Developing an Advanced Software Requirements Classification Model Using BERT: an Empirical Evaluation Study on Newly Generated Turkish Data

Fatih Yucalar *

Posted Date: 20 September 2023

doi: 10.20944/preprints202309.1392.v1

Keywords: software requirements classification; transformer learning; deep neural networks; machine learning; functional requirements non-functional requirements.

Article

Developing an Advanced Software Requirements Classification Model Using BERT: an Empirical Evaluation Study on Newly Generated Turkish Data

Fatih Yucalar

Department of Software Engineering, Manisa Celal Bayar University, Manisa 45400, Turkey; fatih.yucalar@cbu.edu.tr

Abstract: Requirements Engineering (RE) is an important step in the whole software development lifecycle. The problem in RE is to determine the class of the software requirements as functional and non-functional. Proper and early identification of these requirements is vital for the entire development cycle. On the other hand, manual government of requirement classes is a timewasting task, and it needs intensive effort. Automatic requirement classification techniques through advanced Machine Learning (ML) strategies are started to be developed to address this problem. Basically, software requirements are generated as natural language explanations and therefore requirement classification may be handled as a particular Natural Language Processing (NLP) problem. From ML point of view, classification strategies require dataset to be used in the training/learning phase. For this goal, we generated a unique Turkish dataset having collected the requiremenst from real-world software projects with 4600 samples. Of these requirements, 3000 and 1600 were labeled as functional and non-functional, respectively. The data set has been evaluated using (i) traditional machine learning algorithms, (ii) deep learning algorithms, and (iii) transformer models respectively. As a result BERTurk was found to be the most successful algorithm to discriminate functional and non-functional requirements in terms of 95% f-score metric.

Keywords: software requirements classification; transformer learning; deep neural networks; machine learning; functional requirements non-functional requirements

1. Introduction

The advancing technology has evolved into an integral part of our daily lives, and the need for software products is rising constantly. Indeed, software is a technological outcome in itself and is the tool enabling individuals to use technology. As utilized in almost every field, it is possible to define these tools as requirement-oriented software products. The development of a software application consists of successive stages, such as planning, requirements analysis, design, coding, testing, and maintenance, in a specific lifecycle. Software requirements analysis is one of the most fundamental and critical stages of the software development lifecycle since the development process of software applications often relies on satisfying specific customer needs. Hence, it is crucial to explicitly identify the demand's nature, the requirements it addresses, and the services it will provide. Accordingly, software requirements analysis potentially refers to the stages in identifying, outlining, categorizing, and prioritizing steps of the software requirements.

In the software requirement analysis phase, requirement engineers or project analysts attempt to ascertain the demanded system needs in collaboration with the demandant. Accordingly, they generate the software requirement document in light of the identified requirements and share it with all stakeholders. Afterward, requirements engineers or business analysts scrutinize the needs specified in the requirements document in detail. Consequently, they categorize them as functional requirements (FR) and non-functional requirements (NFR) based on the intended use of the system. FR is the specifications of the software details listed directly by the stakeholders, the services provided by the system, and the necessary limits of the system. NFR, however, refers to the qualities of the system and the requirements specifying how the software system will operate tools to perform

 doi:10.20944/preprints202309.1392.v1

functional tasks. Correct classification of FR and NFR will directly impact the project's success since the software requirements classification will serve as a guideline for other stages, such as designing and coding in the software life cycle. However, as FR and NFR are natural language texts within the same requirement document, they are likely to be confused and pose a challenging task to identify manually. The lack of FR in the developed software system leads to system failure. Correspondingly, ignoring NFR results in project failure, loss of system integrity, or cost increase [1].

This study used Machine Learning (ML) approaches on a Turkish data set generated specifically for software requirements analysis, aiming to develop ML algorithms for the automatic classification of software requirements. Since the documentation of the requirements was in the text format written in natural language, natural language processing methods have been applied in addition to ML algorithms. The study also executed experimental designs using traditional machine learning algorithms, deep learning algorithms, and transformer models. Correspondingly, the results acquired from the experimental studies were subject to comparisons through performance evaluation criteria.

The contribution of this study to the current literature may be summarized as follows: (i) developing a software requirement analysis dataset in the Turkish language for the first time using real software projects from various platforms and industries, (ii) utilizing intricate language processing protocols in Turkish, which is a morphologically rich language, (iii) obtaining the most optimally model by utilizing all algorithm types in the ML literature and (iv) being the most detailed experimental study in this field in the Turkish language.

Following an introduction text in the initial section, the subsequent study section involved the subject-related topics. Section 3, on the other hand, focused on elaborating the system architecture, the creation of the data set, the experiments with the artificial intelligence approaches used, the techniques utilized in these experiments, and the criteria used to evaluate the developed models. Section 4 involved the interpretation of the results acquired from classification attempts and visualization techniques. Finally, the last section presented the research findings, explaining the overall conclusions from the study.

2. Related Work

Manual FR and NFR classification through the Software Requirements Specification (SRS) document demands intensive effort, time, and cost. Another difficulty, on the other hand, is the uncertainty surrounding the correct classification of the identified requirements. Hence, such complications reportedly led to limited research to focus on the automatic classifying software requirements. As a result, the lack of data sets for the automatic classification of software requirements appears to be another limiting factor in this predicament.

In their study, Quba et al. [2] proposed a machine learning-based strategy for automatically classifying text data in the SRS document into FR and NFR format. They performed their work on PROMISE_exp, a generic dataset, and retaining labeled requirements. They cleaned the text data in the PROMISE_exp dataset using various techniques and ran the SVM and KNN algorithms for the classification procedure. As a result, they observed that the SVM algorithm produced superior results than the KNN algorithm according to the F-measurement value in all cases.

Limaylla-Lunarejo et al. [3] reportedly indicated that most research on classifying software requirements through machine learning algorithms was in English, with other languages receiving less attention. Hence, they created a new dataset in light of the absence of Spanish datasets. They additionally investigated which combinations of text vectorization techniques with machine learning algorithms performed best for the classification of requirements on a Spanish dataset. As a result, they found that SVM with TF-IDF provided the highest F-measurement value when classifying the FR and NFR.

Halim and Siahaan [4], however, developed a model in their study that potentially identified non-atomic requirements in software requirements written in natural languages. Non-atomic requirements are those for which the system has not just one function, albeit multiple. If a system can fully identify features, requirements, and capabilities, it refers to an atomic requirement. An atomic

 doi:10.20944/preprints202309.1392.v1

requirement may be either FR or NFR. Their requirements collection was from various online sources and categorized into two separate Corpus, Corpusa and Corpusn, retaining atomic statements and non-atomic requirements, respectively. The study dataset comprised 600 requirement statements, of which 404 were from Corpusa (atomic), and 196 were from Corpusn (non-atomic) and employed Bayes Net, Random Forest, and Multilayer Perceptron machine learning algorithms for the classification process. The Bayes Net algorithm created the best model in this study, with a correct classification rate of 84.25%. The model reliability has been deemed appropriate for unbalanced data in identifying non-atomic requirements in the software requirements specification. Yet, the model reliability for balance data in determining non-atomic requirements is considered moderate. Three expert reviews and the proposed model results were used for comparison and testing the model via the Cohen Kappa reliability test. On average, the proposed model displayed the highest reliability rate (0.49) [4].

