TECHNISCHE HOCHSCHULE DEGGENDORF

Fakultät Angewandte Informatik
Studiengang Master Angewandte Informatik

Identifikation und Analyse gefälschter Kundenrezensionen - Empirische Analyse anhand von Amazon-Kundenrezensionen

Identification and Analysis of Fake Customer Reviews - An Empirical Study on Customer Reviews from Amazon

Masterarbeit zur Erlangung des akademischen Grades:
Master of Science (M.Sc.)
an der Technischen Hochschule Deggendorf

Vorgelegt von:
Afzal Alarakhabhai Sufiya
Matrikelnummer: 22108480

Am: 20. September 2024

Prüfungsleitung:
Prof. Dr. Michael Scholz

Ergänzende Prüfende:
Prof. Dr. Florian Wahl

Abstract

In the era of e-commerce, online customer evaluations have become a crucial resource for consumers, aiding them in making informed purchasing decisions. However, the prevalence of fake reviews has raised concerns regarding the accuracy and legitimacy of these endorsements. This thesis addresses these challenges by investigating the frequency of fake reviews and their impact on customer behavior, specifically focusing on Amazon product reviews. Leveraging Artificial Intelligence (AI), this study implements advanced AI models to distinguish fake and real reviews on the Amazon Consumer Market. Additionally, it provides insights into the differences between genuine and fake reviews, as well as classification patterns between various features in reviews.

The study aims to measure the frequency of fraudulent reviews on Amazon and investigate the factors that differentiate genuine reviews from fraudulent ones. The method classifies reviews as authentic or fraudulent by implementing deep learning techniques. The primary goals also include understanding the variation in review rates across various attributes like product categories, product ratings, review length and identifying specific classification patterns or relations between these attributes. The research includes best fitting of various machine learning models to understand the classification patterns in honest and fake customer reviews.

This thesis makes a substantial contribution to our knowledge of fraudulent reviews on Amazon and how they affect consumer behavior. The results will help consumers, researchers, and e-commerce platforms by shedding light on the prevalence, patterns, and identification of consumer reviews. Furthermore, the study shows how modern machine learning models can be used to improve the dependability of online customer evaluations.

Aknowledgement

I am deeply grateful to Prof. Dr. Michael Scholz for his invaluable mentorship and guidance throughout this research journey. His expert insights and support have been crucial to the successful completion of this thesis. With a strong interest in Artificial Intelligence and Data Science, I enthusiastically accepted the challenge of investigating different models and comprehending their real-world applications. This research has been a fulfilling project as well as a valuable learning experience.

Furthermore, I would like to extend my gratitude to my university for providing the necessary academic and technical support, which greatly facilitated this research.

Contents

Acronyms

AI Artificial Intelligence. 6, 14, 16

BERT Bidirectional Encoder Representations from Transformers. iv, 9, 10, 11, 12, 13, 30

BPE byte-pair Encoding. 12

CUDA Compute Unified Device Architecture. 38

FNNs Feedforward Neural Networks. 11, 12

LSTMs Long Short-term Memory Networks. 7

ML Machine Learning. iv, 14, 15, 17, 19, 21, 40

MLM Masked Language Modeling. 9, 10

NLP Natural language processing. iv, 3, 7, 8, 9, 12, 13, 16, 30, 31, 32, 33, 64, 65

NSP Next Sentence Prediction. 9, 10, 12

RNNs Recurrent Neural Networks. 7

RoBERTa Robustly Optimized BERT-Pretraining Approach. iv, v, 3, 5, 9, 10, 11, 12, 13, 30, 31, 32, 35, 37, 45, 46, 63, 64, 65

1. Introduction

In the era of E-Commerce, Consumers spend scattered time searching for exact product information and online product reviews before deciding on and experiencing goods and services. Online product reviews have become an important factor in customer purchasing decisions (1). The online review business is hugely competitive. Reputation, as they say, is everything. Even on the research front, we increasingly see more consumer review reports enter the market. Platforms like Amazon, which hosts millions of products, rely heavily on user-generated reviews to guide potential buyers. According to a study by BrightLocal, 87% of consumers read online reviews for local businesses, and this behavior extends to e-commerce sites like Amazon (2). Reviews help consumers make informed decisions, influencing their purchasing behavior significantly.

Reviews on E-Commerce websites influence not only individual purchasing decisions but also product rankings and visibility. High ratings and good reviews can boost sales, whilst negative reviews can turn off potential customers. Despite their relevance, the presence of fraudulent reviews has harmed internet reviews' legitimacy. Fake reviews are faked assessments meant to deceive consumers. They are frequently coordinated by sellers attempting to artificially boost their product ratings or by competitors wishing to destroy a product's reputation. In order to pursue high ratings and positive reviews, many sellers practice the manipulation of comments to attract more consumers by showing fake public opinion information.

A study by Spigel Research Center discovered that, while products with positive reviews are more likely to be purchased, the presence of fake reviews can drastically affect consumer perception and behavior (3). The impact of fake reviews is multifaceted. Fake reviews have a direct impact on some important factors in the E-Commerce market such as Consumer trust, Purchasing decisions, and Market Integrity. The emergence of fake reviews can have serious consequences for customers and businesses. Fake reviews can influence consumers to make poor purchasing judgments. Fake negative reviews may harm a business's reputation and sales.

Amazon, as one of the largest e-commerce platforms globally, hosts an enormous volume of customer reviews. These reviews are crucial for guiding consumer decisions and influencing product rankings on the platform. Despite the vast number of genuine reviews, the proliferation of fake reviews on Amazon has become a significant concern. A 2020 analysis by Fakespot, a company specializing in detecting fraudulent reviews, estimated that about 40% of reviews on Amazon were fake or unreliable (4). Amazon has been proactively addressing fake reviews by utilizing both manual audits and machine learning algorithms. Amazon stated in 2020 that it had removed more than 200 million allegedly fake reviews before consumers could see them.

On websites like Amazon, the identification of fake reviews requires several technological layers and techniques. To preserve the integrity of the review system, these techniques seek to detect and remove fake reviews. These platforms use various machine learning methods to detect the anomalies that indicate fake reviews. Additionally, these E-Commerce giants

label reviews from verified purchases to add more credibility. Furthermore, manual human audits and community reporting are also pin practice to investigate the fake reviews. These methods have significantly reduced the prevalence of fake reviews. However, in the era of Artificial intelligence advancement, scammers are always improving their methods by using sophisticated technologies to avoid fake review detection.

Current research in fake review detection encompasses a variety approaches(5). Sentiment analysis and keyword spotting are two popular but insufficiently sophisticated text analysis methods for complex fake reviews. Conventional models like Random Forest and Logistic Regression have been used for detection, with differing degrees of effectiveness. However, these models fail to understand complex linguistic patterns identified in fake reviews. Recent advancements in deep learning models such as BERT, have substantially improved the accuracy of text classification tasks. However, there remains a substantial research gap in understanding the classification patterns of fake reviews. It is very important to know the impact of each feature and their interaction with each other to indicate review authenticity. This thesis focuses on building a better review detection algorithm and uses extensive machine learning approaches to understand the patterns in fake review classification.

1.1. Research and Contribution of this thesis work

- **Research Questions:**
  1. How can an advanced machine learning algorithm be used to identify fake consumer reviews?
  2. What is the corresponding significance of different features in predicting the authenticity of consumer reviews?
- **Contributions**
  1. By diving into the linguistic elements of reviews and identifying patterns that distinguish fake reviews from genuine ones, this thesis will be a valuable analysis for data scientists in advancing research of text classification techniques.
  2. The research examines the behavior of reviewers by understanding review length and average review ratings, to understand how these behaviors can indicate the authenticity of a review. This helps e-commerce platforms in their fraud detection systems by implementing behavioral data into their algorithm for better results. This research also helps consumers in their better purchasing decisions by reducing the influence of fraudulent reviews.
  3. The holistic approach of this study explores how various attributes such as product category, review length, and review ratings interact with each other and contribute to the review classification which provides comprehensive insights into the features affecting the legitimacy of customer reviews. This detailed investigation helps data scientists and engineers who need to understand the interaction of various features to build more sophisticated review detection models.

In this thesis, we focus on the application of deep learning and advanced machine learning techniques, with an emphasis on how these models can be used to identify fake reviews and understand the classification patterns in these reviews. We understand and implement NLP model RoBERTa to classify the reviews into fake or genuine based on textual data of reviews. Furthermore, we evaluate how well different machine learning algorithms can explain classification patterns between various attributes such as product category, product ratings, review length in customer reviews. To achieve this goals, this thesis involve several key steps: Data collection through web scraping of Amazon reviews data manipulation, data preprocessing for effective model training, and implementation of RoBERTa model to classify the reviews and machine learning models to understand the feature directions, extract feature importance of various attributes to understand the pattern.

For this research, a dataset consisting of various attributes such as product name, product category, product subcategory, review title, review ratings, and review description has been collected using effective web scraping techniques. The implementations of models for classification and pattern recognition have been developed using Python programming language, utilizing libraries such as Numpy, Pandas, Scikitlearn, Tensorflow and Transformers. To ensure accuracy and reproducibility, the entire development process—from data collection to model evaluation—was completed in a structured environment.

The ultimate objective of this thesis is to create a comprehensive and reliable framework for understanding of Amazon fake customer review. This involves creating a pipeline that covers feature analysis, model training, evaluation, and data preparation. By doing this, we hope to offer a clearer picture of the factors that contribute into a review's legitimacy as well as a reliable process for understanding fake reviews.

2. Theoretical Background

This section provides an overview of the background theories and technologies relevant to this thesis. A detailed investigation of the relevant theoretical knowledge and practical technologies for fake review identification and pattern recognition has been conducted.

2.1. Previous Research

In the past, there have been several research conducted involving various research points such as the identification of fake reviews using textual features, the impact of fake reviews on consumer perception, and the impact of features on fake review detection.

- **Fake review detection using transformer-based enhanced LSTM and RoBERTa(5)**: This study introduces a novel semantic- and linguistic-aware model for fake review identification that combines a RoBERTa transformer-based model with an LSTM layer, allowing it to detect detailed patterns inside false reviews. To improve openness and trustworthiness, this study uses Shapley Additive Explanations (SHAP) and attention approaches to clarify the model's classifications.
- **The Impact of Fake Reviews of Online Goods on Consumers(1)**: This paper investigates how fake reviews influence consumers from the perspective of the formation of consumers' purchase decisions by discussing four dimensions: Demand cognition, Looking for alternative plans, Purchase decisions, and Purchase behavior. This article examines the literature and conducts case studies on the impact of poor information on consumers produced by false evaluations, as well as how to avoid and control fake reviews.
- **Misinformation and Mistrust: The Equilibrium Effects of Fake Reviews on Amazon(6)**: This article examines how sellers on two-sided online platforms manipulate reputation systems, and their influence on consumers. This study examines how fraudulent product reviews on e-commerce platforms like Amazon impact consumer welfare through two channels. Rating manipulation deceives consumers, leading to inferior quality and greater prices. Rating manipulation reduces trust in ratings, perhaps leading to poor product matches if buyers place insufficient value on quality ratings.
- **Fake Reviews Detection through Analysis of Linguistic Features(7)**: This study investigates the use of natural language processing to detect fraudulent reviews by providing a thorough examination of linguistic characteristics to help distinguish between fake and authentic online reviews. This study involves estimating of the significance and value of 15 linguistic variables in relation to the classification methods used in this study.

This research results in important findings such as fraudulent reviews typically include longer sentences and more superfluous phrases and pauses.

This research builds on previous foundational studies to make several advancements over them. While studies like (5) also used the RoBERTa model, this thesis extends the analysis to focus on what really matters in the fake review detection problem—namely, the features that describe the review, and the interaction between different attributes of the review that might be more or less suspicious. Other studies, like (6) and (1), focus on the consumer perspective (which is certainly valid and important) or the equilibrium effects of fake reviews. This research is more about the technical side of how to detect fake reviews and understand the classification pattern of review detection. Lastly, (7) uses linguistic features, but the thesis extends the analysis to include deep learning models and a more comprehensive feature analysis. This study goes beyond mere classification by delving into the significance of different features (e.g., product category, review length, review ratings) in predicting review authenticity. The findings of this study help to understand the reasons behind the classification and thus facilitate improvement of future detection models. Since the interaction of different attributes with each other is studied, the central problem of this investigation provides a much wider perspective in understanding what makes reviews legit. This broadened view expands on past work that may have been limited to studying particular features in the forms alone.

2.2. Fake Consumer Reviews on Amazon

Consumer reviews for products on the internet have become an important factor in the E-commerce economy, highly impacting factors like brand value, purchasing decisions, etc. These consumer reviews are an important part of product sales on e-commerce platforms, social media, and other review websites. As per the published research, product buyers rely highly on information received from product reviews on e-commerce websites (8).

Amazon, one of the leading company with the largest market share in e-commerce, become very successful by offering a big selection of products and the best after-sales customer services. Consumer reviews on Amazon play one of the pivotal roles in Amazon's journey to build the best consumer-focused brand. These reviews on Amazon are very helpful for users to find the suitable product as per their requirements and price range. Additionally, these reviews are an important part of Amazon's algorithm for ranking products, which influences product visibility and sales (9). Amazon's review feature allows users to rate the purchased product on a scale of 5 stars and provide additional information such as review descriptions, product videos, and product images which then can be voted on as helpful or unhelpful by other users.

Regardless of the benefits of customer reviews, the sharp increase in fake reviews raises the question of the integrity of these systems. According to the fraudulent-review-detection service Fakespot, around 42 percent of 720 million Amazon reviews assessed in 2020 were bogus (4). Fake reviews on Amazon are generated using various methods such as paid reviews, bot-generated/AI-generated reviews, etc. These fake reviews are mostly created to pump the product ratings or damage a competitor's reputation. AI-generated reviews make this practice more sophisticated by mimicking human language and genuine review tone, which in the end become very hard to detect for review-detection algorithms.

The prevalence of fake reviews has a significant negative impact on factors such as purchasing decisions and brand trust in the area of Online shopping. The fake reviews become highly influential in case of purchasing decisions which eventually end in purchasing poor quality products. Furthermore, it creates a distrustful environment in online reviews. According to BrightLocal's Consumer Review Survey(2023), only 46% of respondents stated that they trust online reviews as much as personal recommendations from friends or family. The survey also revealed that 62% of consumers believed that they had read a fake review in the last year, with Amazon being the most frequently cited source (2). This lack of trust can lead to the destruction of the brand's reputation and repetitive business. Additionally, these reviews try to end competitiveness by giving unfair hikes in product rankings. Moreover, the manipulation of consumer reviews questions the reliability of Amazon's recommender system, potentially decreasing overall consumer satisfaction.

2.3. Natural Language Processing (NLP)

Natural Language Processing(NLP) is an integrated branch of computer science and artificial intelligence. It is primarily focused on enabling computers to process data encoded in natural language, making it closely important for data retrieval, knowledge representation, and computational linguistics. NLP tasks include speech recognition, text classification, natural language understanding, and natural language generation. In past years, NLP received high attention and advancements due to the integration of Machine learning, particularly deep learning for accuracy and efficiency improvement. In this thesis, NLP based model is used to process and understand Amazon customer reviews and classify them as fake or genuine.

2.3.1. Introduction to NLP

NLP is a tract of AI and Linguistics, dedicated to making machines understand human languages. It was created to make it easier to work with computers without understanding or learning machine languages.

NLP is classified into two parts: Natural Language Understanding(task to understand the human language) and Natural Language Generation(task to generate text in human language) as shown in fig.2.1 (10). In the context of this thesis, Natural Language Understanding enables machines to understand natural human language and its linguistics. it operates mainly on two primary components: syntax and semantics. The syntax is responsible for the proper grouping of the words to create sentences. In the case of semantics, it focuses on the real and exact meaning of the sentence made from Syntax. At syntactic level, techniques such as tokenization, part-of-speech tagging, and parsing are used to determine and process the text. On semantic level, tasks like Named Entity Recognition(NER), Sentiment analysis, and text classification are performed to understand the meaning of the text (11).

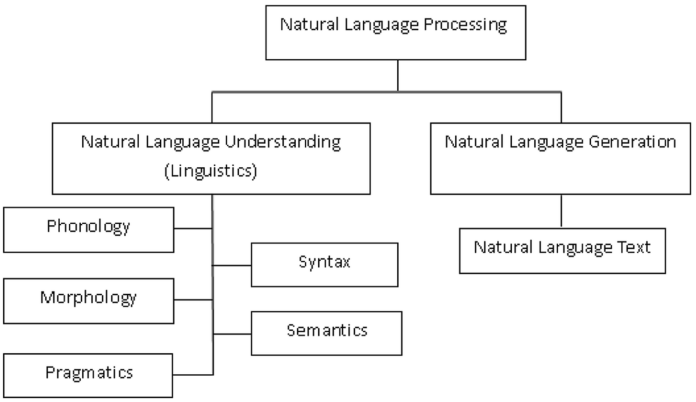


Figure 2.1.: Broad Classification of NLP from (10)

The power of NLP in understanding the human language is very helpful in identifying patterns and anomalies that might show whether a review is genuine or not. In the case of the scale and unstructured nature of the textual data, NLP is more adequate to automate the analysis of text data maintaining accuracy and efficiency compared to other traditional analytical methods.

2.3.2. Transformer Models in NLP

Transformer models are one class of deep-learning models that were introduced in the famous paper called 'Attention is All You Need' by Google researchers in 2017 (Vaswani et al., 2017) (12). This class of deep learning models repositions the traditional models like RNNs and LSTMs, which process text sequentially and find difficulties with long-range dependencies. Instead of this, Transformer models are based on self attention mechanism, which enables model to weigh the importance of each word in a sentence relative to every other word, irrespective of their distance from each other. This helps transformer model to solve the long-range dependencies issue by capturing complex dependencies and contextual relationships in textual data more accurately.

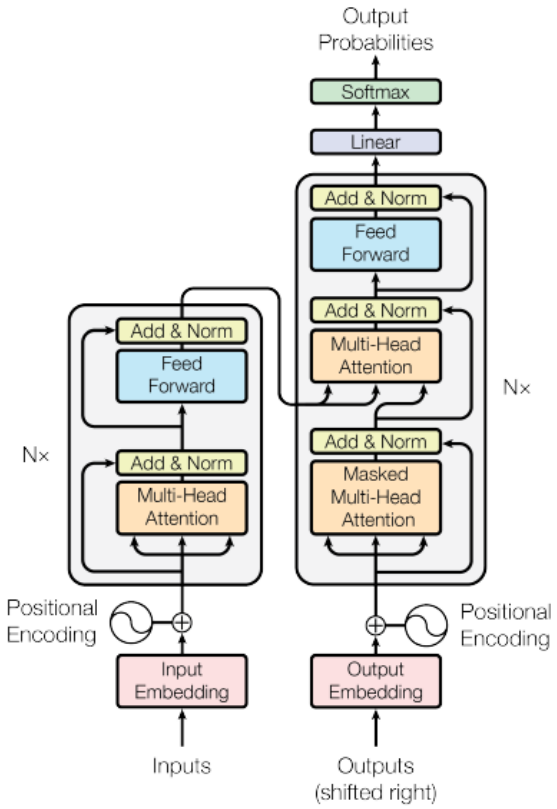


Figure 2.2.: The Transformer - model architecture from (12)

As shown in fig.2.2, the transformer architecture includes an encoder-decoder structure. Encoder works on the input text while Decoder generates the output text. Every layer of the decoder and encoder consists self attention mechanism and feedforward neural networks.

The transfomer model consists these several key components that play important roles in successful NLP tasks:

1. **Self-Attention Mechanism:** This mechanism allows the model to understand complex relationships between words by focusing on different parts of the input sequence simultaneously.

2. **Positional Encoding:** This helps to inject information about the relative position of the words into model as transformers process text non-sequentially.

3. **Multi-Head Attention:** This component extends the self-attention mechanism to enable the model to work on different aspects of the input text in parallel. Eventually, it makes model to capture various linguistic features.

4. **Feedforward Neural Network:** After processing in self-attention mechanism layers, this component allows the model to learn more about complex patterns in textual data.

5. **Layer Normalization and Residual Connections:** This helps the model to stabilize and faster the training process, ensuring efficiency and effectiveness of the model.

The evolution of Transformer models become a foundational element in the development of various state-of-the-art models like BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer). which have shown extraordinary results in language modeling, machine translation, and question-answering (13). Processing text data parallelly rather than sequentially made the transformer the best choice in the case of large datasets, pushing the boundaries of traditional NLP models.

2.4. The RoBERTa Model

RoBERTa (Robustly Optimized BERT Pretraining Approach) is a model based on Transformer architecture. Evolved as an improvement to the BERT model, RoBERTa has shown better performance in the domain of NLP tasks including text classification which is a key focus of this thesis. In this section, we will explore every details about RoBERTa model.

