Università degli Studi di Cagliari
Facoltà di Scienze

Corso di Laurea Triennale in
Informatica Applicata e Data Analytics

Latent Space Exploration for Financial Statement Forecasting

Relatore:
Dott. Alessandro Giuliani

Candidato:
Gabriele Bandino

Co-Relatore:
Prof. Salvatore M. Carta

ACADEMIC YEAR 2023/2024

Abstract

Accurate financial forecasting is critical for effective strategic planning and risk management in today's dynamic economic environment. This thesis presents a software framework designed to predict financial outcomes of companies through advanced machine learning techniques. The framework employs Temporal AutoEncoders with Long Short-Term Memory (LSTM) networks to process and embed financial data into a lower-dimensional space, capturing essential patterns and trends.

The process begins with data preprocessing to ensure consistency and reduce noise. The financial data is then compressed using Temporal AutoEncoders, significantly reducing its dimensionality. Various regression models are applied to the encoded data to forecast future financial states, offering enhanced prediction accuracy and reduced computational costs.

Validation against real-world financial datasets demonstrates the framework's robustness and reliability. The findings illustrate the effectiveness of combining dimensionality reduction with machine learning algorithms to improve financial predictions. This work lays the groundwork for further advancements in financial data analysis and predictive modeling.

Contents

Chapter 1

Introduction

1.1 Context and Motivation

In an increasingly complex and dynamic economic context, the ability to predict the **financial performance** of companies has become a critical component for **strategic planning** and **risk management**. Financial data analysis, combined with advanced **machine learning techniques**, offers powerful tools to anticipate future trends and make informed decisions.

The rapid development of **artificial intelligence** technologies and the increase in **data availability** have enabled the creation of increasingly accurate and efficient predictive models. In this scenario, the application of **dimensionality reduction techniques**, such as the **AutoEncoder**, and the use of advanced **regression models** represent a promising frontier for improving the prediction of financial variables.

1.2 State of the Art

Predicting financial outcomes using machine learning has evolved significantly beyond traditional statistical methods. Advanced techniques like **Random Forest**, **Gradient Boosting Machines**, and **Support Vector Machines** have improved predictive accuracy by capturing complex patterns in financial data. Neural network architectures, particularly **Long Short-Term Memory (LSTM)** networks and **Convolutional Neural Networks (CNNs)**, have also been effective in this domain. [8]

Dimensionality reduction is crucial for enhancing model performance by reducing noise and complexity. While methods like **Principal Component Analysis (PCA)** and **t-Distributed Stochastic Neighbor Embedding (t-SNE)** are commonly used, **AutoEncoders** offer a powerful alternative. AutoEncoders learn efficient data representations, making them suitable for handling complex financial datasets. [15]

Many studies indicate that machine learning models outperform other techniques, particularly statistical approaches (e.g., ARIMA models [12]), especially for nonstationary data. The flexibility and capacity of machine learning algorithms to adapt to changing patterns in data make them superior for dynamic and complex financial forecasting scenarios. [5]

1.3 Objective of the Thesis

The primary objective of this thesis is to design and implement a software framework that applies machine learning techniques to predict financial outcomes based on historical data by exploring the latent space generated by the AutoEncoder. This involves:

- Utilizing AutoEncoder models to effectively compress and decompress financial data, allowing for a reduction in dimensionality and noise.

- Employing various regression models to forecast future financial states from the compressed data representations.

1.4 Thesis Activities

The development and completion of this thesis involved a series of structured activities:

1. **Problem Study:** Initial analysis to understand the specific challenges and requirements in enhancing financial data analysis using machine learning.
2. **Technology Review:** Exploration and evaluation of various machine learning algorithms and tools suitable for data compression and prediction.
3. **System Design:** Architectural planning of the framework, including data flow, model integration, and user interface considerations.
4. **Implementation:** Coding the system using Python and relevant libraries to create a functional prototype.
5. **Testing:** Systematic validation of each component to ensure the reliability and accuracy of the data predictions.
6. **Documentation:** Preparation of comprehensive documentation to support future development and deployment of the framework.
7. **Model Performance Study:** In-depth analysis of the models' performance, including evaluation of various metrics.
8. **Hyperparameter Tuning:** Optimization of model parameters through techniques like grid search and random search to improve performance.

9. **Latent Space Exploration:** Examination of the latent space generated by AutoEncoders to understand the data representation and influence on the reconstruction quality.

10. **Results Analysis:** Critical evaluation of the model performance against benchmark and initial project goals.

Each of these activities contributed significantly to the thesis, ensuring a robust approach to solving the identified problem with advanced technological tools.

1.5 Thesis Structure

This thesis is organized as follows:

In **Chapter 2**, the functionality of the system is presented, describing in general terms what it does without delving into architectural or implementation details.

In **Chapter 3**, the system architecture is described, illustrating the main blocks and their interactions.

In **Chapter 4**, the implementation of the system is detailed, describing the technologies used for each block.

In **Chapter 5**, the validation and testing of the system are discussed, highlighting the results obtained and the accuracy of the predictions.

Finally, in **Chapter 6**, the conclusions are drawn, summarizing the achieved results and proposing possible future developments.

Chapter 2

Model Functionality

The developed system aims to predict future **financial trends** of companies using advanced **data analysis** and **machine learning** techniques. This approach not only enhances prediction accuracy but also offers several advantages that streamline the overall process. The methodology involves several key steps:

2.1 Data Embedding

The system begins by processing the financial data and embedding it into a lower-dimensional space. This **embedding process** is essential as it captures the core features and patterns within the data, significantly reducing its complexity. By transforming the data into a more compact form, the system ensures that it retains the most relevant information, making it more manageable for further analysis and prediction. One significant advantage of this approach is the reduction in **computational cost**, as working with lower-dimensional data requires less processing power and storage.[3] Additionally, this step improves the model's ability to generalize from historical data, leading to more robust predictions.[1]

2.2 Trend Prediction

With the data now embedded in a lower-dimensional space, the system proceeds to predict future financial trends. Utilizing this compact representation, the model analyzes historical patterns and relationships within the data to forecast future performance. This predictive capability is built on **regression algorithms** that can identify subtle trends and make accurate forecasts. Another benefit of this methodology is its ability to handle large datasets efficiently, making it also suitable for **real-time analysis** and decision-making in dynamic financial environments.

2.3 Validation and Testing

The predicted financial data is validated and tested against real data to ensure its accuracy and reliability. This **validation process** involves comparing the system's forecasts with the actual financial data to assess performance. By testing the predictions, the system verifies that the model can effectively generalize from historical data to unseen future scenarios. This step is crucial in establishing the credibility of the predictions and ensuring that they can be trusted for making informed financial decisions.

Overall, the use of this advanced methodology not only enhances the accuracy and reliability of financial predictions but also optimizes computational resources, making it a powerful tool for **strategic financial planning** and decision-making.

Chapter 3

System Architecture

This chapter describes the architecture of the developed system for financial data forecasting, divided into the main functional blocks. Each block is defined and detailed, highlighting the connections between them.