Li et al. [5] conducted a study proposing a novel deep neural network model called NFRNet to extract the NFRs from requirements documents to minimize human labor and time spent and prevent mental exhaustion. They also utilized the PROMISE, a widely used dataset in software requirement classification research, increasing the NFR categories from 11 to 32 and NFR statements from 255 to 6222 in the PROMISE dataset. The NFRNet neural network they developed consisted of two parts. One was a BERT word embedding model based on N-gram masking to learn the context representation of the requirement statement, and the other was the Bi-LSTM classification network. Finally, they used the Softmax classifier to categorize the requirement statements. They applied a novel editing method for the model training process known as multi-sample dropout to potentially reduce the number of training iterations needed, accelerate the training of deep neural networks, and maintain reduced error rates in the trained networks. Furthermore, they employed the Tenfold cross-validation technique on the SOFTWARE NFR dataset to test the proposed model's classification accuracy; accordingly, the NFRNet model indicated the highest performance, with 91% precision, 92% recall, and 91% F-score among the other models they used [5].

Navarro-Almanza et al. [6] reportedly asserted that software requirements can be classified using deep learning approaches. They accordingly proposed a model to analyze requirements documents for large software projects via natural language processing techniques. The basis of their proposed model is the Convolutional Neural Network (CNN), one of the deep learning algorithms. Consequently, they evaluated their proposed model using the PROMISE dataset, including FR and 11 different NFR-labeled requirements. As a result, they stated that software requirements can be classified using deep learning approaches.

Bisi and Keskar [7] proposed a CNN model to classify software requirements as FR and NFR. The CNN retained several hyperparameters that affect prediction performance, such as filter size, number of filters, input insertion size, and CNN architecture. Their study aimed to optimize the CNN parameters for better prediction accuracy using the Binary Particle Swarm Optimization (BPSO) approach. They also utilized PROMISE as a study dataset, comprising 538 labeled FR and NFR. Initially designing a CNN model to classify the software requirements into FR and NFR, they subsequently employed the pre-trained dataset using the bag-of-words (BOW) technique and Wikipedia for preprocessing; in other words, converting the text data into a vector of numerical data. Finally, they developed the CNN-BPSO model to optimize the CNN hyperparameters using BPSO. Considering the experimental results, the proposed CNN-BPSO approach —80% training and 20% test data— for classifying software requirements resulted in an 81% accuracy value, outperforming the CNN model, which retained a 79% accuracy value [7].

The literature review revealed several studies focused on the correct classification of NFR, which played a critical role in improving the software quality. In this context, Haque et al. [1] reportedly emphasized that accurate NFR extraction is crucial in high-quality software development. They further indicated that the presence of the FR and NFR in the same SRS document leads to confusion; thus, differentiating these requirements would require considerably more effort. They also proposed an approach for the automatic NFR classification by combining machine learning feature extraction and classification techniques. Accordingly, they performed an experimental study using seven

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2023 doi:10.20944/preprints202309.1392.v1

machine learning algorithms and four feature selection approaches. They additionally strived to identify the best pair for automatic classification based on the statistical analysis results of the experimental studies. Therefore, they stated that the Stochastic Gradient Descent Support Vector Machine (SGD SVM) classifier and TF-IDF (character level) feature extraction technique delivered the best outcomes.

3. Methodology

This section discussed the algorithms developed, the Turkish data set established for use in the context of this study, and the techniques utilized to organize the data set. The study entailed developing models utilizing deep learning, machine learning, and transformer model-based algorithms and evaluating the performance outcomes. Figure 1 displays the system architecture, consisting of three stages.

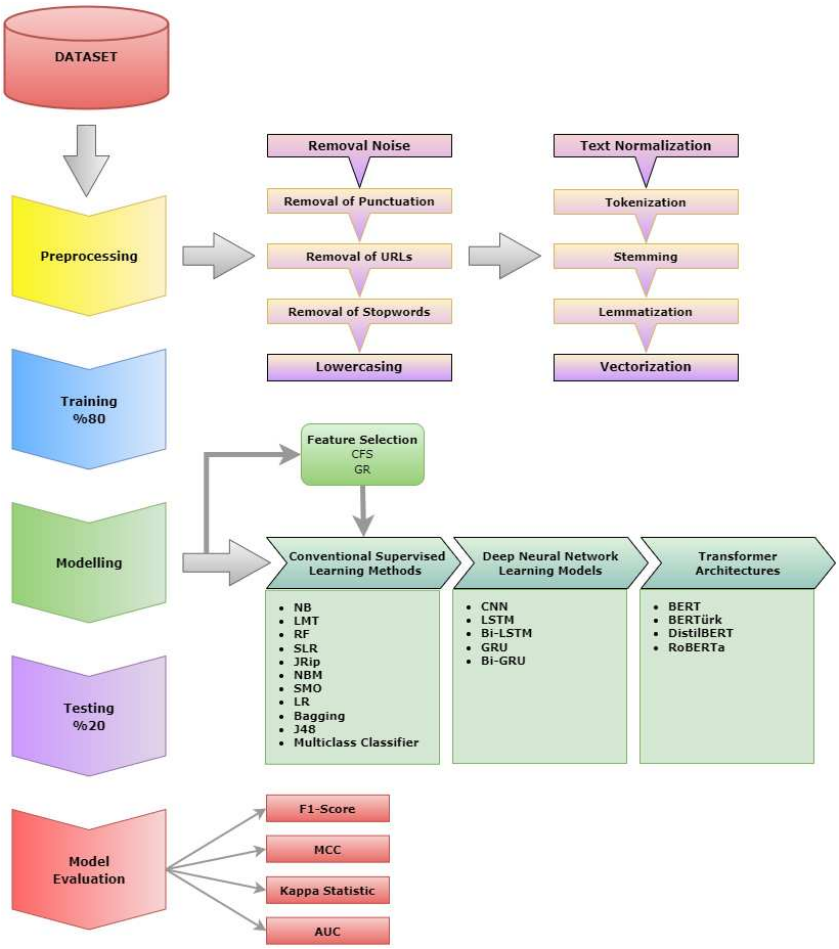

Figure 1. System architecture.

3.1. Subsection Dataset Collection and Preprocessing

Apart from the algorithm used and the model developed in artificial intelligence approaches such as machine learning and deep learning, the data set is also a significant aspect determining the performance outcome. Besides, the samples in the data set should accurately reflect the subject and be adaptive and functional in real life. Considering this information, this study identified the requirements by analyzing the real software projects developed for various platforms and sectors. Two subject-matter experts then labeled these requirements as FR and NFR. During the labeling process, the study utilized the Majority Voting technique and retained the labels for which both experts reached the same conclusion while reviewing those for which they made opposite decisions. Figure 2 displays the distribution of the labeled dataset.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2023 doi:10.20944/preprints202309.1392.v1

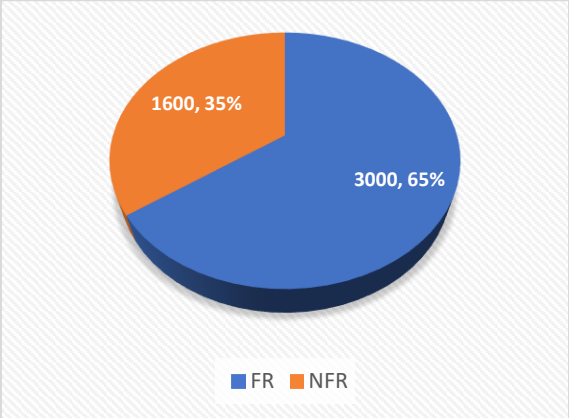

Figure 2. Distribution of the dataset requirement types.

There are overall 4600 requirements in the dataset, which includes the requirements of Windows, web, and mobile-based-like real-world projects. The data set was created in Turkish since there were few studies on the automatic classification of software requirements in Turkish. Table 1 provides samples of the data set.

Table 1. Samples sentences of dataset.

