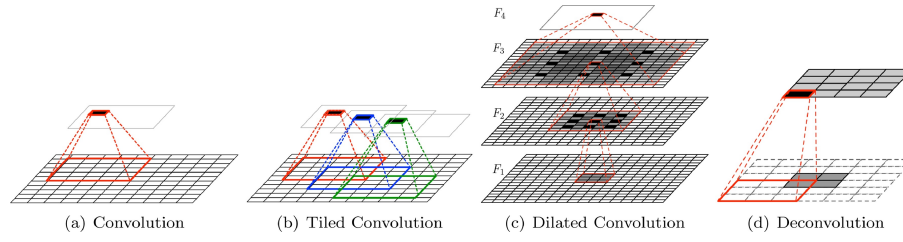


Figure 1.4: Illustration of (a) Convolution, (b) Tiled Convolution, (c) Dilated Convolution, and (d) Deconvolution. [9]

from one time-step to the next, enabling them to capture temporal dependencies in the data, described on Fig. 1.5

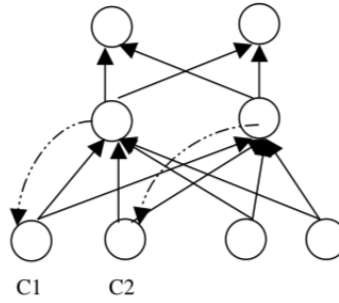


Figure 1.5: Typical recurrent network. [10]

Long Short-Term Memory Networks (LSTMs)

These are a type of RNN that are designed to address the problem of vanishing gradients in traditional RNNs. They use memory cells and gates to selectively retain or forget information over time, making them well-suited for learning from long sequences. As you can see on Fig. 1.6 color indicates degree of memory activation.

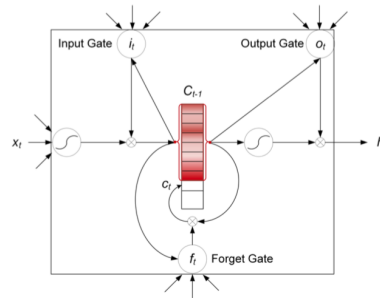


Figure 1.6: Long short-term memory network. [11]

Autoencoder Neural Networks

These networks are used for unsupervised learning and are designed to learn a compressed representation of the input data. As we can see on Fig. 1.7 they consist of an encoder that maps the input data to a compressed representation, and a decoder that maps the compressed representation back to the original data.

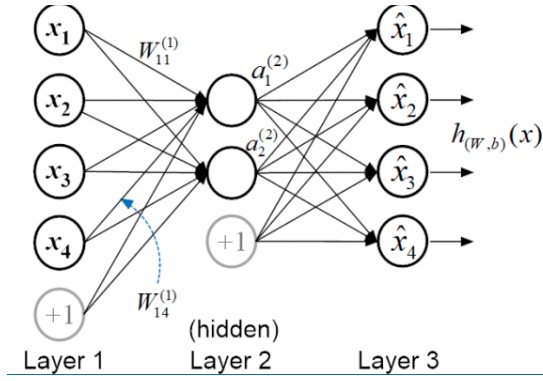


Figure 1.7: An autoencoder neural network. [12]

Generative Adversarial Networks (GANs)

These networks consist of two networks, a generator and a discriminator (see on Fig. 1.8), that are trained together in a game-theoretic framework. The generator is trained to generate realistic data samples, while the discriminator is trained to distinguish between real and generated data samples.

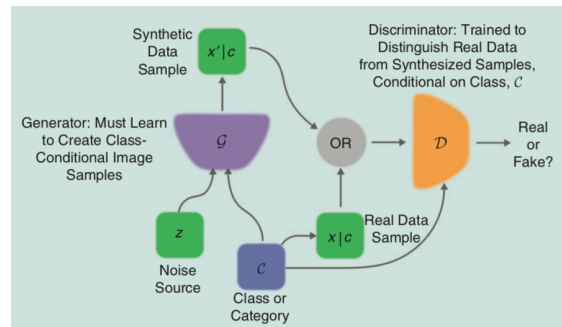


Figure 1.8: The conditional GAN schema. [13]

These are some of the most common types of neural networks, but there are many other specialized types of neural networks that have been developed for specific tasks, such as object detection, speech recognition, and natural language processing.

1.2.1 Classification

Neural network data classification is a technique for categorising data into different classes or categories based on patterns and features present in the data. a neural network is a type of machine learning algorithm that is modeled after the structure and function of the human brain. It is composed of interconnected nodes or neurons that are organised into layers.

In a classification task, the neural network is trained on a dataset that is labeled with the correct class for each example. During training, the network learns to recognise patterns and features in the input data that are associated with each class. The process of training involves adjusting the weights and biases of the neurons in the network to minimise the error between the predicted class and the actual class of each example in the training set.

Once the neural network is trained, it can be used to classify new, unseen examples by inputting the data into the network and obtaining a prediction of the most likely class. The output of the neural network is a probability distribution over the different classes, with the highest probability indicating the predicted class. Neural network data classification has been successfully applied to a wide range of tasks, including image classification, speech recognition, natural language processing, and fraud detection, among others [14].

1.2.2 Activation functions

There are several types of activation functions [15] used in neural networks, as we can see on Fig. 1.9 including:

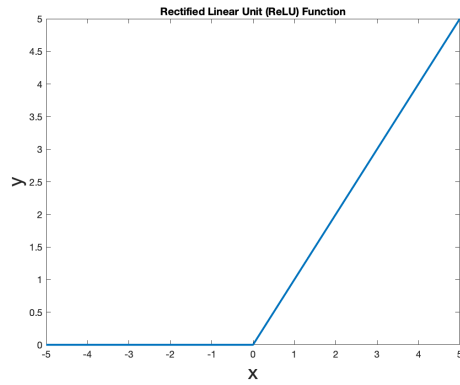
- **Sigmoid Function:** the sigmoid function is a commonly used activation function that maps any input value to a value between 0 and 1. It is typically used in binary classification problems and in the output layer of neural networks that produce probability estimates 1.9b.
- **ReLU (Rectified Linear Unit):** the ReLU function is another popular activation function that maps any input value less than 0 to 0, and any input value greater than or equal to 0 to the input value itself. It is computationally efficient and has been shown to work well in deep neural networks 1.9a.
- **Tanh Function:** the tanh (hyperbolic tangent) function is similar to the sigmoid function, but it maps input values to a range between -1 and 1. It is commonly used in the hidden layers of neural networks 1.9c.

- **Softmax Function:** the softmax function is often used in the output layer of neural networks that produce multi-class classification predictions. It maps the outputs to a probability distribution over the possible classes 1.9d.
- **Leaky ReLU:** the Leaky ReLU function is similar to the ReLU function, but it allows a small, non-zero gradient when the input value is negative. This can help to prevent the "dying ReLU" problem, where some ReLU units become inactive and stop contributing to the network's output 1.9e.
- **ELU (Exponential Linear Unit):** the ELU function is similar to the ReLU function, but it allows negative values to have non-zero outputs. This can help to prevent the "dying ReLU" problem and can improve the performance of deep neural networks 1.9f.

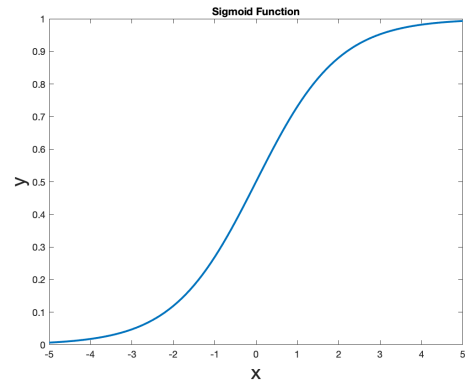
These are some of the most commonly used activation functions in neural networks, but there are many other types of activation functions that have been developed for specific tasks or to address certain problems.

1.2.3 Application of neural networks in prediction

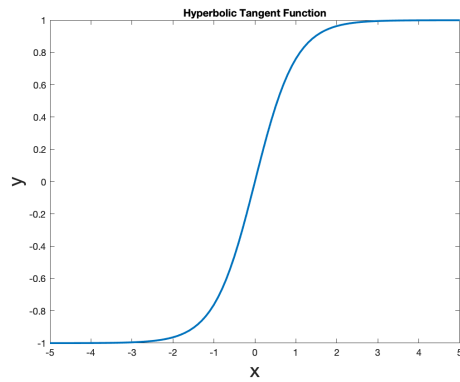
Neural networks can be used for long-term linear prediction in time-series data, including speech, health, and stock data. Neural networks can be trained to model and forecast the patterns in the data, including shifts in the long-term linear prediction. Neural networks have been shown to be effective in capturing the complex relationships between variables in time-series data, and can learn to identify subtle patterns and trends that may be difficult to detect using traditional statistical methods. In the context of long-term linear prediction, neural networks can be trained on historical data to identify previous patterns and trends in the data and make predictions for future values. They can also be used to detect shifts in the long-term linear prediction, which can be useful for identifying changes in the underlying processes that generate the data. However, it's important to note that neural networks can be computationally expensive and require large amounts of data to train effectively. They also require careful tuning of hyperparameters and selection of appropriate architecture to achieve good performance. In addition, the interpretation of the results of a neural network can be more challenging than with traditional statistical methods. Overall, neural networks can be a powerful tool for long-term linear prediction in time-series data, including shifts in the data, but their use should be carefully considered based on the specific application and available data.



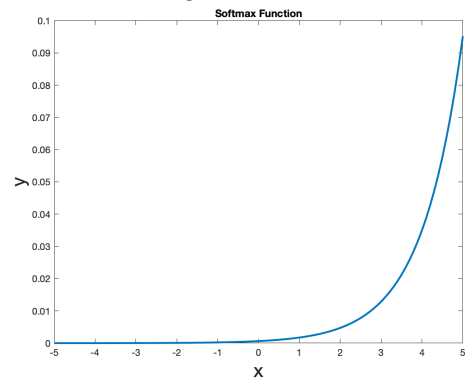
(a) ReLU function



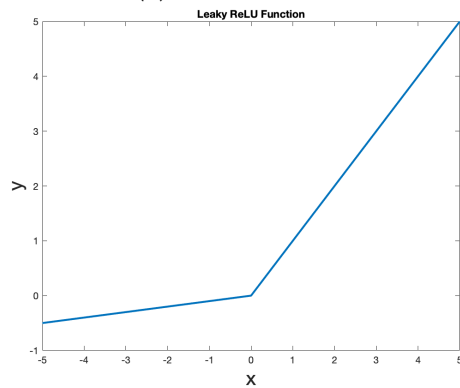
(b) Sigmoid function



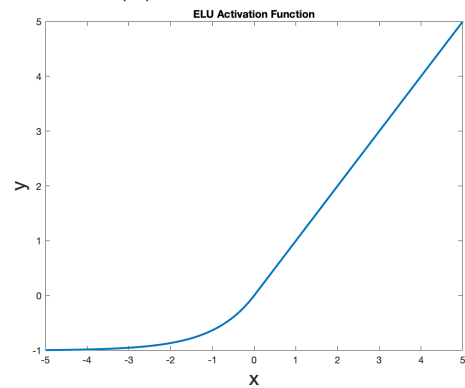
(c) Tanh function



(d) Softmax function



(e) Leaky ReLU function



(f) ELU function

Figure 1.9: Neural network activation functions.

1.3 Linear prediction

Linear prediction is a statistical technique used to forecast future values based on past observations. It is a method for modelling the relationship between a independent variable and one or more dependent variables in a linear form. The goal of linear prediction is to find the best linear approximation of the relationship between the variables, which can then be used to make predictions about future values of the dependent variable.

Linear prediction can be performed using simple linear regression or multiple linear regression, depending on the number of independent variables involved [16]. In simple linear regression, a single independent variable is used to predict the value of the dependent variable, while in multiple linear regression, multiple independent variables are used to make the prediction.

Linear prediction models are commonly used in finance, economics, and engineering, among other fields, to forecast future values of time series data, such as stock prices, sales, or demand. The accuracy of linear prediction models depends on several factors, including the quality of the data, the choice of independent variables, and the degree of linearity in the relationship between the variables.