3.1 Pre-Processing

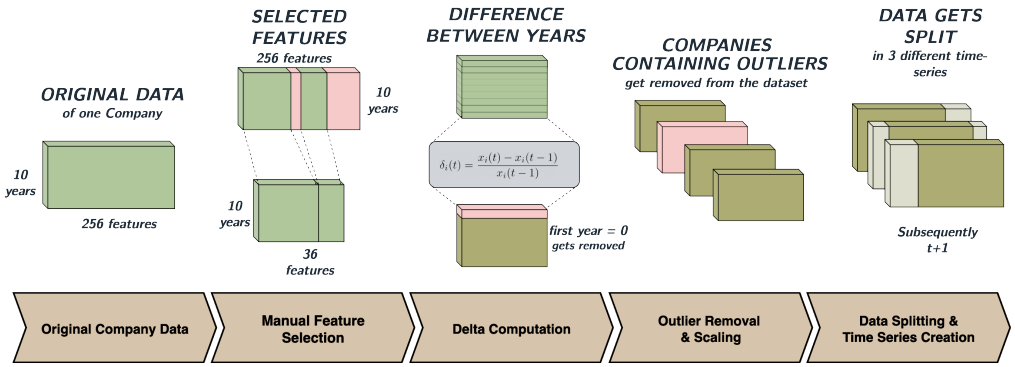

Figure 3.1: Data Pre-Processing Pipeline

Pre-processing is a critical block that prepares raw data for training and forecasting. This block includes several phases:

- **Delta Computation**: The whole data goes through a Delta Computation step, in which the difference between the current and previous year gets computed. This happens in order to capture year-over-year changes and trends, enhancing the model's ability to detect significant financial shifts.

- **Dataset Splitting**: The dataset is divided into **training**, **testing**, and **validation** sets to ensure that the models are properly trained and evaluated on separate data.

- **Time Series Creation**: The financial data is organized into temporal sequences to capture the time-based dependencies and trends. This step is crucial for leveraging the **Temporal AutoEncoder**'s ability to model financial data over time.

- **Data Scaling**: Data normalization techniques such as **MinMaxScaler** are applied to scale the data, ensuring that all features contribute equally to the model training.

3.2 Temporal AutoEncoder

The **AutoEncoder** is the core component of the system, responsible for **dimensionality reduction** and **feature extraction** from the financial data. This block is divided into two main parts: the encoder and the decoder.

- **Encoder**: The encoder transforms the high-dimensional financial data into a lower-dimensional space, capturing essential temporal features and patterns. This process involves **LSTM layers** that are specifically designed to handle sequences of data, learning to compress the input data while retaining important temporal dependencies.

- **Decoder**: The decoder reconstructs the original temporal sequences from the encoded representation, ensuring that key temporal information is preserved.

The Temporal AutoEncoder model is trained using the pre-processed and sequenced training data. The training process involves optimizing the model to minimize the reconstruction error across the sequences, ensuring that the encoded representation is both meaningful and temporally coherent.

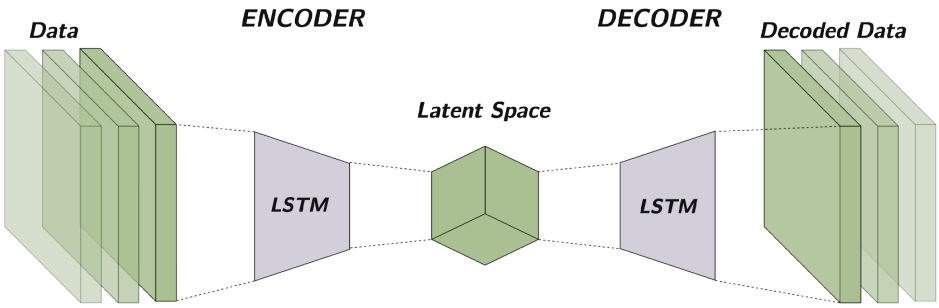

Figure 3.2: Illustration of the LSTM AutoEncoder

3.3 The Regressors

The **regressor block** uses the encoded data from the Temporal AutoEncoder to predict future financial values. Various **regression models** are employed to enhance the accuracy of the forecasts. The primary regressors used in the system include:

- **XGBoost**: A powerful gradient boosting model that combines multiple decision trees to improve prediction accuracy. It is particularly effective in handling large datasets and complex relationships.

- **Random Forest**: An ensemble learning method that constructs multiple decision trees and merges them to obtain more accurate and stable predictions. It is robust against overfitting and can handle high-dimensional data.
- **MLP (Multi-Layer Perceptron)**: A type of feedforward artificial neural network that captures non-linear relationships in the data through multiple hidden layers and non-linear activation functions.
- **AdaBoost and Decision Tree**: Additional regression models used for benchmarking and comparison. AdaBoost combines multiple weak learners to create a strong predictive model, while decision trees provide simple yet effective predictions based on hierarchical data partitioning.

Each model is trained on the encoded training data and evaluated on the testing and validation sets to determine its predictive performance. The best-performing models are selected for the final forecasting stage.

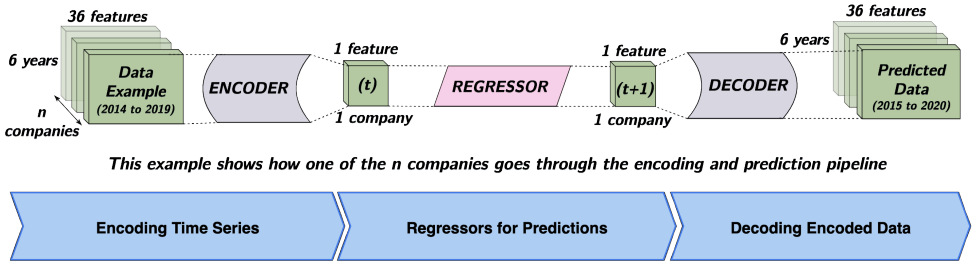

Figure 3.3: Example of a Company's Data Prediction in Mono-Dimensional Latent Space

3.4 Connections Between Blocks

The connections between blocks are essential for the overall functioning of the system. The data flow between the blocks can be summarized as

follows:

1. **Dataset → Pre-Processing**: The raw financial data is collected, cleaned, and standardized in the pre-processing block, and then organized into time series sequences.

2. **Pre-Processing → AutoEncoder**: The pre-processed data is used to train the Temporal AutoEncoder, which encodes the data into a lower-dimensional space.

3. **AutoEncoder → Regressors**: The encoded data from the Temporal AutoEncoder is fed into the regression models to predict future financial values.

4. **Regressors → Output**: The predictions generated by the regression models are compared with actual data and evaluated for accuracy.

This block-based architecture efficiently handles the entire process of financial data forecasting, from data acquisition to prediction generation, ensuring accurate and reliable forecasts.

Chapter 4

Implementation

This section provides implementation details for each block of the architecture defined in the previous chapter.

4.1 Dataset

The dataset forms the foundational layer of the system, consisting of historical financial data collected from the AIDA (Analisi informatizzata delle aziende italiane) database, provided by Bureau van Dijk. AIDA contains comprehensive financial data for all active Italian companies (excluding banks, insurance companies, and public entities), including balance sheets and income statements.