Sample sentences	English translation of sentences	Label
Sistem olayları mevcut zamandan farklılıklarına göre renklendirecektir.	The system will color events according to their difference from the current time.	FR
Kullanıcı Canlı Döviz Takip Uygulaması üzerinden bir bankanın döviz bilgilerini anlık takip edebilecektir.	The user will be able to instantly follow the currency information of a bank through the Live Currency Tracking Application.	FR
Sosyal doku analizi uygulaması üzerinde eklenen kullanıcı bilgileri saklanacaktır.	User information added on the social texture analysis application will be stored.	FR
RemMed uygulaması üzerinden kullanıcı sağlık günlüğünü doktoru ile paylaşabilecektir.	The user will be able to share his health diary with his doctor through the RemMed application.	FR
Kullanıcı e-mail ve şifresi ile sisteme giriş yapabilecektir.	The user will be able to login to the system with his e-mail and password.	FR
Seyahatname uygulaması tarafından kullanıcının yaptığı hatalara karşın doğru hata mesajları verilmelidir.	Correct error messages should be given by the Travelogue application for the mistakes made by the user.	NFR
Mobil tabanlı pazaryeri uygulaması aynı anda en az 1000 kullanıcıya hizmet verebilecektir.	The mobile-based marketplace application will be able to serve at least 1000 users at the same time.	NFR
Network alt yapısı sistem kaynaklarının her biri için ortalama en fazla %50'sini kullanmalıdır.	The network infrastructure should use at most 50% of the system resources on average.	NFR
Hesabını Bil uygulaması üzerinde yer alan ekranların yenilenme süresi en fazla 5 saniye olacaktır.	The refresh time of the screens on the Know Your Account application will be a maximum of 5 seconds.	NFR
Geliştirilecek oyun programı üzerindeki ekran kontrolleri oyuncunun oyunu oynamasına engel olmayacak büyüklükte olmalıdır.	The screen controls on the game program to be developed should be large enough to not prevent the player from playing the game.	NFR

Since the created data set is in natural language format, the study employed a data preprocessing step to convert the data into a format that algorithms would process. The initial stage in natural language processing studies is to execute the data preprocessing after creating the dataset. In this stage, natural language is converted into a machine-understandable text format to prepare algorithms for use in artificial intelligence techniques. Data preprocessing is as crucial as creating the data set since the data quality is directly related to the predictive performance of the generated algorithm to be real-like. With this objective in mind, the current study applied noise removal and text

 doi:10.20944/preprints202309.1392.v1

normalization—including removal of punctuation, special characters, stopwords, and case conversion—preprocessing techniques such as tokenization and vectorization after the dataset creation process. Data preprocessing is necessary for automatic natural language processing when addressing morphologically complex languages like Turkish. In this context, ML algorithms need additional feature engineering applications after the aforementioned preprocessing steps. As a result, the study additionally applied a feature selection method to the dataset to achieve this outcome.

3.2. Feature Selection

Artificial intelligence techniques such as data mining and natural language processing have been developed to control the data spreading and extracting accurate information from the data. In this context, the feature selection aims to create simpler and more comprehensible models to improve data mining performance and prepare clean and intelligible data [8].

A feature potentially refers to an option in each column in the dataset that characterizes the data. It is possible to classify text by the options defining that text. Considering the features that best describe the data for a successful classification process is ideal. Hence, it is essential to consider the features that best characterize the data to acquire real-like results in text classification tasks. Feature selection refers to handling the options that reflect the data more by ignoring the same features in the entire dataset or that do not retain a distinctive effect on the overall dataset. The feature selection strategies aim to increase the prediction performance and hasten the learning process by reducing the dimensionality [9].

High-quality features contributing to computation from the data feature space and improving performance are employed to create a feature subset in machine learning algorithms using feature selection techniques—often employed in data preprocessing [10]. The study also utilized Correlation-based Feature Selection (CFS) and Gain Ratio (GR) feature selection techniques.

CFS is a multivariate filter approach selecting subsets of unrelated but highly correlated features with the class [9]. A heuristic evaluation function is used for the ranking process of the feature subsets in correlation-based feature selection. While more significant features are defined as highly correlated in the training and testing process of the prediction model, the procedure ignores low-correlation features. Furthermore, the prediction model eliminates the unnecessary options [11].

Information gain is calculated for all features in the GR technique. Hence, the features performing at least as much as the average information gain and achieving the best gain ratio are selected. GR outperforms the information gain measure in terms of both accuracy and classifier complexity [12].

3.3. Conventional Machine Learning Methods

Machine learning is a broad discipline that spans information technology, statistics, probability, artificial intelligence, psychology, neurobiology, and many other disciplines. It also refers to teaching computers to think like humans while generating the field of statistics and fundamental statistical-computational theories of learning processes. Machine learning algorithms are categorized into groups, including supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning, depending on the way of seeking a solution to a problem [13]. This study used the supervised learning method. Classes are manually separated and labeled beforehand in supervised learning algorithms. Mathematical models created with machine learning algorithms are trained and tested, aiming to achieve the highest prediction performance. This study utilized the Weka library to test conventional algorithms. It also used many algorithms in this library during the experimental design and selected the best-performing algorithms. Figure 1 lists the relevant algorithms, and the following section briefly mentions these algorithms.

The Naïve Bayes (NB) classifier is a Bayes theorem-based classification technique. It is one of the most effective machine learning algorithms since it predicates a powerful assumption of independence from any condition or event, is easy to implement, and delivers performance outcomes for large datasets [14]. Additionally, it mathematically resides on Bayes' theorem and addresses the probability of whether a sample in the dataset belongs to a class independently of others. As a result,

 doi:10.20944/preprints202309.1392.v1

this method identifies the independent probability value for each class and subsequently makes the classification based on these classes with the highest probability. The NB classifier is one of the most widely used methods for text data mining.

Naïve Bayes Multinomial (NBM) is an advanced version of the current Naïve Bayes classifier and calculates the frequency of each word. It is well-established that frequency is highly effective in classifying the text into different categories. Therefore, the NMB algorithm is considered one of the best in text classification [15].

Logistic Regression Tree (LRT) is a typical decision tree structure with a merged working principle with logistic regression. Each node in the decision tree retains a unique logistic algorithm. As in decision trees, each node contains child nodes. As a result, the prediction values are calculated through logistic computations at these nodes by comparing the feature and threshold values [16].

Sequential Minimal Optimization (SMO) is a widely used classification algorithm to train support vector machines with supervised machine learning models through quadratic programming solvers. SMO is one of the several unique methods for resolving the quadratic programming problem in SVM. The SMO-oriented iterative algorithm breaks the optimization problem into several more manageable subproblems. Then, to entirely avoid numerical QP optimization, these QP-based subproblems are solved analytically [17].

Random Forest (RF) is a machine learning algorithm that uses multiple decision trees on various subsets in a dataset and averages the outcomes to improve prediction accuracy. In other words, instead of relying on a single decision tree, it gathers predictions from multiple trees to get the results. At each node of a tree, a condition is compared with one or more features of the input data. There is a class prediction for each tree, and the most predicted class is indeed the original predicted class by the algorithm. It reveals excellent performance on RF unbalanced datasets, displaying very few classification errors [18].

Logistic Regression (LR) is an advanced technique to categorize linear and nonlinear data. It is frequently employed while modeling the data with binary responses. In binary responses, they typically appear as 0/1, where '1' usually denotes a success and '0' is a failure. Yet, the actual values of 1 and 0 may display a wide range of possibilities depending on the study's objectives. Binary classification allows to label one class as '1' and the other as '0.' Logistic regression is a machine learning algorithm that takes the given input value and multiplies the input by the weight value [19].