1.3.1 Short-term linear prediction

Short-term linear prediction [17] refers to the use of linear prediction techniques to make predictions about the near-term future values of a time series. It is used to forecast future values of a dependent variable based on its past values and any relevant independent variables.

In short-term linear prediction, the focus is on accurately predicting the next few values of the dependent variable, typically in the range of several weeks to a few months. The linear prediction models used for short-term forecasting are typically simple and straightforward, often using a small number of independent variables. The goal is to provide a quick and easily interpretable forecast that can be used to make operational decisions in the short-term. Consider a signal $x(n)$ that is modeled using a linear predictor. The linear predictor is defined as:

$$\hat{x}(n) = \sum_{i=1}^p a_i x(n-i), \quad (1.1)$$

where $\hat{x}(n)$ is the predicted value of $x(n)$, p is the order of the predictor, and a_i are the predictor coefficients. The predictor coefficients can be found by minimizing the prediction error:

$$e(n) = x(n) - \hat{x}(n), \quad (1.2)$$

Using the least squares method, we can find the predictor coefficients by minimizing the sum of squared prediction errors:

$$\min_{a_1, \dots, a_p} \sum_{n=1}^N e^2(n), \quad (1.3)$$

This can be solved analytically using the following matrix equation:

$$\begin{bmatrix} r(0) & r(1) & \cdots & r(p-1) \\ r(1) & r(0) & \cdots & r(p-2) \\ \vdots & \vdots & \ddots & \vdots \\ r(p-1) & r(p-2) & \cdots & r(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} = \begin{bmatrix} r(1) \\ r(2) \\ \vdots \\ r(p) \end{bmatrix}, \quad (1.4)$$

where $r(k)$ is the autocorrelation function of the signal $x(n)$ defined as:

$$r(k) = \frac{1}{N} \sum_{n=k+1}^N x(n)x(n-k), \quad (1.5)$$

Common techniques for short-term linear prediction include moving average models, exponential smoothing, and autoregressive models. These methods use the historical data of the time series to model the relationship between the dependent and independent variables, and to make predictions about future values. The accuracy of short-term linear predictions can be evaluated using metrics such as mean absolute error, mean squared error, or the correlation coefficient between the actual and predicted values.

1.3.2 Long-term linear prediction

Long-term linear prediction is a technique commonly used in the analysis of time-series data, including health data. In the context of health data, the shift of long-term linear prediction refers to the way in which the patterns in the data change over time, reflecting changes in the underlying health status of the individual or population being studied.

For example, in the case of a patient with a chronic disease, the long-term linear prediction of their health data may show a gradual decline over time as the disease progresses. Alternatively, the data may show periodic shifts corresponding to changes in medication or other interventions. In population health studies, long-term linear prediction can be used to identify trends and patterns in health outcomes over time. For example, a long-term linear prediction model might be

used to track changes in the prevalence of a particular disease or condition over a period of several years, taking into account factors such as demographic changes and changes in healthcare policy. Overall, the shift of long-term linear prediction for health data reflects the dynamic nature of health status and healthcare interventions, and can be a powerful tool for understanding trends and patterns in health outcomes over time.

Long-term linear prediction refers to the use of linear prediction techniques to make predictions about the future values of a time series over an extended period of time, typically several months to several years. Unlike short-term linear prediction, which focuses on forecasting the near-term future, long-term linear prediction aims to provide a more comprehensive and accurate forecast of future values. On the figure 1.10 you can see the explanation of the long-term linear prediction shift.

$$\hat{x}(n) = \sum_{i=1}^p a_i x(n-i) + \sum_{i=-q}^q b_i x(n-i-Q), \quad (1.6)$$

where $\hat{x}(n)$ is the predicted value of the signal at time n , $x(n-i)$ are the past p samples of the signal, $x(n-i-Q)$ are the past q shifted samples of the signal, and a_i are the short-term predictor coefficients and b_i is the long-term predictor coefficient. The order of the predictor is p for the autoregressive (AR) part and q for the long-term part. The pitch period Q can be obtained from the autocorrelation function [18].

The long-term linear prediction equation is a generalization of the short-term linear prediction equation that was shown in section 1.3.1. The short-term equation only includes the autoregressive (AR) part, whereas the long-term equation includes both the autoregressive (AR) and the moving average (MA) parts.

In long-term linear prediction, more sophisticated models [19] are typically used, such as multiple linear regression or time series models, and a larger number of independent variables may be considered. The models are also trained on a larger historical dataset to ensure that they capture any long-term trends or patterns in the data.

Long-term linear prediction is commonly used in fields such as finance, economics, and marketing, to make long-term projections about variables such as sales, demand, or stock prices. The goal is to provide a comprehensive and accu-

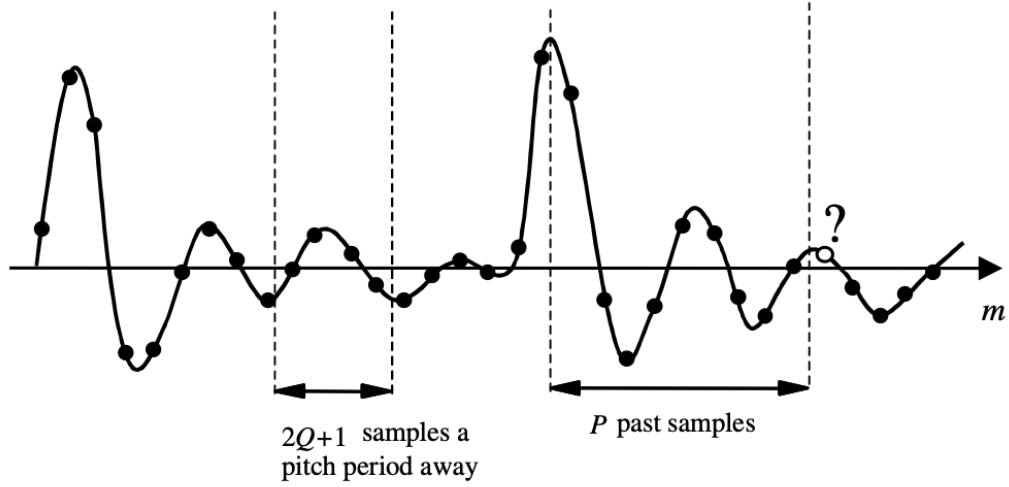


Figure 1.10: Shift explanation [18].

rate forecast that can be used to make strategic decisions in the long-term. The accuracy of long-term linear predictions can be evaluated using the same metrics as for short-term linear predictions [20], as well as additional metrics such as mean absolute percentage error or mean absolute scaled error.

In some long-term prediction use cases it needs to solve the suppression of Late Reverberation Effect² this can be done by with minimal performance degradation by framework developed by Keisuke Kinoshita [21] for both single-channel and multichannel scenarios.

Shift Estimation

Correct estimate of the shift in the long-term linear prediction is important to get correct results. Mechanism to detect the shift is based on detection of repeatable patterns 1.4 mechanism in our datasets. To calculate the shift for long-term linear prediction, we can use the autocorrelation function of the signal. The shift value corresponds to the lag at which the autocorrelation function has its maximum value. Other approaches to estimate shift we can see in section.

²Late Reverberation Effect refers to the decay of sound in an environment after the initial sound source has stopped. This effect results in the persistence of sound in a space for a short period of time and helps create the characteristic ambiance of a room or space. It is an important aspect of room acoustics and is used in sound design and music production to enhance the perceived sound quality and spatial experience of audio.

1.3.3 Estimation of optimal prediction order

The optimal order of linear prediction refers to the number of past observations that should be used to predict the next observation in a time series. There are several methods for finding the optimal order of linear prediction, including the Akaike Information Criterion (AIC), the Bayesian Information Criterion (BIC), and cross-validation. The Akaike Information Criterion (AIC) is a measure of the relative quality of a statistical model for a given set of data. The AIC score is based on the goodness of fit of the model and the complexity of the model. In the context of linear prediction, the AIC can be used to select the optimal order of the autoregressive (AR) model. The AR model uses past observations to predict the next observation in a time series. The optimal order of the AR model is the order that minimizes the AIC score. The Bayesian Information Criterion (BIC) is similar to the AIC but places a greater penalty on model complexity. The BIC can also be used to select the optimal order of the AR model. Cross-validation is a technique that involves splitting the data into training and testing sets. The model is trained on the training set, and the performance of the model is evaluated on the testing set. The optimal order of the AR model is the order that produces the best performance on the testing set. Overall, the optimal order of linear prediction can be found using a combination of these methods, taking into account the complexity of the model, the goodness of fit, and the performance on a testing set.

Autocorrelation can be a useful tool for detecting the optimal shift in long-term linear prediction for time-series data. Autocorrelation measures the linear relationship between lagged versions of a time-series, and can be used to identify the presence of cyclic patterns in the data. In the context of long-term linear prediction, autocorrelation can be used to identify the lag that maximizes the correlation between past and future values of the time-series. This lag can be used as the optimal shift for the long-term linear prediction model. To use autocorrelation for detecting the optimal shift in long-term prediction, one would first calculate the autocorrelation function (ACF) for the time-series data. The ACF measures the correlation between the time-series at different lags. The lag at which the ACF is highest corresponds to the optimal shift for long-term linear prediction. Once the optimal shift has been identified, it can be used to train a long-term linear prediction model using techniques such as autoregressive models, moving average models, or combinations of both. However, it's important to note that while autocorrelation can be a useful tool for identifying the optimal shift for long-term lin-

ear prediction, it may not always be the most appropriate technique for all types of time-series data. Other methods, such as spectral analysis or wavelet analysis, may be more appropriate in some cases.

Autocorrelation function (ACF) of a time series is typically denoted as ρ_k or r_k and can be expressed mathematically using the following equation:

$$\rho_k = \frac{\gamma_k}{\gamma_0} = \frac{\sum_{t=k+1}^T (y_t - \bar{y})(y_{t-k} - \bar{y})}{\sum_{t=1}^T (y_t - \bar{y})^2}, \quad (1.7)$$

where y_t is the value of the time series at time t , \bar{y} is the mean of the time series, k is the lag or time shift, γ_k is the autocovariance at lag k , γ_0 is the autocovariance at lag 0, and T is the total number of observations in the time series. The ACF measures the correlation between the time series and a lagged version of itself, with values ranging between -1 and 1.

1.3.4 Levinson-Durbin scheme

The Levinson-Durbin algorithm [22] is an iterative numerical method used to solve the autoregressive (AR) model of a time series.

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \cdots + \beta_p Y_{t-p} + \epsilon_t, \quad (1.8)$$

where Y_t is the dependent variable at time t , β_0 is the intercept, $\beta_1, \beta_2, \dots, \beta_p$ are the regression coefficients, $Y_{t-1}, Y_{t-2}, \dots, Y_{t-p}$ are the lagged values of the dependent variable up to order p , and ϵ_t is the error term at time t .

AR models are used to model and forecast time series data [23], such as sales data, by assuming that each future value of the series depends on a linear combination of previous values.

The Levinson-Durbin algorithm solves the AR model by iteratively updating the coefficients of the model to minimize the prediction error between the model and the actual data. The algorithm is fast and efficient, and it is widely used in digital signal processing, speech processing, and control systems, among other applications. The Levinson-Durbin recursion schema is a method for solving the autocorrelation equations of a linear prediction problem. It can be expressed in the form of a triangular system of equations:

$$\begin{aligned}\alpha_0 &= r_0 \\ \alpha_k &= \frac{1}{k} \sum_{i=1}^k a_{k-i} r_i \quad (1 \leq k < p) \\ \alpha_p &= \frac{1}{p} \sum_{i=1}^p a_{p-i} r_i,\end{aligned}$$

where r_i is the i th autocorrelation coefficient, α_k is the k th reflection coefficient, and a_k is the k th prediction coefficient.

The recursion begins with $\alpha_0 = r_0$, and computes each subsequent reflection coefficient α_k in terms of the previous coefficients $\alpha_0, \alpha_1, \dots, \alpha_{k-1}$ and the autocorrelation coefficients r_1, r_2, \dots, r_k .