Mapping of balance sheet and income statement features:

- **Balance Sheet (pat):**
  - Total Intangible Assets: FP1
  - Total Tangible Assets: FP2

  - Total Financial Assets: FP3
  - Total Inventory: FP4
  - Total Receivables: FP5
  - Total Cash and Cash Equivalents: AP11
  - Total Financial Activities: AP10
  - Accrued Income and Prepaid Expenses: D1072
  - Net Equity: PP1
  - Total Provisions for Risks: PP2
  - Long-term Debts: PP4
  - Employee Severance Indemnity: D1089
  - Short-term Debts: FP9
  - Accrued Liabilities and Deferred Income: D1119

- **Income Statement (ec):**
  - Total Value of Production: E8
  - Revenue from Sales and Services: FE1
  - Change in Inventory of Finished Goods: D1125
  - Change in Work in Progress: D1126
  - Change in Raw Materials: D1145
  - Capitalization of Internal Work: D1127
  - Other Revenues: D1128
  - Raw Materials and Consumables: FE4
  - Services: D1132
  - Miscellaneous Operating Expenses: D1148
  - Lease and Rental Costs: D1133
  - Total Depreciation and Amortization: E2

- Amortization of Intangible Assets: D1140
- Depreciation of Tangible Assets: D1141
- Total Personnel Costs: FE7
- Total Financial Income and Expenses: E11
- Total Financial Expenses: D1159
- Total Adjustments to Financial Assets: E12
- Provisions for Risks: D1146
- Other Provisions: D1147
- Total Income Taxes: E7

The dataset used to train the model consists of 10 years of financial statement data from 66,680 Italian companies. Each row in the dataset contains 38 features: the first two features are the year and the company's ID, and the remaining 36 are manually selected financial metrics and indicators used for prediction.

4.2 Data Pre-Processing

The **pre-processing of financial data** is a critical phase to ensure that the data is **clean**, **consistent**, and ready for subsequent analysis. This phase includes several key operations:

4.2.1 Delta δ Computation

Delta computation is a fundamental transformation in the pre-processing of financial data. For each financial variable, the **delta** δ is calculated as the **percentage change** compared to the previous year. This captures

trends over time rather than absolute values, facilitating more accurate forecasting. The year **2013** is excluded as the calculated δ would be zero, potentially introducing **noise** into the system.

$$\delta_i(t) = \frac{x_i(t) - x_i(t-1)}{x_i(t-1)}$$

where:

- $\delta_i(t)$ is the delta or percentage change for variable i at time t,
- $x_i(t)$ is the value of variable i at time t,
- $x_i(t-1)$ is the value of variable i at the previous time $t-1$.

Year	**FP1**	**FP2**	**FP3**	**FP4**	...	**E12**	**D1146**	**D1147**	**E7**
2017	0.576	-0.041	0.003	0.105	...	-0.326	-0.815	0.004	-1.299
2018	-0.058	-0.035	0.542	0.054	...	0.097	2.279	0.080	-9.094
2019	1.490	0.021	0.007	0.094	...	0.029	1.373	-0.130	-3.911
2020	-0.381	-0.043	0.127	-0.105	...	0.200	0.274	0.007	-1.976
2021	-0.722	-0.025	-0.098	0.098	...	-0.190	-0.099	0.074	-1.484
2022	-0.357	0.208	0.022	0.123	...	-0.147	-0.388	0.104	-0.390
2017	-1.000	-0.087	0.000	0.120	...	-1.000	-0.167	0.091	-0.122
2018	1.000	-0.181	0.000	0.011	...	0.000	-0.760	0.115	-2.277
2019	0.000	-0.130	0.000	-0.197	...	0.000	0.000	-0.041	-0.663
2020	0.000	-0.328	0.000	-0.131	...	0.000	-1.000	0.049	2.893
2021	-1.000	0.578	0.000	0.335	...	0.000	1.000	0.065	-1.890
2022	1.000	1.718	0.000	0.083	...	0.000	0.354	0.151	-0.082

Table 4.1: Example of δ Computed Dataset

4.2.2 Outliers

To improve data quality and model performance, **outliers** are removed by eliminating the **2nd percentile** of data at the lower and upper extremes of

the distribution of financial variables. This technique, often recommended by economists, helps manage **data noise** and prevent **anomalous values** from distorting analysis results. Removing outliers ensures that the model focuses on the main **patterns in the data**, enhancing the robustness and reliability of the forecasts. [11]

```
# Remove year 2013 (becuase = 0) and sort
data = data[data['bilancio_year'] != 2013]
data = data.sort_values(by=['id', 'bilancio_year'], ascending=[True, True
    ])

# Calculate 2nd percentile thresholds for each feature
percentile_1 = data.drop(columns=['id', 'bilancio_year', 'sector']).
    quantile(0.02)

# Identify outliers and remove them
outliers = (data.drop(columns=['id', 'bilancio_year', 'sector']) <
    percentile_1).any(axis=1)
outlier_ids = data[outliers]['id'].unique()
data_cleaned = data[~data['id'].isin(outlier_ids)]
```

Listing 4.1: Outlier Removal

4.2.3 Data Splitting

Data splitting is a crucial step in the pre-processing phase, aimed at dividing the dataset into distinct subsets for **training**, **testing**, and **validation**. This structured separation facilitates different stages of the model's training and evaluation process:

- **Training Set:** The training dataset, consisting of financial data from the years **2015 to 2020**, is utilized to train the **AutoEncoder**. This data helps the AutoEncoder learn to compress and reconstruct the financial data, capturing essential **temporal patterns** which are crucial for the encoding process.

- **Label & Testing Set:** The testing set, which includes data from the year **2021**, serves a main role in the training of the **regressors**. After training the AutoEncoder, the training set is encoded and then used alongside the testing set to train regressors. The testing set acts as the labels or targets for this stage, providing a basis for the regressors to learn how to predict future financial values based on the compressed data produced by the AutoEncoder.

- **Validation Set:** The validation set comprises data from the year **2022**. This dataset is used to validate the entire model pipeline: starting from encoding the Testing Set data using the trained AutoEncoder, through prediction by the regressors, and finally decoding to reconstruct the original financial values. This step is critical to ensure that the predictions are aligned with actual outcomes, verifying the model's effectiveness in practical scenarios.

This methodical approach to **data splitting** ensures that each component of the model (AutoEncoder and regressors) is appropriately trained and validated, thereby enhancing the model's accuracy and reliability in practical applications.

```
# Split data into training, test, and validation sets
train_data = data_cleaned[(data_cleaned['bilancio_year'] >= 2015) & (
    data_cleaned['bilancio_year'] <= 2020)]
test_data = data_cleaned[(data_cleaned['bilancio_year'] >= 2016) & (
    data_cleaned['bilancio_year'] <= 2021)]
val_data = data_cleaned[(data_cleaned['bilancio_year'] >= 2017) & (
    data_cleaned['bilancio_year'] <= 2022)]
```

Listing 4.2: Data Splitting

4.2.4 Scaling

Scaling is essential to ensure the efficiency and robustness of the AutoEncoder. We use the `MinMaxScaler` from the `sklearn` library to **normalize**

the financial data. This transformation resizes the data to a standard range, between 0 and 1, improving the Temporal AutoEncoder's **convergence** and **numerical stability**.

The formula for the **MinMax Scaler** is given by:

$$X' = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

where:

- X is the original value,
- $X_{\min}$ is the minimum value of the feature,
- $X_{\max}$ is the maximum value of the feature,
- X' is the normalized value.

4.2.5 Temporal Series

Financial data is organized into **temporal series** to maintain sequential relationships between different years. This organization is crucial for the effectiveness of **time series-based models** such as the **LSTM AutoEncoder**. The temporal series are created by grouping data from the datasets established during the **data splitting** phase (namely: training, testing, and validation datasets).

Specifically, the data is divided into blocks of $\boldsymbol{n}$ years, where $n=6$ has been determined to be the most effective interval. This decision is based on a balance between **computational efficiency** and the accuracy of capturing significant temporal trends. The **six-year period** aligns well with the typical business and economic cycles, allowing the model to learn from a complete cycle of upward and downward trends, including anomalies such as COVID-19.