Simple Logistic Regression (SLR) performs outstandingly with linear data, whereas it may perform poorly with nonlinear or complex data. It also fails to address datasets with missing data [18].

Bagging is an algorithm that functions with the ensemble learning method. As depicted in Figure 3, every ensemble model is trained by a data subset of the current dataset. Since the models operate independently, it is possible to train them concurrently. Combining the decisions of the classifiers yields the classification of a new test instance of the ensemble model. The aim is to deliver higher performance with more than one classifier [20].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2023 doi:10.20944/preprints202309.1392.v1

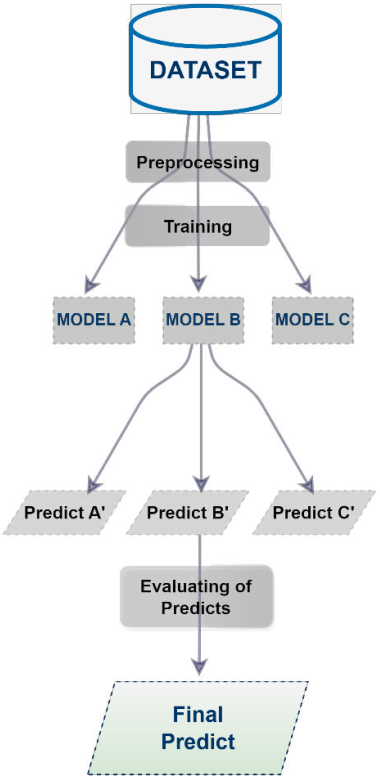

Figure 3. Ensemble learning architecture.

JRip is one of the most popularly used machine learning algorithms. Its operation principle resides in analyzing the classes as they expand and creating the first set of rules for these classes using gradually lowered error rates. It is ideal to use this algorithm to classify all samples of a given dataset in the training data and search for a set of rules that apply to all members of that dataset. It then moves on to the next class and repeats the procedure until evaluating all classes [21].

J48 is a machine learning classifier with features with missing values, rule derivation, and continuous-valued feature ranges. This algorithm develops rules to create a unique data identity. The purpose of using this classifier is to gradually branch the decision tree until it strikes a balance between versatility and accuracy [21].

A Multiclass Classifier is one of the supervised classification algorithms used when there are more than two outcomes of the predictive value in a classification procedure [22]. It functions based on the ensemble theory. Initially, several algorithms are trained by a subset of data, and then the algorithms with the highest performance are evaluated. This classifier is often regarded as a successful classification approach in the field of machine learning since its test procedure incorporates numerous algorithms, each of which undergoes evaluation for its effectiveness.

3.4. Deep Neural Network Learning Models

Deep learning is a significant and trending topic in the artificial intelligence discipline. The development of deep learning techniques resides in exemplifying the human brain, nervous system, and its functions [23]. Deep learning is an extraction process of knowledge from data utilizing artificial neural networks inspired by nerve cells and multi-layered hidden architectures. Data is transferred through several layers in a deep learning algorithm, with each layer progressively extracting features and sending data to the next layer. The initial layers extract low-level features and combine them with subsequent layers to create a comprehensive representation.

The conventional machine learning classification task entails preprocessing, feature extraction and selection, and classification/model setting stages. The correct feature selection is a primary factor in the high prediction performance of machine learning systems. As illustrated in Figure 4, on the

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2023 doi:10.20944/preprints202309.1392.v1

other hand, deep learning models concurrently perform feature extraction, feature selection, learning, and classification procedures. Such versatility of the deep learning algorithms makes it advantageous for executing numerous tasks.

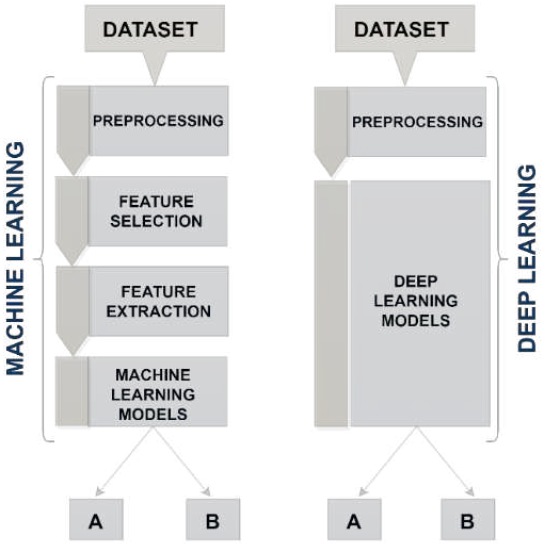

Figure 4. Differences between machine learning and deep learning.

Deep learning techniques using deep neural networks have gained popularity in parallel with the advancements in high-performance computer opportunities. Several newly developed techniques and numerous studies applying these methods to address various challenges have been increasing gradually. These techniques are still in use today and are broadly applicable to diverse branches of natural language processing. Due to neural networks' capacity to learn representations with various degrees of abstraction, deep learning has been applied to natural language processing to attain cutting-edge performance in numerous tasks, including language building [24]. Convolutional Neural Networks, Long Short-Term Memory, Bidirectional Long Short-Term Memory, Gateway Repetitive Units, and Bidirectional Gateway Repetitive Units are all deep learning algorithms.

3.4.1. Convolutional Neural Networks

Convolutional Neural Networks (CNNs), a variety of traditional feed-forward neural networks, are extensively used in image recognition and have more recently drawn interest in natural language processing. In natural language processing, CNNs use a sentence representation that keeps the word order entire. As a result, CNNs may eventually learn and recognize patterns consisting of strings of words that span more than one word in a sentence. This context makes CNN compatible with pattern recognition-related tasks. As depicted in Figure 5, A simple CNN network consists of embedding, convolution, pooling, dropout, fully connected, and output layers [25, 26].

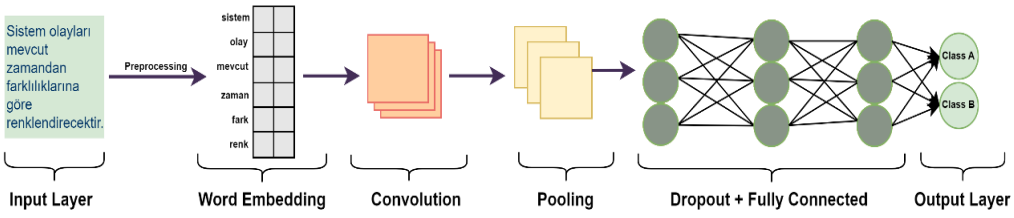

Figure 5. CNN architecture.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2023 doi:10.20944/preprints202309.1392.v1

The Input/Word Embedding layer is the system where the vector representation of the input sentence is created and converted into a 2D matrix [26]. However, the convolution layer is where two-dimensional filtering procedures are performed on the input matrix to extract the feature map. The filtering procedure is applied over each field in the input matrix, called the convolution. Each convolution serves as a neuron, computing the scalar product of its weights and the regional input, and subsequently, the activation function converts into a single feature. The features of each convolution are collected in a feature map for each filter [18]. The pooling layer follows the convolution layer and performs downsampling and dimensionality reduction on the input data, reducing the number of connections in the network. Its primary purpose is to alleviate the computational load and address overlearning problems. The pooling layer also defragments various image dimensions and enables CNNs to recognize objects even if their shapes are distorted or viewed from different angles [24]. The dropout layer is one of the typical systems functioning in neural networks to ensure the rarefication and generalization of the model. This layer also lessens the number of training iterations needed, hastens training procedures, and provides trained networks with lower error rates [5]. The fully connected layer, also referred to as the dense layer, is used for the final prediction. This layer passes the data through a series of entirely interconnected neurons. As a result, it predicts under which class the data falls [25].

