

Talk to GP: Explaining Genetic Programming models through natural language

N.Sweijen

Talk to GP: Explaining Genetic Programming models through natural language

by

N. Sweijen

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday July 5, 2024 at 09:30 AM.

Student number: 4693949
Project duration: November 13, 2023 – July 5, 2024
Thesis committee: Prof. dr. P.A.N. Bosman TU Delft, Algorithmics, supervisor
Dr. T. Alderliesten, LUMC, Radiation Oncology, supervisor
Dr. C. Lofi, TU Delft, Web Information Systems
, J. Harrison, CWI, Daily supervisor

An electronic version of this thesis is available at <http://repository.tudelft.nl/>.

Preface

Over the course of the last 9 months I have been working on this thesis you are about to read. From the moment I followed the first lecture evolutionary algorithms lecture, I was intrigued and knew I wanted to do my thesis there. This is a decision that I have not regretted, during my time at the CWI I have gained a lot of new knowledge about LLMs, model interpretability and much more. Even though this type of research was very new to me, I managed to overcome all the challenges even though I sometimes needed a little help with this.

Throughout this time I have received a lot of support from my supervisors. I would like to Peter and Tanja for all the meetings we had, in which we always had interesting conversations which gave me a lot of inspiration for my project. Besides that, I want to give a thank you to Eduard, with whom I have had a lot of conversations about TalkToModel and what could be possible in this system. Lastly, I would like to give a special thanks to Joe, who has spent countless hours helping and supporting me with this thesis. He has helped me tremendously and always helped me steer the project in the right direction.

Lastly, I want to thank my friends and family for always being there for me and helping me with the not-so-technical side of things. I also want to thank you, the reader; I hope you will enjoy this thesis.

*N. Sweijen
Delft, June 2024*

Abstract

Machine learning (ML) models are used increasingly in high-stakes areas such as health and finance because of their strong performance. However, having good performance in metrics such as accuracy or the f1 score alone is not all that is important as trust is also essential in these areas. TalkToModel is a system that addresses this challenge, using a large language model (LLM), by letting the users interact with their models through natural language. However, this system only allows a user to ask black-box questions. Another way to achieve trust is through models that are constructed with interpretability in mind that a human can understand. Genetic programming (GP) models are such models that have the potential to be interpreted. This thesis investigates if GP models can be made even more interpretable using TalkToModel. To do this an enhanced version of TalkToModel called TalkToGP is created with three main contributions: **1)** Integration of GP Models into TalkToModel, **2)** the ability to ask GP model-specific questions and **3)** the ability to do a comparative analysis between multiple GP models. This system is built using the feedback from GP users who gave insights on their experience with GP as well as their wishes for this system. In the end, the system is evaluated by GP users in an experiment. The experiments showed that the enhanced version of TalkToModel shows a strong indication that it increases the interpretability of GP models. This means the system could be a useful tool for anyone working with GP models.

Contents

1	Introduction	1
1.1	Interpretability	1
1.2	TalkToModel	2
1.3	Genetic Programming	2
1.4	TalkToGP	3
1.5	Research Questions	3
1.6	Contributions	3
1.7	Outline	4
2	Previous works	5
2.1	Explaining Genetic Programming Trees using Large Language Models	5
2.2	Conversational explanations of Machine Learning models using chatbots	6
2.3	Talk-To-Model	8
2.3.1	The system	8
2.3.2	TalkToModel Operations	11
2.3.3	LLMs	13
3	Methods	15
3.1	TalkToGP	15
3.1.1	GP models in TalkToModel	15
3.1.2	Extending the Question Set	15
3.1.3	Multiple Models	18
3.1.4	New Questions	18
3.1.5	Fixing Original TalkToModel Questions	22
3.1.6	Code availability TalkToGP	25
3.2	Exploratory interview	25
3.3	Experiment	25
3.3.1	Objectives	25
3.3.2	Experimental design	26
3.3.3	Procedure	28
4	Results	29
4.1	Results Evaluation Experiment	29
4.1.1	Model chosen	29
4.1.2	Time spent	29
4.2	Results questions	32
4.2.1	Easy of use	32
4.2.2	Interpretability	32
4.2.3	Most likely to use	33
4.3	Results behaviour	33
4.3.1	Jupyter Notebook	33
4.3.2	Talk To Model	33
5	Discussion	35
5.1	Interpretation of results	35
5.2	Contributions	36
5.3	Limitations	36
5.4	Future work	37

6 Conclusion	39
A Exploratory questions	41
A.0.1 General questions about models	41
A.0.2 Questions about the system	41

Introduction

Machine learning (ML) models are advancing at an unprecedented pace, achieving remarkable performance improvements across a wide array of applications. With this increasing performance, people want to use these models more in tasks previously done only by humans. In areas such as medicine, the criminal justice system, and financial markets, ML models are becoming more prominent (Lipton, 2018). For instance, in medicine, ML models can assist in diagnosing diseases, predicting patient outcomes, and personalizing treatment plans. In the criminal justice system, they can help predict recidivism, aid in sentencing decisions, and improve the efficiency of legal processes. Similarly, in financial markets, ML models can be used for fraud detection, algorithmic trading, and risk management.

However, high performance in classic metrics such as accuracy or the F1 score alone is often insufficient for the widespread adoption of ML models in these sensitive and high-stakes areas. Trust is essential in many real-world settings to even consider using an ML model. For example, in healthcare, a model that predicts the likelihood of diabetes, the model must not only be accurate but its predictions must also be trusted by the patients as well as the healthcare providers. Many of the best-performing models today do not suffice this, however.

Building trust in ML models involves ensuring their decision-making processes are transparent and understandable. Unfortunately, many of the most powerful ML models today operate as "black boxes". Often composed of complex neural networks with billions of parameters, these models can make it nearly impossible for humans to understand how specific decisions are made. Thus, these ML models are called black boxes because we can not see what happens inside the systems. This uncertainty poses significant challenges, especially when the consequences of the model's decisions can deeply impact human lives.

1.1. Interpretability

To address these challenges, the field of interpretable ML has grown to incorporate diverse techniques and methods for understanding these models and datasets (Singh et al., 2024). Researchers are developing techniques and tools to make ML models more transparent and their decisions more explainable. One part of this development has focused on post-hoc explanations. Post-hoc explainability methods analyze and interpret the decision-making process of a trained ML model after it has made predictions, providing insights into how the model arrived at its outputs (Retzlaff et al., 2024). Another part of this development is inherently interpretable models. Such as generalized additive models, decision trees, or Genetic Programming (GP) models. These models are different from black-box models because, unlike black-box models, the inside of interpretable models has the potential to be understood. By simply looking at these models and analyzing them, a human can possibly understand and interpret these models.

Lipton, 2018 claims that the task of interpretation remains under-specified. It states that there is no agreed-upon meaning for interpretability. Furthermore, it also states that for research to be meaningful in its claims to offer interpretability, the research should fix a specific definition. In this thesis, the definition of the paper "*Towards A Rigorous Science of Interpretable Machine Learning*" is used. This paper states the definition of interpretability in the context of ML systems is: **the ability to explain or**

to present in understandable terms to a human.

Enhancing interpretability can help us better understand the reasoning behind a model's predictions, identify potential biases, and improve overall trust in the models. This shift towards interpretability is essential for integrating ML models into critical domains.

1.2. TalkToModel

The challenging task of model interpretability in machine learning can be improved using Large Language Models (LLMs) (Singh et al., 2024). These models provide a powerful means to bridge the gap between complex machine learning operations and human understanding, making it easier to explore and comprehend the inner workings of ML systems. A system that utilizes LLMs for this task is TalkToModel.

TalkToModel is a system that allows its users to learn how an ML model works by interacting with it through natural language (Slack et al., 2023). Slack et al., 2023 describes that users can "talk" to a model via a user interface and ask questions about the model, such as: "*What are the most important features in this prediction?*" and "*What could be done to change the prediction?*". The system uses a large language model (LLM) to convert the input of a user to an operation the system can understand and execute. The system contains a predefined set of operations the system can execute. After the input is converted, the system responds to the question with a template answer. As an example, a user could ask the following to the system: "*Applicant #358 wants to know why they were denied a loan. Could you tell me why?*". The system then translates this to a form that it can execute: filter applicant 358 feature importance using an LLM. This is then executed and the system responds in this example with: "They were denied because their credit score is too low".

In the paper, there is also an evaluation of the system performed, by doing a user study. In this evaluation, the users were given a set of questions they had to answer using both TalkToModel and an existing system. It was found that many people want to use this system over the existing system for understanding a disease prediction model. Also, most of the ML professionals in this survey agreed that TalkToModel was easier to use than one of the most popular open-source explainability dashboards (Dijk et al., 2022). Lastly, the evaluation also showed that the participants answered the questions more accurately when using TalkToModel. This evaluation shows that TalkToModel can be very effective for increasing model explainability.

The system, however, only uses post-hoc explanations and treats the model like a black box. As mentioned, post-hoc explanations interpret and analyze a model by only looking at the input and output. This means that the inner workings of the system are not explored. This has the advantage that the system is agnostic about the kind of model that is used since the system only looks at the input and output. This, however, could be a shortcoming when the inner workings of a model can help us increase the explainability of that model.

1.3. Genetic Programming

Symbolic regression (SR) is the task of fitting a mathematical expression to an input-output pair. The mathematical expression consists of operators, like multiplication or subtraction, constants, and variables. Symbolic regression is often done with interpretability in mind, as the resulting expressions have the potential to be interpretable.

Genetic programming (GP) is often used for SR. GP is a collection of evolutionary computation techniques that allow computers to solve problems automatically (Poli et al., 2008). In GP we evolve a population of computer programs. GP tries to stochastically improve the population of each generation to get better programs. How good a program is, is measured by executing the program and measuring how well it performs. The best programs of the population are selected, recombined with each other and mutated to create new programs. This is done each generation until an acceptable solution is found or another stopping criterion is met.

As mentioned, GP can be used for SR. Here, the SR expressions are the programs GP tries to evolve. These expressions are often expressed as trees. A tree consists of internal nodes and leaves. The internal nodes can be operators, while the leaves are constants or input features. Trees can be intuitive to interpret and analyze; however, when they get too big, they can be too difficult to comprehend. This is why when evolving expressions with GP, often not only the accuracy gets optimized.

When we have more than one goal we are optimizing for when running GP we call this Multi-objective

GP (MO-GP). In MO-GP the objectives often are chosen to be the prediction accuracy and the model complexity. The prediction accuracy can be expressed by an error measure such as the mean squared error or the R^2 score. The model complexity can be expressed by for example the tree length or by an expression complexity measure (Kommenda et al., 2015). In MO-GP we obtain a whole front of models instead of a single model. Each of these models is not dominated by another solution, meaning that each of them is not worse in both accuracy and complexity compared to another model. This causes a new issue because now we do not only need to understand a single model but multiple, possibly a lot of models. Apart from just understanding, usually, a single model has to be picked from the front to be used. This means that there is also a decision to be made and questions to be asked like: why is this particular model better than all the other models?

Even though GP models can be inherently more interpretable than other models, such as large, complicated neural networks, depending on the size of the solution, GP models can also become large and difficult to understand. If you pair this together with the fact that it is common to have a front of 10 or more models, one can imagine that this can quickly become a very difficult task for a human to interpret GP models.

1.4. TalkToGP

Since GP models are just models they could in principle be used by TalkToModel and be used to get post-hoc explanations from. This however doesn't make full use of the inherent interpretability of GP models since TalkToModel doesn't look at the model's inner workings.

This is why this thesis wants to combine TalkToModel with GP. There are, however, three problems with this. The first is that TalkToModel does not support all types of models, it only supports classification models whereas the GP models wanted are regression models. The second problem, as mentioned before, is that TalkToModel does not examine the inner workings of a model by default. A third problem is that TalkToModel only accepts a single model with its dataset at the time.

It would be very inefficient to load a single model each time to analyze, especially if you have a big set of models you have to pick one from. A preferred approach would be to load multiple models at once to examine them individually and to make it easier to compare the different models.

This thesis proposes **TalkToGP**, an extension of TalkToModel that allows a user to talk to multiple GP models at once. Apart from asking just black-box questions the system will also allow a user to ask questions about the inner workings of the GP models.

1.5. Research Questions

In this research, the aim is to help users to better understand GP models. This can be achieved by enabling TalkToModel to work with multiple GP models. To be more specific the desire is that TalkToModel can answer the original black box questions as well as answer questions about the specifics of the loaded GP models. In order to achieve this the following research question is asked:

Can Genetic Programming models be made more interpretable using TalkToModel?

To help answer this main research question the following sub-questions are asked:

1. What are the current challenges in interpreting GP models?
2. How can GP models be integrated into the TalkToModel system?
3. What kind of questions about GP should be added in the TalkToModel system?
4. How can multiple GP models be utilized simultaneously in the TalkToModel system?
5. How does TalkToModel facilitate the interpretability of GP models?

