

Explanations by Counterfactual Argument in Recommendation Systems

by

YASH PATHAK
202111053

A Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of

MASTER OF TECHNOLOGY
in
INFORMATION AND COMMUNICATION TECHNOLOGY
to

DHIRUBHAI AMBANI INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGY



May, 2023

Declaration

I hereby declare that


- i) the thesis comprises of my original work towards the degree of Master of Technology in Information and Communication Technology at Dhirubhai Ambani Institute of Information and Communication Technology and has not been submitted elsewhere for a degree,
- ii) due acknowledgment has been made in the text to all the reference material used.



YASH PATHAK

Certificate

This is to certify that the thesis work entitled **Explanations by Counterfactual Argument in Recommendation Systems** has been carried out by YASH PATHAK for the degree of Master of Technology in Information and Communication Technology at *Dhirubhai Ambani Institute of Information and Communication Technology* under my supervision.



Prof. Arpit Rana
Thesis Supervisor

Acknowledgments

First of all, I would like to thank Almighty God for giving me the opportunity and guidance to achieve my goal and to be successful in the journey of pursuing M.Tech. There are many instances where without divine help, I wouldn't be able to achieve the desired outcome.

I would like to appreciate the opportunity to complete my studies successfully and granting me M.Tech degree with Software System Specialization by Dhirubhai Ambani Institute of Information and Communication Technology (DAIICT).

Words cannot express my gratitude to my research supervisor and my mentor Prof. Arpit Rana, for his invaluable guidance and dedicated involvement in every part of my work. The way he conveyed the detailed information required for this research work left a strong impression on me. Also, his humbleness and focused vision steered me in the right direction. I strongly feel blessed to have him as a mentor and his help during the learning process.

At last, I want to thank my parents for their constant support and guidance which enables me to be part of this endeavour.

Contents

Abstract	v
List of Tables	vi
List of Figures	vii
1 Introduction	1
1.1 Counterfactual Arguments	1
1.2 Recommendations	2
1.3 Explanations in Recommendations	3
1.4 Counterfactual Arguments as an Explanations	4
1.5 Motivation and Problem Statement	5
1.6 Organization of Thesis	6
2 Literature Survey	8
2.1 Model Agnostic vs Model Sensitive Approaches	8
2.2 Quality Measures of CFs	9
2.2.1 Proximity	9
2.2.2 Sparsity	10
2.2.3 Diversity	10
2.2.4 Feasibility or Plausibility	11
2.3 Generating Explanations	12
2.3.1 Instance-Centric Approaches	12
2.3.2 Genetic-Centric Approaches	13
2.3.3 Constraint-Centric Approaches	14
2.3.4 Regression-Centric Approaches	15
2.3.5 Probabilistic-Centric Approaches	16
2.4 Research Questions	17
2.5 Chapter Summery	17

3	Experiment & Methodologies	18
3.1	Terminologies	18
3.2	Brute-force Method	18
3.3	Genetic Algorithms	21
3.3.1	What are the Genetic Algorithms?	21
3.3.2	Use of Genetic Algorithms	22
3.4	Code & Hardware Setup	23
3.5	Chapter Summery	24
4	Insights & Results	25
4.1	Insights about explanations	25
4.2	Upward Movement	26
4.3	Optimization by Genetic Algorithms	26
4.4	Chapter Summery	27
5	Conclusion & Future Scope	29
	References	31
	Appendix A Extra observations in Brute-Force Method	33

Abstract

In recent advances in the domains of Artificial Intelligence (AI) and Machine Learning (ML), complex models are used. Due to their complexity and approaches, they have black box type of nature and raise the question of a trustworthy for decision process especially in the high cost decisions scenario. To overcome this problem, users of these systems can ask for an explanation about the decision which can be provided by system in various ways. One way of generating these explanations is by the help of Counterfactual (CF) arguments. Although there is a debate on how AI can generate these explanations, either by Correlation or Causal Inference, in Recommendation Systems (RecSys) the aim is to generate these explanations with minimum Oracle calls and have near optimal length (eg., in terms of interactions) of provided explanations. In this study we analyze the nature of CFs and different methods (eg., Model Agnostic approach, Genetic Algorithms (GA)) to generate them along with the quality measures. Extensive experiments show that the generation of CFs can be done through multiple approaches and selecting optimal CFs will improve the explanations.

List of Tables

3.1	Modified Movie-Lence Dataset	20
A.1	Table for Item Position 1	34
A.2	Table for Item Position 2	34
A.3	Table for Item Position 3	35
A.4	Table for Item Position 4	35
A.5	Table for Item Position 5	36
A.6	Table for Item Position 6	36
A.7	Table for Item Position 7	37
A.8	Table for Item Position 8	37
A.9	Table for Item Position 9	38
A.10	Table for Item Position 10	38
A.11	Table for Item Position 11	39
A.12	Table for Item Position 12	39
A.13	Table for Item Position 13	40
A.14	Table for Item Position 14	40
A.15	Table for Item Position 15	41
A.16	Table for Item Position 16	41
A.17	Table for Item Position 17	42
A.18	Table for Item Position 18	42
A.19	Table for Item Position 19	43
A.20	Table for Item Position 20	43

List of Figures

3.1	Generating Candidates	19
3.2	Checking Candidates for CF	20
3.3	Genetic Algorithm	21
4.1	Avg Min CF across different K	25
4.2	Avg Min CF across different target Item Position (t)	26
4.3	Count of Position Improvement for all user's candidates	27

CHAPTER 1

Introduction

Artificial Intelligence (AI) and Machine Learning (ML) have become integral parts of various digital aspects of human lives. They are employed in a wide range of applications, including recommendation systems (such as movie recommendations) and decision-making processes (like loan approvals). As users become more aware of these AI systems and their impact on their lives, they often seek to understand the reasoning behind the decisions made by these systems[15].

To address this need for transparency and user understanding, explanations for the decisions made by AI and ML systems are provided[2, 17]. These explanations aim to shed light on the factors that influenced the decision and provide insights into the decision-making process. This becomes particularly important when the decision-making process is considered a "black box," meaning it is difficult for humans to understand how the system arrived at a particular decision. Neural networks, a type of AI model, are often considered black boxes due to their complex and non-linear nature[18].

One approach to generating explanations is through the use of Counterfactual (CF) Arguments[17, 11, 4]. Counterfactual Arguments are hypothetical statements or scenarios that describe alternative outcomes based on changes in the input features of a system. In the context of AI and ML explanations, Counterfactual Arguments involve generating explanations by presenting alternative scenarios and analyzing how changes to the input features would have affected the decision.

By using Counterfactual Arguments, explanations can be generated that show users how their decisions would have been different if certain factors were different.

1.1 Counterfactual Arguments

A counterfactual argument is a statement that describes a causal situation by asserting that if a specific event (X) had not occurred, then another event (Y) would

not have happened. It explores the relationship between cause and effect and highlights the influence of a particular factor on the outcome.

For example, consider the statement, "I wouldn't have burned my tongue if I hadn't sipped this hot coffee." In this case, event Y is burning the tongue, and cause X is drinking hot coffee. The counterfactual argument suggests that if the person had not sipped the hot coffee (event X), they would not have burned their tongue (event Y).

In the context of AI and ML explanations, counterfactual arguments can be used to provide insights into the decision-making process of models. When a decision is made by an AI or ML model, there are typically multiple factors or input variables that contribute to the outcome. By exploring counterfactual scenarios, where certain factors are altered or removed, it becomes possible to understand the impact of specific features on the decision.

These types of explanations can be helpful in several ways. Firstly, they provide a clearer understanding of the reasoning behind a particular decision made by the model. Users can gain insights into which features or inputs were influential in the decision-making process. Secondly, counterfactual arguments allow users to explore alternative scenarios and understand how changes in certain factors would have affected the decision. This empowers users to make informed decisions and potentially modify their inputs or actions to achieve different outcomes.

Overall, counterfactual arguments help to elucidate the cause-effect relationships within AI and ML models, providing users with explanations that enhance their understanding of the decision-making process.

1.2 Recommendations

A recommendation system is a type of machine learning (ML) or artificial intelligence (AI) program that provides suggestions or recommendations to users based on available data[14, 13]. The goal of a recommendation system is to help users discover new items or content that align with their preferences and interests.

Recommendation systems utilize various parameters and data sources to generate personalized recommendations. Some common parameters include:

1. Previous Purchases: The system considers the user's past purchases or transactions to understand their preferences and recommend similar items. For example, if a user has previously bought books on science fiction, the system may suggest other science fiction books.

2. Search History: The system takes into account the user's search queries and browsing history to identify patterns and infer their interests. Based on the user's search history, the system can recommend items related to their recent searches.
3. Demographic Data: User demographics such as age, gender, location, or language can be used to personalize recommendations. This information helps the system tailor suggestions based on demographic-specific preferences.
4. Ratings and Feedback: If users have provided ratings or feedback on items they have interacted with, the system can leverage this information to make recommendations. It can identify items that have received positive feedback from users with similar preferences and suggest them to the current user.
5. Collaborative Filtering: Collaborative filtering is a technique where the system analyzes the behavior and preferences of similar users to make recommendations. If two users have similar tastes or purchase patterns, items liked by one user can be recommended to the other.
6. Content-Based Filtering: Content-based filtering focuses on the characteristics of the items themselves. It analyzes the attributes of the items and recommends similar items based on their features. For example, if a user has shown interest in action movies, the system may recommend other action movies based on their genre or actors.

By combining these parameters and utilizing ML or AI algorithms, recommendation systems can generate personalized and relevant suggestions for users. The ultimate aim is to enhance the user experience by presenting them with items or content that align with their preferences, thereby increasing user engagement and satisfaction.

1.3 Explanations in Recommendations

In the given scenario, User X is using an over-the-top (OTT) platform to watch movies. The platform utilizes a recommendation system to suggest movies to users based on their previous interactions and preferences. User X is curious about why the 3rd movie (I3) in the recommended list was suggested to them and asks for an explanation.

In response to User X's query, the system would provide an explanation that takes into account the user's previous interactions and preferences. The system

might respond with an explanation like this: "You have liked movies I2 and I4, and you have also watched movies I4 and I5. Based on your preferences and viewing history, movie I3 is suggested to you."

This explanation aims to provide transparency and clarity to the user by highlighting the specific factors that influenced the recommendation. In this case, the system references the movies that User X has previously shown interest in (I2 and I4) and the movies they have watched (I4 and I5). Based on the user's preferences and viewing history, the system determines that movie I3 is likely to align with User X's tastes and therefore recommends it.

By providing this explanation, the system helps User X understand the reasoning behind the recommendation and offers insight into how their previous interactions influenced the suggested movie. This kind of explanation enhances user understanding, builds trust in the recommendation system, and empowers users to make informed choices about the content they want to consume.

1.4 Counterfactual Arguments as an Explanations

To generate explanations using Counterfactual Arguments, the explanation generating model applies minor changes to the input data. The objective is to make the smallest possible alteration to the input data that would result in crossing the decision boundary of the decision-making model.

Counterfactual explanations are particularly useful in scenarios where a decision has been made, such as a loan request being rejected. These explanations aim to provide feedback to the person whose request was rejected, assisting them in understanding what changes they can make to their features or input data to move to the desirable side of the decision boundary.