3.4.2. Long Short-Term Memory

Long Short-Term Memory (LSTM) is an advanced version of recurrent neural networks. LSTM models are more effective at retaining and utilizing information in longer sequences [24]. In an LSTM architecture consisting of neural network layers, the input data from the processing layer and the output data from the preceding layer are stored in memory in an LSTM architecture, consisting of neural network layers. Thus, it enables us to make predictions based on current and previous data. The LSTM also checks the timing of impending data access to memory and its exit. In principle, input, omission, and output are the three building blocks of the LSTM architecture, and the information flow manifests through these blocks. It can also learn about long-term dependencies.

3.4.3. Bidirectional Long Short-Term Memory

Bidirectional Long Short-Term Memory (Bi-LSTM) is an extension of the LSTM architecture that addresses the drawbacks of standard LSTM models by considering both past and future context in sequential modeling tasks. While conventional LSTM models only process input data in the forward direction, the Bi-LSTM model overcomes this limitation by training the model in both directions. A standard Bi-LSTM architecture has two parallel layers that process the input string forward and backward. This bidirectional processing enables the model to capture information from past and future contexts, providing a more comprehensive understanding of temporal dependencies within the sequence [24].

3.4.4. Gated Recurrent Unit

The Gated Recurrent Unit (GRU) is a type of RNN inspired by the functionality of LSTM. The GRU has two ports—the reset and update ports—and displays the entire status every time without a control mechanism. Resetting allows for erasing unnecessary information. The update port, on the other hand, controls the quantity of the data transferred from the previous remote state. Actual activation is calculated as a linear interpolation of previous and candidate activations [27].

3.4.5. Bidirectional Gated Recurrent Unit

A Bidirectional Gated Recurrent Unit (Bi-GRU) consists of two GRUs. These GRUs have opposite directions and independent parameters. The advantage of Bi-GRU is that it can analyze the relationship between contextual sentences. In this manner, it can make the right decisions about the meaning of each sentence, potentially extracting the text features most closely related to the original one [28].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2023 doi:10.20944/preprints202309.1392.v1

3.5. Transformer Architectures

Transformer-based language models, which reside on the most advanced language models, are a unique class of artificial intelligence that analyzes natural language texts to mimic human language processing. Transformer-based models provide a more thorough interpretation of related words by considering the context of the processed words. These are pre-trained language models; in other words, they are a tested solution for a wide range of natural language processing tasks. In this context, a language model based on transformers is initially trained on many texts before being finely tuned on task-specific data [29, 30]. BERT, BERTurk, DistilBERT, and RoBERTa are transformer-based models.

3.5.1. BERT

The acronym BERT—Bidirectional Encoder Representations from Transformers—is a transformer-based language model developed by Google with the architecture illustrated in Figure 6. It consists of two stages: encoder and decoder. The encoder produces an output after sequentially processing the input in coding layers. The decoding layers subsequently process this output. BERT was trained on 16 GB of texts from BooksCorpus datasets and English Wikipedia. When BERT analyzes words in a text, it considers its morphology and context. BERT potentially scrutinizes language with "Masked Language Modeling" and "Next Sentence Prediction" mechanisms. In masked language modeling, the algorithm initially ignores (masks) a word anywhere in the input sentence; accordingly, it attempts to predict this masked word within the sentence by analyzing the pre- and post-texts of the ignored/masked word. The mechanism used to predict the next sentence follows a similar approach. Instead of any word in the sentence, the model randomly masks a sentence in the input text and subsequently analyzes the pre- and post-sentences to predict the masked sentence. Thus, this model outperforms many other language models with this feature [31]. Since the BERT is a previously trained model with big data sets, this process makes it substantially faster. Because only preliminary training and fine adjustment are sufficient since the development of the model will reside on a pre-existing model.

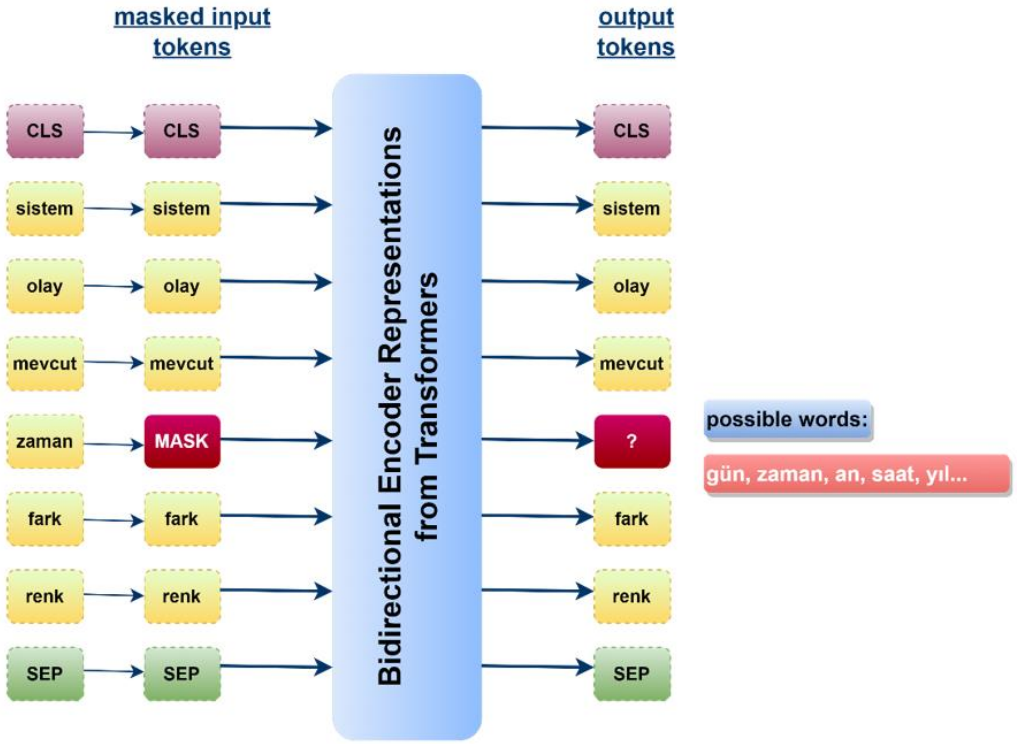

Figure 6. BERT language model architecture.

3.5.2. BERTurk

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2023 doi:10.20944/preprints202309.1392.v1

When initially using the BERT model in natural language processing, there were diversely developed variants of the BERT architecture for numerous issues or different languages. The BERTurk model trained with Turkish data to analyze the Turkish texts is also one of these variants. Since they share the same architecture as BERT, the BERTurk-originated models are also more performant and faster than other natural language processing models . Furthermore, using a ready-made model would significantly impact the performance via correctly operating the pre-processing and fine-tuning by the model requirements [32].

3.5.3. DistilBERT

DistilBERT is also a variant of the BERT architecture, much like BERTurk. While the purpose of developing BERTurk was for a different language than BERT, DistilBERT involved a model modification. It primarily uses the BERT's initial version architecture as a base, replacing heavier architectures with more parameters with a light version of the same architecture with fewer parameters. Hence, it reduces the number of layers by half in the BERT-based model, eliminating identifier embeddings and poolers to yield a significantly faster and smaller version of BERT for widespread use [33]. The model also applies dynamic masking and ignores next-sentence predictions. DistilBERT aims to generate a faster-running version of BERT [31].