1.6. Contributions

In this thesis, the aim is to make contributions to the field of interpretable ML, particularly focusing on GP models by utilizing TalkToModel. Our contributions are multifaceted, addressing both the ease of use and the theoretical understanding of GP models:

1. **Integration of GP Models with TalkToModel:** The first contribution lies in making GP models usable in the TalkToModel system. Integrating GP models into TalkToModel helps lower the barrier to entry for using and understanding GP models. Now users can ask general black-box questions to these models instead of having to implement these methods themselves.
2. **Questions into the Specifics of GP Models:** The second contribution aims to give the system the ability to ask GP-specific questions. This can potentially do away with some of the difficulties with analyzing GP models. It allows users to easily ask questions and understand the components of each model without having to spend too much effort.
3. **Support comparative analysis:** Finally, TalkToModel is enhanced by enabling it to use multiple concurrent GP models trained on the same dataset, at once. This gives users the ability to perform comparative analysis between different GP models by comparing different aspects of the models. By doing this, the user can make comparisons, see what the strengths and weaknesses of each of the different models are, and ultimately make an informed decision on what model they would prefer to select for a certain task.

Through these three key contributions, this thesis tries to advance the accessibility, understanding, and comparative analysis of GP models by using the TalkToModel system. By bridging the gap between theory and application, the aim is to empower other researchers and ML practitioners alike in using the full potential of GP models for a wide range of ML tasks.

1.7. Outline

The rest of this thesis is structured as follows. In chapter 2 all relevant work to this thesis is discussed. Next, in chapter 3 it is explained how TalkToGP is created, the user study to get feedback on the system and the experiment that is used to evaluate the system are explained. Furthermore, in chapter 4 the results of the experiment are presented. Then in chapter 5 the results will be analyzed and discussed also some limitations and future work will be discussed. Lastly, in chapter 6 an overview of the research will be handled.

2

Previous works

In this section, two works that also use LLMs for better explainability of GP models will be discussed. Furthermore, the original TalkToModel paper will be discussed since this will be the starting point of the enhanced application.

2.1. Explaining Genetic Programming Trees using Large Language Models

In this paper, LLMs are used to improve the interpretability of GP (Maddigan et al., 2024). As mentioned in chapter 1, GP can be applied in many different tasks, and where our paper is focused on symbolic regression, this paper focuses on GP for non-linear dimensionality reduction (NLDR). NLDR is a technique used to reduce the dimensionality (reducing the number of features) of a problem, making a problem easier to analyze and understand. GP-NLDR is used for explainable NLDR where the reduced dimensions (embedding) can be directly understood in the context of the original features. In this technique, each dimension in the embedding is represented by a single GP tree.

This paper describes their approach to achieve better explainability of GP models. This is done by a web-based dashboard called GP4NLDR. This dashboard consists of multiple parts working together. The first part allows users to perform GP-NLDR in the application and the outputs of this are visible in the dashboard followed by tree expressions and visualizations for each new dimension. After this, an LLM (GPT-3.5 or GPT-4) gets initialized with some prompts and then finally gets prompted to "Provide an exciting summary of the results". After this the user is free to ask questions to the system, the LLM gets the question as input and outputs a response as output. Because GPT-3.5 is only trained until September 2021 some important research on GP is unknown to the system, for this reason Retrieval Augmented Generation (RAG) is used. RAG addresses this limitation by building a vector store/database of vector embeddings from relevant documents. This allows the system to inject relevant information if necessary.

There is a lot of similarity between this paper and this thesis. Both are concerned with increasing the interpretability of GP using LLMs. Both implement this by creating a chatbot interface with which the user can interact, and ask questions to improve their understanding of GP.

There are, however, also some key differences, the first being the GP task the papers focus on. While this paper focuses on NLDR, the thesis focuses on SR. Another difference is that this paper only uses an LLM (and RAG) to answer questions after having initialized the LLM with some initial prompts. Whereas TalkToModel uses an LLM to translate the user's question into a grammar which is then executed by the execution engine.

A weakness compared to our system is that the way the LLM is used makes it susceptible to hallucinations. LLMs are prone to suffer from hallucinations where the generated content is either in conflict with existing sources or cannot be verified by the available knowledge resources (Li et al., 2023). Hallucinations make using LLMs in real-world applications risky. This is something this thesis wants to avoid, especially in high-stakes environments such as in healthcare. In TalkToModel hallucinations are not possible because of the way the LLM is used. Another weakness is that because this application uses the OpenAI API it is prone to privacy issues. Since everything you ask is being sent to OpenAI

this is unwanted for sensitive data such as patient records. Our system avoids this by having the LLM run locally.

A strength compared to our system is, that in this case more of the full potential of LLMs is used. In our system, the LLM is used to translate user input into grammar. Which executes an operation, this means that all of the answers our system can output have to be manually coded. While in this paper the LLM has the freedom to answer any question. This can result in a better user experience since the users are not limited to a defined set of questions. Another strength of this system is that it can perform NLDR in the dashboard itself. In our application, it is not possible to run a GP algorithm but instead, you have to load in the GP models yourself. Even though loading in your own models gives you more freedom and does not limit you to running any predefined algorithm. This does have the disadvantage that users are forced to generate their GP models themselves, which can make the system harder to use.

2.2. Conversational explanations of Machine Learning models using chatbots

In this section, a thesis will be discussed that proposes a chatbot for explaining decisions of a predictive model (Kužba, n.d.). The thesis wants to find out what human operators would like to ask a model. In order to ask this question they developed the XAI-bot which offers a conversational interface to explanations. In the thesis, they trained a random forest model on the titanic dataset included in the DALEX package (Biecek, 2018) and described in (Biecek and Burzykowski, 2021). The system is model-agnostic and can be used with other models than a random forest model.

There are a lot of similarities between the XAI-bot and the TalkToModel system. Both use chatbots for a user to interact with in order to ask questions about a black box model and the corresponding dataset. Also, the way the answering of a question is done has a lot in common. Just like TalkToModel, the XAI-bot answers questions by first parsing user input and extracting the intention of this by transforming it into a form that the system can execute, this form will then be executed and then returned in the form of templates. Furthermore, they both use explainers to help users to answer questions. Explainers here are methods used to interpret and explain the decisions or outputs of the models. These explainers help users understand why a model made a particular prediction or decision by providing insights into the factors and features that influenced the model's output. The explainers that are used differ between the two systems. TalkToModel uses SHAP and LIME for example, whereas XAI-bot uses CeterisParibus and iBreakDown, but in both systems they are interchangeable and the system is not dependent on them. Even though the implementation differs between the two systems, the overall structure and idea behind the system are similar.

The big difference between this thesis and the TalkToModel paper is the focus of the paper. The main goal of the thesis is to find out what a human operator would like to ask the ML model. While the TalkToModel paper focuses more on what system could increase the explainability of models for users.

A weakness compared to TalkToModel is that this system has much fewer options for questions. As mentioned, the paper focuses more on the interactivity between a human and the system and as such has more functions for conversation such as praising the user among others. However, the system has only a fraction of the operations of TalkToModel which are visible in table 2.1. Another big weakness compared to TalkToModel is that in this system it is much harder to train for a new dataset. As will be explained in section 2.3.1 it is trivial to train a new dataset for the system. Only having to upload the data and run a training script that takes care of everything. In the XAI-bot there is no such system in place. This means that it can be time-consuming to train this for a new dataset which limits its usability.

Because the paper focuses more on the interactivity between the user and the system, this paper has some strengths over the TalkToModel paper. The first is that it uses a dialogue storage that stores all human-chatbot interactions. This storage can be used by the system for training. This way the system becomes better iteratively by training on questions by users the system might not have supported earlier. Another strength is in the presentation of the answers. Both systems use templates to answer however XAI-bot also has some alternatives for the templates to keep conversations fresh. On top of this XAI-bot also allows for more interactivity by including graphics and letting users respond using a button. While the enhanced TTM system described in chapter 3 also contains graphics, the original TTM does not. A screenshot of such an interactive response of the XAI-bot can be seen in figure 2.1.

Overall, the ease with which the datasets and models can be changed makes TalkToModel the

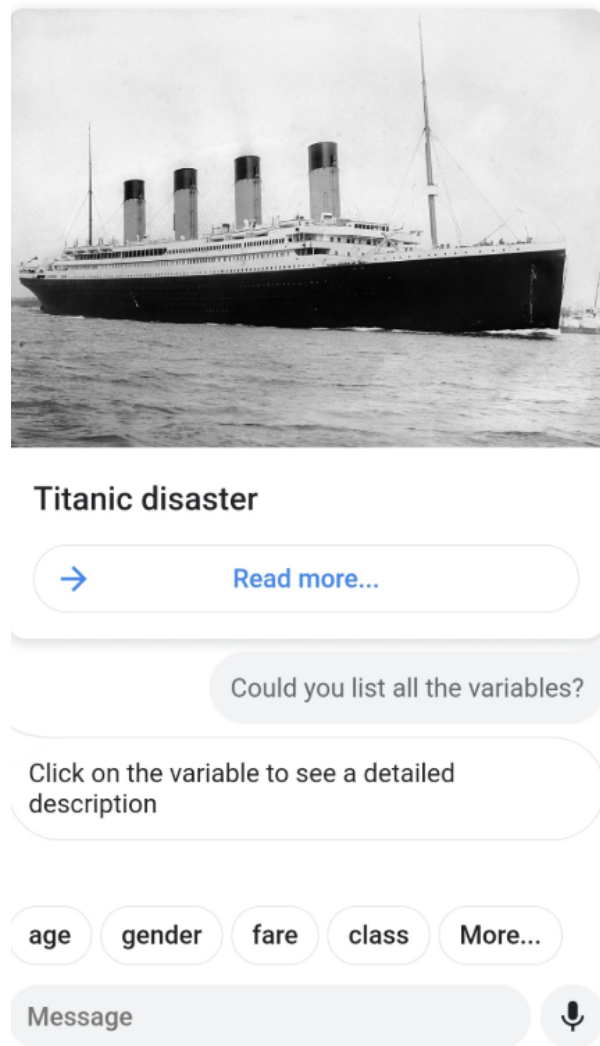


Figure 2.1: Screenshot of the XAI-bot showing extra interactivity with buttons.

preferred system for us, since our system should be able to support multiple models and datasets. Furthermore, the extra features such as the interactivity and the image responses can also be added to the TalkToModel system without too much effort. While the features that TalkToModel has that the XAI-bot lacks, are more difficult to implement.

2.3. Talk-To-Model

TalkToModel is the most important part of this thesis. It is the foundation of our system and it is therefore important to discuss this paper in detail. This section will first explain the system and after that, the operations of the system will be explained.

2.3.1. The system

This sub-section first gives an overview of the system. Then, the interface will be discussed, the dialogue engine will be next, and lastly, the execution engine will be handled.

Overview

As mentioned, TalkToModel is a chatbot that allows users to ask questions about their model and dataset. Using TalkToModel users can have discussions about why predictions occur, how the predictions would change if the data changes, and how to flip predictions, among many other conversation topics (Slack et al., 2023).

In figure 2.2 an overview of the system can be seen. In the system, there is a dataset together with a model loaded in. The user can ask a question to the system through natural language in *step 1*. In *step 2* the user input gets parsed by the system into a form the system can understand and execute. Lastly, in *step 3* the operation which the system got in the previous step gets executed and gets returned to the user in a readable format.

Interface

The system interface can be seen in figure 2.3. The interface is like having a chat with someone. This helps to make the conversations feel more natural. Apart from just chatting, the interface also allows you to pin answers from the system. This will save these messages on the right so you can easily select answers you find important. Furthermore, there is a feature to help users generate a question if they do not know a question to type themselves, this feature can be seen in figure 2.4. As can be seen, the feature allows you to select a category and this will then generate a random question of that type in the input field.

TTM is divided into two parts, the first is the dialogue engine that understands user inputs, maps them to operations and generates text responses based on operations. The second part is the execution engine which handles the operations.

Dialogue engine

The dialogue engine is performed in four steps.

1) the TalkToModel system constructs a grammar for the given dataset and model, this determines what parses there will be. This grammar is made in LARK (Shinan, 2017) which is a parsing toolkit for Python. The grammar specifies everything that the system can execute. Here is an example of how the "define" operation is specified in the grammar. This operation allows a user to get a feature explained.

```
define: defineword allfeaturenames
defineword: " define"
```

This means that the following is an accepted parse given that BMI is a feature of the dataset:

```
" define bmi"
```

TalkToModel changes the grammar based on what dataset is used. This means that you can use datasets and models interchangeably with other datasets and models without modifying the grammar. For example, if another dataset and model are used that does not contain the feature BMI but other features. The system automatically changes the grammar such that *define bmi* is no longer accepted but it would accept other parses that contain the features present in the new dataset.