For example, let's consider a situation where someone's loan request was denied based on their annual income. The counterfactual explanation might state, "If your annual income is 15,000 instead of 12,000, you would be eligible for the loan." This explanation presents a hypothetical scenario where a minor increase in the person's annual income (from 12,000 to 15,000) would lead to a different decision outcome, i.e., loan eligibility.

By providing such counterfactual explanations, individuals are given actionable insights on what changes they can make to improve their chances of a positive outcome. In this case, it suggests that increasing their annual income would make them eligible for the loan.

Counterfactual explanations offer valuable feedback to users by highlighting

the specific changes they can make to their input features to achieve a desired outcome. This helps users understand the impact of different factors on the decision-making process and empowers them to make informed decisions and take actions that can potentially lead to more favorable outcomes.

1.5 Motivation and Problem Statement

In certain domains, especially those involving high costs or significant consequences, users often desire the ability to change or understand the decisions made by recommendation models. This is because the impact of these decisions can have substantial implications.

To generate explanations using Counterfactual Arguments in such contexts, it becomes essential to consider permutations in the input values. By exploring different combinations and variations of the input features, it is possible to generate near-optimal explanations that require only minimal changes in the features.

The goal of utilizing Counterfactual Arguments in this scenario is to provide explanations that offer insights into the decision-making process while minimizing the disruption or modification required to the original input data.

By considering permutations in the input values, the system can identify alternative scenarios that would lead to different outcomes. These permutations involve making small changes or adjustments to the input features, allowing the system to generate explanations that show how slight modifications would impact the decision.

For example, in the context of a high-cost domain, such as a financial investment recommendation, a counterfactual explanation might indicate, "If you had invested 10% more in Stock A instead of Stock B, your returns would have been significantly higher." This explanation demonstrates the impact of a minor change in the investment allocation on the overall outcome.

By leveraging permutations in the input values and using Counterfactual Arguments, near-optimal explanations can be generated. These explanations provide users with valuable insights into how different choices or modifications to the input features would have affected the decision. This empowers users in high-cost domains to understand the reasoning behind recommendations and potentially make more informed decisions or take actions that align with their desired outcomes.

When generating explanations using Counterfactual Arguments and considering permutations in the input values, the search space for these permutations

can increase significantly. The size of the search space depends on the number of input features and the range of values they can take.

For example, if a recommendation model takes into account multiple input features, such as user preferences, demographic information, past behaviors, and contextual factors, the number of potential permutations and combinations can grow exponentially.

The focus of the study is to optimize the process of finding counterfactual explanations by efficiently exploring this large search space. Researchers and practitioners aim to develop algorithms and techniques that can traverse through the search space in an effective and efficient manner.

The goal is to find the most relevant and informative counterfactual explanations while minimizing the computational resources and time required for the search. This optimization involves devising strategies to narrow down the search space, prioritize certain combinations of input features, or leverage heuristics and machine learning techniques to guide the exploration process.

Efficiently navigating the search space is crucial for scalability and practicality, as it allows the generation of counterfactual explanations within acceptable timeframes, even when dealing with large and complex datasets.

By finding optimizations in the process of searching for counterfactual explanations, researchers and practitioners can make these explanations more feasible and practical in real-world applications. It enables users to receive timely and meaningful insights into decision-making processes, enhancing their understanding and enabling them to make informed choices.

1.6 Organization of Thesis

Thesis Organization:

The thesis is structured into five chapters, each addressing a specific aspect of generating explanations for recommendations using Counterfactual Arguments and exploring optimizations in the search space. The organization of the thesis is as follows:

1. Introduction: The first chapter provides an introduction to the topic, highlighting the significance of generating explanations for recommendations and the use of Counterfactual Arguments. It outlines the research objectives, the motivation behind the study, and the scope of the thesis.
2. Literature Survey: The second chapter presents a comprehensive literature

survey on the existing methods and approaches related to generating explanations for recommendations and utilizing Counterfactual Arguments. It reviews relevant research papers, academic works, and industry practices in the field. This chapter provides a foundation for understanding the state of the art and identifying research gaps.

3. Experiment & Methodologies: The third chapter details the experimental setup and methodologies employed in the study. It describes the dataset used, the recommendation models or algorithms under investigation, and the specific Counterfactual Arguments techniques employed. This chapter outlines the process of generating counterfactual explanations and the considerations taken into account to optimize the search space.
4. Insights & Results: The fourth chapter presents the insights and results obtained from the experiments and analyses conducted. It discusses the generated counterfactual explanations and their effectiveness in enhancing user understanding and decision-making. This chapter also highlights any observed patterns, limitations, or challenges encountered during the research process.
5. Conclusion & Future Scope

The fifth chapter of the thesis is dedicated to providing a conclusion and discussing the future scope of the research. It serves as a culmination of the study and provides a summary of the key findings and contributions.

CHAPTER 2

Literature Survey

2.1 Model Agnostic vs Model Sensitive Approaches

When it comes to generating explanations for recommendations, there are multiple approaches that can be employed. These approaches can be broadly categorized into two types: model-sensitive and model-agnostic.

1. **Model-Sensitive Approaches:** Model-sensitive approaches are dependent on the specific type of recommendation model used. These approaches take into account the internal workings and mechanisms of the model to generate explanations. They leverage the specific algorithms, architectures, or characteristics of the model to interpret and explain the recommendations it provides. Model-sensitive approaches are tailored to the specific model and may not be easily applicable to other models or architectures.
2. **Model-Agnostic Approaches:** In contrast, model-agnostic approaches focus solely on the input and output of the recommendation model, irrespective of its internal workings. These approaches do not rely on the specific details of the model but rather concentrate on understanding the relationship between the input data and the generated output[8]. Model-agnostic approaches aim to generate explanations that can be applied across various types of recommendation models. They offer a more versatile and generalizable framework for explaining recommendations.

In this particular study, the focus is mainly on model-agnostic approaches. The reason for this choice is the usefulness and flexibility of model-agnostic techniques when dealing with different types of recommendation models. Since different models may have diverse architectures, algorithms, or implementation details, using model-agnostic approaches allows the generated explanations to be applicable across a wider range of recommendation systems.

By employing model-agnostic approaches, the study can provide insights and techniques that can be utilized in various recommendation settings, regardless of the specific model employed. This approach enhances the practicality and applicability of the explanations, allowing them to be adapted to different recommendation models and supporting better understanding and interpretability of the recommendations generated by these models.

2.2 Quality Measures of CFs

These quality measure ensure better quality CF which can be more interpretable, understandable and actionable for real world scenarios[2, 19, 21].

2.2.1 Proximity

In the context of generating counterfactual arguments (CFs), the distance between the CF and the original input data point plays a crucial role. The distance function is used to measure the dissimilarity or deviation of the CF from the original data point. The choice of distance function can vary depending on the specific approach or methodology employed in the CF generation process.

The goal of selecting an appropriate distance function is to ensure that the CF is as close as possible to the original input data point while still resulting in a change in the decision or outcome. The distance should reflect the extent of modification or perturbation applied to the input features to achieve the desired outcome.

Different distance functions may be used based on the nature of the data and the specific requirements of the CF generation approach. Commonly used distance metrics include Euclidean distance, Manhattan distance, cosine similarity, or custom-defined distance measures tailored to the problem domain.

Ideally, the distance between the CF and the original input data point should be minimized, indicating that the CF is as similar as possible to the original data while leading to a different decision or outcome. The exact definition of "minimum possible distance" may vary depending on the specific context and requirements of the CF generation process.

By considering the distance between the CF and the original input data point, researchers and practitioners can assess the extent of modification or perturbation required to generate meaningful CFs. This evaluation helps ensure that the CFs are meaningful, informative, and provide valuable insights into the decision-making process of the underlying model or system.

2.2.2 Sparsity

When generating counterfactual explanations, minimizing the change in the number of features is an important consideration. The objective is to provide CFs that are more human interpretable, understandable, and meaningful. By minimizing the changes in the number of features, the CFs remain closer to the original input, making the explanations more intuitive and easier to comprehend for individuals.

If the CF involves drastic changes or modifications in multiple features, it can become challenging for users to grasp the cause-and-effect relationship between the modified features and the resulting decision change. On the other hand, when the number of feature changes is minimized, the CFs are more aligned with the original input, and the causal relationship between specific features and the decision outcome becomes more apparent.

Reducing the number of feature changes also helps in maintaining the context and relevance of the CFs. Users can better understand and relate to the explanations when they see that only a few key features have been modified, as it aligns with their mental model and expectations of how those features influence the decision-making process.

Moreover, minimizing the change in the number of features contributes to the overall interpretability and transparency of the CFs. Users can more easily comprehend and trust the explanations when they observe that only a few relevant features have been altered, rather than a complex and extensive modification across multiple dimensions.

By prioritizing minimal changes in the number of features, the CFs become more human interpretable, allowing users to gain insights into the decision-making process and understand the factors that drive the recommended outcomes. This approach enhances the effectiveness and usability of the generated CF explanations.

2.2.3 Diversity

When generating multiple counterfactuals (CFs), it is crucial to ensure that they are not only close to the original data point but also relatively diverse from each other. This approach aims to provide a more comprehensive and nuanced understanding of the decision-making process by offering multiple perspectives and alternative scenarios.

By keeping the CFs close to the original data point, it means that they should share similarities in terms of the input features, except for the minimal changes

necessary to alter the decision outcome. This closeness allows users to relate the CFs to their own situation and better comprehend the specific modifications that would be required to achieve a different decision.

On the other hand, it is equally important to ensure diversity among the CFs. Each CF should represent a distinct pathway or set of changes that lead to a different decision outcome. By introducing diversity, the CFs provide a range of possibilities and highlight various factors that influence the decision process. This diversity encourages users to consider multiple perspectives and helps them explore different strategies or changes that may lead to desired outcomes.

Having multiple CFs that are both close to the original data point and diverse from each other enhances the understandability of the explanations. Users can compare and contrast the different CFs, identifying common patterns or factors that consistently impact the decision, as well as unique variations that yield different outcomes. This comparative analysis allows users to gain insights into the decision-making process, understand the underlying rules or patterns employed by the recommendation system, and make informed choices based on their preferences and goals.

In summary, generating CFs that are close to the original data point and relatively diverse from each other provides a more comprehensive and understandable set of explanations. This approach facilitates a deeper understanding of the decision-making process, empowers users to explore different possibilities, and enables informed decision-making based on individual preferences and objectives.

2.2.4 Feasibility or Plausibility

When generating counterfactuals (CFs), it is important to consider two key qualities: plausibility and feasibility. These qualities ensure that the generated CFs are realistic and align with the constraints and boundaries of the problem domain.

Plausibility refers to the idea that the CFs should be plausible or believable in the context of the problem being addressed. This means that the CFs should not change immutable features that are inherently fixed for an individual, such as gender or age. Modifying these immutable features would result in CFs that are unrealistic and do not align with the inherent characteristics of the person or the problem under consideration. By preserving these immutable features, the CFs remain plausible and maintain a sense of consistency with the original data point.

