The present work was submitted to the Institute of Computational Social Sciences and Humanities

Data-Driven Quasi Taylor-Order Analysis for Collaborative Filtering of Reviews

Master Thesis

Presented by Ashish Rajani
Matr.-Nr. 416943

Supervised by Univ.-Prof. Dr. Jürgen Lerner
Univ.-Prof. Dr. Uwe Naumann

Aachen, July 12, 2023

Contents

I Acknowledgements

II Abstract

III List of Figures

IV List of Definitions

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Thesis Structure . . . 2

2 Theory 3
- 2.1 Regression . . . 3
  - 2.1.1 Taylor Model . . . 4
- 2.2 Separability . . . 4
- 2.3 Model Reduction . . . 7
- 2.4 Training and Model Reassembling . . . 8
- 2.5 Sampling . . . 8
  - 2.5.1 Random sampling . . . 9
  - 2.5.2 Full-factorial sampling . . . 9
  - 2.5.3 Latin-Hypercube sampling . . . 10
  - 2.5.4 Grid sampling . . . 10

3 Implementation 15
- 3.1 Model . . . 16
- 3.2 Sampling . . . 17
  - 3.2.1 Domain Sampling . . . 18
  - 3.2.2 Data sampling . . . 18
- 3.3 Model Reduction via Separability Analysis . . . 20
- 3.4 Modelling Non-Separable Dimensions . . . 22
- 3.5 Configuration Options . . . 23
- 3.6 Usage . . . 23

4 Experiments 25
- 4.1 Experiment I – Synthetic Dataset . . . 28
  - 4.1.1 Data Generation . . . 28
  - 4.1.2 Results . . . 28
- 4.2 Experiment II – Collaborative Filtering of Reviews . . . 34
  - 4.2.1 Literature Review . . . 34
  - 4.2.2 Methodology . . . 35
  - 4.2.3 Results . . . 41

5 Conclusion 47
- 5.1 Summary & Discussion . . . 47
- 5.2 Further Research . . . 48
- 5.3 Additional Resources . . . 49

V References

A Appendix

A.1 YAML Configuration File . . . . . .
A.2 Source Code . . . . . .
A.3 Results . . . . . .
A.3.1 Synthetic Dataset: ElasticNet . . . . . .
A.3.2 Synthetic Dataset: ANN . . . . . .
A.3.3 Synthetic Dataset: RSQTOA + Grid Sampling . . . . . .
A.3.4 Synthetic Dataset: RSQTOA + Interpolation . . . . . .
A.3.5 Yelp Dataset: ElasticNet . . . . . .
A.3.6 Yelp Dataset: ANN . . . . . .
A.3.7 Yelp Dataset: RSQTOA + Grid Sampling . . . . . .
A.3.8 Yelp Dataset: RSQTOA + Interpolation . . . . . .

I Acknowledgements

I would like to extend my deepest gratitude and appreciation to the remarkable individuals who have played a pivotal role in the successful completion of my Master's thesis. Their guidance, support, and unwavering encouragement have been instrumental in shaping this research journey.

First and foremost, I would like to express my sincere thanks to Prof. Jürgen Lerner for his invaluable guidance and continuous support throughout my research. His expertise in the field and insightful suggestions have not only enriched my understanding but also inspired me to explore the social applications of the subject. Prof. Lerner's mentorship and encouragement have been a constant source of motivation for me to persevere and strive for excellence.

I am also deeply grateful to Prof. Uwe Naumann for entrusting me with this topic and for his consistent support and mentorship. His profound knowledge and guidance have been invaluable in navigating the technical challenges that emerged during the course of my research. Prof. Naumann's unwavering support has instilled confidence in me and has played a significant role in the success of this thesis.

Furthermore, I would like to express my heartfelt appreciation to Simon Märtens, my thesis supervisor. His unwavering support, guidance, and availability have been remarkable. His willingness to share his expertise, offer insightful perspectives, and patiently assist me in overcoming technical challenges have been integral to the accomplishment of this thesis.

Lastly, I would like to extend my heartfelt thanks to all the individuals who have supported me throughout this research journey, including my family and friends. Their unwavering encouragement, understanding, and patience have been invaluable to me. Their belief in my abilities and constant motivation have kept me focused and determined, even during challenging times. I am grateful for their presence in my life and for being a constant source of inspiration.

II Abstract

This thesis introduces an innovative approach to address the issue of dimensionality in high-dimensional regression models. Building upon existing research, where the proposed framework proposes an approach reducing the dimensionality of multivariate scalar functions. The framework incorporates a concept known as *Quasi-Separability* to assess the separability of dimensions. Additionally, it integrates an Artificial Neural Network (ANN) to approximate non-reducible dimensions, resulting in surrogate models that are both cost-effective and accurate.

To further enhance the framework's capabilities, this research introduces two new techniques: a grid-based sampling approach and a dataset approximation technique using interpolation. These techniques enable the framework to produce accurate models directly from datasets, overcoming the challenge of lacking target values for each sample.

To evaluate the performance of the proposed framework, it is tested on two real-world use cases. A comparative analysis is conducted, comparing the results with those obtained from linear regression (specifically ElasticNet) and ANN-based regression models. The findings demonstrate the successful implementation of the proposed methods, allowing the framework to generate approximate models from datasets with comparable accuracy to existing models. Moreover, the proposed framework achieves significantly lower training times compared to ANN-based models, making it a promising alternative in terms of efficiency.

III List of Figures

1 Random sampling ($N = 9$) . . . 9
2 Full-factorial sampling ($N = 3^2 = 9$) . . . 10
3 Latin-Hypercube sampling ($N = 9$) . . . 11
4 Multi-Dimensional Grid (s^d sections) . . . 11
5 Plane. . . . 12
6 Projection . . . 12
7 Filtering . . . 12
8 Sampled points . . . 12
9 Illustration of the position of data points within a section with respect to an imaginary line (anchor) parallel to the x_3 axis. . . . 13
10 Regression Analysis of Sub-Surface Relationship for x_3 . . . 13
11 UML Class diagram of the *RSQTOA* framework Märtens [2022] . . . 15
12 Updated UML Class diagram of the *RSQTOA* framework. . . . 16
13 UML Class diagram for the *Model* class and its subclasses . . . 17
14 UML Class diagram for the *Sampler* class and its subclasses . . . 17
15 UML Class diagram for the *DomainSampler* class and its subclasses . . . 18
16 UML Class diagram for the *DataSampler* class and its subclass . . . 18
17 UML diagram of the *Quasi-Separability* regression analysis Märtens [2022] . . . 20
18 Experimental Setup and Methodology Overview . . . 26
19 Average (20 Folds) Loss History for ANN on Synthetic Dataset . . . 29
20 Average (20 Folds) Loss History for RSQTOA with Grid Sampling on Synthetic Dataset . . . 30
21 Average (20 Folds) Loss History for RSQTOA with Nearest Neighbour Interpolator on Synthetic Dataset . . . 31
22 R^2 Scores comparison for Synthetic Dataset . . . 32
23 Training time comparison for Synthetic Dataset . . . 33
24 Mutual Information Regression Scores . . . 40
25 Average (20 Folds) Loss History for ANN on Yelp Dataset . . . 42
26 Average (20 Folds) Loss History for RSQTOA with Grid Sampling on Yelp Dataset . . . 43
27 Average (20 Folds) Loss History for RSQTOA with Nearest Neighbour Interpolator on Yelp Dataset . . . 44
28 R^2 Scores comparison for Yelp Dataset . . . 45
29 Training time comparison for Yelp Dataset . . . 46

IV List of Definitions

1 Definition (Partial Separability) 5
2 Definition (Dimensional Separability) 5
3 Definition (Full Separability) 6
4 Definition (Quasi-Separability) 6
1 Corollary (Quasi-Taylor) 7
5 Definition (Quasi-Reducibility) 7

1 Introduction

1.1 Motivation

The present thesis focuses on proposing a novel approach to mitigate the impact of the *Curse of Dimensionality* in regression methods, building upon previous research that proposes a framework for reducing the dimensionality of multivariate scalar functions Märtens [2022]. High-dimensional multivariate functions are prevalent in large-scale simulations and finding approximations for computationally expensive models becomes increasingly challenging with the number of dimensions. Consequently, several techniques have been developed to reduce the complexity of high-dimensional problems, with researchers and practitioners in Computational Science, Engineering, and Finance aiming to construct inexpensive surrogate models for computationally expensive simulation programs.

Social media platforms are increasingly relying on data analytics to personalize content recommendations for their users. However, the computational complexity and time required for real-time analysis pose challenges in delivering timely and relevant recommendations. To address this, surrogate models have emerged as a viable solution. These models capture user preferences and behaviours based on a subset of interactions and preferences, enabling quick predictions of content relevance. Although surrogate models may not capture the full complexity of individual user preferences, their ability to provide reasonable approximations has been demonstrated in research Smith et al. [2019]. Periodic evaluations and feedback loops help refine the surrogate models over time, enhancing their predictive capabilities Jones et al. [2020]. The adoption of surrogate models in social media platforms not only enhances user experience by delivering relevant and engaging content promptly but also improves computational efficiency by reducing the burden of real-time analysis Brown and Lee [2018].

The proposed approach presented in this study builds upon the Randomised Subspace Quasi Taylor-Order Analysis (RSQTOA) framework Märtens [2022] to enhance its capabilities for generating approximate models from both datasets and functions. The primary goal of the approach is to reduce the computational cost associated with approximating large-scale problems, which is often hindered by the high dimensionality of the data. By exploiting the unique mathematical properties of the function being approximated, such as *quasi-linearity*, it is possible to reduce the dimensionality of the problem and improve efficiency. Research conducted by Kauerauf [2021] specifically explores the concept of *quasi-linearity* and its implications for dimensionality reduction.

The proposed framework builds upon previous research in reducing the dimensionality of regression models by extending the concept to arbitrary ansatz functions, including Taylor, Exponential, and Periodic approximations. To achieve this, the framework introduces the concept of *Quasi-Separability*, which encompasses *quasi-linearity*, allowing for a reduction in the dimensionality of the expensive model. The framework utilizes machine learning techniques, inspired by the work of Christodoulou [2020], to approximate the non-reducible part of the model. The framework draws inspiration from the works of Owen [2000] and Chui and Diamond [1987], which also address higher-dimensional regressions. Additionally, the concept of *Quasi-Regressions* introduced by An and Owen [2001] for function approximations on high-dimensional unit cubes influences the framework's approach.

In order to extend the capabilities of the RSQTOA framework, which focuses on producing approximate models for multivariate scalar functions, this research aims to enhance the framework's compatibility with generating approximate models using datasets. The current

framework operates on functions as input, where scalar regressions for each dimension are obtained through domain and dimension sampling. This process involves acquiring target values from the function, performing regressions, and validating the error against predefined criteria. However, when dealing with datasets instead of actual functions, we face the challenge of having limited data points available. To overcome this challenge, two approaches are proposed. Firstly, we suggest representing the dataset in a functional form using interpolators, which can then be utilized as input to the framework. Secondly, we introduce a multidimensional grid-based sampling approach to address the limitations posed by the dataset.

Consequently, a comparative analysis was conducted to evaluate the performance of the RSQTOA framework with interpolation and grid sampling approaches, in comparison to Artificial Neural Networks (ANNs) as baseline models. This evaluation aimed to assess the effectiveness of the proposed framework in various scenarios. Two datasets were utilized for this purpose. Firstly, a synthetic dataset was generated using a simple mathematical function, enabling a controlled environment for evaluating the performance of the RSQTOA-based approaches. Additionally, the Yelp academic dataset was employed, which encompassed diverse features related to review content, reviewer characteristics, and business attributes. The comparative analysis of these datasets allowed for a comprehensive evaluation of the RSQTOA framework, providing valuable insights into its performance, accuracy, and scalability in different contexts. The findings from this analysis contribute to a better understanding of the advantages and limitations of the framework compared to traditional ANN-based models.

1.2 Thesis Structure

We begin by introducing the RSQTOA framework and its fundamental principles. The frameworks overview and methodology are discussed, emphasizing the concept of scalar regression and its role in reducing model dimensionality. The process of model reduction, approximation of remaining dimensions, and subsequent reassembly to construct the final model are outlined in Section 2. This section provides a detailed explanation of the techniques employed.

In addition to dimensionality reduction, sampling approaches are explored to ensure the acquisition of representative samples across the domain. Furthermore, a grid-based sampling technique is proposed to obtain representative samples from the dataset's data points, enabling the framework to produce approximate models directly from the dataset. These aspects are thoroughly examined in Section 2, offering insights into the sampling strategies utilized.

Moving on to implementation, Section 3 delves into the practical aspects of realizing the RSQTOA framework and the associated sampling techniques. The algorithm for regression analysis is explained, along with a comprehensive discussion of the employed sampling methods.

To demonstrate the effectiveness of the RSQTOA framework, two real-world examples are presented in Section 4. These examples showcase the application of the framework and allow for a comparison with other established approaches, thus highlighting the unique advantages offered by our framework.

Finally, the thesis concludes with a thorough analysis of the obtained results and provides an outlook for future research endeavours. This discussion reflects upon the implications of the findings, potential areas of improvement, and the promising prospects for further exploration within the field.

2 Theory

This chapter delves into the mathematical foundations and theoretical principles that underpin the RSQTOA framework. To establish a solid groundwork, we commence with an extensive overview of scalar regression analysis, which serves as the fundamental cornerstone of the framework. Scalar regression analysis is a statistical technique utilized to model the relationship between a dependent variable and one or more independent variables. Gaining an understanding of the principles of scalar regression analysis is crucial for comprehending the subsequent discussions in this chapter.
Additionally, we introduce two pivotal concepts that hold a paramount significance in the RSQTOA framework: the *Separability of dimensions* and the *Reducibility of the model*. The separability of dimensions pertains to the extent to which the dimensions of a higher-dimensional dataset can be independently examined or treated as distinct entities. It enables the identification of dimensions that exhibit unique behaviours or patterns. On the other hand, the reducibility of the model refers to the potential to reduce the dimensionality of a regression model without compromising its predictive accuracy. These concepts are seamlessly integrated with regression analysis to develop a novel methodology for approximating higher-dimensional data. By amalgamating the principles of scalar regression analysis, separability of dimensions, and reducibility of the model, the RSQTOA framework offers a powerful approach to tackle high-dimensional regression problems.
Furthermore, our discussion will focus on how the framework assembles the regression of separable dimensions and the combined approximation of non-reducible dimensions to construct the final approximate model. We will explore different sampling procedures employed by the framework and elucidate their workings, highlighting how they contribute to obtaining representative samples for achieving the best possible approximations. This in-depth exploration of the framework's inner workings will provide valuable insights into its methodology and the underlying principles that drive its effectiveness.

2.1 Regression

Regression analysis involves analyzing a set of data comprised of samples $\boldsymbol{x} \in \mathbb{R}^m$ and their corresponding observations $\boldsymbol{y} \in \mathbb{R}^m$. These observations can be obtained through various means, such as experiments, simulations, or evaluations of an objective function. In our context, we limit our focus to scalar regression models, denoted as $\tilde{f} : \mathbb{R} \to \mathbb{R}$, where $x \mapsto \tilde{f}(x; \boldsymbol{\theta})$, with $\boldsymbol{\theta} \in \mathbb{R}^n$ representing the model's parameters. The application of such models to the available data results in:

$$y_i = \tilde{f}(x_i; \boldsymbol{\theta}) + \epsilon_i \quad i = 1 \dots m \tag{1}$$

ϵ_i represents the error residuals, defined as the discrepancies between the observed data and the corresponding model values. The aim of the regression analysis is to obtain an optimal fit of the model to the provided data by adjusting the parameters $\boldsymbol{\theta}$. The optimization task entails minimizing the sum of squared residuals, whereby the objective function is expressed as a mathematical formulation.

$$\min_{\boldsymbol{\theta}} \|\boldsymbol{\epsilon}\|_2^2 = \min_{\boldsymbol{\theta}} \sum_{i=1}^{n} \epsilon_i^2 = \min_{\boldsymbol{\theta}} \sum_{i=1}^{m} \left(y_i - \tilde{f}(x_i; \boldsymbol{\theta}) \right)^2 . \tag{2}$$

The framework uses the Euclidian norm, as it is one of the most preferred choices due to its simplicity. The Euclidean norm, while not necessary, is frequently employed due to its inherent simplicity, thus rendering it one of the most favoured options.

The primary distinction among regression models lies in their linearity or non-linearity with respect to the parameters $\boldsymbol{\theta}$. Linear models have a straightforward closed-form solution in the form of a linear system of equations, while non-linear models require iterative optimization techniques. Within the implemented framework, three regression models are available: a linear Taylor model capable of fitting polynomials of arbitrary order, a non-linear exponential model, and a non-linear periodic model. However, for the purpose of this study, our focus will be on the Taylor model, as it offers greater flexibility and power compared to the other two models. The Taylor model allows us to represent complex systems of equations without the need to know the exact functional form in advance.

2.1.1 Taylor Model

The Taylor regression model of order n is a mathematical function $\tilde{f} : \mathbb{R} \to \mathbb{R}$ can defined as:

$$\tilde{f}(x) = \sum_{k=0}^{n} \theta_k * x^k \tag{3}$$

where θ_k represents the coefficients and x^k represents the power of x up to order n. The goal of the regression analysis is to determine the optimal values for the coefficients θ_k that minimize the error between the model predictions and the observed data. This is achieved by solving the following linear system of equations:

$$\boldsymbol{X\theta} = \mathbf{y} \quad \Longleftrightarrow \quad \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & x_m^2 & \cdots & x_m^n \end{bmatrix} * \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_m \end{bmatrix} \tag{4}$$

where $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ is the design matrix, which contains the powers of the input variable x up to order n evaluated at the given samples, $\boldsymbol{\theta} \in \mathbb{R}^n$ is the vector of coefficients, and $\mathbf{y} \in \mathbb{R}^m$ is the vector of observed values.
The design matrix $\boldsymbol{X}$ is constructed by evaluating the powers of x for each sample, resulting in a matrix with m rows (corresponding to the number of samples) and n columns (corresponding to the order of the Taylor model). The vector $\mathbf{y}$ contains the observed values at each sample point.
By solving the linear system of equations, the framework determines the optimal coefficients $\boldsymbol{\theta}$ that best fit the observed data and represent the Taylor model of order n.
This approach allows us to approximate the relationship between the input variable x and the output variable y using a polynomial function. The higher the order n, the more complex the polynomial function can be, allowing for a more flexible model that can capture intricate patterns in the data.
Overall, the Taylor model provides a flexible and powerful regression technique for approximating the relationship between variables based on polynomial functions. It has been widely used in various fields, including physics, engineering, and economics, to model and analyze complex systems.

2.2 Separability

To address the challenges posed by the *Curse of Dimensionality*, it is essential to employ techniques that reduce the number of dimensions. In this context, the concepts of *Separability* and *Quasi-Separability* play a crucial role in dimensionality reduction.

Separability refers to the degree to which dimensions can be considered independent or distinguishable from one another. In the context of dimensionality reduction, identifying dimensions that exhibit high separability allows for their exclusion or reduction, leading to a more compact and manageable representation.
Quasi-Separability, on the other hand, extends the notion of *Separability* by considering dimensions that may not be entirely independent but exhibit certain patterns or relationships that can be exploited for dimensionality reduction. These dimensions may not be completely separable, but they possess characteristics that make them quasi-independent and amenable to reduction.
By leveraging *Separability* and *Quasi-Separability* as tools for dimensionality reduction, we can significantly alleviate the challenges associated with high dimensions. By identifying and reducing dimensions that exhibit *separability* or *quasi-separability*, we can create more concise and meaningful representations.

Definition 1 (Partial Separability) *The term* ***partially separable*** *refers to a property exhibited by a function $f : \mathbb{R}^n \to \mathbb{R}$, where $\boldsymbol{x} \mapsto f(\boldsymbol{x})$. Specifically, a function is said to be partially separable if it can be expressed as:*

$$f(\boldsymbol{x}) = \tilde{f}_0\left((x_i)_{i\in\mathcal{I}_0}\right) + \tilde{f}_1\left((x_i)_{i\in\mathcal{I}_1}\right) \tag{5}$$

where $\tilde{f}_0 : \mathbb{R}^{|\mathcal{I}_0|} \to \mathbb{R}$ and $\tilde{f}_1 : \mathbb{R}^{|\mathcal{I}_1|} \to \mathbb{R}$ are functions defined on two disjoint subsets of indices of input variables $\mathcal{I}_0$ and $\mathcal{I}_1$, respectively.

This notation is taken from (Deussen [2021]) and is commonly used in the field of mathematical optimization and machine learning. By decomposing the function $f(\boldsymbol{x})$ into two distinct components, each dependent only on a subset of the input variables, it is possible to analyze and optimize the behaviour of each component separately.
It should be noted that partial separability is a desirable property in many applications, as it can lead to more efficient optimization algorithms and reduced computational complexity. Therefore, the study of partial separability is of great importance in the fields of optimization and machine learning, among others.

In order to establish a clear framework for our analysis, it is necessary to establish certain parameters with regard to the separations being considered. Specifically, we will limit our focus to separations in which at least one of the two index subsets contains only a single element. It is with this condition in mind that we put forth our definitions of *Dimensional Separability* and *Full Separability*.

Definition 2 (Dimensional Separability) *Consider a function $f : \mathbb{R}^n \to \mathbb{R}$ defined by $\boldsymbol{x} \mapsto f(\boldsymbol{x})$, where $\boldsymbol{x}$ denotes the n-dimensional input vector. In this context, we define a dimension as separable if and only if the function f is additive with respect to the corresponding input variable. More precisely, we say that dimension i is separable if and only if there exist two functions, $\tilde{f} : \mathbb{R}^{n-1} \to \mathbb{R}$ and $\phi : \mathbb{R} \to \mathbb{R}$, such that:*

$$\textit{dimension } i \textit{ is } \textbf{\textit{separable}} \iff f(\boldsymbol{x}) = \tilde{f}(x_1, \ldots, x_{i-1}, x_{i+1}, x_n) + \phi(x_i) \tag{6}$$

with $\tilde{f} : \mathbb{R}^{n-1} \to \mathbb{R}$ and $\phi : \mathbb{R} \to \mathbb{R}$. In this definition, $\tilde{f}$ represents the contribution of all dimensions except i to the value of the function, while ϕ captures the effect of dimension i.

The notion of dimensional separability is a powerful tool for analyzing multivariate functions, as it allows for the decomposition of a complex function into simpler, separable functions. Furthermore, this concept is of particular relevance in the context of machine learning, where

it is often desirable to identify separable dimensions in order to simplify the model and reduce computational complexity.
In conclusion, the concept of dimensional separability provides a useful framework for analyzing and decomposing multivariate functions. Its applications are diverse, ranging from mathematical analysis to machine learning, and it represents a valuable tool for understanding and simplifying complex functions.

Definition 3 (Full Separability) *The term **fully separable** refers to a function $f : \mathbb{R}^n \to \mathbb{R}$ defined by $\boldsymbol{x} \mapsto f(\boldsymbol{x})$, where all dimensions of the function are separable. This condition implies that the function can be expressed as a sum of functions $\phi_i : \mathbb{R} \to \mathbb{R}$ for all dimensions $i = 1, \ldots, n$, along with a constant $C \in \mathbb{R}$. Specifically, the function can be represented as follows:*

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \phi_i(x_i) + C \tag{7}$$

It is important to note that full separability is a strong condition, as it requires that each dimension of the function is separable. This property is often useful in mathematical modelling and optimization problems, as it simplifies the analysis and solution of the function.

The concept of *Separability* has been widely employed to simplify the problem of dimensionality. However, it is important to note that this property is seldom applicable to real-world models due to the presence of dependencies between dimensions. This renders the *Separability* assumption invalid in most cases. As a solution to this problem, a weaker definition of *Separability* can be introduced, known as *Quasi-Separability*. This approach introduces a trade-off between accuracy and separability, acknowledging that exact separability is often unattainable in practice. By utilizing this approach, we can balance the need for accuracy and efficiency in our modelling process, resulting in more realistic and applicable models.

Definition 4 (Quasi-Separability) *The term quasi-separability is used in mathematics to refer to the degree to which a function can be separated or decomposed into lower-dimensional functions. Specifically, given a function $f : \mathbb{R}^n \to \mathbb{R}$ defined by $\boldsymbol{x} \mapsto f(\boldsymbol{x})$ and an error tolerance $\epsilon \geq 0$, a dimension i of the function is said to be **quasi-separable** if and only if the error introduced by separating this dimension from the others is less than or equal to ϵ.*
*More formally, dimension i is **quasi-separable** if and only if the following inequality holds:*

$$\left| f(\boldsymbol{x}) - \left(\tilde{f}(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) + \phi(x_i) \right) \right| \leq \epsilon \tag{8}$$

Here, $\tilde{f} : \mathbb{R}^{n-1} \to \mathbb{R}$ is a function that represents the approximation of f obtained by separating out dimension i, and $\phi : \mathbb{R} \to \mathbb{R}$ is a function that captures the contribution of dimension i to the overall function.