The Levinson-Durbin algorithm is often used as an alternative to the Yule-Walker equations, which are another commonly used method for solving AR models. Unlike the Yule-Walker equations, the Levinson-Durbin algorithm can be easily modified to handle non-stationary time series data, and it is also more robust to numerical issues such as rounding errors.

$$\begin{bmatrix} r_0 & r_1 & \cdots & r_{p-1} \\ r_1 & r_0 & \cdots & r_{p-2} \\ \vdots & \vdots & \ddots & \vdots \\ r_{p-1} & r_{p-2} & \cdots & r_0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} = \begin{bmatrix} -r_1 \\ -r_2 \\ \vdots \\ -r_p \end{bmatrix},$$

where r_0, r_1, \dots, r_{p-1} are the autocorrelation coefficients and a_1, a_2, \dots, a_p are the prediction coefficients. The algorithm proceeds as follows:

1. Set $\alpha_0 = r_0$ and $k = 1$.
2. Compute the reflection coefficient α_k using the formula

$$\alpha_k = -\frac{1}{\alpha_{k-1}} \sum_{i=1}^{k-1} \alpha_{k-i} r_i.$$

3. Compute the new prediction coefficients a_1, a_2, \dots, a_k using the formulas

$$\begin{aligned}a_k &= -\alpha_k \\ a_i &= a_{i-1} - \alpha_k a_{k-i} \quad (1 \leq i < k).\end{aligned}$$

4. If $k = p$, terminate. Otherwise, increment k and repeat from step 2.

1.4 Detection of repeatable patterns

To detect repeatable patterns in time series data, we can use several techniques. As a basic method Autocorrelation is used to measures the correlation between the time series data at different time lags 1.3.3. The next one which is mainly use is th Seasonal decomposition 1.4.3 which separates the time series data into different components such as trend, seasonal, and residual. To analyse the frequency components of the time series data Fourier transform 1.4.2 can be used. Same as another mathematical technique to analyse the time-frequency structure of the time series data called Wavelet analysis. To detect pattern in seasonal trends Hidden Markov models 1.4.1 can be used. These models assume that the data follows a probabilistic process, where the state of the process is not directly observable. The last one method which is as much as used is the Pattern Masking for Dictionary Matching (PMDM) 1.4.4.

These techniques can be used individually or in combination to detect repeatable patterns in time series data. It's important to note that the choice of technique depends on the characteristics of the data and the specific problem being addressed. The shift of long-term linear prediction for example in human speech refers to the way in which the human vocal system produces sounds over time. Specifically, it describes the changes that occur in the speech signal over relatively long periods of time, typically measured in tens to hundreds of milliseconds. Long-term linear prediction is a technique used in speech analysis and synthesis to model the way that the vocal system produces speech sounds. It involves breaking the speech signal into short segments and using mathematical algorithms to predict the signal in each segment based on the signal in preceding segments. The shift of long-term linear prediction in human speech refers to the fact that the vocal system is constantly adjusting and adapting the way that it produces sounds based on a variety of factors, including the phonemes being spoken, the speaker's emotions and intentions, and the context of the speech. This means that the speech signal is not static, but rather is constantly shifting and evolving over time. For example, when a speaker is emphasizing a particular word or phrase, they may change the pitch, intensity, or duration of certain sounds in order to convey their meaning more effectively. These changes will be reflected in the long-term linear prediction of the speech signal, which will show shifts in the predicted signal over time [21].

In the context of stock data analysis, the shift in long-term prediction refers to the way in which the patterns in the data change over time, reflecting changes in

the underlying market conditions and investor sentiment. Long-term linear prediction is a technique that can be used to model and forecast stock prices based on historical price data. It involves breaking the time-series data into short segments and using mathematical algorithms to predict the price in each segment based on the price in preceding segments. The shift in long-term prediction for stock data can occur due to a variety of factors such as changes in the macroeconomic environment, company earnings announcements, news events, investor sentiment, and other market-moving events. These shifts can cause changes in the underlying trends and patterns in the data, which can make it difficult to accurately predict stock prices over longer time horizons. For example, sudden changes in market conditions such as a global financial crisis or a political event can cause significant shifts in long-term prediction for stock data, making it difficult to accurately forecast future prices. Similarly, a company's financial performance or regulatory changes can cause sudden shifts in the long-term prediction for that company's stock. Overall, the shift in long-term prediction for stock data reflects the dynamic and unpredictable nature of the stock market, and underscores the importance of using multiple sources of data and analysis techniques to make informed investment decisions [24].

1.4.1 Hidden Markov models

Hidden Markov Models (HMMs) are probabilistic models that are used to analyse sequential data where the state of the system is not directly observable. HMMs consist of a set of hidden states that are not directly observable, and a set of observable symbols that are emitted from the hidden states [25].

The HMM assumes that the hidden states form a Markov chain, which means that the probability of transitioning from one hidden state to another depends only on the current state and not on the previous states. The emission probabilities of the observable symbols depend only on the current hidden state.

HMMs are typically used for two main tasks:

- **Evaluation:** Given a sequence of observable symbols, HMMs can be used to evaluate the probability of the sequence of symbols occurring, given a particular set of model parameters.
- **Decoding:** Given a sequence of observable symbols, HMMs can be used to

determine the most likely sequence of hidden states that generated the sequence of symbols.

- HMMs are used in a wide range of applications such as speech recognition, handwriting recognition, natural language processing, bioinformatics, and financial time series analysis.

The basic steps involved in building an HMM model are as follows:

1. Define the set of hidden states: Determine the set of hidden states that the system can be in at any given time.
2. Define the set of observable symbols: Determine the set of observable symbols that can be emitted from each hidden state.
3. Define the transition probabilities: Determine the probability of transitioning from one hidden state to another.
4. Define the emission probabilities: Determine the probability of emitting each observable symbol from each hidden state.
5. Estimate the model parameters: Estimate the transition and emission probabilities of the HMM using a training dataset.
6. Use the model for evaluation or decoding: Once the model parameters have been estimated, the HMM can be used for evaluation or decoding tasks.

It's important to note that building an HMM model requires knowledge of probability theory and statistical modelling, and may require significant computational resources.

1.4.2 Fourier transform

Fourier transform is a mathematical technique that is used to analyse the frequency components of a signal or a time series data. It transforms a signal from the time domain into the frequency domain, which allows us to analyse the signal in terms of its frequency components. The Fourier transform is widely used in many fields, including signal processing, image processing, and physics.

The Fourier transform decomposes a signal into a set of sinusoidal waves with different frequencies, amplitudes, and phases. These sinusoidal waves are called the Fourier series, and they represent the signal as a sum of complex exponential

functions.

The Fourier transform can be expressed as an integral equation that takes a time-domain signal and produces a frequency-domain representation of that signal. The formula for the Fourier transform is:

$$F(\omega) = \int f(t)e^{(-i\omega t)}dt, \quad (1.9)$$

where $F(\omega)$ is the frequency-domain representation of the signal, $f(t)$ is the time-domain signal, ω is the frequency, and i is the imaginary unit.

The inverse Fourier transform can be used to transform the frequency-domain representation of a signal back into the time domain. The formula for the inverse Fourier transform is:

$$f(t) = \frac{1}{2\pi} \int F(\omega)e^{(i\omega t)}d\omega, \quad (1.10)$$

where $f(t)$ is the time-domain signal, $F(\omega)$ is the frequency-domain representation of the signal, ω is the frequency, and i is the imaginary unit.

The Fourier transform has many practical applications. For example, it can be used to analyse the frequency components of a music signal, to filter out noise from a signal, or to compress data by removing high-frequency components that are not essential for the representation of the signal. It is a powerful tool for understanding the underlying structure of signals and is widely used in scientific research and engineering applications.

1.4.3 Seasonal decomposition

Seasonal decomposition is a statistical technique that is used to decompose a time series into its underlying components, including trend, seasonal, and residual components. This technique is useful for identifying and understanding the repeating patterns or seasonal effects in a time series data.

The seasonal decomposition of a time series involves separating the data into four components:

- Trend component represents the long-term pattern in the data, such as increasing or decreasing trends over time.

- Seasonal component represents the repeating pattern in the data that occurs over a fixed period, such as daily, weekly, or monthly patterns.
- Residual component represents the remaining variation in the data that cannot be explained by the trend or seasonal components. It may include random noise or other unexplained factors.
- Irregular component represents any unexpected or irregular variation in the data that is not accounted for by the other components.

The seasonal decomposition process involves applying a smoothing algorithm to the time series data to estimate the trend and seasonal components, and then subtracting them from the original data to obtain the residual component. There are various methods for performing seasonal decomposition, including moving averages, exponential smoothing, and regression models.

Once the components of the time series have been separated, they can be analysed and modeled separately. This can help in detecting the repeating patterns or seasonal effects in the data and in making accurate forecasts for future time periods.

Overall, seasonal decomposition is a powerful tool for identifying and analyzing the seasonal patterns in a time series data, and it can be useful in a wide range of applications, including finance, economics, and meteorology.

1.4.4 Pattern Masking for Dictionary Matching

Pattern masking is a technique used in dictionary matching to find occurrences of words or phrases in a given text. In this technique, a pattern is defined as a string of characters that represents the word or phrase being searched for. However, in some cases, the exact spelling of the word or phrase may not be known, or there may be variations in the spelling that need to be accounted for.

To handle such scenarios, pattern masking is used. In pattern masking, certain characters in the pattern are replaced with special characters that can match a range of different characters. For example, the asterisk (*) character can be used to represent any number of characters, while the question mark (?) can represent a single character.

Here's an example to illustrate how pattern masking works. Let's say we want to find occurrences of the word "colour" in a given text, but we want to account for variations in spelling such as "color" or "colours". We can define a pattern for this as "colours", where the question mark represents a single character that may or may not be present.

By using pattern masking, we can search for occurrences of the pattern "colou?rs" in the text, and it will match any of the variations of the word "colour" that we specified in the pattern.

Dictionary matching algorithms can use pattern masking to search for multiple words or phrases simultaneously, which can be useful for tasks such as text classification, information retrieval, or sentiment analysis [26].

2 Goal of the thesis

In this thesis we are focused on improving linear prediction in sales and financial forecasting and combine it with modern machine learning approaches. Machine learning techniques are used to improve the accuracy of linear prediction models. Specifically, there are two ways in which machine learning is applied to linear prediction:

1. Feature engineering¹: In this approach, machine learning is used to extract relevant features from the input signal that can be used as input to a linear prediction model. The extracted features can capture complex patterns in the data that are not captured by the raw input. Feature engineering can be done using techniques such as principal component analysis, wavelet transform, and Fourier transform.
2. Model selection and training: In this approach, machine learning is used to select the best linear model for the prediction task and to estimate its parameters from the data. This can involve selecting the best set of input variables for the linear model, choosing the best regularization parameter to avoid overfitting, and optimizing the model hyperparameters. Common machine learning algorithms used for linear prediction include linear regression, support vector regression, and artificial neural networks.

The goal of long-term linear prediction is to estimate future values of a signal or time series based on its past values using a linear model. The linear prediction model uses a set of coefficients to weight past values of the signal and produce a prediction for future values. The accuracy of the prediction depends on the quality of the model and the complexity of the underlying signal. Combining these principles offers the best results from linear prediction on the sales and financial datasets.

¹Feature engineering involves the extraction and transformation of variables from raw data, such as price lists, product descriptions, and sales volumes so that you can use features for training and prediction

3 Methodology

3.1 Characteristics of the research object

Sales data and financial datasets are two broad categories of data that have different characteristics and are used for different purposes. Here are some general characteristics of each type of dataset:

Sales data

Typically contains transactional information, such as the date, time, location, and amount of a purchase. Can include additional information about the customer, such as their demographic profile, purchase history, and preferences. Often analyzed to understand customer behavior, such as buying patterns, trends, and preferences. May have seasonal or cyclical patterns, depending on the nature of the product or service being sold. Can be used to optimize marketing and sales strategies, such as targeting specific customer segments, promoting certain products, or adjusting prices and discounts.

Financial datasets

Typically contains financial information, such as the revenue, expenses, assets, liabilities, and cash flow of a company or organization. Can include additional information about the market, such as interest rates, exchange rates, and stock prices. Often analyzed to evaluate the financial performance and health of a company, such as profitability, solvency, and liquidity. May have regulatory or compliance requirements, such as financial reporting standards or tax laws. Can be used to make strategic decisions, such as investment, merger and acquisition, or divestiture. Both sales data and financial datasets can be used for forecasting and modeling, but they have different analytical techniques and tools. Sales data often requires customer segmentation, predictive analytics, and machine learning algorithms, while financial datasets require financial ratio analysis, time series forecasting, and risk assessment.