2.4.1. From BERT to RoBERTa

The world of NLP has witnessed a monumental shift with the advent of transformer models, particularly with Google's introduction of BERT (Bidirectional Encoder Representations from Transformers) in 2018 (13). It was a groundbreaking development in transformer models due to its bidirectional training approach, where the model learns from both the right and left side context of a word in a sentence. This development made BERT, a revolutionary in NLP tasks by overpowering its predecessors.

BERT's architecture is based on transformer models, utilizing the encoder part of the transformer architecture by employing a technique called Masked Language Model(MLM). Additionally, BERT has the presence of Next Sentence Prediction (NSP). These two main tasks, responsible for pre-training, involve masking and training for prediction of the random worlds(MLM) and prediction of the sequence of sentences for the understanding sentence relationshipsNSP. BERT is trained on a massive corpus of text and then fine-tuned for specific tasks, setting new benchmarks across a range of NLP tasks, including question answering, language inference, text classification and sentiment analysis. Its worth lies not only in its performance but also in its approach to contextuality and bidirectionality, which allows for a deeper understanding of linguistic variations (14).

Due to some limitations in BERT, the Facebook AI team introduced a modified BERT model, called RoBERTa in 2019 (15). RoBERTa is just a refinement of BERT model by making changes in training procedure, amount of training data, and optimizing certain architectural blocks to improve performance. In particular, RoBERTa was trained on a dataset of 160GB of text, which is more than 10 times larger than the dataset used to train BERT as shown in fig.2.3. Additionally, RoBERTa uses a dynamic masking technique during training that helps the model learn more robust and generalizable representations of words (16).

Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7
$\text{XLNet}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	94.0/87.8	88.4	94.4
+ additional data	126GB	2K	500K	94.5/88.8	89.8	95.6

Figure 2.3.: Development set results for RoBERTa as pretraining over more data (16GB → 160GB of text) and pretraining for longer (100K → 300K → 500K steps) from (15)

By comparison, the RoBERTa model often comes out on top than the BERT model. Its improved training process helps the model to cover more language complexities in a better way. Additionally, removing NSP task used in BERT, makes model more focused on MLM resulting in a simplified training objective and improved understanding of context. However, this performance improvement always comes with a cost of efficiency. The requirement of more computational resources for pretraining of RoBERTa, becomes a limiting factor for researchers and normal practitioners without having high-end computational power. In fig.2.4, we can see the key differences between these two models.

Aspect	BERT	RoBERTa
Training Data	BookCorpus + English Wikipedia (3.3 billion words)	10x more data including CommonCrawl News, OpenWebText, and more (160GB of text)
Training Procedure	Standard training methodology	More iterations, larger mini-batches, and longer sequences during training
Batch Size	Smaller batch size	Larger batch size
Sequence Length	Max sequence length of 512 tokens	Max sequence length of 512 tokens, dynamically changed
Next Sentence Prediction (NSP)	Used in pre-training	Removed from pre-training
Dynamic Masking	Static (fixed during pre-training)	Dynamic (changes during pre-training)
Computational Resources	Considerable, but less than RoBERTa	Significantly more, due to longer training times and larger datasets

Figure 2.4.: The key differences between BERT and RoBERTa from (14)

2.4.2. Architecture and Features of RoBERTa

The RoBERTa model is based on transformer architecture, more focused on encoder component. The architecture of RoBERTa is almost similar to BERT architecture with a few modifications to improve the results.

Architecture of RoBERTa

The architecture includes multiple layers of self-attention mechanism and FNNs to transform input data by focusing on relationships between the words in a sequence. This makes model to build the rich and contextualized word representations.

The primary component of RoBERTa include:

- Self-attention Mechanism: It allows model to focus on various portion of the input sequence when analysing the representation of each word. It helps model to capture distant relationships between words.
- FNNs: It helps model to learn complex patterns in the data by applying non-linear transformations in each layers.
- Layer Normalization: This technique is applied to stabilize and accelerate the training process.
- Residual Connections: This connection helps to make model more efficient and less effected to the vanishing gradient problem.

In fig.2.5, we can see the architecture of RoBERTa model with some additional components and processes implemented for legal citation recommendation tool (17).

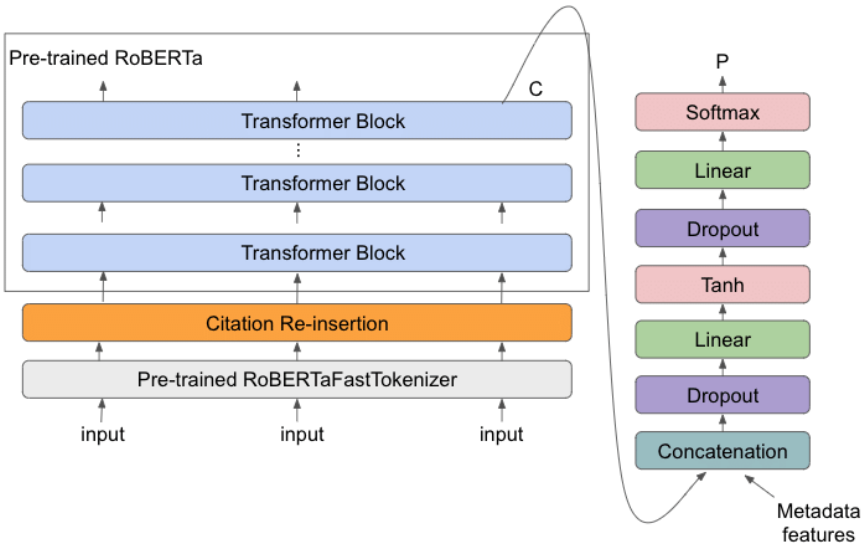


Figure 2.5.: The architecture of BERT model tailored for citation recommendation task from (17)

This architecture shown in fig.2.5, consists mainly three parts. Input and pre-trained RoBERTaFast-Tokenizer which process the input data and convert into tokens with pre-trained information. After tokenization, there's Encoder part consisting Citation re-insertion(custom layer), Transformer blocks for self-attention mechanism and FNNs, and the final output denoted as C in contextualized representation of input data encoded with the semantic information from the pre-training. Third part includes additional features and layers for custom results such as Concatenation, Dropout, softmax etc.

Features of RoBERTa

The RoBERTa model has several key features and advantages that contribute to its robustness and performance:

- **Bidirectional Contextualization**: Like BERT, this features gives ability in RoBERTa model to look at both preceding and succeeding text, eventually helps into building more precise word representations.
- **Dynamic masking**: This technique helps model on downstream major NLP tasks. In contrast to the static masking used in the original BERT model, which masks the same tokens at every epoch of pre-training, dynamic masking involves randomly masking different tokens at different points during pre-training. This lead model to learn more robust and generalizable representations of language by forcing it to predict missing tokens in a variety of different contexts.
- **Full sentence without NSP**: Since the NSP is not employed in RoBERTa model, the problems such as challenge of producing negative samples and the chance of adding biases to the pre-trained model can be avoided. Rather than training the model on sentence pairs, RoBERTa can develop a more dependable representation of the language by using whole sentences (18).
- **Larger BPE**: RoBERTa uses a larger BPE vocabulary size compared to BERT. BPE is a kind of sub-word tokenization that facilitates the efficient handling of uncommon and non-vocabulary words. In comparison to BERT, RoBERTa employs a more aggressive BPE algorithm, which results in a higher number of sub-word units and a more detailed representation of the language (18).
- **Scalability: RoBERTa** model has ability to be scaled to handle large datasets and longer training times, which makes model to more efficient in capturing nuanced patterns in textual data.

Working of RoBERTa

RoBERTa is trained beforehand using a sizable corpus of text. This is a high-level summary of how it functions:

1. **Pre-training:** Before being used, RoBERTa needs to be pre-trained on a sizable text corpus. During pre-training, a portion of the tokens in each sentence are randomly masked. The model is then trained to predict the masked tokens using the context that the unmasked tokens give. This is known as the disguised language modeling objective. For use cases, already pre-trained model can be used using Huggingface library.
2. **Fine-tuning:** After pre-training, the model can be fine-tuned for particular NLP tasks, such as named entity recognition, sentiment analysis, text classification or question answering. The model is trained using the pre-learned weights as initialization on a smaller dataset tailored to the job at hand during fine-tuning as shown in fig.2.6.
3. **Inference:** Once the model has been fine-tuned, it may be applied to new texts for inference. This involves feeding the text into the network and use the learnt representations to make the predictions.

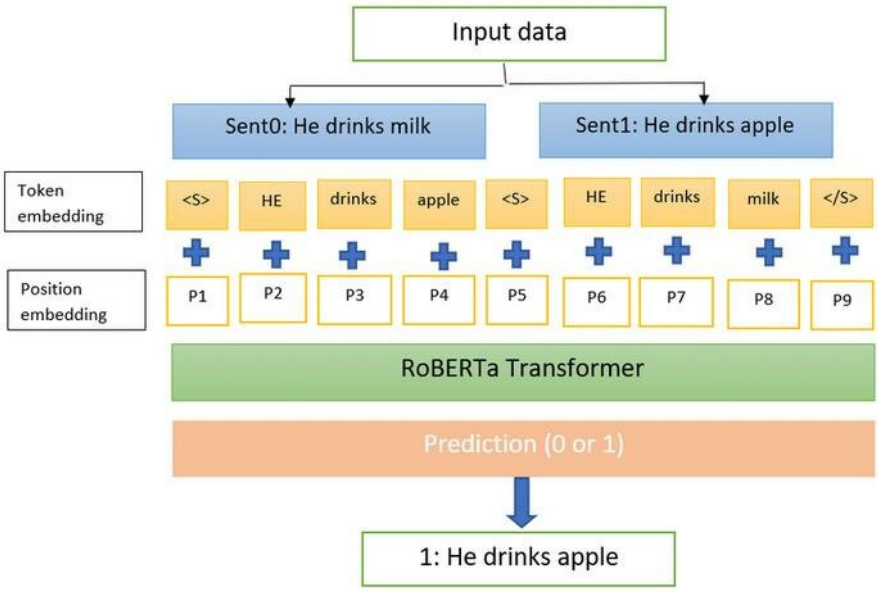


Figure 2.6.: The representation of working of pre-trained RoBERTa model from (19)

Performance and Application of RoBERTa

RoBERTa has shown outstanding results in large range of NLP benchmarks. It has surpassed BERT and other modern models in tasks such as GLUE (General Language Understanding Evaluation), SQuAD (Stanford Question Answering Dataset), and RACE (ReAding Comprehension from Examinations) as shown in fig.2.3. These NLP benchmarks tests the capability of models for tasks such as text classification, sentiment analysis, question answering, etc, focusing on improved ability of RoBERTa in understanding complex language patterns and structures.

Due to its high-level abilities, RoBERTa is broadly used in NLP use cases. It can be used in various applications such as text classification to classify the text in attributes, sentiment

analysis to understand the emotional tone of text, and in machine translation for more accurate language translations. Additionally, this model's robustness makes it suitable for specific tailored tasks such a document analysis, OCR processing, and medical text processing where precision is crucial.

2.5. Machine Learning

This section will discuss machine learning and the models used in this research.

2.5.1. Introduction to Machine Learning

Machine Learning (ML) is a most evolving part of AI field majorly focused on algorithms which lead computer to learn from given data. ML algorithms are developed to understand the patterns within the data and make predictions without much human intervention. With comparison to traditional programming, this ability of computers are much essential in case of handling complex datasets. ML includes different types like supervised learning, unsupervised learning, and reinforcement learning.

By leveraging the benefits of large datasets, the capability of ML shows significant improvement in decision-making processing in various domains. The use of ML in healthcare can be the best example of predicting patients' conditions or disease outbreaks. Similarly, in finance, the power of ML can be very useful in identifying fraudulent transactions or predicting stock prices. The versatility of ML makes it very useful in modern data analysis, offering insights that are both practical and scalable. The constant development in computational resources and data quality have further boosted the development and application of ML techniques, resulting in more reliable and improved models across various tasks (20).

2.5.2. Machine Learning Approaches

ML approaches fall into three major categories: Supervised, Unsupervised, and Reinforcement Learning. In this section, we discuss these learning algorithms broadly.

1. **Supervised Learning**: Supervised learning is a basic method of learning in ML. It trains the model using an already labeled dataset. This algorithm mainly depends on the input-output pairs in the dataset where each output is connected to the input variable. The major goal of this method is to learn the mapping from inputs to outputs so that it can forecast the results for new, unseen data. Linear Regression, Logistic Regression, Decision Trees, Support Vector Machines, and Neural Networks are some of the commonly used examples of supervised learning.

   The process includes splitting dataset into a training and a test dataset. Training dataset used to train the model to learn the patterns in dataset connecting input and output data. To evaluate the trained model, the test dataset is used to assess its performance. This method is highly used in different applications such as Text Classification, Speech Recognition, and Image Recognition. Supervised learning is very important in tasks that involve prediction based on past data (21).

Here are some examples of ML models based on supervised learning (22):

- Logistic Regression: This model is widely used for binary classification problems. It uses one or more variables of the dataset to predict the binary outcome.
- Decision Trees: Decision tree models are used for both classification and regression tasks. It consists tree-like structure created from split data into subsets based on the value of input features. These models are easy to understand and can handle both numerical and categorical datasets.
- Support Vector Machines (SVMs): These models are used for classification tasks by finding the best hyperplane that separates classes in the best way. They are efficient in high-dimensional data and are mostly used in text classification and image recognition.
- Neural Networks: These models incorporate the layers of coordinated nodes commonly known as Neurons. Neural networks are inspired by the human brain and can capture complex patterns in datasets and foundational elements of deep learning.

2. **Unsupervised Learning**: In unsupervised learning, the model is trained on unlabeled data. This learning method relies on patterns and structures in the data without any prior information about the outcomes, contrary to supervised learning, which completely depends on labeled datasets to predict the result. This method of training the model is very effective in such scenarios where labeled data is unavailable or hard to obtain. Techniques such as clustering, hierarchical learning, and dimensionality reduction are commonly used in unsupervised learning to analyze and interpret complex datasets (23). For instance, hierarchical clustering creates a tree of clusters by combining and splitting exisitng clusters based on their similarities in iterations. Another widely used method, K-Means clustering, splits the data into k distinct clusters by reducing the variance within each cluster. Apriori, Clustering using K-Means, and ECLAT are some of the examples of unsupervised learning.

3. **Reinforcement Learning**: In Reinforcement Learning, an agent learns to make decisions by performing actions in an environment to maximize cumulative rewards. Reinforcement learning is highly based on the concept of learning from interactions. It uses rewards or penalties as feedback and makes improvements in future predictions. This method of trial and error lets agent to identify and understand the most optimal strategies for complex tasks.

2.6. Pattern Recognition in Machine Learning

Patter recognition is a concept of understanding the patterns and structures in the data, which makes it a very important element in data analysis and machine learning. In the context of this study, pattern recognition is used to understand the relation and impact of features with predicted outcomes. This section talks about pattern recognition to understand classification patterns in datasets.

Pattern recognition is the ability of machines to identify patterns in data, and then use those patterns to make decisions or predictions using computer algorithms. It's a vital component of modern artificial intelligence AI systems. This process let the machine to classify the data into categories and make predictions. The origin of pattern recognition can be traced back to statistical analysis and engineering, but it has been Incorporated in machine learning techniques due to the presence of big data and high computation powers. Pattern recognition systems are commonly trained from labeled 'training' data. It has several applications such as statistical data analysis, signal processing, image analysis, computer graphics, bioinformatics, and information retrieval (24).

One of the foremost applications of pattern recognition is in image processing and signal processing. For example, in medical image processing, the pattern recognition algorithm can detect tumors or any other anomalies in CT scans or X-rays, which can help in early diagnosis and treatment. Same way, these algorithms enable facial recognition systems in the computer vision field to be used in the security and authentication process.

Pattern recognition also plays an important role in the development of NLP. These algorithms help analyze and understand human language and have become useful in applications such as speech recognition, sentiment analysis, and machine translation. For instance, virtual assistants like Alexa and Siri highly rely on these algorithms to interpret and respond to user instructions accurately.

A pattern can either be seen physically or it can be observed mathematically by implementing algorithms. In pattern recognition, pattern consists of two fundamental things: The collection of observations and the concept behind the observation.

There are some foundational principles and design considerations are important in pattern recognition (25):

1. **Feature Representation/Importance**: It shows the impact of a feature that is more relevant to the problem and can capture the underlying structure easily.

2. **Similarity Measure**: A similarity measure is helpful in comparison between two data points. For different types of data or different problems, different similarity measures are appropriate.

3. **Model Selection**: It is very important to select the model that can fit the data in a better way and understand the pattern easily.

4. **Evaluation**: Evaluating the performance of pattern recognition systems using suitable matrices, allows us to compare the models and algorithms that can be best fit for the dataset.

5. **Feature Selection**: It is the process of selecting the subset of the most important feature from the data which can lead to improved performance and reduced complexity of the model.

2.7. Feature Analysis of Classification Patterns

In the realm of machine learning, understanding the impact of the individual feature on the model's predictive performance is very crucial. For this, there are various methods in machine learning. We discuss the regression table analysis, the basics of feature importance, various methods for calculating feature importance, and feature importance in different machine learning models.

2.7.1. Features in Machine Learning

In machine learning, features are individual variable properties generated in a dataset and used as input in ML models. The concept of "feature" is related to that of explanatory variables used in statistical techniques such as linear regression. There are commonly two types of features used: Continuous or Numerical features and Categorical or Discrete features. Continuous features are numerical values that can take on any value within a certain range. This type of data is often used to represent things such as time, weight, income, temperature, etc. Categorical features are an important part of machine learning. Categorical data is data that can be divided into categories, such as 'male' and 'female' or 'red' and 'blue'. For instance, below table 2.1 shows some categorical features like 'Position' and numerical features like 'Goals', 'Height', and 'Salary'.

Goals	**Height**	**Position**	**Salary**
5	1.78	attacker	100k
6	1.58	defender	120k
3	1.55	midfielder	90k
...	...	...	...
1	1.80	defender	120k

Table 2.1.: Table with numerical and categorical features

Features are one of the most important components in ML modeling. Model predictions directly depend on the quality of features. As a result, in ML, a special emphasis is placed on feature engineering and feature selection. To know which features contribute more and how to the model and, provided that, select or create new features, we would need to measure and anaylse their coefficient and importance somehow.

2.7.2. Regression Table Analysis

Regression analysis is one of the widely used analysis method in machine learning for understanding feature patterns and behavior. In simple words, it is a statistical method that explains the strength of the relationship between a dependent variable and one or more independent variable(s). There are majorly three types of regression analysis models:

1. **Linear Regression**: Simple linear regression is a model that assesses the relationship between a dependent variable and an independent variable and finds a linear function(straight

line). This regression model can be used when the dependent variable is quantitative.

2. **Multiple Linear Regression**: Multiple linear regression analysis is essentially similar to the simple linear model, with the exception that multiple independent variables are used in the model.

3. **Nonlinear Regression**: Nonlinear regression refers to a broader category of regression models where the relationship between the dependent variable and the independent variables is not assumed to be linear. There are majorly two types of nonlinear regression: Parametric non-linear regression(Polynomial regression, Logistic regression, Exponential regression, Power regression, etc.) and Non-parametric non-linear regression(Local polynomial regression, Nearest neighbor regression, etc.).

Regression Table: The regression table displays the statistical product of a regression analysis. It gives a brief description of the relationships between the dependent variable (the target value that you are trying to predict) and one or more independent variables (the factors that drive the output). There are various parameters of the regression table which are explained below:

1. **Coefficient (Coef.)**: The coefficient represents the estimated change in the dependent variable for a one-unit change in the independent variable, holding all other variables constant. The positive coefficient displays a positive effect on the outcome and the negative coefficient shows a negative effect on the outcome.

2. **Standard Error (Std.Err.)**: The standard error tells, on average, how far observed values fall from the regression line. It shows the accuracy of coefficient estimate. Smaller values indicates more precise estimation of the coefficient.

3. **z-value (z)**: It is the coefficient divided by its standard error. It scales the coefficient by its standard error. Larger values suggests the coefficient is further from zero and more likely to be significant and vice versa in case of smaller values.

4. **p-value (P$>|z|$)**: p-value signifies the likelihood that the observed correlation was by random chance. It tests the null hypothesis that the coefficient is equal to zero (no effect). A small p-value (usually less than 0.05) rejects the null-hypothesis and shows that coefficient is significantly different from zero. While a larger p-value(more than 0.05) indicates that the coefficient is not significantly different from zero.

5. **Confidence Interval ([0.025, 0.975])**: The confidence interval provides a range of values within which the true population parameter is expected to lie with 95% confidence. If the interval does not include zero, the coefficient is considered statistically significant. On the other side, the coefficient is not significant if it includes zero.