In other words, *quasi-separability* allows us to approximate a high-dimensional function by breaking it down into lower-dimensional functions, and to do so with a controlled degree of error. According to the given definition, it can be inferred that *Separability* represents a distinctive instance of *Quasi-Separability*, in which the value of ϵ is precisely zero.
The proposed ansatz function ϕ serves as an unconstrained scalar regression. However, in order to facilitate our analysis, we shall employ a Taylor regression model and subsequently introduce relevant notation.

Corollary 1 (Quasi-Taylor) *Considering the function $f : \mathbb{R}^n \to \mathbb{R}$ defined by $\boldsymbol{x} \mapsto f(\boldsymbol{x})$. We call a dimension **Quasi-Taylor of order n** when it is quasi-separable and can be expressed in the form of a polynomial function:*

$$\phi(x) = \sum_{k=0}^{n} a_k * x^k \quad \textit{with} \quad a_k \in \mathbb{R} \quad \forall k = 0 \dots n. \tag{9}$$

Here, a_k is a constant coefficient, and the sum ranges from $k = 0$ to $k = n$. It is noteworthy that quasi-separability is a necessary condition for a dimension to be a quasi-Taylor of order n.

2.3 Model Reduction

The primary aim of the framework is to partition a prescribed function into two distinct components, wherein one part can be approximated using a given set of scalar regression models, and the other part is inherently unattainable through this means. The notion of *Quasi-Separability* serves as a useful framework to establish the conditions under which function reduction is feasible.

Definition 5 (Quasi-Reducibility) *Considering the function $f : \mathcal{X} \to \mathcal{Y} \subset \mathbb{R}$ defined by $\boldsymbol{x} \mapsto f(\boldsymbol{x})$ that maps elements from a finite domain $\mathcal{X} \subset \mathbb{R}^n$ to a subset $\mathcal{Y} \subset \mathbb{R}$. The function f is said to be **quasi-reducible** within the domain $\mathcal{X}$ if and only if there exists at least one dimension that can be regarded as quasi-separable with an accuracy level of ϵ. In other words, there must be at least one dimension in $\mathcal{X}$ that can be effectively partitioned into two subspaces, with the difference between the images of points in these subspaces being no greater than ϵ.*

Detection of *Quasi-Separability* In order to identify quasi-separable dimensions the framework employs a randomized approach. Let $f : \mathcal{X} \subset \mathbb{R}^n$ represent the domain of the function being analyzed, and let ϵ denote the desired error threshold. To detect *quasi-separability*, a set of m random samples is generated across the domain. For each sample, all dimensions except the one of interest denoted as i, are kept fixed and more samples are obtained by varying the value for the dimension being approximated i.e. i. These samples, referred to as dimension samples within the framework, are then used to fit a scalar function using an appropriate regression model introduced in Section 2.1. This process results in m different scalar sub-functions.
Subsequently, next it averages the coefficients of the m regressions and produces a final approximation for the separated dimension i, represented as $\phi_i(x_i)$. By employing this approximation, the concept of *Quasi-Separability* can be applied to construct the reduced model $\tilde{f} : \mathbb{R}^{n-1} \to \mathbb{R}$, which is defined as

$$\tilde{f}(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = f\left(\boldsymbol{x}|_{x_i = \hat{x}_i}\right) - \phi_i(\hat{x}_i) \tag{10}$$

where $\hat{x}_i \in \mathcal{X}|_i$ denotes a random point from the domain of dimension i. By assessing the *Quasi-Separability* of dimension i based on the resulting reduced model and the specified error threshold, the suitability of the approximation can be determined.

Validation Upon substituting the reduced model into the *Quasi-Separability* condition, we obtain the following test equation:

$$|f(\boldsymbol{x}) - (f\left(\boldsymbol{x}|_{x_i = \hat{x}_i}\right) - \phi_i(\hat{x}_i) + \phi_i(x_i))| \overset{!}{\leq} \epsilon \tag{11}$$

This equation is used to validate the reduced model and determine its validity. To accomplish this, a set of test pairs $(\boldsymbol{x}, \hat{x}_i) \in \mathcal{X} \times \mathcal{X}|_i$ is sampled, where $\mathcal{X}$ represents the input space and $\mathcal{X}|_i$ represents the space with dimension x_i fixed. The test equation is then evaluated for each test sample. If the test equation is satisfied for all test samples, it indicates that the dimension is separable and validates the reduced model.

2.4 Training and Model Reassembling

After reducing the model, it is necessary to approximate the non-separable part of the function through various means. The framework opts to use neural networks for approximating the non-reducible part.

To establish two distinct subsets of indices for non-separable and separable dimensions, we consider $\mathcal{I}_0$ and $\mathcal{I}_1$, respectively. Consequently, the reduced model is defined as $\tilde{f} : \mathbb{R}^{|\mathcal{I}0|} \to \mathbb{R}$, and the trained approximation is represented as $\tilde{f}_{trained} : \mathbb{R}^{|\mathcal{I}_0|} \to \mathbb{R}$. According to the definition of *Quasi-Separability*, we can reassemble the model by including all regressions to the trained approximation, resulting in the final approximation $f_{trained} : \mathcal{X} \subset \mathbb{R}^n \to \mathbb{R}$ as follows:

$$f_{trained}(\boldsymbol{x}) = \tilde{f}_{trained}\left((x_i)_{i \in \mathcal{I}_0}\right) + \sum_{i \in \mathcal{I}_1} \phi_i\left(x_i\right) \tag{12}$$

Here, ϕ_i represents the regression function for the ith separable dimension of the function, and $\boldsymbol{x}$ denotes the input vector with n dimensions.

2.5 Sampling

Sampling in high-dimensional domains is a challenging task due to the Curse of Dimensionality Bellman [1961], which makes it impractical to explore the entire high-dimensional space and obtain a representative sample. To overcome this challenge, various approaches have been proposed, including specialized sampling algorithms designed for high-dimensional spaces.

In our framework, we construct an approximate model using two types of samples: domain samples and dimension samples. To approximate a specific dimension of interest, denoted as i, we first obtain m domain samples that cover the entire domain. For each domain sample, we acquire n dimension samples by fixing all dimensions except the one of interest, i.

To make our framework compatible with producing approximate models from datasets, we propose two approaches. The first approach involves obtaining a functional approximation of the dataset and using it as an input function for the framework. This can be achieved through interpolation-based techniques, such as nearest neighbour, which allow us to leverage pre-implemented sampling strategies like random sampling and Latin hypercube sampling.

Alternatively, we can utilize a sampling strategy that operates directly on the dataset and generates the required samples for our framework. To address this, we introduce a novel grid-based sampling technique. This technique partitions the high-dimensional space into a grid structure, enabling the generation of representative samples that capture the underlying distribution while reducing the computational complexity associated with high-dimensional sampling.

By incorporating these approaches, we enhance the capabilities of our framework to handle datasets and facilitate the construction of accurate approximate models from high-dimensional data.

2.5.1 Random sampling

The use of pure random sampling is a common approach in sampling theory due to its simplicity and ease of implementation. However, when working with high-dimensional spaces, this technique can result in a sample distribution that is not representative of the population distribution. Therefore, this method provides no guarantees regarding the final sample distribution. As an example, consider the validation of the *Quasi-Separability* concept (Section 2.3), which requires a large number of samples to produce reliable results [Zhang et al., 2018].

Despite this limitation, random sampling can be advantageous in scenarios where new samples can be easily added without the need to re-sample the entire population. Gao et al. [2020] demonstrated this property in their study, where they utilized random sampling to generate a small dataset for training a deep neural network.

Figure 1 shows an example of random sampling with 9 samples. Although the sample distribution may not accurately reflect the population distribution, this method can still provide valuable insights into the underlying data structure.

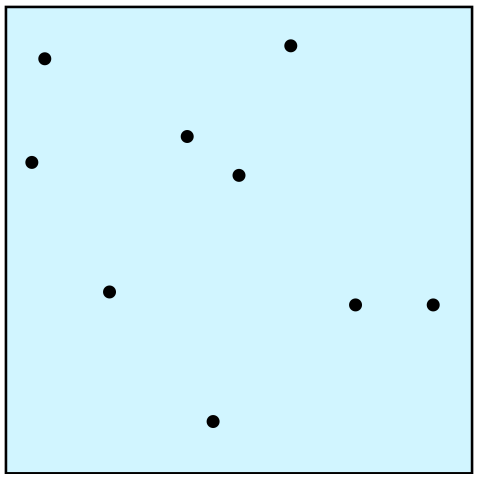

Figure 1: Random sampling ($N = 9$)

2.5.2 Full-factorial sampling

Full-factorial sampling is a common sampling method in experimental design that involves sampling every possible combination of values across all dimensions. This method ensures a very uniform distribution of samples across the domain, which is desirable in many applications. However, as the number of dimensions and subspaces increases, the number of samples required for full-factorial sampling grows exponentially, making it infeasible for high-dimensional domains.

More formally, given a domain $\mathcal{X} \subset \mathbb{R}^n$, full-factorial sampling cuts each dimension into N subspaces resulting in N^n subdomains. In each subdomain, a single random point is sampled, resulting in N^n samples that cover the entire domain. However, the number of samples required for full-factorial sampling grows exponentially with the number of dimensions, as N^n. Therefore, this method is limited to sample sets of size $k * N^n$, where k is a constant that depends on the available computational resources and desired level of coverage.

Figure 2 provides an example of full-factorial sampling in a two-dimensional domain with $N = 3^2 = 9$. As shown, full-factorial sampling covers the entire domain with a very uniform distribution of samples.

While full-factorial sampling is an ideal method for ensuring a uniform distribution of samples across a domain, it becomes infeasible for high-dimensional domains due to the exponential growth in the number of required samples. Therefore, researchers often resort to alternative sampling methods such as Latin hypercube sampling (LHS) and Monte Carlo Sampling, which have been developed to address this issue and provide more efficient sampling techniques.

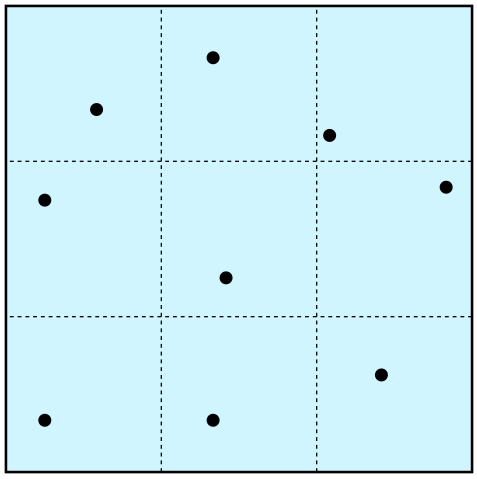

Figure 2: Full-factorial sampling ($N = 3^2 = 9$)

2.5.3 Latin-Hypercube sampling

It is a widely used method for generating samples in high-dimensional parameter space. It is a compromise between random sampling and full-factorial sampling, providing a good representation of the real variability while being computationally efficient. In LHS, each dimension of the parameter space is divided into N subspaces, resulting in N^n subdomains. However, only N of these subdomains are chosen, and they are arranged in a Latin-Hypercube, as shown in Figure 3. In a two-dimensional case, each grid row and column can only contain one sample. LHS has been applied in various fields such as computer experiments Stein [1987] and sensitivity analysis Sobol [2001]. One advantage of LHS is that it provides better coverage of the parameter space than random sampling. Furthermore, the maximal distance between each pair of samples is maximized, leading to a better representation of the variability in the parameter space Stein [1987].
LHS is a popular and efficient method for generating samples in high-dimensional parameter space. It provides good coverage of the parameter space while being computationally efficient, making it a suitable method for various applications in science and engineering. We use *Latin-Hypercube* sampling as the default method in our framework.

2.5.4 Grid sampling

The proposed approach for generating a grid based sampling involves creating a multi dimensional grid. For instance, consider a dataset with d features, denoted by $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$, and one target variable, denoted by z. To create the grid, the d features are represented as d orthogonal axes and form a cube, or a hypercube if the dataset has more than three features. The maximum and minimum values of each feature are used to define the sides of the cube. Subsequently, each side of the cube is divided equally into s parts to form a grid-like structure. In the present example, let us consider a dataset with three features, denoted by x_1, x_2, and x_3, the maximum and minimum values of the features to be 0 and 1, respectively, and s to be

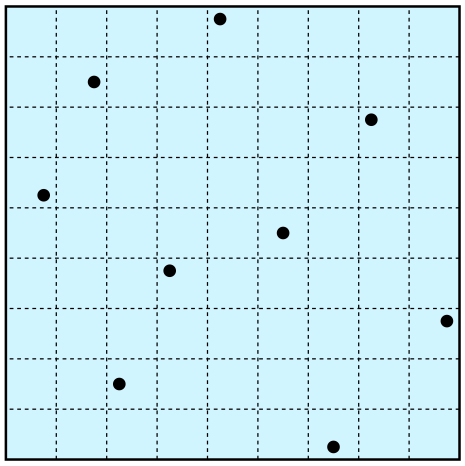

Figure 3: Latin-Hypercube sampling ($N = 9$)

4. Figure 4 illustrates this process. The resulting grid contains s^d sections d is the number of dimensions of the dataset i.e. 3.

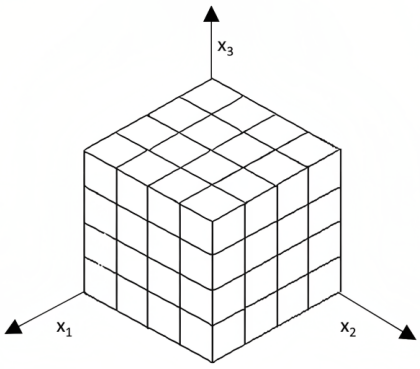

Figure 4: Multi-Dimensional Grid (s^d sections)

In order to derive the regression model for feature x_3, it adopts a technique that involves projecting the data points onto a reduced surface formed by all the features, except for x_3. Specifically, the reduced surface in our example is a two-dimensional plane with x_1 and x_2 as the axes, as depicted in Figure 5. The data points on this surface correspond to the original data points projected onto the reduced surface, as illustrated in Figure 6.
It is notable that certain regions of the reduced surface contain a greater number of data points than others. Accordingly, we apply a filtering criterion to exclude regions with fewer data points than a pre-determined threshold, t. Here, we set t to 3 for the purpose of exposition. The result of this filtering procedure is highlighted in Figure 7, where shaded sections indicate the regions that meet the filtering criterion.
The points lying inside the filtered grid sections become our candidates for domain sampling. Let us assume that we sample four points in our given example. Using random sampling from our eligible candidates, we obtain four points represented by red marks in Figure 8. Using each sample, we would obtain dimension samples and train a scalar sub-function.
To obtain dimension samples, it first fixes the values of x_1 and x_2 by selecting them as the midpoint of the section surface, as illustrated in Figure 9. This approach enables us to obtain a representative set of data points within a given section of the sample space.

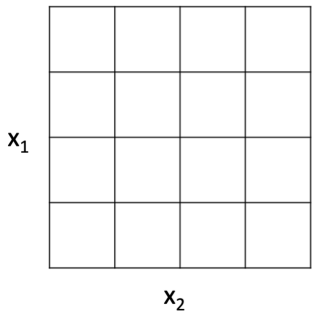

Figure 5: Plane.

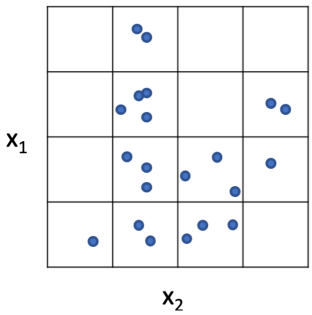

Figure 6: Projection

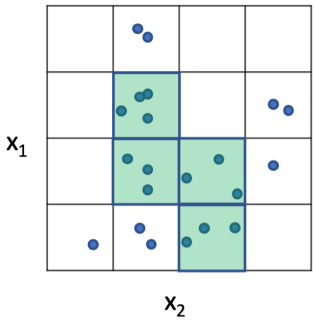

Figure 7: Filtering

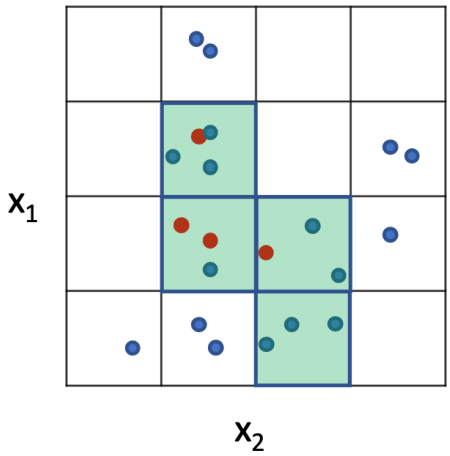

Figure 8: Sampled points

Given the fixed values of x_1 and x_2, it then varies the value of x_3 based on the available data points, allowing us to sample data points across the section. Subsequently, we fit a scalar regression model to predict our target variable.
This sampling approach allows us to obtain a representative set of data points within a section of the sample space, while the scalar regression model enables us to make predictions for the target variable with reasonable accuracy. Overall, this methodology enables us to efficiently explore and analyze the high-dimensional data space under consideration.
Consider a sub-surface or section, where features and target are related as given by Equation 13. In this sub-section, x_1 and x_2 are fixed, and only x_3 varies, represented by Equation 14. Assuming a linear relationship between x_3 and its contribution to z i.e. z', we obtain the regression equation as in Equation 15 by fitting a regression model with given values of x_3 and z' (Figure 10). This procedure is repeated for all eligible sections to obtain their respective regression equations. The regression equations are then combined by averaging the coefficients to obtain an aggregated equation. After validating the aggregated equation, we accept or reject it based on a threshold error. If the error is less than the threshold error, we store the equation; otherwise, we mark the dimension as non-separable. This iterative operation is performed for each dimension.

$$z = f(x_1, x_2, x_3) \tag{13}$$

$$z' = f'(x_3) \tag{14}$$

$$z' = a_1 x_3 + c \tag{15}$$

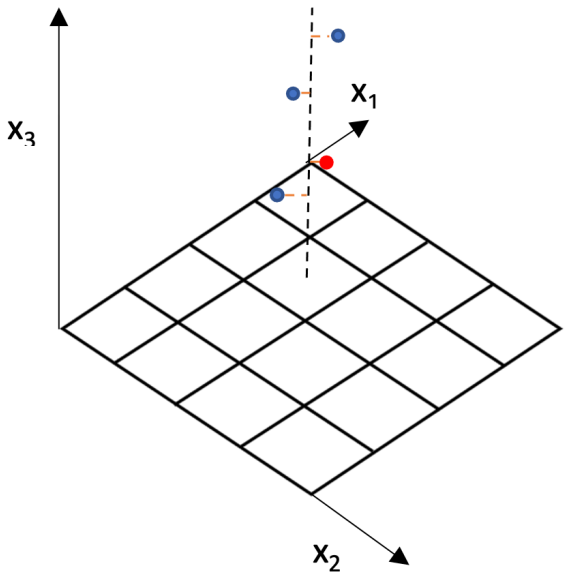

Figure 9: Illustration of the position of data points within a section with respect to an imaginary line (anchor) parallel to the x_3 axis.

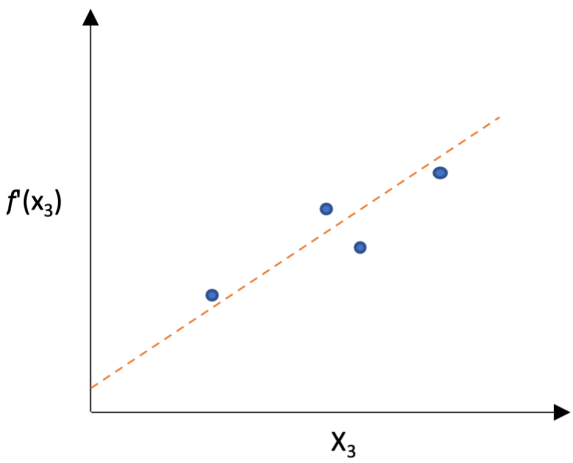

Figure 10: Regression Analysis of Sub-Surface Relationship for x_3

3 Implementation

This chapter concerns itself with the actual implementation of the framework. It covers the ideas and applications of the *Separability* concepts introduced in Section 2 as well as some technical details of the implementation. We shall commence by focusing on the framework's structure, the constituent objects within the system, and their respective behaviours. The UML Class diagram of the RSQTOA framework is illustrated in Figure 11.
The framework comprises three primary components: the sampling, the *Quasi-Separability* detection, and the subdomain splitting. The framework adopts an automatic approach that partitions the domain into smaller subdomains and identifies subspaces that exhibit *Quasi-Separability*. To accomplish this task, a tree-based structure is utilized to traverse the domain. This structure is instrumental in storing the subspace regression, allowing for easy generation of reduced models at each stage of the domain decomposition, as well as facilitating their reassembly at a later stage. The `SubspaceTree` class serves as the root, containing only one `SubspaceNode` representing the entire domain. All other nodes are stored recursively within each other, thus representing the recursive decomposition of the domain.

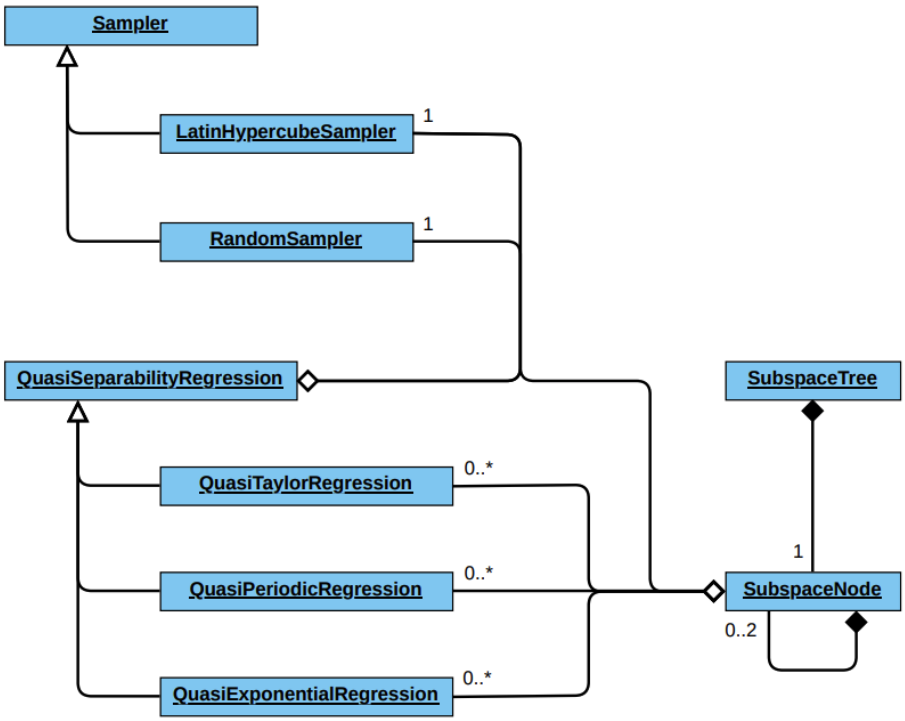

Figure 11: UML Class diagram of the *RSQTOA* framework Märtens [2022]

While our study aims to fit a model over the complete domain, it is worth noting that the framework is capable of constructing approximate models using subdomains. This capability is achieved by leveraging the automatic subdomain generation feature, which searches for subspaces that exhibit *Quasi-Separability*, and fitting a model to each subdomain.
To fit or represent a dimension using an approximate model, the framework relies on the *Quasi-Separability* regressor. Consequently, the framework depends on sampling to obtain a representative sample that can help the regressor identify the underlying structure and relationships between the feature and target variables. Each component plays a crucial role, and we will expand on their inner workings in the following sections. However, to make our framework capable of producing approximate models from the dataset, we made some additional adjustments. Figure 12 describes the updated structure of our framework.

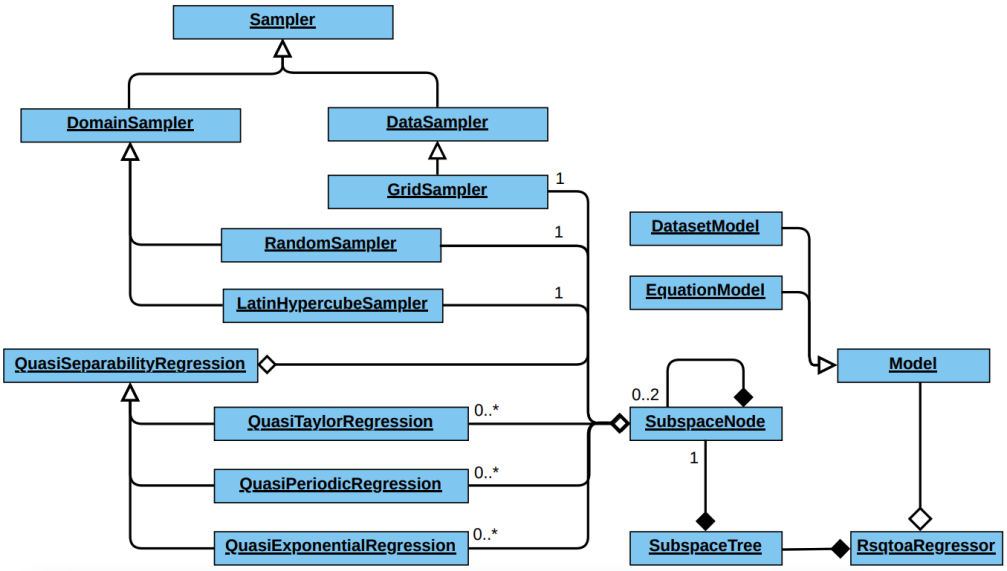

Figure 12: Updated UML Class diagram of the *RSQTOA* framework.