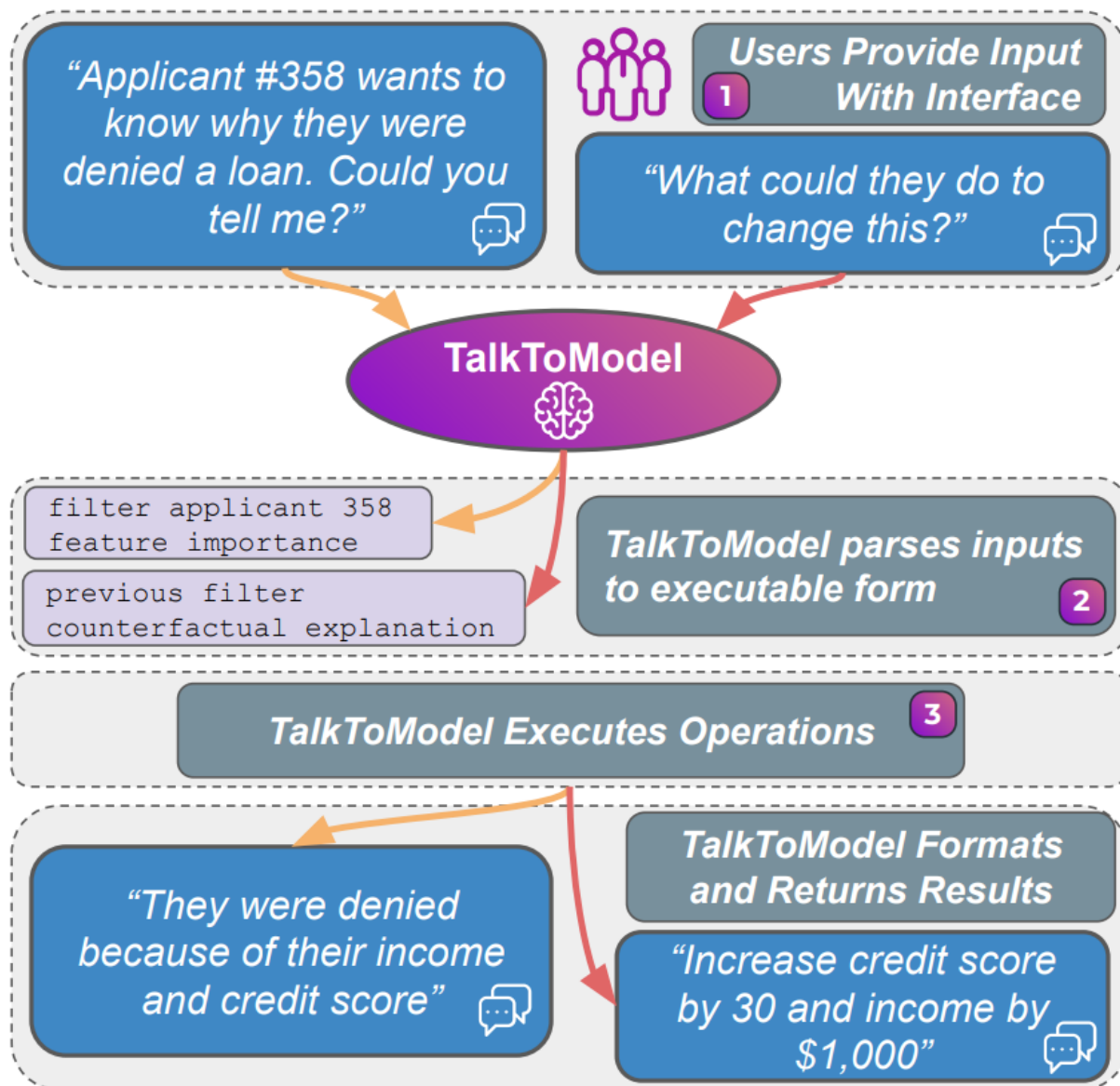


Figure 2.2: Overview of the TalkToModel system

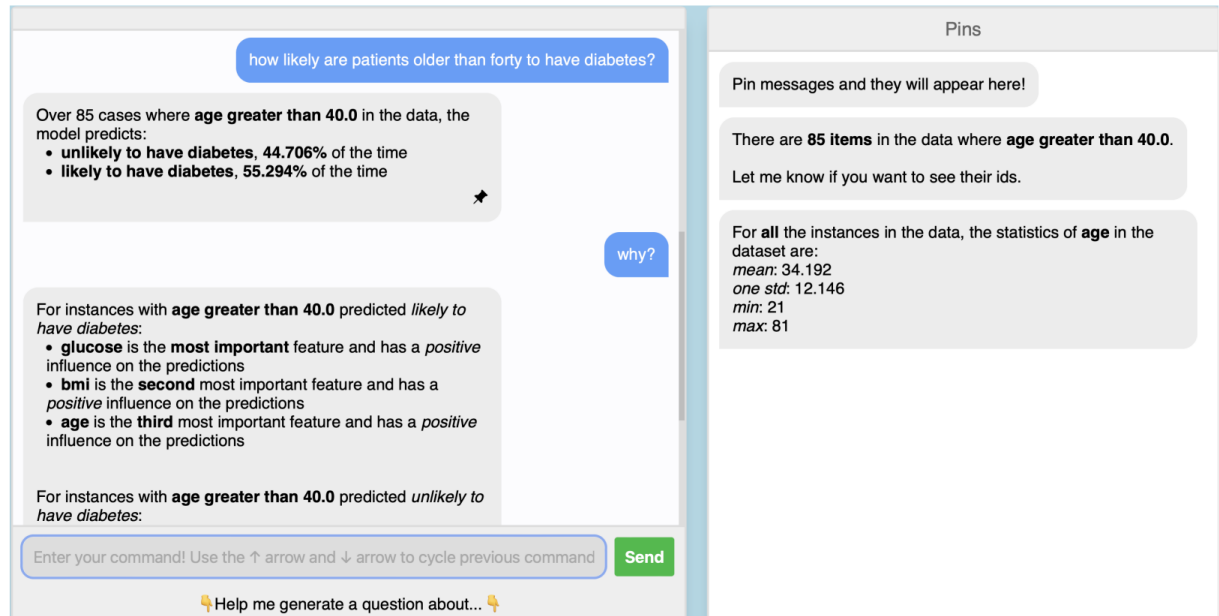


Figure 2.3: An screenshot of the Talk to Model system

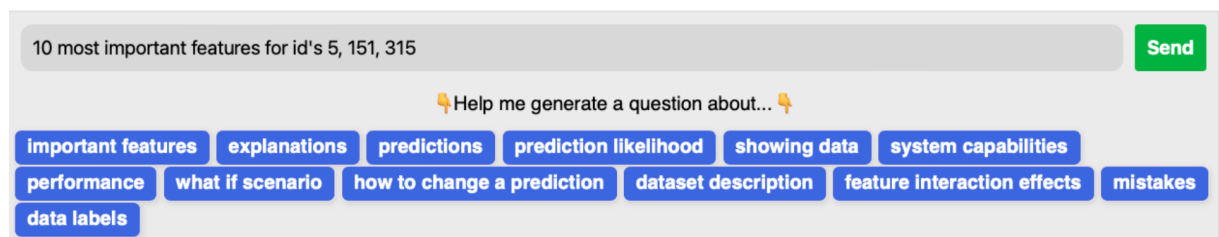


Figure 2.4: A screenshot of "Help me generate a question" section

2) TalkToModel will generate user utterance, parse pairs based on wild cards. These pairs look like this:

```
User: What are the predictions for instances with
      {num_features} less than {num_values}?
```

```
Parsed: filter {num_features} less than {num_values} and predict [E]
```

```
User: Could you show me the predictions on all the data?
```

```
Parsed: predict [E]
```

The line after "User:" is the user utterance and the line after "Parsed:" is the parse. The parse is what the utterance looks like in the grammar. Here the text between are wildcards. These can be filled in by the system. For the num_features wild card, the system can fill in any numerical feature. For num_values the system can fill in any possible numerical value. Then system generates more pairs by filling in these wildcards and adds these on top of the pairs that do not have wildcards.

3) TalkToModel finetunes an LLM using the generated utterances and parse pairs. This then allows a user utterance to be translated into the TalkToModel grammar by the LLM. More information about the different LLMs is given in section 2.3.3.

4) The system responds conversationally to questions asked by the user by executing the parses (which are in the grammar). This is actually in the form of simple templates. The LLM does not write the answer as in other chatbots such as ChatGPT or the system described in section 2.1. Instead, the system executes code, what to execute is specified by the parse. This then does some operations, for example, predicting a data point. The output of this prediction is then formatted into a readable format and returned to the user.

Execution engine

At the core, TalkToModel provides explanations of models using feature importance explanations. This means that TalkToModel treats its models as black-box and doesn't use any of the inner workings of the model for its answers. It is stated, however, that this can easily be extended using inner model workings. The system uses post-hoc explanations and uses multiple feature importance techniques: LIME and SHAP (Ribeiro et al., 2016; Lundberg and Lee, 2017) and uses a faithfulness metric to select the best of these explanations.

Apart from feature importance explanations, the system can also answer different questions. For example, questions regarding counterfactuals, such as: "What would happen to the prediction if I were to change this variable to x?" or questions about the system's function, such as "Explain what model is used." It is easy to expand this system to answer more questions, which makes it very flexible to add new models.

2.3.2. TalkToModel Operations

TalkToModel comes with a predefined set of questions. In table 2.1 you can find an overview of all the questions included in the system. The questions are divided into 5 different categories: Data, Explainability, ML error analysis, Conversation and Description. All of these categories will be discussed to show what they do and how they contribute to interpretability.

Data

The first category of data contains different ways to analyze and select and manipulate the data points the system is loaded in with. The filter category is one of the most important operations in the system and allows you to do an operation on a subset of the data. Here you can specify what subset you want by selecting features and values. Together with the "and" and "or" operations this allows you to make interesting operations for example: "What are the predictions on patients older than 30?" Which will filter the dataset to only include data points with age higher than 30 and then do the prediction on this set. Apart from this, there are ways to analyze the dataset by showing statistics or showing the items in the conversation.

Explainability

The second category can give the users explanations. All these operations can help the user understand or find justification for the models' predictions. These operations can help increase the in-

	operation, arguments, and description
Data	<code>filter(dataset, feature, value, comparison)</code> : filters dataset by using value and comparison operator
	<code>change(dataset, feature, value, variation)</code> : Changes dataset by increasing, decreasing, or setting feature by value
	<code>show(list)</code> : Shows items in list in the conversation
	<code>statistic(dataset, metric, feature)</code> : Computes summary statistic for feature
	<code>count(list)</code> : Length of list
	<code>and(op1, op2)</code> : Logical “and” of two operations
Explainability	<code>or(op1, op2)</code> : Logical “or” of two operations
	<code>explain(dataset, method, class=predicted)</code> : Feature importances on dataset
	<code>cfe(dataset, number, class=opposite)</code> : Gets number counterfactual explanations
	<code>topk(dataset, k)</code> : Top k most important features
	<code>important(dataset, feature)</code> : Importance ranking of feature
	<code>interaction(dataset)</code> : Interaction effects between features
ML	<code>mistakes(dataset)</code> : Patterns in the model’s errors on dataset
	<code>predict(dataset)</code> : Model predictions on dataset
	<code>likelihood(dataset)</code> : Prediction probabilities on dataset
	<code>incorrect(dataset)</code> : Incorrect predictions
Conv	<code>score(dataset, metric)</code> : Scores the model with metric
	<code>prev_filter(conversation)</code> : Gets last filters
	<code>prev_operation(conversation)</code> : Gets last non-filtering operations
Description	<code>followup(conversation)</code> : Respond to system followups
	<code>function()</code> : Overview of the system’s capabilities
	<code>data(dataset)</code> : Summary of dataset
	<code>model()</code> : Description of model
	<code>define(term)</code> : Defines term

Table 2.1: Overview of all operations supported out of the box in TalkToModel. In the table are 5 different categories of operations. The operation, arguments and description are all indicated by their own colour.

repeatability. These explanations can take on several forms such as showing the most important features for the predictions or showing patterns in the model's errors.

ML error analysis

The third category is ML error analysis. This category shows what a model predicts as well as how good those predictions are. Furthermore, there is an option to see what the wrong predictions are. These operations allow users to see how well the model does in terms of performance and enable them to see which data points get predicted correctly and incorrectly.

Conversation

The conversational category helps the system to make an actual conversation. The `prev_filter` and `prev_operation` call back to a previously specified filter or operation. This allows a user to call back to something without actually specifying it. For example, if the user previously asked: "What are the predictions on patients older than 30?". The user can then ask "What are then the most important features for them?". This will use `prev_filter` to see that patients older than 30 was previously filtered and thus only looks for the most important features for these data points.

Description

The last category gives descriptions of what the system can do, its dataset including information about the features and the model. These methods can give the user a good introduction to what exactly is possible with the system as well as seeing what kind of data and model the system is working with.

2.3.3. LLMs

In the TTM paper, three different methods were used to translate the utterances into parses: Nearest neighbor, three different GPT models (Wang and Komatsuzaki, 2022) and a T5 model in three different sizes (Raffel et al., 2020). The T5 model acquired the best results in their paper but requires fine-tuning and therefore takes a little longer to set up. Also, the three different sizes of T5 model do not differ a lot in terms of performance. After testing these models ourselves, it was found that the T5 models perform much better and that indeed the different sizes of the model do not affect the performance much. Thus the T5 small model is the model used for the enhanced TalkToModel system.

3

Methods

This chapter first describes how TalkToModel was altered to handle (multiple) GP models. Next, to further develop the system ideas were gauged for features and interest for the system by doing user interviews. Finally, the methods chapter covers the setup of the evaluation experiment that was performed to evaluate TalkToGP by letting participants use the system.

3.1. TalkToGP

The original TalkToModel system has to be altered to support GP models. This section covers all the changes and additions to the original system. First, the switch from classification to regression models will be explained, followed by the techniques used to add new questions. After that, the change to support multiple models, the addition of new specific GP questions, and finally, the changes made to the original set of questions will be explained.