3.2 Methods

In our thesis was used mainly these list of methods like a linear prediction, regression, backpropagation or statistical measures.

3.2.1 Linear prediction

Linear prediction is a statistical technique used to forecast future values based on past observations. It is a method for modeling the relationship between a dependent variable and one or more independent variables in a linear form. The goal of linear prediction is to find the best linear approximation of the relationship between the variables, which can then be used to make predictions about future values of the dependent variable.

In short-term linear prediction, the focus is on accurately predicting the next few values of the dependent variable, typically in the range of several weeks to a few months. The linear prediction models used for short-term forecasting are typically simple and straightforward, often using a small number of independent variables. The goal is to provide a quick and easily interpretable forecast that can be used to make operational decisions in the short-term. Consider a signal $x(n)$ that is modeled using a linear predictor. The linear predictor is defined as:

$$\hat{x}(n) = \sum_{i=1}^p a_i x(n-i), \quad (3.1)$$

Long-term linear prediction refers to the use of linear prediction techniques to make predictions about the future values of a time series over an extended period of time, typically several months to several years. Unlike short-term linear prediction, which focuses on forecasting the near-term future, long-term linear prediction aims to provide a more comprehensive and accurate forecast of future values. On the figure 1.10 you can see the explanation of the long-term linear prediction shift.

$$\hat{x}(n) = \sum_{i=1}^p a_i x(n-i) + \sum_{i=-q}^q b_i x(n-i-Q), \quad (3.2)$$

3.2.2 Backpropagation

Backpropagation is a widely used algorithm for computing the gradients of the loss function with respect to the weights of a neural network. It is the backbone of training deep neural networks using stochastic gradient descent (SGD) or its variants. the backpropagation algorithm computes the gradient of the loss function with respect to each weight in the network by recursively applying the chain rule of differentiation. the algorithm is typically implemented in two phases:

1. Forward pass: the forward pass involves computing the output of each layer in the network, starting from the input layer and propagating through the hidden layers to the output layer. the output of each layer is computed as a function of the input to the layer and the weights of the layer. the forward pass computes the predictions of the network on a given input.
2. Backward pass: the backward pass involves computing the gradients of the loss function with respect to each weight in the network, starting from the output layer and propagating backwards through the hidden layers to the input layer. The gradients are computed by applying the chain rule of differentiation to the output of each layer. The gradients are then used to update the weights of the network using an optimization algorithm such as SGD.

the backpropagation algorithm can be optimized using various techniques, such as parallel computing, weight sharing, and regularization. It is a powerful tool for training deep neural networks with many layers and millions of parameters, and has enabled significant advances in many areas of machine learning, including computer vision, natural language processing, and speech recognition.

3.3 Datasets

Creating a science research dataset involves several steps, which may vary depending on the nature of the research and the specific field of science. Here are some general steps to consider:

1. Determine the research question: the first step in creating a research dataset is to determine the research question. This will help we identify the data we need to collect and the type of dataset we need to create.
2. Choose the data sources: Once we have identified the research question, we need to determine the data sources we will use to create our dataset.

Depending on the research question, we may use publicly available data, data from surveys or experiments, data from literature, or a combination of these.

3. Collect and organize the data: Collecting data can involve different methods, such as surveys, experiments, literature reviews, or data mining. We need to make sure that the data we collect is reliable, accurate, and relevant to we research question. Once you have collected the data, we need to organize it in a way that is easy to analyze.
4. Clean and preprocess the data: Before we can analyze the data, we need to clean and preprocess it. This involves removing any errors, missing values, or duplicates in the data. We may also need to transform or normalize the data to make it compatible with the analysis methods we plan to use.
5. Perform exploratory analysis: Once the data is cleaned and preprocessed, we can perform exploratory analysis to identify patterns, trends, or relationships in the data. This can help us refine our research question or identify areas that require further investigation.
6. Perform statistical analysis: Depending on the research question, we may need to perform statistical analysis to test hypotheses or evaluate the significance of the results. This can involve using regression analysis, hypothesis testing, or other statistical methods.

Validate the dataset: After we have analyzed the data, we need to validate the dataset to ensure that the results are accurate and reliable. This can involve comparing the results to other datasets or conducting further experiments to verify the results. Overall, creating a science research dataset involves careful planning, data collection, preprocessing, and analysis to ensure that the data is accurate, reliable, and relevant to the research question.

3.4 Comparison criteria

Finally we define the method to compare our models results. Absolute number of income value prediction should not be important for the store owners because of that we calculated the aberration for each month prediction and then we easily calculate quarterly and yearly results. the sum of squared errors (SSE), defined

by:

$$SSE = \sum_{i=1}^n w_i (y_i - \bar{y}_i)^2,$$

between the fitting models and the used data serves as the fitting criterion, with values closer to 0 indicating a smaller random error component of the model. Also some other quality measures were evaluated, *i.e.* the R-square from interval $[0, 1]$, that indicates the proportion of variance satisfactory explained by the fitting-model (*e.g.* R-square = 0.7325 means that the fit explains 73.25% of the total variation in the data about the average); R-square is defined as the ratio of the sum of squares of the regression (SSR) and the total sum of squares (SST). SSR is defined as

$$SSR = \sum_{i=1}^n w_i (\bar{y}_i - \bar{y})^2.$$

SST is also called the sum of squares about the mean, and is defined as

$$SST = \sum_{i=1}^n w_i (y_i - \bar{y})^2,$$

where $SST = SSR + SSE$. Givenm these definition, R-square is expressed as

$$\frac{SSR}{SST} = 1 - \frac{SSE}{SST}.$$

The adjusted R-square statistic, with values smaller or equal to 1, where values closer to 1 indicate a better fit; the root mean squared error (RMSE):

$$RMSE = s = \sqrt{\frac{SSE}{v}}$$

with values closer to 0 indicating a fit more useful for prediction [27].

Mean square error (MSE) is a commonly used metric to measure the average squared difference between the actual and predicted values of a continuous variable. It is a measure of how well a prediction model performs in predicting continuous outcomes.

To calculate the MSE, we first take the difference between the predicted and actual values for each data point, square these differences, and then take the average of the squared differences across all data points.

The formula for calculating MSE is:

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2, \quad (3.3)$$

where n is the total number of data points, y_i is the actual value of the i – th data point, and \hat{y}_i is the predicted value of the i – th data point.

the square of the difference in each data point ensures that the errors are positive and provides a way to measure the magnitude of the error. By squaring the errors, the MSE gives more weight to larger errors and less weight to smaller errors.

MSE is a popular metric used in many fields, such as machine learning, statistics, and engineering, to evaluate the performance of prediction models. a lower MSE indicates better performance of the model in predicting the outcomes.

3.5 Statistical measures

Median

The median is a statistical measure that represents the middle value of a dataset. It is a value that separates the dataset into two halves: half of the values are greater than the median, and half of the values are less than the median. In other words, the median is the value that is exactly in the middle of the dataset when the values are arranged in order of magnitude.

To compute the median of a dataset, we first sort the values in ascending or descending order. If the dataset has an odd number of values, the median is the middle value. For example, in the dataset 1, 2, 3, 4, 5, the median is 3. If the dataset has an even number of values, the median is the average of the two middle values. For example, in the dataset 1, 2, 3, 4, the median is $(2 + 3)/2 = 2.5$.

$$Median = \frac{x(n/2) + x(n/2 + 1)}{2}, \quad (3.4)$$

where n is the number of observations and x is a vector of observation values.

The median is a robust statistic, meaning that it is not affected by outliers or extreme values in the dataset, unlike the mean. This makes it a useful measure of central tendency in datasets with a large number of outliers or skewed distributions. The median is commonly used in various applications, such as finance, economics, and social sciences, to summarize and compare datasets.

Standard deviation

Standard deviation is a statistical measure that quantifies the amount of variation or dispersion in a dataset. It measures how far the values in a dataset deviate from

the mean, or average, of the dataset. The standard deviation is a non-negative number and has the same units as the data being measured.

$$SD = \sqrt{\frac{\sum |x - \mu|^2}{N}}, \quad (3.5)$$

where x is a value in the data set, μ is the mean of the data set, and N is the number of data points in the population.

To compute the standard deviation of a dataset, we first calculate the mean of the dataset. Then, we calculate the difference between each value in the dataset and the mean, square each difference, and sum up the squared differences. Finally, we divide the sum of squared differences by the number of values in the dataset, and take the square root of the result. This gives us the standard deviation of the dataset.

The small standard deviation indicates that the values in the dataset are tightly clustered around the mean, while a large standard deviation indicates that the values are widely spread out from the mean. Standard deviation is commonly used in various applications, such as finance, engineering, and natural sciences, to analyze and compare datasets. It is also an important parameter in many statistical tests and models, such as the normal distribution and the t-test.

4 Syntactic part

Based on the chapter 1 let us create new mathematical models and approaches to made a fast and accurate sales forecasting consists of long-term linear prediction with individual weights calculated for each period all based on Levinson-Durbin scheme caled Extended linear prediction (ELP). We expect to get better results than by using prediction based on short-term or standard long-term linear prediction (see section 1.3). Finally, our approach will return future number of orders for sales companies based on previous data with better aberration than linear prediction has.

4.1 Novel approach to forecast sales data

In this chapter we prepare sub-parts of our novel aproache to forecast orders in sales companies. Here are the steps how to made our aproache function and return accurate results for number of orders.

1. Run neural network to get right order of linear predictor. In this section we have trained Neural network from section 4.3 and detect correct order from our neural network really fast.
2. Run neural network to get righ shift and long shift of linear prediction 4.4. Than ve can run our next neural network trained for detect shift for our prediction. In this neural network as one of the inputs we are using results from previous neural network.
3. Run extended long-term prediction 4.2 to ger predicted values of sales data. When we have optimal order and shift we are able to run linear prediction in a standard way.
4. Apply periodical wages 4.5 to previous response data. Results from previous step are corrected by the weights vector which is calculated over the trained dataset.

5. Calculate comparison criteria for results and plot data and mean square errors. When we run all the models we are able to calculate comparison criteria from 3.4 and get a table of results to plot errors and compare predicted and measured orders from validation dataset.

4.2 Extended long-term prediction

For my purpose i created extended long-term prediction which handle the seasonality and repeated patterns in economics data. This correction mechanism is used to reflect historical peaks in graphs thru dataset. This simple correction increase accurate of the model and get better response due tue psychology, sociology and marketing aspects in dataset.

To set up periodical weights its needs to create a vector of correction parameters from original dataset using median and standard deviation statistical parameters from dataset. As a base equation we will user equation for long-term prediction3.2 with the new weights and get:

$$\hat{x}(n) = \left(\sum_{i=1}^p a_i x(n-i) + \sum_{i=1}^q b_i x(n-i-Q) \right) \gamma(n), \quad (4.1)$$

where $\hat{x}(n)$ is the predicted value of the order at time n , $x(n-i)$ are the past short-term prediction part p samples of the dataset, $x(n-Q-i)$ are the past long-term prediction part with seasonal shift Q , and a_i and b_i are the predictors coefficients. The order of the predictor is p for short-term and q for the long-term linear prediction. The seasonal weights are represented by $\gamma(n)$.

When building a long-term prediction model, the use of weights in this use case have a significant effect on the model's performance. Weights are used to assign different levels of importance to different features or inputs in the model.

In a long-term prediction model, the weight assigned to each input can determine its impact on the model's predictions. Using weights in order prediction model, the weight assigned to a historical company's financial data are more important than the weight assigned to its historical visitors. This is why we are ignore this data for prediction.

If the weights are not carefully chosen, the model may give too much impor-

tance to some inputs and not enough to others, leading to inaccurate predictions. Therefore, selecting the right weights is crucial for creating an accurate and effective long-term prediction model. In our case it reduce the prediction error and returns more usable data for order prediction.