3.5.4. RoBERTa

With an almost similar architecture to BERT and built on the same language masking strategy, RoBERTa (Robustly Optimized BERT Pretraining Approach) is an optimized method for pre-training a self-supervised NLP system [34]. The main difference between them is that BERT uses static masking while RoBERTa uses dynamic masking [33]. RoBERTa allows for better performance by changing the basic hyperparameters in the BERT model.

3.6. Evaluation and Statistical Validation Metrics

This study employed frequently used performance evaluation metrics—F-score and AUC—and statistical validation metrics—MCC and Kappa—to evaluate the performance results of models developed with artificial intelligence approaches.

3.6.1. Performance metrics

A set of metrics is required to compare and evaluate the results of algorithms developed for a classification problem. Confusion matrix (CM) is the primary instrument to acquire the essential metrics for the binary classification of software requirements as functional and non-functional. True-positive (TP) and True-Negative (TN) depicted in Figure 7 are regarded as correct predictions, whereas False-Negative (FN) and False-Positive (FP) are considered incorrect predictions [32].

		Predicted Values	
		FR	NFR
Actual Values	FR	TN	FP
	NFR	FN	TP

Figure 7. Confusion matrix.

Finding the precision and recall values is necessary to calculate the F-score metric. The Precision value defines the number of FR-predicted requirements for the actual FR class presented in Figure 7. In contrast, the Recall value calculates the number of FR-predicted requirements of all FR requirements in the data set. Using Precision and Recall values, the F-score [35] is calculated by the formula in Equation (3).

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2023 doi:10.20944/preprints202309.1392.v1

$$Presicion = \frac{TP}{TP + FP} \quad (1)$$

$$Recall = \frac{TP}{TP + FP + NP} \quad (2)$$

$$F - score = 2x\frac{Presicion\ x\ recall}{Presicion + Recall} \quad (3)$$

Since the definition of the F-score is the harmonic average of precision and recall values, it averts outliers from affecting performance in unbalanced datasets. Additionally, the dataset in the current study was partly unbalanced; as a result, the study identified the F-score as the evaluation metric.

3.6.2. Statistical Validation Metrics

Matthews Correlation Coefficient (MCC) and Kappa statistics-based metrics are ideal for evaluating algorithms developed for prediction problems. All values in the complexity matrix are used in the calculation to assess the performance of a model with the Matthews Correlation Coefficient metric [36]. Considering the correlation between the actual data and the predicted data, Equation (4) is employed to calculate this matrix. The minimum and maximum limits are '-1' and '+1', respectively, meaning that the performance increases and the predictions are correct as the value approaches '1.'

$$MCC = \frac{(TP\ x\ TN - FP\ x\ FN)}{\sqrt{((TP + FP)x(TP + FN)x(TN + FP)x(TN + FN))}} \quad (4)$$

Kappa statistic indicates the correspondence between the actual and predicted values. It also considers whether this correspondence is by chance. As in the MCC metric, the minimum and maximum values vary between '-1' and '+1', respectively. As a result, the model is more successful as the statistical outcome approaches "1" [37], whereas "0" indicates that the existing correspondence is by chance.

4. Experiments and Discussions

This section explores the answers to the following research questions (RQ1, RQ2, RQ3, and RQ4):

- RQ1: How successful are conventional supervised learning methods in identifying software requirements into functional and non-functional?
- RQ2: How successful are deep learning methods in identifying software requirements as functional and non-functional?
- RQ3: How successful are transfer learning models in identifying software requirements as functional and non-functional?
- RQ4: Which of the traditional supervised learning, deep learning and transfer learning methods is more successful in classifying software requirements?

4.1. Procedure Followed in Experiments

In principle, experimental studies involve three stages: machine learning algorithms, deep learning algorithms, and classification of transfer learning methods and software requirements as FR and NFR. All of the experiments utilized the dataset provided in Section 3. While performing experimental studies, the dataset was divided into two parts, 80% and 20% for training and testing, respectively.

4.2. Experimental Results

The study initially used the Weka tool for classification experiments with machine learning algorithms. It also experimented with all machine learning algorithms available on Weka and revealed the findings of the 11 most successful algorithms provided in Table 2.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2023 doi:10.20944/preprints202309.1392.v1

Table 2. F-score and AUC value of conventional algorithms.

Algorithm	F-score	AUC
NB	.830	.899
LMT	.914	.959
RF	.909	.961
SLR	.899	.956
JRip	.901	.887
NBM	**.928**	**.970**
SMO	.913	.902
LR	.888	.933
Bagging	.863	.930
J48	.826	.879
Multiclass Classifier	.888	.933

The data analysis presented in Table 2 revealed that the models developed with machine learning algorithms yielded comparable performance results. Of all the models, the model developed with the Naïve Bayes Multinomial was the highest-performing model with a 92% F-score and 97% AUC value. The model developed by the LMT algorithm followed the Naïve Bayes Multinomial algorithm with a 91% F-score and 95% AUC value. In addition, the models created by the Logistic Regression and Multiclass Classifier displayed the same performance level. Considering the algorithm with the lowest performance among the 11 algorithms analyzed, however, it was the model developed by J48 with an 82% F-score and 87% AUC value.

In addition to conventional machine learning algorithms, feature selection techniques have been applied to the same algorithms to improve performance. In this context, the study used CFS and GR feature selection methods to identify their effects on performance results and displayed the outcomes in Table 3.

Table 3. F-score and AUC value of conventional algorithms with feature selection methods.

Algorithm	F-score		AUC	
	CFS	GR	CFS	GR
NB	.787	.856	.895	.897
LMT	.842	.816	.91	.962
RF	.842	.805	.909	.962
SLR	.841	.904	.915	.957
JRip	.836	.905	.501	.501
NBM	.837	.830	.503	.503
SMO	.834	.915	.792	.903
LR	.845	.891	.917	.929
Bagging	.832	.862	.896	.929
J48	.751	.821	.786	.883
Multiclass Classifier	.845	.830	.917	.929

The performance results of the feature selection algorithms in Table 3 indicated that they failed to generate any positive effect on the performance increase. However, the F-score relatively increased when the GR feature selection technique was applied to the model developed with SMO, Simple Logistic Regression, and Naïve Bayes algorithm. Additionally, in two feature selection techniques, the performance of the model created by the Naïve Bayes Multinomial, which had the best performance in the prior trial, declined.

In the second stage, the study conducted experiments with CNN, LSTM, Bi-LSTM, GRU, and Bi-GRU deep learning algorithms and accordingly displayed the findings in Table 4.

Table 4. F-score and AUC value of deep learning algorithms.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2023 doi:10.20944/preprints202309.1392.v1

Algorithm	F-score	AUC
CNN	**.937**	**.918**
LSTM	.914	.893
Bi-LSTM	.907	.881
GRU	.926	.911
Bi-GRU	.915	.901

The assessment of the models developed with deep learning algorithms revealed that the CNN algorithm was explicitly successful and outperformed other deep learning algorithms in classifying software requirements as FR and NFR.

The study also performed experiments to classify software requirements as FR and NFR using BERT, BERTurk, DistilBERT, and RoBERTa transfer learning methods and presented the results in Table 5.

Table 5. F-score and AUC value of transfer learning methods.

Algorithm	F-score	AUC
BERT	.921	.971
BERTurk	**.954**	**.983**
DistilBERT	.918	.968
RoBERTa	.862	.952

Finally, in the third stage, analysis of the experimental study results related to transfer learning methods indicated that the model developed with the BERTurk algorithm delivered the highest performance with an 95% F-score.