4.3 Temporal AutoEncoder - Encoding

The **Temporal AutoEncoder**, based on **LSTM**, is used to compress financial data into a **lower-dimensional space**. This technique reduces the dimensionality of the data while preserving essential **temporal features**. The model is built with `tensorflow` and `keras`, using **LSTM layers** to capture temporal dependencies.

4.3.1 Python Implementation

The following Python code defines the structure of a **Temporal AutoEncoder** using **Keras**. The 'DMITemporalAutoEncoder' class inherits from the base class 'DMIAutoEncoder'. The 'create_structure' method sets up the architecture with an **encoder** that compresses the input sequences into a lower-dimensional representation using an **LSTM layer**, and a **decoder** that reconstructs the original sequences from the encoded representation. The **inputs** are first passed through an **LSTM layer** to get the encoded output, which is then repeated for the specified time steps and passed through another **LSTM layer** to decode the sequences back to their original dimensions. The class provides the complete model, as well as separate **encoder** and **decoder** models.

```
class DMITemporalAutoEncoder(DMIAutoEncoder):

    def __init__(self, ncol, n_reduced_dims, time_stamps):
        self.ncol = ncol
        self.n_reduced_dims = n_reduced_dims
        self.time_stamps = time_stamps
        self.create_structure(ncol, n_reduced_dims, time_stamps)

    #Current model used for DMI
    def create_structure(self, ncol, n_reduced_dims, time_stamps):
        # define the model architecture
        inputs = Input(shape=(time_stamps, ncol))
```

```
        encoded = LSTM(n_reduced_dims)(inputs)
        decoded = RepeatVector(time_stamps)(encoded)
        decoded = LSTM(ncol, return_sequences=True)(decoded)
        self.model = Model(inputs, decoded)
        self.encoder = Model(inputs, encoded)
        self.decoder = Model(encoded, decoded)
```

Listing 4.3: TemporalAutoEncoder Model Structure

```
class DMIAutoEncoder(object):

    def __init__(self, ncol, n_reduced_dims, reduct=False,
    single_hidden_layer = False):
        self.ncol = ncol
        self.n_reduced_dims = n_reduced_dims
        if single_hidden_layer:
            self.create_structure_single_layer(ncol, n_reduced_dims)
        elif reduct:
            self.create_structure_lite(ncol, n_reduced_dims)
        else:
            self.create_structure(ncol, n_reduced_dims)
        self.encoder = Model(inputs = self.input_dim, outputs = self.
    encoded_layer)

    #Current model used for DMI
    def create_structure(self, ncol, n_reduced_dims):
        self.input_dim = Input(shape = (ncol, ))
        encoded1 = Dense(32, activation = 'relu')(self.input_dim)
        encoded2 = Dense(16, activation = 'relu')(encoded1)
        encoded3 = Dense(8, activation = 'relu')(encoded2)
        self.encoded_layer = Dense(n_reduced_dims, activation = 'relu')(
    encoded3)
        decoded1 = Dense(8, activation = 'relu')(self.encoded_layer)
        decoded2 = Dense(16, activation = 'relu')(decoded1)
        decoded3 = Dense(32, activation = 'relu')(decoded2)
        decoded4 = Dense(ncol, activation = 'sigmoid')(decoded3)
        self.model = Model(inputs = self.input_dim, outputs = decoded4)

    def create_structure_lite(self, ncol, n_reduced_dims):
        self.input_dim = Input(shape = (ncol, ))
        encoded2 = Dense(16, activation = 'relu')(self.input_dim)
        encoded3 = Dense(8, activation = 'relu')(encoded2)
        self.encoded_layer = Dense(n_reduced_dims, activation = 'relu')(
    encoded3)
        decoded1 = Dense(8, activation = 'relu')(self.encoded_layer)
        decoded2 = Dense(16, activation = 'relu')(decoded1)
        decoded4 = Dense(ncol, activation = 'sigmoid')(decoded2)
```

```
35          self.model = Model(inputs = self.input_dim, outputs = decoded4)
36
37      def create_structure_single_layer(self, ncol, n_reduced_dims):
38          self.input_dim = Input(shape = (ncol, ))
39          self.encoded_layer = Dense(n_reduced_dims, activation = 'relu')(
    self.input_dim)
40          decoded = Dense(ncol, activation = 'sigmoid')(self.encoded_layer)
41          self.model = Model(inputs = self.input_dim, outputs = decoded)
42
43      def train(self, X, epochs=10, batch_size=8, optimizer='adam', loss='
    mse'):
44          self.model.compile(optimizer = optimizer, loss = loss)
45          self.model.fit(X, X, epochs=epochs, batch_size=batch_size)
46
47      def encode_data(self, data):
48          reduced_df = pd.DataFrame(self.encoder.predict(data, verbose=1))
49          return reduced_df.add_prefix('feature_')
50
51      #deprecated
52      def save_model(self, model_dir, options):
53          self.model.save(model_dir, options=options)
54          with open(f'{model_dir}/aencoder.pickle', 'wb') as f:
55              pk.dump(self.model, f)
56
57      def load_weights(self, weights_path):
58          self.model.load_weights(weights_path)
59
60      def _save_weights(self, save_folder):
61          self.model.save_weights(f'{save_folder}/weights.h5')
62
63      def _save_parameters(self, save_folder):
64          parameters = [self.ncol, self.n_reduced_dims]
65          with open(f'{save_folder}/parameters.pkl', "wb") as f:
66              pk.dump(parameters, f)
67
68      @classmethod
69      def load(cls, save_folder="."):
70          with open(f'{save_folder}/parameters.pkl', "rb") as f):
71              parameters = pk.load(f)
72          AutoEncoder = DMIAutoEncoder(*parameters)
73          AutoEncoder.load_weights(f'{save_folder}/weights.h5')
74          return AutoEncoder
75
76      def save(self, save_folder="."):
77          self._create_folder_if_it_doesnt_exist(save_folder)
78          self._save_parameters(save_folder)
79          self._save_weights(save_folder)
```

```
80
81      def _create_folder_if_it_doesnt_exist(self, folder):
82          if not os.path.exists(folder):
83              os.makedirs(folder)
```

Listing 4.4: Base AutoEncoder Model Structure

4.4 Prediction with Regressors

Instead of employing a single model approach, the regression phase in our implementation iterates through a variety of **machine learning models**, testing each one across different dimensions of the **latent space** created by the **AutoEncoder**. This iterative method allows for the creation of multiple predicted datasets, each corresponding to a unique combination of regression model and latent space dimension. The purpose of this comprehensive approach is to identify the most effective model and dimensionality that yield the best predictive accuracy.

4.4.1 Embedded Regressors

The models used in this phase include **XGBoost**, **Random Forest**, **AdaBoost**, **Decision Trees**, and **Multi-layer Perceptrons**. These models were selected for their diverse characteristics in handling regression tasks, providing a broad spectrum of learning strategies from **ensemble methods** to **neural networks**.

4.4.2 Implementation Libraries

- `TensorFlow` and `Keras` are used for building and training the **Temporal AutoEncoder**, providing powerful tools for deep learning and neural network models.

- `XGBoost` is implemented using the `xgboost` library, known for its **efficiency** and **effectiveness** in dealing with structured data.

- `Random Forest` and `AdaBoost` are utilized from the `sklearn.ensemble` module, which provides an optimized collection of **ensemble methods**.