2.7.3. Introduction to Feature Importance

Feature Importance helps to understand the influence of each input feature of the dataset on the model's predicted outcome. It determines the degree of usefulness of a specific variable for a current model and prediction. It quantifies the contribution of every feature in predicting the outcome variable. Understanding feature importance eventually allows practitioners to build more interpretable models. The scores can be calculated differently depending on the algorithm. Additionally, the feature importance score helps in feature engineering, where new features are created based on the most impacting ones, possibly resulting in better model performance (26).

It is important in ML tasks, as it allows practitioners to gain valuable information about features. This information can be used in a variety of ways (27):

- **Feature Selection**: Users can select the subset of the most relevant features to use in developing the model, resulting in less dimensionality and noise in data, and improved model performance.
- **Model Interpretability**: By finding the most important features, practitioners can understand the underlying patterns between the features and the outcome.
- **Model debugging**: It can be used to find the irrelevant model which results in the non-working of the model,
- **Business decision-making**: It gives the option to find the importance of the features which can lead in informed decisions about which features to collect and how to manage the resources.
- **Model Performance**: By separating the irrelevant features, the model performance can be improved by less over fitness and training time.

In summary, feature importance is a highly valuable method in machine learning algorithms, giving information about the relevant significance of different features.

2.7.4. Methods For Evaluating Feature Importance

Feature importance score can be determined by various methods, but generally, it can be divided into main two groups:

1. **Model Agnostic Methods**: This group of methods is not specifically related to one particular ML model or algorithm but can be applied to any model, regardless of its underlying architecture or complexity.
2. **Model Dependent Methods**: This type of method for feature importance is specific to a particular machine learning model or algorithm which is built into the model as its part. It is calculated using techniques that are dependent on the specific model being used.

Here, we discuss some widely used methods, including their mathematical foundations and practical applications.

- **Permutation Feature Importance**: Permutation feature importance is a widely used model-agnostic feature importance method that measures the change in the model's performance when the values of a particular feature are randomly shuffled. A feature is 'important' if shuffling its values increases the model error because in this case, the model relied on the feature for the prediction. A feature is 'unimportant' if shuffling its values leaves the model error unchanged because in this case, the model ignored the feature for the prediction (28).

  This technique is easy to implement and can be applied to any model, which makes it a highly acceptable choice for understanding feature impacts. The permutation feature importance can be calculated as:

$$Importance(X_i) = \frac{1}{R}\sum_{r=1}^{R}(Error_r - Error_{r,perm}(X_i))$$

  where:

  - R is the number of repetitions to average the randomness.
  - $Error_r$ is the error in the model on the original data in the $r-th$ run.
  - $Error_{r,perm}(X_i)$ is the error of the model after permuting (X_i) in the $r-th$ run.

  This method of calculating feature importance using permutation is effective. Still, it can be very expensive when large datasets and complex models are present, as it requires high computation power for multiple evaluations of the model.

- **SHAP Values**: SHAP (SHapley Additive exPlanations) is a unified measure of feature importance with model-agnostic behavior. SHAP values are based on a theoretical framework rooted in cooperative game theory. They have several useful properties like additivity, local accuracy, missingness, and consistency that make them effective for interpreting models. SHAP values allocate an importance value to each feature by checking all possible affiliations of features and their impact on the model's predictions. Features with positive SHAP values positively impact the prediction, while those with negative values have a negative impact. The magnitude is a measure of how strong the effect is.

  The SHAP value for a feature i is computed as:

$$\phi_i = \sum_{S\subseteq\{1,\ldots,n\}\setminus\{i\}} \frac{|S|!\cdot(n-|S|-1)!}{n!}\cdot[f(S\cup\{i\}) - f(S)]$$

  where:

  - ϕ_i displays the SHAP values of feature i,

- S is subset of features not including i,
- n is the total number of features,
- $f(S)$ is the model output using only the features in S.

SHAP values have been recognized for their usefulness in complex models like deep learning by their ability to provide consistent and local explanations. However, the exact computation of SHAP values is challenging. But, it can be achieved by combining current additive feature attribution methods with approximation methods like KernelSHAP(Model-agnostic), MaxSHAP(Model-dependent), and DeepSHAP(Model-dependent) (29).

- **Feature Importance in Tree-Based Models**: Tree-based models like Decision Trees, Random Forests, Gradient Boosting Machines, and Extra Tree classifiers, provide inbuilt feature importance metrics as a part of their output. It is based on reducing the criterion used to select split points generally following Gini or Entropy impurity measurements.

  This feature importance score $Imp(X_j)$ are computed as:

$$Imp(X_j) = \sum_{t \in T_j} \frac{N_t}{N} \cdot \Delta I(t, X_j)$$

  where:

  - T_j is the set of ll nodes in all trees that split on feature j,
  - N_t is number of samples reaching node t,
  - $N$is total number of samples,
  - $\Delta I(t, X_j)$ is the descrease in impurity (Giny/Entropy) at node t due to the split on X_j

  The results in these methods are easy to interpret and tree-based models are usually robust of multicollinearity, which makes this technique widely acceptable (28).

- **LIME (Local Interpretable Model-agnostic Explanations)**: LIME is another model-agnostic technique that focuses on providing local explanations for individual predictions rather than global feature importance. The technique attempts to understand the model by perturbing the input of data samples and understanding how the predictions change. LIME modifies a single data sample by tweaking the feature values and observes the resulting impact on the output. It is majorly used for black-box ML models, leverages simple and understandable ideas, and does not require a lot of effort to run.

2.7.5. ML Models Used for Feature Importance

In this research, we employ various ML models to understand the impact of every feature on the outcome by evaluating feature importance. In this section, we discuss these models and their feature importance broadly.

1. **Logistic Regression**: Logistic Regression is a fundamental statistical model majorly used for binary classification tasks. It gives the probability that a given input X belongs to a particular classic using the sigmoid function as shown in fig. 2.7.

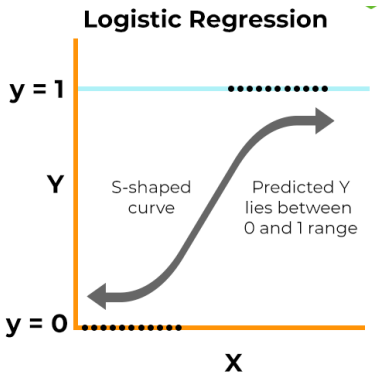


Figure 2.7.: Logistic Regression Curve

The model assumes linear relationship between the features X and the log-odds of the outcome. The logit function is expressed as:

$$\log\left(\frac{1 - P(y = 1 \mid X)}{P(y = 1 \mid X)}\right) = \beta_0 + \sum_{i=1}^{p} \beta_i X_i$$

The probability $P(y = 1 \mid X)$ is calculated using:

$$P(y = 1 \mid X) = \frac{1}{1 + e^{-(\beta_0 + \sum_{i=1}^{p} \beta_i X_i)}}$$

here:

- $P(y = 1 \mid X)$ is the probability that the outcome is 1,
- β_0 is the intercept,
- β_i are the coefficients associated with each feature X_i

Feature Importance in Logistic Regression: Feature importance is directly related to the magnitude of the model coefficients β_i in the Logistic Regression model. A higher absolute value of the coefficient shows a stronger impact of that feature on the outcome of the model. Additionally, the sign of β_i tells whether the feature increases(positive sign) or decreases(negative sign) the likelihood of the prediction. This method for feature importance is straightforward, resulting in easy-to-understand. , it does not naturally handle feature interactions.

2. **Random Forests**: Random Forests is an algorithm resulting from the combination of multiple decision trees during training to improve predictive accuracy and control over-fitting. Random Forests solve the over-fitting problem in Decision Trees by introducing two ways: Bootstrap Sampling and Random Feature Selection (26).

   The prediction is prepared by majority voting among the trees as shown in fig. 2.8. If each tree T_b where $b = 1, 2..., B$ in model provides class outcome, the main outcome is:

$$\hat{y} = \text{mode}\left(\hat{y}^1, \hat{y}^2, \ldots, \hat{y}^B\right)$$

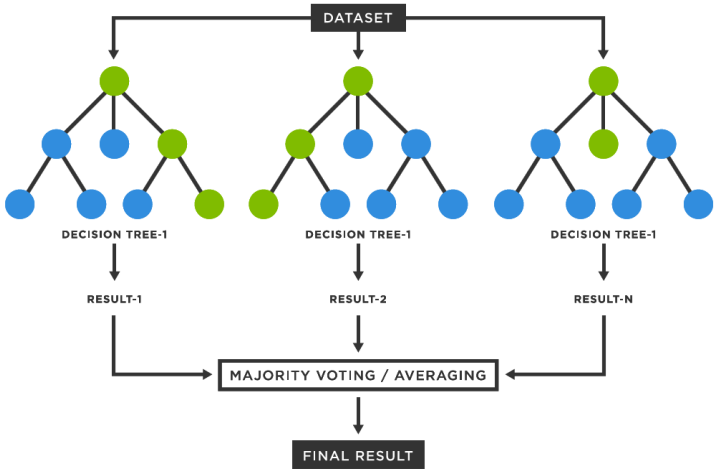


Figure 2.8.: Random Forest Diagram

Feature Importance in Random Forests: In Random Forests, the feature importance is derived from impurity reduction. The model calculates how much each feature contributes to decreasing impurity across all the trees in the forest. The Gini Impurity(for classification) for a node is calculated as:

$$I_g(p) = 1 - \sum_{i=1}^{C} p_i^2$$

Where:

- p_i is the proportion of samples belonging to class i at that node,
- C total number of classes.

The feature importance score for feature X_j is derived by summing up the contributions of impurity reduction across all trees:

$$\text{Imp}(X_j) = \sum_{t \in T_j} \frac{N_t}{N} \cdot \Delta I(t, X_j)$$

Where:

- T_j is the set of all nodes of all trees that split on X_j,
- $N_t$number of samples reaching node t,
- $\Delta I(t, X_j)$ is reduction in impurity at node t due to the split on X_j

This method can handle no-linear relationships and can detect interactions between features.

3. **XGBoost**: XGBoost (Extreme Gradient Boosting) is an optimized approach of the gradient boosting algorithm. It is an ensemble learning method that combines the predictions of multiple weak models to produce a stronger prediction.

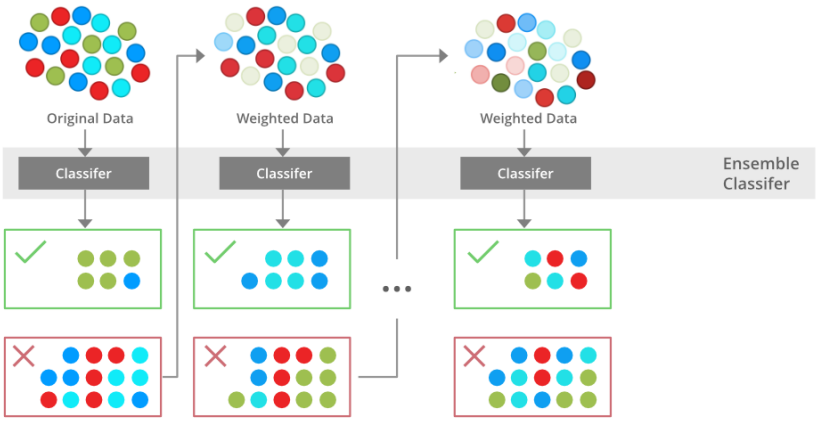


Figure 2.9.: XGBoost working from (30)

In XGBoost, each new tree repairs the errors made by the previous tree, intending to optimize the objective function. The objective function in XGBoost consists of both loss and regularization terms to limit model complexity (31):

$$\text{Obj}(\Theta) = \sum_{i=1}^{n} L(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

Where:

- Θ are the parameters of the model,
- $L(y_i, \hat{y}_i)$ is the loss function,

– $\Omega(f_k)$ represents regularization term for the tree functions f_k.

One of XGBoost's unique features is its effective handling of missing values, which enables it to handle real-world data with missing values without requiring heavy pre-processing. Furthermore, XGBoost includes support for parallel processing, allowing you to train models on big datasets in a reasonable amount of time (30).

Feature Importance in XGBoost: There are several methods to calculate feature importance in XGBoost such as weight, gain, and cover. Among them, Gain is the most commonly used. Gain calculate the improvement in accuracy resulting by feature to the branches it splits. The importance of a feature X_j in gain calculated as:

$$\text{Importance}(X_j) = \sum_{t \in T_j} \text{Gain}(t, X_j)$$

This technique is very effective in catching the contributions of features in complex, high-dimensional spaces.

4. **ExtraTrees Classifier**: ExtraTrees Classifier(Extremely Randomized Trees) is improved version of Random Forest by introducing extra randomness by selecting split points at random instead of choosing best possible split (32). This improvements reduce overfitting and result in a more generalized model, especially useful in high-dimensional datasets. The basic structure of ExtraTrees Classifier shown in fig. 2.10

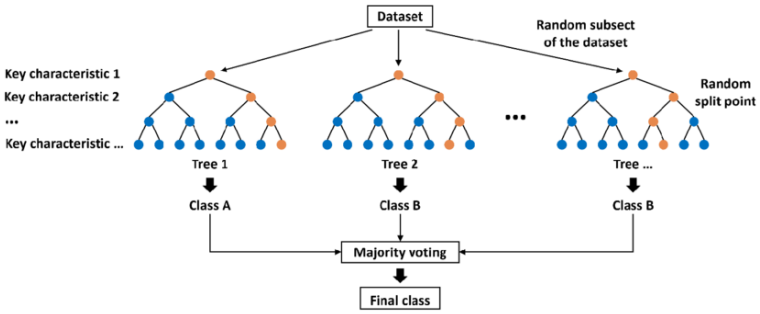


Figure 2.10.: Structure of Extra Trees from (33)

Feature Importance in ExtraTrees Classifier: The feature importance in ExtraTrees Classifier can be determined same way like in Random Forest by reducing impurity each time a feature is used to split the data (32). It can lead to different importance scores compared to more deterministic models due to more randomless, which provides a unique viewpoint on feature impact in the dataset.

3. Methodology & Implementation

The methodology & Implementation part of this thesis report gives an idea about the systematic approach to conducting the practical aspect of this research. It consists of data collection, data pre-processing, model implementations, and analysis tasks. This section is essential as it delivers a broad description of the procedures and technologies used to achieve desired research goals.

For this research, we use:

- **Python**: as a main scripting language,
- **Visual Studio Code**: as a source-code editor,
- **Jupyter Notebook**: for creating computational document,
- **SQLite**: to store dataset and perform SQL queries.

The detailed steps of implementation with methodology explanations for this research are discussed below in several parts.

3.1. Data Collection

The first footstep in this research is data collection to gather customer review data from the Amazon website. The data was prepared using web scraping techniques for multiple product categories in parts. This step was crucial for constructing a rich and diverse dataset across various product categories and subcategories that can be used in further steps. The primary attributes collected consist of:

- **Product Name**: The full product name as displayed on Amazon website
- **Category**: The base category of the product
- **Subcategory**: The base subcategory or niche category of the product
- **Review Title**: Title given to customer review
- **Review Ratings**: The star rating on a scale of 5 given by the reviewer
- **Review Description**: The detailed textual content of review

Libraries and Technologies used:

- Splash
- Docker
- BeautifulSoup
- Sqlite3

3.2. Data Manipulation and Pre-processing

After scraping customer product reviews from the Amazon website, the next stage involved required data manipulation and data pre-processing. It was very essential to prepare the data for further model training and prediction. This section discusses about the various data manipulation and pre-processing related tasks such a merging datasets, removing emojis, translating non-English reviews into English, etc.

3.2.1. Data Manipulation using SQLite

- **Database Setup**: The datasets were stored in an SQLite database. Each category had its own table, containing customer reviews with fields like id, product, subcategory, title, rating, and description. The DB browser were used to manage and perform SQL queries on database file.
- **Combining Datasets**: In data manipulation, the first task was to merge the datasets of customer reviews for various 10 product categories into a single database file using SQLite. Each category's dataset was initially stored separately, and by merging these datasets, Performing uniform processing and analysis on the entire dataset became easy.

  The SQL command shown below used in the DB browser allowed the combination of these tables into a single main review table that contains all customer reviews. This table was essential for subsequent tasks.

```
CREATE TABLE all_reviews AS
SELECT * FROM HealthPersonal
UNION ALL
SELECT * FROM Books
UNION ALL
...
UNION ALL
SELECT * FROM Electronics;
```

- **Removing Blank Data**: The combined dataset contained some records that don't have review description data. The review description was very crucial for further review classification tasks. So, it was required to remove blank records.

  SQL queries were performed to subtract the blank records in the 'description' column as shown in the code snippet. This step was important as the missing review description would not add any value to upcoming classification and analysis tasks.

```
DELETE FROM all_reviews
WHERE description IS NULL OR description = '';
```

3.2.2. Removing Emojis and Other Noisy Data

Generally, Customer product reviews often contain non-relevant characters or texts such as Emojis, Non-alphabetical symbols, etc, which don't contribute anything to further tasks. The following steps were taken to clean the data:

- **Removing Emojis**: To remove the emojis from the description column, the `'emoji'` Python library was used. The function iterated through each review, stripped out any emojis, and updated the database accordingly as shown in the code snippet.

```
import emoji

def remove_emojis(text):
return emoji.replace_emoji(text, replace='')

for row in rows:
    id = row[0]
    description = row[1]
    new_desc = remove_emojis(description)
    cursor.execute("UPDATE reviews
                    SET description = ?
                    WHERE id = ?",
                    (new_desc, id))
conn.commit()
```

- **Cleaning other Noisy Data**: Additional noise such as URLs, excessive punctuation, and other non-essential characters were removed using the cleantext library. The `cleaning_noise` function was designed to preserve the integrity of the text while removing unnecessary elements.

```
from cleantext import clean

def cleaning_noise(description):
clean_noise = clean(description,
                    lower=False,
                    no_urls=True,
                    no_numbers=False,
                    no_digits=False,
                    no_punct=False,
                    no_currency_symbols=False,
                    replace_with_punct="",
                    replace_with_url="",
                    replace_with_number="",
                    replace_with_digit="",
                    replace_with_currency_symbol="",
                    additional_allowed_characters="@$")
return clean_noise
```

```
    for row in rows:
        id = row[0]
        description = row[1]
        new_desc = cleaning_noise(description)
        cursor.execute("UPDATE reviews
                        SET description = ?
                        WHERE id = ?",
                        (new_desc, id))

    conn.commit()
```

3.2.3. Translating Non-English Text

The review dataset was taken from the Amazon USA website, which contained non-English such as Spanish language in the description of the review. To ensure consistency and effectiveness of model prediction and further analysis, all non-English reviews were translated into the English Language.

First, the language of each review description was detected using `langdetect` library. After detecting the non-English reviews, the Google Translate API was employed to translate these non-English review descriptions. The translated text replaced the original review in the dataset.

```
from langdetect import detect
from googletrans import Translator

def translate_to_english(text):
    try:
        lang = detect(text)
        if lang == 'en':
            return text
        else:
            translator = Translator()
            translation = translator.translate(text,
            src=lang, dest='en')
            return translation.text
    except:
        return text

for id, description in rows:
    translated_desc = translate_to_english(description)
    cursor.execute("UPDATE reviews
                    SET description = ?
                    WHERE id = ?",
                    (translated_desc, id))
    conn.commit()
```

This translation process ensured that all reviews were standardized in English, facilitating easier review classification and analysis.

Libraries and Technologies used:

- SQLite
- DB Browser
- Pandas
- Re
- Emoji
- Clean-text
- Langdetect
- Googletrans

3.3. Review Classification Using RoBERTa

In this phase of the thesis, the objective was to classify the consumer reviews as real or fake based on review description using transformer-based pre-trained RoBERTa model. RoBERTa, an optimized version of BERT model, was selected for the text classification task for its outstanding performance in tasks related to NLP field.

Using the RoBERTa model for the classification of consumer reviews consists of various steps. The dummy dataset was taken from Kaggle containing 50% fake and 50% real customer reviews to train and evaluate the model. This dataset was processed before feeding into the model. These steps are explained more below:

3.3.1. Fine-Tuning the RoBERTa Model

- **Data Prepartion**: First, the dataset was loaded using pandas data frame and a new column 'target' was created to store numeric label values using 'label' column from loaded dataset in '0' for fake and '1' for real review. It was achieved using `'encode_label'` function as shown in code snippets.

```
df = pd.read_csv("TrainingDataSet.csv")

# Encoding labels
encoded_label_dict = {"CG": 0, "OR": 1}
def encode_label(x):
    return encoded_label_dict.get(x, -1)

df["target"] = df["label"].apply(lambda x: encode_label(x))
```

- **Model Configuration**: The model configuration steps were done before the fine-tuning of the model. It was essential to set the configurations and hyperparameters for optimized performance.