In addition, the choice of weights can also affect the model's ability to adapt to changing conditions over time. a model with carefully chosen weights can be more resilient and adaptable to changes in the input data, while a model with poorly chosen weights may be less flexible and struggle to adjust to new information. So it's really important to set correct shift and period to recalculation of the shift. This is ride by neural network from 4.4.2.

Overall, the use of weights is an important consideration in building our predictive model, and careful selection can significantly improve the accurate and effectiveness of our long-term prediction model.

4.3 Prediction order estimation

This section describes the created Neural Network which was used to detect optimal order of predictor (long-term and short-term) for the dataset to get the best results of final forecasting.

Neural network for prediction order estimation

Creating a neural network for linear prediction and optimal order detection involves several steps. Here's a general overview of the process:

1. **Data Preparation:** Collect a dataset of input/output pairs that represent the relationship we want the neural network to learn. For linear prediction, this could be a time-series dataset where we want to predict the next value based on the previous values. For optimal order detection, this could be a dataset where we have inputs and the corresponding optimal order.
2. **Data Preprocessing:** Preprocess the dataset by normalizing the inputs and outputs to a common range, shuffling the dataset, and splitting it into training, validation, and testing sets.
3. **Model Architecture:** Choose an appropriate neural network architecture for our problem. For linear prediction, we could use a recurrent neural net-

work (RNN) such as LSTM or GRU. For optimal order detection, we could use a feedforward neural network. Choosing the right activation function for linear prediction order detection in neural networks depends on several factors such as the type of data, the complexity of the problem, and the architecture of the neural network. In our approach we test some most common used activation functions to test and choose the right one. Linear activation function was not return good results because of our problem is not linear. ReLU activation function is one of the most popular activation functions used in deep learning. Sigmoid activation function maps any input value to a value between 0 and 1 so for our problem not good because of we are not solving binnary problems. In general, it's a good idea to start with the ReLU activation function and see if it works well for our problem.

4. Training: Train the neural network on the training set using an appropriate optimization algorithm such as stochastic gradient descent (SGD), Adam, or RMSProp. During training, monitor the performance of the network on the validation set to detect overfitting and adjust the hyperparameters accordingly. To training our neural network we used training dataset created from our main dataset as half a year order data. Then we feed our neural network by that dataset to adjusting the weights of its connections between neurons to minimize the difference between the network's output and the correct output for each input. This process is known as backpropagation. The training process is typically repeated many times, with the network being presented with different sets of data each time, until it reaches a satisfactory level of accuracy on a validation set of data. Once the network is trained, it can be used to make predictions or decisions on new, unseen data.
5. Testing: Evaluate the performance of the trained neural network on the testing set to get an estimate of its generalization ability.
6. Using neural network to get optimal order: Deploy the trained neural network to make predictions on new data. Our trained Neural network is consist of 4 layers with 40 neurons in input layer than two hidden layers with 20 and 10 neurons and 5 neuron in output layer. Our trained network use ReLU activation function. Training was made over 120 of observations. Once a neural network is trained, there are several steps we can take to get the best performance out of it:

- a Use the validation set to tune hyperparameters: the validation set is a dataset that is separate from the training and testing sets and is used to tune hyperparameters such as learning rate, batch size, and number of epochs. By experimenting with different hyperparameters, we can find the optimal settings that result in the best performance.
- b Test the model on a separate dataset: Once the model is trained and the hyperparameters are tuned, it's important to test the model on a separate dataset to evaluate its performance. This dataset should be completely separate from the training and validation sets.
- c Use data augmentation techniques: Data augmentation techniques such as rotation, translation, and scaling can be used to generate additional training data, which can improve the performance of the model.
- d Use ensembling techniques: Ensembling techniques such as bagging and boosting can be used to combine the predictions of multiple models to improve the overall performance.
- e Use regularization techniques: Regularization techniques such as L1 and L2 regularization can be used to prevent overfitting and improve the generalization of the model.
- f Fine-tune the model on new data: If new data becomes available, the model can be fine-tuned on the new data to further improve its performance.
- g By following these steps, we can get the best performance out of a trained neural network. It's important to remember that there is no single approach that works best for all problems, and experimentation is often necessary to find the optimal solution.

4.4 Long-term prediction shift estimation

In this section we see second Neural Network which replace the basic Autocorrelation approach. This neural network will be used for long-term linear prediction.

4.4.1 Autocorrelation approach

1. Compute the autocorrelation function of the signal using the `xcorr` function in MATLAB. The `xcorr` function returns the cross-correlation sequence of the signal with itself.

$$[Rxx, lag] = xcorr(x, maxlag);$$

where x is the input signal and $maxlag$ is the maximum lag to compute. Rxx is the cross-correlation sequence, and lag is the corresponding lag vector.

2. Find the index of the maximum value in the autocorrelation function. We can use the `max` function to find the maximum value and its index.

$$[val, idx] = max(Rxx);$$

val is ignored value of the maximum, and idx is the index of the maximum value in the Rxx vector.

3. Calculate the shift value from the lag index. The shift value is simply the lag value corresponding to the maximum value in the autocorrelation function.

$$shift = lag(idx);$$

The shift value is the lag at which the autocorrelation function has its maximum value, which corresponds to the shift value for long-term linear prediction. Note that the shift value is expressed in samples, so we may need to convert it to a time delay if our signal has a specific sampling rate.

4.4.2 Neural network approach for shift estimation

To create a neural network for shift and long shift prediction, we can follow these steps:

1. Define the problem: the first step is to define the problem we want to solve. In this case, the problem is to predict the shift and long shift of a time series data.
2. Collect and preprocess the data: Collect the data that we want to use for training and testing the neural network. Preprocess the data by normalizing it and splitting it into training, validation, and testing sets.
3. Choose the architecture: Choose the architecture for our neural network based on the problem we want to solve. In this case, a recurrent neural network (RNN) architecture such as a Long Short-Term Memory (LSTM)

or Gated Recurrent Unit (GRU) may be suitable, as they are specifically designed to handle sequential data. Choosing the activation function for a neural network for shift and long shift prediction depends on the architecture of the network and the problem being solved.

For a recurrent neural network (RNN) architecture such as Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU), the most commonly used activation function is the hyperbolic tangent (\tanh) function. This is because the \tanh function produces outputs in the range $[-1, 1]$, which makes it suitable for RNNs that use recurrent connections to store and update memory over time.

However, other activation functions such as the Rectified Linear Unit (ReLU) or its variants, such as leaky ReLU or exponential ReLU, can also be used in RNNs for shift and long shift prediction. These activation functions are computationally efficient and can be used in deep neural networks without the problem of vanishing gradients.

In general, the choice of activation function should be based on the characteristics of the problem being solved, the architecture of the network, and the properties of the activation function itself. Experimentation with different activation functions can help to determine the most appropriate one for a given problem.

4. Define the input and output: Define the input and output of our neural network. The input should be the time series data, and the output should be the predicted shift and long shift.
5. Define the loss function: Choose an appropriate loss function to measure the difference between the predicted shift and long shift and the actual shift and long shift.
6. Train the model: Train the model using the training data and tune the hyperparameters to improve the performance of the model. Monitor the model's performance using the validation set. To train our neural network, we utilized a training dataset derived from our main dataset consisting of six months of order data. The neural network was fed this dataset in order to adjust the weights of its connections between neurons and minimize the difference between the network's output and the correct output for each input, a process commonly referred to as backpropagation.

It's important to experiment with different architectures and hyperparameters to find the best model for our data. Additionally, it's important to properly preprocess the data and monitor the model's performance to avoid overfitting. With these steps, we can train a neural network for shift and long shift prediction.

7. Evaluate the model: Evaluate the performance of the model on the testing set to see how well it generalizes to new data.
8. Fine-tune the model: If the model does not perform well on the testing set, fine-tune the model by adjusting the architecture, hyperparameters, or training data.
9. Use the model: Once we have a well-performing model, we can use it to predict the shift and long shift of new time series data.

Run neural network to get optimal shift and long shift

By following these steps, we can create a neural network for shift and long shift prediction. It's important to experiment with different architectures and hyperparameters to find the best model for our data.

Our trained Neural network is consist of 4 layers with 40 neurons in input layer than two hidden layers with 17 and 8 neurons and 6 neuron in output layer. Our trained network use leaky ReLU activation function. Training was made over 120 of observations.

To run a trained neural network for shift and long shift prediction, we can follow these steps:

1. Preprocess the data: If we have new data that we want to use for prediction, preprocess it in the same way as the training data by normalizing it and formatting it into sequences.
2. Load the trained model: Load the trained model into memory using a deep learning framework such as Tensorflow, PyTorch, or Keras.
3. Prepare the input data: Prepare the input data by formatting it into sequences that match the input shape of the model.

4. Make predictions: Use the predict function of the model to make predictions on the input data. the output of the model will be the predicted shift and long shift values.
5. Postprocess the output: If necessary, postprocess the output of the model by denormalizing it or transforming it back into the original format.
6. Evaluate the predictions: Evaluate the quality of the predicted shift and long shift values by comparing them with the actual shift and long shift values. Calculate performance metrics such as mean squared error or mean absolute error to quantify the accurate of the predictions.
7. Refine the model: If the predicted shift and long shift values are not accurate enough, refine the model by adjusting the architecture, hyperparameters, or training data and repeat the process.

With these steps, we can run a trained neural network for shift and long shift prediction and obtain optimal shift and long shift values. It's important to note that the quality of the predictions depends on the quality and quantity of the data used for training and the architecture and hyperparameters of the model. Therefore, it's important to experiment with different models and training data to find the optimal solution.

4.5 Estimation of the seasonal weights

To create a mathematical model for predicting sales data with periodical trends, we can use a seasonal ARIMA (SARIMA) model. This model takes into account seasonal variations in the data and uses autoregressive and moving average terms to capture the patterns and trends in the data.

To create the weights for the SARIMA model, we can follow these steps:

1. Identify the seasonal period: Determine the seasonal period of the sales data. This can be done by analyzing the autocorrelation function (ACF) and partial autocorrelation function (PACF) of the data.
2. Estimate the parameters: Estimate the parameters of the SARIMA model using the sales data. This can be done using statistical software such as R or Python.

3. Interpret the weights: Once the parameters have been estimated, interpret the weights of the model to understand how they contribute to the prediction. The weights represent the strength of the relationship between the sales data and the predictor variables.
4. Validate the model: Validate the model by comparing the predicted sales data with the actual sales data. Use statistical measures such as mean squared error or mean absolute error to evaluate the accuracy of the predictions.
5. Refine the model: Refine the model by adjusting the weights or the parameters of the model based on the validation results. Repeat the validation process until the model provides accurate predictions.

It's important to note that creating a SARIMA model can be complex and requires knowledge of time series analysis and statistical modeling. Therefore, it's recommended to seek the help of a data scientist or a statistician for assistance.

Based on these steps we prepared the vector of seasonal weights γ for each record in our dataset using median and standard deviation 3.5 to create the equation:

$$\gamma_i = 1 + \left(x_i - m - \frac{S}{p} \right) c, \quad (4.2)$$

where γ_i is the weight for each record for our dataset, for our purpose is the day weight, x_i is the value from dataset, m and S are the median and standard deviation of our dataset, p is the order of our linear prediction model and the c is correction coefficient which was set up experimentally.

5 Evaluation

Lets prepare the experiment to validating a long-term linear prediction model for online store orders. We will use experimentation by these steps:

1. Define the problem and objectives: Clearly define the problem and objectives that the long-term linear prediction model is intended to solve. In this case, the objective is to predict online store orders over a long period of time using a linear model.
2. Gather data: Collect historical data on online store orders, such as order dates, order amounts, and other relevant variables. This data will be used to train and validate the long-term linear prediction model. Our data was collect from online system storePredictor which is available on storepredictor.com and data are anonymized and pseudonimized to be used for next calculations.
3. Prepare the data: Clean and preprocess the data to ensure that it is consistent and suitable for use in the long-term linear prediction model. This may involve tasks such as removing duplicates, handling missing values, and transforming variables as needed. Preprocessing of our data is described in 5.1
4. Develop the long-term linear prediction model: Using the historical data, develop a long-term linear prediction model that can forecast online store orders over a specified time period, which is detailed described in 4.2 and practically applied in 5.2 over a specified time period, described in 5.
5. Evaluate the results: Analyze the results of the experiment to determine the accuracy of the long-term linear prediction model. This will involve calculating metrics such as the mean square error (MSE), r-squared (R^2) and the root mean square error (RMSE) 5.3.