Upon comparison, two primary distinctions become apparent in our proposed framework: the new sampling module and an additional model wrapper layer. Since our framework is designed to support the production of approximate models from both equations and datasets, the model wrapper layer plays a critical role in initializing the framework instance to be compatible with the type of input.

The type of model instance that is initialized has a direct impact on the sampling method that can be employed to fit a regression. For instance, when approximating an equation, we can leverage the fact that any data point in the complete domain of each dimension can be sampled. Similarly, in the case of a dataset, we can represent the dataset as a function by employing various interpolation techniques, which enables us to approximate the dataset and provides an interface for our framework to sample data points across the domain of dimensions. In addition to the equation model, we can also directly sample from the dataset and produce an approximate model. To achieve this, an instance of the dataset model can be instantiated, which in turn initializes an instance of our proposed grid sampler capable of sampling in higher dimensions directly from the dataset. The grid sampler also serves as an adapter to the framework, enabling us to train an approximate model directly from the dataset.

After discussing the structure of our framework and understanding the interactions between the different components, let us delve deeper into each of the components and examine how we translate theory into a practical realization. For the purposes of our discussion, we will focus on four key components: models, sampler, regressor, and model reduction.

3.1 Model

The `Model` class serves as a crucial interface between the target relationship to be approximated, whether in the form of an equation or a dataset. `EquationModel` and `DatasetModel` implement the `Model` class for the respective use cases outlined in our previous discussion. Furthermore, the specific instance of the `Model` class serves as a key differentiator regarding the sampling algorithm to be utilized, as presented in Section 2.5.

As depicted in Figure 13, the base `Model` class provides the essential structure required for seamless compatibility within our framework, while the various subclasses implement specific procedures accordingly. For example, the `get_target` procedure within the `EquationModel`

subclass retrieves the target value by evaluating the underlying function that was passed in the constructor as the variable *equation*. In contrast, for our `DatasetModel` subclass, the `get_target` procedure searches for the matching row corresponding to the feature values passed and returns the corresponding target variable value. Please refer to Code 6, Code 7 and Code 8 for the source code of `Model`, `EquationModel` and `DatasetModel` classes respectively.

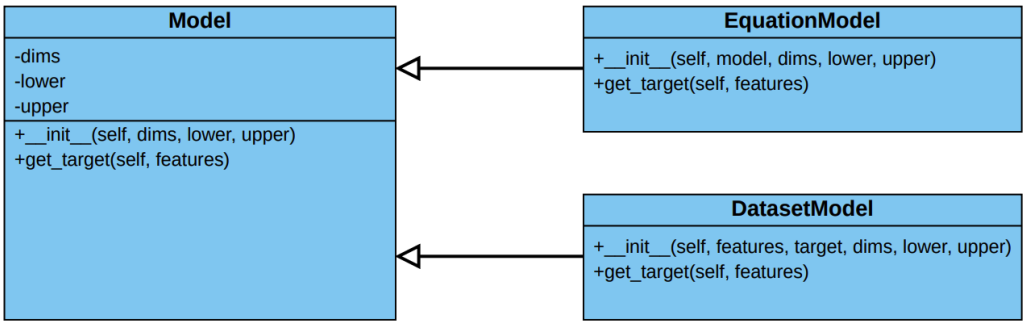

Figure 13: UML Class diagram for the *Model* class and its subclasses

3.2 Sampling

Sampling represents a critical component within our framework. Figure 14 illustrates the structure of the sampling module, highlighting the various methods available to us. The `Sampler` interface guarantees that any implemented sampling algorithm remains compatible with our framework. Our use case is characterized by two conditions - one in which data may be sampled across the entire domain, and the other in which samples can only belong to the available dataset.

To account for these distinctions, the framework bifurcates the sampling module into two distinct sections - one dedicated to domain sampling and the other for dataset sampling, as per the sampling procedures discussed in Section 2.5. As a consequence, we implement two interfaces, namely, `DomainSampler` and `DataSampler`, to facilitate the implementation of the relevant sampling techniques through the appropriate interface derivation. Please refer to Code 9, Code 10 and Code 13 for the source code of `Sampler`, `DomainSampler` and `DataSampler` classes respectively.

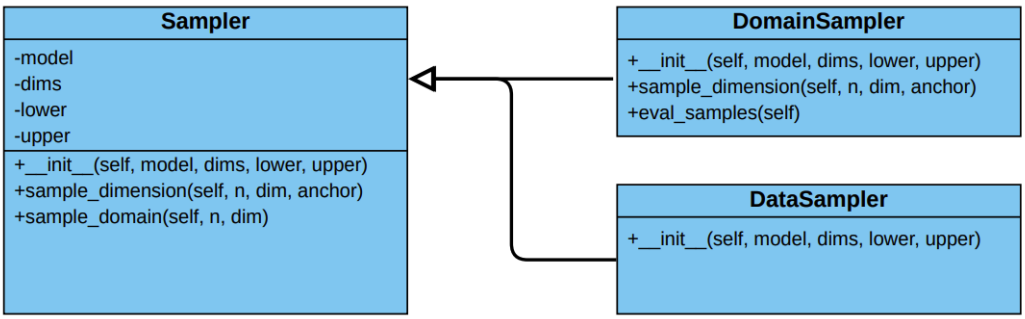

Figure 14: UML Class diagram for the *Sampler* class and its subclasses

3.2.1 Domain Sampling

The framework enables domain sampling by implementing concrete sampling techniques, such as Random sampling and Latin-Hypercube sampling, as discussed in Section 2.5.1 and Section 2.5.3. Figure 15 depicts the UML class diagram of the sampling module, illustrating that the base class provides subspace sampling capabilities and an approach to evaluate the generated samples. The subclasses then implement the high-dimensional sampling techniques. To implement Latin-Hypercube sampling, the framework utilizes the *pyDOE* (Huan [2021]) module as a backend. Please refer to Code 10, Code 11 and Code 12 for the source code of `DomainSampler`, `RandomSampler` and `LatinHypercubeSampler` classes respectively.

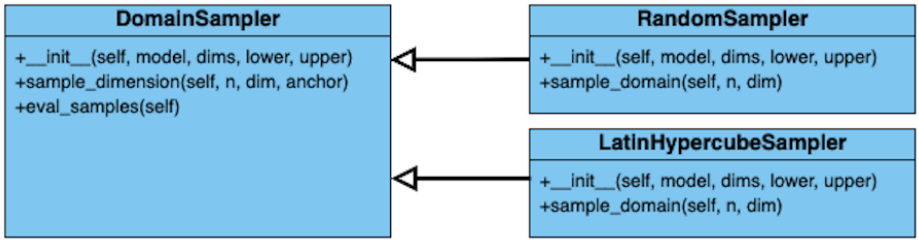

Figure 15: UML Class diagram for the *DomainSampler* class and its subclasses

3.2.2 Data sampling

Our proposed framework also provides support for obtaining samples from the dataset, as discussed in the Sampling Theory section. Figure 16 illustrates the structure of the dataset sampling module, highlighting the availability of necessary sampling procedures, as well as a mechanism for evaluating the samples.

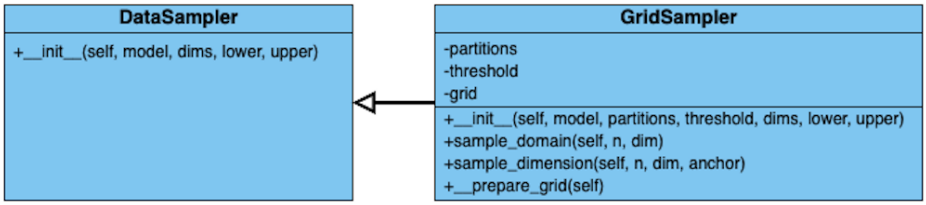

Figure 16: UML Class diagram for the *DataSampler* class and its subclass

The framework employs a grid-based sampling approach, as discussed in Section 2.5.4, to obtain samples from the dataset. Please refer to Code 13 and Code 14 for the source code of `DataSampler` and `GridSampler` classes respectively.

Grid sampling The grid sampling approach, as discussed in Section 2.5.4, involves obtaining samples from a multi-dimensional grid while approximating a dimension. The approach samples from the multi-dimensional grid are formed by all dimensions except the one being approximated. Since the sampling process depends on the grids, the module creates a grid for each dimension based on the conditions discussed in Section 2.5.4, such as the number of partitions of a dimension and the minimum data points threshold required to obtain grid sections and data points available for obtaining domain and dimension samples.
When instantiating the object for the sampler, the framework precomputes and stores the grids and necessary information in a dictionary for later use. For each dimension, it stores a

`mapping` object that stores the bounds of the grid section and the index for the corresponding data points that fall inside the bounds. Furthermore, to speed up the sampling process, it also stores the indices of all the valid data points in the dictionary as `valid_elements`.
The implementation for obtaining domain samples from our dataset is presented in Code 1. It takes input as the number of samples to be returned and the dimension for which samples are to be obtained. If the dimension is not set, it samples random data points and returns them as samples. However, when the dimension variable is set, it leverages the precomputed indices that are stored in the `valid_elements` property for the dimension in the dictionary and samples indices present in the array to return the data point at the corresponding indices as the samples.

```python
def sample_domain(self, n: int, dim: int = -1) -> tuple[np.ndarray, np.array]:
    """
    Sample the domain by selecting a subset of points and corresponding target values.

    Args:
        n (int): Number of samples to be selected.
        dim (int): Dimension of interest (-1 for all dimensions). Default is -1.

    Returns:
        tuple[np.ndarray, np.array]: Tuple containing the selected domain samples and
                                                        their corresponding target values.
    """
    # Select a subset of indices for domain samples
    domain_samples = random.sample(range(len(self.features)), n)

    # If a specific dimension is specified
    if dim != -1:
        # Get the valid element indices for the given dimension
        valid_element_indices = self.grid[dim]['valid_elements']

        # Select samples only from the valid element indices
        domain_samples = random.sample(valid_element_indices, n)

    # Return the selected domain samples and their corresponding target values
    return self.features[domain_samples], self.target[domain_samples]
```

Code Listing 1: Domain Sampling using Grid Sampler

The implementation for obtaining dimension samples is presented in Code 2. We first obtain the index of the anchor and use the index to identify the grid section where the anchor point lies. On obtaining the grid section, we sample the number of sample data points needed and return the sampled data points by replacing the values of the dimension beside the dimension to be estimated by the corresponding midpoints of the grid section where the anchor data point belonged.

```python
def sample_dimension(self, n: int, dim: int, anchor: List[float]) -> Tuple[np.ndarray,
                                                np.array]:
    """
    Samples a specific dimension of the feature space.

    Args:
        n: The number of samples to generate.
        dim: The index of the dimension to sample.
        anchor: The anchor point for the dimension sampling.

    Returns:
        A tuple containing the feature samples and their corresponding target values.

    """
    # Find the index of the anchor point in the features array
    anchor_index = Counter(np.where(self.model.features == anchor)[0]).most_common(1)[
                                                    0][0]

    sample_indices = []
```

```
    sample_grid_section_anchor = []

    # Iterate through the grid sections and find the one containing the anchor point
    for key, indices in self.grid[dim]['mapping'].items():
        if anchor_index in set(indices):
            # Calculate the sample grid section anchor
            sample_grid_section_anchor = [np.mean([key[i], key[i + (int(len(key) / 2))
                                                    ]]) for i in
                                          range(int(len(key) / 2))]
            # Randomly sample indices from the grid section
            sample_indices = random.sample(indices, n)
            break

    # Get data points based on the sampled indices
    feature_samples = self.model.features[sample_indices]

    # Set the anchor point for the sampled dimension by keeping the values for other
                                                dimensions constant
    feature_samples[:, [i for i in range(self.dims) if i != dim]] =
                                                sample_grid_section_anchor

    return feature_samples, self.model.target[sample_indices]
```

Code Listing 2: Dimension Sampling using Grid Sampler

3.3 Model Reduction via Separability Analysis

The detection of *Quasi-Separability* is a fundamental aspect of our framework, as it allows for the identification of separable dimensions. An overview of the regression classes can be seen in Figure 17, where the base class `QuasiSeparabilityRegression` implements the detection strategy introduced in Section 2.3. The actual regression models, namely the Taylor model, the exponential model, and the periodic model, are implemented in the subclasses. The source code for the implementations of `QuasiSeparabilityRegression` and `QuasiTaylorRegression` can be found in Code 15 and Code 16 respectively.

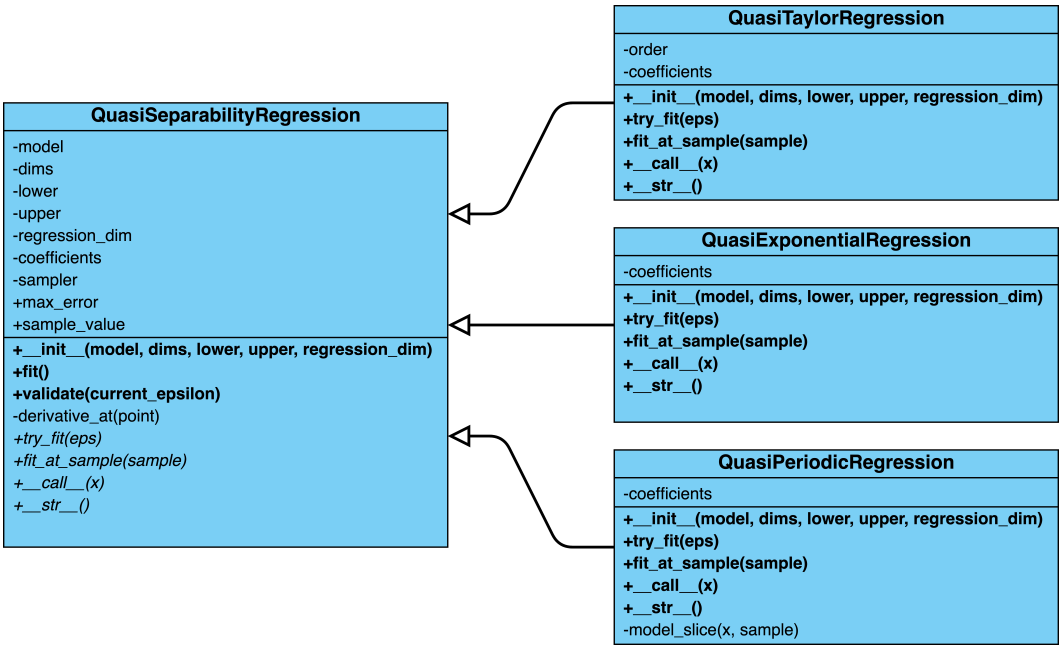

Figure 17: UML diagram of the *Quasi-Separability* regression analysis Märtens [2022]

Detection Algorithm The implementation of the *Quasi-Separability* detection algorithm is presented in Code 3. This algorithm aims to identify the *quasi-separability* of dimensions in a given dataset. The algorithm follows the following steps:

- **Sampling**: A sampling of the current domain is performed to obtain anchor points for scalar regressions (line 5). These anchor points serve as reference points in the domain for the subsequent regression analysis.
- **Regression Analysis**: For each anchor point, a scalar regression analysis is performed to model the relationship between the dimensions and the target variable. The coefficients of the regression models are averaged to obtain a single regression model for the current domain (lines 10-12). This regression model provides insights into the separability of the dimensions.
- **Sampling**: The current domain is sampled again, along with the regression dimension. These sample-value pairs are considered as test candidates for validating the separability of dimensions (lines 22-33).
- **Validation**: For each sample-value pair, the accuracy is calculated using Equation 11 (line 41). This accuracy measure helps assess the separability of dimensions. The largest error encountered during the validation process is recorded and stored for further analysis (lines 44-45).

```
def fit(self):
    """ Tries to fit the regression model over a subspace of the domain """
    # Get random samples in the entire domain
    X, y = self.sampler.sample_domain(config.regression_samples, self.regression_dim)
    # Perform subspace regression at samples and take average of coefficients
    try:
        self.coefficients = np.zeros_like(self.coefficients)
        for X_sample in X:
            self.coefficients += self.fit_at_sample(X_sample)
        self.coefficients /= config.regression_samples
    except KeyboardInterrupt:
        sys.exit()
    # Quick return in case of infinite or NaN coefficeints
    if any(np.isnan(self.coefficients)) or any(np.isinf(self.coefficients)):
        self.max_error = np.Inf
        return
    # Get random test points in the entire domain (N=config.test_samples)
    X_test, y_test = self.sampler.sample_domain(config.test_samples, self.
                                                regression_dim)
    # Errors
    err_matrix = np.zeros((config.test_samples, config.test_values))

    for i in range(config.test_samples):
        # Evaluate regression model with x_i from the test point
        y_test_reg = self(X_test[i][self.regression_dim])
        # For each test point, sample a certain number of points in dimension x_i
        # with the test point as anchor
        X_tilde, y_tilde = self.sampler.sample_dimension(
            config.test_values, self.regression_dim, X_test[i])

        for j in range(config.test_values):
            # Evaluate regression model with the additionally sampled x_i
            y_tilde_reg = self(X_tilde[j][self.regression_dim])
            # Calculate error (see Quasi-Separability definition)
            err_matrix[i, j] = abs(y_test_reg + y_tilde[j] - y_tilde_reg - y_test[i])
    # Caluclate maximal error and choose a sample value for the model reduction
    self.max_error = err_matrix.max()
    self.sample_value = (X_tilde[err_matrix.max(0).argmin()][self.regression_dim])
```

Code Listing 3: Separability detection Märtens [2022]

The detection algorithm combines sampling, regression analysis, and validation to determine the quasi-separability of dimensions in a given dataset. The accuracy of the regression models and the errors calculated during the validation process provides insights into the separability of dimensions and guide subsequent modelling and analysis tasks.

Regression Algorithm Within the framework, each regression model incorporates a method called `fit_at_sample`, which is responsible for conducting scalar regression in a single dimension using an anchor sample from the current domain. The implementation details of the Taylor regression model can be found in Code 4. This model employs the normal equation to solve the least-squares problem, leveraging the computational capabilities of the *Numpy* library (Harris et al. [2020]) as the solver. Furthermore, the framework offers support for fitting exponential and periodic regression models, which rely on the *Scipy* library (Virtanen et al. [2020]) to solve their respective non-linear least-squares problems. The utilization of these libraries allows for efficient and accurate regression modelling within the framework, enhancing its capabilities for handling various regression tasks.

```python
def fit_at_sample(self, sample):
    """Performs the regression over the subspace defined by the sample.
    Args:
        sample: The sample defining the subspace.
    Returns:
        The coefficients of the regression model, excluding the constant term.
    """
    # Number of samples is at least as large as the number of coefficients
    n_coefficients = self.order + 1
    n_samples = n_coefficients + config.taylor_regression_samples

    # If the subspace has only one dimension, convert sample to a 1-element array
    sample = np.full(1, sample) if self.dims == 1 else sample

    # Sample random points in the subspace
    X, y = self.sampler.sample_dimension(n_samples, self.regression_dim, sample)

    # Compute regression matrix X (linear system)
    X_reg = np.empty((n_samples, n_coefficients))
    for i in range(n_samples):
        X_i = np.full(1, X[i]) if self.dims == 1 else X[i]
        for j in range(n_coefficients):
            X_reg[i][j] = X_i[self.regression_dim] ** j

    # Perform regression via normal equation
    coeffs = np.linalg.solve(
        np.matmul(X_reg.transpose(), X_reg),
        np.matmul(X_reg.transpose(), y))

    # Return all coefficients apart from the constant term
    return coeffs[1:]
```

Code Listing 4: Taylor regression model Märtens [2022]

3.4 Modelling Non-Separable Dimensions

After the *Quasi-Separability* analysis is complete, the non-separable dimensions must be modelled using other methods. The framework uses ANNs to learn the non-separable dimensions combined. Since the framework allows for the creation of many subspaces with different regressions and dimensions, we need to train an ANN for each subdomain.
The subspace regression tree provides an interface to access the subdomain nodes, which are referred to as *leaves*. A leaf node contains a subdomain and potentially non-separable dimensions that require learning by an ANN. However, all dimensions in a specific subdomain

Type	Hyperparameters	Description
General	output_dir	Output directory for generated files.
	debug	Flag to print debug messages or not.
RSQTOA	epsilon	Error threshold. The framework maintains the regression errors below the specified threshold.
	sampling_strategy	Sampling method for regression anchors and validation samples can be set to 'random', 'latin', or 'grid'.
	regression_samples	Number of regression anchors in the domain.
	test_samples	Number of validation samples.
	test_values	Number of scalar subspace values sampled for validation.
	max_taylor_order	Maximum order for Taylor regression.
	taylor_regression_samples	Number of samples used for Taylor regression.
	periodic_regression_samples	The number of samples used for periodic regression
	exponential_regression_samples	The number of samples used for exponential regression.
	regression_selection	Possible values: 'greedy', 'best'. Greedy regression selection stops analysis when valid models are found; best is chosen from all possible models.

Table 1: RSQTOA Framework Hyperparameters

may be separable, resulting in the reduced function being only a constant. In that case, an ANN is not necessary for that particular subdomain.
We need to sample and evaluate each subdomain, which we accomplish through the interface provided by the subspace node. Using this data, we train an ANN for each subdomain. In the end, we provide the subspace node with the learned model, allowing us to evaluate the combined reassembled model.

3.5 Configuration Options

The framework provides several options for configuring regression analysis. The user can utilize a *YAML* configuration file, pass command-line arguments, or set options from within Python. A *YAML* file that configures the most important options is demonstrated in Appendix A.1. Table 1 provides explanations for the configuration options.

3.6 Usage

The presented framework is implemented in Python 3, and to facilitate the interaction with the framework, an instance of the `RsqtoaRegressor` class can be created. This class consolidates the various components of the framework and builds an approximate model based on the specified parameters during the instantiation of the regressor.
To instantiate the `RsqtoaRegressor` object, three parameters are required for the constructor. These parameters include the configuration file path, a model object, and a function to instantiate a new approximator instance. The configuration file path is used to specify the desired configuration settings. The model object can be an instance of any of the model subclasses discussed in Section 3.1, such as `EuquationModel` or `DatasetModel`. Lastly, as discussed in Section 3.4, a new instance of an approximator is required for all non-separable dimensions, and a function can be passed to instantiate the required approximator, for this study, we use an Artificial Neural Network.
Once the `RsqtoaRegressor` object is instantiated, the model can be trained and used for prediction. Public methods `fit` and `predict` can be used for this purpose.

4 Experiments

This section presents the methodology and results of two experiments conducted to assess the validity of our approach. We evaluate the performance of our models based on the time required to train them and their measure of fitness. We begin by discussing the methodology used to construct our models and the evaluation metrics chosen. Subsequently, we describe in detail the two experiments conducted, one using a synthetic dataset and the other using the Yelp dataset, for collaborative filtering of user reviews to predict review helpfulness. To facilitate comparison, we train four models for each experiment: Linear Regression (ElasticNet), Artificial Neural Network (ANN), RSQTOA with Grid Sampling, and RSQTOA with interpolator.

Construction of Predictive Models In this empirical study, we aim to construct various predictive models through two experiments. The methodology employed in this research is presented in Figure 18. Firstly, for our synthetic data study, we generate data by utilizing a concrete equation. In the case of our collaborative filtering experiment, we collect data related to restaurants, their corresponding reviews, and the reviewers from the Yelp academic dataset.

Once the data is collected, we proceed with preprocessing and computing meta-data features. The dataset is then divided into three parts: training, validation, and testing. To ensure a fair comparison among features and avoid any feature dominating the learning process, we apply min-max scaling to the input features. This scaling technique transforms the features to a common range, typically between 0 and 1. In our study, we train four regression models: ElasticNet, Artificial Neural Network (ANN), RSQTOA with Grid Sampler, and RSQTOA with interpolator. The interpolator we use is the Nearest Neighbour, and for linear regression, we employ ElasticNet, which combines the advantages of Lasso and Ridge regression to provide a flexible and robust approach Zou and Hastie [2005].

Finally, the hyperparameters derived from the model development phase are utilized to conduct a 20-fold cross-validation procedure. In each fold, a new model is trained, and performance metrics are computed. This meticulous approach enables a thorough assessment of the models' performance and provides valuable insights for subsequent statistical tests.

Performance Measure To assess the efficacy of our trained models, we employ a 20-fold cross-validation technique and conduct statistical tests to determine the significance of improvement. We evaluate their performance through two distinct metrics: (1) analyzing the measure of fit, and (2) measuring the time required to train the model. For every fold, we train each of the four models from the beginning and assess their performance based on the following performance criteria.

- **Measure of Fitness (R^2 Score)**

  There are several evaluation metrics available to assess the accuracy and efficacy of regression models. The selection of an appropriate performance measure is contingent upon the objective of the regression problem and the characteristics of the data. These measures facilitate the identification of the most suitable model from among the candidate models and enable comparison of the performance of different models.