3.1.1. GP models in TalkToModel

The first problem with the original TalkToModel system is that it only supports classification models, and since this research wants the GP models to perform symbolic regression, the system needs to be changed to support this. To help support regression models, a fork of the original TalkToModel repository is used, called TalkToModel-TrustAI, which introduces the possibility of using regression models (Krkv, 2023). This repository includes changes allowing users to use regression models instead of classification models. Even though not all functionalities are implemented, such as predicting data, it provides a good starting point.

The next step is to load GP models into the system. The original TalkToModel system only accepts Sklearn models (Pedregosa et al., 2011). Since GP models can simply be expressed by mathematical expressions, these expressions were embedded in a Sklearn base class. A GPModel class that extends the Sklearn base estimator is created and this class has a string of a mathematical expression as a parameter. Using SymPY (Meurer et al., 2017) this expression is parsed and lambdified such that it can be used for numeric evaluation. Lambdifying is the process of transforming expressions into lambda functions which can be used to calculate numerical values quickly. The only other requirement for the class was that a predict method was defined. This method is to predict the value of data points. It takes one or more data instances as input and outputs the corresponding numbers the model predicts. This is done using the lambdified expression.

3.1.2. Extending the Question Set

The system can now be extended beyond just black-box questions. Since the GPModel class now has access to the expression, the system can be extended to ask questions about the internal structure and functioning of the model. First, a set of simple questions was made. This included questions like *"How many nodes does the model have?"* or *"What operators does the model have?"*

As mentioned in Chapter 2, TalkToModel uses an LLM to parse user utterances into a grammar the system understands. This grammar uses words that are linked to an operation. In order to add new questions to the system four steps have to be performed:

1. Adding new prompts
2. Extending the grammar
3. Writing execution code
4. Training the LLM

In the rest of this section, it will be explained how those four steps are performed to add new questions to the system.

Adding prompts

First, new user utterance-parse pairs have to be created. These are stored in a file in the form of:

```
User: Select models containing the * operator and show their features.
Parsed: select selectop * and featuresget [E]
```

```
User: plot the approximation front
Parsed: approximationplot [E]
```

Each pair consists of a user and a parsed part. In step 4 the LLM uses the pairs as examples during finetuning to see how to translate certain user input into the grammar. This way the LLM can learn how to parse different questions. Because a pretrained LLM is used, which is trained on general natural language tasks, the LLM has a general language understanding. This means that the users do not have to type the questions as specified by the pairs exactly, because the LLM can understand that their question has a lot of similarity with an existing user-utterance-parse pair. When the LLM is fine-tuned, the user can type a question and TalkToModel can translate it to grammar.

The first pair in the example above is for filtering models with a multiplication operator and showing the features present in those models. Here the * means the multiplication operator. If the user uses a different operator, for example -, the system would understand that it now has to filter on the minus operator.

The TalkToModel paper states that the more pairs are added, the better the performance of questions will be. To help generate more pairs ChatGPT (OpenAI, 2022) is used. First, one to three pairs without the help of ChatGPT are created. After ChatGPT is given the pairs it is explained to ChatGPT what the goal and use of the pairs is. ChatGPT is then asked to generate 10-20 more pairs depending on the question. Some questions are more complex and thus require more prompts to be generated. In figure 3.1 you can see an example prompt used in ChatGPT to generate more user utterance-parse pairs.

Extending the grammar

```
plotsubtree: " subtreeplot"
deletenode: " nodedelete" adhocnumvalues
```

The next step is to extend the existing grammar. This is done by adding new operations in the grammar. An example of two operations is shown above. They consist of three parts:

1. The first part is the operation name which in these cases are *plotsubtree*. and *deletenode*.
2. Next are the action words. These action words are mapped to a specific operation the system can execute. In the examples above they are " *subtreeplot*" and " *nodedelete*".
3. Aside from this, the more complex questions also have other parameters that have to be specified in the grammar. The "deletenode" operation for example is specified as: " nodedelete" adhocnumvalues. Where nodelete is the action word and adhocnumvalues is a parameter for this operation and is allowed to be any integer. This way the nodenumber that has to be deleted can be found.

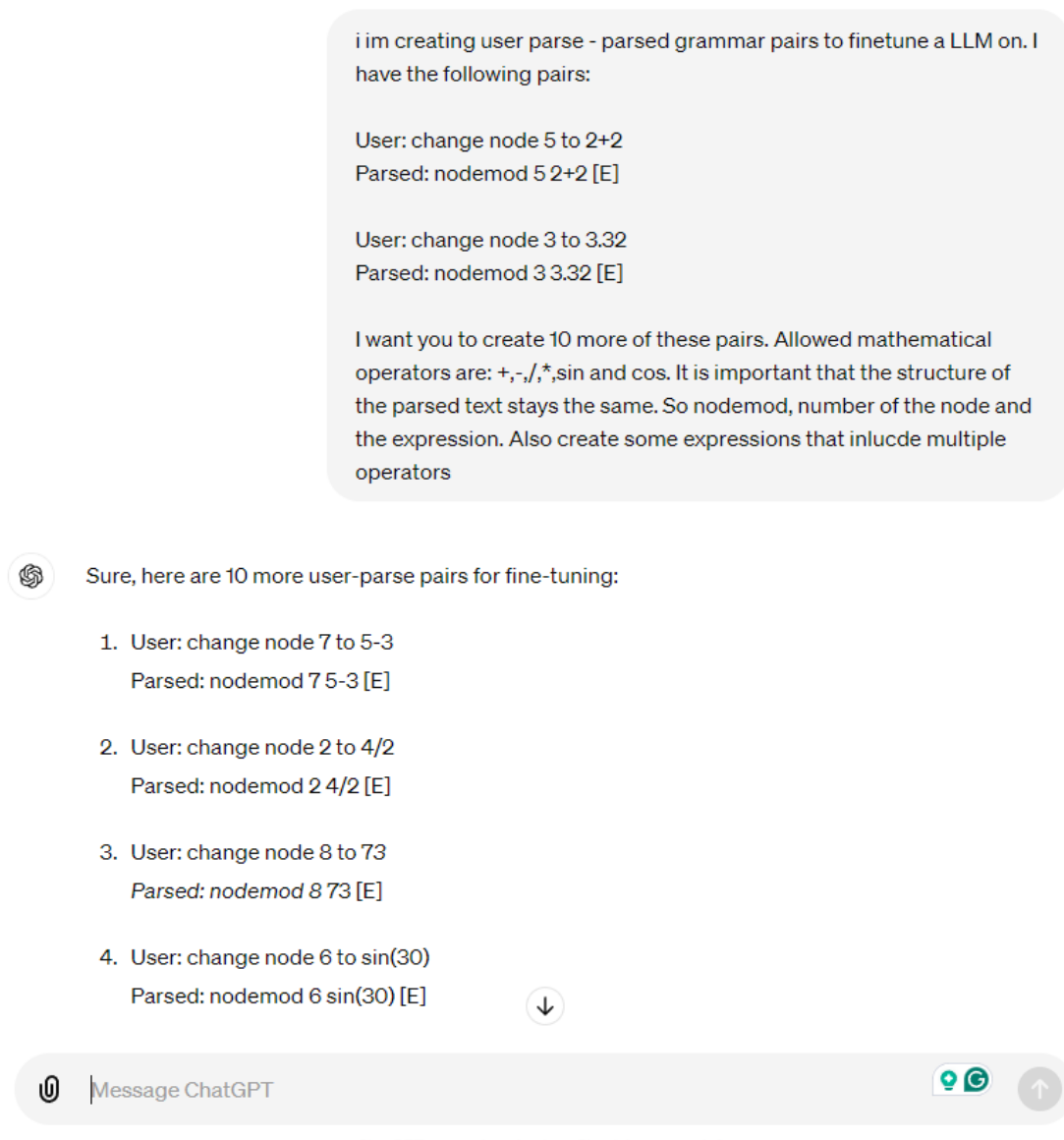


Figure 3.1: A screenshot asking ChatGPT to generate more user utterance-parse pairs

Writing the execution code

Furthermore, to add a question, some new execution code has to be written. This is done by first mapping the action word to the operation. In this operation, any other information that is of interest is parsed (for example, on which operator to filter). Then, using all parsed information, the code is executed, which usually involves getting some information from the GPModel class such as its number of nodes or getting all subtrees of the expression. These results are then formatted into a string or image which is then displayed by the system. Because of this way of answering questions, the system is unable to hallucinate answers. This means it is possible to validate that the system's answers are correct.

Training the LLM

Lastly, for the system to be able to answer new questions, the T5 model has to be fine-tuned. As mentioned in Chapter 2, the T5 model is a pretrained model that comes in three sizes: small, base and large. While experimenting with the different sizes, the conclusion was drawn that the small size has acceptable performance. This model is chosen because of its much quicker training time.

3.1.3. Multiple Models

Another issue is that the original TalkToModel system only supports a single model, however, in order to compare different GP models, multiple models should be loaded at once.

This thesis is not made with a specific GP algorithm in mind. However, MO-GP is a very good fit to evaluate the system, since this gives a front of GP models instead of just one. Specifically, Multi-Objective Multi-Modal GOMEA (MM-MO-GOMEA) is used (Sijben et al., 2022). With multi-objective GP the individuals do not only get optimized for accuracy but also for complexity. The complexity of individuals can be measured in several ways but in this thesis, it was opted to go with the number of nodes. This means the lower the number of nodes the lower the complexity will be.

The system wants to be able to compare multiple models as in MM-MO-GOMEA not only one model but a front of models is returned. This way a user can ask questions about multiple models simultaneously and compare them. To enable this the system was changed to instead of loading a single model to load and save multiple models. This caused the system to be changed in multiple ways to adapt to these changes. This is discussed in more detail in section 3.1.5

3.1.4. New Questions

Now that there is an option to add new questions and to load in multiple GP models, new GP-related questions can be made as well as questions specific to multiple models. Table 3.1 contains an overview of all the new questions that were added to the system. These questions are separated into four different categories:

- GP info
- Graphic
- Tree modification
- Select

Each of the categories will be briefly explained in terms of what kind of questions they contain and why they are added to the system.

GP Info

This category of questions contains questions that give information about the GP models that are loaded into the system. Most of them have only the set of models as their parameter. All these questions look at the current set of models and answer their particular questions. All questions give some information that is useful when analyzing a model, such as how many nodes are present in each model. The function of these questions is to give the user a quick and intuitive way to inspect the models to get to know them better. This allows a user to ask: "Can you show me the node sizes of the models" which would return the respective sizes of the models.

The big advantage of these questions is that users can get an answer to these questions in seconds. If someone had to analyze a model themselves, even if it is something simple like counting the number of nodes, it takes a while to figure out the answer to these questions especially if you have a front of models. Also doing it by hand is more error-prone while this system can give a guarantee that the answer is correct. This makes it a much easier and more reliable process than if you had to analyze these models by hand.

Graphic

This category contains questions that return a graphic as an answer. There are four questions in this category: `plot_tree`, `plot_approximation_front`, `show_effect` and `set_feature`. Both `plot_tree` and `plot_approximation_front` operations have only the models as a parameter, nothing else has to be specified. For the `show_effect` operation also a feature and node have to be specified. The advantage of the plots is that they can make information much easier to understand through visualization. Even though there are only 4 questions here now, this could be easily expanded upon in the future. Figure 3.2 shows what the `plot_tree` operation looks like and figure 3.3 show the `show_effect` operation.

	operation, arguments, and description
GP Info	<code>num_ops(models)</code> : Gives for every model the number of operators present in the model
	<code>get_ops(models)</code> : Gives for every model the each operator present in the model
	<code>num_nodes(models)</code> : Gives for every model the number of nodes present in the model
	<code>num_features(models)</code> : Gives for every model the number of features present in the model
	<code>get_features(models)</code> : Gives for every model each feature present in the model
	<code>most_common_features(models)</code> : Gives a list of the most common features across each model
	<code>get_expressions(models)</code> : Lists all the expressions for each model
	<code>common_trees(models)</code> : Lists the most common sub-trees across each model together with the number of occurrences
	<code>outliers(models, percentile=99)</code> : Returns for each selected model the worst predictions from a specified percentile. If no percentile is specified it defaults to 99.
	<code>most_important_subtrees(model)</code> : Shows a ranking of the most important sub-trees in the model
<code>bad_trees(model)</code> : Shows a ranking of nodes based on how little they affect the score when they would be removed from the model	
Graphic	<code>plot_tree(models)</code> : Plots the expression tree for each model
	<code>plot_approximation_front(models)</code> : Plots the approximation front containing the accuracy and complexity for each model
	<code>show_effect(model, feature, node)</code> : Plots a graph showing the effect of a feature on a particular node in the tree. Sets all other features to their average unless there are set to a value using <code>set_feature</code>
	<code>set_feature(model, feature, value)</code> : Sets a feature to a particular value. These values are used when using <code>show_effect</code> .
Tree mod	<code>mod_node(model, nodenumber, math_expression)</code> : Changes the node which corresponds with the nodenumber of a specified model to the mathematical expression given
	<code>delete_node(model, nodenumber)</code> : Deletes the node which corresponds to the nodenumbers of a specified model
	<code>simplify(models)</code> : Simplifies all the expressions of the selected models such that the new expressions correspond to the old ones
	<code>revert(model)</code> : Reverts all the changes made to a model and returns it to its original state
Select	<code>select(models, operation, value, comparison)</code> : Filters the set models by filtering on a specific operation which is then checked by a particular value and comparison.
	Operations include: accuracy, complexity, number of constants, number of features, id, number of nodes, operation, number of operators, all models, feature present

Table 3.1: Overview of all operations added to support GP models. The table is divided into four categories of questions: 1) GP Info, 2) Graphic, 3) Tree Modification and 4) Select questions. Each line consists of the operation name, the parameters each operation takes and the description of that operation. Each operation is coloured blue, each parameter red and the description is coloured black.