- `Decision Tree` models are accessed via `sklearn.tree`, offering a straightforward approach to learning **decision rules** from data.

- `Multi-layer Perceptron` regressors are implemented using `sklearn.neural_network`, which provides a flexible framework for constructing **neural networks**.

Each combination of model and latent dimension generates a new predicted dataset, which is subsequently utilized in the validation phase.

4.5 Temporal AutoEncoder - Decoding

After prediction, the data is decoded from the compressed space using the decoding part of the **Temporal AutoEncoder**. This step reconstructs the original financial data from the low-dimensional representations.

4.6 Inverse Pre-Processing

The reconstructed data is returned to its original scale using the inverse of the scaling transformation and series creation applied during pre-processing.

4.7 Model Evaluation

The predictions generated by the regression models are compared with the actual data and evaluated for accuracy using **Mean Absolute Error (MAE)**.

```
import numpy as np
import pandas as pd
from sklearn.metrics import mean_absolute_error
import joblib
from sklearn.ensemble import RandomForestRegressor, AdaBoostRegressor
from sklearn.multioutput import MultiOutputRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.neural_network import MLPRegressor
import xgboost as xgb
import os

# Define the dimensions to test
dims_to_test = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
     18, 19, 20]
ncol = train_series_scaled.shape[2]
time_stamps = train_series_scaled.shape[1]
num_features = train_series_scaled.shape[2]

# Define the regressor models
models = {
    'XGBoost': xgb.XGBRegressor(objective='reg:absoluteerror',
    n_estimators=100, learning_rate=0.1),
    'RandomForest': RandomForestRegressor(n_estimators=100),
    'AdaBoost': MultiOutputRegressor(AdaBoostRegressor(n_estimators=100,
    learning_rate=0.1)),
    'DecisionTree': MultiOutputRegressor(DecisionTreeRegressor()),
    'MLP': MultiOutputRegressor(MLPRegressor(verbose=False))
}

var_epochs = 5
var_batch = 64

# Placeholder for results
results = []

# Function to encode data using the Temporal AutoEncoder
def encode_data(AutoEncoder, data):
    return AutoEncoder.encode_data(data)

```

```
# Function to decode data using the Temporal AutoEncoder
def decode_data(AutoEncoder, data):
    return AutoEncoder.decode_data(data)

# Loop over each dimension
for n_reduced_dims in dims_to_test:
    AutoEncoder = DMITemporalAutoEncoder(ncol, n_reduced_dims, time_stamps)
    AutoEncoder.train(train_series_scaled, epochs=var_epochs, batch_size=var_batch, optimizer='adam', loss='mse')

    encoded_train_series = encode_data(AutoEncoder, train_series_scaled)
    encoded_test_series = encode_data(AutoEncoder, test_series_scaled)
    encoded_val_series = encode_data(AutoEncoder, val_series_scaled)

    # Loop over each model
    for model_name, model in models.items():
        # Train the model on encoded training data
        model.fit(encoded_train_series, encoded_test_series)

        # Predict on the validation set
        pred = model.predict(encoded_test_series)
        pred = pred.reshape(encoded_val_series.shape[0], n_reduced_dims, 1)
        decoded_pred = decode_data(AutoEncoder, pred)

        # Extract every 6th row for MAE computation
        val_series_last_rows = val_series_reshaped[5::6]
        descaled_pred_last_rows = decoded_pred[5::6]

        # Compute Mean Absolute Error (MAE) on the extracted rows
        mae = mean_absolute_error(val_series_last_rows, descaled_pred_last_rows)
        results.append({'Dimension': n_reduced_dims, 'Model': model_name, 'MAE': mae})
```

Listing 4.5: Iterative Model Evaluation Script

The code snippet above details the iterative process of testing various **machine learning models** across different **latent space** dimensions. Each model is trained on the **encoded training and test data** and then evaluated on the **validation data** to determine the most effective model and dimension for accurate financial forecasting. The **batch size** and **number of epochs** were chosen through a careful process to balance training time

and model performance, ensuring that the models could learn effectively without overfitting or underfitting.

Chapter 5

Testing

This chapter outlines how the system is validated, highlighting the **metrics** used, the **data comparison**, and the role of the **benchmark** in determining model performance.

5.1 Benchmark Definition

The **benchmark** for this testing phase is set by expecting the data point for a company in year t to be the same as its data from year $t-1$. This method, often used in the financial sector, serves as a **conservative estimate** against which the performance of more complex models is measured. The computed benchmark **Mean Absolute Error** (MAE) for this study is determined to be 1.16008 on year 2022, as calculated from historical data points in the given dataset. A higher error rate than the benchmark is considered suboptimal.

5.2 Model Evaluation

To validate the model, an **iterative approach** is adopted, where multiple **regression models** are tested across various dimensions of the **latent space**. The aim is to identify the combination of model and dimensionality that minimizes **prediction MAE**.

The **Mean Absolute Error** (MAE) is calculated using the following equation:

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{5.1}$$

where:

- n is the number of data points.
- y_i is the actual value for the i-th data point.
- $\hat{y}_i$ is the predicted value for the i-th data point.
- $|\cdot|$ denotes the **absolute value**.

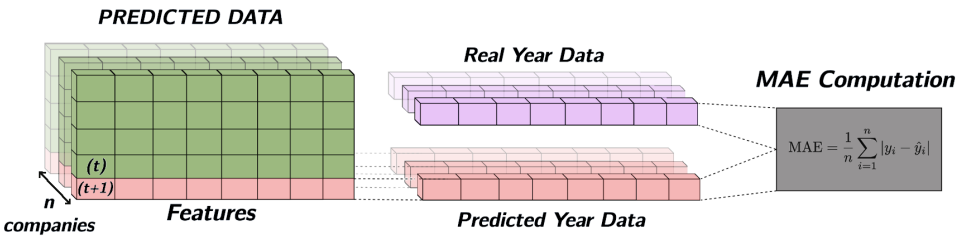

Figure 5.1: Example of MAE Computation for Model Validation

5.2.1 Latent Space Exploration

The **latent space dimensions** tested range from 1 to 20. For each dimension, the **Temporal AutoEncoder** encodes the training data, which is then used to train the **regressors**. The regressors predict the encoded test data, which is subsequently decoded to compare against the validation set.

5.3 Results

As stated before, the performance of each model across different dimensions is evaluated using **Mean Absolute Error** (MAE). The results indicate varying levels of accuracy, with certain models and dimensions achieving lower error rates, thus outperforming the benchmark. These findings are visualized through **graphs** that plot MAE against the latent space dimensions for each model, providing a clear depiction of each model's performance relative to others. In Table 4.1 the data the model has been validated on is the predicted 2022 (using encoded data being from the previous 6 years) and the real 2022.

