IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 17, 2022, accepted January 28, 2022, date of publication February 7, 2022, date of current version March 28, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3149480

Sequence Embeddings Help Detect Insurance Fraud

IVAN FURSOV[1], ELIZAVETA KOVTUN[1], RODRIGO RIVERA-CASTRO[1], ALEXEY ZAYTSEV[1], RASUL KHASYANOV[1], MARTIN SPINDLER[2], AND EVGENY BURNAEV[1,3]

[1]Skolkovo Institute of Science and Technology, 121205 Moscow, Russia
[2]Faculty of Business Administration, University of Hamburg, 20146 Hamburg, Germany
[3]AIRI, 105067 Moscow, Russia

Corresponding author: Rodrigo Rivera-Castro (rodrigo.riveracastro@skoltech.ru.)

This work was supported in part by the Russian Foundation for Basic Research under Grant 21-51-12005 NNIO_a, and in part by the German Research Foundation (DFG) under Project 448795504.

ABSTRACT Roughly 10 percent of the insurance industry's incurred losses are estimated to stem from fraudulent claims. One solution is to use tabular data to construct models that can distinguish between claims that are legitimate and those that are fraudulent. However, while canonical tabular data models enable robust fraud detection, complex sequential data have been out of the insurance industry's scope. For health insurance, we propose deep learning architectures that process insurance data consisting of sequential records of patient visits and characteristics. Both the sequential and tabular components improve the quality of the model, generating new insights into the detection of health insurance fraud. Empirical results derived using relevant data from a health insurance company show that our approach outperforms state-of-the-art models and can substantially improve the claims management process. We obtain a ROC AUC metric of 0.873, while the best competitor based on state-of-the-art models achieves 0.815. Moreover, we demonstrate that our architectures are more robust to data corruption. As more and more semi-structured event sequence data become available to insurers, our methods will be valuable for many similar applications, particularly when variables have a large number of categories, such as those from the International Classification of Disease (ICD) codes or other classification codes.

INDEX TERMS Deep learning, embeddings, fraud detection, health insurance, social media and text, structured data.

I. INTRODUCTION

Fraud causes substantial costs and losses for the finance and insurance industries. Examples include fraudulent credit card transactions and insurance fraud. Indeed, experts estimate that each year roughly 10 percent of the insurance industry's incurred losses and loss adjustment expenses stem from fraudulent claims.[1] Fraud detection is a critical function and core competence in these industries and their claims management processes.

The proliferation of digitization in finance and insurance has led to big datasets suited to fraud detection. In this paper, we propose architectures for categorical sequence embeddings via deep learning that help improve the classification of fraudulent and valid claims compared to other machine learning methods.

Analyzing fraud with statistical and machine learning methods poses particular challenges. First, claims data are often available in a so-called unstructured format that is challenging to process using classic machine learning approaches. Second, fraud data are highly unbalanced because the number of fraudulent cases is minimal compared to the number of non-fraudulent ones, and we can consider each fraud an anomaly. Third, claims do not have a fixed length because the number of items billed in a claim varies. These characteristics influence the choice of classification approach and performance measures.

It is well known that deep learning outperforms other machine learning methods for analyzing unstructured data, such as text or images. In this paper, we develop deep learning architectures tailored to claims data and to handling each of the challenges mentioned above. For our

The associate editor coordinating the review of this manuscript and approving it for publication was Massimo Cafaro.

[1]Cf. https://www.iii.org/article/background-on-insurance-fraud.

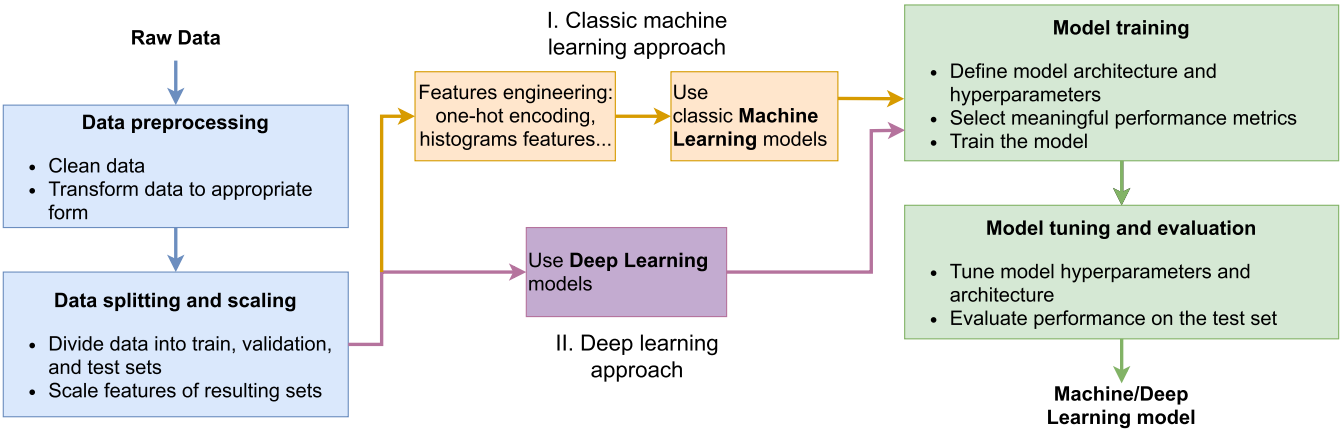


FIGURE 1. **Workflows for a prediction model for classic machine learning and deep learning. The development of machine learning models requires accurate feature engineering, whereas deep learning models can handle unstructured categorical sequences without sophisticated preprocessing and, thus, we can skip the *Feature engineering* step. In the paper we briefly consider each step of the data-based model construction, while focusing on approaches in the middle stage related to the model construction.**

analysis, we use claims from outpatient doctor visits, which have a particular structure. These consist of unstructured categorical sequences of treatments and have properties of text data (for example, the varying number of billed items mentioned above). Moreover, medical claims usually encode treatments as categorical variables with thousands of categories. In this paper, we develop methods for analyzing such (semi)unstructured data.

We test our methods on a dataset from a major health insurance company. Our empirical results show that they outperform other state-of-the-art methods in the prediction of fraudulent claims, making the claims management process more efficient. The summarized workflow we follow is presented in Figure 1.

We start with some information on insurance fraud and fraud detection and an overview of the literature in section II. In section III, we present models and methods for analyzing general text data and analyzing the unique structure of claims data to detect insurance fraud. After this, we describe our data in section IV. Similarly, we describe the models available for these tasks in section V. Further, in section VI, we present the results of our analysis and experiments. Finally, we conclude in section VII. Additional results of our analysis can be found in Appendix.

A. INNOVATION

Many approaches to anomaly/fraud detection require problem-specific solutions. At the same time, deep learning promises to provide more general models that will extract insights as embeddings or representations: from unstructured, complex and high-dimensional data, deep learning generates a meaningful representation of small dimensions that we can easily use to indicate a particular system state. There are many successful cases and applications where this was achieved in complex settings. Numerous cases back this hope for complex scenarios with unstructured data, sequential data, and imbalanced class distribution. Nevertheless, one should still carefully examine the robustness of the various deep learning models by deploying them.

We aim to address this gap in the literature. There are no deep learning models that consider fraud prediction for medical insurance based on both claims data and general patient features.

II. BACKGROUND AND RELATED WORK

The literature review consists of four parts related to different aspects of the studied domain. We start with a description of anomaly detection as a major challenge in machine learning and its relation to fraud detection. After that, we provide an overview of the application of fraud detection algorithms in various fields, such as healthcare and insurance sector. Then, we discuss how the concept of embedding is leveraged in order to work with diverse types of sequential data. Finally, we review different ways to cope with class imbalance problem that is typical for fraud detection.

1) ANOMALY DETECTION

Detecting anomalies in data is one of the core problems in data analysis ([1]). Researchers from many disciplines investigate it for application domains, including time-series modeling, [2], [3], predictive maintenance of technical systems, [4], [5], and applications in the finance and insurance industries, [6].

Anomalies in data are worthy of attention because they can translate into important and often critical actionable information in a wide variety of application domains. For example, in credit card transactions, anomalies can indicate when unauthorized purchases have occurred due to credit card or identity theft, [7]. Money laundering, as a type of financial fraud, can be detected by a simultaneous analysis of trading networks and features of its entities, [8]. The outcome of seeking a low-rank approximation and analyzing

the residuals of a graph-based similarity matrix and feature matrix leads not only to detection of fraud patterns but also to tracing of the suspicious features. For a comprehensive overview, we refer to the reviews by [1] and [9], which are dedicated to various fraud detection problems and based on machine learning.