  The **R-squared (R^2) score**, which represents the coefficient of determination, is a widely employed performance metric for assessing the explanatory capability of a model. It quantifies the proportion of variance in the dependent variable that can be explained by the model. When comparing different models, the R^2 score serves as a valuable measure to evaluate and compare their respective abilities to account for the variance

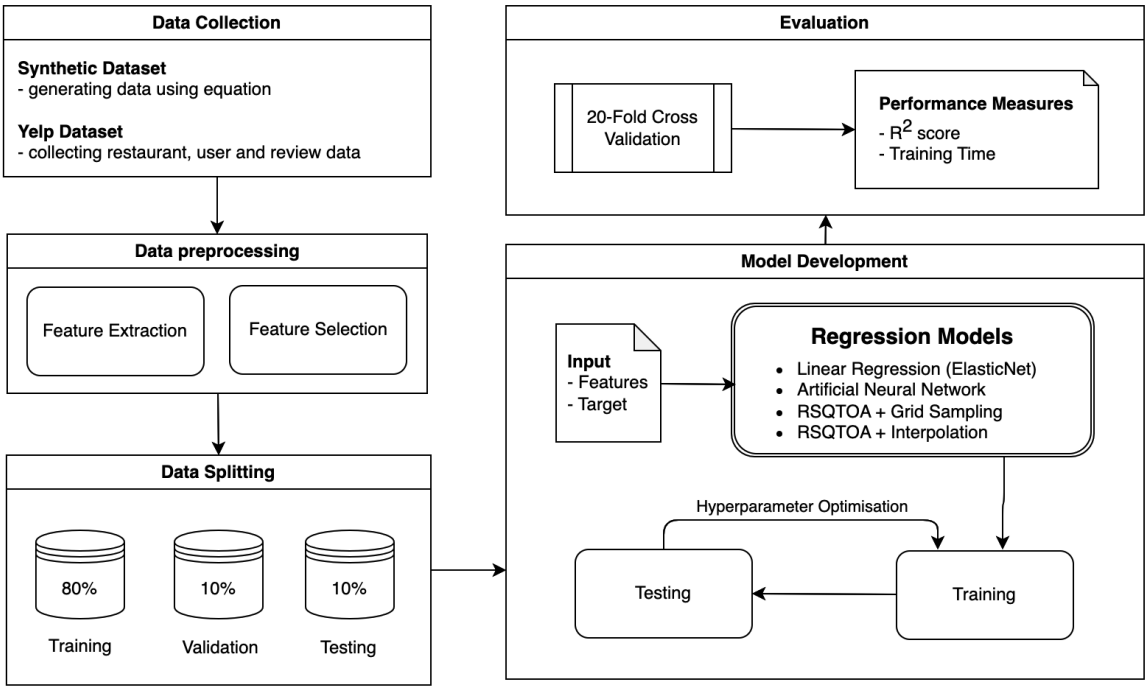

Figure 18: Experimental Setup and Methodology Overview

in the target variable. It provides insights into the relative performance and explanatory power of the models under consideration, enabling informed model selection and comparison.

$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}} \tag{16}$$

Here, SS_{res} represents the sum of squared differences between the actual values of the dependent variable and the predicted values by the model, while SS_{tot} denotes the total sum of squared differences between the actual values and the mean of the dependent variable.

It is important to note that the RSQTOA approach operates on the premise of achieving relatively efficient model construction within a shorter timeframe. Consequently, the performance scores of the RSQTOA models are expected to be lower compared to our baseline model, such as ANN. However, it is crucial to ensure that the RSQTOA scores remain commensurate with the baseline model. This indicates that despite the expedited modelling process, the RSQTOA approach maintains a satisfactory level of performance and provides a viable alternative for constructing predictive models.

- **Training Time**

  In the field of machine learning, the curse of dimensionality refers to the increase in time and complexity as the number of dimensions in a dataset grows. To tackle this challenge, our framework proposes a solution that involves segregating dimensions into separable and non-separable categories and employing different regression techniques for each category, namely scalar regression and neural networks. The reduction in dimensions not only aids in improving model training efficiency but also becomes particularly significant when working with datasets containing a large number of features.

To assess the performance of our proposed model, we focus on measuring the time required for model training, which is measured in milliseconds. It is worth emphasizing that reducing training time holds paramount importance when dealing with large datasets. By employing our framework, we aim to mitigate the computational burden associated with high-dimensional datasets and expedite the training process, thereby enabling more efficient and timely model development.

Statistical Hypothesis Testing The central objective of our study is to adapt the RSQTOA framework to facilitate the production of approximate models from datasets. This approach allows us to approximate multivariate scalar functions in a shorter timeframe, with some trade-off in the accuracy of the resulting models. As the RSQTOA framework utilizes Artificial Neural Networks (ANN) to approximate the non-separable aspects of the function, we consider ANN as the baseline for comparison. To achieve this, we train four different models on each dataset, which includes ElasticNet, ANN, RSQTOA with Grid Sampling, and RSQTOA with Nearest Neighbour Interpolator, as elaborated in the Paragraph 4.
To assess the performance of these models, we employ the results obtained from our 20–fold cross-validation and conduct statistical tests to determine whether there are significant differences in the scores produced by the different approaches. The RSQTOA approach, as proposed, aims to achieve shorter training times compared to ANN while producing models with comparable performance to our baseline. Additionally, we compare the performance measures of ANN with ElasticNet to gain insights into how both approaches, ANN and RSQTOA, fare in comparison to a traditional modelling approach such as Linear Regression when considering our datasets.
In this study, we begin by conducting normality tests on the training times and R^2 scores of our models. Based on the normality of our observations for the scores from the 20–folds, we choose to employ parametric or non-parametric tests. For the comparative analysis of R^2 scores, we utilize a two-tailed two-sample test to examine whether there is a significant difference between the scores of the two groups: ANN as the first group and ElasticNet or RSQTOA with Grid Sampling and RSQTOA with Nearest Neighbour as the second group. Directionality is not considered as the main objective is to determine if there is a significant difference in the scores produced by the two models. As previously mentioned, the ideal outcome would be the absence of any significant difference between the scores.
Regarding the comparison of training times, we hypothesize that the RSQTOA-based approach will exhibit shorter training times compared to ANN. Therefore, we employ a right-tailed two-sample test for this comparison, assuming that the ANN-based model would take more time than the RSQTOA-based models or ElasticNet. In all our tests, we set the significance level (alpha) to 0.05 to assess the statistical significance of the observed differences.

4.1 Experiment I – Synthetic Dataset

In this section, we investigate the performance enhancement achieved by utilizing the RSQTOA framework in constructing regression models. To conduct this experiment, we generate a synthetic dataset by employing a given underlying function and utilize it for training our regressors. Subsequently, we compare the function approximation derived from machine learning algorithms, such as ElasticNet and ANN, with the RSQTOA-guided approaches, namely, RSQTOA with Grid Sampler and Interpolator.
Let us consider the function $f : \mathbb{R}^3 \to \mathbb{R}$ defined as:

$$f(\boldsymbol{x}) = e^{x_0+x_1} + x_2 + 3 \tag{17}$$

Visualizing a function with three dimensions and a resultant dimension can be challenging. However, based on the definition of *Quasi-Separability* in Section 2.3, it seems that the dimension x_2 may be *quasi-separable*. Moreover, our framework may be capable of approximating this dimension by utilizing Taylor regression with an order of 1.

4.1.1 Data Generation

The primary goal of this study is to demonstrate the advantage of the RSQTOA framework in generating approximate models not only from functions but also from datasets. To this end, we use the Equation (17) to generate a synthetic dataset comprising 10,000 samples. The dataset is created by randomly sampling values for the dimensions x_0, x_1, and x_2, and computing the corresponding target value by evaluating the equation.

4.1.2 Results

Following the methodology described in Paragraph 4, a total of four regressors were trained for the purpose of this study. These regressors include ElasticNet, ANN, RSQTOA with Grid Sampling, and RSQTOA with Nearest Neighbour Interpolator.
To achieve optimal performance, each regressor was trained using carefully selected hyperparameters and configuration parameters. The specific choices made for these parameters, along with the corresponding results obtained for each model, are presented in the subsequent sections. The rigorous evaluation of different models and their associated parameters provides valuable insights into their performance and facilitates meaningful comparisons between the regressors.

Linear Regressor (ElasticNet) In this study, we employed the ElasticNet regressor, which combines L1 and L2 regularization, making it an ideal choice for achieving a balanced approach. Furthermore, we incorporated a polynomial representation of the feature variables during the training of our regression models. To determine the optimal hyperparameters for the ElasticNet model, we conducted a Grid Search over a predefined search space. The search space encompassed various options for the polynomial degree (ranging from 1 to 4), ElasticNet Alpha (ranging from 0.0001 to 10.0), and ElasticNet L1 Ratio (ranging from 0.1 to 0.9).
From the Grid Search results, we identified the best combination of hyperparameters, which included a polynomial degree of 4, ElasticNet Alpha of 0.0001, and the ElasticNet L1 Ratio of 0.9. These particular hyperparameters yielded the highest performance during the search process.
Using the obtained best hyperparameters, we conducted a thorough evaluation of the ElasticNet model's performance. This evaluation involved performing a 20-fold cross-validation procedure to assess the model's ability to generalize to unseen data. For each fold, we recorded

important performance measures, such as the R^2 score and training time. Detailed results of the 20 trained models can be found in Appendix A.3.1.
It is worth noting that ElasticNet was the best-performing model out of all the four modelling approaches with an outstanding R^2 score of 0.999, indicating a high degree of accuracy in capturing the variance in the dependent variable. Additionally, this model demonstrated remarkable efficiency, with a training time of only 0.1 seconds. These results highlight the effectiveness of the ElasticNet regressor with the selected hyperparameters in achieving both accuracy and efficiency in our regression modelling task.

Artificial Neural Network In this study, we employed an Artificial Neural Network (ANN) architecture consisting of five dense hidden layers. The node configurations for these layers were set as 50, 100, 100, 50, and 10, respectively. The activation function chosen for the hidden layers was the Exponential Linear Unit (ELU) Clevert et al. [2016]. During the training process, the ANN was trained for 100 epochs with a batch size of 256, utilizing the Adam optimizer. To prevent overfitting, we implemented an early stopping criterion based on a loss threshold of 10^{-5}. To evaluate the performance of the model, we conducted a 20-fold cross-validation, training a new model for each fold.
Figure 19 presents the average loss history for the models trained with the aforementioned configuration during the cross-validation process. In Table Appendix A.3.2, we provide detailed performance measures obtained from the 20 trained models. The average R^2 score achieved by the ANN was 0.994, indicating a strong ability to capture the underlying relationship between the variables. Furthermore, the average training time for the model was 10.671 seconds, indicating reasonable computational efficiency. These results highlight the effectiveness of the chosen ANN architecture and hyperparameters in achieving accurate predictions while maintaining computational efficiency.
However, it is worth noting that the model obtained using ANN demonstrated comparable performance to our ElasticNet model, but it required approximately 10 times more training time than ElasticNet. This observation highlights the trade-off between model performance and computational efficiency when selecting the appropriate regression approach for a given dataset.

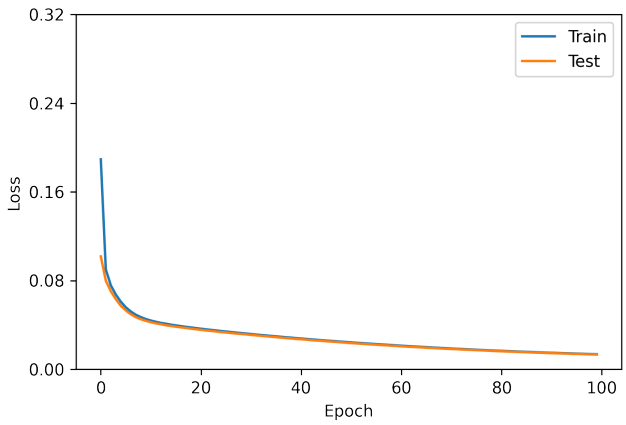

Figure 19: Average (20 Folds) Loss History for ANN on Synthetic Dataset

RSQTOA with Grid Sampler In this study, we employed the Grid Sampler algorithm, as discussed in Section 2.5.4, which relies on two critical hyperparameters: the number of

grid sections and the threshold. To determine the optimal values of these hyperparameters, we conducted several tests and settled on 8 and 50, respectively. In addition to ϵ as 0.1 for *Quasi-Separability* analysis.
For the RSQTOA framework to approximate the non-separable part, we used a dense neural network with five hidden layers, each comprising 50, 100, 100, 50, and 10 nodes, respectively. Exponential linear units were used as activation functions to approximate the non-separable dimensions. The batch size was set to 256 for 100 epochs, the Adam optimizer was utilized, and an early stop loss of 10^{-5} was implemented.
We then performed 20-fold cross-validation to obtain the performance metrics for our comparative study. The RSQTOA framework with the Grid Sampler algorithm was successful in separating the separable dimension, x_2, with a Taylor model of order 1. Equation (18) represents the Taylor regression generated by our framework for the dimension x_2, averaging the coefficients across 20 folds.
Additionally, Figure 20 displays the average loss history across 100 epochs for 20 folds for our ANN approximators representing the relationship for the dimensions x_0 and x_1, indicating that our neural network, starting with a higher error, converged much faster compared to the pure machine learning approach, i.e., Artificial Neural Network, but with higher loss.
The average R^2 score obtained was 0.969, indicating that the model was able to capture the underlying relationship quite well. Furthermore, the average training time taken to train the model was 8.472 seconds, indicating reasonable computational efficiency. Furthermore, the grid creation time for the sampler algorithm was 0.89 seconds. Which is an overhead when compared to the rest of the approaches. For a summary of the performance metrics for our RSQTOA framework with the Grid Sampler algorithm, please refer to Appendix A.3.3.

$$\phi(x_2) = 0.144 \times x_2 \tag{18}$$

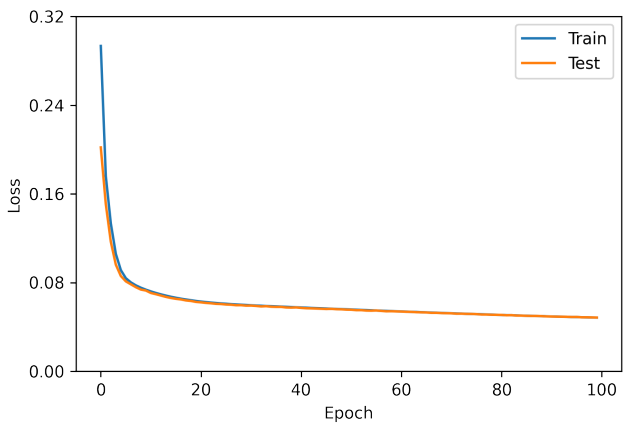

Figure 20: Average (20 Folds) Loss History for RSQTOA with Grid Sampling on Synthetic Dataset

RSQTOA with Interpolator In this study, we utilized an RSQTOA framework with the nearest neighbour interpolator to obtain an approximate model for the given dataset. To capture the non-separable part of the dataset, we trained a dense neural network with the RSQTOA framework. The neural network had five dense hidden layers with 50, 100, 100, 50, and 10 nodes respectively, and we used exponential linear units as the activation function. The model was trained with a batch size of 256 for 100 epochs using the Adam optimizer, and an

early stop loss of 10^{-5} was implemented. Additionally, we used ϵ as 0.09 for *Quasi-Separability* analysis and Latin-Hypercube for sampling.
To compare the performance of our approach with other methods, we used 20-fold cross-validation and obtained the relevant performance metrics. The results are summarized in Appendix A.3.4. Our framework with the nearest neighbour interpolator achieved an average R^2 score of 0.992, indicating that the model was successful in capturing the underlying relationship. Moreover, the average training time for the model was 8.439 seconds, indicating reasonable computational efficiency. The framework with the interpolator was able to segregate the separable dimension (x_2) using a Taylor model of order 1. The obtained Taylor regression (averaged coefficients for 20 folds) for the dimension x_2 is represented by Equation (19).
Furthermore, Figure 21 displays the average loss history across 100 epochs for 20 folds for our ANN approximators representing the relationship for the dimensions x_0 and x_1. The figure indicates that our neural network converged much faster compared to the grid sampler approach but had higher loss as compared to the pure machine learning approach, i.e., Artificial Neural Network.

$$\phi(x_2) = 0.142 \times x_2 \tag{19}$$

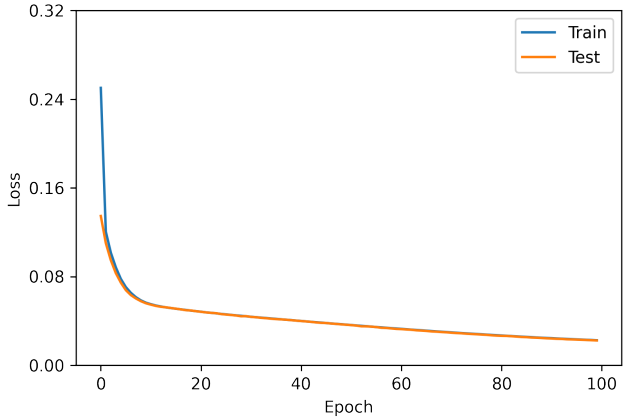

Figure 21: Average (20 Folds) Loss History for RSQTOA with Nearest Neighbour Interpolator on Synthetic Dataset

Discussion To compare the performance, we employ results from our 20-fold cross-validation and perform a statistical test to deduce whether the difference between the scores obtained from different approaches was significant or not. For this study, we begin by performing normality tests on the training time and R^2 scores for our models. However, after performing the Shapiro-Wilk test with α of 0.05, we concluded that the distribution of our scores failed the normality test. Hence, we decided to use a non-parametric test i.e. Mann-Whitney U Test for the comparative study.

- **R^2 Score**

  In order to compare the performance of different approaches with the baseline, we conducted statistical tests using a significance level (alpha) of 0.05. Specifically, we employed a two-tailed two-sample Mann-Whitney U test to assess the significance of the differences in R^2 scores between the RSQTOA-based approaches, ElasticNet, and ANN.

The results of the statistical tests revealed that the R^2 scores of ElasticNet were significantly different from those of ANN, with a p-value of 1.451×10^{-11} and a large standardized effect size of 0.85. This indicates that ElasticNet performed significantly better than ANN in terms of explaining the variance in the dependent variable.

Similarly, the R^2 scores of RSQTOA with Grid Sampler was significantly different from those of ANN, with a p-value of 6.529×10^{-10} and a large standardized effect size of 0.82. This suggests that the Grid Sampler-based approach achieved significantly different R^2 scores compared to ANN, with RSQTOA performing slightly better.

On the other hand, the R^2 scores of RSQTOA with Interpolator did not show a statistically significant difference from those of ANN, with a p-value of 0.08591 and a small standardized effect size of 0.27. This means that the performance of RSQTOA with Interpolator was not significantly different from that of ANN, indicating that both approaches performed similarly.

Although the statistical tests indicated significant differences in the R^2 scores, it is important to note that the median scores observed for ANN (0.974) and RSQTOA with Grid Sampler (0.995) were not substantially different. The lower performance of the Grid Sampler-based approach can be attributed to the limited samples available for training compared to the Interpolator-based approach, which was able to sample the entire domain of the dimensions. Additionally, the simplicity of the underlying function allowed the nearest neighbour Interpolator to effectively interpolate missing target values for non-present samples.

On the other hand, ElasticNet, with its regularization and higher degree, was able to capture the underlying relationship quite well compared to all three approaches. The combination of L1 and L2 regularization in ElasticNet allowed for a balance between feature selection and model complexity, leading to a more robust and accurate model. As a result, ElasticNet achieved nearly perfect scores on the R^2 metric, indicating its ability to explain the variance in the target variable.

Figure 22 presents the R^2 scores of the four models for each fold of the 20-fold cross-validation. The figure clearly illustrates that the model produced by RSQTOA with Interpolator exhibits a similar score to ANN, while the Grid Sampler-based approach shows a lower score, and ElasticNet achieves nearly perfect scores. This visual representation further reinforces the findings from the statistical tests and provides a comprehensive view of the models' performance across different folds.

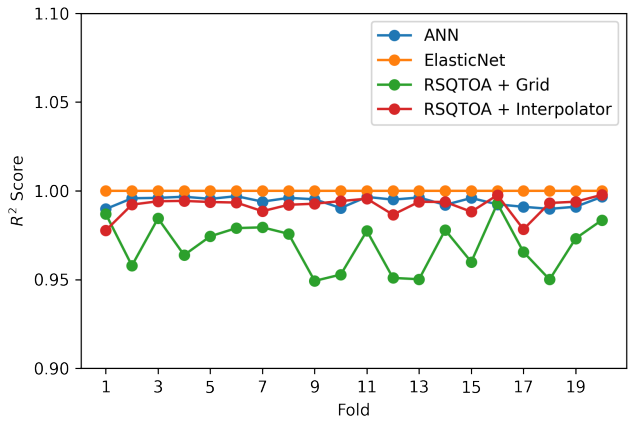

Figure 22: R^2 Scores comparison for Synthetic Dataset

- **Training Time**

  We conducted a comparative analysis of the training time required by ElasticNet and RSQTOA-based approaches in comparison to ANN. To ensure a rigorous evaluation, we performed three separate statistical tests: (1) ANN with ElasticNet, (2) ANN with RSQTOA + Grid Sampler, and (3) RSQTOA + Interpolator. For each test, we employed a right-tailed two-sample Mann-Whitney U test with a significance level (α) of 0.05. The alternative hypothesis stated that the training time for both ElasticNet and RSQTOA-based approaches would be lower than that for ANN. The obtained p-values were 1.451e-11 for ElasticNet and 7.254e-12 for both RSQTOA-based approaches. Since these p-values were below the significance level, we rejected the null hypothesis and concluded that the training time of the ANN-based approach was significantly higher than that of the RSQTOA-based approaches. Moreover, the observed standardized effect size for all three tests was large (0.85), indicating a substantial difference in training time between ANN and the ElasticNet as well as RSQTOA-based approaches.

  Figure 23 visually represents the training time taken by the three models for each fold of our 20-fold cross-validation. The graph clearly illustrates that the RSQTOA-based approaches exhibited shorter training times compared to ANN. This improvement in performance can be attributed to the ability of our framework to separate and represent the dimension x_2 as a first-order Taylor regression. As a result, the ANN used in the RSQTOA framework had fewer parameters to train and required fewer samples to converge. Additionally, the figure also demonstrates that both RSQTOA-based approaches exhibited similar training times, while ElasticNet showcased promising performance compared to the other three models. Furthermore, ElasticNet demonstrated faster training times compared to the other models due to its efficient combination of L1 and L2 regularization, which helped in reducing the dimensionality and improving computational efficiency.

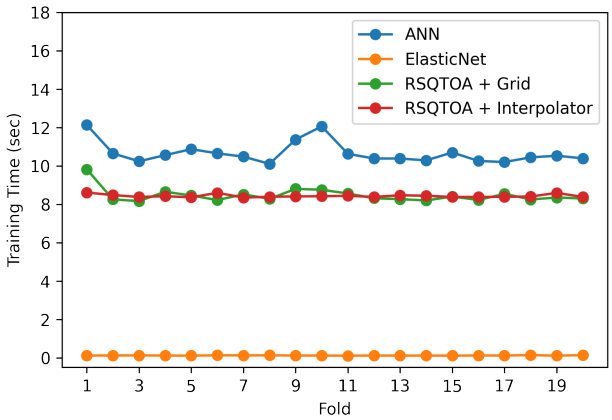

Figure 23: Training time comparison for Synthetic Dataset

4.2 Experiment II – Collaborative Filtering of Reviews

In digital platforms, recommendation systems play a crucial role in helping users discover and engage with relevant content, products, or services. These systems leverage user preferences, behaviour, and feedback to provide personalized recommendations. In this section, we explore a practical application where our framework can be utilized. We not only evaluate the performance of our approximated model in terms of both computational efficiency and prediction accuracy but also compare it with a model generated using another widely used machine learning technique, namely Artificial Neural Network (ANN). This comparison allows us to gain insights into the effectiveness of our framework in the context of recommendation systems.

4.2.1 Literature Review

- **Role of Electronic Word-of-Mouth in Promoting Business Sustainability**

  The role of eWOM in promoting business sustainability has been widely recognized, particularly in the context of experiential goods such as hotels and restaurants that are featured with intangible attributes (Vlachos [2012]). In the restaurant industry, customers are unable to objectively assess the characteristics of products or services prior to consumption or spatial movement due to the intangible and impalpable elements of the industry. As a result, customers tend to obtain detailed information from various sources to reduce their perceived uncertainty and risk before making a decision (Mauri and Minazzi [2013]). The volume of restaurant reviews written by customers has been found to be an indicator of the popularity of a restaurant (Zhang et al. [2010]), and a positive relationship between the number of online reviews and restaurant performance has been established (Kim et al. [2016]).