First, the model was imported from the Hugging Face's 'transformers' library. RoBERTa model has been pre-trained on a large corpus of data, making it more convenient for fine-tuning on specific review classification tasks. Additionally, the RoBERTa tokenizer was imported and initialized to convert the text into token IDs which can be easily processed by the model. This involves tokenizing the text and creating attention masks to differentiate between actual tokens and padding.

```
from transformers import RobertaForSequenceClassification, RobertaTokenizer

model_name = "roberta-base"
model = RobertaForSequenceClassification.from_pretrained(model_name)

tokenizer = RobertaTokenizer.from_pretrained(model_name)
```

Primary hyperparameters were defined to control the training process which include:

1. **MAX_LEN**: The maximum length of the input sequences (set to 256 tokens) to ensure that reviews are adequately represented while keeping within the model's input size constraints.
2. **TRAIN_BATCH_SIZE & VALID_BATCH_SIZE**: The batch size for both tasks was set to 8 which defines how many samples are processed before updating the model weights.
3. **EPOCHS**: This hyperparameter defines how many times the model will iterate over the training dataset, set to 1.
4. **LEARNING_RATE**: The learning rate (set to 1e-5) for the Adam optimizer, controlling the step size during optimization.

```
MAX_LEN = 256
TRAIN_BATCH_SIZE = 8
VALID_BATCH_SIZE = 8
EPOCHS = 1
LEARNING_RATE = 1e-05
```

Furthermore, the model was set to use CUDA GPU resources if it's available, significantly speeds up training and validation processes.

```
from torch import cuda

device = 'cuda' if cuda.is_available() else 'cpu'
model.to(device)
```

- **Tokenization**: Tokenization, in the realm of NLP and machine learning, refers to the process of converting a sequence of text into smaller parts, known as tokens. These tokens can be as small as characters or as long as words. The main objective of tokenization is to represent text in a meaningful manner that machines can understand without losing its context. By converting text into tokens, algorithms can more easily identify patterns. This pattern recognition is crucial because it makes it possible for machines to understand

and respond to human input. There are mainly three main methods for tokenization depending on the granularity of the text breakdown and the specific requirements of the task at hand:

- **Word Tokenization**: This method breaks text down into individual words effective for languages with clear word boundaries like English.
- **Character Tokenization**: In this, the text is segmented into individual characters, beneficial for languages lacking clear word boundaries.
- **Subword Tokenization**: This method breaks text into units that might be larger than a single character but smaller than a full word, displaying a balance between word and character tokenization. This approach is especially useful for languages that form meaning by combining smaller units or when dealing with out-of-vocabulary words in NLP tasks.

For this text classification task, `'roberta-base'` model from Hugging Face's library was imported. Due to its deep contextualized understanding of language, the model's architecture is a proper fit for the review classification task. The text reviews were tokenized first using a tokenizer, which converted input review description text into tokens and attention masks that the model could process. To maintain the evenness of the input size, these tokenized inputs were padded and truncated to a maximum length of 256 tokens.

- **Dataset Splitting & Dataloader**: The dataset splitting method is used to split data into two or more subsets. Generally, with two split-part splits, one part is used to evaluate the performance of the model or test the model, and the other is used to train or fine-tune the model. This technique helps to ensure that the model is accurate

  In this thesis, the dataset was split into 80:20 ratios for fine-tuning and testing tasks. The dataset splitting was performed following a Random sampling approach.

  To handle the tokenization and text transformation, the customized `'Triage'` dataset class were implemented. This class was imported from PyTorch's `'Dataset'` and customized to return tokenized inputs and corresponding labels required by RoBERTa model

```
from torch.utils.data import Dataset

class Triage(Dataset):
    def __init__(self, dataframe, tokenizer, max_len):
        self.len = len(dataframe)
        self.data = dataframe
        self.tokenizer = tokenizer
        self.max_len = max_len

    def __getitem__(self, index):
        title = str(self.data.text_[index])
        title = " ".join(title.split())
        inputs = self.tokenizer.encode_plus(
            title,
            None,
```

```
            add_special_tokens=True,
            max_length=self.max_len,
            pad_to_max_length=True,
            return_token_type_ids=True,
            truncation=True
        )
        ids = inputs['input_ids']
        mask = inputs['attention_mask']

        return {
            'ids': torch.tensor(ids, dtype=torch.long),
            'mask': torch.tensor(mask, dtype=torch.long),
            'targets': torch.tensor(self.data.target[index],
            dtype=torch.long)
        }

    def __len__(self):
        return self.len
```

After that, it was essential to split the dataset into training (80%) and validation(20%) datasets. Data Laoders were created for both tasks using `'DataLaoder'` and `'train_test_split'` functions as shown in code snippets.

```
from sklearn.model_selection import train_test_split
from torch.utils.data import DataLoader

train_dataset, valid_dataset = train_test_split(df, test_size=0.2,
                                shuffle=True, random_state=2021)

training_set = Triage(train_dataset, tokenizer, MAX_LEN)
testing_set = Triage(valid_dataset, tokenizer, MAX_LEN)

train_params = {'batch_size': TRAIN_BATCH_SIZE,
                'shuffle': True,
                'num_workers': 0}
valid_params = {'batch_size': VALID_BATCH_SIZE,
                'shuffle': True,
                'num_workers': 0}

training_loader = DataLoader(training_set, **train_params)
testing_loader = DataLoader(testing_set, **valid_params)
```

- **Model Fine-tuning**: Fine-tuning in machine learning is the process of adapting a pre-trained model for specific tasks or use cases. It has become a fundamental deep learning technique, particularly in the training process of foundation models used for complex NLP tasks. It is an approach to transfer learning in which the parameters of a pre-trained model are trained on new data. Fine-tuning can be done on entire architecture, or on only a subset of its layers, in that case, the layers that are not fine-tuned are 'Frozen'. While fine-tuning is apparently a technique used for model training, it's a process different from the conventional training process. In a training process or pre-training, the model has not yet learned anything. Conversely, fine-tuning involves a technique to train the

model further whose parameters have already been updated through pre-training. Using the prior knowledge of the base model as an initial point, fine-tuning modify the model by training it on a smaller, task-focused dataset.

In model fine-tuning, the first task was to initialize an optimizer. Here, we have chosen the 'Adam' optimizer with a pre-defined learning rate for its faster convergence and stable training process for large datasets.

```
optimizer = torch.optim.Adam(params=model.parameters(),
lr=LEARNING_RATE)
```

In the next steps, as shown in code snippets, we defined the `'train'` function for model fine-tuning with the review dataset. This includes tasks, like fine-tuning the model, computing the loss, updating the model weights using backpropagation, and reporting the accuracy after 100 iterations.

```
def train(epoch):
    model.train()
    tr_loss = 0
    n_correct = 0
    nb_tr_steps = 0
    nb_tr_examples = 0

    for _, data in enumerate(training_loader, 0):
        ids = data['ids'].to(device, dtype=torch.long)
        mask = data['mask'].to(device, dtype=torch.long)
        targets = data['targets'].to(device, dtype=torch.long)

        optimizer.zero_grad()
        outputs = model(ids, attention_mask=mask, labels=targets)
        loss = outputs.loss
        logits = outputs.logits
        tr_loss += loss

        big_val, big_idx = torch.max(logits, dim=1)
        n_correct += (big_idx == targets).sum().item()
        nb_tr_steps += 1
        nb_tr_examples += targets.size(0)

        if _ != 0 and _ % 100 == 0:
            print(f"Training Loss per 100 steps:
            {tr_loss / nb_tr_steps}")
            print(f"Training Accuracy per 100 steps:
            {(n_correct * 100) / nb_tr_examples}")

        loss.backward()
        optimizer.step()

    print(f'Total Accuracy for Epoch {epoch}:
    {(n_correct * 100) / nb_tr_examples}')
```

- **Model Validation**: Model Validation is the process of checking whether the model meets the assumptions and requirements of the chosen algorithm, and whether it fits

the data well. Validation helps in avoiding over-fitting or underfitting of the model and Performance evaluation allows to estimate the predictive power and universality of the model. .

After fine-tuning, the RoBERTa model was validated on the validation dataset. A standalone function was designed to compute the accuracy and loss of the model on unseen data. This step is very important in understanding how well the model processes or understands the new data or unseen data not encountered during fine-tuning or training.

The `'valid'` function was defined to evaluate the model on the validation dataset to compute accuracy and loss without updating the model parameters.

```
def valid(model, testing_loader):
    model.eval()
    n_correct = 0
    tr_loss = 0
    nb_tr_steps = 0
    nb_tr_examples = 0

    with torch.no_grad():
        for _, data in enumerate(testing_loader, 0):
            ids = data['ids'].to(device, dtype=torch.long)
            mask = data['mask'].to(device, dtype=torch.long)
            targets = data['targets'].to(device, dtype=torch.long)

            outputs =model(ids, attention_mask=mask, labels=targets)
            loss = outputs.loss
            logits = outputs.logits
            tr_loss += loss

            big_val, big_idx = torch.max(logits, dim=1)
            n_correct += (big_idx == targets).sum().item()
            nb_tr_steps += 1
            nb_tr_examples += targets.size(0)

    print(f"Validation Accuracy Epoch:
            {(n_correct * 100) / nb_tr_examples}")
```

After getting the desired model performance in validation, the model was saved with trained parameters using "torch.save()" function, which can be employed for future prediction use-cases.

3.3.2. Model Evaluation

Model performance evaluation is the method of measuring how well the model performs on new or unseen data. This section focuses on conducting inference for review example and evaluating the fine-tuned RoBERTa model for review classification task. This includes loading the fine-tuned model from local path, performing inference on sample texts, and evaluating the performance using the validation dataset.

- **Model Inference**: To conduct the inference for sample reviews, the step was to load the saved model from the local path. After loading the model, the sample review descrip-

tion text was used to predict the probability of a review being fake or real. It involves encoding the text into tokens, feeding them into the fine-tuned model, and interpreting the probabilities as shown in code snippets.

```
query = """I work in the wedding industry and have to work long days
, outside for most of the day. If it were not for that, I think these
.....................................................................
might be the only shoes I'd wear all summer. If you are looking for a
reasonable priced, comfortable shoe that you can wear and walk in
all day."""

tokens = tokenizer.encode(query, return_tensors="pt")
mask = torch.ones_like(tokens)

with torch.no_grad():
    logits = model(tokens.to(device),
             attention_mask=mask.to(device))[0]
    probs = logits.softmax(dim=-1)

fake, real = probs.detach().cpu().flatten().numpy().tolist()
print(f"Real Probability: {real}\nFake Probability: {fake}")
```

- **Model Evaluation**: In this steps, the model evaluation was measured by applying model for prediction on validation dataset. First, the prediction function was defined with tokenization and maskinga shown in snippet.

```
def predict(query, model, tokenizer, device="cuda"):
    tokens = tokenizer.encode(query)
    tokens = tokens[:tokenizer.model_max_length - 2]
    tokens = torch.tensor([tokenizer.bos_token_id] + tokens +
    [tokenizer.eos_token_id]).unsqueeze(0)
    mask = torch.ones_like(tokens)

    with torch.no_grad():
        logits = model(tokens.to(device),
        attention_mask=mask.to(device))[0]
        probs = logits.softmax(dim=-1)

    fake, real = probs.detach().cpu().flatten().numpy().tolist()
    return real
```

After defining the prediction function, the function was used to predict the outcomes of the validation dataset.

```
preds, preds_probas = [], []
for i, row in valid_dataset.iterrows():
    query = row["text_"]
    pred = predict(query, model, tokenizer)
    preds_probas.append(pred)
    if pred >= 0.5:
        preds.append(1)
    else:
        preds.append(0)
```

In the last step, the performance metrics such as accuracy, precision, and recall were computed to rate the model's performance on the validation dataset uisng 'sklearn' library.

```
from sklearn.metrics import confusion_matrix, accuracy_score,
                            precision_score, recall_score,
                            classification_report

y_true = valid_dataset.target.values
y_pred = preds

# Confusion Matrix
conf_matrix = confusion_matrix(y_true, y_pred)
print("Confusion Matrix:")
print(conf_matrix)

# Performance Metrics
acc = accuracy_score(y_true, y_pred)
precision = precision_score(y_true, y_pred)
recall = recall_score(y_true, y_pred)

print(f"Accuracy: {acc * 100:.2f}%")
print(f"Precision: {precision * 100:.2f}%")
print(f"Recall: {recall * 100:.2f}%")
```

- **Predictions using saved Model Parameters**: Once the model was validated and evaluated as per the expectations, the learned parameter weights were saved. Using these weights, the predictions for the actual review dataset were conducted and the reviews were classified into real and fake categories.

3.3.3. Review Classification Using Fine-Tuned RoBERTa Model

After successfully fine-tuning and evaluating the RoBERTa model, the next task was about predicting and classifying the reviews from the actual Amazon review dataset. This section describes the loading of the SQLite review database, using the fine-tuned model for predictions, and saving it for future analysis tasks.

- **Loading and Preparing The Dataset**: Before using a model for predictions, it was required to load the SQLite database file and the fine-tuned model as shown in code snippets.

```
# Load the trained model
model_path = 'local_path'
model = torch.load(model_path)
model.to(device)
model.eval()

# Function to load data from SQLite database
def load_data_from_db(db_path, table_name):
    conn = sqlite3.connect(db_path)
    query = f"SELECT * FROM {table_name}"
```

```
    df = pd.read_sql(query, conn)
    conn.close()
    return df
```

In the end, the required DataLoader was created using the Dataset class explained previously.

- **Making Predictions and Saving the Results**: To predict the review being 'fake' or 'real', the predictions function was defined with some changes focusing on `'description'` column. This function processes each batch of data, performs inference, and collects the prediction probabilities. the fine-tuned model as shown in code snippets.

```
def predict(model, loader, device):
    model.eval()
    predictions = []
    with torch.no_grad():
        for _, data in enumerate(loader, 0):
            ids = data['ids'].to(device, dtype=torch.long)
            mask = data['mask'].to(device, dtype=torch.long)
            outputs = model(ids, attention_mask=mask)
            logits = outputs.logits
            probs = logits.softmax(dim=-1)
            preds = probs[:, 1].cpu().numpy()
            predictions.extend(preds)
    return predictions

predictions = predict(model, new_loader, device)

# Add predictions to the dataframe
new_df["result"] = predictions
```

After saving the result in dataframe, the result column was created and saved back in databse file using the defined function for saving.

The results were saved in the new column 'results' in the existing dataset Furthermore, the 'prediction' column was created by binarization of the 'results' column to classify the results into '1(real)' and '0(fake)' categories with the threshold value of 0.5.

Libraries and Technologies Used:

- PyTorch
- Transformers(Hugging Face)
- Scikit-learn
- CUDA

3.4. Feature Analysis of Classification Patterns

In this section, we discuss about the Feature Analysis part of the thesis. The main objective behind this task was to understand the impact of various features on the target values. It displays how features impact the outcome indicating the direction of the outcome. Additionally, it shows which feature of the dataset has more influence in predicting whether a review is real or fake. Additionally, the descriptive analysis on the review dataset was also performed.

To anaylse the feature coefficient and its direction, the regression table was extracted using the logit function.

To ensure the robustness of the result of feature importance, multiple machine learning models were applied to extract feature importance scores. Each model has a unique method and perception in understanding how much each feature contributes to the classification task. We discuss about the models and their feature importance score below:

1. **Logistic Regression**: As a linear model, Logistic Regression doesn't have a direct function to calculate the feature importance score. Usually, the score can be obtained from the magnitude of the coefficients associated with each feature. There are various methods to obtain feature importance score in Logistic Regression. To maintain the similarity with other machine-earning models' methods, we use a method that involves taking the absolute value of the coefficients and averaging them to extract the feature importance score for each feature. This approach is commonly known as the Coefficient Magnitude method.

   In the logistic regression algorithm, the coefficients display the variation in the log odds of the outcome for a one-unit change in the predictor variable, holding all other variables constant. By determining the magnitude of these coefficients, the relative importance of each feature in predicting the outcome can be extracted.

2. **Random Forest**: Random Forests, a famous ensemble machine learning algorithm, works by building different Decision Trees during training and giving the final prediction by calculating the average of all individual tree predictions.

   There are several methods that can be employed to calculate the feature importance score in the Random Forests model, each offering unique insights. In this research, the basic in-built feature importance method has been used to keep the similarity between other models' methods. This method uses internal calculations such as Gini Importance and mean decrease in Impurity when a feature was used to split the data, to determine the feature importance score. Actually, this method calculates how much impurity was reduced within a node of a decision tree by using a specific feature to split the data. By this means, features that contributed more to reducing the impurity were considered the more important compared to others.

3. **ExtraTrees Classifier**: ExtraTrees Classifier is an improved version of the Random Forests algorithm. The feature importance scores in this algorithm are calculated by the same method involving a reduction in impurity. Basically, this machine learning algorithm differs from the Random Forests by the addition of extra randomization, which results in different feature importance scores.

4. **XGBoost**: A benefit of using XGBoost, a gradient boosting algorithm, is that after the boosted trees are constructed, it is relatively straightforward to retrieve importance scores for each attribute.

   This algorithm also consists in-built method to calculate the feature importance score. For a single decision tree, feature importance is computed as the amount by which each attribute's split point enhances the performance measure, weighted by the number of observations for which the node is responsible. The feature importance scores are then averaged over all decision trees in the model.

After extracting the feature importance score for various attributes of the dataset using different ML models, the results were aggregated to understand and identify the consistent patterns in features by finding the most important features, reducing the likelihood of model-specific biases. The process of implementation is explained in detail in the next section.

3.4.1. Descriptive Analysis on Customer Review Dataset

Before going into the feature importance analysis, it's very important to understand the basic properties of the dataset through various analytical steps. These steps consist of summarizing the dataset's main properties, using various statistical and visualization methods. These steps serve various important purposes:

- **Data Distribution**: By examining measures such as mean, median, standard deviation, and distribution of numerical features (e.g., review_length, ratings), we can identify patterns, outliers, or anomalies. This helps in understanding the overall structure of the data.

  ```
  stats1 = data.groupby('prediction')['review_length'].describe()

  stats2 = data.groupby('prediction')['rating'].describe()
  ```

  Additionally, plot visualization helps more in understanding the data distribution between all features and the outcomes.

- **Categorical Feature Relations**: Analyzing categorical variables like category, subcategory, and rating distributions allows us to identify trends or biases. It helps to understand how a particular category, subcategory, or rating has more fake or real reviews. The visualization plots give insights into categorical relations between features and review authenticity prediction.

The all visualization steps were performed using `matplotlib` and `seaborn` libraries.

3.4.2. Feature Encoding and Dataset Preparation

In the next steps, we prepared a dataset with the required feature engineering steps like one-hot-encoding and new feature creation. To make a suitable dataset for the Logistic Regression model, we use scaling methods as well.

- **One-Hot Encoding**: To make categorical variables like 'category' and 'subcategory' ingestable in machine learning models, we used one-hot encoding to convert them into numerical formats. It is very crucial step in the case of using non-numerical data into machine learning models. In below given code example, we convert the non-numeric 'category' feature into numeric using one-hot encoding

```
data_cat = pd.get_dummies( data,
                           columns=['category'],
                           drop_first=False )
```

- **Feature Creation**: To make the dataset more interpretable and understandable, we created a new feature from the existing data. We used the 'description' column of the dataset, which consists customer review description, to create the new feature 'review_length' having the length of the whole review text in numbers using `str.len()` function.

```
#creating new feature "review_length"
data['review_length']=data['description'].str.len()
```

- **Dataset Segmentation**: To evaluate the impact of different features separately, we created individual pairs of datasets (Feature dataset and Target dataset) focused on specific feature sets. This allows for a more granular analysis of how each feature or group of features contributes to the model's predictions. For different purposes, the individual data frame was created. In the code snippet below, the example of dataset preparation for the 'category' feature can be seen.

```
x_cat = data_cat.drop(columns= ['review_length', 'subcategory',
                                'rating', 'result', 'product',
                                'title', 'description',
                                'language', 'id', 'prediction',
                                'preds'])
y_cat = data_cat['prediction']
```

- **Scaling**: Scaling the features was a very important part of data preparation. Tree-based models can be trained without the scaling but, linear models like Logistic Regression require all features to be on a comparable scale before feeding into the model. The scaling was done using `StandardScaler()` from `sklearn.preprocessing` library.

```
scaler = StandardScaler()
x_rr_scaled = scaler.fit_transform(x_rr)
```

3.4.3. Imbalanced Data Handling with SMOTE

Real-world datasets often suffer from class imbalance, where one class is much less frequent than others. In our case, the fake review class was very low in comparison to real review class. This inequality can result in biased models favoring only the majority class. To resolve this, we have used the Synthetic Minority Over-sampling Technique - SMOTE, which generates synthetic samples for the minority class to create a more balanced dataset.

The reason behind using the SMOTE method is its feature of creating new synthetic examples by interpolating between existing samples. Traditional techniques like random oversampling

can be prune overfitting by duplicating existing minority class samples. Using the SMOTE technique reduces the risk of overfitting while improving the model's ability to generalize.