Similarly, anomalies in health insurance claims can flag up potential insurance fraud carried out to gain inappropriate compensation for services not rendered, [10].

2) MACHINE LEARNING FOR HEALTHCARE AND INSURANCE

While experts have been using machine learning methods to solve healthcare and insurance problems successfully for several years, the literature does not seem to have covered deep learning and embeddings for fraud detection until recently. Reference [11] provide one of the few examples focusing on detecting automobile insurance fraud. They process text descriptions of the accidents, extract standard text features manually, and combine these with deep-learning models. Although the accuracy of their model is superior to that of existing approaches, neither the precise architecture of the best model nor the approach to training and validating it is outlined clearly in their paper. Another example is [12]. The authors use hierarchical clustering with deep neural networks to detect fraud in candidates' descriptions during job recruitment, significantly improving the prediction accuracy of conventional methods. Reference [13] uses manually-crafted features to predict instances of automobile insurance fraud. Reference [14] highlight the importance of constructing robust data-based models in healthcare. The authors generate adversarial examples for predictive models based on multivariate electronic health records (EHR), represented by temporal sequences of numerical data. To find efficient attacks on medical records, they propose an optimization-based attack strategy limiting the size of the perturbation of the initial input. An analysis of the best attacks helps to identify susceptible locations in each patient's medical records and subsequently prevent mistakes in the most critical measurements. Reference [15] develop an unsupervised deep learning model for fraud detection by utilizing the information on insured people. The authors use the Autoencoder to obtain the aggregate reconstruction error (A-RE) for the underlying data and further indicate high A-RE instances as fraud. Reference [16] present a comparison of metrics when using various machine/deep learning models with combination of data-imbalance techniques to detect health insurance fraud.

3) MACHINE LEARNING EMBEDDINGS

We can use embeddings to address anomaly detection problems in different application domains. For example, [17] develop an approach to embedding entities, representing events from computer systems, into a common latent space. Each event involves heterogeneous attributes such as time, user, source process, destination process, and more. Reference [18] study the problem of detecting structurally inconsistent nodes in graphs, to identify, for example, outlier authors in a network in which different authors connect through co-authorship of papers.

At the moment, however, the community is especially interested in embeddings for applications related to natural language processing. Given the breadth of the field, we focus here on works with embeddings of simple entities, such as words. These include the classic TF-IDF approach, [19], the more recent and well-known word2vec, [20], and GloVE, [21]. The last two methods consider concurrences of words, whereas TF-IDF is solely a normalized one-hot-encoding for a dictionary of words at hand.

For event sequences related to sequences of clients' visits, there are also numerical approaches such as that described in [22]. For a related use case, [23] construct unsupervised embeddings based on deep learning architecture for various types of event sequence data. Financial transaction data are another type of event sequence. Here, deep learning provides significant quality improvements, allowing better model and embeddings quality, for example, in the works of [24] and [25]. However, [26] emphasizes the importance of assessing the robustness of this type of approach.

In addition to embeddings for single events, we need those for whole sequences. There are multiple approaches to concatenate embeddings of simple entities. For example, we can construct a text embedding from an embedding of each word within a given text. Simple heuristics include taking the maximum value among each dimension for word embeddings or taking mean values, [27]. More complex approaches use convolutional neural networks, recurrent neural networks and transformers, [27]–[29].

4) IMBALANCED CLASSIFICATION PROBLEMS

Skewed distributions or imbalanced classes are one of the most critical challenges to solving fraud detection problems. Generally speaking, there are far fewer instances of fraudulent items than normal ones. We refer to classes with fewer objects as minority classes and other classes as majority classes in the literature. The resulting imbalance makes it difficult for learners to detect patterns in the minority class data. Reference [30] mentions three broad approaches to learning from imbalanced data:

- Data-level methods that modify the dataset to achieve balance between the minority and majority classes in their distributions and remove difficult observations,
- Algorithm-level methods that directly modify existing learning algorithms to alleviate the bias towards majority class objects and adapt them to mining data with skewed distributions,
- Hybrid methods that combine the advantages of these two approaches.

For the data-level approach, [31] use under-sampling for a skewed class in a fraud detection system for credit cards, and [32] assess how resampling the multiplier selection influences classification accuracy. For the algorithmic-level approach, [33] use cost-sensitive classifiers to address

the class imbalance problem. In turn, [34] propose the FraudMiner model to handle class imbalance by entering the unbalanced data directly into the classifier. Some papers try to combine both approaches to make them work for the general data scenario such as [35]. More general-purpose approaches include over-sampling, [36], combinations of over- and under-sampling, [37], and meta-learning to automate the selection of imbalanced classification methods, [38]. Reference [39] explore highly imbalanced datasets connected with credit card frauds and show that features selection methods can enhance the metrics of the classifiers. Reference [40] consider financial transactions and develop the proactive strategy to fraud prevention that helps to overcome the issue of imbalanced data. The proposed conversion of the time-series into a transformed domain allows exploiting only a legitimate class, thereby making it possible to operate even in the absence of previous fraudulent cases. Reference [41] provide the comprehensive overview of existing classic machine learning and deep learning techniques to address class imbalance. Moreover, the authors discuss various metrics and their particular application to achieve a better reflection of models performance.

III. METHODS

A. LEARNING OF CLASSICAL DATA-BASED MODELS

The typical scenario for supervised learning starts with data consisting of a sample of observations. More formally, we define $\{(\mathbf{x}_p, y_p)_{p=1}^{P}\}$ as our dataset with P input/output pairs. For each input $\mathbf{x}_p$, an N-dimensional vector, we have a corresponding output y_p, a categorical or discrete variable with so-called labels.

Each sample, $(\mathbf{x}_p, y_p)$, contains a description of an object given by features, x_p, and the value/label of the target variable for that object, y_p. In disability insurance, for example, annual income, education, occupation, age, and past medical records describe a customer. Furthermore, the target variable, y_p, is a binary label, $y_p \in \{0, +1\}$, that represents whether the customer's claim is fraudulent $(+1)$ or valid (0).

Our dataset stems from a large international health insurance company. The observations consist of medical bills, including information about the treatments provided, as well as their costs, types, total amounts, and general client features such as age and occupation. The target variable depends on whether the bill was classified as fraudulent by a clerk handling the claim.

Thus, we can learn a model that predicts the target variable, y_p, by taking a new object's features, $\mathbf{x}_p$, as its input.

The power of machine learning is that we can learn a model that adapts to a given sample and is able to generalize well to unseen data similar to that in the training sample. Machine-learning methods make it possible to learn non-linear and complex relationships in datasets.

Data scientists have devised various ways to generate features manually from complex but unstructured data such as images and texts. They use these features as input for classic machine learning models. The standard limitation of these approaches is that they require object descriptions in a restrictive format. Usually, they use fixed, small-length vectors, which is not feasible for many real-world objects such as texts or images with millions of pixels.

For insurance-related tasks, the corpus consisting of bills has different lengths for different patients and describes visits to a doctor. Each patient has a different number of visits or medical bills listing a varying number of treatments. We can construct one-hot-encoding features counting the number of specific prescriptions or specific visit types for a given patient. In other fields, such as economics, manually generated features are also widespread. For example, the variable ''age'' is often constructed from the variable ''date of birth''. However, we observe that such approaches yield results of reasonable but limited quality.

B. THE DEEP LEARNING REVOLUTION

The deep learning revolution changed the rules of the game for machine learning data-based models, [42]. Now algorithms can learn representations or embeddings of object descriptions to generate features that are informative enough to provide accurate predictions while using relatively simple machine learning models. Examples are the fully connected neural networks (FCNs) with only a few layers. The strength of deep learning lies in feature extraction, which means learning informative features from high-dimensional unstructured and complex input data.

The most successful applications of deep learning are in the field of image processing. However, deep learning impacts other areas, [43], such as natural language processing, [44], or graph data, [45].

The key idea behind deep learning is to apply a sequence of non-linear transformations on object descriptions, the so-called layers of the neural network. The objective is to produce an informative embedding and then use it as input for a final classifier.