  Analyzing online reviews can also provide valuable insights into customer attitudes and behaviour, which can be used by business owners and marketers to improve their services and create a sustainable competitive advantage. Previous studies have examined online reviews to understand customer experiences and satisfaction in various service contexts, such as the hotel industry, short-term rental industry, airline industry, and wellness industry. Within the restaurant industry, (Pantelidis [2010]) conducted an empirical study of meal experiences using 2471 online restaurant comments and identified six salient factors in a diner's evaluation of a restaurant: food, service, ambience, price, menu, and décor. (Yan et al. [2015]) analyzed 10,136 Chinese restaurant reviews and found that food quality, price and value, service quality, and atmosphere were the key factors influencing customers' revisit intention.

- **Navigating the Ocean of Online Reviews**

  The volume of online reviews has experienced exponential growth, reaching 148.3 million on Yelp, more than twice the amount recorded in 2014 Statista [2021]. This abundance of reviews can overwhelm consumers, leading them to rely on a limited number of reviews for their purchase decisions Malhotra [1984], Kwon et al. [2015]. To aid customers in this decision-making process, helpful reviews have been identified as valuable peer-generated product evaluations that provide assistance Mudambi and Schuff [2010]. By presenting helpful reviews, e-commerce platforms can reduce the time and effort consumers spend sifting through vast amounts of reviews Li et al. [2013].

  To tackle the challenge of information overload in products with numerous reviews, leading websites like Amazon.com and Yelp.com employ a "Most Helpful First" sorting option, which relies on readers' "useful" votes to rank reviews by their usefulness Gilbert

and Karahalios [2010]. However, this approach may not be suitable for newly written reviews that have not had the opportunity to accumulate helpfulness votes Ghose and Ipeirotis [2011], Kim et al. [2006]. To address this limitation, automatic helpfulness estimation methods, such as predictive analytics, are employed. In this study, we adopt a regression approach to model the prediction of review helpfulness and estimate its value.

- **Studies on Review Helpfulness Prediction**

  Several previous studies have identified factors that influence the "helpfulness" of online customer reviews. According to the heuristic systematic model (HSM), these factors can be classified into two types: (1) Central route cues, which are associated with the features of the review content, such as review quality, length, readability, type, and extremity (Chua and Banerjee [2016], Kwok and Xie [2016] and Pan and Zhang [2011]); and (2) peripheral cues, which are associated with the features of the information source, such as reviewer expertise, gender, and reputation (Baek et al. [2012], Racherla and Friske [2012] and Liu and Park [2015]). However, Hong et al. [2017] noticed that the predictors of review helpfulness often yield inconsistent results. In their meta-analysis, they found that review readability and review ratings did not significantly influence review helpfulness. However, it is observed that the aforementioned determinants of review helpfulness are dominantly featured by numerical features (Wang et al. [2019]).

  In recent years, research on review helpfulness has focused on semantic and linguistic style features in online reviews (Wang et al. [2019] and Luo and Tang [2019]). Racherla and Friske [2012] examined the impact of the reviewer and review factors on perceived review usefulness across three categories, where restaurants were chosen as an experiential-based service. Their results showed that experiential-based services are a function of individual taste. Liu and Park [2015] later extended this work by focusing exclusively on restaurant reviews. They added review readability and customer-perceived enjoyment as two qualitative antecedents of review usefulness and identified these two variables as the most influential factors of review helpfulness. Ngo-Ye and Sinha [2014] conducted recency, frequency, and monetary value (RFM) analysis in predicting review helpfulness. In addition to widely examined review and reviewer-specific characteristics, other factors influencing review helpfulness include review orders (Zhou and Guo [2017]), emotions, and linguistic styles (Wang et al. [2019]), as well as temporal, exploratory, and sensory cues (Yan et al. [2015]). Studies have focused on restaurants in New York City, Las Vegas, and Los Angeles as the most likely samples (Wang et al. [2019], Racherla and Friske [2012] and Li et al. [2018]). However, few studies have taken into account both emotions and restaurant experience aspects in predicting restaurant review helpfulness.

4.2.2 Methodology

- **Data Collection**

  The Yelp Open Dataset, a subset of Yelp's business, reviews, and user data, was used for this study and was obtained from `https://www.yelp.com/dataset`. Yelp Inc., an American company, is responsible for Yelp.com, which publishes crowd-sourced reviews about businesses. The dataset contains approximately 70 million user reviews written by 20 million users for 1.5 million businesses across 11 metropolitan areas. It consists of six JSON files, each containing information about a different type of entity, including businesses, users, reviews, tips, and check-ins. Each file is composed of a single object type, with one JSON object per line. For the purpose of this study, the JSON

Entity	Feature	Description
Restaurant	Stars	Average rating received by the restaurant
	Review Count	Number of reviews received by the restaurant
	Categories	Type of business
User	Review Count	Number of reviews posted by the user
	Yelping Since	Date time since user is on the platform
	Useful	Number of useful votes received across review posted
	Elite	Number of years user has been an elite user
	Friends	Number of friends
	Fans	Number of fans
	Average Stars	Average rating across reviews posted by user
Review	Stars	Rating associated with the review
	Useful	Number of useful votes received
	Text	Review Text
	Date	Date the review was posted

Table 2: Yelp Dataset Raw features by Entities

dataset files were transformed into a tabular format, specifically CSV, to facilitate data analysis. Our focus in this study is solely on businesses categorized as restaurants. Analyzing reviews within the context of restaurants allows for a more specific and targeted investigation of the subject matter.

Since users rely solely on visible content while reading reviews, we only used the businesses, users, and review data entities to create the final dataset, which can be used to develop a model for predicting the helpfulness score of a review. Reviews that received less than ten helpful votes were excluded from the analysis to avoid introducing biases in the model. Table 2 outlines the features present in each data entity that will be used in our future analysis.

- **Model Features**

  This section presents the relevant dependent and independent variables that are utilized in constructing the review helpfulness prediction model. In this study, the prediction of review helpfulness is formulated as a regression problem, where the dependent variable is a real number indicating the relative helpfulness of a review compared to others posted for the restaurants. We begin by discussing the derivation and elaboration of the dependent variable, referred to as the "helpfulness score." Subsequently, we examine the independent variables that are readily available in the provided dataset. Additionally, we consider several auxiliary variables that have been identified in previous research as potential contributors to a review receiving more useful votes. These auxiliary variables aim to capture the intricacies associated with review helpfulness and enhance the predictive performance of the model.

  - **Dependent Variable**

    Helpfulness Score

    The dependent variable in our review helpfulness prediction model is derived from the variable "useful." However, it is essential to consider the limitations of directly comparing the number of useful votes across different restaurants. Such comparisons may not be suitable due to potential variations in votes based on the popularity and size of each restaurant.

To address this limitation and enable fair comparisons and evaluations, we introduce the concept of the "Helpfulness Score". The Equation (20) represents the mathematical formulation for deriving the helpfulness score:

The significance of the Helpfulness Score lies in its ability to provide a normalized measure of usefulness for reviews, taking into account the context of each restaurant. By incorporating this metric, our model facilitates fair and meaningful comparisons of review helpfulness across different restaurants, thereby enhancing the reliability and validity of the evaluation process.

$$Helpfulness\ Score = \frac{\#Useful\ Votes\ of\ a\ Review}{\#Useful\ Votes\ of\ all\ Reviews\ of\ a\ Restaurant} \tag{20}$$

In our study, we made the decision to only consider reviews with 10 or more 'useful' votes to mitigate potential biases. This threshold was chosen to ensure that the reviews selected for analysis have a significant number of useful votes, indicating a higher level of credibility and reliability. By excluding reviews with fewer useful votes, we aim to minimize the impact of potential outliers and subjective judgments, resulting in a more robust analysis.

After computing the helpfulness scores using Equation (20), we observed that the distribution of scores was positively skewed. To address this skewness, we performed a cube root transformation on the helpfulness scores. This transformation helps to make the distribution more symmetric and reduces the influence of extreme values, thereby improving the suitability of the data for regression analysis. By applying the cube root transformation, we ensure that the regression model is more robust and provides reliable estimates of the relationships between variables.

- **Independent Variables**

  Yelp, being an online platform, offers users the ability to assess the usefulness of reviews. The assessment of a review's usefulness by a user is influenced by various cues and factors that come into play during their decision-making process. In our investigation, these factors are categorized into three groups: restaurant, user, and review factors. The tables (Table 3, Table 4, and Table 5) outline the factors associated with each category and provide descriptions of how these factors were derived and rationale behind selecting them as independent variables.

- **Feature Selection**

  Feature selection is a crucial step in building an accurate and efficient machine-learning model. It involves identifying and selecting a subset of relevant features or variables that contribute the most to the prediction performance of the model. Using all available features may result in overfitting and poor generalization to new data. As mentioned in Section 4.2.2 under Independent Variables, we have 28 independent variables. To select the features that contribute the most, we followed a step-by-step approach.

  - **Variance Threshold**

    We begin with the variance threshold feature selection approach. Which involves removing features with low variance and retaining those with high variance. This step is based on the assumption that features with low variance do not exhibit much variation across samples and may not provide valuable information to the learning algorithm. On the other hand, features with high variance are more likely to contain informative and useful information for the learning algorithm.

Feature	Description
Stars	Average rating
Total reviews	Number of reviews received
Elapsed days	Duration (days) between the date of the latest review and the oldest review posted.
Positive reviews	Number of reviews with rating 4 stars and more
Ratio positive reviews	Ratio of Positive reviews to the total reviews received
Negative reviews	Number of reviews with rating 2 stars and less
Ratio negative reviews	Ratio of Negative reviews to the total reviews received
Immediacy Index Malik and Hussain [2017]	Restaurant with large score shows that there is less number of reviews posted but all reviews have high ratings $= \frac{Average\ review\ rating\ of\ the\ restaurant}{Total\ reviews\ for\ the\ restaurant}$

Table 3: Restaurant Features

Feature	Description
Recency	Days between reviewer's current and previous review
Frequency	Reviewer's previous review count
Monetary Value Ngo-Ye and Sinha [2014]	Average number of helpfulness votes received by the reviewer across all the reviews one has written $= \frac{\sum_{i=1}^{N} x_{\mathrm{i}} - x_{\mathrm{current}}}{N-1}$ where N is the total number of reviews written by the reviewer, x_i is the number of useful votes for a review i written by the reviewer, and $x_current$ is the number of useful votes for the current review.
Activity length	Days between the first and last review
Total useful votes	Useful votes received across all the reviews posted
Total reviews	Number of reviews posted
Fans	Number of fans
Average stars	Average rating given across all the reviews posted
Elite count	Number of years the user has been an elite user of the platform
Number of friends	Number of friends

Table 4: User Features

To perform this filtering based on variance, we utilized the `VarianceThreshold` module from the `scikit-learn` library. In our analysis, we set a threshold of 0.05, which determines which features are retained. As a result of this variance thresholding, certain features such as average user stars, star rating of the restaurant, and subjectivity score of the review were dropped, and we are left with 25 independent variables for further analysis.

– **Correlated Features**

The next step in our analysis involves employing the correlation feature selection technique to identify the most relevant features in our dataset. This technique is widely used and is based on the assumption that highly correlated features

Feature	Description
Stars	Rating associated
Elapsed days	Number of days passed since posted
Polarity score	Sentiment score between -1.0 (negative sentiment) and 1.0 (positive sentiment)
Subjectivity score	Subjectivity score between 0.0 (very objective) and 1.0 (very subjective)
Number of nouns	Number of nouns in the text
Number of sentences	Number of sentences in the text
Words per sentences	Average words per sentence
No of difficult words	Number of difficult words in the text
Automated readability Index	Measures text understandability and represents US grade level
Coleman–Liau index	Readability score that relies on characters instead of syllables per word

Table 5: Review Features

are likely to be redundant and can be eliminated without significantly impacting the performance of the model. To implement this method, we utilized a curated dataset containing 25 independent variables from the previous step. We calculated the Pearson correlation coefficient between each feature and the rest of the features, setting a threshold of 0.9. Subsequently, we removed any independent variables that exhibited a correlation coefficient greater than 0.9. This process resulted in the removal of two additional features, namely the count of positive reviews for the restaurant and the ratio of negative reviews for the restaurant. Consequently, our dataset now consists of 23 independent variables, which have been selected based on their lower correlation with other features.

– **Mutual Information**

It is a statistical technique used to quantify the dependence between two variables and examine the relationship between input features and the target variable. The objective is to identify the features that have the highest Mutual Information (MI) with the target variable, indicating their importance in predicting the target variable. The MI regression technique calculates the scores between each feature and the target variable, which measures the amount of information about the target variable that can be derived from the feature. Features with high MI scores are considered more significant in predicting the target variable and are selected.

To obtain the features with the highest scores, we utilize the `mutual_information_regression` module from the `scikit-learn` `feature_selection` package. In Figure 24, we present the MI scores for our features, which were obtained from the correlation filtering step. Additionally, we employ the `SelectPercentile` `feature_selection` module from `scikit-learn` to select the top 20th percentile of features using `mutual_information_regression` as the scoring function. After filtering, we are left with five features, namely `total_reviews`, `elapsed_days_restaurant`, `ratio_positive_reviews`, `negative_reviews`, and `immediacy_index`, which we utilize to develop our helpfulness prediction model.

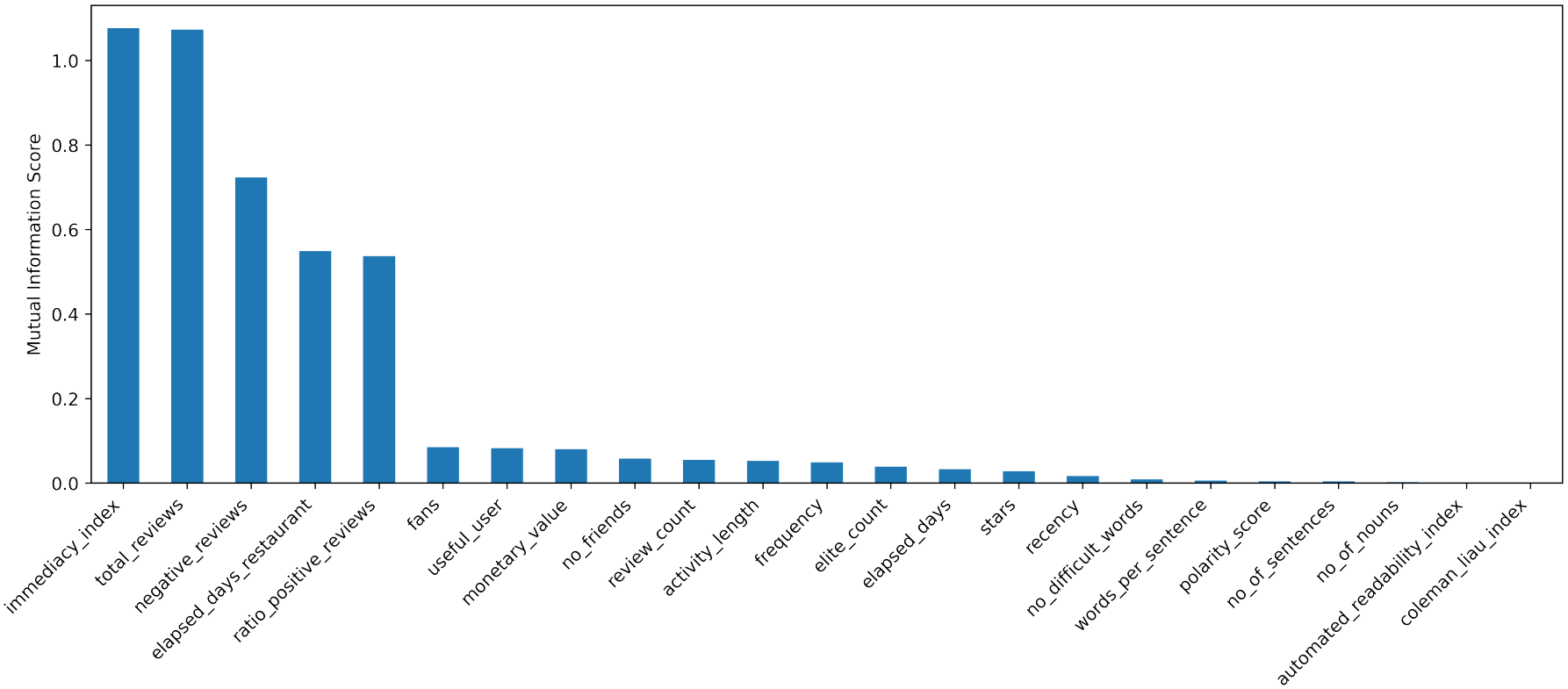

Figure 24: Mutual Information Regression Scores

4.2.3 Results

Following the methodology described in Paragraph 4, a total of four regressors were trained for the purpose of this study. These regressors include ElasticNet, ANN, RSQTOA with Grid Sampling, and RSQTOA with Nearest Neighbour Interpolator.
To achieve optimal performance, each regressor was trained using carefully selected hyperparameters and configuration parameters. The specific choices made for these parameters, along with the corresponding results obtained for each model, are presented in the subsequent sections. The rigorous evaluation of different models and their associated parameters provides valuable insights into their performance and facilitates meaningful comparisons between the regressors.

Linear Regressor (ElasticNet) We opted for the ElasticNet regressor, which integrates both L1 and L2 regularization techniques, providing a balanced approach for our regression task. Additionally, during the training phase of our regression models, we introduced a polynomial representation of the feature variables to capture potential non-linear relationships. To identify the optimal hyperparameters for the ElasticNet model, we conducted a Grid Search over a predefined parameter space. The parameter space encompassed various options for the polynomial degree (ranging from 1 to 4), ElasticNet Alpha (ranging from 0.0001 to 10.0), and ElasticNet L1 Ratio (ranging from 0.1 to 0.9).
Based on the results obtained from the Grid Search, we determined the best combination of hyperparameters, which included a polynomial degree of 4, an ElasticNet Alpha value of 0.0001, and an ElasticNet L1 Ratio of 0.1. These specific hyperparameters exhibited the highest performance during the search process.
Using the identified best hyperparameters, we conducted a comprehensive evaluation of the ElasticNet model's performance. The evaluation involved employing a 20-fold cross-validation procedure to assess the model's ability to generalize to unseen data. For each fold, we recorded essential performance metrics, such as the R^2 score and training time. Detailed results of the 20 trained models can be found in Appendix A.3.5.
It is essential to note that, unlike the Synthetic Dataset Experiment, ElasticNet was not the top-performing model among the four approaches, in the current study, it demonstrated a performance closely resembling RSQTOA with Grid Sampling, achieving an R^2 score of 0.686. Moreover, the ElasticNet model exhibited remarkable efficiency, with an average training time of only 3.745 seconds. These results underscore the effectiveness of the ElasticNet regressor with the selected hyperparameters in achieving both accuracy and efficiency in our regression modelling task.

Artificial Neural Network In this study, we employed an Artificial Neural Network (ANN) architecture consisting of five dense hidden layers. The node configurations for these layers were set as 50, 100, 100, 50, and 10, respectively. The activation function chosen for the hidden layers was the ELU. During the training process, the ANN was trained for 100 epochs with a batch size of 256, utilizing the Adam optimizer. To prevent overfitting, we implemented an early stopping criterion based on a loss threshold of 10^{-5}. To evaluate the performance of the model, we conducted a 20-fold cross-validation, training a new model for each fold.
Figure 25 presents the average loss history for the models trained with the aforementioned configuration during the cross-validation process. Analysis of the residual plot suggests that the model began with a low loss and rapidly converged within the first ten epochs. However, after that point, the performance remained relatively stable. In Appendix A.3.6, we provide detailed performance measures obtained from the 20 trained models. The average R^2 score achieved by the ANN was 0.711. Furthermore, the average training time for the model was 43.912 seconds, indicating reasonable computational efficiency.

However, it is worth noting that the model obtained using ANN demonstrated comparable performance to our ElasticNet model, but it required approximately 10 times more training time than ElasticNet. This observation highlights the trade-off between model performance and computational efficiency when selecting the appropriate regression approach for a given dataset.

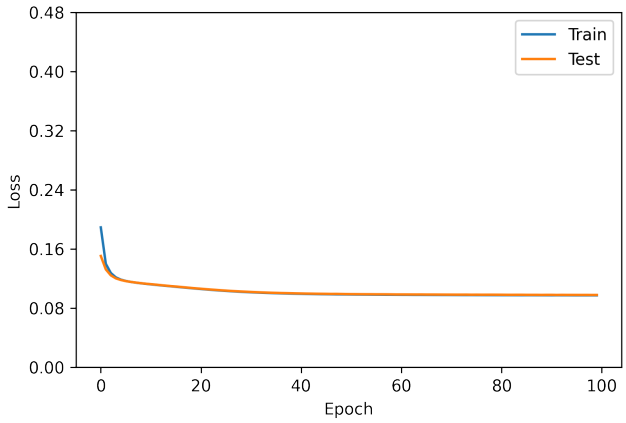

Figure 25: Average (20 Folds) Loss History for ANN on Yelp Dataset

RSQTOA with Grid Sampler We utilized the Grid Sampler algorithm, as discussed in Section 2.5.4. The sampler algorithm depends on two critical hyperparameters, namely, the number of grid sections and the threshold. We conducted several tests to determine the optimal values of these hyperparameters and settled on 30 and 100, respectively. In addition to ϵ as 0.8 for *Quasi-Separability* analysis.

To approximate the non-separable part in the RSQTOA framework, we employed a dense neural network with five hidden layers. Each hidden layer consisted of 50, 100, 100, 50, and 10 nodes, respectively. The activation functions used were exponential linear units to approximate the non-separable dimensions. We set the batch size to 256 for 100 epochs, utilized the Adam optimizer, and implemented an early stop loss of 10^{-5}.

We performed 20-fold cross-validation to obtain the performance metrics for our comparative study with the other two models. Our approach was successful in separating the separable dimensions, `ratio_positive_reviews` and `elapsed_days_restaurant`, with a Taylor model of order 1. Equations (21) and (22) represent the Taylor regression generated by our framework for the dimensions `ratio_positive_reviews` and `elapsed_days_restaurant`, respectively, averaging the coefficients across 20 folds.

Additionally, Figure 26 displays the average loss history across 100 epochs for 20 folds for our ANN approximators representing the relationship for the rest of the dimensions and target value. This indicates that our neural network started with a higher error compared to the pure ANN and converged to approximately the same error after 20 epochs, a bit later than the pure machine learning approach, i.e., Artificial Neural Network.

The average R^2 score obtained was 0.689, indicating that the model captured the underlying relationship at par with the ANN-based approach and marginally better than ElasticNet. Furthermore, the average training time taken to train the model was 24.266 seconds, indicating that the approach was approximately twice as faster as compared to the ANN-based approach but five times slower than ElasticNet. For a summary of the performance metrics for our RSQTOA framework with the Grid Sampler algorithm, please refer to Appendix A.3.7.

$$\phi(ratio_positive_reviews) = -0.162 \times ratio_positive_reviews \tag{21}$$

$$\phi(elapsed_days_restaurant) = -0.031 \times elapsed_days_restaurant \tag{22}$$

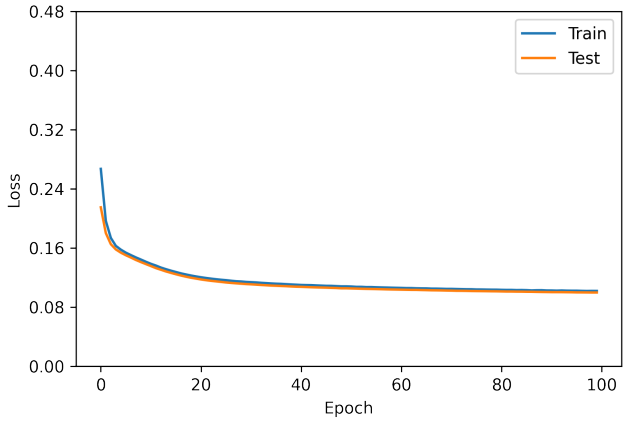

Figure 26: Average (20 Folds) Loss History for RSQTOA with Grid Sampling on Yelp Dataset

RSQTOA with Interpolator We implemented the RSQTOA with the nearest neighbour interpolator to create an approximate model for the given dataset. To address the non-separable portion of the dataset, we employed a dense neural network with five hidden layers containing 50, 100, 100, 50, and 10 nodes, respectively. The activation function used was exponential linear units. During the training process, we utilized the Adam optimizer with a batch size of 256 and employed early stopping with a criterion of 10^{-5} for a total of 100 epochs. Additionally, we used ϵ as 0.9 for *Quasi-Separability* analysis and Latin-Hypercube for sampling.

For assessing the performance of our approach, we conducted a 20-fold cross-validation and computed relevant performance metrics. The results, summarized in Appendix A.3.8, revealed that our approach with the nearest neighbour interpolator achieved an average R^2 score of 0.389. This score suggests that the model had limited success in capturing the underlying relationship within the dataset. Nevertheless, it is worth noting that the average training time for the model was 36.155 seconds, indicating reasonable computational efficiency.

Notably, in most of the folds, the approach with the interpolator struggled to separate any dimension effectively, and the separable dimensions varied across the folds. This observation indicates that while the approach generated an approximate model in a reasonable amount of time, it failed to adequately explain the underlying relationship, resulting in the worst R^2 scores among the four approaches.

Furthermore, Figure 27 in the appendix displays the average loss history across 100 epochs for the ANN approximators. The figure illustrates that our neural network for non-separable dimensions encountered challenges in converging, leading to higher errors at the end of the 100 epochs.