In our implementation, we used the SMOTE function in every final feature and target dataset before feeding into the machine-learning models as shown in code snippet.

```
# Example SMOTE application
smote = SMOTE(random_state=42)
xs_cat, ys_cat = smote.fit_resample(x_cat, y_cat)
```

3.4.4. Fitting Datasets into Machine-Learning Models

After preparing the datasets and addressing the class imbalance, we feed the data into various machine learning models. These trained models helped in finding the importance of each feature in review classification. The different models such as Logistic models and tree-based models, are used to make the result robust. In the given code snippet, the 'subcategory' feature has been used to train the model with prediction as a target variable. This implementation gives insights into the impact of 'subcategory' feature in review being fake or real.

```
#logistic_regression
logreg_sub = LogisticRegression()
logreg_sub.fit(xs_sub,ys_sub)

#random_forest
ranfor_sub = RandomForestClassifier()
ranfor_sub.fit(xs_sub,ys_sub)


#extraa_trees
exttre_sub = ExtraTreesClassifier()
exttre_sub.fit(xs_sub,ys_sub)

#XGBoost
xgbc_sub = XGBClassifier()
xgbc_sub.fit(xs_sub,ys_sub)
```

3.4.5. Regression Table Generation

To understand the effect and direction of features on the target variable of review being fake or real, the regression table was used. The regression table gives various parameters that can help in understanding the patterns of features in making reviews fake or real.

To extract the table, the `logit` function from `statsmodels.api` was employed as shown in below code snippet below.

```
#regression table
logit_model = sm.Logit(ys_cat, xs_cat)
result = logit_model.fit()
print(result.summary2())
summary_table = result.summary2().tables[1]
summary_table = summary_table.sort_values(by='Coef.', ascending=False)
summary_table.to_csv('reg_cat.csv')
```

3.4.6. Feature Importance Extraction

The next step was to extract and analyze the feature importance score from each model. This part helps to analyze which feature has more or less impact on making reviews fake or real.

- **Logistic Regression**: The extraction of feature importance score in the Logistic regression model was not directly possible as it doesn't have an in-built function to retrieve the score. To overcome this, we used coefficients that represent the weight of each feature in predicting the outcome. We extracted the score using the calculation of the mean of the absolute value of the coefficient.

```
coefficients = logreg_cat.coef_
fp_lr_cat = np.mean(np.abs(coefficients), axis=0)
```

- **Random Forest, Extra Trees, XGBoost**: The tree-based model provides the direct function for feature importance score. We used the in-built `feature_importances_` function to extract the score for each feature.

```
fp_rf_cat = ranfor_cat.feature_importances_ #randomforest
fp_et_cat = exttre_cat.feature_importances_ #extratree
fp_xg_cat = xgbc_cat.feature_importances_   #xgboost
```

- **Avarage the Feature Importance Score**: After extracting the feature importance score from each model, we averaged all scores from all model scores to prepare a robust list of scores for every feature. These averaged scores helped into gain insights into classification patterns and their impact on predictions.

```
# Calculating average feature importance
fp_sub['Average_Importance'] = fp_sub[['LogReg_Importance',
        'RandomForest_Importance', 'ExtraTrees_Importance',
        'XGBoost_Importance']].mean(axis=1)

# Sorting by average importance
fp_sub = fp_sub.sort_values(by='Average_Importance',
         ascending=False)
```

3.4.7. Feature Interactions

First, we analyzed feature importance scores for isolated features. In this part, we investigated how the combined effects of multiple features influence the model's prediction. This is crucial because real-world data often have relationships between features that cannot be captured by using individual features. For example, the combination of review_length and rating might be more predictive impact on the review being fake or real than either feature alone.

To capture these relationships and feature importance scores, we multiplied the features to create interaction features and used in training models. Generally, Tree-based inherently capture feature interactions due to their hierarchical structure and ability to model non-linear relationships. While, Logistic Regression being linear model, does not naturally capture interactions between features unless explicitly included. In the case of Logistic regression models, we are required to manually create the feature interaction terms by multiplying them.

In this step, we first created the interaction terms using various features by multiplying them and combining them with the existing original features. So in the final dataframe, we had original features as well as manually created interaction features. In the example below, we create feature interaction terms between rating and category(having multiple distinct values) columns.

```
#multiplying features to get interaction features
for col in categories.columns:
    interaction_features[f'rating_{col}'] = rating * categories[col]

# Combine the interaction features with the original dataset if desired
data_comb_cr = pd.concat([data_cat, interaction_features], axis=1)
```

After creating, the interaction features and combining them with the original feature, we followed the previously described steps to train the model and extract the feature importance scores. These extracted feature importance scores show the impact of the feature as an isolated and as a combined with other features.

Libraries and Technologies:

- Scikit-learn
- NumPy
- Pandas
- XGBoost
- Matplotlib
- Seaborn
- Imbalanced-Learn
- SMOTE.
- Statsmodels

4. Results

In this section, we describe the results achieved from this research. It includes the collected database results, RoBERTa implementation, and final feature analysis. We present the important findings from our research, focusing on the review classification task and feature coefficients and importance results captured by employing various machine learning models, as well as the insights derived from feature interaction. The results are divided into various sections, each highlighting a key aspect of this study.

4.1. Amazon Customer Review Dataset

After successfully executing the web scraping method and combining the dataset of various categories, the final dataset of Amazon Customer Review was ready to use for further purposes. The dataset contains around 50000 reviews from various categories and products. The dataset was taken in month of February 2024. All reviews are the most recent reviews for each product. In table 4.1, the example of some records of reviews can be seen.

ID	Product	Category	Subcategory	Title	Description	Review Length	Rating
1	Oral-B Pro	Health&Personal	personalcare	Good but loud	I like that it	47	5.0
2	Echo Glow ..	Home&Kitchen	Lighting	grate nightlight	works well and...	203	3.0
3	Little Remedies..	Health&Personal	babycare	it works	easy to use, ...	134	4.0
...	...	...	...	...	...	...	...

Table 4.1.: Amazon Customer Review Dataset

Overview of the datset: The dataset consists of customer reviews from various product categories, each containing specific attributes. Below table 4.2 depicts the detailed description of the columns in the dataset.

Column	Description
product	The name of the product
subcategory	The subcategory the product belongs to
title	The title of the review
rating	The rating given by the customer (scale from 1.0 to 5.0)
review length	The length of review description
description	The text content of the review
Category	The broader category of the product

Table 4.2.: Column Overview

- **Numerical columns**: The dataset consists of two numerical columns: Ratings and Review Length.
    1. **Ratings**:
        a) Mean=4.5363
        b) Standard deviation=0.8059
        c) Minimum Value=1.0
        d) Maximum Value=5.0
    2. **Review Length**:
        a) Mean=400.6065
        b) Standard deviation=681.1548
        c) Minimum Value=10
        d) Maximum Value=24153
- **Categorical Columns**: The dataset contains Category, Product Name, and Subcategory in categorical type. For each categorical column, the below list provides the count of unique values.
    1. **Category**: The dataset consists total of 10 categories: Automotive&Industrial, Appliances, Books, Fashion, Health&Personal, OfficeProducts, Sports&Outdoor, Toys&Games, Home&Kitchen, Electronics.
    2. **Subcategory**: The dataset includes a total of 50 subcategories, 5 subcategories for each category.
    3. **Product Count**: The dataset includes reviews from a total of 500 products. Each category contains 50 products, consisting of 100 reviews for each product.

4.2. Review Classification Using RoBERTa

To classify the reviews as real and fake, we implemented a transformer-based RoBERTa model.

During the training, the accuracy and loss of the model were calculated every 100 steps. In the end, the overall loss and accuracy(94%) were printed.

Similarly, The process of validation on the 20% validation dataset were performed. The overall accuracy on the validation dataset was around 96%.

After satisfactory results in fine-tuning and validation of RoBERTa, the model weights were saved and used for predictions on the actual Amazon reviews dataset. The modified database can be seen in table 4.3

ID	Product	Category	...	Description	Rating	Result	Preidiction
1	Oral-B Pro	Health&Personal	...	I like that it	5.0	0.24565	0
2	Echo Glow ..	Home&Kitchen	...	works well and...	3.0	0.75659	1
3	Little Remedies..	Health&Personal	...	easy to use, ...	4.0	0.85475	1
...	...	...	...	...		...	...

Table 4.3.: Review dataset with predictions

4.3. Feature Patterns Analysis

In this part, we derived insights into the impact of various features on the outcome prediction. Using various machine learning models, we extracted the feature coefficients and importance scores and used them for further analysis.

4.3.1. Overall Statistical Summary

In the final dataset with predictions, we gained various insights about overall numerical and categorical relationships within the features.

After the classification, we captured 43433 reviews as real and 6477 reviews as fake in total of 49910 reviews, which shows that almost 13% of customer reviews are fake in our dataset.

As shown in the table 4.5 and 4.4, we can see the statistical summary of numerical features such as ratings and review_length. The mean value of review length differs for both fake and real reviews. Also, the max values of real and fake reviews are 24153 and 17534, which shows fake reviews are shorter compared to real reviews overall. While in the case of ratings, everything seems equally distributed between fake and real reviews.

Pred.	Count	mean	std	min	25%	50%	75%	max
0	6477	410.984	698.675	2.0	83	192	433	17534
1	43433	389.396	662.22	1.0	79	186	420	24153

Table 4.4.: Summary of review_length

Pred.	Count	mean	std	min	25%	50%	75%	max
0	6477	4.539	0.79	1.0	4.0	5.0	5.0	5.0
1	43433	4.532	0.81	1.0	4.0	5.0	5.0	5.0

Table 4.5.: Summary of ratings

In the case of categorical features, we can see the distribution of fake and reviews by Category feature in fig.4.1 in pie chart form. From that chart, we can derive that the Electronics category contains the most fake reviews while the Automotive&Industrial category has the least portion of fake reviews. In real reviews, it's vice versa.

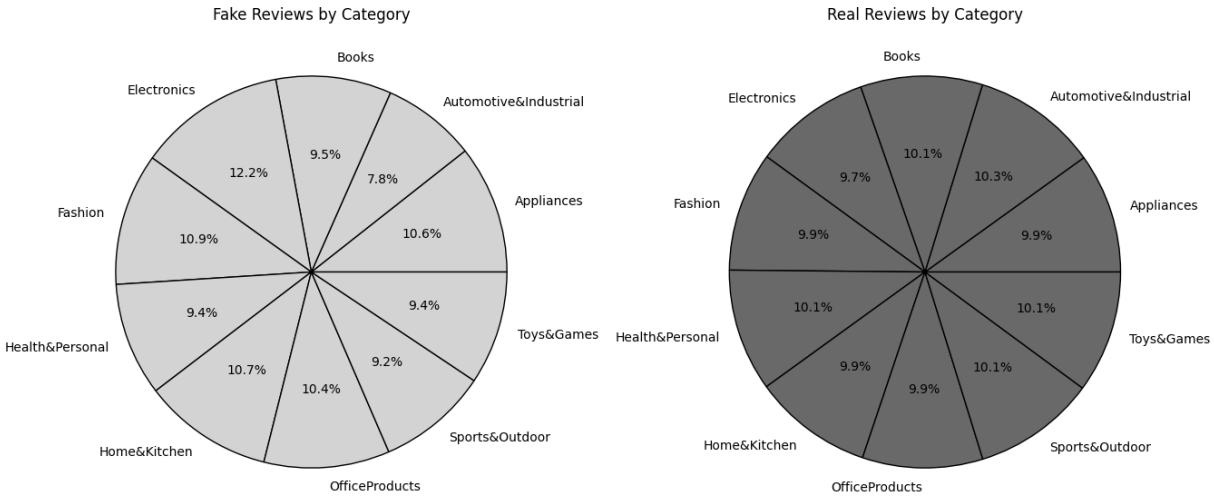
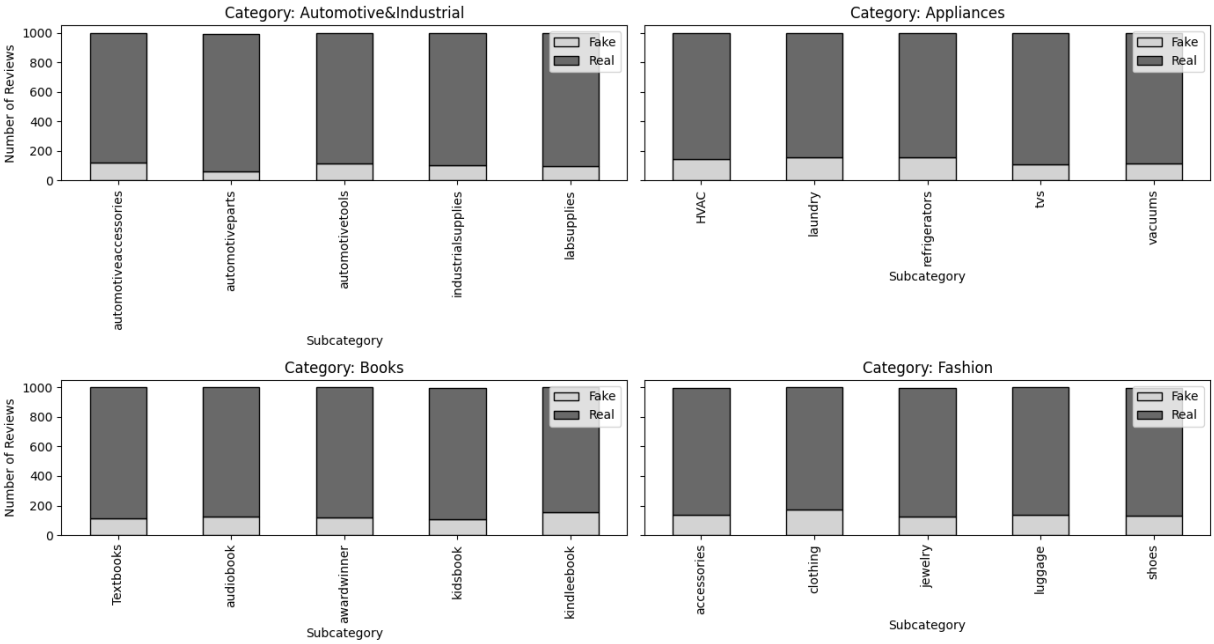


Figure 4.1.: Fake and Real Reviews Distribution

Fig.4.2 shows the proportions of fake and real reviews in the various subcategories.



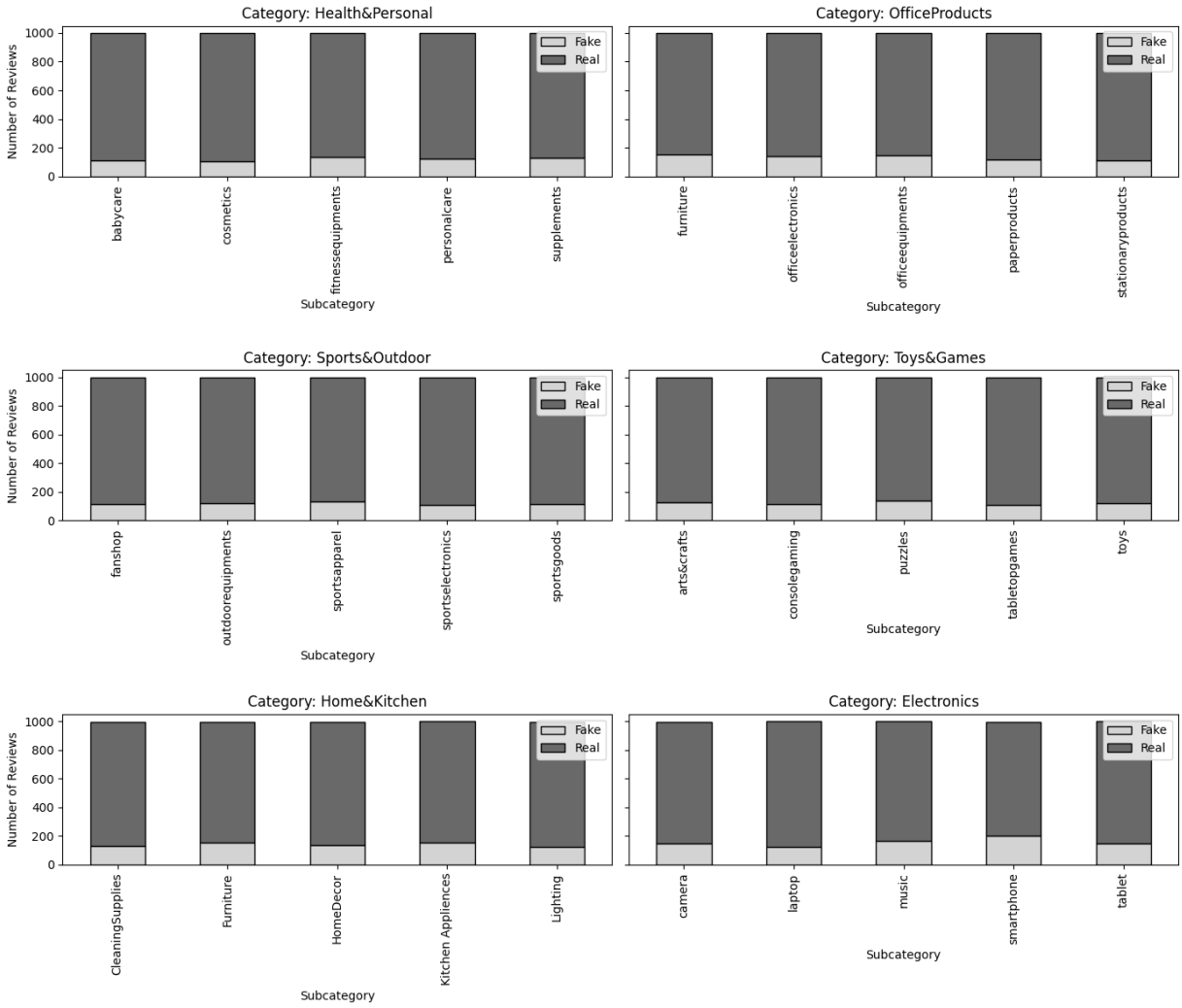


Figure 4.2.: Distribution of Reviews in Subcategories

4.3.2. Feature Analysis of Isolated Features

To extract the importance and effects of features on the review classification, we employed logistic regression for the regression table and four machine learning models for importance score.

These coefficient values and importance score help to assess the impact of different features on the prediction result of review being fake or real. In this section, we analyze the effects of individual features on the results.

'Category' Feature:

In this analysis, we focus on 'Category' variables containing different 10 categories. In the table 4.6 shows the effect of each category on the outcome. The highest positive coefficient value(0.2976) of the Automotive&Industrial category shows that this category has more likely real reviews with high significance (p-value = 0.0000¡0.05). On the other side, Electronics with the highest negative value shows that this category reviews are the most likely to be fake among all the categories listed with high significance (p-value = 0.0000¡0.05). Only the Toys&Games category is marginally significant with a p-value close to 0.05 (0.059). Since the confidence interval includes zero (Lower Bound: -0.0016 and Upper Bound: 0.083), it suggests that the effect of the Toys&Games category on the dependent variable is not statistically significant at the 95% confidence level. This means that we cannot be confident that there is a true effect of this category on reviews being real or fake.

Feature	Coefficient	Std. Error	z	P> $\|z\|$	Conf[0.025]	Conf[0.975]
cat-Automotive&Industrial	0.2976	0.0229	13.0138	0.0000	0.2528	0.3424
cat-Sports&Outdoor	0.1191	0.0220	5.4178	0.0000	0.0760	0.1622
cat-Books	0.0860	0.0218	3.9383	0.0001	0.0432	0.1289
cat-Health&Personal	0.0762	0.0218	3.5000	0.0005	0.0335	0.1188
cat-Toys&Games	0.0407	0.0216	1.8879	0.0590	-0.0016	0.0830
cat-Appliances	-0.0699	0.0212	-3.3017	0.0010	-0.1115	-0.0284
cat-OfficeProducts	-0.0708	0.0211	-3.3484	0.0008	-0.1122	-0.0294
cat-Home&Kitchen	-0.0947	0.0211	-4.4878	0.0000	-0.1361	-0.0533
cat-Fashion	-0.1029	0.0211	-4.8823	0.0000	-0.1442	-0.0616
cat-Electronics	-0.2114	0.0208	-10.1812	0.0000	-0.2521	-0.1707

Table 4.6.: Regression Table of 'Category' feature

The coefficients also depict the direction of classification. If more reviews from the Automotive&Industrial category are present in the dataset, then the result will have more real reviews.

The table 4.7 shows the Feature importance score extracted by employing different machine-learning models and in last column, the average importance score.