Dimension	1	2	3	4	5	6	7	8	9
AdaBoost	**0.940**	1.073	0.931	1.179	0.967	1.166	1.162	1.181	1.117
DecisionTree	**0.917**	1.068	0.995	1.240	1.149	1.228	1.277	1.347	1.246
MLP	**0.868**	0.975	0.871	1.110	0.937	1.049	1.081	1.071	1.030
RandomForest	**0.901**	1.028	0.937	1.124	0.991	1.111	1.111	1.141	1.050
XGBoost	**0.874**	1.009	0.934	1.118	0.997	1.108	1.124	1.142	1.059

Table 5.1: Performance of Different Models Across Various Dimensions in 6 years long Time Series

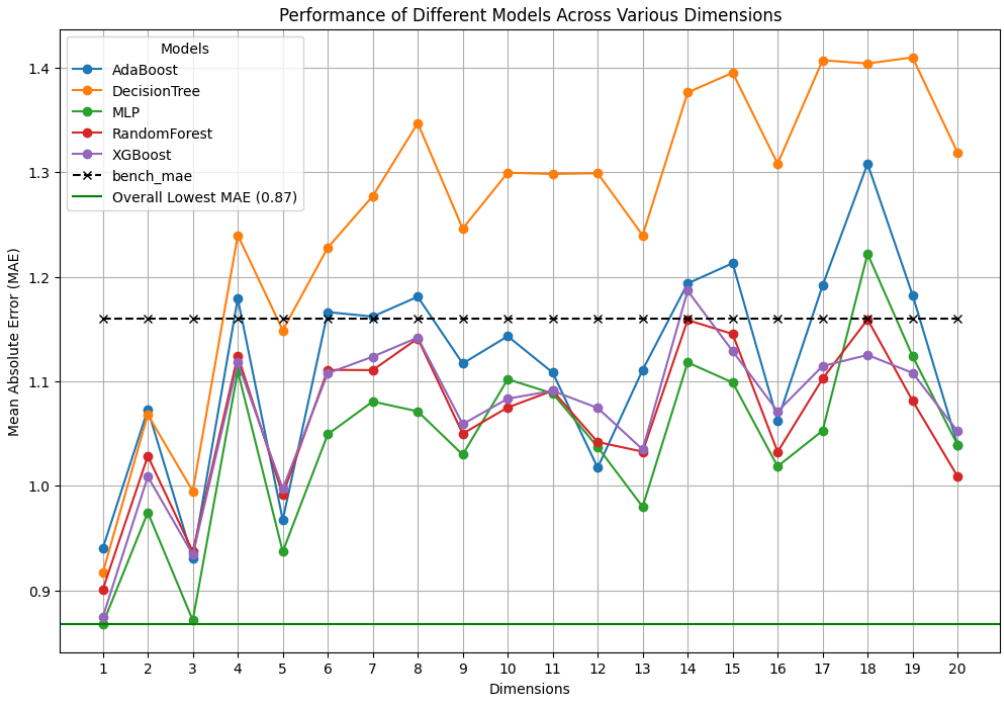

Table 5.2: Result Comparative Line Graph with 6 years long Time Series

5.4 Discussion

The graph (Figure 3.1) displays the performance of different models across various dimensions, specifically measuring the **Mean Absolute Error** (MAE) on the y-axis and the dimensions on the x-axis. Each model's performance is represented by a different **color** and **marker style**.

The observed trend in the plot where lower **latent space dimensions** often result in lower prediction errors is attributed to the reduced complexity of the model. Lower dimensions may generalize better, avoiding **overfitting** to noise in the data. This effect is particularly pronounced in **financial data**, where simpler models can capture significant trends without being misled by minor fluctuations [6].

5.4.1 Problem Assessment

The collection of data during the **COVID-19 pandemic** presents a significant challenge for accurate predictions. The pandemic has introduced unprecedented volatility and economic disruptions, making the financial data from this period highly atypical and harder to predict. [10]

As shown in Table 5.3, the benchmark MAE values are considerably higher during the COVID-19 years, reflecting the increased difficulty in making accurate predictions. This increased error rate demonstrates the impact of the pandemic on financial data, validating the problem of predicting in such volatile conditions. [13]

Year	**2015**	**2016**	**2017**	**2018**	**2019**	**2020**	**2021**
Before Outlier Removal							
MAE	1.548	1.613	1.545	1.622	1.804	1.870	1.931
After Removing 2% Outliers							
MAE	0.995	0.969	0.942	0.900	1.105	1.281	1.160

Table 5.3: Benchmark MAE for Each Year Before and After Outlier Removal.

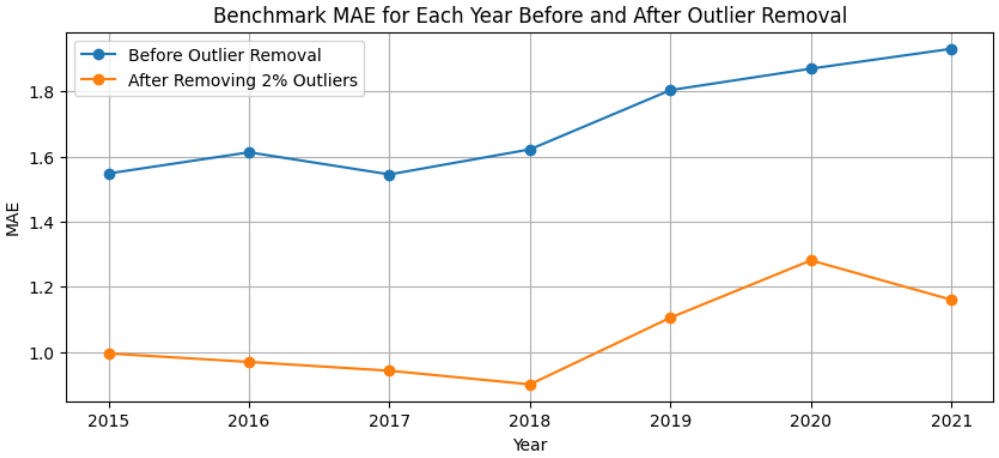

Table 5.4: Line Graph displaying increased Benchmark MAE during COVID years.

Given this challenge, the next step was to investigate with shorter time series on data pre-COVID, predicting the year 2018. This approach yielded the lowest error MAE in MLP with 1 dimension at 0.717. This suggests that, as expected, stable data before the pandemic can be predicted more accurately.

Dimension	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
AdaBoost	**0.778**	0.835	0.776	0.865	0.854	0.939	1.019	0.912	1.049
DecisionTree	**0.785**	0.846	0.850	0.991	1.017	1.043	1.076	1.105	1.196
MLP	**0.717**	0.774	0.792	0.798	0.828	0.799	0.878	0.828	0.993
RandomForest	**0.767**	0.807	0.809	0.837	0.870	0.881	0.883	0.881	1.009
XGBoost	**0.744**	0.794	0.810	0.834	0.880	0.885	0.875	0.877	1.018

Table 5.5: Performance of Different Models Across Various Dimensions with 3 years long Time Series on Stable Pre-COVID Data.

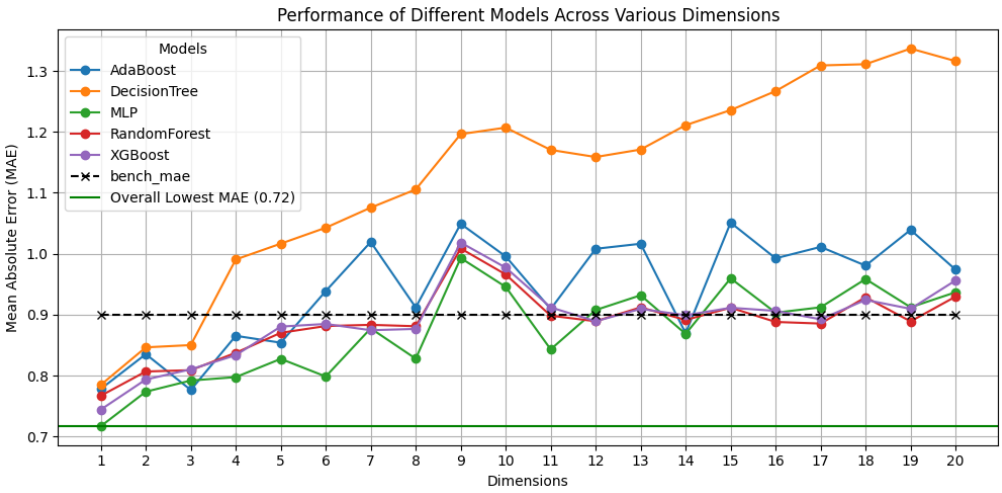

Table 5.6: Result Comparative Line Graph with 3 years long Time Series on Stable Pre-COVID Data.