Discussion To compare the performance, we employ results from our 20-fold cross-validation and perform a statistical test to deduce whether the difference between the scores obtained

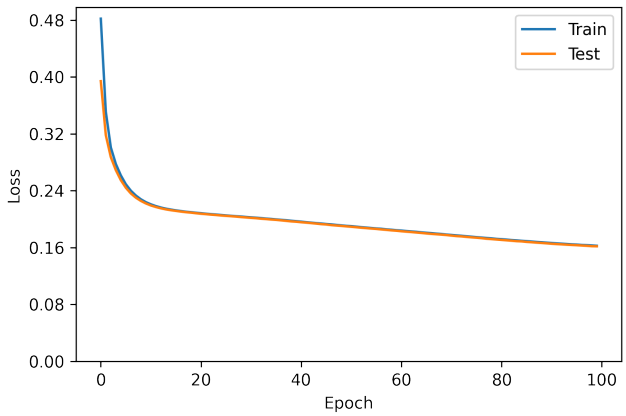

Figure 27: Average (20 Folds) Loss History for RSQTOA with Nearest Neighbour Interpolator on Yelp Dataset

from different approaches was significant or not. For this study, we begin by performing normality tests on the training time and R^2 scores for our models. However, after performing the Shapiro-Wilk test with α of 0.05, we concluded that the distribution of our scores failed the normality test. Hence, we decided to use a non-parametric test i.e. Mann-Whitney U Test for the comparative study.

- **R^2 Score**

  To compare the performance of different approaches with the baseline, we conducted statistical tests using a significance level (α) of 0.05. Specifically, we utilized a two-tailed two-sample Mann-Whitney U test to assess the significance of the differences in R^2 scores between the RSQTOA-based approaches, ElasticNet, and ANN.

  The results of the statistical tests showed that the R^2 scores of ElasticNet were significantly different from those of ANN, with a p-value of 4.184×10^{-6} and a large standardized effect size of 0.72. Further examination revealed that the scores were lower than those of ANN, which had an average R^2 score of 0.686, quite close to the average scores achieved by ANN and RSQTOA with Grid Sampling approaches.

  Similarly, the comparison of ANN with RSQTOA + Grid Sampler yielded a p-value of 2491×10^{-8}, indicating a statistically significant difference between the randomly selected R^2 scores of both groups. The observed standardized effect size was large (0.63), suggesting substantial differences between the values of both groups. However, it is worth noting that despite the significant difference, the median R^2 score observed was 0.713 for ANN, compared to 0.69 for Grid Sampler, which may not be substantially different. The low performance of our Grid Sampler-based approach can mainly be attributed to the limitation of available training samples.

  On the other hand, the R^2 scores of RSQTOA with Interpolator yielded a p-value of 1.451×10^{-11}, indicating a statistically significant difference between the R^2 scores of ANN and RSQTOA with Interpolator. The standardized effect size was 0.85, suggesting a large difference between the values of both groups. The potential poor performance of the Interpolator-based approach may be attributed to the challenges posed by high-dimensional cases, where data points are sparsely located, and the surface forms a saddle Bengio [2007]. Consequently, using the nearest neighbour interpolator might find it difficult to locate the nearest points in high-dimensional space, leading to incorrect target value computation for sampled data points.

The R^2 scores of our four models for each fold of the 20-fold cross-validation are illustrated in Figure 28. It is evident that the models produced by RSQTOA with Grid Sampler and ElasticNet show similar scores to ANN, whereas the Interpolator-based approach exhibits lower scores and higher variance.

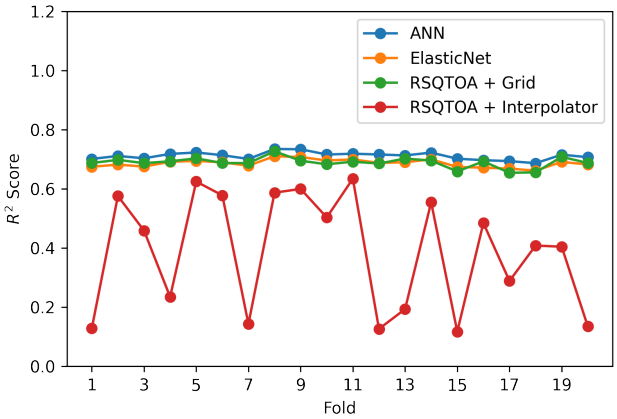

Figure 28: R^2 Scores comparison for Yelp Dataset

- **Training Time**

  We conducted a comparative analysis of the training time required by ElasticNet and RSQTOA-based approaches in comparison to ANN. To ensure a rigorous evaluation, we performed three separate statistical tests: (1) ANN with ElasticNet, (2) ANN with RSQTOA + Grid Sampler, and (3) RSQTOA + Interpolator. For each test, we employed a right-tailed two-sample Mann-Whitney U test with a significance level (α) of 0.05. The alternative hypothesis stated that the training time for both ElasticNet and RSQTOA-based approaches would be lower than that for ANN.

  The obtained p-values were 1.451×10^{-11} for ElasticNet and for both RSQTOA-based approaches. Since these p-values were below the significance level, we rejected the null hypothesis and concluded that the training time of the ANN-based approach was significantly higher than that of the RSQTOA-based approaches. Moreover, the observed standardized effect size for all three tests was large (0.85), indicating a substantial difference in training time between ANN and the ElasticNet as well as RSQTOA-based approaches.

  Figure 29 visually represents the training time taken by the three models for each fold of our 20-fold cross-validation. The graph clearly illustrates that the RSQTOA-based approaches exhibited shorter training times compared to ANN. This improvement in performance can be attributed to the ability of our framework to separate and represent the dimension as a first-order Taylor regression. As a result, the ANN used in the RSQTOA framework had fewer parameters to train and required fewer samples to converge. Additionally, the figure also demonstrates that both RSQTOA-based approaches exhibited similar training times, while ElasticNet showcased promising performance compared to the other three models. Furthermore, ElasticNet demonstrated faster training times compared to the other models due to its efficient combination of L1 and L2 regularization, which helped in reducing the dimensionality and improving computational efficiency.

It is important to note that although our RSQTOA with Grid Sampler approach yielded comparable performance to ANN (Artificial Neural Network) and required considerably less

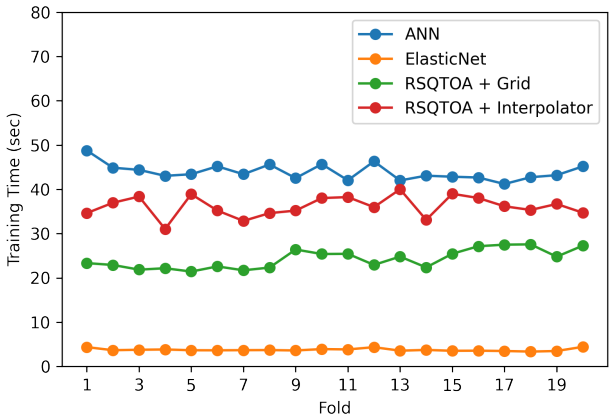

Figure 29: Training time comparison for Yelp Dataset

time for training, it took 8.287 minutes to prepare the grid needed for sampling. This grid preparation time becomes a bottleneck when we increase the dimensions we want to consider for modelling. This means that as the number of dimensions in our model increases, the time required for grid preparation also increases significantly.

The grid preparation time is sensitive to the number of data points and dimensions we want to model. As the dataset size grows or the number of dimensions increases, the grid preparation process becomes more computationally intensive and time-consuming. Therefore, careful consideration is required when applying the Grid Sampler approach to high-dimensional datasets or datasets with a large number of data points.

Additionally, as discussed, the Grid Sampler approach also yielded two different scalar approximations for the dimensions `ratio_positive_reviews` and `elapsed_days_restaurant`. This discrepancy in the scalar approximations raises concerns about the consistency and reliability of the model. To ensure accurate and reliable modelling results, it is crucial to address this issue in future research and develop improved techniques for generating consistent scalar approximations for all dimensions.

Overall, while the RSQTOA with Grid Sampler approach shows promise in terms of performance and training time, it is essential to consider the computational cost of grid preparation and address the issue of inconsistent scalar approximations to enhance the reliability and scalability of the approach.

5 Conclusion

5.1 Summary & Discussion

In this study, we introduce a framework based on the concept of *Quasi-Separability* to reduce the dimensionality of multivariate scalar functions. The framework identifies dimensions that exhibit *quasi-Taylor*, *quasi-Periodic*, and *quasi-Exponential* behaviour in the domain or subdomain. It approximates the non-separable part of the function using machine learning techniques, particularly Artificial Neural Networks. By combining both reducible and non-reducible part approximations, the framework produces approximate models with reasonable accuracy, but with significantly less computation time compared to Artificial Neural Networks, as demonstrated in Märtens [2022].

To enhance the capability of the framework to handle datasets, we present two alternative approaches. These approaches make the core of our framework agnostic of the underlying representation of the relationship, whether it is a dataset or a mathematical function. However, when dealing with datasets, the target value for sampled data points may not be available. To address this, we propose two approaches: an interpolator-based method and a grid-based sampling technique. The interpolator-based approach performs well for smaller and simpler datasets, while the grid-based approach excels for both smaller and larger/complex datasets and is faster on average.

We test these approaches on two different datasets: a synthetic dataset and a subset of the Yelp academic dataset. Through a 20-fold cross-validation technique, we evaluate the performance of four different models: ElasticNet, Artificial Neural Network, RSQTOA with Interpolator, and RSQTOA with Grid Sampler.

The results indicate that the interpolator-based approach outperforms the grid sampling-based approach for simpler datasets, but its performance diminishes for larger and more complex datasets. On the other hand, the grid-based approach yields results comparable to Artificial Neural Networks for both simple and complex datasets, and it is twice as fast on average. However, the size of the grid and computation time become critical factors when dealing with larger datasets and higher dimensions.

Moreover, we observed some inconsistencies in the framework's ability to identify *quasi-separable* dimensions, highlighting the need for further investigation and refinement of the techniques used.

Furthermore, our framework achieves results comparable to those obtained with linear regression (specifically ElasticNet) in terms of predictive performance. However, detailed studies are required to compare the two approaches comprehensively, especially when dealing with complex and high-dimensional datasets.

In conclusion, this study provides valuable insights into the strengths and limitations of the RSQTOA framework. While it shows promise in identifying *quasi-separable* dimensions and achieving comparable results to linear regression models, challenges related to grid creation and consistent identification of *quasi-separable* dimensions need to be addressed.

5.2 Further Research

There are several avenues for enhancing and expanding the capabilities of the proposed RSQ-TOA framework. These areas are essential for refining the framework and making it more versatile, adaptive, and efficient in handling diverse datasets and complex relationships.

One fundamental aspect that requires further exploration is enabling the framework to identify *quasi-separable* dimensions not only for the entire domain but also within subdomains. Currently, the framework focuses on identifying *quasi-separable* dimensions on a domain level, but extending this capability to subdomains would significantly enhance its adaptability and applicability to more complex and heterogeneous datasets.

Another promising direction for future research involves exploring complex interpolation techniques based on radial basis functions. These techniques have shown great potential in capturing intricate relationships and patterns in data. By incorporating such methods into our framework, we can potentially achieve more accurate and robust approximate models, especially when dealing with datasets that exhibit complex behaviors.

A more sophisticated modelling approach involving n-dimensional models with hyperplanes should be considered. The current framework utilizes arbitrary ansatz functions and processes one dimension at a time for identifying *quasi-separable* dimensions. Adopting hyperplanes would enable the simultaneous consideration of interactions and dependencies among multiple dimensions, leading to more comprehensive and precise approximations.

To address the computational bottleneck caused by the grid creation process, the implementation of parallelization techniques is crucial. Parallelizing the grid computation would significantly reduce processing time, particularly for larger datasets and higher dimensions, thereby improving the framework's efficiency and scalability.

Additionally, the observed inconsistency in identifying *quasi-separable* dimensions warrants careful investigation and refinement. Further research is necessary to improve the reliability and consistency of the framework's performance in accurately identifying *quasi-separable* dimensions.

In conclusion, these potential areas of future work hold significant promise for advancing the effectiveness and efficiency of the RSQTOA framework. By focusing on extending the framework's capabilities to subdomains, exploring complex interpolation techniques, adopting hyperplanes for n-dimensional modelling, optimizing grid computation through parallelization, and addressing the issue of inconsistency, we can elevate the framework's utility and applicability, making it a more powerful tool for dimensionality reduction and modelling in various real-world applications.

5.3 Additional Resources

In addition to the content presented in this study, supplementary resources are available to enhance the understanding and utilization of the research findings. These resources include the codebase, data, analysis, and evaluation materials that were employed throughout the study.

The GitHub repository contains the entire codebase developed for this thesis, including the implementation of the proposed framework, data preprocessing scripts, and evaluation scripts. Additionally, the repository includes comprehensive documentation and instructions on how to reproduce the experiments conducted in this study.

To access these additional resources, please visit the following GitHub repository:
https://github.com/STCE-at-RWTH/RSQTOA

These resources aim to foster transparency, collaboration, and the advancement of research in the field. Researchers and practitioners are encouraged to utilize these materials to replicate, validate, and build upon the findings presented in this study.

V References

Jian An and Art Owen. Quasi-regression. *Journal of Complexity*, 17(4):588–607, 2001. ISSN 0885-064X. doi: https://doi.org/10.1006/jcom.2001.0588. URL https://www.sciencedirect.com/science/article/pii/S0885064X01905886.

Hyejin Baek, Joonbum Ahn, and YoonJung Choi. Helpfulness of online consumer reviews: Readersâ objectives and review cues. *International Journal of Electronic Commerce*, 17(2): 99–126, 2012.

Richard E. Bellman. *Adaptive Control Processes*. Princeton University Press, Princeton, 1961. ISBN 9781400874668. doi: doi:10.1515/9781400874668. URL https://doi.org/10.1515/9781400874668.

Yoshua Bengio. Scaling learning algorithms towards ai. *Large-scale kernel machines*, pages 1–34, 2007.

Mohamed Amine Bouhlel, John T. Hwang, Nathalie Bartoli, Rémi Lafage, Joseph Morlier, and Joaquim R.R.A. Martins. A python surrogate modeling framework with derivatives. *Advances in Engineering Software*, 135:102662, 2019. ISSN 0965-9978. doi: https://doi.org/10.1016/j.advengsoft.2019.03.005. URL https://www.sciencedirect.com/science/article/pii/S0965997818309360.

A. Brown and B. Lee. Enhancing efficiency in social media platforms: The role of surrogate models. *Journal of Social Media Analytics*, 5(2):125–138, 2018.

Sebastian F. Christodoulou. Approximartion of expensive functions through neural networks. Bachelor's thesis, RWTH Aachen University, 2020.

Alton YK Chua and Syagnik Banerjee. Helpfulness of user-generated reviews as a function of review sentiment, product type and information quality. *Computers in Human Behavior*, 54:547–554, 2016. ISSN 0747-5632. doi: https://doi.org/10.1016/j.chb.2015.08.057. URL https://www.sciencedirect.com/science/article/pii/S074756321530131X.

Charles Chui and Harvey Diamond. A natural formulation of quasi-interpolation by multivariate splines. *Proceedings of The American Mathematical Society - PROC AMER MATH SOC*, 99:643–643, 04 1987. doi: 10.1090/S0002-9939-1987-0877032-6.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus), 2016.

Jens Deussen. *Global derivatives*. Dissertation, RWTH Aachen University, Aachen, 2021. URL https://publications.rwth-aachen.de/record/837052. Veröffentlicht auf dem Publikationsserver der RWTH Aachen University 2022; Dissertation, RWTH Aachen University, 2021.

Yang Gao, Yi-Fan Li, Yu Lin, Hang Gao, and Latifur Khan. Deep learning on knowledge graph for recommender system: A survey. *CoRR*, abs/2004.00387, 2020. URL https://arxiv.org/abs/2004.00387.

Anindya Ghose and Panagiotis G Ipeirotis. Estimating the helpfulness and economic impact of product reviews: mining text and reviewer characteristics. *IEEE Transactions on Knowledge and Data Engineering*, 23(10):1498–1512, 2011.

Eric Gilbert and Karrie Karahalios. Understanding deja reviewers. In *Proceedings of the 2010 ACM conference on Computer Supported Cooperative Work*, pages 225–228. ACM, 2010.

Charles R. Harris et al. Array programming with numpy. *Nature*, 585(7825):357–362, 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

Hua Hong, Dongwon Xu, Guan Ang Wang, and Wei Fan. Understanding the determinants of online review helpfulness: A meta-analytic investigation. *Decision Support Systems*, 102:1–11, 2017. ISSN 0167-9236. doi: https://doi.org/10.1016/j.dss.2017.06.007. URL https://www.sciencedirect.com/science/article/pii/S0167923617301197.

Xiaodong Huan. pydoe2. https://github.com/clicumu/pyDOE2, 2021.

C. Jones et al. Refining surrogate models in social media platforms: A feedback loop approach. In *Proceedings of the International Conference on Social Media Analysis*, pages 87–94, 2020.

Gero F. Kauerauf. Randomized subspace quasi-linearity analysis. Bachelor's thesis, RWTH Aachen University, 2021.

Soo-Min Kim, Patrick Pantel, Timothy Chklovski, and Marco Pennacchiotti. Automatically assessing review helpfulness. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 423–430. Association for Computational Linguistics, 2006.

Woo Gon Kim, Jing Jian Li, and Robert A Brymer. The impact of social media reviews on restaurant performance: The moderating role of excellence certificate. *International Journal of Hospitality Management*, 55:41–51, 2016. ISSN 0278-4319. doi: https://doi.org/10.1016/j.ijhm.2016.03.001. URL https://www.sciencedirect.com/science/article/pii/S0278431916300184.

Leung Kwok and Karen L Xie. Factors contributing to the helpfulness of online hotel reviews: Does manager response play a role? *International Journal of Hospitality Management*, 28: 2156–2177, 2016. doi: 10.1108/IJCHM-03-2015-0107.

Bong Chan Kwon, Seong-Hoon Kim, Timothy Duket, Ana Catalán, and Ji Soo Yi. Do people really experience information overload while reading online reviews? *International Journal of Human-Computer Interaction*, 31(12):959–973, 2015.

H. Li, C.R. Wang, F. Meng, and Z. Zhang. Making restaurant reviews useful and/or enjoyable? the impacts of temporal, explanatory, and sensory cues. *International Journal of Hospitality Management*, 71:1–12, 2018. doi: 10.1016/j.ijhm.2017.11.003.

Mengxiang Li, Li Huang, Chee-Wee Tan, and Keli Wei. Helpfulness of online product reviews as seen by consumers: Source and content features. *International Journal of Electronic Commerce*, 17(1):101–136, 2013.

Zheng Liu and Sangwon Park. What makes a useful online review? implication for travel product websites. *Tourism Management*, 47:140–151, 2015. ISSN 0261-5177. doi: https://doi.org/10.1016/j.tourman.2014.09.020. URL https://www.sciencedirect.com/science/article/pii/S0261517714001903.

Yan Luo and Rong-Lin Tang. Understanding hidden dimensions in textual reviews on Airbnb: An application of modified latent aspect rating analysis (LARA). *International Journal of Hospitality Management*, 80:144–154, 2019. ISSN 0278-4319. doi: https://doi.org/10.

1016/j.ijhm.2019.02.008. URL https://www.sciencedirect.com/science/article/pii/S0278431918306856.

Naresh K Malhotra. Reflections on the information overload paradigm in consumer decision making. *Journal of Consumer Research*, 10(3):436–440, 1984.

Saba Malik and Abid Hussain. Helpfulness of product reviews as a function of discrete positive and negative emotions. *Computers in Human Behavior*, 73:290–302, 2017. doi: 10.1016/j.chb.2017.03.053.

Simon L. Märtens. Randomized subspace quasi taylor-order analysis. Master's thesis, RWTH Aachen University, 2022.

Aurelio G Mauri and Roberta Minazzi. Web reviews influence on expectations and purchasing intentions of hotel potential customers. *International Journal of Hospitality Management*, 34:99–107, 2013. ISSN 0278-4319. doi: https://doi.org/10.1016/j.ijhm.2013.02.012. URL https://www.sciencedirect.com/science/article/pii/S0278431913000236.

Susan M Mudambi and David Schuff. What makes a helpful online review? a study of customer reviews on amazon.com. *MIS Quarterly*, 34(1):185–200, 2010.

Thomas L. Ngo-Ye and Atish P. Sinha. The influence of reviewer engagement characteristics on online review helpfulness: A text regression model. *Decision Support Systems*, 61:47–58, 2014. ISSN 0167-9236. doi: https://doi.org/10.1016/j.dss.2014.01.011. URL https://www.sciencedirect.com/science/article/pii/S0167923614000128.

Art B. Owen. Assessing linearity in high dimensions. *The Annals of Statistics*, 28(1):1 – 19, 2000. doi: 10.1214/aos/1016120362. URL https://doi.org/10.1214/aos/1016120362.

Yue Pan and Jia Qing Zhang. Born unequal: A study of the helpfulness of user-generated product reviews. *Journal of Retailing*, 87(5):598–612, 2011. ISSN 0022-4359. doi: https://doi.org/10.1016/j.jretai.2011.05.002. URL https://www.sciencedirect.com/science/article/pii/S0022435911000406.

Ioannis S Pantelidis. Electronic meal experience: A content analysis of online restaurant comments. *Cornell Hospitality Quarterly*, 51(4):483–491, 2010. doi: 10.1177/1938965510378574. URL https://doi.org/10.1177/1938965510378574.

Pradeep Racherla and Warren Friske. Perceived âusefulnessâ of online consumer reviews: An exploratory investigation across three services categories. *Electronic Commerce Research and Applications*, 11(6):548–559, 2012. ISSN 1567-4223. doi: https://doi.org/10.1016/j.elerap.2012.06.003. URL https://www.sciencedirect.com/science/article/pii/S1567422312000464. Information Services in EC.

J. Smith et al. Surrogate models for personalized content recommendations in social media platforms. *Journal of Computational Social Science*, 12(3):201–216, 2019.

I.M Sobol. Global sensitivity indices for nonlinear mathematical models and their monte carlo estimates. *Mathematics and Computers in Simulation*, 55(1):271–280, 2001.

Statista. Cumulative number of reviews submitted to yelp from 2004 to 2021 (in millions), 2021. URL https://www.statista.com/statistics/278032/cumulative-number-of-reviews-submitted-to-yelp/.

Michael Stein. Large sample properties of simulations using latin hypercube sampling. *Technometrics*, 29(2):143–151, 1987.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, Ílhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. Scipy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, 17(3):261–272, 2020. doi: 10.1038/s41592-019-0686-2. URL https://doi.org/10.1038/s41592-019-0686-2.

G. Vlachos. Online travel statistics. Infographics Mania, 2012. URL http://infographicsmania.com/online-travel-statistics-2012/.

Xiaofan Wang, Lijun Ryan Tang, and Eunseon Kim. More than words: Do emotional content and linguistic style matching matter on restaurant review helpfulness? *International Journal of Hospitality Management*, 77:438–447, 2019. ISSN 0278-4319. doi: https://doi.org/10.1016/j.ijhm.2018.08.007. URL https://www.sciencedirect.com/science/article/pii/S0278431918300458.

Xiaohui Yan, Jing Wang, and Michael Chau. Customer revisit intention to restaurants: Evidence from online reviews. *Information Systems Frontiers*, 17(3):645–657, 2015. doi: 10.1007/s10796-013-9446-5. URL https://doi.org/10.1007/s10796-013-9446-5.

Xiao Zhang, Simon Du, and Quanquan Gu. Fast and sample efficient inductive matrix completion via multi-phase procrustes flow. In *International Conference on Machine Learning*, pages 5756–5765. PMLR, 2018.

Zhe Zhang, Qing Ye, Rob Law, and Yiyang Li. The impact of e-word-of-mouth on the online popularity of restaurants: A comparison of consumer reviews and editor reviews. *International Journal of Hospitality Management*, 29(4):694–700, 2010. ISSN 0278-4319. doi: https://doi.org/10.1016/j.ijhm.2010.02.002. URL https://www.sciencedirect.com/science/article/pii/S0278431910000198.

Shasha Zhou and Bin Guo. The order effect on online review helpfulness: A social influence perspective. *Decision Support Systems*, 93:77–87, 2017. ISSN 0167-9236. doi: https://doi.org/10.1016/j.dss.2016.09.016. URL https://www.sciencedirect.com/science/article/pii/S0167923616301658.

Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

A Appendix

A.1 YAML Configuration File

```yaml
# General Options

# Specifies the directory where the generated files will be saved
output_dir: ~

# Flag to enable or disable debug mode
debug: True



# RSQTOA hyperparameters

# The threshold for the overall error in the approximation
epsilon: 10e-5

# The strategy for sampling points in the domain
# Possible values: random, latin, latin_maximin, latin_corr, grid
sampling_strategy: "grid"

# The number of points used for subspace regression
regression_samples: 20

# The number of points used for testing the regression in the entire domain
test_samples: 20

# The number of values tested with the test points
test_values: 20

# The maximum Taylor order for regression
max_taylor_order: 2

# The number of samples used to improve the Taylor regression
# Minimum number of samples is used to ensure the least squares
# problem is not underdetermined
taylor_regression_samples: 10

# The number of samples used to improve the periodic regression
# Set to zero to disable periodic regression
periodic_regression_samples: 20

# The number of samples used for exponential regression
# Set to zero to disable exponential regression
exponential_regression_samples: 20

# The strategy for selecting the regression method
# Possible values: greedy, best
regression_selection: "best"
```

Code Listing 5: RSQTOA Framework Hyperparameters YAML configuration file

A.2 Source Code

```python
class Model(ABC):
    """
    Abstract base class for models.

    Attributes:
    - dims (int): Number of dimensions in the model.
    - lower (List[int]): List of lower bounds for each dimension.
    - upper (List[int]): List of upper bounds for each dimension.
    """

    def __init__(self, dims: int, lower: List[int], upper: List[int]):
        """
        Initialize the Model class.

        Args:
        - dims (int): Number of dimensions in the model.
        - lower (List[int]): List of lower bounds for each dimension.
        - upper (List[int]): List of upper bounds for each dimension.
        """
        self.dims = dims
        self.lower = lower
        self.upper = upper

    @abstractmethod
    def get_target(self, feature: List[List[Union[float, int]]]) -> Union[float, int]:
        """
        Abstract method to get the target value for a given feature.

        Args:
        - feature (List[List[Union[float, int]]]): Input feature.

        Returns:
        - Union[float, int]: Target value for the input feature.
        """
        pass
```

Code Listing 6: Model class

```python
class EquationModel(Model):
    """
    Represents a model for an equation-based function.
    Inherits from the Model class.
    """

    def __init__(self, equation: Callable, dims: List[float], lower: List[float],
                                                upper: List[float]):
        """
        Initializes an EquationModel object.

        Parameters:
        - equation (Callable): The equation function to be modeled.
        - dims (List[float]): List of dimension values.
        - lower (List[float]): List of lower bounds for each dimension.
        - upper (List[float]): List of upper bounds for each dimension.
        """
        self.equation = equation
        Model.__init__(self, dims, lower, upper)

    def get_target(self, feature: List[List[Union[float, int]]]) -> Union[float, int]:
        """
        Retrieves the target value for a given feature using the equation/function
                                                    passed.

        Parameters:
        - feature (List[List[Union[float, int]]]): The input feature.

        Returns:
        - Union[float, int]: The target value calculated based on the equation.
        """
        return self.equation(feature)
```

Code Listing 7: EquationModel class

```python
class DatasetModel(Model):
    """
    A class representing a dataset-based model.

    Inherits from the Model class.

    Parameters:
    - features (numpy.ndarray): The input features of the dataset.
    - target (array-like): The target values of the dataset.
    - lower (List[int]): The lower bounds of the input features.
    - upper (List[int]): The upper bounds of the input features.
    """

    def __init__(self, features: NDArray, target: ArrayLike, lower: List[int], upper:
                                                List[int]):
        """
        Initialize the DatasetModel.

        Args:
        - features (numpy.ndarray): The input features of the dataset.
        - target (array-like): The target values of the dataset.
        - lower (List[int]): The lower bounds of the input features.
        - upper (List[int]): The upper bounds of the input features.
        """
        self.features = features
        self.target = target

        Model.__init__(self, features.shape[1], lower, upper)

    def get_target(self, features: List[List[Union[float, int]]]) -> Union[float, int]
                                                :
        """
        Get the target value for the given input features.

        Args:
        - features (List[List[Union[float, int]]]): The input features for which
         to retrieve the target value.

        Returns:
        - Union[float, int]: The corresponding target value.
        """
        if any(isinstance(el, list) | (type(el) is np.ndarray) for el in features):
            targets = []
            for feature in features.T:
                targets.append(
                    self.target[
                        Counter(np.where(self.features == feature)[0])
                            .most_common(1)[0][0]
                    ]
                )
            return targets
        else:
            return self.target[
                Counter(np.where(self.features == features)[0])
                    .most_common(1)[0][0]
            ]
```

Code Listing 8: DatasetModel class

```python
class Sampler(ABC):
    """
    Base class for samplers.

    Args:
        model (Model): The model to sample data from.
        dims (int): The number of dimensions.
        lower (List[float]): The lower bounds of the domain.
        upper (List[float]): The upper bounds of the domain.
    """

    def __init__(self, model: Model, dims: int, lower: List[float], upper: List[float]
                                                ):
        """
        Constructor of the sampler base.

        Args:
            model (Model): The model to sample data from.
            dims (int): The number of dimensions.
            lower (List[float]): The lower bounds of the domain.
            upper (List[float]): The upper bounds of the domain.
        """
        self.model = model
        self.dims = dims
        self.lower = lower.astype(float)
        self.upper = upper.astype(float)

    @abstractmethod
    def sample_dimension(self, n: int, dim: int, anchor: List[float]) -> Tuple[np.
                                                ndarray, np.ndarray]:
        """
        Sample points in a specific dimension.

        Args:
            n (int): The number of samples to generate.
            dim (int): The dimension to sample.
            anchor (List[float]): The anchor point to fix rest of dimension values

        Returns:
            Tuple[np.ndarray, np.ndarray]: The sampled points and their corresponding
                                                        function values.
        """
        pass

    @abstractmethod
    def sample_domain(self, n: int, dim: int = -1) -> Tuple[np.ndarray, np.ndarray]:
        """
        Sample points in the entire domain.

        Args:
            n (int): The number of samples to generate.
            dim (int, optional): The specific dimension to sample. Defaults to -1,
                                                        which means all dimensions.

        Returns:
            Tuple[np.ndarray, np.ndarray]: The sampled points and their corresponding
                                                        function values.
        """
        pass
```

Code Listing 9: Sampler class

```python
class DomainSampler(Sampler):
    def __init__(self, model: Model, dims: int, lower: List[float], upper: List[float]):
        """
        Constructor for the DomainSampler class.

        Args:
            model (Model): The model used for evaluation.
            dims (int): The number of dimensions in the domain.
            lower (List[float]): The lower bounds for each dimension.
            upper (List[float]): The upper bounds for each dimension.
        """
        Sampler.__init__(self, model, dims, lower, upper)

    def eval_samples(self) -> np.array:
        """
        Evaluate the current set of samples.

        Returns:
            np.array: The evaluated target values for the samples.
        """
        if self.model is not None and self.samples is not None:
            n = np.shape(self.samples)[0]
            Y = np.empty(n)
            for i in range(n):
                Y[i] = self.model.get_target(self.samples[i])
        else:
            raise RuntimeError("Either model or samples are not defined.")

        return Y

    def sample_dimension(self, n: int, dim: int, anchor: List[float]) -> Tuple[np.ndarray, np.array]:
        """
        Sample in one dimension from an anchor in the domain.

        Args:
            n (int): The number of samples to generate.
            dim (int): The dimension to sample from.
            anchor (List[float]): The anchor point in the domain.

        Returns:
            Tuple[np.ndarray, np.array]: The generated samples and their corresponding target values.
        """
        X = np.empty((n, self.dims))

        # Broadcast anchor point and sample in the given dimension
        X[:] = np.copy(anchor)
        X[:, dim] = self.lower[dim]
        X[:, dim] += np.random.rand(n) * (self.upper[dim] - self.lower[dim])

        return X, self.eval_samples()
```

Code Listing 10: DomainSampler class Märtens [2022]

```python
class RandomSampler(DomainSampler):
    def __init__(self, model: Model, dims: int, lower: List[float], upper: List[float]
                                                ):
        """
        Constructor of the Random Sampler class.

        Args:
            model (Model): The model used for sampling.
            dims (int): The number of dimensions in the domain.
            lower (List[float]): The lower bounds of the domain for each dimension.
            upper (List[float]): The upper bounds of the domain for each dimension.
        """
        DomainSampler.__init__(self, model, dims, lower, upper)

    def sample_domain(self, n: int, dim: int = -1) -> Tuple[np.ndarray, np.array]:
        """
        Generate random samples in the domain.

        Args:
            n (int): The number of samples to generate.
            dim (int, optional): The dimension to sample in. Defaults to -1.

        Returns:
            Tuple[np.ndarray, np.array]: A tuple containing the generated samples and
                                                        their corresponding evaluations
                                                        .
        """
        X = np.random.rand(n, self.dims)
        X = self.lower + np.multiply(X, self.upper - self.lower)

        self.samples = np.copy(X)
        return X, self.eval_samples()
```

Code Listing 11: RandomSampler class Märtens [2022]

```python
class LatinHypercubeSampler(DomainSampler):
    def __init__(self, model: Model, dims: int, lower: List[float], upper: List[float]
                                                ):
        """
        Constructor of the Latin Hypercube Sampler.

        Args:
        - model (Model): The model used for sampling.
        - dims (int): The number of dimensions in the domain.
        - lower (List[float]): The lower bounds of the domain for each dimension.
        - upper (List[float]): The upper bounds of the domain for each dimension.
        """
        DomainSampler.__init__(self, model, dims, lower, upper)

    def sample_domain(self, n: int, dim: int = -1) -> tuple[np.ndarray, np.array]:
        """
        Samples the domain using Latin Hypercube Sampling.

        Args:
        - n (int): The number of samples to generate.
        - dim (int): The dimension index to sample (default: -1, sample all dimensions
                                                    ).

        Returns:
        - Tuple[np.ndarray, np.array]: A tuple containing the sampled points and their
                                                     corresponding function values.
        """
        X = lhs(self.dims, samples=n)
        X = self.lower + np.multiply(X, self.upper - self.lower)

        self.samples = np.copy(X)
        return X, self.eval_samples()
```

Code Listing 12: LatinHypercubeSampler class Märtens [2022]

```python
class DataSampler(Sampler):
    def __init__(self, model: Model, dims: int, lower: List[float], upper: List[float]
                                                ):
        """
        Initialize the DataSampler class.

        Args:
            model (Model): The model used for sampling.
            dims (int): The number of dimensions.
            lower (List[float]): The lower bounds of the sampling domain.
            upper (List[float]): The upper bounds of the sampling domain.
        """
        Sampler.__init__(self, model, dims, lower, upper)
```

Code Listing 13: DataSampler class

```python
class GridSampler(DataSampler):
    def __init__(self, model: DatasetModel, partitions: int, threshold: int,
                 dims: int, lower: List[float], upper: List[float]):
        """
        GridSampler constructor.
        Args:
            model (DatasetModel): The dataset model.
            partitions (int): Number of partitions for the grid.
            threshold (int): Threshold for the minimum number of elements in each grid section.
            dims (int): Number of dimensions in the dataset.
            lower (List[float]): Lower bounds for each dimension.
            upper (List[float]): Upper bounds for each dimension.
        """
        self.partitions = partitions
        self.threshold = threshold
        self.grid = self.__prepare_grid()

        DataSampler.__init__(self, model, dims, lower, upper)

    def __prepare_grid(self):
        """
        Prepare the grid for sampling.
        Returns:
            dict: The prepared grid.
        """
        grids = {}

        # Calculate grid sections for each variable of interest (voi)
        for col_idx in range(self.model.features.shape[1]):
            grids[col_idx] = {}
            for projection_column in set(
                    range(self.model.features.shape[1])).copy().difference(
                {col_idx}):
                min_val, max_val = self.model.features[:,
                                    projection_column].min(
                    axis=0), self.model.features[:, projection_column].max(
                    axis=0)
                min_val -= 1e-3
                max_val += 1e-3
                grids[col_idx][projection_column] = [
                    min_val + (i * (max_val - min_val) / self.partitions) for i
                    in range(self.partitions + 1)]

        # Generate coordinates for each grid section
        for voi in grids:
            mesh = np.meshgrid(*[grids[voi][var] for var in grids[voi].keys()])
            coordinates = list(np.reshape(
                                list(zip(*[x.flat for x in mesh])),
                                np.roll(np.shape(mesh), -1)))
            rolled_coordinates = np.roll(
                coordinates, np.full(np.shape(coordinates)[-1], -1),
                axis=list(range(np.shape(coordinates)[-1])))
            diag_coordinates = np.concatenate(
                [coordinates, rolled_coordinates],
                axis=np.shape(coordinates)[-1])

            filtered_coordinates = diag_coordinates.copy()
            for i in range(len(np.shape(filtered_coordinates)) - 1):
                filtered_coordinates = np.delete(
                    filtered_coordinates, -1, axis=i)
            filtered_coordinates = filtered_coordinates.reshape(
                -1, filtered_coordinates.shape[-1])
            grids[voi]['bounds'] = filtered_coordinates

        # Determine valid elements in each grid section
        for col_idx in range(self.model.features.shape[1]):
            features_subset = np.delete(self.model.features, [col_idx], axis=1)
            bounds = np.array(grids[col_idx]['bounds'])

            mapping = {}
```

```python
            valid_elements = []

            for index, bound in enumerate(bounds):
                allInBounds = None
                for i in range(features_subset.shape[1]):
                    eval = ((features_subset[:, i] >= bound[i]) & (
                            features_subset[:, i] <= bound[
                        i + features_subset.shape[1]]))
                    allInBounds = allInBounds & eval if allInBounds is not None else
                                                                        eval

                elements = np.argwhere(allInBounds)

                if len(elements) >= self.threshold:
                    rows = list(itertools.chain.from_iterable(elements))
                    valid_elements.extend(rows)
                    mapping[tuple(bound)] = rows

            grids[col_idx]['mapping'] = mapping
            grids[col_idx]['valid_elements'] = list(set(valid_elements))

        return grids

    def sample_dimension(self, n: int, dim: int, anchor: List[float]) -> Tuple[
        np.ndarray, np.array]:
        """
        Sample data points for a specific dimension.
        Args:
            n (int): Number of samples to generate.
            dim (int): Dimension to sample.
            anchor (List[float]): Anchor point for sampling.

        Returns:
            Tuple[np.ndarray, np.array]: Sampled features and target values.
        """
        anchor_index = \
            Counter(np.where(self.model.features == anchor)[0]).most_common(1)[
                0][0]
        sample_indices = []
        sample_grid_section_anchor = []
        for key, indices in self.grid[dim]['mapping'].items():
            if anchor_index in set(indices):
                sample_grid_section_anchor = [
                    np.mean([key[i], key[i + (int(len(key) / 2))]]) for i in
                    range(int(len(key) / 2))]
                sample_indices = random.sample(indices, n)
                break
        feature_samples = self.model.features[sample_indices]
        feature_samples[:,
        [i for i in range(self.dims) if i != dim]] = sample_grid_section_anchor
        return feature_samples, self.model.target[sample_indices]

    def sample_domain(self, n: int, dim: int = -1) -> Tuple[
        np.ndarray, np.array]:
        """
        Sample data points from the entire domain or a specific dimension.
        Args:
            n (int): Number of samples to generate.
            dim (int): Dimension to sample (optional).

        Returns:
            Tuple[np.ndarray, np.array]: Sampled features and target values.
        """
        domain_samples = random.sample(range(len(self.features)), n)
        if dim != -1:
            valid_element_indices = self.grid[dim]['valid_elements']
            domain_samples = random.sample(valid_element_indices, n)
        return self.features[domain_samples], self.target[domain_samples]
```

Code Listing 14: GridSampler class

```python
class QuasiSeparabilityRegression(ABC):
    """
    Base class for Quasi-Separability Regression.

    Parameters:
    - model: The underlying model used for regression.
    - dims: The number of dimensions in the data.
    - lower: Lower bounds of the domain in each dimension.
    - upper: Upper bounds of the domain in each dimension.
    - regression_dim: The dimension on which regression is performed.
    """
    def __init__(self, model, dims, lower, upper, regression_dim):
        self.model = model
        self.dims = dims
        self.lower = lower.astype(float)
        self.upper = upper.astype(float)
        self.regression_dim = regression_dim
        self.coefficients = None
        self.max_error = 0.0
        self.sample_value = 0.0

        # Initialize sampler based on the chosen strategy in config
        if config.sampling_strategy == 'random':
            self.sampler = RandomSampler(model, dims, lower, upper)
        elif config.sampling_strategy == 'dataset':
            self.sampler = DatasetSampler(model, dims, lower, upper)
        else:
            self.sampler = LatinHypercubeSampler(model, dims, lower, upper)

    def derivative_at(self, point):
        """
        Returns the partial derivative at a given sample point.

        Parameters:
        - point: The sample point for which the partial derivative is calculated.

        Returns:
        The partial derivative at the given sample point.
        """
        return _fd_derivative(self.model, self.lower, self.upper, self.regression_dim,
                                                point)

    def fit(self):
        """
        Tries to fit the regression model over a subspace of the domain.

        This method performs subspace regression at random samples and takes the
                                                average of coefficients.
        It then evaluates the regression model on test points and calculates the
                                                maximal error.
        """
        # Get random samples in the entire domain
        X, y = self.sampler.sample_domain(config.regression_samples, self.
                                                regression_dim)

        # Perform subspace regression at samples and take the average of coefficients
        try:
            self.coefficients = np.zeros_like(self.coefficients)
            for X_sample in X:
                self.coefficients += self.fit_at_sample(X_sample)
            self.coefficients /= config.regression_samples
        except KeyboardInterrupt:
            sys.exit()

        # Quick return in case of infinite or NaN coefficients
        if any(np.isnan(self.coefficients)) or any(np.isinf(self.coefficients)):
            self.max_error = np.Inf
            return

        # Get random test points in the entire domain (N=config.test_samples)
```

```python
        X_test, y_test = self.sampler.sample_domain(config.test_samples, self.regression_dim)

        # Errors
        err_matrix = np.zeros((config.test_samples, config.test_values))

        for i in range(config.test_samples):
            # Evaluate regression model with x_i from the test point
            y_test_reg = self(X_test[i][self.regression_dim])

            # For each test point, sample a certain number of points in dimension x_i
            # with the test point as the anchor
            X_tilde, y_tilde = self.sampler.sample_dimension(config.test_values, self.regression_dim, X_test[i])

            for j in range(config.test_values):
                # Evaluate regression model with the additionally sampled x_i
                y_tilde_reg = self(X_tilde[j][self.regression_dim])

                # Calculate error (see Quasi-Separability definition)
                err_matrix[i, j] = abs(y_test_reg + y_tilde[j] - y_tilde_reg - y_test[i])

        # Calculate maximal error and choose a sample value for the model reduction
        self.max_error = err_matrix.max()
        self.sample_value = X_tilde[err_matrix.max(0).argmin()][self.regression_dim]

    def validate(self, current_epsilon):
        """
        Validates the maximal error against the given error threshold.

        Parameters:
        - current_epsilon: The current error threshold.

        Returns:
        True if the maximal error is less than the error threshold, False otherwise.
        """
        success = self.max_error < current_epsilon
        if success and config.debug:
            print("success: max error (" + str(self.max_error) + ") < epsilon (" + str(current_epsilon) + ")")
        elif config.debug:
            print("fail: max error (" + str(self.max_error) + ") > epsilon (" + str(current_epsilon) + ")")

        return success

    @abstractclassmethod
    def try_fit(self, eps):
        pass

    @abstractclassmethod
    def fit_at_sample(self, sample):
        pass

    @abstractmethod
    def __call__(self, x):
        pass

    @abstractmethod
    def __str__(self):
        pass
```

Code Listing 15: Base class for *Quasi-Separability* detection with subspace regression Märtens [2022]

```python
class QuasiTaylorRegression(QuasiSeparabilityRegression):
    def __init__(self, model, dims, lower, upper, regression_dim):
        """
        Constructor of the Taylor subspace regression model

        Args:
            model: The underlying model for regression
            dims: The dimensions of the function domain
            lower: The lower bound of the domain
            upper: The upper bound of the domain
            regression_dim: The dimension for regression
        """
        QuasiSeparabilityRegression.__init__(self, model, dims, lower, upper,
                                             regression_dim)
        self.order = 1
        self.coefficients = np.empty(self.order)

    def try_fit(self, eps):
        """
        Applies the Taylor regression in one dimension

        Args:
            eps: The overall error threshold for the approximation

        Returns:
            True if the regression is successful, False otherwise
        """
        # Try up to the maximal Taylor order. Return immediately on success.
        for p in range(1, config.max_taylor_order + 1):
            self.order = p
            self.coefficients = np.empty(self.order)

            if config.debug:
                print("Dimension " + str(self.regression_dim) + " Taylor (order = " +
                                                        str(p) + ") regression ...
                                                        ", flush=True)

            # Tries to fit the current regression config
            self.fit()
            if self.validate(eps):
                return True

        return False

    def fit_at_sample(self, sample):
        """
        Performs the regression over the subspace defined by the sample

        Args:
            sample: The sample point in the subspace

        Returns:
            The coefficients of the regression model
        """
        # Number of samples is at least as large as the number of coefficients
        n_coefficients = self.order + 1
        n_samples = n_coefficients + config.taylor_regression_samples
        sample = np.full(1, sample) if self.dims == 1 else sample

        # Sample random points in the subspace
        X, y = self.sampler.sample_dimension(n_samples, self.regression_dim, sample)

        # Compute regression matrix X (linear system)
        X_reg = np.empty((n_samples, n_coefficients))
        for i in range(n_samples):
            X_i = np.full(1, X[i]) if self.dims == 1 else X[i]
            for j in range(n_coefficients):
                X_reg[i][j] = X_i[self.regression_dim] ** j

        # Perform regression via normal equation
```

```
        coeffs = np.linalg.solve(np.matmul(X_reg.transpose(), X_reg), np.matmul(X_reg.
                                                    transpose(), y))

        # Return all coefficients apart from the constant term
        return coeffs[1:]

    def __call__(self, x):
        """
        Predicts the value at point x

        Args:
            x: The point at which to predict the value

        Returns:
            The predicted value at point x
        """
        y = 0
        for i in range(len(self.coefficients)):
            y += self.coefficients[i] * (x ** (i+1))

        return y

    def __str__(self):
        """
        Creates a string representation with possible line breaks

        Returns:
            The string representation of the regression model
        """
        code = ""
        for i in range(len(self.coefficients)):
            code += str(self.coefficients[i])
            if i == 0:
                code += " * x"
            else:
                code += " * std::pow(x, " + str(i+1) + ")"
            code += " +$lb$"

        return code[:-6]
```

Code Listing 16: Quasi-Taylor regression model Märtens [2022]

```python
class RsqtoaRegressor(Regressors):

    def __init__(self, config_path: str, model: Model, get_new_approximator_instance:
                                                Callable, validation_fraction: float =
                                                0.2, random_state: Union[int, None] =
                                                None):
        """
        Initialize the RSQTOA Regressor.

        Args:
            config_path (str): Path to the configuration file.
            model (Model): Model object representing the underlying function.
            get_new_approximator_instance (Callable): Callable that returns a new
                                                        instance of the approximator
                                                        for approximating non-reducible
                                                         part.
            validation_fraction (float): Fraction of the data to use for validation.
            random_state (Union[int, None]): Random state for reproducibility.
        """
        self._config = rsqtoa.create_config(cfg=config_path)
        self._model = model
        self._get_new_approximator_instance = get_new_approximator_instance

        Regressors.__init__(self=self, validation_fraction=validation_fraction,
                                                    random_state=random_state)

    def fit(self, features, target):
        """
        Fit the RSQTOA Regressor to the training data.

        Args:
            features (array-like): Training data features.
            target (array-like): Target values.

        """
        self._rsqtoa_tree = rsqtoa.create_subspace_tree(self._config, self._model)

        # Train all reduced models
        coeff, N_test, N_training = 0, 2000, 10000

        # Returns all subspaces
        leaves = self._rsqtoa_tree.get_leafs()
        coeff = sum([1 if len(leaf.non_separable_dims) > 0 else 0 for leaf in leaves])

        for leaf in leaves:
            if len(leaf.non_separable_dims) > 0:
                # Create samples in the subspace (training & test data)
                training_data_x, training_data_y = leaf.sample_domain(int(N_training /
                                                                    coeff))
                test_data_x, test_data_y = leaf.sample_domain(int(N_test / coeff))

                # Evaluate the reduced model at the training samples
                training_data_y = leaf.evaluate_reduced_model(np.transpose(
                                                            training_data_x))
                training_data_y = np.expand_dims(training_data_y, axis=1)

                # Evaluate the reduced model at the test samples
                test_data_y = leaf.evaluate_reduced_model(np.transpose(test_data_x))
                test_data_y = np.expand_dims(test_data_y, axis=1)

                # Normalize the data for training
                idx = rsqtoa.normalize_data(training_data_y, test_data_y)

                # Create reduced data set
                input_n = len(leaf.non_separable_dims)
                for i in reversed(range(leaf.dims)):
                    if i not in leaf.non_separable_dims:
                        training_data_x = np.delete(training_data_x, i, 1)
                        test_data_x = np.delete(test_data_x, i, 1)

                # Create tensorflow model
```

```python
                approximator = self._get_new_approximator_instance(input_dimension=
                                                                   input_n)

                # Train the tensorflow model
                loss_hist = rsqtoa.train_model(
                    approximator, training_data_x, training_data_y, test_data_x,
                                                                   test_data_y)
                self.loss_hist.append(loss_hist)

                # Set the trained tensorflow model in the subspace node
                leaf.approximated_model = approximator

                # Set the normalization index so that we can denormalize the results
                leaf.denormalization_idx = idx

        # Print some debug info
        self._rsqtoa_tree.print_info()

    def predict(self, features):
        """
        Predict the target values for the given features.

        Args:
            features (array-like): Input features for prediction.

        Returns:
            array-like: Predicted target values.
        """
        return self._rsqtoa_tree.evaluate_reassembled_model(features.T)
```