TalkToModel

select model 6 and show tree

Selected model(s): 6

I use a *Multiple GP models* model to predict the price of a house.

```
graph TD; Sub_0((Sub_0)) --> 87694.852_1((87694.852_1)); Sub_0 --> Mult_2((Mult_2)); Mult_2 --> Min_3((Min_3)); Mult_2 --> median_income_5((median_income_5)); Min_3 --> 33431.155_4((33431.155_4));
```

6) $87694.852 - (-33431.155) * \text{median_income}$

Feedback

Enter your command! Use the ↑ arrow and ↓ arrow to cycle previous commands. Send

Help me generate a question about...

Figure 3.2: A screenshot of the tree graphic. The format for each node here is `value_nodenum`. The nodes are ordered by preorder traversal. This means the tree is traversed recursively by first visiting the root, then the left node and then the right node.

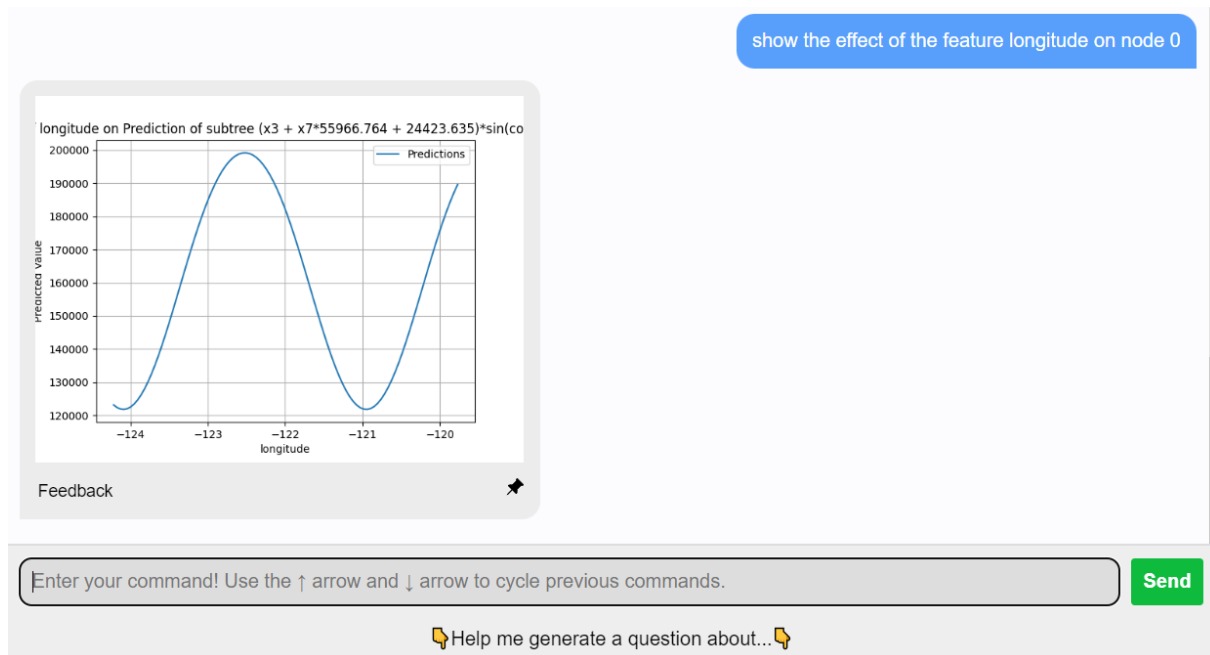


Figure 3.3: A screenshot of the show_effect operation. This operation returns a graph that shows the effect of a feature on the outcome of the node specified. On the x-axis is the value of the feature; this starts at the minimum value of the feature and goes until the maximum of the feature. On the y-axis is the resulting value of the specified node.

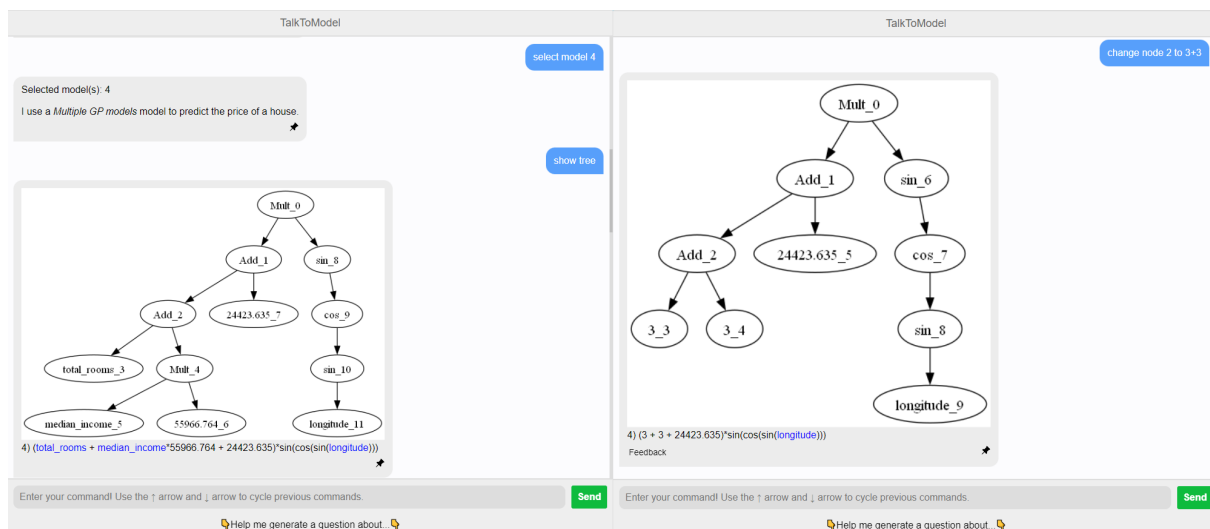


Figure 3.4: A screenshot of the tree modification operation. On the left a model is selected and shown as a tree. On the right this model gets modified by changing node 2 to 3+3.

Tree Modification

These questions allow the user to change models. When seeing the GP expressions as a tree it becomes intuitive to change things. These operations allow users to do that.

The operation `mod_node` takes a model, nodenumber and a math expression as input. The model is the model that will be changed, the nodenumber corresponds to a node in the tree. These nodenumbers are visible when the tree is plotted, as can be seen in figure 3.2. Lastly, the subtree rooted at the node specified will be changed to the mathematical expression specified, creating a new model that overwrites the old tree.

The operation `node_delete` works similarly but doesn't change the node but instead deletes the whole subtree rooted at the specified node. These functions allow a user to interact with the nodes of models and see what effects this has on the predictions and other explanations. This can potentially increase the understanding of the user of the models.

These operations also allow us to have a human in the loop. If there is for example some prior knowledge about a variable a user could apply this knowledge and modify the model to contain this information. This way it is possible to combine the power of GP with the knowledge of a subject that might already exist.

Select

The last category is select questions. These questions allow users to select a subset of models from the original set. The operation takes a set of models, an operation, a value and a comparison operator. There are several operations specified here, such as the accuracy, number of constants or the id of a model.

If applicable, the comparison operator will compare with the value to find which models satisfy the condition. The models are then added to the current subset and when concatenated with a different question will only answer for this set of models. For example, a user can ask: "For models with less than 15 nodes predict all data instances". The system would then select all models that fit the criteria of having less than 15 nodes and would then execute the predict operation.

These "select" operations can help a user explore the models, especially if there are a lot of models or if the user is only interested in models with specific characteristics.

3.1.5. Fixing Original TalkToModel Questions

Apart from the newly added questions, the original black-box questions in the TalkToModel system could also be very useful in the new system. Not all questions still worked while other questions still worked but were not properly changed to support regressions or multiple models. In this section, an overview is given of the most important changes that were conducted to adjust the system compared to the original operation in table 2.1.

In table 3.2 you can see an overview of what happened to each of the original questions. There is one of three possible situations for each question: Changed, Removed or stayed the same. The changed questions have been altered in order to either support multiple models or to support regression models instead of classification models. The removed questions also did not work as intended anymore just as the changed questions. However, for some of these questions, it did not make sense to add them again or it was decided against adding them because of the complexity. Lastly, the questions that stayed the same, still worked as intended and did not have to be altered. Next, the justification as to why these questions are either changed or removed is discussed.

It was decided to remove the `cfe` operation. Out of the box, the `cfe` operation only works with classification models, this means our symbolic regression GP models are not supported. It was decided that this was not the focus of this thesis, therefore the question was left out.

In the original TalkToModel system, the `topk` and `importance` questions only support a single model. These importances are calculated and then cached when loading a new model in the system. The system had to be extended to allow for multiple models.

The `mistakes` question looks for patterns in the mistakes of the model. This question was made to support classification models therefore it did not work for our models. This got replaced by a new operation which is in table 3.1 called "outliers", this operation looks for the worst predictions, which means it looks at where the model makes mistakes.

The `predict` operation still worked but it was set to output the results of each label. In the case of a classification problem, this is a small set. However, in the regression version, each output was consid-

	Changed, Removed, Stayed the same
Data	filter(dataset, feature, value, comparison): filters dataset by using value and comparison operator
	change(dataset, feature, value, variation): Changes dataset by increasing, decreasing, or setting feature by value
	show(list): Shows items in list in the conversation
	statistic(dataset, metric, feature): Computes summary statistic for feature
	count(list): Length of list
	and(op1, op2): Logical “and” of two operations
Explainability	or(op1, op2): Logical “or” of two operations
	explain(dataset, method, class=predicted): Feature importances on dataset
	cfe(dataset, number, class=opposite): Gets number counterfactual explanations
	topk(dataset, k): Top k most important features
	important(dataset, feature): Importance ranking of feature
	interaction(dataset): Interaction effects between features
ML	mistakes(dataset): Patterns in the model's errors on dataset
	predict(models, dataset): Model predictions on dataset
	likelihood(dataset): Prediction probabilities on dataset
	incorrect(dataset): Incorrect predictions
Conv	score(models, dataset, metric): Scores the model with metric
	prev_filter(conversation): Gets last filters
	prev_operation(conversation): Gets last non-filtering operations
Description	followup(conversation): Respond to system followups
	function(): Overview of the system's capabilities
	data(dataset): Summary of dataset
	model(): Description of model
	define(term): Defines term

Table 3.2: Overview of what happened to all operations supported out of the box in TalkToModel to be used in TalkToGP. Blue means that the operation got changed. Red mean that the operation got removed. Black means that the stayed the same.

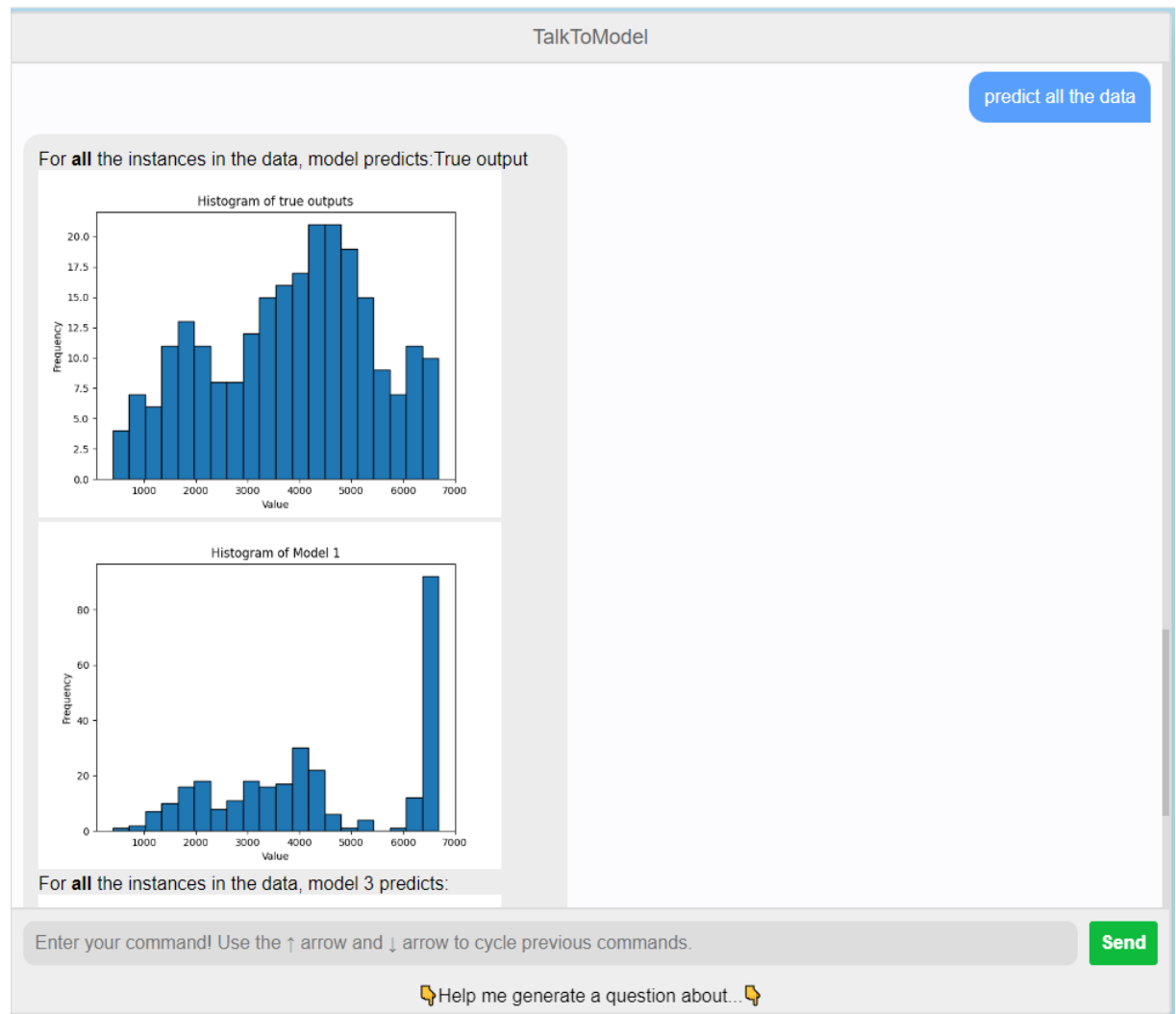


Figure 3.5: A screenshot of changed predict operation

ered a label, which meant there was the same amount of labels as data points. On top of that, there are also multiple models that have to predict. This there are means amount_of_datapoints*amount_of_models outputs. This, of course, is infeasible to output. Instead, it was changed to output a histogram of the predictions, when predicting a single value the operation is kept the same and it just outputs the value and not a histogram. In figure 3.5 you can see what the operation looks like. Even though you now do not see exactly which data point is predicted to which value, this gives a good overview of what predicting all data does without cluttering the whole system with hundreds, potentially thousands of outputs.