Feasibility, on the other hand, refers to the notion that the CFs should be feasible or achievable in practice. The CFs should propose changes that are practically

possible and do not lead to paradoxical interpretations. For example, if the goal is to generate CFs for improving loan eligibility, it would not be feasible to suggest a CF that triples the income of an individual overnight. Such a drastic and unrealistic change would not be practical or achievable. Feasibility ensures that the CFs provide actionable and meaningful suggestions that can be reasonably implemented by the individual.

By considering both plausibility and feasibility, the generated CFs maintain a balance between believability and practicality. They provide meaningful and actionable suggestions that respect the inherent constraints and limitations of the problem domain. This approach enhances the usefulness and applicability of the CFs, as they are more likely to be accepted and implemented by the individuals seeking explanations or considering alternative decision scenarios.

In summary, plausibility ensures that the generated CFs do not change immutable features, preserving the inherent characteristics of the individuals. Feasibility ensures that the CFs propose changes that are practical and achievable within the given problem context, avoiding paradoxical interpretations. By adhering to these qualities, the CFs provide realistic and actionable explanations that align with the constraints and boundaries of the problem domain.

2.3 Generating Explanations

There are multiple approaches available in the literature to generate and evaluate CFs[2, 17, 11, 19].

2.3.1 Instance-Centric Approaches

Instance-centric algorithms in generating counterfactuals rely on random feature permutations and aim to find counterfactuals that are close to the original case within a certain distance metric. These algorithms often employ unique optimization techniques and loss functions to search for suitable counterfactuals[12]. However, due to their reliance on random permutations, instance-centric algorithms may face challenges in terms of feasibility and diversity of the generated counterfactuals.

To overcome these issues, some instance-centric algorithms incorporate additional mechanisms into their loss functions. These mechanisms are designed to improve the plausibility, feasibility, and diversity of the generated counterfactual explanations. By incorporating these mechanisms, the algorithms attempt to ad-

dress the potential violations of these qualities and enhance the overall quality of the generated counterfactuals.

The exact nature of the mechanisms embedded in the loss functions may vary depending on the specific instance-centric algorithm. These mechanisms could involve penalty terms, regularization techniques, or constraints that guide the optimization process towards more feasible and diverse counterfactuals. By incorporating such enhancements, the instance-centric algorithms aim to generate counterfactual explanations that are not only close to the original case but also satisfy the desired qualities of feasibility and diversity.

The inclusion of these mechanisms within the loss functions helps to refine the search process for counterfactuals and improve the overall quality of the generated explanations. However, it is important to note that even with these enhancements, instance-centric algorithms may still face challenges in fully satisfying the qualities of feasibility and diversity, and further research and development are necessary to address these limitations effectively.

2.3.2 Genetic-Centric Approaches

The category referred to as "Genetic-Centric" encompasses methods that utilize genetic algorithms as an optimization technique to search for counterfactuals. Genetic algorithms are inspired by natural evolution and mimic the process of genetic crossover and mutation to explore the search space[16, 6, 3].

In the context of generating counterfactuals, genetic-centric methods apply genetic algorithms to iteratively refine and explore feature vectors. These algorithms typically start with an initial population of feature vectors, which are treated as potential counterfactuals. Through successive generations, the feature vectors undergo genetic operations such as crossover and mutation.

The genetic operations involve combining or exchanging features among different feature vectors and introducing random changes to feature values. These operations allow the exploration of different combinations and modifications of features, leading to a diverse set of potential counterfactuals.

By leveraging genetic algorithms, these methods often satisfy characteristics like diversity and plausibility. The crossover and mutation processes enable the generation of counterfactuals that may exhibit diverse combinations of features, reflecting different potential scenarios or changes in input variables. This diversity enhances the range of explanations and provides a more comprehensive understanding of the decision-making process.

Furthermore, genetic-centric methods can also address plausibility. The itera-

tive nature of genetic algorithms allows for the refinement of counterfactuals over multiple generations. Through the optimization process, the methods can converge towards counterfactuals that are more plausible and closer to satisfying the desired properties or constraints.

Overall, genetic-centric methods offer an effective approach to generating counterfactual explanations by utilizing genetic algorithms for optimization. By permitting feature vectors to crossover and mutate, these methods achieve diversity and plausibility, contributing to more comprehensive and meaningful counterfactual explanations.

2.3.3 Constraint-Centric Approaches

The category referred to as "Constraint-Centric" includes methods that tackle counterfactual generation as a constraint satisfaction problem. These algorithms formulate the problem by modeling constraints and use various techniques, such as satisfiability modulo theory (SMT) solvers, to find solutions that satisfy these constraints[22].

In constraint-centric methods, the goal is to identify counterfactual explanations that satisfy specific properties or constraints. These properties can encompass a range of qualities, including diversity and plausibility. By formulating the problem as a constraint satisfaction issue, these methods provide a general framework that allows for the satisfaction of multiple counterfactual qualities.

One of the main advantages of constraint-centric methods is their generality. They can handle different types of constraints and are flexible in accommodating various counterfactual properties. This generality enables the methods to satisfy a wide range of qualities, including diversity and plausibility, as required by the problem at hand.

The use of techniques such as SMT solvers allows constraint-centric methods to effectively search for solutions that fulfill the specified constraints. SMT solvers are powerful tools that can handle complex logical constraints and efficiently explore the search space to find feasible solutions. By leveraging these solvers, constraint-centric methods can generate counterfactual explanations that satisfy the desired qualities.

In summary, constraint-centric methods excel at modeling the counterfactual generation problem as a constraint satisfaction issue. Their generality and utilization of techniques like SMT solvers enable them to readily satisfy diverse counterfactual qualities, including diversity and plausibility. By leveraging these methods, researchers and practitioners can obtain counterfactual explanations

that meet specific constraints and provide valuable insights into the decision-making process.

2.3.4 Regression-Centric Approaches

The category referred to as "Regression-Centric" encompasses methods that generate explanations by leveraging the weights of a regression model [20, 1, 10]. These methods share similarities with LIME (Local Interpretable Model-Agnostic Explanations) in their approach. The underlying assumption is that by permuting the features of the input data, an interpretable model, typically a linear regression model, can be fitted to the modified data. The weights assigned to each feature in the regression model are then used as explanations.

The main idea behind regression-centric methods is to use the weights of the regression model as indicators of the importance or influence of each feature on the decision or outcome. By examining the magnitude and sign of the weights, one can infer the degree of contribution each feature has in determining the output. These weights are often used to explain why a certain decision or recommendation was made.

However, it is important to note that regression-centric methods may face challenges in meeting certain quality measures, including plausibility and diversity, in the generated counterfactuals. Plausibility refers to the degree to which the counterfactual explanations align with real-world possibilities and constraints. While the regression model provides explanations based on feature weights, these explanations may not always be realistic or plausible in practical terms.

Similarly, diversity refers to the variety and range of possible counterfactual explanations. Regression-centric methods, due to their reliance on the weights of the regression model, may have limitations in producing diverse explanations. The explanations generated by these methods might be limited to the factors considered by the regression model and may not encompass a wide range of potential changes or scenarios.

While regression-centric methods offer interpretability through the use of regression model weights, their limitations in achieving plausibility and diversity in counterfactual explanations should be taken into account. These methods can provide insights into the relative importance of features but may not capture the full complexity and richness of the decision-making process.

2.3.5 Probabilistic-Centric Approaches

The category referred to as "Probabilistic-Centric" includes approaches that frame the counterfactual generation problem as a probabilistic problem. These methods leverage probabilistic techniques and models to generate counterfactual explanations[5, 9].

Probabilistic-centric approaches employ various strategies to address the counterfactual generation problem. These strategies may involve concepts such as random walks, Markov sampling, variational autoencoders (VAEs), and probabilistic graphical models (PGMs). Each of these techniques contributes to learning efficient data representations and generating counterfactuals in a probabilistic framework.

Random walks are often used in probabilistic-centric approaches to explore the data space and generate counterfactual instances. By taking random steps or transitions in the data space, these methods can create alternative instances that satisfy certain constraints or properties. Random walks allow for the generation of diverse counterfactual explanations by exploring different regions of the data distribution.

Markov sampling is another technique utilized in probabilistic-centric approaches. It involves drawing samples from a Markov chain, where each sample represents a potential counterfactual instance. Markov sampling allows for the exploration of different possible scenarios and facilitates the generation of diverse counterfactual explanations.

Variational autoencoders (VAEs) are unsupervised learning models that can learn efficient data encodings or representations. In the context of counterfactual generation, VAEs can capture the underlying structure of the data and generate counterfactual instances that satisfy specific constraints or properties. These models can generate new instances that deviate from the original data distribution while maintaining certain characteristics or features.

Probabilistic graphical models (PGMs) are frameworks that represent the dependencies and relationships among variables using probabilistic models. PGMs provide a flexible and expressive way to model the counterfactual generation problem. They can capture complex dependencies and generate counterfactual explanations based on probabilistic reasoning.

In summary, probabilistic-centric approaches leverage probabilistic techniques such as random walks, Markov sampling, VAEs, and PGMs to tackle the counterfactual generation problem. These methods focus on learning efficient data representations and generating counterfactual explanations in a probabilistic frame-

work. By incorporating probabilistic reasoning and modeling, these approaches offer valuable insights into the generation of counterfactual instances with diverse characteristics and adherence to specific constraints or properties.

2.4 Research Questions

These are the Research Questions obtained from the study,

- RQ-1 : Which Solution can be considered near optimal solution also known as minimal length CFs to generate explanations?
- RQ-2 : How to reduce computational complexity to find a near optimal solution?

2.5 Chapter Summary

In this chapter, we explored various approaches for generating counterfactual arguments and discussed different quality measures to evaluate the interpretability, feasibility, and actionability of the explanations. We examined methods such as modifying input data, applying perturbations, and employing optimization algorithms to generate counterfactual arguments.

To assess the quality of the explanations, we considered factors such as human interpretability, feasibility, and actionability. Human interpretability focused on ensuring that the explanations were easily understandable and comprehensible to users without requiring specialized technical knowledge. Feasibility involved evaluating whether the suggested modifications or adjustments in the explanations were practical and could be implemented in real-world scenarios. Actionability assessed the extent to which the explanations provided actionable insights and suggestions that users could act upon to influence or change the decision outcome.

By exploring different approaches and employing these quality measures, we aimed to provide a comprehensive analysis of the generated counterfactual arguments. This analysis helps us understand the effectiveness and usefulness of the explanations in decision-making processes, enhancing transparency and interpretability.

CHAPTER 3

Experiment & Methodologies

3.1 Terminologies

Here are notations for the experiment,

- U : Number of of users in Dataset
- I_c : Number of total item catalog in Dataset
- I : Interacted Itemset of particular user
- n : Size of I
- R : Recommendation list of interacted itemset I
- t : Target item (Explanation asked by User)
- t_{pos} : Target item position in Recommendation list R
- E : Candidate (subset of I)
- R' : Recommendation list of $I - E$ itemset
- CF : If $t \notin R'$ than E is CF else not

3.2 Brute-force Method

In order to analyze and identify the nature of minimal length counterfactuals (CFs), a Brute-Force search approach is utilized. This approach involves exhaustively searching through all possible combinations of input interacted itemsets and their corresponding output recommendation lists in the recommendation system (RecSys)[8].