Feature	LR-Imp	RF-Imp	ET-Imp	XGB-Imp	Avg-Imp
category-Automotive&Industrial	0.2911	0.3877	0.4069	0.3852	0.3677
category-Electronics	0.2176	0.2295	0.2273	0.1953	0.2174
category-Sports&Outdoor	0.1128	0.0746	0.0796	0.0923	0.0898
category-Books	0.0797	0.0477	0.0498	0.0835	0.0652
category-Health&Personal	0.0698	0.0455	0.0411	0.0963	0.0632
category-Fashion	0.1093	0.0646	0.0540	0.0132	0.0603
category-Home&Kitchen	0.1010	0.0529	0.0466	0.0132	0.0534
category-Toys&Games	0.0344	0.0245	0.0309	0.1045	0.0486
category-Appliances	0.0763	0.0387	0.0308	0.0085	0.0386
category-OfficeProducts	0.0771	0.0344	0.0328	0.0081	0.0381

Table 4.7.: Feature Importance Score for 'Category'

Here, we analyze that the Automotive&Industrial has the most impact in making predictions. While OfficeProducts has the least impact. Similarly, the fig.4.3 shows the bar plot for visualization purposes.

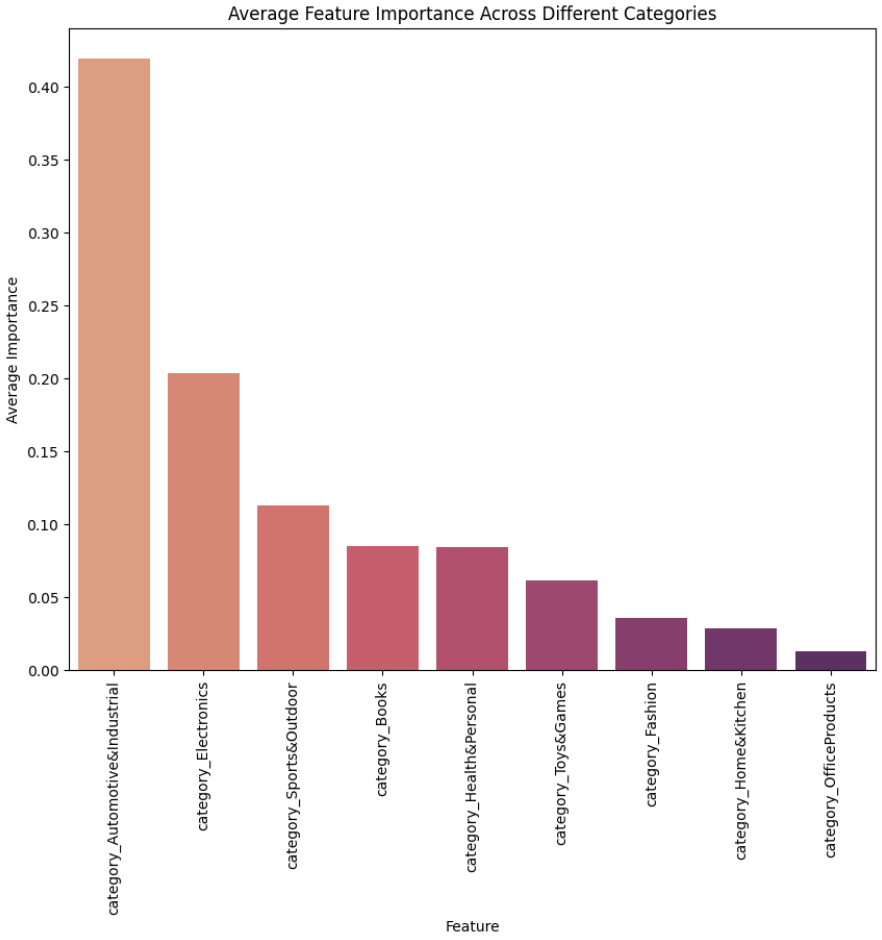


Figure 4.3.: Bar-plot of Feature Importance Score for 'Category'

'Subcategory' Feature:

Similarly, for the 'subcategory' feature having distinct 50 values, we analyze the feature effects on the classification.

Table 4.8, shows the coefficeints of the subcategory feature. The automotiveparts subcategory strongly increases the likelihood of a review being real while sub-smartphone Strongly increases the likelihood of a review being fake. Additionally, there are many subcategories such as audiobook, luggage, shoes, etc that include zero in their confidence interval, showing that these subcategories do not have a true effect on review classification.

Table 4.8.: Regression Table of 'Subcategory' feature

Feature	Coefficient	Std. Error	z	P> |z|	Conf[0.025]	Conf[0.975]
sub-automotiveparts	0.8763	0.0604	14.5045	0.0000	0.7579	0.9947
sub-labsupplies	0.3015	0.0511	5.8946	0.0000	0.2012	0.4017
sub-sportselectronics	0.2784	0.0510	5.4607	0.0000	0.1785	0.3783
sub-kidsbook	0.2696	0.0509	5.3006	0.0000	0.1699	0.3692
sub-cosmetics	0.2243	0.0501	4.4738	0.0000	0.1260	0.3225
sub-industrialsupplies	0.2220	0.0502	4.4271	0.0000	0.1237	0.3203
sub-tabletopgames	0.1877	0.0498	3.7663	0.0002	0.0900	0.2854
sub-Textbooks	0.1633	0.0497	3.2876	0.0010	0.0659	0.2606
sub-sportsgoods	0.1625	0.0495	3.2796	0.0010	0.0654	0.2595
sub-tvs	0.1607	0.0495	3.2501	0.0012	0.0638	0.2577
sub-personalcare	0.1458	0.0496	2.9404	0.0033	0.0486	0.2430
sub-awardwinner	0.1440	0.0495	2.9095	0.0036	0.0470	0.2410
sub-automotivetools	0.1366	0.0492	2.7738	0.0055	0.0401	0.2331
sub-fanshop	0.1347	0.0493	2.7297	0.0063	0.0380	0.2314
sub-babycare	0.1297	0.0491	2.6425	0.0082	0.0335	0.2259
sub-stationaryproducts	0.1234	0.0490	2.5181	0.0118	0.0274	0.2195
sub-consolegaming	0.1155	0.0491	2.3521	0.0187	0.0193	0.2118
sub-vacuums	0.1064	0.0489	2.1736	0.0297	0.0105	0.2023
sub-automotiveaccessories	0.1032	0.0490	2.1048	0.0353	0.0071	0.1993
sub-audiobook	0.0709	0.0486	1.4575	0.1450	-0.0244	0.1662
sub-paperproducts	0.0573	0.0484	1.1844	0.2362	-0.0375	0.1521
sub-Lighting	0.0468	0.0484	0.9672	0.3334	-0.0480	0.1416
sub-outdoorequipments	0.0444	0.0483	0.9183	0.3584	-0.0504	0.1392
sub-toys	0.0406	0.0482	0.8426	0.3994	-0.0538	0.1350
sub-laptop	0.0362	0.0483	0.7490	0.4539	-0.0585	0.1309
sub-jewelry	0.0280	0.0483	0.5797	0.5621	-0.0667	0.1227
sub-arts&crafts	0.0255	0.0481	0.5295	0.5964	-0.0689	0.1199
sub-sportsapparel	-0.0058	0.0481	-0.1202	0.9043	-0.1000	0.0884
sub-supplements	-0.0115	0.0479	-0.2393	0.8109	-0.1053	0.0824
sub-shoes	-0.0262	0.0478	-0.5493	0.5828	-0.1199	0.0674
sub-HomeDecor	-0.0446	0.0478	-0.9325	0.3511	-0.1384	0.0491
sub-CleaningSupplies	-0.0451	0.0475	-0.9491	0.3426	-0.1381	0.0480
sub-accessories	-0.0522	0.0477	-1.0957	0.2732	-0.1456	0.0412
sub-officeelectronics	-0.0752	0.0474	-1.5863	0.1127	-0.1681	0.0177
sub-tablet	-0.0780	0.0476	-1.6392	0.1012	-0.1712	0.0153
sub-fitnessequipments	-0.0834	0.0472	-1.7677	0.0771	-0.1759	0.0091
sub-luggage	-0.0898	0.0471	-1.9059	0.0567	-0.1822	0.0025
sub-HVAC	-0.1157	0.0470	-2.4619	0.0138	-0.2078	-0.0236
sub-puzzles	-0.1406	0.0466	-3.0183	0.0025	-0.2319	-0.0493
sub-camera	-0.1562	0.0468	-3.3349	0.0009	-0.2480	-0.0644
sub-kindleebook	-0.1742	0.0466	-3.7354	0.0002	-0.2656	-0.0828
sub-Furniture	-0.1872	0.0465	-4.0243	0.0001	-0.2784	-0.0960
sub-officeequipments	-0.1876	0.0463	-4.0510	0.0001	-0.2783	-0.0968
sub-refrigerators	-0.2241	0.0463	-4.8420	0.0000	-0.3148	-0.1334
sub-KitchenAppliences	-0.2246	0.0461	-4.8699	0.0000	-0.3149	-0.1342

Feature	Coefficient	Std. Error	z	P> |z|	Conf[0.025]	Conf[0.975]
sub-laundry	-0.2298	0.0462	-4.9713	0.0000	-0.3204	-0.1392
sub-furniture	-0.2375	0.0461	-5.1564	0.0000	-0.3278	-0.1472
sub-music	-0.3181	0.0455	-6.9857	0.0000	-0.4074	-0.2289
sub-clothing	-0.3430	0.0455	-7.5399	0.0000	-0.4322	-0.2539
sub-smartphone	-0.4841	0.0451	-10.7355	0.0000	-0.5724	-0.3957

Table 4.9 shows the feature importance score from various models and the average score. Here, we derive that the AutomotiveParts make the most impact on the review being fake or real. While Arts&Crafts shows low impact. Additionally, we can see that AutomotiveParts and Smartphones are the most impacting subcategories of Automotive&Industrial and Electronics respectively, which are also having more influence as described in previous "Category" scores. Fig.4.4 shows the visualization of these importance scores.

Table 4.9.: Feature Importance Score for 'Subcategory'

Feature	LR-Imp	RF-Imp	ET-Imp	XGB-Imp	Avg-Imp
sub-automotiveparts	0.8504	0.2411	0.2494	0.2212	0.3905
sub-smartphone	0.4983	0.1309	0.1352	0.1169	0.2203
sub-clothing	0.3576	0.0647	0.0670	0.0638	0.1383
sub-music	0.3328	0.0564	0.0582	0.0576	0.1263
sub-labsupplies	0.2870	0.0352	0.0358	0.0395	0.0994
sub-sportselectronics	0.2637	0.0302	0.0309	0.0345	0.0898
sub-furniture	0.2524	0.0319	0.0334	0.0342	0.0880
sub-kidsbook	0.2548	0.0274	0.0283	0.0340	0.0861
sub-laundry	0.2447	0.0287	0.0317	0.0319	0.0843
sub-Kitchen Appliences	0.2395	0.0296	0.0305	0.0295	0.0823
sub-refrigerators	0.2390	0.0273	0.0302	0.0304	0.0817
sub-officeequipments	0.2027	0.0221	0.0220	0.0230	0.0674
sub-cosmetics	0.2090	0.0193	0.0186	0.0227	0.0674
sub-Furniture	0.2024	0.0207	0.0220	0.0223	0.0669
sub-industrialsupplies	0.2068	0.0184	0.0185	0.0231	0.0667
sub-kindleebook	0.1894	0.0192	0.0191	0.0201	0.0619
sub-camera	0.1715	0.0164	0.0167	0.0159	0.0551
sub-tabletopgames	0.1723	0.0128	0.0125	0.0170	0.0537
sub-puzzles	0.1559	0.0145	0.0143	0.0135	0.0496
sub-sportsgoods	0.1468	0.0105	0.0083	0.0114	0.0442
sub-Textbooks	0.1476	0.0096	0.0086	0.0108	0.0441
sub-tvs	0.1451	0.0104	0.0079	0.0107	0.0435
sub-HVAC	0.1312	0.0100	0.0107	0.0109	0.0407
sub-personalcare	0.1302	0.0084	0.0064	0.0097	0.0387
sub-awardwinner	0.1283	0.0084	0.0061	0.0091	0.0379
sub-automotivetools	0.1208	0.0071	0.0052	0.0086	0.0354
sub-fanshop	0.1189	0.0065	0.0050	0.0084	0.0347
sub-babycare	0.1138	0.0057	0.0049	0.0070	0.0329
sub-luggage	0.1054	0.0074	0.0072	0.0067	0.0317
sub-stationaryproducts	0.1076	0.0056	0.0042	0.0067	0.0310

Feature	LR-Imp	RF-Imp	ET-Imp	XGB-Imp	Avg-Imp
sub-fitnessequipments	0.0991	0.0064	0.0068	0.0063	0.0297
sub-consolegaming	0.0997	0.0045	0.0035	0.0058	0.0284
sub-tablet	0.0937	0.0062	0.0063	0.0056	0.0280
sub-officeelectronics	0.0909	0.0054	0.0060	0.0053	0.0269
sub-vacuums	0.0905	0.0046	0.0028	0.0049	0.0257
sub-automotiveaccessories	0.0873	0.0052	0.0028	0.0047	0.0250
sub-accessories	0.0680	0.0039	0.0040	0.0033	0.0198
sub-CleaningSupplies	0.0608	0.0039	0.0035	0.0029	0.0178
sub-HomeDecor	0.0604	0.0034	0.0037	0.0030	0.0176
sub-audiobook	0.0549	0.0035	0.0011	0.0017	0.0153
sub-shoes	0.0421	0.0027	0.0024	0.0018	0.0122
sub-paperproducts	0.0413	0.0017	0.0008	0.0009	0.0112
sub-Lighting	0.0308	0.0014	0.0008	0.0003	0.0084
sub-supplements	0.0274	0.0021	0.0017	0.0011	0.0080
sub-outdoorequipments	0.0284	0.0011	0.0007	0.0003	0.0076
sub-toys	0.0246	0.0011	0.0007	0.0001	0.0066
sub-sportsapparel	0.0217	0.0019	0.0013	0.0009	0.0065
sub-laptop	0.0202	0.0015	0.0006	0.0001	0.0056
sub-jewelry	0.0120	0.0016	0.0007	0.0000	0.0036
sub-arts&crafts	0.0095	0.0015	0.0007	0.0000	0.0029

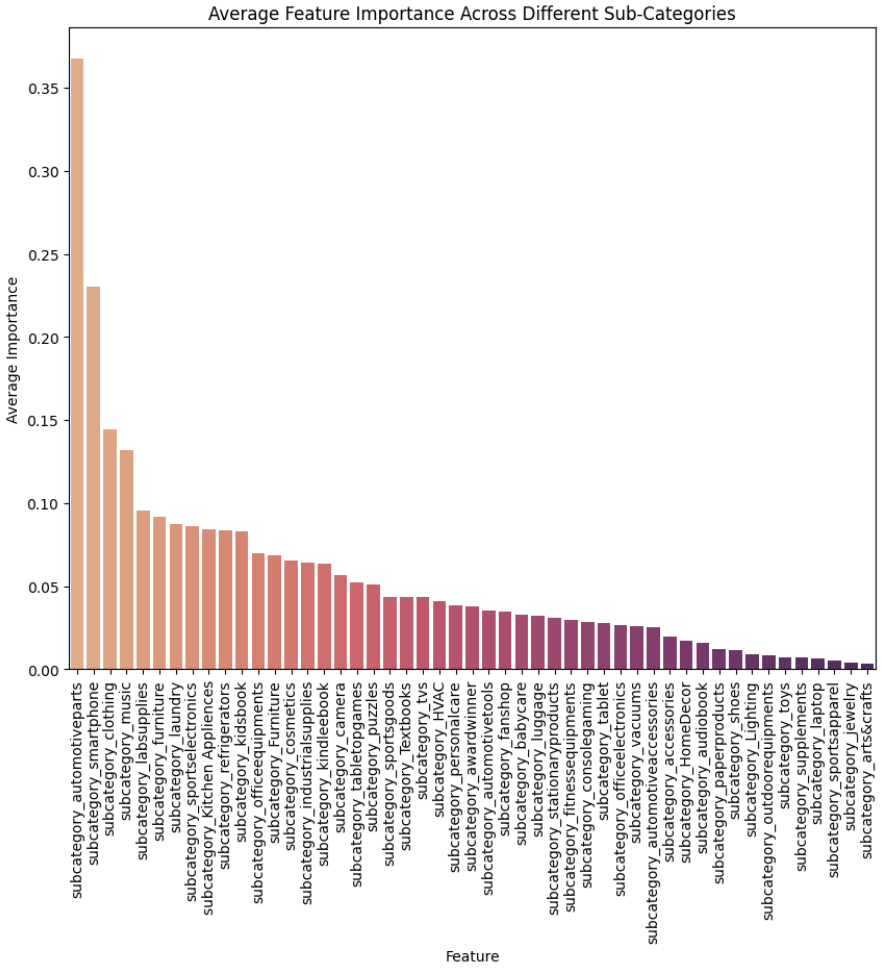


Figure 4.4.: Bar-plot of Feature Importance Score for 'Subcategory'

4.3.3. Feature Analysis of Feature Interactions

In this section, we analyze the impact of feature interactions. We gain insights into how combinations of features influenced the classification of reviews. By creating interaction terms, we modeled complex relationships between features. Below, we discuss the different interaction impact using various feature pairwise.

'Ratings * Review_length':

In this interaction, we multiply ratings and review_length feature to analyse the impact of both features on the review classification.

Table 4.10 shows the coefficient for individual features as well as feature interactions. The coefficient for interaction represents the interaction between the rating and the review length. A negative coefficient suggests that as the combined effect of rating and review length increases,

the likelihood of the review being real decreases slightly. Also, both higher ratings and longer review lengths are individually associated with a slight increase in the probability of the review being fake. The p-values of these interactions are less than 0.05 and confidence intervals for these coefficients do not include zero, reinforcing the significance of these findings.

Feature	Coefficient	Std. Error	z	P> |z|	Conf[0.025]	Conf[0.975]
rating*review_length	-0.0131	0.0056	-2.3291	0.0199	-0.0242	-0.0021
ratings	-0.0139	0.0070	-1.9803	0.0477	-0.0277	-0.0001
review_length	-0.0373	0.0070	-5.3156	0.0000	-0.0511	-0.0236

Table 4.10.: Regression Table of 'rating*review_length' feature interaction

Fig.4.11 shows that review length has most importance in predicting the outcome which almost similar to interaction term review_length * rating. while only rating feature provide very less impact on the outcome. this rating feature makes the review length feature when it combined as a whole interaction term.

Feature	LR-Imp	RF-Imp	ET-Imp	XGB-Imp	Avg-Imp
rating*review_length	0.0132	0.4966	0.4987	0.3691	0.3444
ratings	0.0373	0.4962	0.4958	0.3372	0.3416
review_length	0.0138	0.0072	0.0054	0.2936	0.0800

Table 4.11.: Feature Importance Score for 'ratings * review_length' interaction

Additionally, we can visualize the same in fig.4.5.

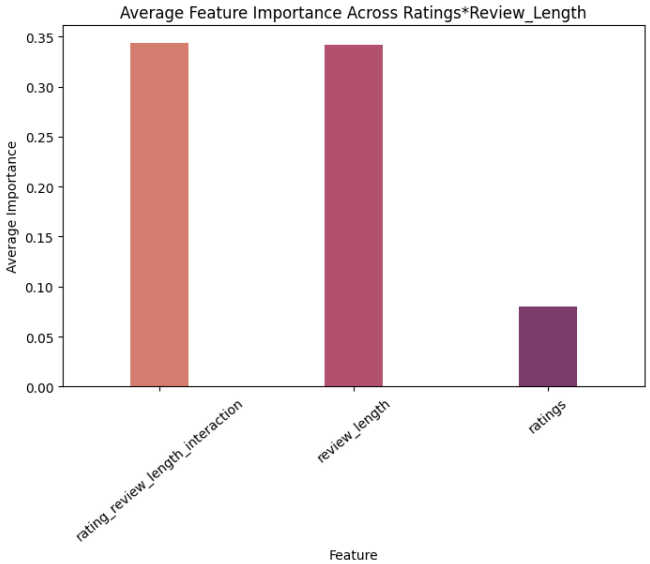


Figure 4.5.: Bar-plot of Feature Importance Score for 'ratings * review_length' interaction

'Ratings * Category':

In this interaction, we multiply ratings and category feature to analyse the impact of both features on the review being fake or real. (Note: Here, we are not including original features to avoid multicollinearity and complexity for ease of understanding)

Feature	Coefficient	Std. Error	z	P> |z|	Conf[0.025]	Conf[0.975]
rating*cat-Automotive&Industrial	0.0590	0.0136	4.3542	0.0000	0.0325	0.0856
rating*cat-Sports&Outdoor	0.0029	0.0133	0.2150	0.8298	-0.0232	0.0289
rating*cat-Health&Personal	0.0024	0.0135	0.1781	0.8586	-0.0240	0.0288
rating*cat-Books	-0.0078	0.0134	-0.5869	0.5573	-0.0340	0.0183
rating*cat-Toys&Games	-0.0144	0.0134	-1.0769	0.2815	-0.0406	0.0118
rating*cat-OfficeProducts	-0.0512	0.0132	-3.8729	0.0001	-0.0772	-0.0253
rating*cat-Appliances	-0.0528	0.0132	-3.9957	0.0001	-0.0787	-0.0269
rating*cat-Home&Kitchen	-0.0579	0.0133	-4.3583	0.0000	-0.0840	-0.0319
rating*cat-Fashion	-0.0593	0.0133	-4.4703	0.0000	-0.0853	-0.0333
rating*cat-Electronics	-0.0887	0.0132	-6.7041	0.0000	-0.1146	-0.0627

Table 4.12.: Regression Table of 'rating*category' feature interaction

Table 4.12 shows some interesting insights after the interaction of category and ratings. When Higher ratings are present in the Automotive&Industrial category significantly increases the likelihood of the review being real due to the positiveness of the interaction term. However, overall reviews in the Electronics category are more likely to be fake as this individual category feature has the highest negative coefficient. Additionally, the reviews of Health&Personal category with high ratings is more likely to be real compare to high ratings in books category. However, this trend is the opposite when we see these categories in individual category-focused analysis as shown in table 4.6.