4.3. Statistical Validation Results

The previous section discussed the F-score and AUC metrics results of experimental studies conducted with machine learning, deep learning, and transfer learning approaches. In addition, it statistically evaluated the experimental results and used validation metrics MCC and Kappa to analyze them as a whole. Accordingly, MCC and Kappa values were calculated for all experiments performed in all three stages. Table 6 lists the MCC and Kappa values of the experiments conducted with machine learning algorithms, while Table 7 displays the MCC and Kappa values calculated by applying future selection techniques to the same algorithms.

Table 6. Statistical validation results of machine learning algorithms.

Algorithm	MCC	Kappa
NB	.663	.627
LMT	.809	.808
RF	.797	.794
SLR	.775	.772
JRip	.779	.778
NBM	.841	.843
SMO	.806	.806
LR	.753	.752
Bagging	.694	.691
J48	.616	.615
Multiclass Classifier	.753	.752

Table 7. Statistical validation results of machine learning algorithms with feature selection methods.

Algorithm	MCC		Kappa	
	CFS	GR	CFS	GR

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2023 doi:10.20944/preprints202309.1392.v1

NB	.787	.856	.507	.626
LMT	.842	.816	.640	.816
RF	.844	.805	.634	.805
SLR	.841	.904	.637	.783
JRip	.836	.905	.627	.618
NBM	.837	.830	.629	.626
SMO	.834	.915	.621	.810
LR	.845	.891	.646	.759
Bagging	.832	.862	.691	.688
J48	.751	.821	.431	.601
Multiclass Classifier	.845	.834	.646	.759

Table 8 presents the MCC and Kappa values of experimental studies with deep learning algorithms.

Table 8. Statistical validation results of deep learning algorithms.

Algorithm	MCC	Kappa
CNN	.837	.835
LSTM	.801	.802
Bi-LSTM	.814	.817
GRU	.828	.824
Bi-GRU	.787	.793

Table 9 displays the MCC and Kappa values of experimental studies conducted by transfer learning techniques.

Table 9. Statistical validation results of transfer learning methods.

Algorithm	MCC	Kappa
BERT	.874	.873
BERTurk	.898	.897
DistilBERT	.857	.854
RoBERTa	.789	.788

An overall assessment of tables from 6 to 9 revealed that MCC and Kappa values supported the F-score and AUC values. As both metrics approach '1,' the accuracy of the predictions is supported. Considering the three groups of experiments, the average MCC and Kappa values of 0.85, particularly in Table 9, statistically confirmed the accuracy of the results. As a result, it is viable to conclude that the algorithms provided consistent results.

5. Conclusions

Just as the correct identification of business needs is crucial for successful software projects, defining and addressing these requirements is equally essential to satisfy the business needs explicitly, consistently, concisely, and summarily in the analysis documents in a way that is explicit to all stakeholders to comprehend and leave no room for argument. Furthermore, a thorough analysis and classification of these requirements is necessary to develop high-quality and reliable software. Checking whether the identified requirements retain sufficient detail, pose internal consistency with each other, and meet the business needs are also among the critical issues to consider during the requirements analysis procedure. Subsequently, these identified requirements should be classified into functional and non-functional requirements. The manual identification process of functional and non-functional requirements is a highly challenging task since they are likely to be confused when written in natural language. The lack of functional requirements in a system under development

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2023 doi:10.20944/preprints202309.1392.v1

process would result in the system failure. Similarly, ignoring non-functional requirements would equally lead to troubles, such as project failure, corruption of system integrity, or cost increase.

This study performed experiments for the automatic classification of software requirements using artificial intelligence algorithms on a unique dataset created in the Turkish language for the first time. Since the documentation of these requirements was in the text form written in natural language, the study operated natural language processing methods in addition to artificial intelligence approaches. The study additionally carried out experimental studies within its scope, using traditional machine learning algorithms, deep learning algorithms, and transformer models. As a result, it achieved successful and generalizable results in classifying software requirements as functional and non-functional. The Naïve Bayes Multinomial was the best performing algorithm—with an 92% F-score—among the models created using machine learning methods. The CNN algorithm, however, performed the best—with an 93% F-score—among the deep learning algorithms developed using artificial neural networks. In addition, the dataset underwent a training process with transfer learning methods, frequently used in natural language processing recently, and achieved very high-performance results. Considering the transfer learning methods, the BERTurk algorithm achieved generalizable classification success with an F-score of 95%. As a result, Figure 8 illustrates the algorithms with the highest performance values and the performance evaluation metrics of these algorithms in the experimental studies carried out in three stages to classify the software requirements.

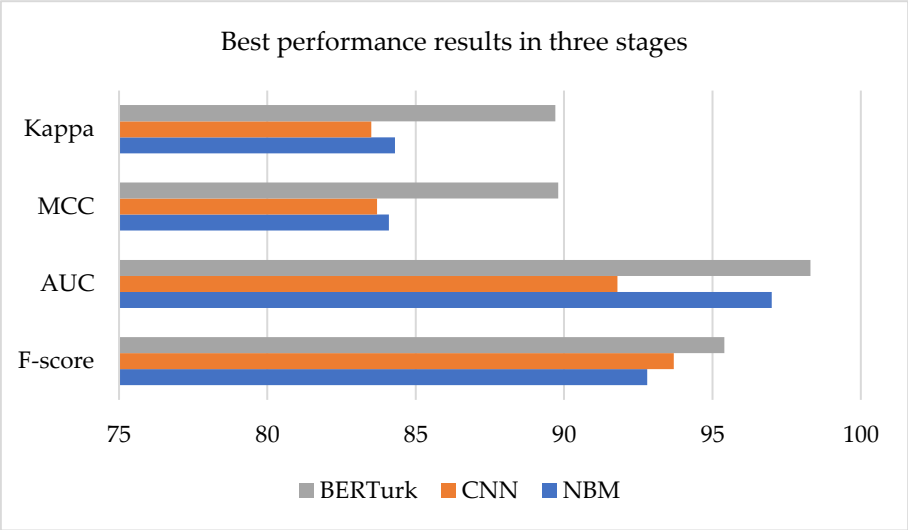

Figure 8. Best performance results in three stages.