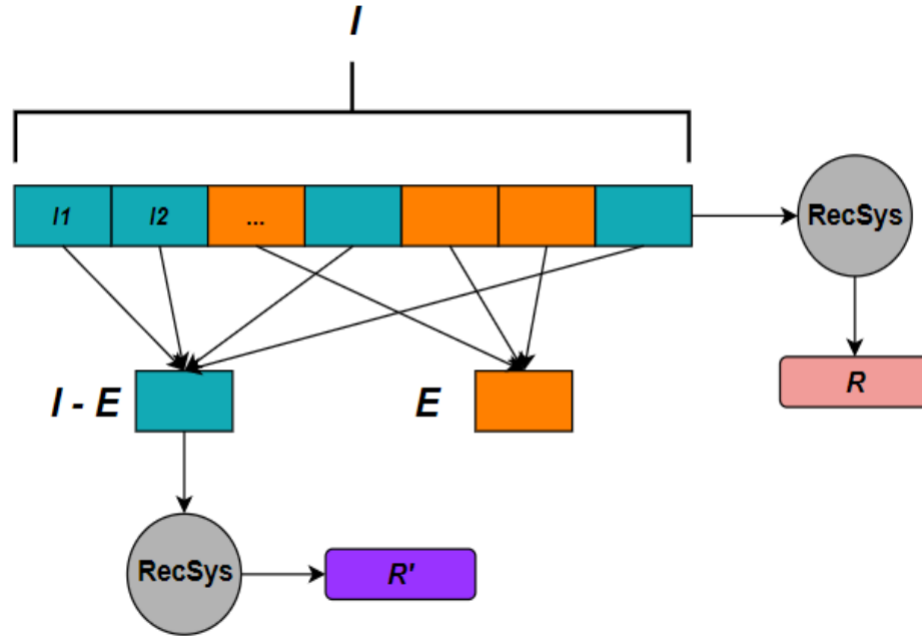


Figure 3.1: Generating Candidates

One important assumption made in this analysis is that the platform or system storing the input interacted itemset and recommendation list does not retain any specific user information. This means that the analysis focuses solely on the relationship between the input and output without considering any personal user data. This assumption helps maintain privacy and data security by avoiding the storage of user-specific information.

By performing a Brute-Force search, the analysis aims to uncover the minimal length CFs, which are the most concise and minimal changes required in the input itemset to cause a different recommendation outcome. The search process systematically explores all possible combinations to identify these minimal CFs.

The utilization of a Brute-Force search approach allows for a comprehensive analysis of the CFs without any specific assumptions or biases. It provides a thorough examination of the relationships between input itemsets and recommendations, helping to uncover patterns and insights that can improve the understanding and interpretability of the recommendation system.

Overall, the use of a Brute-Force search approach, coupled with the assumption of not storing user information, enables an objective analysis of minimal length CFs and contributes to the transparency and privacy-preserving nature of the analysis.

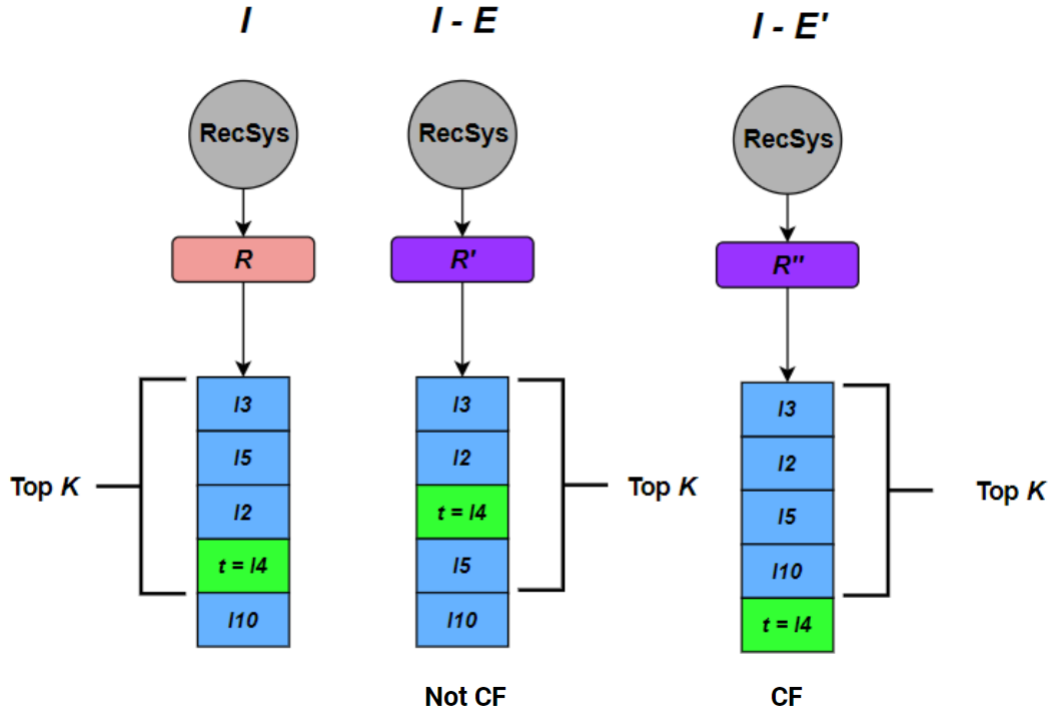


Figure 3.2: Checking Candidates for CF

For every subset candidate E , $I - E$ will be fed to RecSys and from R' the position of t will be considered according to CF condition mentioned in notation. It can be seen in Figure 3.1 & 3.2. Among all CFs, minimal length CFs will be considered for the explanation.

RecSys will provide ranking for all items in the catalog hence top K to be considered as a recommendation list. **Movie-Lence dataset of 100K Interaction** has been used[7]. Only users with more than 20 interactions have been considered rest users are discarded. Details of the modified dataset is mentioned in Table 3.1. The latest 20 interactions of users have been considered in two parts. (1) First

No.	Feature	Count
1	Number of Users (U)	943
2	Total Item Catalog (Ic)	1682
3	Size of Interacted Itemset (n)	10
4	Length of Recommendation List (K)	20

Table 3.1: Modified Movie-Lence Dataset

10 Interactions, (2) First 11 to 20 Interactions. Although the approach is Model Agnostic, for the experimental setup LSTM based RecSys has been used for both modified datasets.

3.3 Genetic Algorithms

3.3.1 What are the Genetic Algorithms?

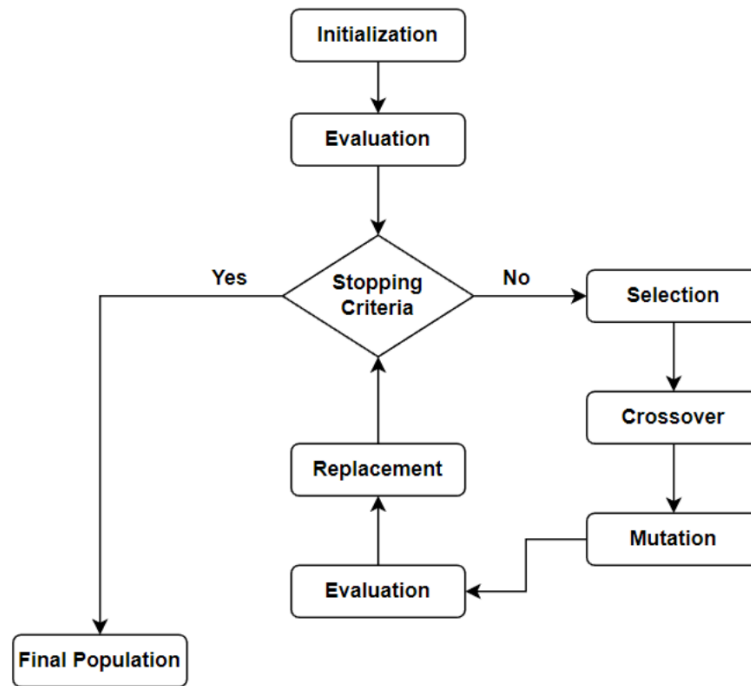


Figure 3.3: Genetic Algorithm

Genetic algorithms (GAs) are a class of computational optimization algorithms inspired by the principles of natural selection and genetics. They are commonly used to solve complex problems that involve finding the best solution among a large search space.

The basic idea behind genetic algorithms is to mimic the process of evolution by iteratively improving a population of potential solutions to a problem. The algorithm starts with an initial population of individuals, each representing a possible solution. These individuals are evaluated using a fitness function, which quantitatively measures how well each solution solves the problem. The flow for the GA can be seen in Figure 3.3.

The GA then applies a set of genetic operators, including selection, crossover, and mutation, to create a new generation of individuals. The selection operator favors individuals with higher fitness, making them more likely to be chosen for reproduction. Crossover combines genetic material from selected individuals to create offspring, while mutation introduces random changes to the genetic information. This process is inspired by the biological processes of reproduction and

genetic variation.

The new generation of individuals is evaluated using the fitness function, and the process of selection, crossover, and mutation continues for multiple generations. Over time, the population evolves, and solutions with higher fitness become more prevalent.

Genetic algorithms can be applied to a wide range of optimization problems, including combinatorial optimization, function optimization, scheduling, and machine learning, among others. They are particularly useful when the search space is large and complex, and traditional optimization methods may be impractical.

One of the strengths of genetic algorithms is their ability to explore multiple areas of the search space simultaneously, allowing them to potentially find globally optimal or near-optimal solutions. However, they are not guaranteed to find the best solution in every case and may require careful tuning of parameters and problem-specific adaptations to achieve good results.

3.3.2 Use of Genetic Algorithms

In the proposed solution utilizing a genetic algorithm to find counterfactuals (CFs), several methods have been employed to guide the evolutionary process. Here are the key methods and strategies used:

1. **Parent Selection:** The roulette wheel selection method is utilized for parent selection. This method assigns a probability of selection to each candidate solution based on its fitness value. Candidates with higher fitness values have a higher chance of being selected as parents for the next generation.
2. **Fitness Function:** The fitness function is designed to evaluate the quality of each candidate solution. In this case, the fitness is determined by considering the position difference of the target item in the recommendation list compared to the original recommendation list. The fitness value is assigned based on the difference, where a higher fitness value indicates a larger difference to the original recommendation. This means that candidates affecting the target items place in the recommendation list receive higher fitness values.
3. **Crossover:** The crossover operation is used to combine genetic information from selected parent solutions to create new offspring solutions. In this approach, the two-point crossover method is employed. This method selects two random points in the candidate solutions and exchanges the genetic

material between these points. By performing crossover, the offspring inherit characteristics from both parents, allowing for exploration of different combinations of features.

4. Preservation of Interactions: To ensure the preservation of important interactions, the first three interactions (or features) are not changed during the crossover process. This means that the genetic material from the first three interactions remains intact in the offspring solutions, providing some level of consistency and continuity with the original data point.

By utilizing these methods within the genetic algorithm framework, the search for CFs is guided towards finding solutions that exhibit desirable properties, such as having the target item positioned at a more favorable location in the recommendation list. The roulette wheel selection method ensures diversity in the parent population, the fitness function guides the evaluation of candidate solutions based on position differences, and the two-point crossover operation facilitates the exploration of different combinations of genetic information. Additionally, preserving the initial interactions maintains the relevance of the original data point in the CF generation process.