Feature	LR-Imp	RF-Imp	ET-Imp	XGB-Imp	Avg-Imp
rating*cat-Automotive&Industrial	0.0592	0.2628	0.2816	0.2122	0.2040
rating*cat-Electronics	0.0890	0.1613	0.1484	0.1872	0.1465
rating*cat-Books	0.0082	0.1098	0.1180	0.1044	0.0851
rating*cat-Health&Personal	0.0021	0.0826	0.0908	0.1249	0.0751
rating*cat-Home&Kitchen	0.0584	0.0795	0.0726	0.0774	0.0720
rating*cat-Sports&Outdoor	0.0026	0.0731	0.0712	0.1280	0.0687
rating*cat-Toys&Games	0.0148	0.0763	0.0857	0.0549	0.0579
rating*cat-Fashion	0.0598	0.0654	0.0565	0.0258	0.0519
rating*cat-Appliances	0.0532	0.0482	0.0416	0.0515	0.0487
rating*cat-OfficeProducts	0.0517	0.0410	0.0334	0.0336	0.0399

Table 4.13.: Feature Importance Score for 'ratings * category' interaction

In these interactions, we can see that some categories become very influential after interacting with ratings as shown in table 4.13, while some become less. For example, Books became more impacting when it interacted with the ratings. On other side, Home&Kitchen gets low-

ered importance when it is combined with ratings feature. Fig.4.6 shows the overall visualization of these interactions.

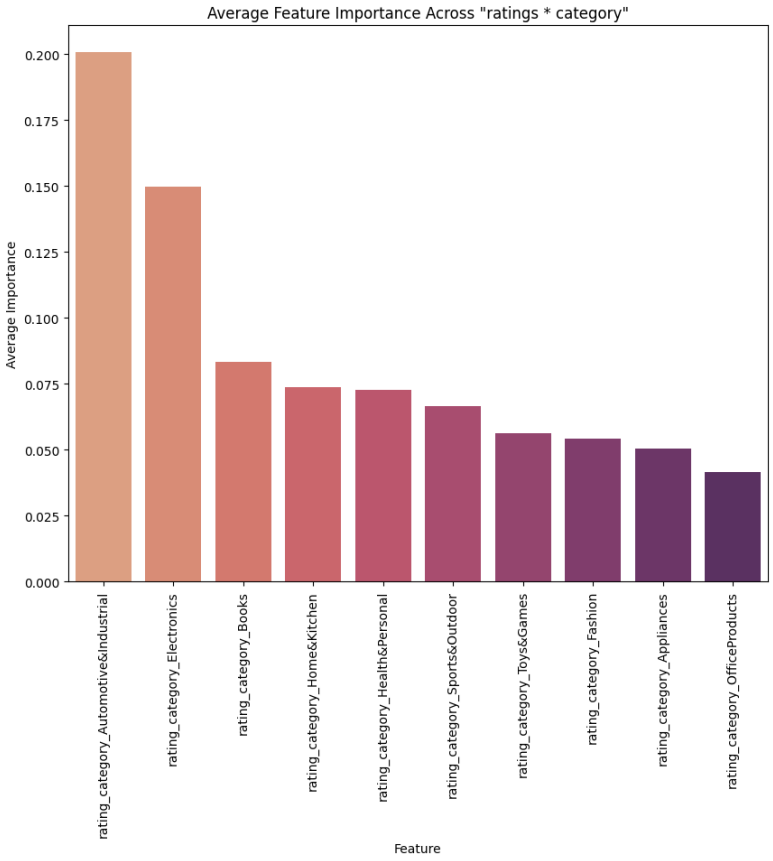


Figure 4.6.: Bar-plot of Feature Importance Score for 'ratings * category' interaction

'Review_length * Category':

In this interaction, we multiply the review length and category feature to analyse the impact of both features on the review classification.

Feature	Coefficient	Std. Error	z	P> |z|	Conf[0.025]	Conf[0.975]
review-length*cat-Toys&Games	0.0507	0.0097	5.2147	0.0000	0.0316	0.0697
review-length*cat-Automotive&Industrial	0.0453	0.0076	5.9791	0.0000	0.0304	0.0601
review-length*cat-Sports&Outdoor	0.0153	0.0070	2.1930	0.0283	0.0016	0.0290
review-length*cat-Health&Personal	-0.0009	0.0067	-0.1379	0.8903	-0.0141	0.0123
review-length*cat-Appliances	-0.0058	0.0070	-0.8229	0.4106	-0.0195	0.0080
review-length*cat-Books	-0.0132	0.0065	-2.0341	0.0419	-0.0259	-0.0005
review-length*cat-Home&Kitchen	-0.0177	0.0069	-2.5613	0.0104	-0.0313	-0.0042
review-length*cat-Fashion	-0.0295	0.0069	-4.2507	0.0000	-0.0430	-0.0159
review-length*cat-OfficeProducts	-0.0304	0.0066	-4.5825	0.0000	-0.0434	-0.0174
review-length*cat-Electronics	-0.0496	0.0073	-6.8290	0.0000	-0.0638	-0.0354

Table 4.14.: Regression Table of 'review_length*category' feature interaction

As shown in table 4.14, longer reviews in the Toys&Games category increase the likelihood of the review being real (0.0507) with high significance (p=0.00 $<$ 0.05). However, Toys&Games without review length does not much increase the probability of the review being real as shown in table 4.6. On the other hand, the OfficeProducts category becomes more likely to be involved in fake review-making after interacting with review length. Automotive&Industrial maintains a positive association both individually and when interacting with review length, though the effect size is smaller with the interaction. The categories like Books, Home&Kitchen, Fashion, Office Products, and Electronics show a shift from positive or neutral associations individually to negative associations when interacting with review length. This suggests that longer reviews in these categories are associated with a decrease in the response variable.

Feature	LR-Imp	RF-Imp	ET-Imp	XGB-Imp	Avg-Imp
review-length*cat-Electronics	0.0497	0.1184	0.1180	0.1064	0.0981
review-length*cat-Automotive&Industrial	0.0453	0.0909	0.0911	0.1212	0.0871
review-length*cat-Books	0.0131	0.1110	0.1119	0.1010	0.0843
review-length*cat-Appliances	0.0058	0.1146	0.1143	0.0953	0.0825
review-length*cat-Toys&Games	0.0507	0.0885	0.0884	0.0952	0.0807
review-length*cat-Home&Kitchen	0.0177	0.0961	0.0958	0.1101	0.0799
review-length*cat-OfficeProducts	0.0302	0.0997	0.0998	0.0872	0.0793
review-length*cat-Fashion	0.0295	0.0865	0.0864	0.0968	0.0748
review-length*cat-Sports&Outdoor	0.0154	0.0939	0.0938	0.0957	0.0747
review-length*cat-Health&Personal	0.0010	0.1005	0.1006	0.0909	0.0732

Table 4.15.: Feature Importance Score for 'review_length * category' interaction

In this interaction, we can analyze the high impact of review length on all category features as shown in table 4.15. Review length tops the importance score. Additionally, the review length feature increases the impact of all category features by combining with them except Automotive&Industrial. The visualization chart can be seen in fig.4.7.

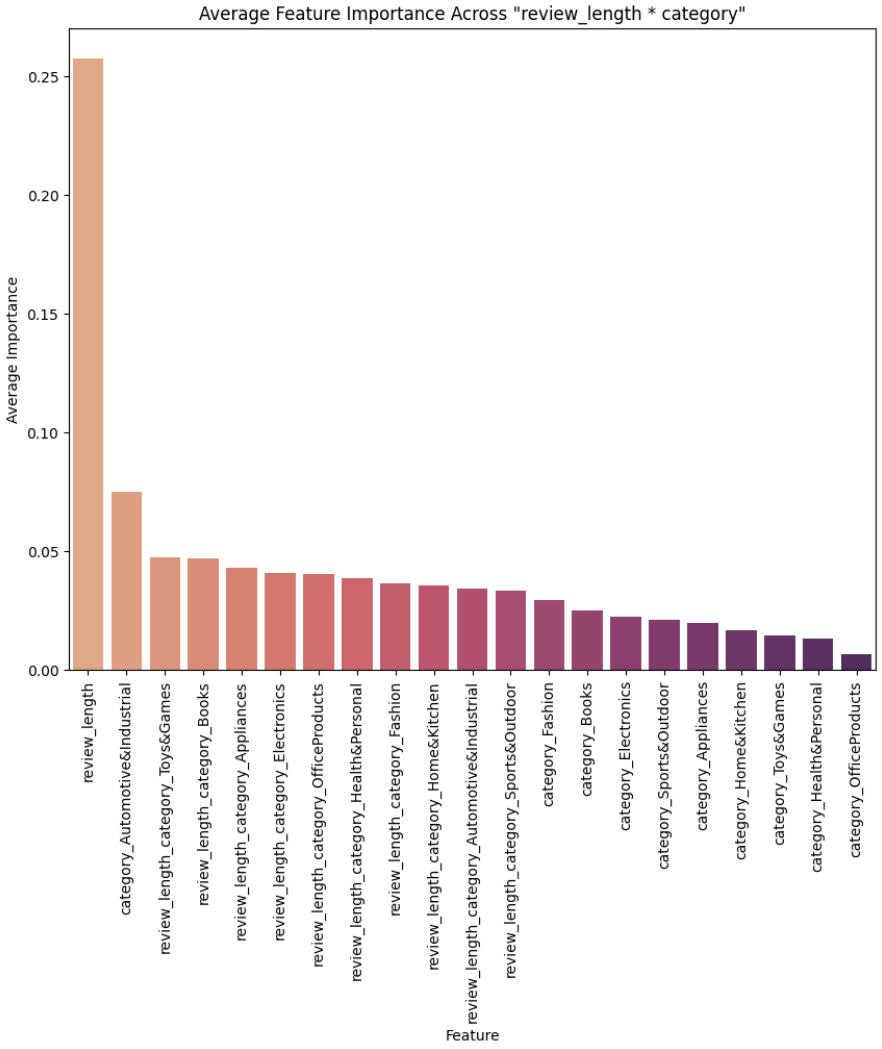


Figure 4.7.: Bar-plot of Feature Importance Score for 'review_length * category' interaction

Interactions with Subcategory feature

Similarly, the "subcategory" feature can be interacted with review length and ratings.

The interaction of ratings with the subcategory feature gives insights into how these subcategories influence classification review with ratings. The reviews in automotive parts with higher ratings are most likely to be real(coeff=0.1045) and with high statistical significance($p=0.0000 < 0.001$). On the other hand, smartphone subcategory with higher rating reviews are most likely to have a negative impact on the classification making it more fake. Subcategories like automotive parts and lab supplies show strong positive impacts alone and are even stronger

when ratings are considered, while subcategories such as smartphones, clothing, and music have strong negative impacts, exacerbated when ratings are factored in. For more details, the table A.1 in the appendix section can be referred to.

Similarly, we can find some results when the subcategory feature interacts with the review length. Some subcategories such as Automotive Parts (0.1735) and Sports Electronics (0.0846) show positive interactions with review length, indicating that longer reviews in these subcategories are associated with real reviews. On the other hand, smartphone(-0.0557) and clothing (-0.0367) subcategory make classification more towards fake reviews. Subcategories such as automotiveparts and sportselectronics show that longer reviews enhance their already positive impact, making these products appear even more realistic with more detailed reviews. Categories like smartphone, clothing, and music show that longer reviews tend to soften their negative impacts. This indicates that detailed reviews might provide more balanced views or additional context that moderates initial negative perceptions. For more details, the table A.2 in the appendix section can be referred to.

4.3.4. Feature Analysis of All Independent Features Together

In this section, we analyze the influence of features, when all independent attributes are present in the model. Here, we are taking category, review length, and ratings as a feature. Subcategory is not included because of its dependency and correlation with category feature.

As shown in table 4.16, Automotive&Industrial stays highest in significantly increasing the likelihood of a review being genuine. Similarly, the Electronics category becomes most likely to be prone to fake reviews. Review length and rating both have negative effects, suggesting that longer reviews or lower ratings might be associated with being more likely classified as fake.

Feature	Coefficient	Std. Error	z	P$>\|z\|$	Conf[0.025]	Conf[0.975]
cat-Automotive&Industrial	0.2978	0.0229	12.9964	0.0000	0.2529	0.3428
cat-Sports&Outdoor	0.1158	0.0220	5.2540	0.0000	0.0726	0.1590
cat-Books	0.0997	0.0221	4.5150	0.0000	0.0564	0.1429
cat-Health&Personal	0.0734	0.0218	3.3660	0.0008	0.0306	0.1161
cat-Toys&Games	0.0390	0.0217	1.7993	0.0720	-0.0035	0.0814
review$_{l}ength$	-0.0214	0.0071	-3.0317	0.0024	-0.0352	-0.0076
rating	-0.0221	0.0070	-3.1654	0.0015	-0.0358	-0.0084
cat-Appliances	-0.0669	0.0213	-3.1362	0.0017	-0.1087	-0.0251
cat-OfficeProducts	-0.0715	0.0212	-3.3796	0.0007	-0.1130	-0.0300
cat-Home&Kitchen	-0.0946	0.0212	-4.4731	0.0000	-0.1361	-0.0532
cat-Fashion	-0.1076	0.0212	-5.0843	0.0000	-0.1491	-0.0661
cat-Electronics	-0.2105	0.0210	-10.0371	0.0000	-0.2516	-0.1694

Table 4.16.: Regression table for all independent features combined

The table 4.17, shows the importance score for all features. Here, we can see that review length is most influential in making predictions, while rating has the least impact.

Feature	LR-Imp	RF-Imp	ET-Imp	XGB-Imp	Avg-Imp
review-length	0.0214	0.9722	0.9772	0.0728	0.5109
cat-Automotive&Industrial	0.2909	0.0025	0.0026	0.1194	0.1039
cat-Electronics	0.2172	0.0020	0.0018	0.0997	0.0802
cat-Fashion	0.1144	0.0006	0.0004	0.0877	0.0507
cat-Home&Kitchen	0.1014	0.0007	0.0006	0.0824	0.0463
cat-Sports&Outdoor	0.1090	0.0009	0.0008	0.0701	0.0452
cat-Books	0.0929	0.0012	0.0009	0.0821	0.0443
cat-OfficeProducts	0.0783	0.0008	0.0006	0.0804	0.0400
cat-Health&Personal	0.0666	0.0013	0.0007	0.0811	0.0374
cat-Appliances	0.0736	0.0011	0.0008	0.0742	0.0374
cat-Toys&Games	0.0322	0.0007	0.0004	0.0857	0.0298
rating	0.0221	0.0159	0.0132	0.0642	0.0289

Table 4.17.: Feature Importance Score for all independent features combined

The same can be visualized in the plot shown in fig.4.8.

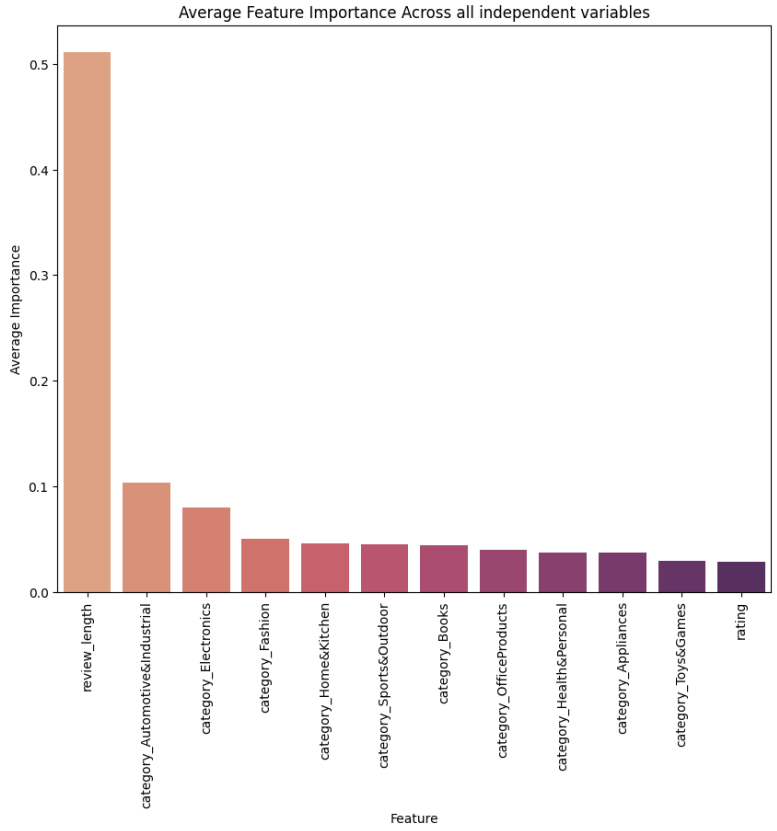


Figure 4.8.: Bar-plot of Feature Importance Score for all independent features combined

5. Challenges Faced

During the whole research process, we faced various technical challenges. These challenges spanned several stages of the project, including data collection, preprocessing, RoBERTa implementation and feature importance analysis. These problems countered systematically to insure the smooth execution of the research. This chapter outlines all primary challenges faced and the solutions employed to address them.

- **Data Acquisition**: During data collection, we faced issues with Amazon's anti-scraping policies. To bypass them, we used local rendering service. Additionally, it was hard to find different products manually and make the non-biased and equally distributed database covering all categories. To speed up this process, we extracted product codes from the URLs and used them to direct into code. The code was designed with fix Amazon review URL, where we just modified product name and page number using loops.
- **Data Preprocessing**: The acquired data were full of emojis, non-English text, and other noise elements. It was very crucial to normalize the text. To overcome this, we used various text processing techniques.
- **RoBERTa fine-tuning**: During training the model, we faced several issues in epoch processing. Due to high computational resource requirement of this transformer model, it was taking very long to fine-tune it. To resolve this, we used CUDA to use internal graphic power the system which helped into boost the process significantly.
- **Class Imbalance in Feature Importance Analysis**: After carrying out the review classification, we got to know that the fake review class was very low. This low number of fake reviews can bias the feature importance analysis. To avoid this, we employed SMOTE technique to balance the classes by generating synthetic examples of the minority class.
- **Model Selection for Feature Importance Analysis**: It was very difficult to decide which model is suitable for this type of data set, which can help into understanding the classification pattern using feature importance. To solve this, we applied AIC BIC metrics to understand the performance of this models on the dataset and suit best out of it. Additionally, we used four different models for robustness in feature importance score.

6. Conclusion

The objective of this research is to study the Consumer product reviews on Amazon platform. This includes several tasks such as the implementation of AI model to distinguish reviews into genuine and fake categories, understating the classification into various product categories, and gain insights into underlying classification patterns in these reviews. Here, we summarize our research with important key findings and contributions.

6.1. Key Findings

- The web scraping method involving BeutifulSoup algorithm for gathering the data is very effective and time saving. But, to avoid the anti-scraping policies and dynamic page content, it's always better to use local rendering service which can be used to load the pages and helps into parse the dynamic pages one by one with extraction of particular information.
- The transformer-based RoBERTa is a very good option when it comes execute the NLP tasks like text classification and Sentiment Analysis. The pre-training of this model on large corpus of data make it high efficient in capturing nuanced textual patterns and contextual information from the review descriptions. Additionally, the fine-tuned model achieved very good accuracy, showcasing its high capability in distinguishing between real and fake reviews based on the textual content.
- Regression table gives insights about how the feature impacts the outcome and in which direction. It helps to understand the feature patterns in the classification of reviews. Some key findings like Automotive&Industrial has more contribution in the real reviews and the Electronics category being the main contributor in fake reviews. These insights help in deciding which feature contributes to which type of review: Real or fake. It shows the direction of the feature.
- Feature Important Analysis reveals the depth of classification patterns using various machine-learning models. Some key insights like Automotive&Industrial and Electronics categories being the most influential features in classification of the reviews, some features like Toys&Games and OfficeProducts having almost non impact on the review being fake or real, helps into gain the important information.
- This study explores the impact of various feature interactions on the classification results. The multiplication of different numerical and categorical features shows how this interactions impact the review classification. For example, categories shows increased importance after interacting with review length feature. Similarly, we can analyse that the less effective features makes other features less effective after the interactions.