Deep learning models enjoy numerous architectures and variations that can be chosen for the task at hand. With this in mind, in this study, we test several basic architectures from the deep learning literature, to find the best one for our settings. These are Gated Recurrent Unit (GRU), Long Short-Term Memory (LSTM), Convolutional Neural Network (CNN), Simple Word Embeddings-based Models (SWEM), and Transformer. GRU and LSTM belong to the family of Recurrent Neural Networks (RNN), which have unique mechanisms for regulating the flow of data and handling the problems of the canonical RNN by enabling long memory and efficient training. CNN is an architecture widely used in computer vision tasks. However, the leveraging of processing data with convolutional kernels might also be beneficial in other domains. Furthermore, the SWEM is sometimes comparable with more complex models and deserves study when working with embeddings. In addition, we consider the Transformer model, whose attention-based mechanism has revolutionized deep learning for natural language processing.

C. CONCEPT OF EMBEDDINGS

In this paper, we address the problem of representing healthcare insurance data by using embeddings for fraud detection. An embedding is a transformation of object descriptions to vectors that belong to the same low-dimensional space. These low-dimensional representations have the characteristic that similar instances have a smaller distance between them in the embedded space. For example, a helpful embedding provides vector representations of words so that the relationship between two vectors mirrors the relationship between the two terms. We can see this in the popular word2vec model by [46] from the natural language processing literature. It constructs a low-dimensional vector of real numbers. As a result, words appearing in a similar context have similar vector representations. In short, embeddings are a general framework for dimensionality reduction and a practical approach to extracting features of intrinsic relations between complex objects.

In our case, we learn an embedding space explicitly constructed for sequential data from healthcare insurance claims. Such representation significantly helps in the detection of fraudulent patterns.

To make these ideas clear, let us suppose that a text consists of words from a dictionary of V different words. The classic way to transform the text into numeric features is with a one-hot encoding for each word in the text. In other words, we represent each word in the text sequence as a V-dimensional vector consisting of zeros except for one entry at the location corresponding to that word. This approach is standard for encoding categorical variables. However, the representation is not efficient if the dictionary is extensive, as is typical for general texts and healthcare data. In turn, via modern approaches, we can represent embeddings of words from the dictionary with real-valued s-dimensional vectors, such that s is much lower than the size of V. This allows for a compressed representation of the textual input description. In this representation, the entries of the embedded vector are usually all different from zero. The embedding of the dictionary into the vector space should also maintain some of the relations between the words. For example, the desirable property of the word embedding is that the difference in the vector space between the words ''queen'' and ''king'' should be similar to that between the words ''woman'' and ''man''. Learning word embeddings with such properties makes them a powerful tool for text analysis [47]. We can learn such embeddings from unstructured data in a supervised or unsupervised way.

D. APPLICATION OF EMBEDDINGS

In recent years, learning embeddings that represent complex relationships within data is becoming common in the machine learning community. Authors apply different embedding types to various domains, such as natural language processing (NLP), network analysis, and computer vision.

As noted above, word embeddings, such as the word2vec, GloVe by [21], AdaGram by [48] and others, provide vector representations of words such that the relationship between two vectors mirrors some linguistic connection between the two terms. In supervised problems, word and sentence embeddings have proven effective for natural language processing tasks such as part-of-speech tagging, [49], phrase-based machine translation, [50], named-entity recognition, [51], and word sense disambiguation, [48].

Beyond text, we can use embeddings for all types of data representations. For example, graph and network embeddings attempt to capture local and global attributes on graphs. One option consists of engineered graph features, and another of training on graph data. Classical approaches for graph embeddings include feature-based methods, such as graph kernels, [52], [53], and data-driven algorithms that yield distributed graph representations, [54]–[56]. Using such embeddings, we can solve various tasks related to network data analysis. One example is [57] who use anonymous walk embeddings for graph influence set completion.

IV. DATA DESCRIPTION

A. OVERVIEW

Our dataset consists of many health insurance claims from the outpatient care setting. It comprises about 0.38 million patients' medical bills with 3.3 million items in total. Each data point is a sequence of treatments encoded with anonymized IDs.

There are 15 input features in total. In the data, we have two types of feature for each patient: general and visit-specific. The first type includes age, sex, insurance type, and doctor specialty. These features relate to the patient, insurance, and doctor in general. For each patient, visit-specific features describe individual outpatient visits. For this type of feature, we consider treatment IDs, the type of treatment, the number of treatments, the cost of therapy, a factor that increases the cost of treatment due to potential complications, the total billable amount, the billing type, the cost category, and the performance type. We encode the type of treatment using one of more than two thousand categories. However, some of the available features are uninformative, and our experiments reveal that the models perform better if we discard them. In the following discussion, all features except treatment IDs will be referred to as global features.

For each record, we have a label. The label is either ''fraudulent'' and coded as 1 or ''non-fraudulent'' (valid) and coded as 0. Here ''fraudulent'' refers to the fact that the final amount of the bill was amended, which can happen for many reasons. About 2% of records are fraudulent. The challenge is to identify whether the record corresponds to a fraudulent activity based on the input features available.

We present in Table 1 a sample of the final version of the refined data, which consist of both input features and a target to train all our models.

The number of treatments on a bill varies significantly among patients. In Figure 2, we provide a histogram of the number of treatments and items for the dataset. The

distribution of treatments is nonuniform. Most patients have only a small number of treatments in the outpatient setting.

We make a represenatative random subsample of our data and its description on a public repository available.[2]

B. TREATMENTS

Our approach aims to determine whether information from labels of treatments can help automatically identify fraud. Because the number of treatments varies for each patient, we must aggregate all treatments into one vector. To do so, we construct an embedding of all treatments into a vector of a fixed dimensional size. In the literature, we can find approaches to dealing with varying input sizes. One example is [58].

In our case, the natural way to construct embeddings is to use methods that have their roots in natural language processing (NLP) because a medical bill lists a series of individual treatments from a sizeable but finite dictionary. Each anonymized treatment belongs to a dictionary of size 2205. We summarize treatments in up to 17 upper-level treatment groups. Similarly, we create another dictionary feature with a size of 24 for the type of benefit.

Also, we emphasize that specific treatments are not more prone to fraud. If we measure the correlation between the presence of a particular treatment and the target variable, the maximum absolute value for correlation is only 0.0243. We must therefore use more sophisticated machine learning approaches to make it possible to identify fraudulent series of treatments.

C. DISTRIBUTION OF TREATMENTS

To understand the essence of our data, we rank the number of treatments in the dataset. For example, the most frequent treatment has a rank of one. This approach is close to the empirical Zipf's law, [59], commonly found in NLP. Intuitively, the frequency of any treatment is roughly inversely proportional to its rank in the frequency table. Figure 3 demonstrates this behavior for our dataset as a log-log plot. However, we observe a heavier tail with rarer treatments having higher frequencies than what we would expect from Zipf's law. One interpretation is that there are few rare treatments in our data and that the diversity is higher for tasks using natural language texts.

V. MODELS

A. CLASSIC MACHINE LEARNING APPROACH

Using classic machine learning algorithms, we consider two types of representations for the sequence of treatments. These are bag of words (BoW) and term frequency-inverse document frequency (TF-IDF).

The idea behind BoW is to represent a sequence $t = \{w_1, w_2, \ldots, w_{i_t}\}$ by counting the number of times $n_{w,t}$ a token w appears in it. This technique generates a vector of frequencies of the tokens in the considered sequence. Some terms, such as "a", or "the", appear multiple times but provide little information in sentences. Therefore, a normalized version of BoW leads to TF-IDF. A TF-IDF is the product of the term frequency and inverse document frequency for texts and documents. For the frequency term, we divide $n_{w,t}$ by the total number of words in the text $\sum_{w'} n_{w',t}$, whereas, for the inverse document frequency, we divide the logarithm of the total number of documents by the number of records that contain the considered word w. We can swiftly transfer the idea behind the TF-IDF to other contexts beyond words, texts, and documents. We emphasize that neither of these approaches considers the order of a token in a sequence. However, neglecting the order can decrease performance in many problems.

To apply the ideas of BoW and TF-IDF in our setting, we form a dictionary of all unique treatment IDs. In this case, the quality of the classic machine learning models with BoW features is slightly better than with TF-IDF. Hence, we discard the latter and consider only a more straightforward BoW processing in the following discussions.

1) LOGISTIC REGRESSION, RANDOM FOREST AND GRADIENT BOOSTING

The literature considers logistic regression a common baseline for predictive modeling. Despite the simplicity of the rules underlying the construction of the relationship between features and the probabilities of belonging to a particular class, this model gives solid results for many problems.