Overall, this approach aims to enhance the effectiveness and efficiency of the genetic algorithm in generating CFs that provide meaningful and actionable explanations in the context of recommendation systems.

3.4 Code & Hardware Setup

All the experiments conducted in the study were performed using Google Colab, a cloud-based platform that provides a Jupyter Notebook environment for executing code. Google Colab offers the advantage of flexibility and accessibility, allowing researchers to work on their experiments from any device with an internet connection.

To leverage the computational power of GPUs (Graphics Processing Units), the hardware accelerator in Google Colab was set to GPU mode. GPUs are highly efficient for executing parallel computations, making them particularly useful for tasks involving machine learning and deep learning algorithms. Utilizing GPU acceleration can significantly speed up the execution of computationally intensive tasks, such as training complex models or performing large-scale data processing.

The exact specifications of the GPU available in the free version of Google Colab may vary based on availability. The free version provides access to a range

of GPU options, including NVIDIA Tesla K80, T4, P4, or P100. The specific GPU assigned to a user's session is dependent on the availability at the time of usage.

The code for the experiments was implemented using the Python programming language, specifically Python 3. Python is widely used in the field of data science and machine learning due to its extensive ecosystem of libraries and frameworks, which provide various tools and functionalities for data manipulation, modeling, and analysis.

By utilizing Google Colab with GPU acceleration and implementing the experiments in Python 3, the researchers were able to take advantage of efficient computation and leverage the rich ecosystem of Python libraries to facilitate the execution of their experiments and analysis.

3.5 Chapter Summery

The experiments in the study were conducted using a modified version of the MovieLens dataset. The specific modifications made to the dataset are detailed in the Method subsection of the thesis. These modifications could include pre-processing steps such as data cleaning, filtering, or feature engineering to suit the specific requirements of the research.

To conduct the experiments, the researchers utilized Google Colab, a cloud-based platform that provides a Jupyter notebook environment for running Python code. Python 3 was chosen as the programming language for implementing the experiments and analyzing the dataset.

By conducting the experiments on Google Colab using Python 3, the researchers were able to implement the necessary algorithms, apply the designated methodologies, and perform analyses on the modified MovieLens dataset. This setup allowed for efficient and reproducible experimentation, facilitating the investigation and evaluation of the proposed methods for generating explanations of recommendations using counterfactual arguments.

CHAPTER 4

Insights & Results

4.1 Insights about explanations

The average minimum size of the solution for different target items positioned (t_pos) and K (Size of candidate) is nearly $n/2$. The observation of average minimum length of CFs = $n/2$ is observed in both of the modified datasets as well as the upward movement shown in Figure 4.1 & 4.2. This shows that items have only positive impact with respect to the position in the recommendation list and more items (instead of one or two) are responsible for the target item to appear in the recommendation list.

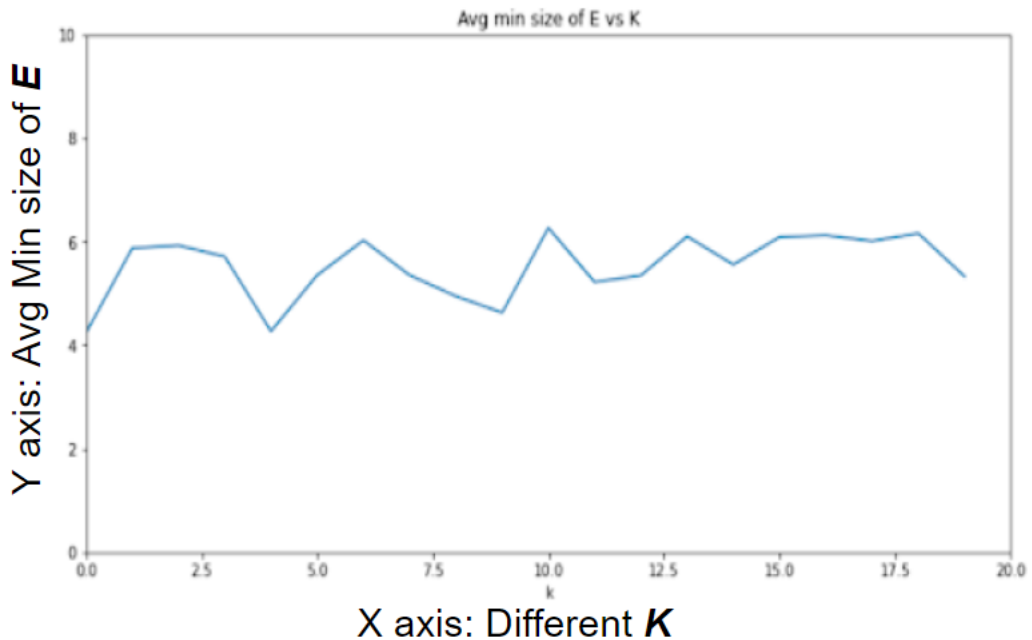


Figure 4.1: Avg Min CF across different K

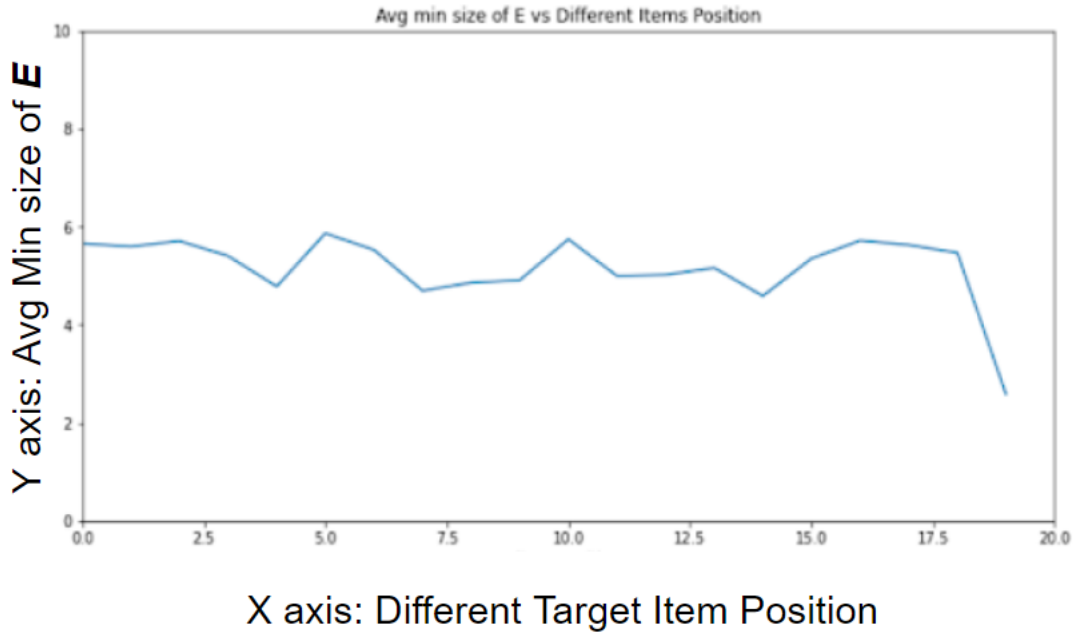


Figure 4.2: Avg Min CF across different target Item Position (t)

4.2 Upward Movement

From R to R' , the upward movement of target item t in recommendation list is very rare (15,16,918 out of 1,92,74,920 - 7.9%). The improvement is largely for one position. It can be seen in Figure 4.3.

4.3 Optimization by Genetic Algorithms

By incorporating genetic algorithms with a specific fitness function, selection method, and crossover method, our approach successfully achieved optimization in terms of reducing recommendation calls and minimizing search space traversal. The reduction in recommendation calls is directly influenced by factors such as the population size and the number of generations allowed during the experiment. In our case, we observed a significant reduction of 50% in oracle calls compared to the brute-force approach.

For a single user query with one item, the proposed method took approximately 20 seconds to generate a near-optimal CF explanation. This time includes the process of population initialization, evolution through generations, and convergence to a satisfactory solution.

It's important to note that the effectiveness and efficiency of the proposed

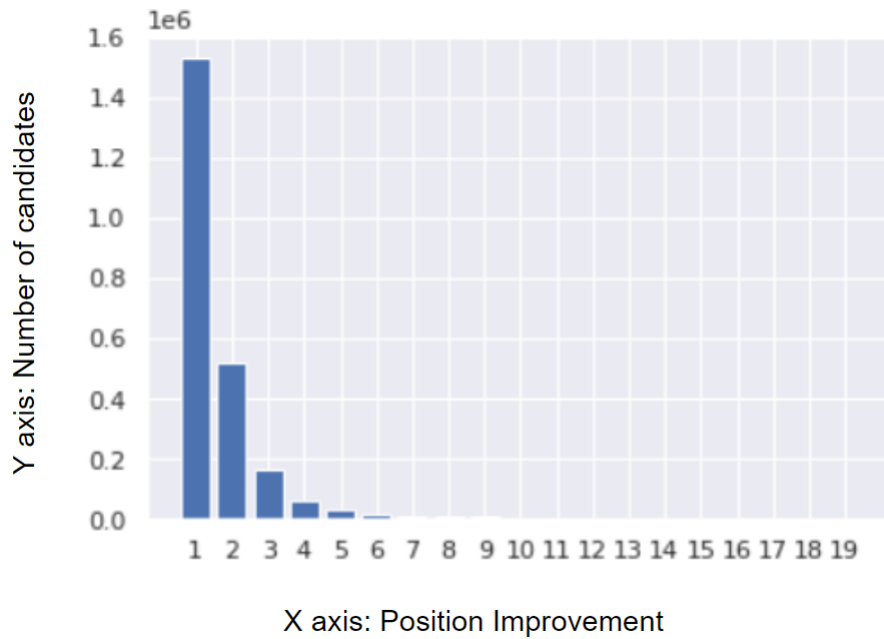


Figure 4.3: Count of Position Improvement for all user’s candidates

method heavily rely on the choice of fitness function, crossover method, and selection method. Depending on the specific requirements and characteristics of the recommendation system under study, further optimization can be achieved by fine-tuning these components. This optimization process warrants future experiments and exploration.

By successfully reducing the number of recommendation calls and optimizing the CF generation process, our approach offers a promising direction for generating counterfactual explanations in a more efficient and scalable manner, ultimately enhancing the interpretability and usefulness of recommendation systems.

4.4 Chapter Summery

The results obtained from the experiments indicate that suitable explanations based on Counterfactual Arguments tend to have a length ranging from 4 to 6 items. This suggests that in order to generate meaningful and effective explanations, a small set of modifications or changes in the input data is often sufficient. These modifications aim to influence the recommendation outcome and provide insights into the factors that led to the specific recommendation.

Furthermore, the analysis of the recommendation lists reveals that the upward movement of the targeted item in the recommendation list of a candidate is a

relatively rare occurrence, accounting for only 7.9% of the cases examined. This means that in most instances, the targeted item is recommended in a similar or slightly different position compared to its original placement in the candidate's recommendation list.