6. During solving previous steps the model was redefined and revalidate when the results are not satisfactory, refine the model and repeat the validation process until an accurate and effective long-term linear prediction model is developed.

In summary, to prepare an experiment to validate a long-term linear prediction model for online store orders, we need to define the problem and objectives, gather and prepare the data, develop the model, validate it, evaluate the results, and refine and revalidate the model if necessary.

5.1 Preprocessing of input data

Preprocessing the data is an important step in preparing the data for a linear order prediction model. Here are some common steps to preprocess the data for linear order prediction we used:

1. Data cleaning: Remove any irrelevant data or duplicate records in the dataset. In our purpose is to clean the unsuccessful orders, returned orders and fraud orders from competitors to detect power of the store.
2. Handling missing data: for fill in any missing values in the dataset machine learning algorithms K-Nearest Neighbors (KNN) 1.1.4 will be used to check the dataset.
3. Feature selection: Select the relevant features (predictors) that are likely to have a strong influence on the order prediction. This may include variables such as customer demographics, purchase history, product attributes, and marketing campaigns.
4. Feature scaling: Scale the features so that they are on the same scale to ensure that each feature has equal importance. This can be done using normalization or standardization techniques.
5. Handling categorical variables: Convert categorical variables into numerical values using techniques such as one-hot encoding, ordinal encoding, or label encoding.
6. Dimensionality reduction: If the dataset contains many features, use dimensionality reduction techniques such as principal component analysis (PCA) or linear discriminant analysis (LDA) to reduce the number of features and simplify the model.

7. Handling outliers: Detect and handle any outliers in the dataset using appropriate techniques such as Z-score, Tukey's method, or machine learning algorithms such as Isolation Forest or Local Outlier Factor.

Overall, the goal of this steps for linear order prediction experiment is to ensure that the data is clean, complete, and properly formatted for use in the linear prediction model. This helps to improve the accuracy and effectiveness of the model in predicting online store orders.

5.2 Mathematical model for sales forecasting

Let's prepare 4 models to test our theory and result expectation. Set up MATLAB live script to prepare our dataset, calculate statistical variables and develop several versions of linear predictors to get results of our experiment. We will use this models:

1. Short-term linear prediction Based on linear prediction 1.3.1 we calculate optimal coefficients by 1.3.4 and made prediction over our dataset. This model is described in equation 1.1.

$$\hat{x}(n) = \sum_{i=1}^p a_i x(n-i), \quad (5.1)$$

where $\hat{x}(n)$ is the predicted value of $x(n)$, p is the order of the predictor, and a_i are the predictor coefficients.

2. Extended short-term linear prediction This model is based on linear prediction 1.3.1 with optimal parameters by levinson schema 1.3.4. To compare our developed model and we will prepare similar principle as in equation 4.1 to the short-term linear prediction with results.

$$\hat{x}(n) = \sum_{i=1}^p a_i x(n-i) \gamma(n), \quad (5.2)$$

where $\hat{x}(n)$ is the predicted value of $x(n)$, p is the order of the predictor, and a_i are the predictor coefficients. The predictor. The seasonal weights is represent by $\gamma(n)$

3. Long-term linear prediction To compare more linear prediction and check theorem about application of long-term prediction on sales data we will

prepare the model based on 1.3.2 and use the equation 3.2.

$$\hat{x}(n) = \sum_{i=1}^p a_i x(n-i) + \sum_{i=-q}^q b_i x(n-i-Q), \quad (5.3)$$

where $\hat{x}(n)$ is the predicted value of the signal at time n , $x(n-i)$ are the past p samples of the signal, $x(n-i-Q)$ are the past q shifted samples of the signal, and a_i id the short-term predictor coefficients and b_i is the long-term predictor koeficient. The order of the predictor is p for the autoregressive (AR) part and q for the long-term part. The pitch period T can be obtained from the autocorrelation function [18].

4. Extended long-term linear prediction the last model in our comparison will be our developed model which is based on long-term linear prediction describe in 1.3.2 and use weights coefficients 4.5 to get better results in economics data. This model is describe in equation 4.1.

$$\hat{x}(n) = \left(\sum_{i=1}^p a_i x(n-i) + \sum_{i=1}^q b_i x(n-i-Q) \right) \gamma(n), \quad (5.4)$$

where $\hat{x}(n)$ is the predicted value of the order at time n , $x(n-i)$ are the past short-term predition part p samples of the dataset, $x(n-i-Q)$ are the past long-term prediction part with seasonal shift Q , and a_i and b_i are the predictors coefficients. The order of the predictor is p for short-term and q for the long-term linear prediction. The seasonal weights is represent by $\gamma(n)$.

All the models will be used dataset prepared in 5.1 and results of prediction is described in 5.3.

5.3 Results of the experiment

The metrics described in 3.4 are commonly used to evaluate the performance of each prepared models.

Short-term linear prediction

From our experiment we get R-squared (R^2) value of 0.8872 which means that 88.7% of the variation in the dependent variable can be explained by the independent variables in the model.

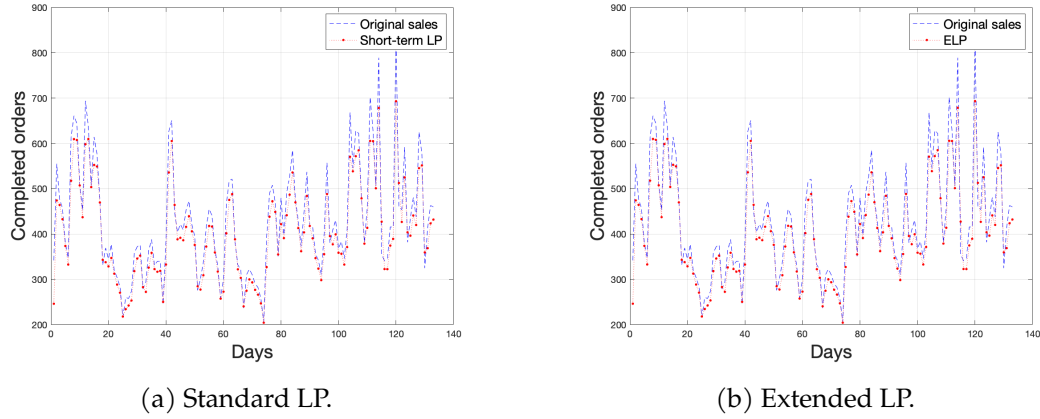


Figure 5.1: Results of short-term linear prediction.

In our case, an RMSE of 40.6215 means that on average, the predicted values from the model are about 40.6 units away from the actual values. And the MSE of 1650.1 means that on average, the squared difference between the predicted and actual values is 1650.1.

On figure 5.1a we can see predicted orders and on figure 5.3a is the prediction error for short-term linear prediction.

Extended short-term linear prediction

After conducting our experiment, we calculated an R-squared (R^2) value of 0.8881. This indicates that approximately 88.8% of the variation in the dependent variable can be explained by the independent variables included in our model.

We also determined that the root mean squared error (RMSE) is 40.4662. This means that, on average, there is a difference of approximately 40.4 units between the predicted values from our model and the actual values.

Additionally, we found that the mean squared error (MSE) is 1637.5. This suggests that, on average, the squared difference between the predicted and actual values is approximately 1650.1.

Long-term linear prediction

an R^2 value of 0.9160 means that 91.6% of the variation in the dependent variable can be explained by the independent variables in the model.

RMSE (Root Mean Square Error) and MSE (Mean Squared Error) are measures of the error or difference between the actual values of the dependent variable and the predicted values from the model. RMSE is the square root of the average squared difference between the predicted and actual values, while MSE is simply

the average squared difference between them.

In this case, an RMSE of 35.0163 means that on average, the predicted values from the model are about 35.02 units away from the actual values. And the MSE of 1223.3 means that on average, the squared difference between the predicted and actual values is 1223.3. On figure 5.2 we can see predicted orders and on figure 5.3b is the prediction error for short-term linear prediction.

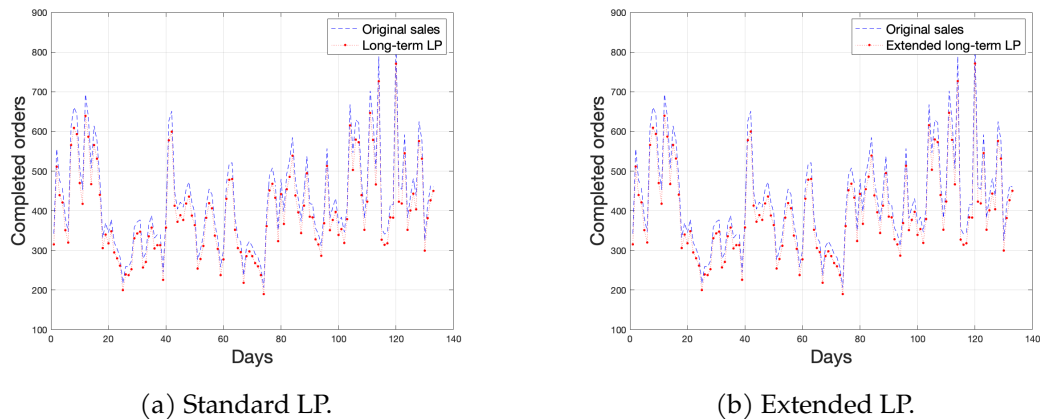


Figure 5.2: Results of long-term linear prediction.

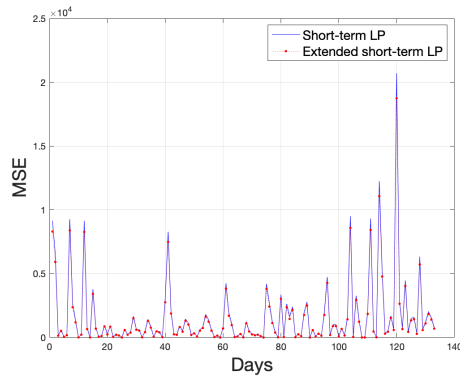
Extended long-term linear prediction

the R-squared value of 0.9171 indicates that the independent variables in the model can explain about 91.7% of the variation observed in the dependent variable. the RMSE and MSE are metrics used to evaluate the accuracy of the model's predictions compared to the actual values. The RMSE value of 34.7782 means that the average difference between the predicted and actual values is approximately 34.7 units. The MSE value of 1206.7 indicates that the average squared difference between the predicted and actual values is approximately 1206.7.

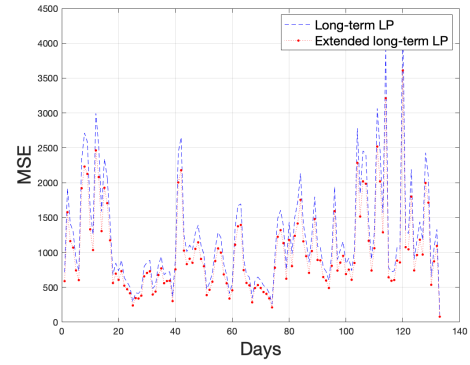
Overall, higher R-squared values and lower RMSE and MSE values in this model are desirable as they indicate a better fit of the model to the data and better predictive accuracy.

Comparison of the predictors

Let's see results from our experiment in table 5.1 as we expected long term prediction is worst than long term prediction and apply extended weights to linear prediction made all comparison parameters better.



(a) Short-term LP.



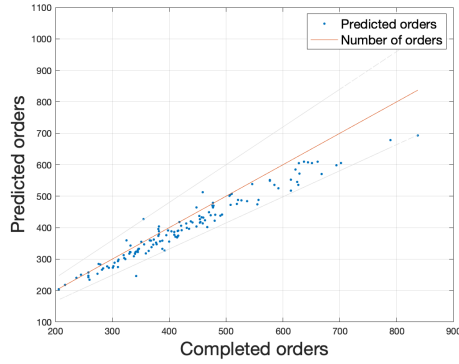
(b) Long-term LP.

Figure 5.3: Comparison of MSE for standard and extended LP.