Another popular model in the classic machine learning literature is the Ensemble of decision trees according to [60]. In this model, each decision tree distributes the input objects to the leaves based on the features of the object and learned rules in the nodes. In a leaf, the classifier returns the probability of belonging to a specific class. In the Ensemble, we use a weighted sum of basic decision tree classifiers. The ensemble of decision trees offers many benefits: it is fast to construct, can almost avoid overfitting, successfully handles missing values and outliers, and provides competitive performance, [60].

A popular and efficient way to perform classification using the predictions of several decision trees is the Gradient boosting algorithm. It trains sequentially by setting the target to the next tree based on the errors of previous trees. One of the many advantages of Gradient boosting is its ability to solve imbalanced classification problems and easily incorporate various imbalanced classification heuristics, [35].

We select the LightGBM framework presented by [61] as our preferred implementation of Gradient boosting for our experiments. This high-speed implementation provides state-of-the-art performance, needs less memory to run, and supports learning on graphical processing units (GPUs). An added benefit is that we can tune it by adjusting a vast number of hyperparameters.

[2]https://github.com/fursovia/fraud_detection/tree/2021_update

TABLE 1. Example of data used in our model. The target indicates whether the record is fraudulent (1) or not (0). D_{XX} corresponds to a specific treatment given in a particular order to a patient.

Patient ID	Age	Sex	Insurance type	Amount	Doctors speciality	Treatments	Target
5	37	1	1	126	1	[D_23, D_69, D_156]	0
98	49	0	1	346	1	[D_53, D_129, D_6, D_353]	0
273	28	1	0	789	0	[D_74, D_1]	1

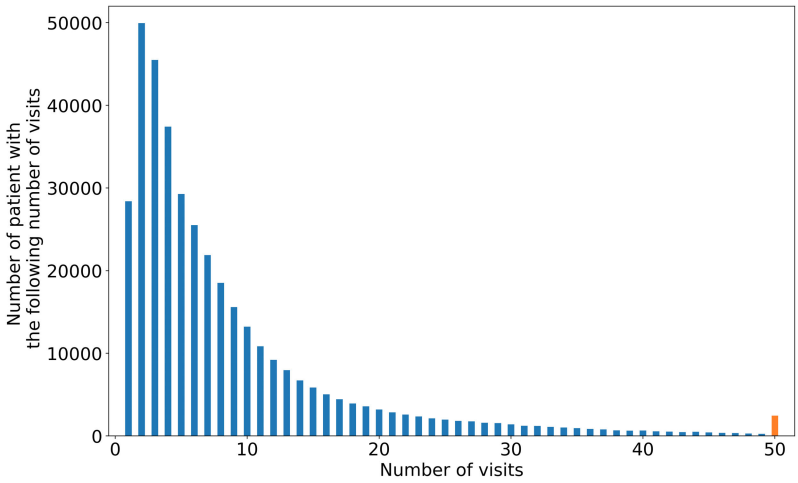


FIGURE 2. Histogram of the number of billing items for patients. The right orange bar represents the number of patients with 50 or more billing items. Most patients have less than 5, with most of them having 2. The orange column corresponds to the number of patients with more than 50 treatments.

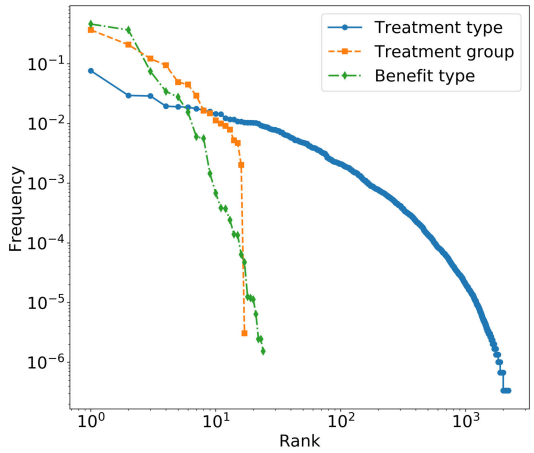


FIGURE 3. Log-log plot for ranks and corresponding frequencies and groups of treatments, as well as benefits in the dataset. The chart significantly deviates from the straight line expected according to Zipf's law.

B. DEEP LEARNING APPROACHES

We examine several deep learning models to identify those capable of processing our specific data best and yielding the most precise predictions. For this, we describe the overall pipeline of working jointly with diverse types of features to obtain the fraud probability distribution for each insurance claim.

1) MODEL PIPELINE

a: TREATMENTS EMBEDDING LAYER

First, we embed each treatment in a sequence into a vector of dimension d. The model identifies its optimal values during the learning process. However, there are a different numbers of treatments for each patient. Thus, we pad all lists of treatments with empty treatments to achieve an equal length across our vectors. In conjunction with this, we construct masks for the padded parts to pass them to the subsequent layers and ignore them.

b: OBTAINING THE ENCODING VECTOR FROM TREATMENT EMBEDDINGS

Then, we pass the resulting embeddings to the input of one of the following neural models: CNN, GRU, LSTM, SWEM, and Transformer to get a single output vector. At this stage, we construct a so-called encoding vector, which should encapsulate as much information as possible from a sequence of embeddings.

c: APPROACHES TO USE GLOBAL FEATURES

Third, the global features consist of numerical and categorical features. As part of our pipeline, we scale them beforehand. Then, we try to use information from categorical features in two ways for our classification problem. In the first approach, we treat all global features in each record as a vector and pass it as an input for several feed-forward layers. After that, we concatenate the resulting vector with the encoding vector obtained from the treatments, i.e., the visit-specific features. In the second approach, we obtain the embedding of each distinct global feature and then receive an encoding vector by simply averaging all embeddings.

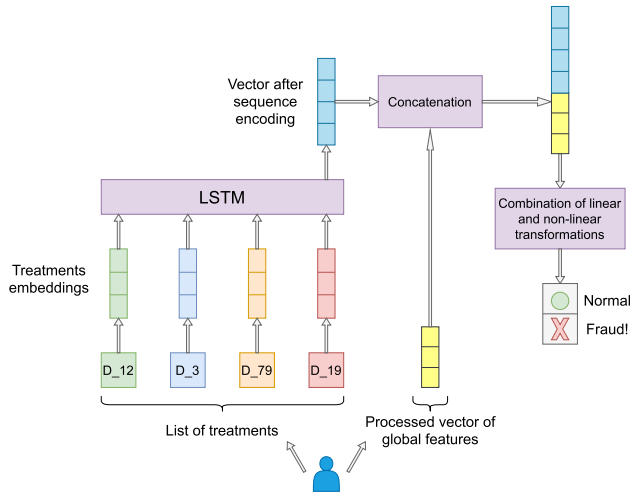


FIGURE 4. The model architecture to identify fraudulent claims. We pass each treatment's embedding to the input of a Neural Network model, for example, an LSTM, to obtain the encoding vector. Then, we concatenate this encoding vector with processed global features and pass it to the several layers of linear and non-linear transformations to obtain the final probability distribution.

d: OBTAINING THE FINAL PROBABILITY DISTRIBUTION

Fourth, we process it further to get the final probability distribution after obtaining the concatenated vector, which incorporates global and visit-specific features. In particular, we apply dropout and then several fully connected layers such as a gated combination of linear and non-linear transformations. Finally, we implement a linear layer to obtain logits, which we can transform into probabilities through the softmax function. Then, by analyzing the output probabilities, we can decide whether a record is fraudulent.

An example of the architecture we use to obtain the encoding vector from treatment embeddings with an LSTM model is available in Figure 4. We depict the global features related to the patient as a single vector. We assume that we process them with one of the above mentioned methods.

2) TRAINING DEEP-LEARNING MODELS

We train the full model for 100 epochs and set the parameter "patience" to 5 epochs. This means that if our target metric does not improve after five epochs, we stop training. We use the Adam optimization algorithm by [62] with a learning rate equal to 0.001 and minimize cross-entropy loss. Our classification problem is imbalanced, so we train the model with balanced batches for more effective learning, implying oversampling from the minority class, because this approach shows the best results in general and specifically for our problem.

VI. RESULTS

A. METRICS

To evaluate the classification models, we can use many metrics. However, handling our imbalanced classification problem requires detecting the minority class with high precision. For fraud detection, it is crucial to find all actual positive events. An outstanding classifier would correctly predict all instances of the minority class with a low false alarm rate.

TABLE 2. Confusion matrix for binary classification.

	Actual Positive	Actual Negative
Predicted Positive	True Positive (TP)	False Positive (FP)
Predicted Negative	False Negative (FN)	True Negative (TN)

Below we consider the canonical metrics for measuring the quality of imbalanced classification problems.