To further demonstrate the fact that it is harder to predict during the COVID-19 years, and to assess the reliability of the model, predictions for the year 2022 were made using data from the previous 3 years (2019, 2020, 2021), which are the COVID-19 years. As expected, the results were suboptimal, with a best prediction MAE of 0.99 for the year 2022.

Dimension	1	2	3	4	5	6	7	8	9
AdaBoost	1.061	1.090	1.114	1.175	1.119	1.147	1.154	1.380	**1.063**
DecisionTree	**1.044**	1.131	1.195	1.299	1.163	1.268	1.302	1.436	1.266
MLP	**0.989**	1.047	1.046	1.103	1.030	1.055	1.050	1.292	0.996
RandomForest	**1.035**	1.077	1.101	1.120	1.054	1.095	1.078	1.238	1.050
XGBoost	**1.012**	1.057	1.098	1.117	1.034	1.106	1.069	1.207	1.077

Table 5.7: Performance of Different Models Across Various Dimensions with 3 years long Time Series on COVID Data.

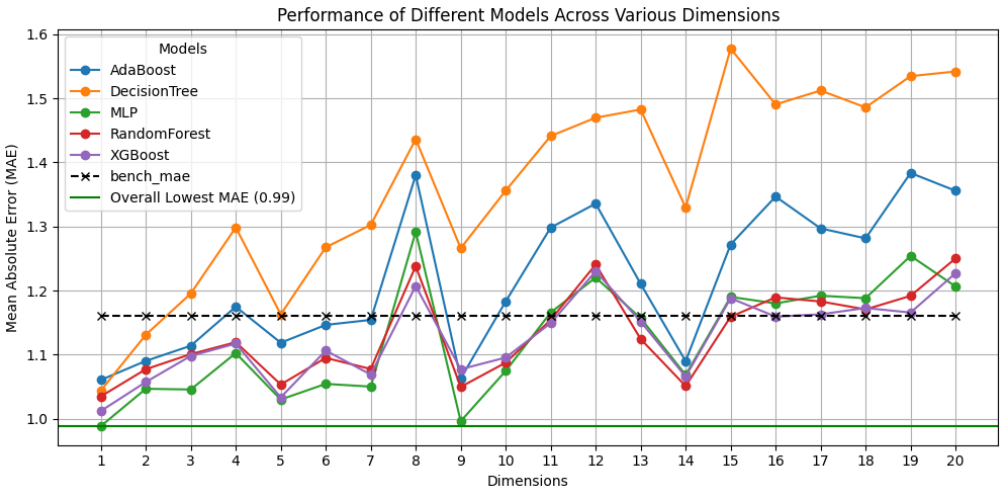

Table 5.8: Result Comparative Line Graph with 3 years long Time Series on COVID Data.

This investigation also showed that a 6-year time series improves the robustness of the model towards crisis events. The model using the 6-year time series managed to obtain a prediction MAE of 0.867 for the year 2022 (as shown in Figure 5.2), demonstrating its superior performance in handling volatile periods compared to shorter time series. This finding reinforces the value of incorporating longer historical data to enhance the predictive capability and resilience of the model.

Chapter 6

Conclusions

This thesis has successfully demonstrated the integration of advanced **machine learning** techniques into financial forecasting, showing a marked improvement in prediction accuracy over traditional methods. The primary objective was to design and implement a software framework capable of predicting financial outcomes by exploring the latent space generated by an AutoEncoder, and this has been achieved through state-of-the-art methodology and rigorous testing.

Achievements and Summary of Findings

The use of **Temporal AutoEncoders** with LSTM networks has proven effective in embedding financial data into a lower-dimensional space, capturing essential patterns that are not immediately apparent in high-dimensional raw data. This transformation not only facilitates more accurate predictions but also reduces computational costs, making the process more efficient. The iterative approach of testing various models and latent dimensions helped in pinpointing the optimal configuration for our predictive models, outperforming the set **benchmark**.

The impact of **COVID-19** on financial data prediction was specifically investigated, demonstrating the increased difficulty in making accurate predictions during such high volatility periods. The findings indicated that while shorter time series (3 years) during the COVID-19 years resulted in higher prediction errors, a longer time series (6 years) improved the model's robustness and accuracy, achieving a prediction MAE of 0.867 for the year 2022.

Challenges and Limitations

Despite the successes, this work is not without its limitations. The reliance on financial data from annual reports means that the dataset predominantly represents well-established companies, potentially overlooking the volatility and unique challenges faced by newer businesses and startups. Additionally, the scope of data, covering a span of ten years, may not encompass the full economic cycles necessary for a comprehensive financial forecast.

The unpredictable nature of events such as the COVID-19 pandemic poses a significant challenge for financial models. This study highlights the need for models to adapt to sudden economic disruptions, as evidenced by the higher MAE during the pandemic years.

Future Directions

Looking ahead, there are several avenues for further research and development:

1. **Expansion of Data Scope**: Including a broader array of companies, especially newer and smaller enterprises, could enhance the

model's applicability and robustness across different economic sectors and conditions.

2. **Longer Time Frames**: The data is limited to 10 years. Extending the dataset to cover more economic cycles could provide deeper insights into long-term trends and shifts, further refining prediction accuracy.

3. **Advanced Models and Techniques**: Exploring newer or more complex machine learning models could uncover subtler patterns in the data, potentially leading to breakthroughs in predictive accuracy. Although there is potential for improvement, the data is limited to a 10-year series, and employing more complex models may introduce a computational bottleneck, potentially limiting their practical utility.

4. **Adaptability to Economic Shocks**: Enhancing the models to better handle unexpected events such as financial crises or pandemics can improve their robustness and reliability. This could involve integrating external economic indicators or developing hybrid models that can adapt to changing conditions.

Concluding Remarks

The journey through this thesis has not only underscored the potential of **machine learning** in financial forecasting but also highlighted the critical importance of data quality, model choice, and the need for continuous adaptation to changing economic landscapes. As we advance, the integration of more dynamic data sources and the continuous refinement of models will be key to maintaining the relevance and efficacy of these predictive frameworks. The findings from this study provide a compelling case for the continued exploration of machine learning techniques in financial

time series analysis. The ability of these models to capture intricate patterns and relationships within financial data, positions them as powerful tools for enhancing prediction accuracy and decision-making in financial markets [4].

Acknowledgments

Mi sento di ringraziare tutti coloro che hanno reso possibile questo percorso. Senza di voi, non sarei mai arrivato fin qui.

Un ringraziamento al mio relatore Alessandro Giuliani per avermi seguito durante la stesura della tesi. La sua pazienza e i suoi consigli sono stati fondamentali per portare a termine questo lavoro.

Un ringraziamento speciale va a mia madre e mio padre. Grazie per essere sempre stati d'aiuto nei momenti di bisogno, per non avermi mai fatto mancare niente e per avermi sopportato anche quando, stressato dai mille impegni, diventavo insopportabile.