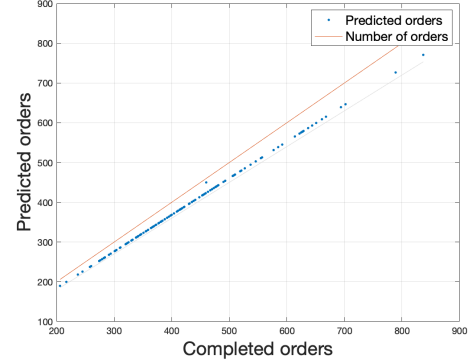
On the figures with MSE representation we can see the main better point of long-term prediction for economical datasets. From R^2 and RMSE the improvements of the models is not see so higher than on the figure 5.4.

Model	R^2	RMSE	MSE
Short-term linear prediction	0.8872	40.6215	1650.1
Extended short-term linear prediction	0.8881	40.4662	1637.5
Long-term linear prediction	0.9160	35.0163	1223.3
Extended Long-term linear prediction	0.9171	34.7782	1206.7

Table 5.1: Comparison of linear prediction models



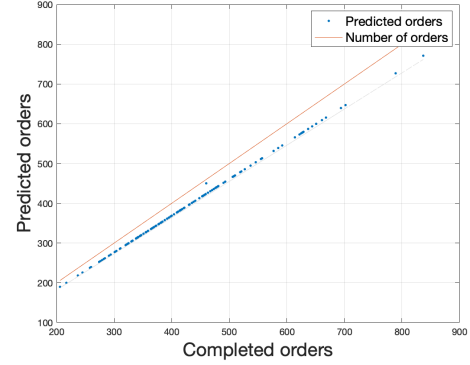
(a) Short-term linear prediction.



(b) Long-term linear prediction.



(c) Extended short-term LP.



(d) Extended long-term LP.

Figure 5.4: Results of prediction performance.

On the figure 5.4 we can see the main results and differences between the models. We can see the less variance in long-term predictors (figures 5.4b, 5.4d) than in short one (figures 5.4a, 5.4c).

6 Summary

The aim of this master's thesis was to improve the accuracy of linear prediction models for sales and financial forecasting using modern machine learning techniques. To achieve this goal, a mathematical model based on long-term linear prediction was created, and periodical weights were implemented to improve the accuracy of the linear prediction models.

Two main techniques were employed to apply machine learning to linear prediction. The first technique, called feature engineering, involved using machine learning to extract meaningful features from the input signal, which were then used as inputs for a linear prediction model. Several feature engineering techniques were used, including principal component analysis, wavelet transform, and Fourier transform.

The second technique involved model selection and training, which used machine learning to choose the best linear model for the prediction task and estimate its parameters from the data. Common machine learning algorithms such as linear regression, support vector regression, and artificial neural networks were used for model selection and training.

The ultimate goal of long-term linear prediction was to estimate future values of a signal or time series based on its past values using a linear model. The extended long-term linear prediction method proved to be the most effective, with comparison parameters such as R^2 getting 0.9171, RMSE of 34.7782, and MSE ending with a value of 1206.7.

The dataset used for the analysis was collected in 2022 as real orders. The first part of the dataset was used to train the models, and the second part was used to validate the results.

The results of this study were achieved in collaboration with the E-commerce Association o.z, the creator of the storePredictor platform. The organization provided valuable business insights into the industry, which helped set mechanisms for periodical weights to extend linear prediction models with minimal deflection. As a result, the next steps should focus on clearly defining and validating mechanisms to set independent industry weights only on ordinary accessible data without consultation with business owners.

In conclusion, this study successfully improved the accuracy of linear prediction models for sales and financial forecasting by utilizing modern machine learning techniques. The results of this study have practical implications for businesses, as accurate forecasting can help them make better decisions, increase profits, and achieve their goals.

7 Resume

7.1 Analýza

Lineárna predikcia je štatistická metóda používaná na predpovedanie budúcich hodnôt na základe historických dát pre identifikáciu parametrov používa Durbin-Levinsonov algoritmus čož je metóda riešenia sústav rovníc pre autoregresívne (AR) modely, ktoré sú modely, kde aktuálny výstup závisí od predchádzajúcich výstupov. Algoritmus rieši problém lineárnej predikcie nájdením koeficientov AR modelu, ktoré minimalizujú chybu predikcie. Výsledné AR koeficienty môžu byť použité na predpovedanie budúcich hodnôt na základe minulých pozorovaní. Táto metóda by sa mala používať s použitím vzoru lineárneho vzťahu medzi nezávislými a závislými premennými. Tu je základný prehľad krokov pri použití lineárnej predikcie na predpovedanie dát o predaji:

1. Zbierajte dáta o predaji
Získajte historické dáta o predaji produktu alebo služby, ktorú chcete predpovedať.
2. Vykreslite dáta
Vykreslite dáta o predaji v čase, aby ste vizuálne preskúmali trend a identifikovali akékoľvek vzory.
3. Vyberte model
Vyberte vhodný lineárny model na zobrazenie vzťahu medzi nezávislými a závislými premennými v dátach. Napríklad by ste si mohli vybrať jednoduchý lineárny regresný model.
4. Natrénujte model
Natrénujte vybraný model na historických dátach o predaji napríklad pomocou metódy najmenšie štvorcov.

Autoregresívne (AR) modely sú modely časových radov, ktoré popisujú vzťah medzi súčasnou hodnotou premennej a jej minulými hodnotami.

V autoregresívnom modeli je každé pozorovanie modelované ako lineárna kombinácia minulých pozorovaní, s váhami nazývanými AR koeficienty. AR modely sú široko používané v rôznych oblastiach, ako je ekonomika, inžinierstvo a financie na modelovanie a predpovedanie časových radov. Rád AR modelu, označované ako "p", sa vzťahuje na počet minulých hodnôt použitých na predpovedanie súčasnej hodnoty. Napríklad AR(1) model používa len predchádzajúce pozorovanie na predpovedanie súčasnej hodnoty, zatiaľ čo AR(2) model používa predchádzajúce dve pozorovania.

Predpovedanie budúcich predajov

Použijeme trénovaný model na predpovedanie budúcich predajov. Môžete chcieť generovať predpovede na niekoľko mesiacov alebo rokov dopredu. Postupujte nasledovne:

1. Zhodnoťte model

Zhodnoťte presnosť predpovedí porovnaním s faktickými predajovými dátami. Na kvantifikáciu výkonu modelu použite metriky ako priemerná absolútna chyba alebo koreňová stredná štvorcová chyba.

2. Vylepšite model

Ak je to potrebné, pridajte ďalšie nezávislé premenné alebo transformujte existujúce premenné, aby ste vylepšili model.

3. Opakujte kroky trénovania a hodnotenia, kým nebudete mať model, ktorý poskytuje presné prognózy.

4. Pre výpočet posunu v dlhodobom predpovedaní môžete použiť autokorelačnú metódu. V tejto práci sa vyvinie neurónová sieť na identifikáciu posunu a podobný mechanizmus pre optimálne určenie poradia.

Použite trénovaný model na predpovedanie budúcich predajových dát. Môžete chcieť generovať predpovede na niekoľko mesiacov alebo rokov dopredu.

Modely používané pre predikcie ekonomických dát

Existuje niekoľko matematických modelov používaných pre predpovedanie predaja, vrátane:

1. Modely časových radov

Tieto modely sa používajú na analýzu a predpovedanie ekonomických dát v čase, ako sú sezónne výkyvy, trendy a fluktuácie. Príklady zahŕňajú ARIMA (AutoRegressive Integrated Moving Average), SARIMA (Seasonal ARIMA) a exponenciálne vyhladzovanie.

2. Regresné modely

Tieto modely používajú historické údaje na určenie vzťahu medzi predajom a jednou alebo viacerými nezávislými premennými, ako sú cena, propagácia a reklama. Príklady zahŕňajú lineárnu regresiu, logistickú regresiu a viacnásobnú regresiu.

3. Modely stromových rozhodnutí

Tieto modely používajú štruktúru stromu na rozhodovanie založené na vzťahu medzi predajom a viacerými nezávislými premennými. Príklady zahŕňajú CART (Klasifikácia a regresia rozhodovacích stromov) a náhodných stromov.

4. Modely strojového učenia

Tieto modely používajú algoritmy ako neurónové siete a stroje s podpornými vektormi na predikovanie na základe vzorov v údajoch.

Neurónove siete

Neurónová sieť je druh algoritmu strojového učenia inšpirovaný štruktúrou a funkciou biologických neurónov v ľudskom mozgu. Skladá sa z prepojených uzlov, nazývaných neuróny, ktoré sú usporiadané do vrstiev. Vstupná vrstva prijíma surové dáta, ako sú obrázky alebo text, a prenáša ich do skrytých vrstiev, ktoré vykonávajú výpočty a váhy sa aplikujú na vstupné dáta pre vytvorenie predikcie. Nakoniec výstupná vrstva produkuje konečnú predikciu alebo klasifikáciu.

Ako môžete vidieť na obrázku 1.2, každý vstup X_n by mal byť správne ohodnotený určitou váhou W_n predtým, než všetky signály vstúpia do sumovacej fázy. Potom sa vážené súčty prenášajú do aktivačnej jednotky produkujúcej výstupný signál neurónu.

Neurónové siete sa trénujú na veľkých dátových súboroch pomocou procesu nazývaného spätné šírenie chyby, ktorý upravuje váhy a sklon neurónov, aby minimalizoval rozdiel medzi predpovedaným výstupom a skutočným výstupom. Akonáhle

je neurónová sieť natrénovaná, môže sa použiť na predpovedanie nových dát.

Neurón je základnou stavebnou jednotkou neurónovej siete, známy aj ako umelý neurón alebo perceptrón. Modeluje sa podľa biologického neurónu v ľudskom mozgu, ktorý prijíma vstupné signály z iných neurónov, spracováva ich a posiela výstupné signály do ďalších neurónov.

V neurónovej sieti neurón prijíma vstup od iných neurónov alebo priamo od vstupných dát, aplikuje na vstup matematickú funkciu a produkuje výstup, ktorý sa posiela do ďalších neurónov v sieti. Vstupom do neurónu je zvyčajne vektor čísel a každý vstup sa násobí príslušnou váhou.

Potom neuron sčíta vážené vstupy, pridáva sklon a aplikuje aktivačnú funkciu na výsledok. Úlohou aktivačnej funkcie je zaviesť nelinearitu do neurónu, čo umožňuje neuronovej sieti naučiť sa zložité vzorce a vzťahy v dátach. Existuje niekoľko rôznych typov aktivačných funkcií, ktoré sa môžu použiť, ako napríklad sigmoidná funkcia, ReLU (Rectified Linear Unit) funkcia a tanh (hyperbolická tangens) funkcia.

Výstup neurónu sa zvyčajne posúva do ďalších neurónov v nasledujúcej vrstve neurónovej siete. Váhy a sklon neurónov sa počas tréningu prispôbujú technikou spätného šírenia chyby, ktorá zahŕňa výpočet gradientu chyby vzhľadom na váhy a aktualizovanie ich pomocou optimalizačného algoritmu, ako je stochastický gradientový zostup.

Celkovo neuróny v neurónovej sieti spolupracujú na učení vzorcov a vzťahov vstupných dát a produkujú výstup, ktorý sa môže použiť pre rôzne úlohy, ako je klasifikácia, regresia a predikcia.

Neurónové siete sa úspešne uplatňujú v širokej škále oblastí, vrátane rozpoznávania obrazov a reči, spracovania prirodzeného jazyka a autonómnych vozidiel, medzi inými.

7.2 Syntéza

Vychádzajme z kapitoly 7.1 a na ich základe vytvoríme nové matematické modely a prístupy, aby sme mohli vykonať rýchle a presné predpovede predaja, ktoré sa

skladajú z long-term lineárnej predikcie s individuálnymi váhami vypočítanými pre každé obdobie, založené na Levinson-Durbinovej schéme, nový mode nazývane Extended Linear Prediction (ELP). Očakávame lepšie výsledky než pri použití predikcie založenej na krátkodobej alebo štandardnej long-term lineárnej predikcii (viď časť 1.3). Nakoniec, náš prístup bude predikovať budúce hodnoty predaja na základe predchádzajúcich dát s lepšou odchýlkou, než to dokáže lineárna predikcia.