The confusion matrix forms the basis of our metrics. Using Table 2, we can generate the most common metrics in the literature to estimate the performance of a classifier with different focuses, such as the area under the ROC curve (ROC AUC) and the area under the PR curve (PR AUC). These metrics use standard concepts such as recall, precision, and false-positive rate. To be more precise in our discussion, we introduce the following notation.

The *recall* is the true-positive rate, $\frac{TP}{TP+FN}$. It is the percentage of positive instances correctly classified. When this metric is equal to one, it means that we can identify all fraud cases.

The *precision* is the percentage of positive instances among positive predictions. We define it as $\frac{TP}{TP+FP}$. A high value for precision means that our model captures the underlying fraud behavior.

The *false-positive rate* is $\frac{FP}{FP+TN}$ and represents the percentage of positive instances that the model misclassifies.

We define the F1 score as $2\frac{\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}}$. It lies in the interval [0, 1]. Hence, we prefer higher F1 scores.

The canonical ROC curve shows how well the model classifies items in the two classes. Ideally, the first class must display a high value for the true-positive ratio (TPR), whereas the second class must show a low false positive ratio (FPR). Therefore, a high-quality prediction corresponds to a "balance" between these values.

Similarly, the PR AUC describes how well the model classifies the minor class, or fraud, class. We want to maximize the TP number. This metric is well-suited for imbalanced datasets, because it reflects the model's ability to identify fraudulent behavior. The focus on the minority class enables the PR AUC metric to predict the most relevant category better than than other metrics.

To sum up, we consider four relevant metrics to evaluate the performance of our models: ROC AUC, PR AUC, F1 score, and Confusion matrix. ROC AUC, PR AUC, and F1 score lie in the interval [0, 1], and our goal is to maximize them. Confusion matrices that we provide have the following structure: $[[TN, FP], [FN, TP]]$. Reference [41] prove that our chosen metrics ensure the comprehensive performance review.

B. VALIDATION PROCEDURE

To evaluate the performance of the model, we use data splitting. We randomly split the dataset and follow a standard

approach among data science practitioners, using 60% of the medical bills during the model *training* phase and the remaining 40% to validate and test our models. Given our imbalanced data, we do the splits in a stratified way: the ratios of classes in the training and test samples coincide with those of the initial selection. In practice, to get a more reliable estimate of model performance, we generate several partitions and distribute them into train and test sets (cross-validation). In subsection A in Appendix we show that the metrics evaluated on several splits and a single split are close. Due to this fact, we provide an analysis of the experiments for a fixed train-test split.

C. RESULTS

In this section, we evaluate the metrics of our classic machine learning and deep learning models. We compare the results for different subsets of features and identify the model that performs best. Besides, we try several techniques to alleviate class imbalance issue. Also, we provide dependencies of metrics on the size of the train set, the dimension of encoding, and embedding vectors. Moreover, we explore the robustness of our models by corrupting the initial data. Such a thorough analysis of the models provides a clear picture of the best way to construct the classification model considering the nature of the data. For reproducibility purposes, all implementations and experiments are available on a public repository.[3] All necessary packages and their versions to train deep learning models are indicated in poetry.lock and pyproject.toml files in our repository. To build classic machine learning models we use scikit-learn library [63] and LightGBM framework [61]. The technical details for the experiments are specified in subsection B in Appendix.

1) PERFORMANCE OF MACHINE LEARNING MODELS. USEFULNESS OF VISIT-SPECIFIC FEATURES

We recall that we have two types of feature. One set consists of global features, and the other is visit-specific. With this in mind, we generate BoW features from the visit-specific ones and compare three different sets of features. These are global, visit-specific, and a combination of the two. We present the results of Machine Learning models in Table 3. The results indicate that using all available features leads to the most accurate predictions and, consequently, the highest ROC AUC and PR AUC values among all models. Thus, we focus on the use of both sets of features in further discussion. With the LightGBM model, we achieve the best ROC AUC score.

2) PERFORMANCE OF NEURAL NETWORK MODELS

In Table 5, we report the performance across different models to obtain the encoding vector from treatment embeddings in our overall pipeline.

GRU and LSTM process input embeddings sequentially, taking one vector at each step and computing the corresponding functions to use on further timesteps. In the CNN encoder, each convolution layer outputs a vector of fixed dimension. This output dimension corresponds to the number of filters learned by that layer. After several convolutions interleaved with max-pooling layers, we receive an encoding vector, and we transfer it to the next steps in the pipeline. In turn, SWEM does not have learnable parameters. This model produces an encoding vector by simply taking the average of the treatment embeddings. Lastly, in the Transformer model, we encode a sequence of treatment embeddings into another series of vectors of the exact dimensions with the attention mechanism. Subsequently, we average the sequence of new vectors to obtain a final encoding vector. We note that higher-quality results are produced by the Optimized LSTM with hyperparameters optimized for better performance.

We find that *processing treatment embeddings with SWEM yields the best results in our settings*. Meanwhile, the Transformer performs considerably worse, possibly due to an inappropriate architecture for constructing the encoding vector for this particular task. This more complex model fails to capture the data patterns and yields results comparable to those of other models. Obtaining the encoding vector with a simpler model has therefore proven to be beneficial for our problem.

We can observe *better results when we process our global features with linear layers rather than obtaining embeddings*. The reason for this might again be that a more straightforward approach is more beneficial here. We hypothesize that we can better preserve valuable information when we apply some transformations to initial global features, whereas constructing new representations becomes harder.

In some tasks, processing the data from two directions and using both the previous and future contexts might improve quality. To check whether this is the case for our problem, we compare metrics from uni-directional and bi-directional GRU and LSTM models in Table 4. As we can see, under a bi-directional setting, we slightly improve the results of both models.

3) OPTIMIZATION OF HYPERPARAMETERS

The model that performs slightly worse than SWEM but outperforms all of the remaining models and has learnable parameters is the LSTM. We decide to find optimal hyperparameters of our entire pipeline and, especially for the LSTM, to assess any possible improvements in the metrics and whether we manage to beat the performance of the classic Machine Learning model, LightGBM. To optimize the hyperparameters, we use a modern software library for automated hyperparameter search, Optuna ([64]).

We search the optimal values of the 13 parameters, including dropout rates, dimension of treatment embeddings, the learning rate of the optimizer, number of linear layers, and type of activation function between them, among others. We present the results from the optimized model in Table 5. The optimization procedure allows us to achieve metrics that are superior to LightGBM for the same set of features.

[3]https://github.com/fursovia/fraud_detection/tree/2021_update

TABLE 3. Metrics for predicting fraudulent cases with classic machine learning models. Best metrics are highlighted in bold. We present the results for different types of feature. The combined use of global and visit-specific features leads to the best performance.

Model	Used features	ROC AUC	PR AUC	F1-score	Confusion matrix
Logistic Regression	global	0.816	0.078	0.145	[[73968, 1094], [966, 175]]
	visit-specific	0.781	0.077	0.149	[[72924, 2138], [878, 263]]
	both	0.818	0.088	0.164	[[73825, 1237], [928, 213]]
LightGBM	global	0.783	0.063	0.121	[[74248, 814], [1015, 126]]
	visit-specific	0.770	0.151	0.236	[[74097, 965], [859, 282]]
	both	0.863	0.186	0.271	[[74116, 946], [814, 327]]
Optimized LightGBM	global	0.793	0.070	0.132	[[74003, 1059], [986, 155]]
	visit-specific	0.815	0.156	0.244	[[74080, 982], [846, 295]]
	both	**0.881**	**0.187**	**0.274**	[[73571, 1491], [724, 417]]

TABLE 4. Comparison of ROC AUC metrics for uni-directional and bi-directional models.

	GRU	LSTM
Uni-directional	0.874	0.873
Bi-directional	**0.876**	**0.877**

A deep learning model with properly fine-tuned hyperparameters and an optimal architecture is thus a suitable classifier for our problem. Detrimentally, classic Machine Learning models require constructing high-dimensional BoW vectors, which may pose problems with memory and longer training times if the dictionary size of unique tokens is large.

4) ADDRESSING CLASS IMBALANCE PROBLEM

Data in fraud detection problem are inherently imbalanced. We try basic data-level approaches to address this problem in case of using classic machine learning models. In particular, we consider Random Under Sampling, Random Over Sampling, SMOTE and ADASYN. In Random Under Sampling method we remove random samples from the majority class. When using Random Over Sampler, we duplicate random fraudulent examples in the train set. SMOTE and ADASYN generate new synthetic samples of the minority class. We provide the performance of the models trained on the initial set and on the resampled sets in Table 6. The results are given for the optimal ratio of the number of samples in the minority class over the number of samples in the majority class. Some of the approaches significantly enhance the metrics.