The likelihood and incorrect operations are removed because they only make sense for classification models.

The score operation only supported scores for classification models such as the accuracy and f1 score. This again did not work for our regression models, this was changed to support the mean squared error and the R^2 score.

Apart from this, some of the questions have had some formatting changes made to better fit our new setting. New questions can easily be added to the system if deemed necessary. Overall the new questions discussed in section 3.1.4 and the original questions discussed in this section provide a good set of questions to gauge the system's effectiveness.

3.1.6. Code availability TalkToGP

The enhanced TalkToModel system described in this section is available at

<https://github.com/nealsweijen61/TalkToGP>. Here you can install the system locally on your system.

3.2. Exploratory interview

Before finishing the application a preliminary experiment was conducted. This was done in the form of an interview with people for whom the TalkToModel system is potentially useful. The goal of this experiment was to understand what potential users would find important in our system as well as to gauge their interest.

Some things the interviews want to find out are: what kind of features do the users find important, find out whether they are interested and find out what they will use it for. To achieve this goal, an interview with 10 questions was created which can be found in the appendix. The interview questions consisted of 2 parts. The first part is general questions about trusting, difficulties and understanding models. The second part consists of questions about the TalkToModel system.

In total the interviews were conducted with 5 different people, all of whom had a background and some experience with GP. The interviews were conducted on Zoom and the audio was recorded there. From the audio files, a transcript was generated by using Open AI Whisper (Radford et al., 2023).

After getting the data from the participants, this had to be evaluated. After this reflective thematic analysis (Braun and Clarke, 2006) was performed since this was the most fitting for this case because of its flexibility. This analysis was conducted using (Byrne, 2022), this work shows a worked example of thematic analysis, which was followed during the analysis of this experiment.

3.3. Experiment

After the exploratory interview, the system was altered to take into account what the participants said that they liked in TalkToModel as well as implemented relevant suggested features obtained from this experiment.

To evaluate how this system performs, an experiment is conducted. In this experiment, users are asked to do two tasks, one with the TalkToModel system and one without. During this experiment, the participants are asked to analyze a set of models and afterward answer some questions about their experience using the two systems.

3.3.1. Objectives

This experiment has several objectives. First and foremost, it's important to determine whether the system can indeed **make GP models more interpretable**. Additionally, it's important to determine whether the system is easy to use, how well it performs, and whether users enjoy using it. Since the system was built with these goals in mind and it was built using the feedback of the participants from

the experiment discussed in section 3.2, the system is expected to achieve these objectives.

3.3.2. Experimental design

In this section, it will be explained how and why the experiment was designed.

As mentioned, the experiment contains two tasks for each user. Both tasks have the same goal, only the way the user can work towards this goal is different. The goal is to select a single GP model from a set of 12 GP models trained on the same dataset. The two tasks have a different set of models trained on a different dataset. For both datasets, 12 models were trained using a MO-GP version of GOMEA (Sijben et al., 2022) and came from one training session. The goal for the user is to select the model they think is best for each of the two tasks. The participants are not bound by any restrictions while working on the tasks as long as they are performing the task using the respective system.

Datapoints

In our experiment three categories of data points will be evaluated; metrics, answers to questions and behavioral results. The metrics consist of which participant choose what model as well as the time each participant spent on both tasks. The answers the participants give to the three questions will be qualitatively analyzed. Lastly, the way users interact with the two systems will be analyzed.

Datasets

As mentioned, the experiment uses two different datasets. The California housing dataset (Pace and Barry, 1997) and the bike sharing dataset (Fanaee-T and Gama, 2014).

The California housing dataset contains all kinds of information about blocks of houses such as the longitude, the average income and the average number of houses among others. The independent variable here is the median house price. The bike-sharing dataset contains the daily bike rentals for the years 2011 and 2012. Aside from the amount of bike rentals each day has features for the weather and seasonal information.

These two datasets were chosen because they are easy to understand. Since the participants have to do 2 tasks and time is limited, it is useful if the users do not have to spend too much time understanding the dataset. Also because it is desired to simulate a real-life situation where the users know their problem but do not know their models yet. Making the datasets easy to understand achieves this.

It was decided to go for two datasets for two reasons. The first reason is that this way the user does not get used to one dataset. If the user had the same dataset for both tasks, the second task could be easier to do since the user already has more knowledge. This could influence our results. Secondly this way different users can use the same systems with different datasets. This way the answers of the users are less dependent on the datasets.

Jupyter Notebook Task

One of the tasks is executed in a Jupyter notebook (Kluyver et al., 2016). A Jupyter Notebook is an interactive web-based environment that allows you to write and execute Python code in real-time, alongside narrative text, equations, and visualizations. Here, the user joins a Google collab where a Jupyter Notebook is loaded in together with one of the two datasets. The Jupyter Notebook has three code blocks for analyzing the 12 GP models. The three codeblocks will be explained as well as their justification for being there. In figure 3.6 you can see one of the code blocks, this one is the accuracy code block.

The first is for making predictions, here a user can select any number of data points, select a model and make the model predict the data points. This prints out the true values of those data points together with the predictions of the model for the data points. This code block is there for the purpose of allowing the user to actually use the models. Since their function is to predict values it seems only natural to allow them to use them for this.

The second block prints out the expressions of all 12 models. This block is present to let the users analyze the expressions and see for example how complex they are since it was noted that when people analyze GP models they often look at the expressions. It was opted to not print out any more information such as the number of nodes or how many operators are in each expression to simulate a very basic analysis. This ensures that the user has to do most of the work themselves without relying on too many tools.

Run to test accuracy

Here you can test the accuracy, select data to test on, select a model and select which score metric to use(MSE or R2)

```
#Accuracy Score
df = pd.read_csv(f'{csv}.csv')

BEGIN_INDEX = 0
END_INDEX = df.shape[0]
MODEL = 1

#MSE = 0 R2 = 1
SCORE = 1

row_slice = slice(BEGIN_INDEX, END_INDEX)
selected_row = df.iloc[row_slice, 1:12]
y_true = df.iloc[row_slice, 12:13]
scores = []
for i, model in enumerate(models):
    y_pred = model.predict(selected_row)
    if SCORE == 0:
        score = mean_squared_error(y_true, y_pred)
    else:
        score = r2_score(y_true, y_pred)
    scores.append(score)
    print(f'{i+1}) Score:", score)
```

```
1) Score: 0.5268349381546689
2) Score: 0.7059139758706379
3) Score: 0.33160192266231003
4) Score: 0.6993668535514092
5) Score: 0.31113831007820225
6) Score: 0.5165689375573834
7) Score: 0.6682924776308712
8) Score: 0.2850590676977848
9) Score: 0.6100482082775174
10) Score: -0.23934812519040483
11) Score: 0.6919388174221472
12) Score: 0.6560658325868958
```

Figure 3.6: A screenshot of the accuracy code block

The last block outputs the test scores for all the models. By default, the score takes into consideration all the data points but the user is allowed to select a subset of the data points. Aside from this, the user can also select between either the mean squared error (MSE) or the R2 score. This block is also present because of the exploratory experiment where users indicated that performance was one of the most important metrics they look at when analyzing a model.

It was decided to do one task with the Jupyter Notebook as there was no real other alternative for TalkToModel. During the interviews discussed in section 3.2 it was found that most people analyze their models by simply looking at them if they would analyze at all. To capture that idea the Notebook was used. It was deliberately kept very simple but allowed the experiment to have a controlled experiment where the environment for each participant was the same.

Talk To Model Task

The other task is to be performed in TalkToModel. Here the user gets all the functionality of the GP-adjusted TalkToModel as described in section 3.1.

3.3.3. Procedure

The two tasks are performed with two different datasets one for each task. The dataset that is used for the tasks as well as the order of the tasks is evenly distributed such that a fair comparison can be made between the two.

Before the tasks start, the overall structure of the interview is explained and a short introduction to TalkToModel is given.

Before the Jupyter Notebook task is performed a short introduction for the notebook will be given. Here it is explained how they can get it up and running. As well as what the function of each block is. It is also explained that the user is free to use anything or nothing at all when using this task, the blocks are not mandatory to use.

Before the TalkToModel task is performed an introduction for the system is given. Here the user is shown how to use the system. This is done by showing some example prompts and making a small conversation. Apart from this "the suggest me a question" buttons as seen in figure 2.4 are also shown and it is explained that this is a good point to start if you do not know what to ask.

During both tasks, the time is recorded which is stopped when the user decides on what model to choose. After they have decided on the model, they get asked a couple of questions about their decision.

Finally, after finishing both tasks, the interviewees are asked a last set of questions. These questions are to compare the two systems and see what the user prefers using and whether they think TalkToModel increases the interpretability of GP models.

4

Results

In this section, the results of the experiments in section 3.3 are presented and discussed. The results are divided into three parts: first, the numerical results will be analyzed, next the qualitative data will be analyzed and finally the observed behaviour of the users will be presented.

4.1. Results Evaluation Experiment

4.1.1. Model chosen

First, the models that the users have chosen are analyzed. In table 4.2 the model that each participant choose for both tasks is presented. Moreover, in table 4.1 the order of the tasks of the participants as well as which data set was used in which task can be seen. In this section, the models that were chosen and the reasons of the participants for choosing each model are discussed.

Before analyzing these results any further, one important data point has to be discussed. In table 4.2 you can see that one of the participants could not decide upon a model at all. The participant spent a lot of time deciding but eventually drew the conclusion that no model was good enough.

When looking at table 4.2 you can see that out of the 12 models, a lot of the same models were chosen: people often preferred the same models. This is most prevalent in the Bike Daily dataset where of the 12 models during the 6 experiments only 2 different models were chosen. Even though in the California Housing dataset this happened less, still only 4 different models in total were chosen. The system used seems to have little impact on the consistency here and is more impacted by the dataset.

Participants 1, 2 and 3 all chose the same model with Jupyter Notebook and between the three of them, 2 different models with TalkToModel. Participants 4, 5 and 6 choose 2 different models when using the notebook and 1 when using TalkToModel. Even though the system used might not have an impact on the consistency, when looking at the data it can be observed that the people who used a particular system for a dataset tended to choose the same model as each other. For both systems, the choice of the model seemed consistent but the type of system did not seem to increase the consistency. Thus, the type of system used might have an impact on which model is chosen. This could however also be explained by what people find important in a model since the two groups differed in how they analyzed the models.

There was a difference observed in how the selection was done by participants based on the system they used. As will be explained in the next section participants 4,5 and 6 all spend little time on the notebook task, quickly making their decision. When these participants did the TalkToModel task, they spent more time analyzing all the different models. This could be because they found it easier to analyze and they felt there were more options to analyze. This is also in line with what the participants said themselves, which will be discussed in section 4.2; they said that TalkToModel makes the analysis easier and better because there were more options.

4.1.2. Time spent

In tables 4.3, 4.4, and 4.5, the times it took to complete each task are shown together with the average and standard deviation. As mentioned in section 4.1.1, one participant decided not to choose any

Participant	First Task	Second Task
Participant 1	Notebook Bikes	TalkToModel House
Participant 2	TalkToModel House	Notebook Bikes
Participant 3	TalkToModel House	Notebook bikes
Participant 4	Notebook House	TalkToModel Bike
Participant 5	TalkToModel Bike	Notebook house
Participant 6	Notebook House	TalkToModel Bike

Table 4.1: Table to see which tasks the participants did and in which order

Participant	Bike model chosen	House model chosen
Participant 1	2	7
Participant 2	2	12
Participant 3	x	7
Participant 4	9	9
Participant 5	9	4
Participant 6	9	4

Table 4.2: Table that shows the model that each participant chose for each of the two tasks

model when using the Jupyter Notebook, and the task was stopped without them choosing any model. This means that the time noted can be considered meaningless since the task was not completed. The time noted is a big outlier because of this reason the decision was made to also the analysis without the value. This leads to a more meaningful analysis.