However, when an upward movement does occur, it is typically a minor shift of just one position higher in the recommendation list. This suggests that the counterfactual changes made to the input data have a limited impact on the positioning of the targeted item in the recommendation list. Despite this, the explanations derived from the counterfactual arguments still offer valuable insights into the decision-making process of the recommendation system.

These findings shed light on the nature of suitable explanations generated using counterfactual arguments. They suggest that concise explanations can be achieved with a small number of modifications to the input data, and while the upward movement of the targeted item is infrequent, it does occur occasionally, albeit in a limited manner. These insights contribute to a better understanding of the characteristics and effectiveness of counterfactual-based explanations in recommendation systems.

The utilization of genetic algorithms for search space optimization has proven fruitful in our study. By leveraging the principles of natural selection and evolution, genetic algorithms enable the exploration and exploitation of the search space to find near-optimal solutions. In our context of generating counterfactual explanations, genetic algorithms efficiently traverse the large search space of feature combinations and permutations, leading to improved optimization in terms of explanation quality and computational resources required. This approach offers benefits such as handling complex problem domains, adapting to different fitness functions and constraints, and enhancing the effectiveness and efficiency of the search process.

Overall, the application of genetic algorithms for search space optimization yields improved results in our study. It provides a robust and adaptive framework for generating interpretable and actionable explanations in recommendation systems. By effectively exploring the search space and utilizing genetic operators like crossover and mutation, we achieve a reduction in recommendation calls and search space traversal. The use of genetic algorithms opens up possibilities for further advancements and optimizations by experimenting with different fitness functions, crossover techniques, and selection methods. Through these innovations, we can continue to enhance the generation of near-optimal counterfactual explanations in recommendation systems.

CHAPTER 5

Conclusion & Future Scope

To pursue the research objectives outlined in Section II, we conducted an in-depth analysis of minimal length counterfactuals (CFs) using the Modified Movie-Lens 100K Dataset and an LSTM-based recommendation system. Our findings revealed an interesting pattern, indicating that regardless of the dataset used, the average minimum length of CFs converged to $n/2$, where n represents the total number of interacted items. This observation suggests a consistent behavior in CF generation across different datasets, providing valuable insights into the nature of explanations in recommendation systems.

However, we also encountered a rare occurrence of upward movement in the recommendation list, where the target item was placed one position higher than the original recommendation. This anomaly prompted us to investigate whether this phenomenon is specific to the type of recommendation system employed or if it reflects a more fundamental characteristic of CFs. Understanding the underlying factors behind this upward movement will help establish the extent to which CFs can truly be considered model-agnostic.

To delve deeper into the nature of CFs, we recognize the need for broader experimentation involving diverse real-world datasets and a wide range of recommendation system architectures. By analyzing different combinations of datasets and recommendation models, we aim to gain a comprehensive understanding of the factors influencing CF generation and their implications for explainable recommendations.

To tackle the optimization challenge associated with the large search space of subsets of the interacted itemset I , we have explored the use of genetic algorithms (GA). Our initial experiments utilizing GA have demonstrated promising results in terms of reducing the number of recommendation system calls and optimizing the search space traversal. By applying GA-based techniques, we were able to significantly reduce the number of oracle calls by 50% compared to the brute-force approach.

Exploring different facets of GA optimization, such as diverse selection methods, innovative crossover strategies, and refined fitness functions, holds great potential. These investigations can lead to the fine-tuning of the CF generation process and the discovery of computationally efficient approaches that effectively balance the generation of high-quality explanations with minimized computational overhead.

References

- [1] P. S. R. Aditya and M. Pal. Local interpretable model agnostic shap explanations for machine learning models. 2022.
- [2] Y.-L. Chou, C. Moreira, P. Bruza, C. Ouyang, and J. Jorge. Counterfactuals and causability in explainable artificial intelligence: Theory, algorithms, and applications, 2021.
- [3] S. Dandl, C. Molnar, M. Binder, and B. Bischl. Multi-objective counterfactual explanations. pages 448–469, 2020.
- [4] R. M. B. de Oliveira and D. Martens. A framework and benchmarking study for counterfactual generating methods on tabular data. *Applied Sciences*, 11(16):7274, aug 2021.
- [5] A. Ghazimatin, O. Balalau, R. S. Roy, and G. Weikum. PRINCE: Provider-side Interpretability with Counterfactual Explanations in Recommender Systems. jan 2020.
- [6] R. Guidotti, A. Monreale, S. Ruggieri, D. Pedreschi, F. Turini, and F. Giannotti. Local rule-based explanations of black box decision systems. 2018.
- [7] F. M. Harper and J. A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4), dec 2015.
- [8] V. Kaffes, D. Sacharidis, and G. Giannopoulos. Model-agnostic counterfactual explanations of recommendations. page 280–285, 2021.
- [9] A.-H. Karimi, J. von Kügelgen, B. Schölkopf, and I. Valera. Algorithmic recourse under imperfect causal knowledge: a probabilistic approach. 2020.
- [10] S. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. 2017.

- [11] R. K. Mothilal, D. Mahajan, C. Tan, and A. Sharma. Towards unifying feature attribution and counterfactual explanations: Different means to the same end. jul 2021.
- [12] R. K. Mothilal, A. Sharma, and C. Tan. Explaining machine learning classifiers through diverse counterfactual explanations. pages 607–617, 2020.
- [13] A. Paterek. Improving regularized singular value decomposition for collaborative filtering. *Proceedings of KDD Cup and Workshop*, 01 2007.
- [14] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. page 285–295, 2001.
- [15] R. Shang, K. J. K. Feng, and C. Shah. Why am i not seeing it? understanding users’ needs for counterfactual explanations in everyday recommendations. page 1330–1340, 2022.
- [16] S. Sharma, J. Henderson, and J. Ghosh. CERTIFAI. feb 2020.
- [17] J. Tan, S. Xu, Y. Ge, Y. Li, X. Chen, and Y. Zhang. Counterfactual explainable recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. ACM, oct 2021.
- [18] K. H. Tran, A. Ghazimatin, and R. S. Roy. Counterfactual explanations for neural recommenders. jul 2021.
- [19] S. Verma, V. Boonsanong, M. Hoang, K. E. Hines, J. P. Dickerson, and C. Shah. Counterfactual explanations and algorithmic recourses for machine learning: A review. 2022.
- [20] A. White and A. d’Avila Garcez. Measurable counterfactual local explanations for any classifier. 2019.
- [21] S. Xu, Y. Li, S. Liu, Z. Fu, X. Chen, and Y. Zhang. Learning post-hoc causal explanations for recommendation. 2021.
- [22] W. Yang, J. Li, C. Xiong, and S. C. H. Hoi. Mace: An efficient model-agnostic framework for counterfactual explanation. 2022.

CHAPTER A

Extra observations in Brute-Force Method

As part of the brute-force experiment, an analysis was conducted on one user to examine all possible subsets of the interacted itemset. This analysis aimed to explore the various combinations of items that the user had interacted with and investigate the potential counterfactual arguments (CFs) that could be derived from these subsets.

Let's consider an example scenario where User A has interacted with a set of movies on a streaming platform. The interacted itemset consists of movies I1, I2, I3, and I4. To analyze all possible subsets, we start by considering subsets of size 1, which include individual movies: I1, I2, I3, and I4. Next, we move on to subsets of size 2, such as I1, I2, I1, I3, I1, I4, I2, I3, I2, I4, and I3, I4. We continue this process, considering subsets of size 3 and 4 until we have examined all possible combinations.

For each subset, we analyze the CFs that can be generated by performing minor changes to the input data, such as removing or adding a specific movie. These CFs provide insights into how the recommendation system might have behaved differently if certain movies were not part of the user's interaction history.

By conducting this analysis for all possible subsets, we can explore the relationships between the user's interactions and the recommended movies. This process helps in identifying patterns, determining which subsets of interacted movies contribute significantly to the recommendations, and generating meaningful explanations based on counterfactual arguments.

Overall, the brute-force experiment involves systematically examining all possible subsets of the interacted itemset for a particular user. This analysis provides valuable insights into the relationships between user interactions and the resulting recommendations, aiding in the generation of relevant and informative counterfactual explanations. Please refer below tables from Table A.1 to Table A.20 .

Size_of_K	Total_CFs	1	2	3	4	5	6	7	8	9
k1	78	0	0	0	2	12	21	28	11	4
k2	32	0	0	0	0	1	5	13	9	4
k3	31	0	0	0	0	1	5	12	9	4
k4	31	0	0	0	0	1	5	12	9	4
k5	31	0	0	0	0	1	5	12	9	4
k6	31	0	0	0	0	1	5	12	9	4
k7	31	0	0	0	0	1	5	12	9	4
k8	31	0	0	0	0	1	5	12	9	4
k9	31	0	0	0	0	1	5	12	9	4
k10	31	0	0	0	0	1	5	12	9	4
k11	31	0	0	0	0	1	5	12	9	4
k12	31	0	0	0	0	1	5	12	9	4
k13	31	0	0	0	0	1	5	12	9	4
k14	31	0	0	0	0	1	5	12	9	4
k15	31	0	0	0	0	1	5	12	9	4
k16	31	0	0	0	0	1	5	12	9	4
k17	31	0	0	0	0	1	5	12	9	4
k18	31	0	0	0	0	1	5	12	9	4
k19	31	0	0	0	0	1	5	12	9	4
k20	31	0	0	0	0	1	5	12	9	4

Table A.1: Table for Item Position 1

Size_of_K	Total_CFs	1	2	3	4	5	6	7	8	9
k1	975	10	45	120	208	241	194	104	43	10
k2	31	0	0	0	0	1	5	12	9	4
k3	31	0	0	0	0	1	5	12	9	4
k4	31	0	0	0	0	1	5	12	9	4
k5	30	0	0	0	0	1	5	11	9	4
k6	30	0	0	0	0	1	5	11	9	4
k7	30	0	0	0	0	1	5	11	9	4
k8	30	0	0	0	0	1	5	11	9	4
k9	30	0	0	0	0	1	5	11	9	4
k10	30	0	0	0	0	1	5	11	9	4
k11	30	0	0	0	0	1	5	11	9	4
k12	30	0	0	0	0	1	5	11	9	4
k13	30	0	0	0	0	1	5	11	9	4
k14	30	0	0	0	0	1	5	11	9	4
k15	30	0	0	0	0	1	5	11	9	4
k16	30	0	0	0	0	1	5	11	9	4
k17	30	0	0	0	0	1	5	11	9	4
k18	30	0	0	0	0	1	5	11	9	4
k19	30	0	0	0	0	1	5	11	9	4
k20	30	0	0	0	0	1	5	11	9	4