As for deep learning models, we incorporate balanced batch sampler in the training procedure. It samples each batch in such a way that the numbers of examples of each class are equal. Balanced batch sampler may facilitate more effective training of neural networks if we compare with the case when we pick samples in the batch randomly. The evidence of balanced batch benefit for some models is observed in Table 6.

5) DEPENDENCE OF MODEL QUALITY ON SAMPLE SIZE

We examine how the size of the training set affects the quality of the model on the test set. We train the model with increasing random subsets of 10, 20, . . ., 100 percent of the initial training data. In Figure 5, we see that the PR AUC and ROC AUC continue to increase as we increase the proportion of used training data. Having limited training data is more detrimental for LightGBM than for the LSTM model. This fact should be taken into account when we are in a regime of small data. We also see that the results are stable for different datasets used. Thus, we can conclude that our deep learning methods and LightGBM will provide satisfactory quality for different subsets of our data.

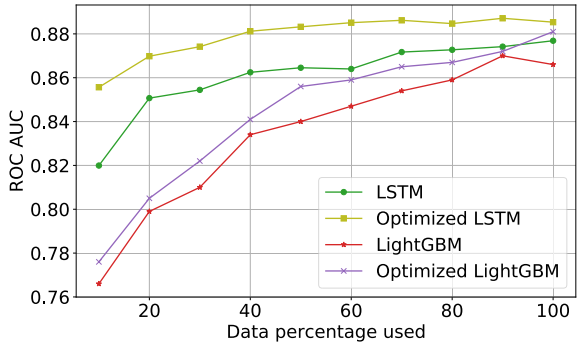


FIGURE 5. Dependence of the ROC AUC on the size of training data. An increase in the proportion of training data leads to a further increase in the quality of the fraud detection model.

6) DEPENDENCE OF METRICS ON EMBEDDING AND ENCODING DIMENSIONS IN LSTM MODEL

To understand the dependency of metrics on the dimensions of treatment embeddings and the encoding vector obtained from them for the LSTM model, we conduct two experiments.

First, we fix the size of the embeddings and increase only the dimension of the encoding vector. We depict the result in Figure 6a. After reaching some optimal value of about 100, a subsequent increase of the encoding vector dimension leads only to a deterioration in performance.

Second, we show that we can improve our performance if we augment the dimension of the embedding vector along with the size of the encoding vector. The evolution of metrics when we simultaneously change the embeddings and encoding sizes and set them to one value is visible in Figure 6b.

Mapping from the initial feature space to an embedded space of small dimension results in information loss and consequently an increase in unsatisfactory model performance.

TABLE 5. Metrics when predicting with neural network models. An asterisk * indicates that we construct the embeddings for global features. Otherwise, we process global features with linear layers. We mark the best values in **bold**. The second best values are underlined.

Model	Used features	ROC AUC	PR AUC	F1-score	Confusion matrix
CNN	visit-specific	0.859	0.115	0.196	[[73066, 1996], [800, 341]]
	both	0.872	0.124	0.205	[[72705, 2357], [742, 399]]
CNN*	visit-specific	0.855	0.137	0.208	[[73486, 1576], [825, 316]]
	both	0.856	0.096	0.179	[[72860, 2202], [813, 328]]
GRU	visit-specific	0.868	0.124	0.203	[[72826, 2236], [759, 382]]
	both	0.876	0.130	0.208	[[73084, 1978], [779, 362]]
GRU*	visit-specific	0.876	0.133	**0.222**	[[72979, 2083], [738, 403]]
	both	0.863	0.119	0.206	[[73589, 1473], [841, 300]]
LSTM	visit-specific	0.868	0.120	0.206	[[73130, 1932], [789, 352]]
	both	0.877	0.127	0.208	[[73185, 1877], [791, 350]]
LSTM*	visit-specific	0.873	0.126	0.219	[[73429, 1633], [800, 341]]
	both	0.862	0.111	0.200	[[72933, 2129], [777, 364]]
SWEM	visit-specific	0.858	0.109	0.192	[[73008, 2054], [801, 340]]
	both	0.884	**0.139**	0.221	[[72782, 2280], [717, 424]]
SWEM*	visit-specific	0.844	0.105	0.186	[[73514, 1548], [866, 275]]
	both	0.836	0.084	0.160	[[73289, 1773], [889, 252]]
Transformer	visit-specific	0.742	0.042	0.030	[[0, 75062], [0, 1141]]
	both	0.824	0.091	0.146	[[73063, 1999], [893, 248]]
Transformer*	visit-specific	0.727	0.042	0.030	[[0, 75062], [0, 1141]]
	both	0.793	0.059	0.090	[[65825, 9237], [652, 489]]
Optimized LSTM	both	**0.891**	0.136	0.209	[[72797, 2265], [744, 397]]

TABLE 6. Comparison of metrics when models are trained on the initial set or on the resampled ones. For classic machine learning models four basic resampling approaches are considered. For deep learning models we show how the use of balanced batch sampler changes the models performance.

Model	Used method	ROC AUC	PR AUC	F1-score	Confusion matrix
Logistic Regression	No Resampling	0.818	0.088	0.164	[[73825, 1237], [928, 213]]
	Random Under Sampling	0.847	0.092	0.164	[[73261, 1801], [878, 263]]
	Random Over Sampling	0.872	0.115	0.196	[[73198, 1864], [814, 327]]
	SMOTE	0.865	0.114	0.201	[[73469, 1593], [836, 305]]
	ADASYN	0.865	0.113	0.197	[[73507, 1555], [847, 294]]
Optimized LightGBM	No Resampling	0.881	0.187	0.274	[[73571, 1491], [724, 417]]
	Random Under Sampling	**0.893**	0.155	0.245	[[73631, 1431], [782, 359]]
	Random Over Sampling	0.879	0.192	0.270	[[74102, 960], [813, 328]]
	SMOTE	0.888	0.200	0.274	[[74243, 819], [830, 311]]
	ADASYN	0.890	**0.202**	**0.279**	[[73943, 1119], [774, 367]]
SWEM	No Resampling	0.867	0.118	0.190	[[73119, 1943], [817, 324]]
	Balanced Batch Sampler	0.884	0.139	0.221	[[72782, 2280], [717, 424]]
LSTM	No Resampling	0.880	0.128	0.190	[[73119, 1943], [817, 324]]
	Balanced Batch Sampler	0.877	0.127	0.208	[[73185, 1877], [791, 350]]

7) RELIABILITY OF MODELS

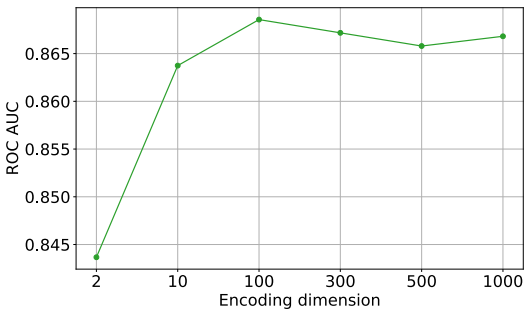
Two significant issues for machine learning models are reliability and resistance to malicious attacks. A relevant challenge in fraud detection is when malicious users of a decision model can provide slightly distorted data to the system and fool it. In such a case, the method is of limited use. We can see examples of this in [65] and [66] and surveys in [67] and [68].

We consider two approaches to evaluate the reliability of our models. The first assesses whether "a model is robust concerning random errors in data submitted to a system", [69]. The second verifies whether "a model is robust to malicious efforts when someone tries to break the system in a particular way by corrupting the input to the system," [65].

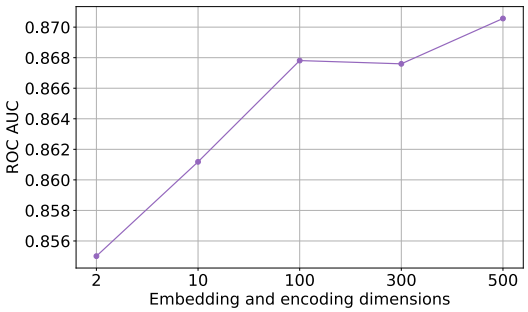
In our case, we test the reliability of the history of treatments. We ask ourselves: Can we change this slightly and obtain an entirely different outcome with the model? To do so, we compare the quality of the model before and after corrupting the test data.