Grazie alla mia famiglia: nonna Teresa, zia Giulia, zio Massimo e Zia Manuela, zio Stefano, zia Simona e Thor (anche lui ha fatto la sua parte con il suo supporto peloso). Grazie per avermi sempre ascoltato e dato una mano quando avevo dubbi e bisogno di consigli.

Un ringraziamento speciale alla mia ragazza Carlotta per essere sempre stata al mio fianco, avermi sempre aiutato in tutto e per tutto, e per essere bellissima. La tua presenza è stata una costante fonte di supporto e ispirazione.

Grazie ai miei colleghi Pietro, Alessandro, Giovanni, Marco, Arturo e Samuele. Grazie per le sessioni di studio, le speedrun, i progetti che ci facevano impazzire, ma anche le uscite e le cavolate che hanno reso questi

3 anni divertenti e molto più leggeri. Abbiamo affrontato insieme tante sfide e condiviso molti momenti indimenticabili.

Un ringraziamento di cuore va al mio amico Matteo per avermi sempre dato conforto e sostenuto durante tutti questi anni. Sei l'amico che tutti vorrebbero avere, e sono grato di averti nella mia vita. La tua presenza ha reso più facili i momenti difficili.

Un grandissimo ringraziamento va al mio amico Roberto. Grazie per avermi sempre motivato, per avermi spinto a dare il massimo in ogni situazione e per avermi fatto credere in me stesso anche nei momenti più difficili. Ci sono poche persone come te, capaci di infondere fiducia e ambizione. Sono sicuro che insieme faremo grandi cose e raggiungeremo traguardi straordinari.

Infine, un ringraziamento va ai miei amici del gruppo "Savoia" – Nicola, Daniel, Mattia, Diego, Alessandro, Lorenzo, Riccardo, Edoardo e Alberto. Compagni di pause studio e nottate passate a parlare e giocare, avete reso questi anni ancora più memorabili e sopportabili.

Grazie a tutti voi, senza il vostro supporto e la vostra amicizia, questo percorso non sarebbe stato lo stesso.

Bibliography

[1] Bao, D.: A generalized model for financial time series representation and prediction. Applied Intelligence **29**(1), 1–11 (2008). https://doi.org/10.1007/s10489-007-0063-1, https://doi.org/10.1007/s10489-007-0063-1

[2] Barra, S., Carta, S.M., Corriga, A., Podda, A.S., Recupero, D.R.: Deep learning and time series-to-image encoding for financial forecasting. IEEE/CAA Journal of Automatica Sinica **7**(3), 683–692 (2020)

[3] Berahmand, K., Daneshfar, F., Salehi, E.S., Li, Y., Xu, Y.: Autoencoders and their applications in machine learning: a survey. Artificial Intelligence Review **57**(2), 28 (2024). https://doi.org/10.1007/s10462-023-10662-6, https://doi.org/10.1007/s10462-023-10662-6

[4] Bieganowski, B., Ślepaczuk, R.: Supervised autoencoder mlp for financial time series forecasting. SSRN Electronic Journal (2024), https://api.semanticscholar.org/CorpusID:268856570

[5] Cai, B., Yang, S., Gao, L., Xiang, Y.: Hybrid variational autoencoder for time series forecasting. ArXiv **abs/2303.07048** (2023), https://api.semanticscholar.org/CorpusID:257496496

[6] Chen, S., Guo, W.: Auto-encoders in deep learning—a review with new perspectives. Mathematics **11**(8) (2023). https://doi.org/10.3390/math11081777, https://www.mdpi.com/2227-7390/11/8/1777

[7] Cortés, D., Onieva, E., López, I., Trinchera, L., Wu, J.: Autoencoder-enhanced clustering: A dimensionality reduction approach to financial time series. IEEE Access **PP**, 1–1 (01 2024). https://doi.org/10.1109/ACCESS.2024.3359413

[8] Durairaj, D.M., Mohan, B.H.K.: A convolutional neural network based approach to financial time series prediction. Neural Computing and Applications **34**(16), 13319–13337 (2022). https://doi.org/10.1007/s00521-022-07143-2, https://doi.org/10.1007/s00521-022-07143-2

[9] Ito, H., Murakami, A., Dutta, N., Shirota, Y., Chakraborty, B.: Clustering of ETF Data for Portfolio Selection during Early Period of Corona Virus Outbreak. Gakushuin Economic Papers **58**(1), 99–114 (2021), https://ideas.repec.org/a/abc/gakuep/58-1-6.html

[10] Laborda, R., Olmo, J.: Volatility spillover between economic sectors in financial crisis prediction: Evidence spanning the great financial crisis and covid-19 pandemic. Research in International Business and Finance **57**, 101402 (2021). https://doi.org/https://doi.org/10.1016/j.ribaf.2021.101402, https://www.sciencedirect.com/science/article/pii/S0275531921000234

[11] Lee, K., Jeong, Y., Joo, S., Yoon, Y.S., Han, S., Baik, H.: Outliers in financial time series data: Outliers, margin debt, and economic recession. Machine Learning with Applications **10**, 100420 (2022). https://doi.org/https://doi.org/10.1016/j.mlwa.2022.100420, https://www.sciencedirect.com/science/article/pii/S2666827022000950

[12] Liu, J.: Navigating the financial landscape: The power and limitations of the arima model **88**, 747–752 (Mar 2024).

https://doi.org/10.54097/9zf6kd91, https://drpress.org/ojs/index.php/HSET/article/view/19082

[13] Mroua, M., Lamine, A.: Financial time series prediction under covid-19 pandemic crisis with long short-term memory (lstm) network. Humanities and Social Sciences Communications **10**(1), 530 (2023). https://doi.org/10.1057/s41599-023-02042-w, https://doi.org/10.1057/s41599-023-02042-w

[14] Nguyen, N.H., Quanz, B.: Temporal latent auto-encoder: A method for probabilistic multivariate time series forecasting (2021), https://api.semanticscholar.org/CorpusID:231709435

[15] Pareek, J., Jacob, J.: Data compression and visualization using pca and t-sne. In: Goar, V., Kuri, M., Kumar, R., Senjyu, T. (eds.) Advances in Information Communication Technology and Computing. pp. 327–337. Springer Singapore, Singapore (2021)

[16] Puspita, P.E., Zulkarnain: A practical evaluation of dynamic time warping in financial time series clustering. In: 2020 International Conference on Advanced Computer Science and Information Systems (ICACSIS). pp. 61–68 (2020). https://doi.org/10.1109/ICACSIS51025.2020.9263123

[17] Van Hoa, T., Tuan Anh, D., Ngoc Hieu, D.: Foreign exchange rate forecasting using autoencoder and lstm networks. In: Proceedings of the 2021 6th International Conference on Intelligent Information Technology. p. 22–28. ICIIT '21, Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3460179.3460184, https://doi.org/10.1145/3460179.3460184

[18] Wong, L., Liu, D., Berti-Équille, L., Alnegheimish, S., Veeramachaneni, K.: Aer: Auto-encoder with regression for time series anomaly

detection. 2022 IEEE International Conference on Big Data (Big Data) pp. 1152–1161 (2022), https://api.semanticscholar.org/CorpusID:255186008

[19] Wu, R., Keogh, E.J.: Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress. IEEE Transactions on Knowledge and Data Engineering **35**, 2421–2429 (2020), https://api.semanticscholar.org/CorpusID:221995939

List of Figures

List of Tables