Table A.2: Table for Item Position 2

Size_of_K	Total_CFs	1	2	3	4	5	6	7	8	9
k1	1022	10	45	120	210	252	210	120	45	10
k2	1022	10	45	120	210	252	210	120	45	10
k3	48	0	0	0	0	3	9	21	10	5
k4	37	0	0	0	0	2	7	14	10	4
k5	33	0	0	0	0	1	5	13	10	4
k6	33	0	0	0	0	1	5	13	10	4
k7	32	0	0	0	0	1	5	13	9	4
k8	32	0	0	0	0	1	5	13	9	4
k9	32	0	0	0	0	1	5	13	9	4
k10	32	0	0	0	0	1	5	13	9	4
k11	32	0	0	0	0	1	5	13	9	4
k12	32	0	0	0	0	1	5	13	9	4
k13	32	0	0	0	0	1	5	13	9	4
k14	31	0	0	0	0	1	5	12	9	4
k15	31	0	0	0	0	1	5	12	9	4
k16	31	0	0	0	0	1	5	12	9	4
k17	31	0	0	0	0	1	5	12	9	4
k18	31	0	0	0	0	1	5	12	9	4
k19	31	0	0	0	0	1	5	12	9	4
k20	31	0	0	0	0	1	5	12	9	4

Table A.3: Table for Item Position 3

Size_of_K	Total_CFs	1	2	3	4	5	6	7	8	9
k1	1022	10	45	120	210	252	210	120	45	10
k2	1022	10	45	120	210	252	210	120	45	10
k3	1006	10	45	120	210	250	206	112	44	9
k4	31	0	0	0	0	1	5	12	9	4
k5	31	0	0	0	0	1	5	12	9	4
k6	31	0	0	0	0	1	5	12	9	4
k7	31	0	0	0	0	1	5	12	9	4
k8	31	0	0	0	0	1	5	12	9	4
k9	31	0	0	0	0	1	5	12	9	4
k10	31	0	0	0	0	1	5	12	9	4
k11	30	0	0	0	0	1	5	11	9	4
k12	30	0	0	0	0	1	5	11	9	4
k13	30	0	0	0	0	1	5	11	9	4
k14	30	0	0	0	0	1	5	11	9	4
k15	30	0	0	0	0	1	5	11	9	4
k16	30	0	0	0	0	1	5	11	9	4
k17	30	0	0	0	0	1	5	11	9	4
k18	30	0	0	0	0	1	5	11	9	4
k19	30	0	0	0	0	1	5	11	9	4
k20	30	0	0	0	0	1	5	11	9	4

Table A.4: Table for Item Position 4

Size_of_K	Total_CFs	1	2	3	4	5	6	7	8	9
k1	1022	10	45	120	210	252	210	120	45	10
k2	1022	10	45	120	210	252	210	120	45	10
k3	1022	10	45	120	210	252	210	120	45	10
k4	1022	10	45	120	210	252	210	120	45	10
k5	48	0	0	0	0	3	9	22	10	4
k6	31	0	0	0	0	1	5	12	9	4
k7	31	0	0	0	0	1	5	12	9	4
k8	31	0	0	0	0	1	5	12	9	4
k9	31	0	0	0	0	1	5	12	9	4
k10	31	0	0	0	0	1	5	12	9	4
k11	31	0	0	0	0	1	5	12	9	4
k12	31	0	0	0	0	1	5	12	9	4
k13	31	0	0	0	0	1	5	12	9	4
k14	31	0	0	0	0	1	5	12	9	4
k15	31	0	0	0	0	1	5	12	9	4
k16	31	0	0	0	0	1	5	12	9	4
k17	31	0	0	0	0	1	5	12	9	4
k18	31	0	0	0	0	1	5	12	9	4
k19	31	0	0	0	0	1	5	12	9	4
k20	31	0	0	0	0	1	5	12	9	4

Table A.5: Table for Item Position 5

Size_of_K	Total_CFs	1	2	3	4	5	6	7	8	9
k1	1022	10	45	120	210	252	210	120	45	10
k2	1022	10	45	120	210	252	210	120	45	10
k3	1022	10	45	120	210	252	210	120	45	10
k4	1022	10	45	120	210	252	210	120	45	10
k5	1022	10	45	120	210	252	210	120	45	10
k6	117	0	0	0	4	20	33	39	16	5
k7	73	0	0	0	2	11	20	24	12	4
k8	57	0	0	0	1	7	14	21	10	4
k9	43	0	0	0	0	3	9	17	10	4
k10	37	0	0	0	0	2	7	14	10	4
k11	33	0	0	0	0	1	5	13	10	4
k12	33	0	0	0	0	1	5	13	10	4
k13	33	0	0	0	0	1	5	13	10	4
k14	33	0	0	0	0	1	5	13	10	4
k15	32	0	0	0	0	1	5	13	9	4
k16	32	0	0	0	0	1	5	13	9	4
k17	32	0	0	0	0	1	5	13	9	4
k18	32	0	0	0	0	1	5	13	9	4
k19	32	0	0	0	0	1	5	13	9	4
k20	32	0	0	0	0	1	5	13	9	4

Table A.6: Table for Item Position 6

Size_of_K	Total_CFs	1	2	3	4	5	6	7	8	9
k1	1022	10	45	120	210	252	210	120	45	10
k2	1022	10	45	120	210	252	210	120	45	10
k3	1022	10	45	120	210	252	210	120	45	10
k4	1022	10	45	120	210	252	210	120	45	10
k5	1022	10	45	120	210	252	210	120	45	10
k6	996	10	45	120	208	245	200	115	43	10
k7	111	0	0	0	3	17	30	37	19	5
k8	43	0	0	0	0	2	7	20	10	4
k9	35	0	0	0	0	1	5	15	10	4
k10	32	0	0	0	0	1	5	13	9	4
k11	31	0	0	0	0	1	5	12	9	4
k12	31	0	0	0	0	1	5	12	9	4
k13	31	0	0	0	0	1	5	12	9	4
k14	31	0	0	0	0	1	5	12	9	4
k15	31	0	0	0	0	1	5	12	9	4
k16	31	0	0	0	0	1	5	12	9	4
k17	31	0	0	0	0	1	5	12	9	4
k18	31	0	0	0	0	1	5	12	9	4
k19	31	0	0	0	0	1	5	12	9	4
k20	31	0	0	0	0	1	5	12	9	4

Table A.7: Table for Item Position 7

Size_of_K	Total_CFs	1	2	3	4	5	6	7	8	9
k1	1021	10	45	120	210	252	210	119	45	10
k2	1021	10	45	120	210	252	210	119	45	10
k3	1021	10	45	120	210	252	210	119	45	10
k4	1016	10	45	120	210	251	208	118	44	10
k5	1004	10	45	120	210	250	206	109	44	10
k6	965	10	45	120	208	241	194	98	40	9
k7	912	10	45	120	205	228	174	88	33	9
k8	30	0	0	0	0	1	5	11	9	4
k9	30	0	0	0	0	1	5	11	9	4
k10	30	0	0	0	0	1	5	11	9	4
k11	30	0	0	0	0	1	5	11	9	4
k12	30	0	0	0	0	1	5	11	9	4
k13	30	0	0	0	0	1	5	11	9	4
k14	30	0	0	0	0	1	5	11	9	4
k15	30	0	0	0	0	1	5	11	9	4
k16	29	0	0	0	0	1	5	10	9	4
k17	29	0	0	0	0	1	5	10	9	4
k18	29	0	0	0	0	1	5	10	9	4
k19	29	0	0	0	0	1	5	10	9	4
k20	29	0	0	0	0	1	5	10	9	4

Table A.8: Table for Item Position 8

Size_of_K	Total_CFs	1	2	3	4	5	6	7	8	9
k1	1022	10	45	120	210	252	210	120	45	10
k2	1022	10	45	120	210	252	210	120	45	10
k3	1022	10	45	120	210	252	210	120	45	10
k4	1022	10	45	120	210	252	210	120	45	10
k5	1022	10	45	120	210	252	210	120	45	10
k6	1022	10	45	120	210	252	210	120	45	10
k7	1022	10	45	120	210	252	210	120	45	10
k8	1021	10	45	120	210	252	210	119	45	10
k9	113	0	0	0	3	17	32	38	17	6
k10	59	0	0	0	0	6	15	24	9	5
k11	36	0	0	0	0	2	7	14	9	4
k12	36	0	0	0	0	2	7	14	9	4
k13	32	0	0	0	0	1	5	13	9	4
k14	32	0	0	0	0	1	5	13	9	4
k15	32	0	0	0	0	1	5	13	9	4
k16	32	0	0	0	0	1	5	13	9	4
k17	32	0	0	0	0	1	5	13	9	4
k18	32	0	0	0	0	1	5	13	9	4
k19	32	0	0	0	0	1	5	13	9	4
k20	32	0	0	0	0	1	5	13	9	4

Table A.9: Table for Item Position 9

Size_of_K	Total_CFs	1	2	3	4	5	6	7	8	9
k1	1022	10	45	120	210	252	210	120	45	10
k2	1022	10	45	120	210	252	210	120	45	10
k3	1022	10	45	120	210	252	210	120	45	10
k4	1022	10	45	120	210	252	210	120	45	10
k5	1022	10	45	120	210	252	210	120	45	10
k6	1022	10	45	120	210	252	210	120	45	10
k7	1022	10	45	120	210	252	210	120	45	10
k8	1021	10	45	120	210	252	210	119	45	10
k9	1005	10	45	120	210	251	206	113	41	9
k10	151	0	0	1	9	30	46	41	19	5
k11	110	0	0	0	4	20	33	34	14	5
k12	43	0	0	0	0	3	9	17	10	4
k13	41	0	0	0	0	3	9	15	10	4
k14	37	0	0	0	0	2	7	14	10	4
k15	37	0	0	0	0	2	7	14	10	4
k16	33	0	0	0	0	1	5	13	10	4
k17	33	0	0	0	0	1	5	13	10	4
k18	33	0	0	0	0	1	5	13	10	4
k19	33	0	0	0	0	1	5	13	10	4
k20	33	0	0	0	0	1	5	13	10	4

Table A.10: Table for Item Position 10

Size_of_K	Total_CFs	1	2	3	4	5	6	7	8	9
k1	1022	10	45	120	210	252	210	120	45	10
k2	1021	10	45	120	210	252	210	119	45	10
k3	1021	10	45	120	210	252	210	119	45	10
k4	1021	10	45	120	210	252	210	119	45	10
k5	1020	10	45	120	210	252	210	119	44	10
k6	1016	10	45	120	210	251	208	118	44	10
k7	1010	10	45	120	210	250	206	115	44	10
k8	994	10	45	120	209	246	201	109	44	10
k9	961	10	45	120	207	238	189	103	40	9
k10	906	10	45	119	201	224	171	92	35	9
k11	34	0	0	0	0	1	5	15	9	4
k12	31	0	0	0	0	1	5	12	9	4
k13	31	0	0	0	0	1	5	12	9	4
k14	31	0	0	0	0	1	5	12	9	4
k15	31	0	0	0	0	1	5	12	9	4
k16	31	0	0	0	0	1	5	12	9	4
k17	31	0	0	0	0	1	5	12	9	4
k18	31	0	0	0	0	1	5	12	9	4
k19	31	0	0	0	0	1	5	12	9	4
k20	31	0	0	0	0	1	5	12	9	4