We test these issues in two ways. First, to test the model's reliability, we randomly add a different number of treatments from the vocabulary to the end of the sequence of treatments for each patient. Second, to test the model's resistance to malicious attacks, we select a subset of 100 treatments from the vocabulary and add them one by one to the patients' treatment history to find the model output most affected by the addition of a single treatment. After selecting the most harmful treatment from our subset, we repeat the same procedure to choose the second most malicious treatment. This represents a greedy approach to performing an adversarial attack on our data, [26].

In Figure 7, we show the ROC AUC values after corrupting the sequences of treatments with a varying number of tokens. We see that changing inputs to the trained model leads to a drop in its quality, particularly if we use the greedy strategy to attack the LightGBM model. At the same time, we can observe that the LSTM model is more stable to data corruption and shows only a slight drop in quality even after adding a substantial number of malicious tokens.



(a) Dependence of the ROC AUC metric on the dimension of the encoding vector obtained after processing the treatment embeddings.



(b) Dependence of the ROC AUC metric on simultaneously increasing the dimensions of the embeddings of treatment and the encoding vector.

FIGURE 6. Dependency of metrics on embeddings and encoding dimensions for LSTM model. A higher amount of dimensions leads to less information loss, and therefore, better results.

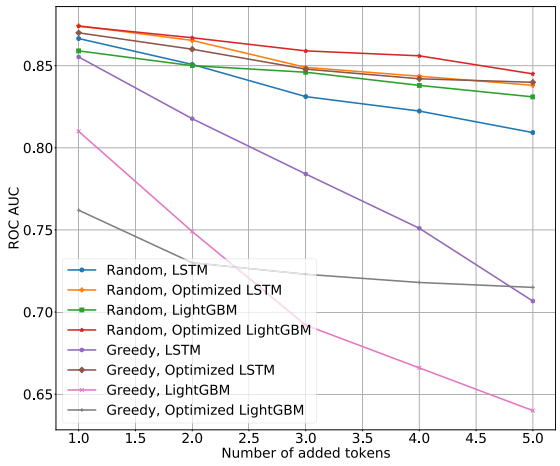


FIGURE 7. Dependence of ROC AUC value on the number of added malicious tokens for different models. The LSTM neural network is more robust to malicious corruption of input data by the greedy approach than is LightGBM.

However, to make the models more robust, we should augment the training data with more cases and possible distortions of the initial data, a so-called data augmentation, and keep the model undisclosed to avoid malicious attacks.

VII. CONCLUSION

A constant challenge for insurers and financial companies, insurance fraud is, in essence, an anomaly detection problem. In this paper, we propose and examine deep learning architectures that are tailor-made for insurance claims data based on embeddings for unstructured data and compare them with classic machine learning approaches based on careful feature engineering. During the model training we also construct embeddings for treatment IDs.

We analyze the performance of classical machine learning models and our proposed methods in solving the task of claim classification for a real-world data. Processing unstructured categorical sequences related to outpatient doctor visits with our best model, we get ROC AUC score equal to 0.873, whereas state-of-the-art model shows a worse result with ROC AUC 0.815. Moreover, our empirical experiments confirm that we can improve our model further by optimizing the neural network architecture, increasing the volume of data used for training and incorporating techniques for addressing class imbalance problem. The significance of choosing the proper embedding and encoding dimensions in our deep learning models is also demonstrated. In addition, we identify that our architecture is robust to random disturbances of the data, as well as adversarial and malicious changes, and can enhance the claims management process. If we add 5 malicious tokens in a sequence, classic machine learning performance degrades to ROC AUC 0.640, while the deep learning model has small performance degradation with ROC AUC value for corrupted input 0.840.

As digitization continues to proliferate, increasing amounts of unstructured data in the form of text will become available, including electronic health records, claims data, personnel files, and financial statements. These data will often have a unique structure and contain variables with many categories that classical methods cannot handle. The deep learning architectures and embeddings that we propose in this paper are bespoke for such data. As a result, our approach is relevant to researchers, administrators and mangers in healthcare, organizational economics, insurance and other fields.

APPENDIX
ADDITIONAL TECHNICAL RESULTS
A. CROSS-VALIDATION FOR PERFORMANCE EVALUATION

For a more credible assessment of the generalization ability of the models, it is helpful to evaluate the performance not on a single train-test split but several sets repeatedly. A popular procedure for that is called cross-validation. The main idea behind cross-validation is that we split our initial data into k sets (folds). Then, we train the model on $k-1$ folds and test them on the remaining part.

We calculate the model metrics on a single train-test split. To understand whether we get biased results, we evaluate several models with a cross-validation procedure. We split our dataset into three folds to preserve the testing to training sizes ratio as for a single train-test split. In Table 7, we provide

TABLE 7. Comparison of metrics with a single train-test split and a 3-fold cross-validation procedure. We use both features. The results from these two approaches are close.

Model	Metric	Train-test split	3-fold cross-validation
Log Reg	ROC AUC	0.818	0.817 ± 0.004
	PR AUC	0.088	0.090 ± 0.004
	F1-score	0.164	0.168 ± 0.004
Opt LightGMB	ROC AUC	0.881	0.876 ± 0.004
	PR AUC	0.187	0.179 ± 0.005
	F1-score	0.274	0.265 ± 0.008
Opt LSTM	ROC AUC	0.891	0.890 ± 0.002
	PR AUC	0.136	0.132 ± 0.009
	F1-score	0.209	0.218 ± 0.010

TABLE 8. Comparison of training duration for different models.

Model	Training duration, s
Logistic Regression	9.84
LightGBM	15.27
Optimized LightGBM	13.66
CNN	121.54
GRU	107.65
LSTM	129.09
SWEM	353.28
Transformer	511.99
Optimized LSTM	149.66

the comparison of results calculated with a single train-test split and 3-fold cross-validation.

As we observe, almost all metrics evaluated on a single split lie within the standard deviation of the mean values that we calculate with the cross-validation procedure. Therefore, we can conclude that the size of our dataset is large enough to get a reliable estimation of the models' performance by implementing a single train-test split.

B. EXPERIMENT DETAILS

We perform the experiments with a single NVIDIA TITAN RTX GPU with 24 GB memory and Intel(R) Xeon(R) CPU E5-2698 v4 @ 2.20GHz. All Deep Learning models were trained using GPU. During the training process of Machine Learning models only CPU was utilized. The overall size of our data is about 240 MB. We launch all the experiments on the system with Linux Mint 19.3 (Tricia) OS.

The training time for different models is given in Table 8. We provide the averaged result of five runs. The running times are acceptable in all cases given typical requirements for the model training in industry. Moreover, they allow to run hyperparameter optimization and utilization of large sample sizes.

ACKNOWLEDGMENT

(Ivan Fursov, Elizaveta Kovtun, and Rodrigo Rivera-Castro are co-first authors.)

REFERENCES

[1] X. Zhou, S. Cheng, M. Zhu, C. Guo, S. Zhou, P. Xu, Z. Xue, and W. Zhang, "A state of the art survey of data mining-based fraud detection and credit scoring," in *Proc. MATEC Web Conf.*, vol. 189, 2018, p. 03002.

[2] A. Artemov, E. Burnaev, and A. Lokot, "Nonparametric decomposition of quasi-periodic time series for change-point detection," *Proc. SPIE*, vol. 9875, pp. 418–422, Dec. 2015.

[3] V. Ishimtsev, A. Bernstein, E. Burnaev, and I. Nazarov, "Conformal k-NN anomaly detector for univariate data streams," in *Proc. 6th Workshop Conformal Probabilistic Predict. Appl.*, in Proceedings of Machine Learning Research, vol. 60, A. Gammerman, V. Vovk, Z. Luo, and H. Papadopoulos, Eds., Stockholm, Sweden, Jun. 2017, pp. 213–227.

[4] A. Artemov and E. Burnaev, "Detecting performance degradation of software-intensive systems in the presence of trends and long-range dependence," in *Proc. IEEE 16th Int. Conf. Data Mining Workshops (ICDMW)*, Dec. 2016, pp. 29–36.

[5] D. Smolyakov, N. Sviridenko, E. Burikov, and E. Burnaev, "Anomaly pattern recognition with privileged information for sensor fault detection," in *Artificial Neural Networks in Pattern Recognition*, L. Pancioni, F. Schwenker, and E. Trentin, Eds. Cham, Switzerland: Springer, 2018, pp. 320–332.