Considering the performance evaluation metrics and statistical validation metrics, the study identified the most successful performance with the BERTurk algorithm. Classification of software requirements is a critical issue. The number of Turkish studies on this subject is insufficient in the literature. Therefore, an original dataset created by the samples gathered specifically from Turkish texts, various platforms (windows, web, and mobile projects), diverse sectors, and actual project requirements will significantly contribute to the subject-related studies.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data will be available on a reasonable request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Haque, M. A.; Rahman, M. A.; Siddik, M. S. Non-functional Requirements Classification with Feature Extraction and Machine Learning: An Empirical Study. In 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), IEEE, Dhaka, Bangladesh, (3-5 May 2019), pp. 1-5, doi: 10.1109/ICASERT.2019.8934499.
2. Quba, G. Y.; Al Qaisi, H.; Althunibat A.; AlZu'bi, S. Software Requirements Classification Using Machine Learning Algorithm's, 2021 International Conference on Information Technology (ICIT), IEEE, Amman, Jordan, (14-15 July 2021), pp. 685-690.
3. Limaylla-Lunarejo, M. -I.; Condori-Fernandez, N.; Luaces, M. R. Towards an Automatic Requirements Classification in a New Spanish Dataset, 2022 IEEE 30th International Requirements Engineering Conference (RE), Melbourne, Australia, (15-19 August 2022), pp. 270-271.
4. Halim, F.; Siahaan, D. Detecting Non-Atomic Requirements in Software Requirements Specifications Using Classification Methods. In 2019 1st International Conference on Cybernetics and Intelligent System (ICORIS), IEEE, Bali, Indonesia, (22-23 August 2019), *vol. 1*, pp. 269-273.
5. Li, B.; Li, Z.; Yang, Y. NFRNet: A Deep Neural Network for Automatic Classification of Non-Functional Requirements. In 2021 IEEE 29th International Requirements Engineering Conference (RE), IEEE, Notre Dame, IN, USA, (20-24 September 2021), pp. 434-435.
6. Navarro-Almanza, R.; Juarez-Ramirez, R.; Licea, G. Towards Supporting Software Engineering Using Deep Learning: A Case of Software Requirements Classification, 2017 5th International Conference in Software Engineering Research and Innovation (CONISOFT), IEEE, Merida, Mexico, (25-27 October 2017), pp. 116-120.
7. Bisi, M.; Keskar, K. CNN-BPSO Approach to Select Optimal Values of CNN Parameters for Software Requirements Classification. In 2020 IEEE 17th India Council International Conference (INDICON), IEEE, New Delhi, India, (10-13 December 2020), pp. 1-6.
8. Li, J.; Cheng, K.; Wang, S.; Morstatter, F.; Trevino, R. P.; Tang, J.; Liu, H. Feature selection: A data perspective. *ACM Computing Surveys*, **2017**, *50(6)*, pp. 1-45.
9. Remeseiro, B.; Bolon-Canedo, V. A Review of Feature Selection Methods in Medical Applications. *Computers in Biology and Medicine*, **2019**, *112*, 103375.
10. Shah, F. P.; Patel, V. A Review on Feature Selection and Feature Extraction for Text Classification. In 2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), IEEE, Chennai, India, (23-25 March 2016), pp. 2264-2268.
11. Gokulnath, C. B.; Shantharajah, S. P. An Optimized Feature Selection Based on Genetic Approach and Support Vector Machine for Heart Disease. *Cluster Computing*, **2019**, *22*, pp. 14777-14787.
12. Demir, M. Comparison of the Performances of Classification Algorithms Using Feature Selection Methods. Master's Thesis, Institute of Natural and Applied Sciences, Afyon Kocatepe University, 2021.
13. Nasteski, V. An Overview of the Supervised Machine Learning Methods. *Horizons*. **2017**, *4*, pp. 51-62.
14. Salmi, N.; Rustam, Z. Naïve Bayes Classifier Models for Predicting the Colon Cancer. *IOP Conference Series: Materials Science and Engineering*, **2019**, *546(5)*.
15. Surya, P. P.; Seetha, L. V.; Subbulakshmi, B. Analysis of User Emotions and Opinion Using Multinomial Naive Bayes Classifier. In 2019 3rd International Conference on Electronics, Communication and Aerospace Technology (ICECA), IEEE, Coimbatore, India, (12-14 June 2019), pp. 410-415.
16. Nematallah, H.; Rajan, S.; Cretu, A. M. Logistic Model Tree for Human Activity Recognition Using Smartphone-Based Inertial Sensors. In 2019 IEEE SENSORS, IEEE, Montreal, QC, Canada, (27-30 October 2019), pp. 1-4.
17. Asif, A.; Majid, M.; Anwar, S. M. Human Stress Classification Using EEG Signals in Response to Music Tracks. *Computers in Biology and Medicine*, **2019**, *107*, pp. 182-196.
18. Sadiq, A. Intrusion Detection Using the WEKA Machine Learning Tool. Master's Thesis, Department of Electrical and Computer Engineering, University of Victoria, Canada, 2021.
19. Aborisade, O.; Anwar, M. Classification for Authorship of Tweets by Comparing Logistic Regression and Naive Bayes Classifiers, In 2018 IEEE International Conference on Information Reuse and Integration (IRI), IEEE, (06-09 July 2018), Salt Lake City, UT, USA, pp. 269-276.
20. Cahya, R. A.; Bachtiar, F. A.; Mahmudy, W. F. Comparison of Bagging Ensemble Combination Rules for Imbalanced Text Sentiment Analysis. *Journal of Information Technology and Computer Science*, **2021**, *6(1)*, 33-49.

21. Ali, A. T.; Abdullah, H. S.; Fadhil, M. N. Voice recognition system using machine learning techniques. *Materials Today: Proceedings*, **2021**.
22. Alsafy, B. M.; Aydam, Z. M.; Mutlag, W. K. Multiclass Classification Methods: A Review. *International Journal of Advanced Engineering Technology and Innovative Science*, **2019**, *5(3)*, pp. 1-10.
23. Borandag, E. Software Fault Prediction Using an RNN-Based Deep Learning Approach and Ensemble Machine Learning Techniques. *Applied Sciences*. **2023**, *13(3)*, 1639.
24. Shiri, F. M.; Perumal, T.; Mustapha, N.; Mohamed, R. A Comprehensive Overview and Comparative Analysis on Deep Learning Models: CNN, RNN, LSTM, GRU. *ArXiv*, **2023**.
25. Bisi, M.; Keskar, K. CNN-BPSO approach to Select Optimal Values of CNN Parameters for Software Requirements Classification. In 2020 IEEE 17th India Council International Conference (INDICON), IEEE, New Delhi, India, (10-13 December 2020), pp. 1-6.
26. Fong, V. L. Software Requirements Classification Using Word Embeddings and Convolutional Neural Networks. Master's Thesis, Department of Computer Science, California Polytechnic State University, San Luis Obispo, 2018.
27. Santhanam, S.; Shaikh, S. A Survey of Natural Language Generation Techniques with a Focus on Dialogue Systems - Past, Present and Future Directions. *ArXiv*, **2019**.
28. Wei, W.; Zhao, X. Fault Text Classification of On-Board Equipment in High-Speed Railway Based on Labeled-Doc2vec and BiGRU. *Journal of Rail Transport Planning & Management*, **2023**, *26*, 100372.
29. Bouschery, S. G.; Blazevic, V.; Piller, F. T. Augmenting Human Innovation Teams with Artificial Intelligence: Exploring Transformer-Based Language Models. *Journal of Product Innovation Management*, **2023**, *40(2)*, pp. 139-153.
30. Lee, J.; Tang, R.; Lin, J. What Would Elsa Do? Freezing Layers During Transformer Fine-Tuning. *ArXiv*, **2019**.
31. Acheampong, F. A.; Nunoo-Mensah, H.; Chen, W. Transformer Models for Text-Based Emotion Detection: A Review of Bert-Based Approaches. *Artificial Intelligence Review*, **2021**, 1-41.
32. Bozuyla, M.; Ozcift, A. Developing a Fake News Identification Model with Advanced Deep Language Transformers for Turkish COVID-19 Misinformation Data. *Turkish Journal of Electrical Engineering and Computer Sciences*, **2022**, *30(3)*, pp. 908–926.
33. Joshy, A.; Sundar, S. Analyzing the Performance of Sentiment Analysis Using BERT, DistilBERT, and RoBERTa. 2022 IEEE International Power and Renewable Energy Conference (IPRECON), IEEE, Kollam, India, (16-18 December 2022), pp. 1-6.
34. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *ArXiv*, **2019**.
35. Thi, H.D.; Andres, F.; Quoc, L.T.; Emoto, H.; Hayashi, M.; Katsumata, K.; Oshide, T. Deep Learning-Based Water Crystal Classification. *Applied Sciences*. **2022**, *12(2)*, 825.
36. Ozcift, A.; Gulten, A. Classifier Ensemble Construction with Rotation Forest to Improve Medical Diagnosis Performance of Machine Learning Algorithms. *Computer Methods and Programs in Biomedicine*, **2011**, *104*, 443–451.
37. Ozhan, E. Improving the Information Extraction Process from the Web with Machine Learning Methods. *Afyon Kocatepe University International Journal of Engineering Technologies and Applied Sciences*, **2020**, *3(2)*, 52-59.