Table A.11: Table for Item Position 11

Size_of_K	Total_CFs	1	2	3	4	5	6	7	8	9
k1	1022	10	45	120	210	252	210	120	45	10
k2	1022	10	45	120	210	252	210	120	45	10
k3	1022	10	45	120	210	252	210	120	45	10
k4	1022	10	45	120	210	252	210	120	45	10
k5	1022	10	45	120	210	252	210	120	45	10
k6	1022	10	45	120	210	252	210	120	45	10
k7	1020	10	45	120	210	252	210	118	45	10
k8	1014	10	45	120	210	251	208	116	44	10
k9	1002	10	45	120	210	249	204	110	44	10
k10	988	10	45	120	210	246	198	106	44	9
k11	940	10	45	120	206	233	182	95	40	9
k12	31	0	0	0	0	1	5	12	9	4
k13	31	0	0	0	0	1	5	12	9	4
k14	31	0	0	0	0	1	5	12	9	4
k15	31	0	0	0	0	1	5	12	9	4
k16	31	0	0	0	0	1	5	12	9	4
k17	31	0	0	0	0	1	5	12	9	4
k18	31	0	0	0	0	1	5	12	9	4
k19	31	0	0	0	0	1	5	12	9	4
k20	31	0	0	0	0	1	5	12	9	4

Table A.12: Table for Item Position 12

Size_of_K	Total_CFs	1	2	3	4	5	6	7	8	9
k1	1022	10	45	120	210	252	210	120	45	10
k2	1022	10	45	120	210	252	210	120	45	10
k3	1022	10	45	120	210	252	210	120	45	10
k4	1022	10	45	120	210	252	210	120	45	10
k5	1022	10	45	120	210	252	210	120	45	10
k6	1022	10	45	120	210	252	210	120	45	10
k7	1022	10	45	120	210	252	210	120	45	10
k8	1022	10	45	120	210	252	210	120	45	10
k9	1022	10	45	120	210	252	210	120	45	10
k10	1022	10	45	120	210	252	210	120	45	10
k11	1022	10	45	120	210	252	210	120	45	10
k12	1011	10	45	120	210	250	206	116	44	10
k13	32	0	0	0	0	1	5	13	9	4
k14	32	0	0	0	0	1	5	13	9	4
k15	32	0	0	0	0	1	5	13	9	4
k16	31	0	0	0	0	1	5	12	9	4
k17	31	0	0	0	0	1	5	12	9	4
k18	31	0	0	0	0	1	5	12	9	4
k19	31	0	0	0	0	1	5	12	9	4
k20	31	0	0	0	0	1	5	12	9	4

Table A.13: Table for Item Position 13

Size_of_K	Total_CFs	1	2	3	4	5	6	7	8	9
k1	1022	10	45	120	210	252	210	120	45	10
k2	1022	10	45	120	210	252	210	120	45	10
k3	1022	10	45	120	210	252	210	120	45	10
k4	1022	10	45	120	210	252	210	120	45	10
k5	1022	10	45	120	210	252	210	120	45	10
k6	1022	10	45	120	210	252	210	120	45	10
k7	1022	10	45	120	210	252	210	120	45	10
k8	1022	10	45	120	210	252	210	120	45	10
k9	1022	10	45	120	210	252	210	120	45	10
k10	1022	10	45	120	210	252	210	120	45	10
k11	1022	10	45	120	210	252	210	120	45	10
k12	1022	10	45	120	210	252	210	120	45	10
k13	1022	10	45	120	210	252	210	120	45	10
k14	43	0	0	0	0	3	9	17	10	4
k15	37	0	0	0	0	2	7	14	10	4
k16	37	0	0	0	0	2	7	14	10	4
k17	37	0	0	0	0	2	7	14	10	4
k18	37	0	0	0	0	2	7	14	10	4
k19	37	0	0	0	0	2	7	14	10	4
k20	33	0	0	0	0	1	5	13	10	4

Table A.14: Table for Item Position 14

Size_of_K	Total_CFs	1	2	3	4	5	6	7	8	9
k1	1022	10	45	120	210	252	210	120	45	10
k2	1022	10	45	120	210	252	210	120	45	10
k3	1022	10	45	120	210	252	210	120	45	10
k4	1022	10	45	120	210	252	210	120	45	10
k5	1022	10	45	120	210	252	210	120	45	10
k6	1022	10	45	120	210	252	210	120	45	10
k7	1022	10	45	120	210	252	210	120	45	10
k8	1022	10	45	120	210	252	210	120	45	10
k9	1022	10	45	120	210	252	210	120	45	10
k10	1022	10	45	120	210	252	210	120	45	10
k11	1022	10	45	120	210	252	210	120	45	10
k12	1022	10	45	120	210	252	210	120	45	10
k13	1022	10	45	120	210	252	210	120	45	10
k14	1022	10	45	120	210	252	210	120	45	10
k15	97	0	0	1	6	16	23	33	13	5
k16	59	0	0	0	1	7	14	21	11	5
k17	43	0	0	0	0	3	9	17	10	4
k18	43	0	0	0	0	3	9	17	10	4
k19	43	0	0	0	0	3	9	17	10	4
k20	40	0	0	0	0	3	9	15	9	4

Table A.15: Table for Item Position 15

Size_of_K	Total_CFs	1	2	3	4	5	6	7	8	9
k1	1022	10	45	120	210	252	210	120	45	10
k2	1022	10	45	120	210	252	210	120	45	10
k3	1022	10	45	120	210	252	210	120	45	10
k4	1022	10	45	120	210	252	210	120	45	10
k5	1022	10	45	120	210	252	210	120	45	10
k6	1022	10	45	120	210	252	210	120	45	10
k7	1022	10	45	120	210	252	210	120	45	10
k8	1022	10	45	120	210	252	210	120	45	10
k9	1022	10	45	120	210	252	210	120	45	10
k10	1022	10	45	120	210	252	210	120	45	10
k11	1022	10	45	120	210	252	210	120	45	10
k12	1022	10	45	120	210	252	210	120	45	10
k13	1022	10	45	120	210	252	210	120	45	10
k14	1022	10	45	120	210	252	210	120	45	10
k15	1013	10	45	120	210	252	210	112	44	10
k16	93	0	0	1	7	18	25	24	14	4
k17	51	0	0	0	1	6	12	18	10	4
k18	41	0	0	0	0	3	9	15	10	4
k19	40	0	0	0	0	3	9	15	9	4
k20	36	0	0	0	0	2	7	14	9	4

Table A.16: Table for Item Position 16

Size_of_K	Total_CFs	1	2	3	4	5	6	7	8	9
k1	1022	10	45	120	210	252	210	120	45	10
k2	1022	10	45	120	210	252	210	120	45	10
k3	1022	10	45	120	210	252	210	120	45	10
k4	1022	10	45	120	210	252	210	120	45	10
k5	1022	10	45	120	210	252	210	120	45	10
k6	1022	10	45	120	210	252	210	120	45	10
k7	1022	10	45	120	210	252	210	120	45	10
k8	1022	10	45	120	210	252	210	120	45	10
k9	1022	10	45	120	210	252	210	120	45	10
k10	1022	10	45	120	210	252	210	120	45	10
k11	1022	10	45	120	210	252	210	120	45	10
k12	1022	10	45	120	210	252	210	120	45	10
k13	1022	10	45	120	210	252	210	120	45	10
k14	1022	10	45	120	210	252	210	120	45	10
k15	989	10	45	119	205	243	201	114	43	9
k16	967	10	45	119	203	237	194	111	39	9
k17	42	0	0	0	0	3	9	17	9	4
k18	37	0	0	0	0	2	7	15	9	4
k19	33	0	0	0	0	1	5	14	9	4
k20	33	0	0	0	0	1	5	14	9	4

Table A.17: Table for Item Position 17

Size_of_K	Total_CFs	1	2	3	4	5	6	7	8	9
k1	1022	10	45	120	210	252	210	120	45	10
k2	1022	10	45	120	210	252	210	120	45	10
k3	1022	10	45	120	210	252	210	120	45	10
k4	1022	10	45	120	210	252	210	120	45	10
k5	1022	10	45	120	210	252	210	120	45	10
k6	1022	10	45	120	210	252	210	120	45	10
k7	1022	10	45	120	210	252	210	120	45	10
k8	1022	10	45	120	210	252	210	120	45	10
k9	1022	10	45	120	210	252	210	120	45	10
k10	1022	10	45	120	210	252	210	120	45	10
k11	1021	10	45	120	210	252	210	119	45	10
k12	1021	10	45	120	210	252	210	119	45	10
k13	1017	10	45	120	210	251	208	118	45	10
k14	1016	10	45	120	210	251	208	117	45	10
k15	1004	10	45	120	209	247	203	115	45	10
k16	1004	10	45	120	209	247	203	115	45	10
k17	1003	10	45	120	209	247	203	115	44	10
k18	42	0	0	0	0	2	7	20	9	4
k19	40	0	0	0	0	2	7	18	9	4
k20	40	0	0	0	0	2	7	18	9	4

Table A.18: Table for Item Position 18

Size_of_K	Total_CFs	1	2	3	4	5	6	7	8	9
k1	1022	10	45	120	210	252	210	120	45	10
k2	1022	10	45	120	210	252	210	120	45	10
k3	1022	10	45	120	210	252	210	120	45	10
k4	1022	10	45	120	210	252	210	120	45	10
k5	1022	10	45	120	210	252	210	120	45	10
k6	1022	10	45	120	210	252	210	120	45	10
k7	1022	10	45	120	210	252	210	120	45	10
k8	1022	10	45	120	210	252	210	120	45	10
k9	1022	10	45	120	210	252	210	120	45	10
k10	1022	10	45	120	210	252	210	120	45	10
k11	1022	10	45	120	210	252	210	120	45	10
k12	1022	10	45	120	210	252	210	120	45	10
k13	1022	10	45	120	210	252	210	120	45	10
k14	1022	10	45	120	210	252	210	120	45	10
k15	1022	10	45	120	210	252	210	120	45	10
k16	1022	10	45	120	210	252	210	120	45	10
k17	1022	10	45	120	210	252	210	120	45	10
k18	1022	10	45	120	210	252	210	120	45	10
k19	512	1	9	36	84	126	126	85	36	9
k20	489	1	9	36	84	123	120	76	31	9

Table A.19: Table for Item Position 19

Size_of_K	Total_CFs	1	2	3	4	5	6	7	8	9
k1	1022	10	45	120	210	252	210	120	45	10
k2	1022	10	45	120	210	252	210	120	45	10
k3	1022	10	45	120	210	252	210	120	45	10
k4	1022	10	45	120	210	252	210	120	45	10
k5	1022	10	45	120	210	252	210	120	45	10
k6	1022	10	45	120	210	252	210	120	45	10
k7	1022	10	45	120	210	252	210	120	45	10
k8	1022	10	45	120	210	252	210	120	45	10
k9	1022	10	45	120	210	252	210	120	45	10
k10	1022	10	45	120	210	252	210	120	45	10
k11	1022	10	45	120	210	252	210	120	45	10
k12	1022	10	45	120	210	252	210	120	45	10
k13	1022	10	45	120	210	252	210	120	45	10
k14	1022	10	45	120	210	252	210	120	45	10
k15	1022	10	45	120	210	252	210	120	45	10
k16	1022	10	45	120	210	252	210	120	45	10
k17	1022	10	45	120	210	252	210	120	45	10
k18	1022	10	45	120	210	252	210	120	45	10
k19	667	9	36	84	130	148	129	86	36	9
k20	122	0	0	0	2	17	34	43	20	6

Table A.20: Table for Item Position 20