System

Table 4.3 shows that, on average, more time is spent on the TalkToModel task. What is also interesting to note is that the system used is the only variable that caused all the participants to spend more time (when not considering the participant's 3 notebook task). One reason for this is that the participants all needed a little time to get used to the TalkToModel system as this was their first time using it. Whereas the Notebook is very straightforward in its capabilities and the participants were only given three code blocks to execute. Another reason for this difference is that it was observed that especially for participants 4,5 and 6, the effort spent analyzing increased when using the TalkToModel system.

Dataset

In table 4.4 it can be seen in the altered average time that average times are very similar. Here it can be seen that the participants who had their dataset on the TalkToModel system spent more time on that dataset. It could be argued from this data that the dataset had little to do with their time spent.

Order

Also here in table 4.5 there is not a significant difference in the times spent between the first and second tasks meaning that the order of tasks did not have a big influence on their time spent.

Bike rental Dataset

1. $(58.520 * 59.715)$
2. $((57.795 + ((x1 + x7) * 58.646)) * 40.257)$
3. $(((((38.145 * 26.640) - (x2 * -56.715)) * ((x1 + x7) + \sin(x6))) / (\sin(\cos(\cos(x10))) + 1e-6))$
4. $((50.720 * 57.785) / (\sin(\cos(x1)) + 1e-6))$
5. $((57.795 * (x0 + 53.218)) / (\cos(x1) + 1e-6))$

Participant	Time spent on Talk-ToModel	Time spent on Jupyter notebook
Participant 1	16:50	12:58
Participant 2	16:00	10:22
Participant 3	19:11	26:56
Participant 4	17:30	4:28
Participant 5	5:00	2:22
Participant 6	13:41	4:50
Average time	14:42	10:19
Standard deviation	4.642	8.279
Average time without 3 notebook	14:42	6:48
Standard deviation	4.642	3.994

Table 4.3: Table that shows the time each participant took to complete each of the two tasks. The two columns are separated by task, in the first column is the time it took for the TalkToModel task, and the second column is for the Jupyter Notebook task.

Participant	Time spend on Bike	Time spend on House
Participant 1	12:58	16:50
Participant 2	10:22	16:00
Participant 3	26:56	19:11
Participant 4	17:30	4:28
Participant 5	5:00	2:22
Participant 6	13:41	4:50
Average time	14:24	10:36
Standard deviation	6.754	6.834
Average time	11:54	10:36
Standard deviation	4.171	6.834

Table 4.4: Table that shows the time each participant took to complete each of the two tasks now ordered on the dataset. The two columns are separated by dataset; here, the first column is for the bikes dataset, and the second is for the housing dataset.

Participant	Time spent on First task	Time spent on second Task
Participant 1	12:58	16:50
Participant 2	16:00	10:22
Participant 3	19:11	26:56
Participant 4	4:28	17:30
Participant 5	5:00	2:22
Participant 6	4:50	13:41
Average time	10:24	14:36
Standard deviation	5.924	7.46
Average time	10:24	12:09
Standard deviation	5.924	5.156

Table 4.5: Table that shows the time each participant took to complete each of the two tasks now ordered on task order. Here the two columns are separated by the order in which they did the tasks; now, the first column is for the first task, and the second is for the second task.

6. $((57.795 * ((x1 + x7) * 58.462)) / (\cos(\cos(x1)) + 1e-6))$
7. $((((-48.589 * 27.965) / (\sin(-53.204) + 1e6)) * \sin((x1 + x7))) * \cos((x1 / (x0 + 1e-6))))$
8. $((((-48.589 * 27.965) / (\sin(-53.204) + 1e-6)) * \sin((x1 + x7))) * \cos((x1 / ((x0 * x0) + 1e-6))))$
9. $((((-48.589 * 27.965) / (\sin(-53.204) + 1e-6)) * \sin(x8)) / (\cos(x1) + 1e-6))$
10. $((((-48.589 * 26.640) / (\sin(-53.204) + 1e-6)) * \sin(x8)) / (\sin(\cos(x1)) + 1e-6))$

11. "((((-48.589*27.965) / (sin(-53.204)+1e-6)) *sin((x1+x7))) *cos(((x1-x8) / ((0.306-x0)+1e-6))))"
12. "((57.795*56.798) / (cos(x1)+1e-6))"

Housing Dataset

1. "(128038.565-65500.791)"
2. "(90747.418*sqrt(x7))"
3. "(87694.852-(x7*-33431.155))"
4. "((((x4*x2)+(x7*55966.764))+16510.250)*sin(cos(sin(x0))))"
5. "((((x3+(x7*55966.764))+((88382.076/ (x7+1e-6))+x4)) *sin(cos(sin(x0))))"
6. "((((x4*x2)+(x7*55966.764))+((60840.333/ (x7+1e-6))+x4)) *sin(cos(sin(x0))))"
7. "((38714.865-6404.511)+(x7*41230.930))"
8. "((((x4*x2)+(x7*55966.764))+((59111.773/ (x7+1e-6))+(x2*x2))) *sin(cos(sin(x0))))"
9. "(x0-62467.385)-(x7*-33431.155)"
10. "((((x6/ (x5+1e-6)) *139366.254) -56759.704) +((((42076.555*x7) +x5) - (x2*(119829.645/(x0+1e-6)))))"
11. "((((x6/ (x5+1e-6)) *139366.254) -60222.572) +((((42076.555*x7) (x6*x2)) - (x2*(88422.150/(x0+1e-6)))))"
12. "((((x3+(x7*55966.764)) +24423.635) *sin(cos(sin(x0))))"

4.2. Results questions

This section presents the results of the questions the participants were asked after doing both tasks. These questions are all comparisons between the two systems. The three questions asked were:

1. Which system did you find easier to use and why?
2. Did one of the two systems make the models more interpretable?
3. Which of these 2 systems would you use and why?

For each of these questions, representative quotes are provided to illustrate common themes in the responses.

4.2.1. Easy of use

The first question the participants were asked was: Which system did you find easier to use and why? Most people agreed that TalkToModel was the easier system to use. One participant said:

"It is more intuitive, easier and faster. Otherwise, I would have to do it manually"

This opinion was more or less shared by all participants. One participant who was used to coding this analysis themselves did say that for now the notebook was easier. But in time the TalkToModel would probably be easier since they would not have to do everything themselves.

4.2.2. Interpretability

The second question was: Did one of the two systems make the models more interpretable? Everyone agreed that TMM made the models easier to interpret than the notebook.

One participant said:

"TalkToModel is easier to interpret because of the nice way of asking questions."

Another person said:

"Talk To Model was easier to interpret because of the help of things such as the visualizations."

4.2.3. Most likely to use

The last question was: Which of these two systems would you use yourself, and why?

Most participants agreed that TalkToModel is the system they would use over the Jupyter Notebook, with some added remarks. Some would use it without a doubt, while others still required more information. As the participants did not use the system very long, there are still some unknowns to them.

One participant said:

"I would use TalkToModel but I would have to try for a while and get better at it."

For one participant it was also dependent on what kind of information they needed and what the system could provide. Since in the Notebook, you could in theory get every bit of information the TalkToModel system can also provide. Even though implementing this yourself in the Notebook can be tedious.

4.3. Results behaviour

Lastly, the behaviour of the participants will be analyzed. In this section, the patterns and other interesting things that were observed are discussed.

4.3.1. Jupyter Notebook

The use of the Jupyter Notebook was very different among participants. As mentioned in section 4.1.1 participants 4, 5 and 6 did not do a lot of analyzing within the notebook as they were done quickly with the task. They only looked at the accuracy of the models as well as quickly looking at the structure of the models. They made their choice without using much other than that. Participants 1,2 and 3 spent more time trying to understand the structure of the models and tried to choose one that made sense to them. This means that they used the provided code blocks as well as altered some code and made new code blocks to help them out in analyzing the models.

4.3.2. Talk To Model

When observing the people using TalkToModel also some interesting patterns can be found.

Not asking the right questions

When starting the system for the first time, the participants were all shown some difficulties with using the system. This was mainly because of the type of questions they asked. As mentioned in section 2.3.1 the system uses a large language model to translate to a grammar which defines everything that the system can execute. This causes two problems for people asking questions. The first is that people expect the system to be able to answer every question. This leads to questions giving a lot of "sorry I do not understand" responses back, since there is only a defined set of questions.

The second problem is when people do ask something that the system is able to do but the system does not translate the question into the right grammar for this question but instead another one. Or answers with another "sorry I do not understand" response. These two problems hindered people from using all the functions from the start.

It could be seen however that when participants used the "suggest me a question button" as in figure 2.4 they had a much better start. These buttons do a good job of showing a lot of the functionality of the system as well as showing how to write these questions. It could be observed that people using these buttons at the start had a much smoother experience.

Using a limited set of questions

Another interesting pattern to note was that most participants only used a subset of the functionalities and a lot were not used at all. The operations to list the models, plot a Pareto front, show the R^2 score and simplify the models were used a lot. While operations such as plotting the tree of models or asking the GP info questions were rarely used. This shows that the participants might need more time to fully grasp what the system can do. When given more time using the system the participants might have been able to use the system better.

Natural conversation

It was also observed that the users interacted very naturally with the TalkToModel system. All users could quickly come up with different questions and it was intuitive for them to use. This is probably because people are very much used to using chatbots like ChatGPT and it is a natural environment for them. This natural way of conversing could also cause some problems for the system answering the questions as was also mentioned earlier in this section.

5

Discussion

This section discusses the results and whether the research question posed in section 1.5 is answered. Afterward, the main contributions will be discussed. Lastly, some of the limitations of the research will be handled as well as some possible future research directions.

5.1. Interpretation of results

In this section, the results will be discussed and evaluated as a whole. It will explain what the results mean and how they answer the research question: **Can Genetic Programming models be made more interpretable using TalkToModel?**. As mentioned in chapter 1, in this thesis interpretable is defined as: **the ability to explain or to present in understandable terms to a human**. This experiment aims to determine whether TalkToGP enhances the ability to explain or present GP models in understandable terms to a human, compared to using a Jupyter Notebook.

The first evidence that shows that the enhanced version of TalkToModel increases interpretability is found in the time spent on the tasks. When looking at the times spent in tables 4.3, 4.4 and 4.5 it can be observed that using TalkToModel has a significant impact on the time spend for the users. It can be seen that, especially, the participants who spend little time on the Jupyter Notebook task spend much more time when performing the task using TalkToModel. Even though part of this extra time spent can be explained by the fact that the users needed a little time to get used to the system compared to the Notebook. It was also observed that these participants went more in-depth with their analysis when using TalkToModel. Whereas in the Notebook these users mostly only looked at the accuracy and the size of the models, these same users looked at more details of the models in TalkToModel. In these cases, TalkToModel stimulated the users to do more analysis and in turn, the participants got more knowledge on the models. This shows that TalkToModel increased the ability to explain the models compared to the Notebook for these users.

Secondly, most participants agreed that TalkToModel was easier to use. The participants who still doubted which of the systems was easier had confidence that TalkToModel would be easier to use if they got more experience with it. This is also something that was observed during the experiment. Where the participants who wanted to analyze something by altering an existing or creating their own codeblock in the notebook, spent quite some time on easy tasks and also often encountered multiple bugs before reaching a correct implementation. TalkToModel might also sometimes have trouble answering questions but when the users got familiar with asking the right questions, they spent very little time on similar tasks. This ease of use can lower the barrier to analyzing GP models. Especially for people inexperienced with using such models, as you do not need to understand how to implement such questions and can express yourself in natural language. This also shows increased interpretability because the participants got their explanations more easily when using TalkToModel which points to an increased ability to explain GP models.

Furthermore, the participants noted that the models were easier to interpret using TalkToModel. They said that the operations such as visualizations helped with this. This also shows that TalkToModel increases the ability to explain and present GP models because it offers more tools that the participants appreciate. Besides this, the participants also noted that the GP models were easier to interpret with

TalkToModel because of the intuitive way of asking questions. This also aligns with our observations since it was seen that the users interacted very naturally with the system. All users quickly came up with questions and the conversations were very intuitive for them to use. This also shows that TalkToModel is better at explaining GP models than the Notebook.

When comparing the two systems, it is evident that for these participants the enhanced version of TalkToModel offers superior tools for model interpretability. The participants' preference for using TalkToModel, despite their differences in analyzing the models, indicates that the system provides a more user-friendly and efficient means of analyzing GP models. These results show a strong indication that the enhanced TalkToModel system improves the interpretability of GP models. By providing an intuitive interface that supports a broad selection of questions, TalkToModel empowers users to understand and utilize GP models more effectively. The system's ability to facilitate model evaluation and comparison makes it a valuable tool for users working with GP models. This strong indication addresses our main research question, demonstrating that GP models have good potential to be made more interpretable using TalkToModel. However to conclude this further research has to be done. Especially an experiment with a higher number and more diverse users have to be done. This will be further discussed in sections 5.3 and 5.4. What can be concluded from the results is that TalkToModel lays the groundwork for a lot of potentially interesting research in the field of model interpretability.