Code Listing 17: RSQTOA Regressor Class

A.3 Results

A.3.1 Synthetic Dataset: ElasticNet

fold	train time	train rmse	train r^2	test rmse	test r^2
1	0.124876022338867	0.00392924801047067	0.999990211561025	0.00408349057431563	0.999990342367524
2	0.13041090965271	0.0039441042263091	0.999990175473259	0.00386126700653568	0.999990813430075
3	0.133577823638916	0.00393195804339377	0.999990244497572	0.00406137986983329	0.99998966352483
4	0.125513792037964	0.00394551132284327	0.999990177778964	0.00397238353027467	0.999990093377195
5	0.120535612106323	0.00393884062487009	0.999990205836793	0.00364724874313453	0.999991738727382
6	0.140825033187866	0.00394285172176566	0.999990190997625	0.00413264694151038	0.999989288971441
7	0.131842136383057	0.00394622559686697	0.999990159831626	0.00412615858533264	0.99998960779212
8	0.138062238693237	0.0039353399804581	0.999990233179492	0.00384703352974772	0.999990625791357
9	0.125287055969238	0.00393206814798599	0.999990245162466	0.00398744153943297	0.999990016953429
10	0.126037359237671	0.00394200798618258	0.999990213779705	0.00370066155238813	0.999991070287711
11	0.116370916366577	0.00395043332519651	0.999990220439639	0.00388867602875133	0.999989102046619
12	0.127572059631348	0.0039409925375143	0.999990208378152	0.00407650532719442	0.999989407368558
13	0.120898723602295	0.00393467107559758	0.999990222096813	0.00387267241588501	0.99999076376602
14	0.124786853790283	0.00395795600363783	0.999990097199948	0.00451646288663334	0.999987588573366
15	0.118348121643066	0.00393366380763949	0.999990281007071	0.00380195008489028	0.999990079183896
16	0.129584789276123	0.00394846901932497	0.999990137381933	0.00388262464956116	0.999990955650791
17	0.126770973205566	0.00391745910385142	0.999990311123276	0.00375985455471803	0.999991211454428
18	0.148088932037354	0.00393916310657711	0.99999021206277	0.00404254983356085	0.999989688663232
19	0.120054721832275	0.00393349891804784	0.999990233735563	0.00411403691264569	0.9999894557645
20	0.146769762039185	0.00394003646547269	0.999990256043493	0.0035786819952933	0.999991084900281

Table 6: Results 20 fold cross-validation ElasticNet (Synthetic Data)

A.3.2 Synthetic Dataset: ANN

fold	train time	train rmse	train r^2	test rmse	test r^2
1	12.137307882309	0.017465422190927	0.990623758725492	0.0191032482643712	0.989752969737139
2	10.6550991535187	0.0116449455909248	0.995847918879126	0.0118662558516217	0.995793729772082
3	10.2439119815826	0.0106644100718352	0.996520782447318	0.0114223375184388	0.996036204081536
4	10.5639691352844	0.0109053745189297	0.996362019858327	0.0104351732240949	0.996685647939034
5	10.87575507164	0.0129245979804518	0.994887413562831	0.0122832689224755	0.995457235715266
6	10.6589560508728	0.0100497871857401	0.996910460740323	0.0099303534370124	0.997001665902648
7	10.4849910736084	0.0141591300880281	0.9938583097824	0.0143450680062064	0.993910280147504
8	10.1106457710266	0.0112373309058055	0.99613907733602	0.0113791019125445	0.996023741177131
9	11.3774919509888	0.0129144946318894	0.994898371129057	0.0125256228049578	0.995224166505699
10	12.062313079834	0.0185916651380368	0.989446593807513	0.0175050376675086	0.990313187959903
11	10.6330080032349	0.0109123630972345	0.996382208705982	0.00991358612421799	0.996566167739494
12	10.3893020153046	0.0127245001246607	0.995051188419377	0.0126563779598555	0.995049799574166
13	10.387286901474	0.0113891522600692	0.996028196889687	0.0112507361253583	0.996220700206415
14	10.2915189266205	0.0166241198047695	0.991530270876284	0.0163789925402969	0.992086379787645
15	10.7030138969421	0.0112860969536863	0.996121262394743	0.0110312117878899	0.995950913240981
16	10.2625670433044	0.0161159505945021	0.992034323434698	0.0161847099162835	0.992380746386383
17	10.2043919563293	0.015016229573402	0.993098200069085	0.0172833968805421	0.99099654516411
18	10.4487249851227	0.0177664601600231	0.990347026395533	0.0182303811595444	0.989833492315444
19	10.5334708690643	0.0163530265539524	0.992325404648456	0.0177863957057298	0.991039269053764
20	10.3978922367096	0.0113064647486011	0.996142253528921	0.0101032089642312	0.996583795570753

Table 7: Results for 20 fold cross-validation ANN (Synthetic Data)

A.3.3 Synthetic Dataset: RSQTOA + Grid Sampling

fold	equation	train time	train rmse	train r^2	test rmse	test r^2
1	0.145 * x_2	9.82622075080872	0.0199330691374121	0.987787095600649	0.0215330925321987	0.986980436367814
2	0.143 * x_2	8.26096415519714	0.0355495990306952	0.961304510582289	0.037568529105063	0.957838295234492
3	0.146 * x_2	8.1701238155365	0.0215722784279629	0.985763606524796	0.0226062967120438	0.984473979975273
4	0.147 * x_2	8.65511059761047	0.0359501673681952	0.960465067159817	0.0344912092594849	0.963791029352174
5	0.145 * x_2	8.46494317054749	0.02950403627761	0.973357839297395	0.0291851476403995	0.974354212565567
6	0.143 * x_2	8.22423005104065	0.0254875879230496	0.980128187470749	0.0262900889138255	0.978984741508304
7	0.144 * x_2	8.52138996124268	0.0263732866949083	0.978691970935034	0.0263665556037018	0.979426942356935
8	0.144 * x_2	8.28684282302856	0.0274557682242129	0.976952068821279	0.0280886722449158	0.975771800205805
9	0.142 * x_2	8.80992388725281	0.0415322907672072	0.947237447510837	0.040841662305316	0.949224136636053
10	0.144 * x_2	8.76067304611206	0.0388984030112583	0.953802381447206	0.0386408500336646	0.952799333279308
11	0.146 * x_2	8.56440019607544	0.0271094074063125	0.977672198407023	0.0254100449370027	0.977440558281542
12	0.145 * x_2	8.31949520111084	0.0407580005767406	0.949225552421874	0.0398541692922501	0.950914704085498
13	0.144 * x_2	8.27154922485352	0.0402413887101843	0.950414939844811	0.040853807765565	0.950167286073129
14	0.145 * x_2	8.20214104652405	0.028104171324669	0.97579340454689	0.0273597531729322	0.977918670541067
15	0.143 * x_2	8.40749025344849	0.0347316218472418	0.963267289898572	0.0347054295480353	0.959922051554886
16	0.142 * x_2	8.2300238609314	0.0161183284616499	0.99203197282737	0.0156802001625127	0.992848357890209
17	0.142 * x_2	8.55026936531067	0.0318810947724235	0.968889499545175	0.033825665924127	0.9655139312104
18	0.144 * x_2	8.25645971298218	0.0395656431573388	0.952126435498206	0.0404264002115524	0.950006860742475
19	0.144 * x_2	8.35535430908203	0.0297342221067466	0.974626966688054	0.03077212065427	0.973178540923425
20	0.147 * x_2	8.3073103427887	0.022915779286007	0.984152918374407	0.022218542798252	0.983478239815021

Table 8: Results for 20 fold cross-validation RSQTOA with Grid Sampling (Synthetic Data)

A.3.4 Synthetic Dataset: RSQTOA + Interpolation

fold	equation	train time	train rmse	train r^2	test rmse	test r^2
1	0.142 * x_2	8.61742806434631	0.0261086292849004	0.979047359823227	0.028184565836754	0.977694790142799
2	0.144 * x_2	8.48396253585815	0.0160929500886678	0.99207019953564	0.0161346917902325	0.992223375542164
3	0.141 * x_2	8.38601994514465	0.0129231839074819	0.99489086947013	0.0139098358441631	0.994121787196039
4	0.144 * x_2	8.42545104026794	0.0144189008995754	0.993640199594407	0.0136404269709544	0.9943368864924
5	0.144 * x_2	8.36614918708801	0.0147887654932408	0.99330623561407	0.0144624990134922	0.993702344008451
6	0.143 * x_2	8.60157895088196	0.0142951336226926	0.993748900006754	0.0147488133634798	0.993385990926614
7	0.145 * x_2	8.35401010513306	0.0198466042915541	0.987933336825972	0.0197010242529597	0.988513977693167
8	0.141 * x_2	8.39586901664734	0.0155572334686118	0.992600038042896	0.015993006577598	0.992145492531099
9	0.143 * x_2	8.41556811332703	0.0151079450816893	0.993018241531681	0.0154350217489748	0.992747879894915
10	0.143 * x_2	8.42550396919251	0.0136476731546713	0.994313130377149	0.0135170606906525	0.994224104628888
11	0.142 * x_2	8.44446992874146	0.0136659971069588	0.994326010018629	0.0112607048205632	0.99556954193298
12	0.142 * x_2	8.39115953445435	0.0207403285150597	0.986852271511933	0.0208624061901002	0.986549677900068
13	0.143 * x_2	8.47404909133911	0.0147111016200148	0.993373330421046	0.0143971727736103	0.993811236443678
14	0.141 * x_2	8.4495222568512	0.015505152330235	0.992632091594241	0.0145654892265574	0.993741777205513
15	0.142 * x_2	8.38376092910767	0.0199781506464224	0.987846162906371	0.0188197803777281	0.988214724583215
16	0.143 * x_2	8.38744926452637	0.00946861874707098	0.99725030791637	0.00910671154769461	0.997587731034105
17	0.142 * x_2	8.39370894432068	0.0239653373238381	0.982420454717517	0.0268367204059908	0.978292497348681
18	0.143 * x_2	8.40884399414063	0.0146779219036825	0.993411471172568	0.0149680226759542	0.993146551852851
19	0.146 * x_2	8.59738516807556	0.0135927898407233	0.994697551832628	0.0147094629923003	0.993871401333898
20	0.142 * x_2	8.38841080665588	0.00971977567767162	0.997149030852504	0.00822006480060608	0.997738609040105

Table 9: Results for 20 fold cross-validation RSQTOA with Nearest Neighbour Interpolator (Synthetic Data)

A.3.5 Yelp Dataset: ElasticNet

fold	train time	train rmse	train r^2	test rmse	test r^2
1	4.35128116607666	0.0925027302493302	0.687506225334651	0.0928348705247901	0.674694374816273
2	3.65365600585938	0.0925108527650028	0.68703935121514	0.0929823831728076	0.681853241924966
3	3.75197792053223	0.0923678405865706	0.687421371699015	0.0956098834527884	0.675257342904831
4	3.82155585289001	0.0924684508981778	0.686624639312656	0.0935046625001052	0.691500596930838
5	3.64496684074402	0.0926734306224856	0.686441499520466	0.089711639805585	0.694496412495565
6	3.62987112998962	0.0924901941930671	0.686717721412079	0.0931155928432854	0.689669105341251
7	3.66295909881592	0.0924359987589644	0.687272602380959	0.0943023238418417	0.678147769488701
8	3.68484616279602	0.0927209624407575	0.685555390563684	0.0888365475549311	0.71041720722626
9	3.59483313560486	0.0924566003200807	0.685569298003554	0.0939843113858241	0.707501528741379
10	3.90515804290772	0.0926242015474174	0.686460853820542	0.0903593910922355	0.695894772884232
11	3.8297221660614	0.0925370821125336	0.686222479656967	0.0921217523112083	0.699369405470867
12	4.33672094345093	0.092547682483551	0.686863122168241	0.0919106785745158	0.687698491692227
13	3.54536724090576	0.0925140108575776	0.686744068121163	0.092583643651335	0.689758599729412
14	3.71695423126221	0.0926358288081692	0.686145935714499	0.0904583247141822	0.699782113138795
15	3.52141618728638	0.0923456292729479	0.687475501346669	0.0960760159144794	0.673938789254604
16	3.54597115516663	0.0925497890068458	0.687473613845945	0.0924273167394454	0.671600592525161
17	3.47480702400208	0.0924669371054289	0.687882579452835	0.0933366954067982	0.668288861033094
18	3.34369015693665	0.0923934690434688	0.688155450109393	0.0949377597413594	0.661537514787777
19	3.47413611412048	0.0926512146329709	0.686636143346787	0.0900688055594315	0.690824063054006
20	4.4143443107605	0.0925362737866616	0.687192538173702	0.0919711171202333	0.682444588493686

Table 10: Results for 20 fold cross-validation ElasticNet (Yelp Data)

A.3.6 Yelp Dataset: ANN

fold	train time	train rmse	train r^2	test rmse	test r^2
1	48.779146194458	0.0973870955166883	0.713338738666423	0.0978018105016182	0.701188734136927
2	44.8805141448975	0.09752094432037	0.712171317551367	0.0973917621978851	0.711128614474812
3	44.4033639431	0.0974457522547273	0.712076315340731	0.100449418504054	0.70333762094484
4	43.0458319187164	0.0972464785823957	0.713147064094289	0.0982593557399322	0.718051679353508
5	43.4100058078766	0.0975097033520052	0.712698642538967	0.0939248560405559	0.722850747130563
6	45.2009162902832	0.0974212967213811	0.712335588594429	0.0982908325648415	0.713819435740734
7	43.4106419086456	0.097797744288484	0.710282171262264	0.0999118196350675	0.700994381388779
8	45.6158940792084	0.0979833275385113	0.709379269608856	0.0934410069518781	0.73484592419319
9	42.5294599533081	0.0970597934971925	0.713211506713616	0.0985690231647084	0.733726656212174
10	45.6887459754944	0.0978631212637225	0.710322460822306	0.0959797869595957	0.716031138394075
11	42.0417230129242	0.0980067638403451	0.708702854133689	0.0979491656941264	0.718716568498431
12	46.3202240467072	0.0972477906127527	0.713848177913018	0.0963168173864476	0.716155632472809
13	42.0062429904938	0.097528445537194	0.711875101457676	0.097893612405809	0.712939255504576
14	43.0845010280609	0.0980971302557163	0.708716218407287	0.0956066677030017	0.722444538066468
15	42.8473680019379	0.0976411426302771	0.710831483494558	0.100990375813579	0.701830616177934
16	42.668398141861	0.0974280552253604	0.713359344919262	0.0976080823363591	0.696885199639397
17	41.2012901306152	0.0972885313599284	0.714041986448955	0.0985952343504054	0.693661928605925
18	42.7375378608704	0.0970694843157126	0.715124609256494	0.100405498805473	0.686684769512601
19	43.1995508670807	0.0978798214430071	0.710554452959649	0.094984367387862	0.715426129067593
20	45.1764070987701	0.0975108685762991	0.712529675641201	0.0971220329624974	0.706920106338151

Table 11: Results for 20 fold cross-validation ANN (Yelp Data)

A.3.7 Yelp Dataset: RSQTOA + Grid Sampling

fold	equation	train time	train rmse	train r^2	test rmse	test r^2
1	-0.153 * ratio_positive_reviews	23.329611063	0.099814015	0.698873333	0.100031783	0.687407021
2	-0.163 * ratio_positive_reviews	22.911858320	0.099964236	0.697568120	0.099592834	0.697923991
3	-0.180 * ratio_positive_reviews	21.855125904	0.099515950	0.699712686	0.103203247	0.686848578
4	-0.182 * ratio_positive_reviews	22.172381639	0.101620033	0.686765026	0.102508548	0.693138883
5	-0.031 * elapsed_days_restaurant	21.419018030	0.101125237	0.690998132	0.097231441	0.702993423
6	-0.144 * ratio_positive_reviews	22.583911820	0.102070667	0.684223177	0.102734365	0.687359259
7	-0.173 * ratio_positive_reviews	21.708087921	0.100396250	0.694681912	0.102236909	0.686915868
8	-0.151 * ratio_positive_reviews	22.322629929	0.099135122	0.702506637	0.094833102	0.726886469
9	-0.165 * ratio_positive_reviews	26.403454542	0.102947889	0.677360235	0.105499077	0.694968919
10	-0.158 * ratio_positive_reviews	25.386448145	0.104907374	0.667119196	0.101398929	0.683059362
11	-0.171 * ratio_positive_reviews	25.429505110	0.102607294	0.680713476	0.102535769	0.691756817
12	-0.022 * elapsed_days_restaurant	22.926193237	0.102248865	0.683660074	0.101391942	0.685454905
13	-0.035 * elapsed_days_restaurant	24.803186178	0.099072155	0.702681861	0.099664891	0.702457154
14	-0.146 * ratio_positive_reviews	22.360116005	0.103395969	0.676398232	0.100157721	0.695391337
15	-0.162 * ratio_positive_reviews	25.441089869	0.104015727	0.671841776	0.108025376	0.658842721
16	-0.037 * elapsed_days_restaurant	27.147615194	0.098265742	0.708409077	0.098268494	0.692769608
17	-0.028 * elapsed_days_restaurant	27.500000238	0.102375739	0.683354687	0.104744924	0.654255656
18	-0.154 * ratio_positive_reviews	27.568541050	0.103609824	0.675442700	0.105269237	0.655595014
19	-0.029 * elapsed_days_restaurant	24.784371614	0.099286361	0.702175955	0.096193258	0.708136324
20	-0.160 * ratio_positive_reviews	27.274719954	0.099871458	0.698442755	0.100303631	0.687403723

Table 12: Results for 20 fold cross-validation RSQTOA with Grid Sampling (Yelp Data)

A.3.8 Yelp Dataset: RSQTOA + Interpolation

fold	equation	train time	train rmse	train r^2	test rmse	test r^2
1	-0.395 * negative_reviews	34.629176140	0.170933938	0.116873819	0.167065357	0.128081266
2	-0.407 * negative_reviews	36.966429949	0.115622929	0.595399705	0.118030007	0.575727465
3	0.738 * immediacy_index	38.368922949	0.132836606	0.464958903	0.135680359	0.458745365
4	0.769 * immediacy_index	31.032809734	0.162722713	0.196829153	0.161891151	0.234636206
5	-0.399 * negative_reviews	38.907992840	0.113596711	0.610081794	0.109321998	0.624536484
6	-0.446 * negative_reviews	35.211400986	0.117934385	0.578440150	0.119355201	0.578015221
7	0.805 * immediacy_index	32.861184835	0.167909650	0.145978377	0.169145244	0.143030802
8	-0.269 * elapsed_days_restaurant	34.627967119	0.119099714	0.570618201	0.116621873	0.586968352
9	-0.357 * negative_reviews	35.191020012	0.119162527	0.567722717	0.120840369	0.599805739
10	-0.537 * total_reviews	38.023925304	0.127578407	0.507698605	0.126954453	0.503170853
11	-0.461 * negative_reviews	38.249353886	0.113100700	0.612068801	0.111670222	0.634390513
12	-0.421 * negative_reviews	35.973614216	0.168313547	0.142813815	0.168979860	0.126333009
13	0.748 * immediacy_index	40.018521309	0.163831224	0.186960712	0.164120234	0.193156378
14	-0.487 * negative_reviews	33.097737074	0.122029524	0.549252316	0.121117293	0.554563610
15	0.723 * immediacy_index	38.994995117	0.170266191	0.120689789	0.173802376	0.116890639
16	-0.473 * total_reviews	38.034450054	0.127294216	0.510686649	0.127328432	0.484194041
17	-0.487 * total_reviews	36.166328430	0.151049518	0.310685126	0.150262430	0.288475384
18	-0.502 * total_reviews	35.340193272	0.136588351	0.435951002	0.137997044	0.408157642
19	-0.457 * total_reviews	36.692008257	0.139064541	0.415730278	0.137427541	0.404285280
20	-0.579 * total_reviews	34.718953848	0.169784904	0.128464391	0.166862002	0.134901830

Table 13: Results for 20 fold cross-validation RSQTOA with Nearest Neighbour Interpolator (Yelp Data)

Zentrales Prüfungsamt/Central Examination Office

RWTH AACHEN UNIVERSITY

Eidesstattliche Versicherung
Declaration of Academic Integrity

Rajani, Ashish

Name, Vorname/Last Name, First Name

416943

Matrikelnummer (freiwillige Angabe)
Student ID Number (optional)

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Arbeit/Bachelorarbeit/ Masterarbeit* mit dem Titel
I hereby declare under penalty of perjury that I have completed the present paper/bachelor's thesis/master's thesis* entitled

Data-Driven Quasi Taylor-Order Analysis for Collaborative Filtering of Reviews

selbstständig und ohne unzulässige fremde Hilfe (insbes. akademisches Ghostwriting) erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.
independently and without unauthorized assistance from third parties (in particular academic ghostwriting). I have not used any other sources or aids than those indicated. In case that the thesis is additionally submitted in an electronic format, I declare that the written and electronic versions are fully identical. I have not previously submitted this work, either in the same or a similar form to an examination body.

Aachen, 12.07.2023

Ort, Datum/City, Date

Ashish

Unterschrift/Signature

*Nichtzutreffendes bitte streichen/Please delete as appropriate

Belehrung:
Official Notification:

§ 156 StGB: Falsche Versicherung an Eides Statt
Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.
§ 156 StGB (German Criminal Code): False Unsworn Declarations
Whosoever before a public authority competent to administer unsworn declarations (including Declarations of Academic Integrity) falsely submits such a declaration or falsely testifies while referring to such a declaration shall be liable to imprisonment for a term not exceeding three years or to a fine.
§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt
(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.
(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.
§ 161 StGB (German Criminal Code): False Unsworn Declarations Due to Negligence
(1) If an individual commits one of the offenses listed in §§ 154 to 156 due to negligence, they are liable to imprisonment for a term not exceeding three years or to a fine.
(2) The offender shall be exempt from liability if they correct their false testimony in time. The provisions of § 158 (2) and (3) shall apply accordingly.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:
I have read and understood the above official notification:

Aachen, 12.07.2023

Ort, Datum/City, Date

Ashish

Unterschrift/Signature

Zentrales Prüfungsamt/Central Examination Office

Antrag auf Verwendung des Logos in der neuen Logosystematik auf einer Prüfungsarbeit

Application for the Use of the University Logos according to the Corporate Design/Logo System in a Written Paper or Thesis

Studiengang/Degree Program: Computational Social Systems

Abschluss/Degree: Master of Science

416943

Matrikelnummer/Matriculation No.

Rajani, Ashish

Name, Vorname/Last Name, First Name

Telefonnr. für Rückfragen/
Phone number (for possible inquiries) +49 163 142 3160

Bezeichnung der Prüfungsarbeit/Type of Written Paper:

Final Thesis

(z.B. Hausarbeit, Projektarbeit, Abschlussarbeit/e.g. seminar paper, project paper, final thesis)

Name der Institution, deren Logo verwendet werden soll/Name of the institution whose logo is to be used in the written paper:

Institute of Computational Social Sciences and Humanities (CSSH)

Mir ist bekannt, dass/I understand that:

- ich das kombinierte Logo nur verwenden darf, wenn es von der gewünschten Institution für die Nutzung auf Prüfungsarbeiten freigegeben ist/I am only allowed to use the logo if its use in written papers is generally approved by the relevant institution
- ich das kombinierte Logo selbständig und zeitnah bei dem betreuenden Institut/Lehrstuhl anfordern muss/I have to request the logo on my own and in due time from the relevant institution
- das Logo nicht verändert, beschnitten oder aus der Logosystematik losgelöst werden darf/the logo is not to be modified, cut, or used without immediate contextual elements (i.e. the logo system)

- das RWTH Aachen University-Logo eine eingetragene und geschützte Wortmarke ist und ein Verstoß gegen die Vorgaben zur Verwendung des Logos Rechtsfolgen hat/the RWTH Aachen University logo is a registered and protected trademark. Use of it that violates the specifications on its use may have legal consequences
- das kombinierte Logo nur für die hier angegebene Arbeit verwendet werden darf/the combined logo is only to be used for the above named written work
- das kombinierte Logo nur mit dem Zusatz „Diese Arbeit wurde vorgelegt am [Name Institution]“ angebracht werden darf/the combined logo can only be used in combination with the statement, “The present work was submitted to [Name of institution]”
- das Logo in keiner Art verändert werden darf/it is not permitted to modify the logo in any way.

Diese und weitere Informationen (inkl. Anwendungsbeispiel) kann ich auf der Webseite www.rwth-aachen.de/go/id/hjxv jederzeit nachlesen.
For this and further information (including examples of use), please visit the ZPA website www.rwth-aachen.de/go/id/hjxv/?lidx=1 .

25.05.2023

Datum/Date

Unterschrift/Signature

Diese Bescheinigung ist in der Regel zu Beginn und spätestens bei Abgabe der Prüfungsarbeit beim ZPA einzureichen.
This form is to be filled out and submitted to the ZPA, as a rule prior to commencing the written work, but at the latest when submitting the written work to the ZPA.