6.2. Limitations

- The research was conducted on data specific to Amazon reviews. Therefore, the findings may not be directly applicable to reviews from other e-commerce platforms with different user behaviors and review formats.
- The study highlighted the impact of feature interactions on classification results. However, the complexity of these interactions means that some relationships may not have been fully captured or understood, necessitating further investigation.
- The characteristics of genuine and fake reviews can evolve over time as fraudsters develop new techniques. This research may need periodic updates to maintain its relevance and accuracy in detecting fake reviews.
- The classification and its patterns were studied based on some limited features such as review length, ratings, category, and description.

6.3. Contributions

- This study showcase the how transformer-based model like RoBERTa can be employed easily and effective in NLP tasks such a text classification, question answering, and sentiment analysis.
- The successful execution of the feature importance analysis in this work demonstrate how these methods into understanding the underlying patterns within the models which can be beneficial in important areas like feature engineering and model optimization.
- The findings from this thesis work provide practical applicable insights for businesses, online-buying customers, and researchers. By showcasing influential key features in the authenticity of product reviews, this study provides important assistance in developing tools and methods to detect and counter fraudulent reviews.

In conclusion, this research work displays the effectiveness of advanced machine-learning techniques and comprehensive analysis, to counter the challenge of fake review detection. The research underscores the importance of integrating multiple approaches to achieve a robust understanding of review authenticity and feature relevance. As the landscape of online reviews continues to evolve, this study contribute into helping customers into improving purchase decision, and businesses into finding phony products ratings for personal sales influence.

7. Future Scope

There are several scope for improving the current study work which can lead to more better understanding of review classification and helps into more effective measures for fake review detection.

- The large dataset can be obtained from different e-commerce website such as eBay, Alibaba, and Walmart to understand the customer review behaviour with diversified datasets. This can lead to gain insights about which platform are more prune to the fake reviews and help into development of more generalised models
- This Amazon customer review data collection can be extended by collecting data from different Amazon websites for other countries. Additionally, this future study can focus on different languages as well by employing models having different language processing features.
- To make this review classification more robust and reliable, other advanced deep-learning pre-trained models like XLNet, ERNIE, and T5 can be used. Additionally, using advanced feature engineering methods, more precised optimization techniques can be made in these model for improved performance.
- In our study, we have used only textual review information for classification task. Other multi modal data like Images and videos can be incorporated to enhance the classification model.
- Temporal features like review posting date, time, and season can provide more insights into authenticity of reviews. Additionally, leveraging the user behavior patterns such as review posting history, frequency of reviews, and engagement with platform can be very beneficial in understanding the patterns in fake reviews.
- For better understanding of underlying patterns within review classification models, other advanced methods such as SHAP, LIME, and PDP techniqueus can be used.

Bibliography

[1] C. Cao, "The impact of fake reviews of online goods on consumers," *BCP Business Management*, vol. 39, pp. 420–425, 02 2023.

[2] BrightLocal. (2023) Local consumer review survey. [Online]. Available: https://www.brightlocal.com/research/local-consumer-review-survey/#

[3] M. S. R. Centre. (2017) How online reviews influence sales. [Online]. Available: https://spiegel.medill.northwestern.edu/how-online-reviews-influence-sales/

[4] Intelligencer. (2022) Amazon's war on fake reviews. [Online]. Available: https://nymag.com/intelligencer/2022/07/amazon-fake-reviews-can-they-be-stopped.html

[5] R. Mohawesh, H. B. Salameh, Y. Jararweh, M. Alkhalaileh, and S. Maqsood, "Fake review detection using transformer-based enhanced lstm and roberta," *International Journal of Cognitive Computing in Engineering*, vol. 5, pp. 250–258, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666307424000196

[6] B. H. Ashvin Gandhi and Z. Li, "Smisinformation and mistrust: The equilibrium effects of fake reviews on amazon.com," 2024. [Online]. Available: https://brettthollenbeck.com/wp-content/uploads/2024/06/gandhi_hollenbeck_li_fakereviews.pdf

[7] F. Abri, L. F. Gutierrez, A. S. Namin, K. S. Jones, and D. R. W. Sears, "Fake reviews detection through analysis of linguistic features," 2020. [Online]. Available: https://arxiv.org/abs/2010.04260

[8] J. A. Chevalier and D. Mayzlin, "The effect of word of mouth on sales: Online book reviews," *Journal of Marketing Research*, vol. 43, no. 3, pp. 345–354, 2006. [Online]. Available: https://doi.org/10.1509/jmkr.43.3.345

[9] C. Forman, A. Ghose, and B. Wiesenfeld, "Examining the relationship between reviews and sales: The role of reviewer identity disclosure in electronic markets," *Information Systems Research*, vol. 19, pp. 291–313, 09 2008.

[10] D. Khurana, A. Koli, K. Khatter, and S. Singh, "Natural language processing: state of the art, current trends and challenges," *Multimedia Tools and Applications*, vol. 82, no. 3, p. 3713–3744, Jul. 2022. [Online]. Available: http://dx.doi.org/10.1007/s11042-022-13428-4

[11] D. Jurafsky and J. H. Martin, *Speech and language processing*, 2nd ed., ser. Prentice Hall series in artificial intelligence. London [u.a.]: Prentice Hall, Pearson Education International, 2009. [Online]. Available: http://aleph.bib.uni-mannheim.de/F/?func=find-b&request=285413791&find_code=020&adjacent=N&local_base=MAN01PUBLIC&x=0&y=0

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423

[14] D. Stream. (2024) Roberta vs. bert: Exploring the evolution of transformer models. [Online]. Available: https://dsstream.com/roberta-vs-bert-exploring-the-evolution-of-transformer-models/

[15] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," 2019. [Online]. Available: https://arxiv.org/abs/1907.11692

[16] GeekForGeeks. (2023) Overview of roberta model. [Online]. Available: https://www.geeksforgeeks.org/overview-of-roberta-model/

[17] Z. Huang, C. Low, M. Teng, H. Zhang, D. Ho, M. Krass, and M. Grabmair, "Context-aware legal citation recommendation using deep learning," 06 2021.

[18] Comet. (2023) Roberta: A modified bert model for nlp. [Online]. Available: https://www.comet.com/site/blog/roberta-a-modified-bert-model-for-nlp/

[19] S. Azizah, H. Cahyono, S. Sihwi, and W. Widiarto, "Performance analysis of transformer based models (bert, albert and roberta) in fake news detection," 08 2023.

[20] M. Jordan and T. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science (New York, N.Y.)*, vol. 349, pp. 255–60, 07 2015.

[21] N. M. Seel, *Encyclopedia of the Sciences of Learning*. Springer New York, NY, 2014. [Online]. Available: https://link.springer.com/referencework/10.1007/978-1-4419-1428-6#bibliographic-information

[22] V. Nasteski, "An overview of the supervised machine learning methods," *HORIZONS.B*, vol. 4, pp. 51–62, 12 2017.

[23] M. Usama, J. Qadir, A. Raza, H. Arif, K.-l. A. Yau, Y. Elkhatib, A. Hussain, and A. Al-Fuqaha, "Unsupervised machine learning for networking: Techniques, applications and research challenges," *IEEE Access*, vol. 7, pp. 65 579–65 615, 2019.

[24] Wikipedia. (2024) Pattern recognition. [Online]. Available: https://en.wikipedia.org/wiki/Pattern_recognition

[25] GeeksforGeeks. (2023) Pattern recognition — basics and design principles. [Online]. Available: https://www.geeksforgeeks.org/pattern-recognition-basics-and-design-principles/

[26] L. Breiman, *Random Forests*. USA: Kluwer Academic Publishers, oct 2001, vol. 45, no. 1. [Online]. Available: https://doi.org/10.1023/A:1010933404324

[27] aporia. (2022) Feature importance: 7 methods and a quick tutorial. [Online]. Available: https://www.aporia.com/learn/feature-importance/feature-importance-7-methods-and-a-quick-tutorial/

[28] C. Molnar, *Interpretable Machine Learning*, 2nd ed., 2022. [Online]. Available: https://christophm.github.io/interpretable-ml-book

[29] S. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," 2017. [Online]. Available: https://arxiv.org/abs/1705.07874

[30] GeeksForGeeks. (2023) Xgboost. [Online]. Available: https://www.geeksforgeeks.org/xgboost/

[31] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 785–794. [Online]. Available: https://doi.org/10.1145/2939672.2939785

[32] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, vol. 63, pp. 3–42, 04 2006.

[33] Y. Lou, Y. Ye, Y. Yang, W. Zuo, G. Wang, M. Strong, S. Upadhyaya, and C. Payne, "Individualized empirical baselines for evaluating the energy performance of existing buildings," *Science and Technology for the Built Environment*, vol. 29, pp. 1–15, 10 2022.

A. Appendix

1. **Regression table for 'rating*subcategory' interaction**

Table A.1.: Feature Importance Score for 'rating*subcategory' interaction

Feature	Coeff	Std. Err.	z	P> |z|	Conf[0.025]	Conf[0.975]
rating*sub-automotiveparts	0.1045	0.0101	10.3232	0.0000	0.0847	0.1243
rating*sub-labsupplies	0.0248	0.0091	2.7383	0.0062	0.0070	0.0425
rating*sub-kidsbook	0.0201	0.0090	2.2263	0.0260	0.0024	0.0378
rating*sub-cosmetics	0.0163	0.0090	1.8135	0.0698	-0.0013	0.0339
rating*sub-sportselectronics	0.0161	0.0089	1.8060	0.0709	-0.0014	0.0336
rating*sub-industrialsupplies	0.0081	0.0089	0.9104	0.3626	-0.0093	0.0254
rating*sub-tabletopgames	0.0077	0.0089	0.8621	0.3886	-0.0098	0.0251
rating*sub-personalcare	0.0059	0.0090	0.6583	0.5103	-0.0117	0.0235
rating*sub-sportsgoods	0.0045	0.0088	0.5146	0.6068	-0.0128	0.0219
rating*sub-babycare	0.0006	0.0088	0.0674	0.9463	-0.0167	0.0179
rating*sub-Textbooks	-0.0001	0.0088	-0.0101	0.9919	-0.0173	0.0171
rating*sub-automotivetools	-0.0003	0.0088	-0.0324	0.9742	-0.0176	0.0170
rating*sub-tvs	-0.0012	0.0087	-0.1333	0.8939	-0.0183	0.0160
rating*sub-fanshop	-0.0034	0.0087	-0.3926	0.6946	-0.0206	0.0137
rating*sub-stationaryproducts	-0.0041	0.0088	-0.4680	0.6398	-0.0213	0.0131
rating*sub-consolegaming	-0.0050	0.0087	-0.5751	0.5652	-0.0222	0.0121
rating*sub-awardwinner	-0.0053	0.0087	-0.6070	0.5439	-0.0224	0.0118
rating*sub-automotiveaccessories	-0.0064	0.0088	-0.7343	0.4628	-0.0236	0.0107
rating*sub-vacuums	-0.0066	0.0087	-0.7529	0.4515	-0.0237	0.0105
rating*sub-laptop	-0.0098	0.0087	-1.1250	0.2606	-0.0269	0.0073
rating*sub-paperproducts	-0.0101	0.0087	-1.1518	0.2494	-0.0272	0.0071
rating*sub-arts&crafts	-0.0118	0.0088	-1.3412	0.1799	-0.0289	0.0054
rating*sub-audiobook	-0.0128	0.0087	-1.4758	0.1400	-0.0298	0.0042
rating*sub-supplements	-0.0140	0.0087	-1.6054	0.1084	-0.0311	0.0031
rating*sub-toys	-0.0140	0.0087	-1.6168	0.1059	-0.0310	0.0030
rating*sub-Lighting	-0.0141	0.0087	-1.6224	0.1047	-0.0311	0.0029
rating*sub-jewelry	-0.0146	0.0087	-1.6812	0.0927	-0.0316	0.0024
rating*sub-outdoorequipments	-0.0148	0.0086	-1.7096	0.0873	-0.0317	0.0022
rating*sub-sportsapparel	-0.0191	0.0087	-2.1953	0.0281	-0.0361	-0.0020
rating*sub-shoes	-0.0239	0.0086	-2.7761	0.0055	-0.0407	-0.0070
rating*sub-accessories	-0.0257	0.0087	-2.9660	0.0030	-0.0426	-0.0087
rating*sub-HomeDecor	-0.0257	0.0086	-2.9810	0.0029	-0.0427	-0.0088
rating*sub-CleaningSupplies	-0.0265	0.0086	-3.0777	0.0021	-0.0434	-0.0096
rating*sub-fitnessequipments	-0.0275	0.0086	-3.1867	0.0014	-0.0444	-0.0106
rating*sub-tablet	-0.0298	0.0086	-3.4637	0.0005	-0.0467	-0.0129

Feature	Coeff.	Std. Err.	z	P> |z|	Conf[0.025]	Conf[0.975]
rating*sub-officeelectronics	-0.0307	0.0086	-3.5867	0.0003	-0.0475	-0.0139
rating*sub-luggage	-0.0319	0.0086	-3.7130	0.0002	-0.0488	-0.0151
rating*sub-puzzles	-0.0347	0.0086	-4.0381	0.0001	-0.0515	-0.0178
rating*sub-HVAC	-0.0363	0.0086	-4.2344	0.0000	-0.0531	-0.0195
rating*sub-camera	-0.0409	0.0085	-4.8008	0.0000	-0.0576	-0.0242
rating*sub-kindleebook	-0.0419	0.0086	-4.8765	0.0000	-0.0587	-0.0250
rating*sub-officeequipments	-0.0431	0.0085	-5.0896	0.0000	-0.0597	-0.0265
rating*sub-Furniture	-0.0446	0.0085	-5.2336	0.0000	-0.0614	-0.0279
rating*sub-laundry	-0.0484	0.0085	-5.6905	0.0000	-0.0651	-0.0317
rating*sub-Kitchen Appliences	-0.0489	0.0085	-5.7292	0.0000	-0.0656	-0.0322
rating*sub-refrigerators	-0.0514	0.0084	-6.1070	0.0000	-0.0679	-0.0349
rating*sub-furniture	-0.0550	0.0084	-6.5630	0.0000	-0.0714	-0.0386
rating*sub-music	-0.0626	0.0085	-7.4000	0.0000	-0.0792	-0.0460
rating*sub-clothing	-0.0652	0.0084	-7.7164	0.0000	-0.0817	-0.0486
rating*sub-smartphone	-0.0820	0.0085	-9.6906	0.0000	-0.0985	-0.0654

2. **Regression table for ‘review_length*subcategory‘ interaction**

Table A.2.: Feature Importance Score for ‘review_length*subcategory‘ interaction

Feature	Coeff	Std. Err.	z	P> |z|	Conf[0.025]	Conf[0.975]
review-length*sub-automotiveparts	0.1735	0.0171	10.1436	0.0000	0.1400	0.2071
review-length*sub-sportselectronics	0.0846	0.0112	7.5687	0.0000	0.0627	0.1065
review-length*sub-consolegaming	0.0821	0.0135	6.0717	0.0000	0.0556	0.1086
review-length*sub-industrialsupplies	0.0443	0.0091	4.8676	0.0000	0.0264	0.0621
review-length*sub-kidsbook	0.0427	0.0091	4.6752	0.0000	0.0248	0.0606
review-length*sub-vacuums	0.0399	0.0086	4.6199	0.0000	0.0230	0.0569
review-length*sub-tabletopgames	0.0363	0.0088	4.1259	0.0000	0.0190	0.0535
review-length*sub-tvs	0.0356	0.0087	4.1074	0.0000	0.0186	0.0526
review-length*sub-babycare	0.0345	0.0083	4.1646	0.0000	0.0183	0.0508
review-length*sub-stationaryproducts	0.0280	0.0080	3.5227	0.0004	0.0124	0.0436
review-length*sub-CleaningSupplies	0.0201	0.0081	2.4652	0.0137	0.0041	0.0360
review-length*sub-jewelry	0.0164	0.0076	2.1630	0.0305	0.0015	0.0312
review-length*sub-camera	0.0161	0.0089	1.8114	0.0701	-0.0013	0.0334
review-length*sub-automotiveaccessories	0.0128	0.0073	1.7564	0.0790	-0.0015	0.0271
review-length*sub-Textbooks	0.0114	0.0072	1.5820	0.1136	-0.0027	0.0256
review-length*sub-cosmetics	0.0109	0.0068	1.6137	0.1066	-0.0023	0.0242
review-length*sub-labsupplies	0.0096	0.0072	1.3330	0.1825	-0.0045	0.0237
review-length*sub-arts&crafts	0.0085	0.0072	1.1726	0.2410	-0.0057	0.0227
review-length*sub-personalcare	0.0082	0.0069	1.1872	0.2351	-0.0053	0.0216
review-length*sub-paperproducts	0.0060	0.0072	0.8307	0.4061	-0.0081	0.0201
review-length*sub-supplements	0.0056	0.0072	0.7730	0.4395	-0.0085	0.0196
review-length*sub-toys	0.0037	0.0073	0.5010	0.6164	-0.0107	0.0180
review-length*sub-HomeDecor	0.0027	0.0072	0.3738	0.7086	-0.0114	0.0167
review-length*sub-outdoorequipments	0.0021	0.0070	0.2976	0.7660	-0.0117	0.0159
review-length*sub-Lighting	0.0018	0.0071	0.2591	0.7955	-0.0121	0.0158

Feature	Coeff.	Std. Err.	z	P> |z|	Conf[0.025]	Conf[0.975]
review-length*sub-officeelectronics	0.0018	0.0069	0.2599	0.7949	-0.0117	0.0153
review-length*sub-sportsapparel	0.0001	0.0071	0.0111	0.9911	-0.0139	0.0140
review-length*sub-awardwinner	-0.0005	0.0067	-0.0795	0.9366	-0.0136	0.0126
review-length*sub-automotivetools	-0.0008	0.0067	-0.1194	0.9049	-0.0139	0.0123
review-length*sub-tablet	-0.0011	0.0070	-0.1538	0.8778	-0.0148	0.0127
review-length*sub-HVAC	-0.0012	0.0071	-0.1654	0.8687	-0.0150	0.0127
review-length*sub-laptop	-0.0014	0.0070	-0.1950	0.8454	-0.0150	0.0123
review-length*sub-audiobook	-0.0017	0.0062	-0.2753	0.7831	-0.0139	0.0105
review-length*sub-accessories	-0.0036	0.0072	-0.5025	0.6153	-0.0177	0.0104
review-length*sub-shoes	-0.0050	0.0069	-0.7268	0.4673	-0.0186	0.0085
review-length*sub-sportsgoods	-0.0056	0.0062	-0.8928	0.3720	-0.0178	0.0066
review-length*sub-puzzles	-0.0078	0.0070	-1.1251	0.2606	-0.0215	0.0058
review-length*sub-fanshop	-0.0110	0.0058	-1.8927	0.0584	-0.0224	0.0004
review-length*sub-luggage	-0.0146	0.0068	-2.1497	0.0316	-0.0280	-0.0013
review-length*sub-Furniture	-0.0162	0.0068	-2.3887	0.0169	-0.0294	-0.0029
review-length*sub-fitnessequipments	-0.0164	0.0068	-2.3980	0.0165	-0.0298	-0.0030
review-length*sub-Kitchen Appliences	-0.0192	0.0069	-2.7664	0.0057	-0.0327	-0.0056
review-length*sub-refrigerators	-0.0221	0.0068	-3.2640	0.0011	-0.0353	-0.0088
review-length*sub-laundry	-0.0265	0.0068	-3.9166	0.0001	-0.0398	-0.0132
review-length*sub-officeequipments	-0.0307	0.0066	-4.6805	0.0000	-0.0435	-0.0178
review-length*sub-furniture	-0.0323	0.0068	-4.7205	0.0000	-0.0457	-0.0189
review-length*sub-kindleebook	-0.0325	0.0064	-5.0830	0.0000	-0.0451	-0.0200
review-length*sub-clothing	-0.0367	0.0075	-4.9186	0.0000	-0.0513	-0.0221
review-length*sub-music	-0.0446	0.0070	-6.4047	0.0000	-0.0583	-0.0310
review-length*sub-smartphone	-0.0557	0.0075	-7.3912	0.0000	-0.0705	-0.0409