5.2. Contributions

In this section, the main contributions are discussed as well as why we care about these contributions. As mentioned in section 1.6 there are three main contributions:

1. **Integration of GP Models with TalkToModel:** The first contribution of this thesis allows GP models to be used in TalkToModel. Previously this was not possible, as only classification models could be used. Now GP models can also be used and most of the original TalkToModel operations can be used for TalkToModel as was shown in table 3.2. This has lowered the barrier of entry for asking black-box questions to GP models.
2. **Questions into the Specifics of GP Models:** The second contribution gives the system the ability to ask GP-specific questions. Now users cannot only ask black-box questions to models but can also get answers to questions regarding the structure of GP models. These new operations can be seen in table 3.1. This has allowed users to quickly analyze a GP model and understand its components more easily by using operations such as visualizing the model as a tree.
3. **Support comparative analysis:** The last contribution of this thesis, enables the enhanced TalkToModel to use multiple models simultaneously. Instead of asking questions to one model at a time, the enhanced system instead allows a user to load in as many models as desired. This has allowed users to ask questions to multiple models simultaneously and compare different aspects of them this way. Most operations in table 3.1 can be used for multiple models. Enabling users to more easily make decisions when faced with a set of models.

5.3. Limitations

The research done does have some limitations which will be discussed in this section.

The first limitation is the limited sample size. The experiment has been conducted with 6 different GP practitioners. This means that our results are dependent on those 6 participants. This is partly because the experiments are quite timely and also require some setting up and the writer of this thesis had to be present during every interview. This means that conducting many more experiments was not feasible. Still, the limited sample size gives us good qualitative insights and shows that TalkToGP has a high potential to be further developed into a very useful tool.

In this experiment, the decision was made to only interview people who have experience with using GP models. This does however give the experiment a sampling bias and makes it unclear how the system would be used by people who do not have any experience with GP. This would be interesting to find out in a future experiment.

As mentioned in chapter 4 TalkToModel had issues with answering some questions. Moreover, during our experiment, it was the first time for all the users to try out the system. Some participants

had never even heard about TalkToModel. The results could be improved if participants had some experience with the system to see how different their use would be.

Lastly, the results of the experiment were also used to further improve TalkToGP. From the results, changes were made in some of the formatting of the questions to make certain answers more clear. Also, new ideas for operations that came from observing the participants were added. This even further improved version has not been evaluated, however.

5.4. Future work

In this section, possible directions for the systems as well as the research will be discussed.

The first recommendation would be to use a different LLM to do the translation from user parses to grammar. The T5 model that is being used while good is outdated and there are a lot of excellent other LLMs (Chung et al., 2024). It would be interesting to see if the performance of the system could be improved by using a better LLM. This improved performance would make the users of the system less restricted in the questions they are able to ask. LLMs such as llama 3 or GPT 4, would be better at getting their meaning and translating their questions to the appropriate parse in the grammar.

The second recommendation would be to look into removing the grammar system altogether from the system. As mentioned in chapter 2 the upside of having the grammar is that it does not allow hallucinations to occur which is a big advantage. However, it also limits the questions that can be asked as was observed in the experiment. Because of the rise of chatbots like ChatGPT people are used to being able to ask anything to a chatbot and when this is not possible they might get frustrated. A hybrid system, where the user can switch between grammar answers and answers without grammar, may be a viable option. This could be done by for example adding a follow-up button when the grammar can not answer a question. This button would specify clearly that the question is outside of its understanding and would propose using the full LLM without grammar. This would result in the system always answering the question. What would be important is that the button should specify that hallucinations can appear in the resulting answer. Therefore this should preferably be used for questions where this is not such a big issue. For example when explaining background information that the system does not know about such as more information about the features.

Since there were only 6 participants in the experiment that did not have a lot of experience with the application. It would be interesting to see how the experiment would change if there were more participants who had some experience with the system. This could be done by giving all participants tasks to complete with TalkToModel a month prior to to doing the experiment. We could handle more participants by letting the participants do the experiment without someone present and looking at the logs. This would be a very time-consuming process for us as well as the participants, though. This would enable a more definitive answer to the research question.

The operations available in the system do a good job of allowing the user to analyze the GP models. It would be interesting to see if even more useful operations could be found. The work in section 2.2 described how they looked at all the questions users asked to see how they used the system. These questions could later be used to improve the system iteratively. Such a system would also be valuable in TalkToModel. We could look at questions the system was not able to answer and then see what exactly the user wanted to know. This way new questions can be added that users could not get answered previously, thus iteratively improving the application

The last recommendation would be to allow the users to run GP inside of the application. Similar to the system discussed in section 2.1. The system now only allows a user to provide GP models that are trained elsewhere. It would be very useful, however, to allow the training of the models in the application itself. This could improve the ease of use tremendously for users who have no experience with GP since they would only have to provide the system with a dataset now. You could also implement different GP algorithms and analyze how they differ in their results and see how the individual algorithms change their models when altering their parameters. Something else that could be done with this is to let the user guide the evolution process. The user could, for example, be provided with all the models after a training generation. These models could be analyzed using the TalkToModel system and even altered and then given back to the training population such that users can understand and maybe even help in the training process. This way a user can guide the evolution of the models for each generation.

6

Conclusion

This thesis aimed to find out whether GP models could be made more interpretable using an enhanced version of TalkToModel called TalkToGP. In chapter 3 it was shown that our research successfully extends the capabilities of TalkToModel to handle GP models. The system not only supports black-box questions about the models but also allows users to ask about the internal workings of GP models. These capabilities enhance the users' tools to analyze GP models more in-depth. Additionally, the enhanced TalkToModel supports the evaluation and comparison of multiple GP models simultaneously, addressing the challenge of selecting the most appropriate model from a set of models generated by an MO-GP system. By doing user interviews, the system was altered to take into account relevant features obtained from this experiment. Finally, the system was evaluated by real users.

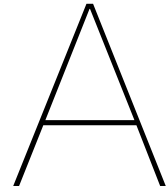
The primary research question posed in this thesis is: **Can Genetic Programming models be made more interpretable using TalkToModel?** The results of the study performed in this thesis provide an answer to this question. The enhanced TalkToModel system shows a strong indication to improve the interpretability of GP models, enabling users to better understand the behaviour of these models. The system's capabilities in supporting black-box questions and facilitating analysis of the internals of GP models highlight its effectiveness in enhancing interpretability.

This thesis has three main contributions:

1. **Integration of GP Models with TalkToModel**
2. **Questions into the Specifics of GP Models**
3. **Support comparative analysis**

Even though this application shows success, some possible improvements can be made in the future. For one, a newer, better LLM could be used to improve the application. Also, future work could look into the possibility of changing the system to allow for more flexibility in answering user questions. Furthermore, a new experiment could be performed on more users to give a more definitive answer on whether the interpretability of GP models gets increased. Lastly, the system could be altered to allow for running GP inside of TalkToModel. These improvements could help make TalkToModel an even more complete system, making it better and easier to use.

In conclusion, this thesis has demonstrated that there is a strong indication that the interpretability of GP models can be enhanced using an improved version of TalkToModel. The system's ability to provide insights into model behavior and structure, combined with its user-friendly interface, makes it potentially a valuable tool for anyone working with GP models. The contributions of this work not only answer the main research question but also lay the groundwork for future innovations in the field of model interpretability. As we continue to develop and refine tools such as TalkToModel, we move closer to making machine learning models more accessible and understandable to a broader audience, advancing the field and its applications.



Exploratory questions



A.0.1. General questions about models

1. Are you familiar with GP(genetic programming)
 - (a) Have you worked with GP models?
 - (b) What are some difficulties you encountered with GP?
2. Do you work with machine learning models in general?
3. How would you compare multiple models?
 - (a) When is an explanation of a model satisfying/sufficient to you?
4. What does trusting a model mean to you?
5. When do you 'trust' a model?
6. Do you make an effort understanding a model?
 - (a) How do you go about understanding a model?

A.0.2. Questions about the system

7. What questions would you ask to the system
8. What features would think are usefull in the system
9. What do you think this system does for the interpretability of models
10. Would you consider using this system?
11. What would you use it for?
12. Anything else you would like to add that I have not asked a question about?

Bibliography

- Biecek, P. (2018). Dalex: Explainers for complex predictive models in r. *Journal of Machine Learning Research*, 19(84), 1–5.
- Biecek, P., & Burzykowski, T. (2021). *Explanatory Model Analysis*. Chapman; Hall/CRC, New York. <https://pbiecek.github.io/ema/>
- Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative research in psychology*, 3(2), 77–101.
- Byrne, D. (2022). A worked example of braun and clarke’s approach to reflexive thematic analysis. *Quality & quantity*, 56(3), 1391–1412.
- Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., et al. (2024). Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70), 1–53.
- Dijk, O., oegasam, Bell, R., Lily, Simon-Free, Serna, B., rajgupt, yanhong-zhao ef, Gädke, A., Hugo, & Okumus, T. (2022, April). Oegedijk/explainerdashboard: V0.3.8.2: Reverses set_shap_values bug introduced in 0.3.8.1. <https://doi.org/10.5281/zenodo.6408776>
- Fanaee-T, H., & Gama, J. (2014). Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, 2, 113–127.
- Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., & Willing, C. Jupyter notebooks – a publishing format for reproducible computational workflows (F. Loizides & B. Schmidt, Eds.). In: *Positioning and power in academic publishing: Players, agents and agendas* (F. Loizides & B. Schmidt, Eds.). Ed. by Loizides, F., & Schmidt, B. IOS Press. 2016, 87–90.
- Kommenda, M., Beham, A., Affenzeller, M., & Kronberger, G. (2015). Complexity measures for multi-objective symbolic regression. *Computer Aided Systems Theory–EUROCAST 2015: 15th International Conference, Las Palmas de Gran Canaria, Spain, February 8-13, 2015, Revised Selected Papers 15*, 409–416.
- Krkv. (2023). GitHub - krkv/TalkToModel-TrustAI: Modified version of TalkToModel for the Trust-AI project  . <https://github.com/krkv/TalkToModel-TrustAI>
- Kužba, M. (n.d.). Conversational explanations of machine learning models using chatbots.
- Li, J., Cheng, X., Zhao, X., Nie, J.-Y., & Wen, J.-R. (2023, December). HaluEval: A large-scale hallucination evaluation benchmark for large language models. In H. Bouamor, J. Pino, & K. Bali (Eds.), *Proceedings of the 2023 conference on empirical methods in natural language processing* (pp. 6449–6464). Association for Computational Linguistics. <https://doi.org/10.18653/v1/2023.emnlp-main.397>
- Lipton, Z. C. (2018). The mythos of model interpretability. *Communications of the ACM*, 61, 36–43. <https://doi.org/10.1145/3233231>
- Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems, 2017-December*.
- Maddigan, P., Lensen, A., & Xue, B. (2024). Explaining genetic programming trees using large language models. *arXiv preprint arXiv:2403.03397*.
- Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J. K., Singh, S., Rathnayake, T., Vig, S., Granger, B. E., Muller, R. P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., ... Scopatz, A. (2017). Sympy: Symbolic computing in python. *PeerJ Computer Science*, 3, e103. <https://doi.org/10.7717/peerj-cs.103>
- OpenAI. (2022, January). ChatGPT: A conversational ai model. <https://openai.com/blog/chatgpt>
- Pace, R. K., & Barry, R. (1997). Sparse spatial autoregressions. *Statistics & Probability Letters*, 33(3), 291–297.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct), 2825–2830.
- Poli, R., Langdon, W., & McPhee, N. (2008). *A field guide to genetic programming*.

- Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., & Sutskever, I. (2023). Robust speech recognition via large-scale weak supervision. *International Conference on Machine Learning*, 28492–28518.
- Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21. <https://arxiv.org/pdf/1910.10683.pdf>
- Retzlaff, C. O., Angerschmid, A., Saranti, A., Schneeberger, D., Röttger, R., Müller, H., & Holzinger, A. (2024). Post-hoc vs ante-hoc explanations: Xai design guidelines for data scientists. *Cognitive Systems Research*, 86, 101243.
- Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "why should i trust you?" explaining the predictions of any classifier. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 13-17-August-2016*. <https://doi.org/10.1145/2939672.2939778>
- Shinan, E. (2017). Lark-a parsing toolkit for python. URL <https://github.com/lark-parser/lark>, 608.
- Sijben, E. M., Alderliesten, T., & Bosman, P. A. (2022). Multi-modal multi-objective model-based genetic programming to find multiple diverse high-quality models. *GECCO 2022 - Proceedings of the 2022 Genetic and Evolutionary Computation Conference*. <https://doi.org/10.1145/3512290.3528850>
- Singh, C., Inala, J. P., Galley, M., Caruana, R., & Gao, J. (2024). Rethinking interpretability in the era of large language models. *arXiv preprint arXiv:2402.01761*.
- Slack, D., Krishna, S., Lakkaraju, H., & Singh, S. (2023). Explaining machine learning models with interactive natural language conversations using talktomodel. *Nature Machine Intelligence*, 5. <https://doi.org/10.1038/s42256-023-00692-8>
- Wang, B., & Komatsuzaki, A. (2022). Gpt-j-6b: A 6 billion parameter autoregressive language model, 2021. URL <https://github.com/kingoflolz/mesh-transformer-jax>.