Pre vytvorenie matematického modelu na predikciu predajných dát s periodickými trendmi môžeme použiť model sezónnej ARIMA (SARIMA). Tento model zohľadňuje sezónne variácie v dátach a používa autoregresívne a kľzavé priemerové členy na zachytenie vzorov a trendov v dátach.

Pre náš účel sme vytvorili rozšírenú dlhodobú predikciu, ktorá sa vysporiada so sezónnymi a opakujúcimi sa vzormi v ekonomických dátach. Tento korekčný mechanizmus sa používa na zohľadnenie historických vrcholov v grafoch naprieč datasetom. Táto jednoduchá korekcia zvyšuje presnosť modelu a získava lepšiu odpoveď vďaka psychologickým, sociologickým a marketingovým aspektom v datasete.

Na nastavenie periodických váh je potrebné vytvoriť vektor korekčných parametrov z pôvodného datasetu pomocou štatistických parametrov mediánu a štandardnej odchýlky z datasetu. Ako základnú rovnicu použijeme rovnicu pre dlhodobú predikciu 4.2 s novými váhami a získame tak lepšie výsledky.

7.3 Experiment

Pripravme experiment na overenie dlhodobého lineárneho predikčného modelu objednávok internetového obchodu. Zostavíme experiment podľa týchto krokov:

1. Definujeme problém a ciele

Jasne definujte problém a predmet ktoré má vyriešiť dlhodobý lineárny predikčný model. V našom prípade je to predpovedať objednávky v internetovom obchode na dlhé časové obdobie pomocou lineárneho modelu.

2. Zhromažďovanie údajov

Zhromažďujeme historické údaje o objednávkach online obchodov, ako je napríklad objednávka dátumy, sumy objednávok a ďalšie relevantné pre-

menné. Tieto údaje budú použité trénovať a overovať model long-term lineárnej predikcie. Naše údaje boli zbierať z online systému storePredictor, ktorý je dostupný na storepredictor.com a údaje sú anonymizované a pseudonimizované, aby sa mohli použiť na ďalšie potrebné výpočty.

3. Pripravíme údaje

Vyčistíme a predspracujeme údaje, aby sme sa uistili, že sú konzistentné a vhodné na použitie v dlhodobom lineárnom predikčnom modeli. Toto môže zahŕňať postupy ako je odstraňovanie duplikátov, spracovanie chýbajúcich hodnôt a transformovanie premenné podľa potreby. Predspracovanie našich údajov je popísané v 5.1.

4. Vytvoríme model long-term lineárnej predikcie

Pomocou historických údajov, vyvineme dlhodobý lineárny predikčný model, ktorý dokáže predpovedať internetový obchod objednávky počas určitého časového obdobia, ktoré je podrobne popísané v bode 4.1 a prakticky aplikovaný v 5

5. Overenie modelu

Keď je model vytvorený, je potrebné ho overiť, aby sme zabezpečili, že je presný a účinný.

6. Vyhodnotíme výsledky

Analyzujeme výsledky experimentu, aby sme určili presnosť modelu long-term lineárnej predikcie. To bude zahŕňať výpočet metrík, ako je stredná kvadratická chyba (MSE), r-square (R^2) a relatívna stredná kvadratická chyba (RMSE) 5.3

7. Počas riešenia predchádzajúcich krokov bol model predefinovaný a revalidovaný, keď výsledky neboli uspokojivé, upravili sme model a zopakovali validáciu až kým nevzniknul presný a efektívny dlhodobý lineárny predikčný model.

Stručne povedané, museli sme pripraviť experiment na overenie long-term lineárnej predpovede model pre objednávky v internetovom obchode, kde sme museli definovať problém a ciele, zhromažďovať a pripravovať údaje, vyvíjať model, overovať ho, hodnotiť výsledky, a ak je to potrebné, model spresniť a znovu overovať.

7.4 Záver

Cieľom tejto diplomovej práce bolo zlepšiť presnosť lineárnej predikcie a vytvoriť vlastné modely pre predaj a finančné prognózy využívajúce moderné technológie strojového učenia. Na dosiahnutie tohto cieľa sme vytvorili matematický model založený na dlhodobom lineárnom predikčnom modeli a boli implementované periodické váhy na zlepšenie presnosti lineárnych predikčných modelov.

Na aplikáciu strojového učenia na lineárne predikcie sme použili dve metódy. Prvá technika, nazývaná inžinierstvo funkcií, zahŕňala použitie strojového učenia na extrahovanie zmysluplných vlastností zo vstupného signálu, ktoré boli vtedy používané ako vstupy pre lineárny predikčný model. Boli použité niques, vrátane analýzy hlavných komponentov, vlnkovej transformácie, a Fourierova transformácia.

Druhá technika zahŕňala výber modelu a tréning, ktorý využíval schopnosť naučiť sa vybrať najlepší lineárny model pre predikčnú úlohu a odhadnúť jeho parametre z údajov. Bežné algoritmy strojového učenia, ako napríklad lineárna regresia, podporná vektorová regresia a umelé neurónové siete pre výber modelu a tréning.

Konečným cieľom dlhodobej lineárnej predikcie bolo odhadnúť budúce hodnoty objednávok alebo časového radu ekonomických dát na základe minulých hodnôt pomocou lineárneho modelu. Rozšírená dlhodobá predikcia z nášho experimentu sa ukázala ako najefektívnejšia a porovnávacie parametre R^2 0,9171, RMSE 34,7782 a MSE 1206,7 bola zo všetkých testovacích modelov najlepší.

Súbor údajov použitý na analýzu sme zozbierali v roku 2022 ako skutočné objednávky. Prvu časť súboru údajov bola použitá na tréning modelov a druhá časť bola použitá na overenie výsledkov.

Výsledky tejto štúdie boli dosiahnuté vo spolupráci s E-commerce Association o.z, tvorca platformy storePredictor. Organizácia poskytla cenné obchodné poznatky v tomto odvetví, ktoré pomohli nastaviť mechanizmy pre periodické váhy na rozšírenie modelov lineárnej predikcie s minimálnou odchádzkou. V dôsledku toho by sa ďalšie kroky mali zamerať na jasné definovanie a overenie mechanizmov na nastavenie nezávislých odvetvových váh len na bežne dostupné údaje bez

konzultácie s majiteľmi firiem.

Na záver, táto štúdia úspešne zlepšila presnosť lineárnej predikcie pre predaj a finančné prognózy s využitím moderného strojového učenia. Výsledky tejto štúdie majú praktické dôsledky pre podniky, keďže presné predpovede im môžu pomôcť robiť lepšie rozhodnutia, zvýšiť zisky a dosiahnuť svoje ciele.

Bibliography

1. VAIDYANATHAN, P. p. *The Theory of Linear Prediction*. 2007.
2. MATHWORKS. *What Is a Live Script or Function?* 2021.
https://www.mathworks.com/help/matlab/matlab_prog/what-is-a-live-script-or-function.html.
3. CRYER, Jonathan D. *Time series analysis*. Duxbury Press Boston, 1986.
4. FAHRMEIR, Ludwig; KNEIB, Thomas; LANG, Stefan; MARX, Brian D. Regression models. In: *Regression: Models, methods and applications*. Springer, 2022, pp. 23–84.
5. KOTSIANTIS, Sotiris B. Decision trees: a recent overview. *Artificial Intelligence Review*. 2013, vol. 39, pp. 261–283.
6. CHEN, Po-Hsuan Cameron; LIU, Yun; PENG, Lily. How to develop machine learning models for healthcare. *Nature materials*. 2019, vol. 18, no. 5, pp. 410–414.
7. MOURGIAS-ALEXANDRIS, G.; TSAKYRIDIS, A.; PASSALIS, N.; TEFAS, A.; VYRSOKINOS, K.; PLEROS, N. An all-optical neuron with sigmoid activation function. *Opt. Express*. 2019, vol. 27, no. 7, pp. 9620–9630. Available from doi: 10.1364/OE.27.009620.
8. SVOZIL, Karl. Quantum logic. 1997.
9. GU, Jiuxiang; WANG, Zhenhua; KUEN, Jason; MA, Lianyang; SHAHROUDY, Amir; SHUAI, Bing; LIU, Ting; WANG, Xingxing; WANG, Gang; CAI, Jianfei; CHEN, Tsuhan. Recent advances in convolutional neural networks. *Pattern Recognition*. 2018, vol. 77, pp. 354–377. ISSN 0031-3203. Available from doi: <https://doi.org/10.1016/j.patcog.2017.10.013>.
10. MEDSKER, Larry R; JAIN, LC. Recurrent neural networks. *Design and Applications*. 2001, vol. 5, pp. 64–67.
11. CHENG, Jianpeng; DONG, Li; LAPATA, Mirella. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*. 2016.

12. LUO, Tie; NAGARAJAN, Sai G. Distributed anomaly detection using autoencoder neural networks in WSN for IoT. In: *2018 IEEE International Conference on Communications (ICC)*. 2018, pp. 1–6.
13. CRESWELL, Antonia; WHITE, Tom; DUMOULIN, Vincent; ARULKUMARAN, Kai; SENGUPTA, Biswa; BHARATH, Anil A. Generative adversarial networks: An overview. *IEEE signal processing magazine*. 2018, vol. 35, no. 1, pp. 53–65.
14. FÉRAUD, Raphael; CLÉROT, Fabrice. A methodology to explain neural network classification. *Neural networks*. 2002, vol. 15, no. 2, pp. 237–246.
15. GÉRON, Aurélien. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. "O'Reilly Media, Inc.", 2022.
16. PARKS, P. Further comments on "An symmetric matrix formulation of the Hurwitz-Routh stability criterion. *IEEE Transactions on Automatic Control*. 1963, vol. 8, no. 3, pp. 270–271.
17. RIAHY, G.H.; ABEDI, M. Short term wind speed forecasting for wind turbine applications using linear prediction method. *Renewable Energy*. 2008, vol. 33, no. 1, pp. 35–41.
18. VASEGHI, Saeed V. *Advanced Digital Signal Processing and Noise Reduction*. John Wiley & Sons, 2008.
19. NAVE, G.; COHEN, A. ECG compression using long-term prediction. *IEEE Transactions on Biomedical Engineering*. 1993, vol. 40, no. 9, pp. 877–885. Available from DOI: 10.1109/10.245608.
20. D. N. BAKER R. L. McPherron, T. E. Cayton et al. Linear prediction filter analysis of relativistic electron properties at 6.6 RE. *Space Physics*. 1990, vol. 95, no. A9, pp. 15133–15140.
21. KINOSHITA, Keisuke; DELCROIX, Marc; NAKATANI, Tomohiro; MIYOSHI, Masato. Suppression of Late Reverberation Effect on Speech Signal Using Long-Term Multiple-step Linear Prediction. *IEEE Transactions on Audio, Speech, and Language Processing*. 2009, vol. 17, no. 4, pp. 534–545. Available from DOI: 10.1109/TASL.2008.2009015.
22. PHAM, Dinh Tuan; LE BRETON, Alain. Levinson-Durbin-type algorithms for continuous-time autoregressive models and applications. *Mathematics of Control, Signals and Systems*. 1991.
23. DURBIN, J. *The Fitting of Time-Series Models*. 1960.

24. MILLS, Terence C. *Time series econometrics: a concise introduction*. Springer, 2015.
25. JANDERA, Ales; SKOVRANEK, Tomas. Customer Behaviour Hidden Markov Model. *Mathematics*. 2022, vol. 10, no. 8. ISSN 2227-7390. Available from DOI: 10.3390/math10081230.
26. CHARALAMPOPOULOS, Panagiotis; CHEN, Huiping; CHRISTEN, Peter; LOUKIDES, Grigorios; PISANTI, Nadia; PISSIS, Solon; RADOSZEWSKI, Jakub. Pattern masking for dictionary matching. In: *International Symposium on Algorithms and Computation*. 2021, 65:1–65:19. No. 212. Available from DOI: 10.4230/LIPIcs.ISAAC.2021.65.
27. JANDERA, Aleš. *Customers behavior modeling in e-commerce*. Košice: Technical University of Košice, Faculty of Mining, Ecology, Process Control and Geotechnologies, 2021. 37s. Supervisor: Tomáš Škovránek.