[6] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 15:1–15:58, Jul. 2009.

[7] J. Jurgovsky, M. Granitzer, K. Ziegler, S. Calabretto, P.-E. Portier, L. He-Guelton, and O. Caelen, "Sequence classification for credit-card fraud detection," *Expert Syst. Appl.*, vol. 100, pp. 234–245, Jun. 2018.

[8] D. Huang, D. Mu, L. Yang, and X. Cai, "CoDetect: Financial fraud detection with anomaly feature detection," *IEEE Access*, vol. 6, pp. 19161–19174, 2018.

[9] C. Phua, V. Lee, K. Smith, and R. Gayler, "A comprehensive survey of data mining-based fraud detection research," 2010, *arXiv:1009.6119*.

[10] M. Kirlidog and C. Asuk, "A fraud detection approach with data mining in health insurance," *Proc.-Soc. Behav. Sci.*, vol. 62, pp. 989–994, Oct. 2012.

[11] Y. Wang and W. Xu, "Leveraging deep learning with LDA-based text analytics to detect automobile insurance fraud," *Decis. Support Syst.*, vol. 105, pp. 87–95, Jan. 2018.

[12] J. Kim, H.-J. Kim, and H. Kim, "Fraud detection for job placement using hierarchical clusters-based deep neural networks," *Appl. Intell.*, vol. 49, pp. 1–20, Feb. 2019.

[13] M. V. Balasubramanian, "Ensemble modeling & prediction interpretability for insurance fraud claims classification," Ph.D. dissertation, Dublin Bus. School, Dublin, Ireland, 2019.

[14] M. Sun, F. Tang, J. Yi, F. Wang, and J. Zhou, "Identify susceptible locations in medical records via adversarial attacks on deep predictive models," 2018, *arXiv:1802.04822*.

[15] C. Gomes, Z. Jin, and H. Yang, "Insurance fraud detection with unsupervised deep learning," *J. Risk Insurance*, vol. 88, no. 3, pp. 591–624, Sep. 2021.

[16] R. Y. Gupta, S. S. Mudigonda, and P. K. Baruah, "A comparative study of using various machine learning and deep learning-based fraud detection models for universal health coverage schemes," *Int. J. Eng. Trends Technol.*, vol. 69, no. 3, pp. 96–102, Mar. 2021.

[17] T. Chen, L.-A. Tang, Y. Sun, Z. Chen, and K. Zhang, "Entity embedding-based anomaly detection for heterogeneous categorical events," 2016, *arXiv:1608.07502*.

[18] R. Hu, C. C. Aggarwal, S. Ma, and J. Huai, "An embedding approach to anomaly detection," in *Proc. IEEE 32nd Int. Conf. Data Eng. (ICDE)*, May 2016, pp. 385–396.

[19] A. Rajaraman and J. D. Ullman, *Mining of Massive Datasets*. Cambridge, U.K.: Cambridge Univ. Press, 2011.

[20] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.

[21] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1532–1543.

[22] S. Xiao, J. Yan, M. Farajtabar, L. Song, X. Yang, and H. Zha, "Learning time series associated event sequences with recurrent point process networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 10, pp. 3124–3136, Oct. 2019.

[23] V. Zhuzhel, R. Rivera-Castro, N. Kaploukhaya, L. Mironova, A. Zaytsev, and E. Burnaev, "COHORTNEY: Deep clustering for heterogeneous event sequences," 2021, *arXiv:2104.01440*.

[24] D. Babaev, M. Savchenko, A. Tuzhilin, and D. Umerenkov, "E.T.-RNN: Applying deep learning to credit loan applications," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 2183–2190.

[25] D. Babaev, I. Kireev, N. Ovsov, M. Ivanova, G. Gusev, and A. Tuzhilin, "Event sequence metric learning," 2020, *arXiv:2002.08232*.

[26] I. Fursov, M. Morozov, N. Kaploukhaya, E. Kovtun, R. Rivera-Castro, G. Gusev, D. Babaev, I. Kireev, A. Zaytsev, and E. Burnaev, "Adversarial attacks on deep models for financial transaction records," 2021, *arXiv:2106.08361*.

[27] S. Arora, Y. Liang, and T. Ma, "A simple but tough-to-beat baseline for sentence embeddings," in *Proc. ICLR*, 2016, pp. 1–16.

[28] Y. Wang, H. Huang, C. Feng, Q. Zhou, J. Gu, and X. Gao, "CSE: Conceptual sentence embeddings based on attention model," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, vol. 1, 2016, pp. 505–515.

[29] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler, "Skip-thought vectors," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 3294–3302.

[30] B. Krawczyk, "Learning from imbalanced data: Open challenges and future directions," *Prog. Artif. Intell.*, vol. 5, no. 4, pp. 221–232, Nov. 2016.

[31] E. Duman, A. Buyukkaya, and I. Elikucuk, "A novel and successful credit card fraud detection system implemented in a Turkish bank," in *Proc. IEEE 13th Int. Conf. Data Mining Workshops*, Dec. 2013, pp. 162–171.

[32] P. Erofeev, E. Burnaev, and A. Papanov, "Influence of resampling on accuracy of imbalanced classification," *Proc. SPIE*, vol. 9875, pp. 9875-1–9875-5, Dec. 2015.

[33] Y. Sahin, S. Bulkan, and E. Duman, "A cost-sensitive decision tree approach for fraud detection," *Expert Syst. Appl.*, vol. 40, no. 15, pp. 5916–5923, 2013.

[34] K. R. Seeja and M. Zareapoor, "FraudMiner: A novel credit card fraud detection model based on frequent itemset mining," *Sci. World J.*, vol. 2014, Sep. 2014, Art. no. 252797.

[35] N. Kozlovskaia and A. Zaytsev, "Deep ensembles for imbalanced classification," in *Proc. 16th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2017, pp. 908–913.

[36] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, no. 1, pp. 321–357, Jan. 2002.

[37] J. A. Sáez, J. Luengo, J. Stefanowski, and F. Herrera, "SMOTE–IPF: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering," *Inf. Sci.*, vol. 291, pp. 184–203, Jan. 2015.

[38] D. Smolyakov, A. Korotin, P. Erofeev, A. Papanov, and E. Burnaev, "Meta-learning for resampling recommendation systems," *Proc. SPIE*, vol. 11041, Mar. 2019, Art. no. 110411S.

[39] A. Singh and A. Jain, "Adaptive credit card fraud detection techniques based on feature selection method," in *Advances in Computer Communication and Computational Sciences*. Singapore: Springer, 2019, pp. 167–178.

[40] R. Saia and S. Carta, "Evaluating the benefits of using proactive transformed-domain-based techniques in fraud detection tasks," *Future Gener. Comput. Syst.*, vol. 93, pp. 18–32, Apr. 2019.

[41] J. M. Johnson and T. M. Khoshgoftaar, "Survey on deep learning with class imbalance," *J. Big Data*, vol. 6, no. 1, pp. 1–54, Dec. 2019.

[42] S. Makridakis, "The forthcoming artificial intelligence (AI) revolution: Its impact on society and firms," *Futures*, vol. 90, pp. 46–60, Jun. 2017.

[43] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, and G. Chen, "Deep speech 2: End-to-end speech recognition in English and Mandarin," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 173–182.

[44] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *IEEE Comput. Intell. Mag.*, vol. 13, no. 3, pp. 55–75, Aug. 2018.

[45] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," 2017, *arXiv:1709.05584*.

[46] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, *arXiv:1301.3781*.

[47] A. Drozd, A. Gladkova, and S. Matsuoka, "Word embeddings, analogies, and machine learning: Beyond king-man+woman=queen," in *Proc. 26th Int. Conf. Comput. Linguistics, Tech. Papers (COLING)*, 2016, pp. 3519–3530.

[48] S. Bartunov, D. Kondrashkin, A. Osokin, and D. Vetrov, "Breaking sticks and ambiguities with adaptive skip-gram," in *Proc. Artif. Intell. Statist.*, 2016, pp. 130–138.

[49] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *J. Mach. Learn. Res.*, vol. 12, pp. 2493–2537, Aug. 2011.

[50] W. Y. Zou, R. Socher, D. Cer, and C. D. Manning, "Bilingual word embeddings for phrase-based machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2013, pp. 1393–1398.

[51] X. Ma and E. Hovy, "End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, 2016, pp. 1064–1074.

[52] S. V. N. Vishwanathan, N. N. Schraudolph, I. R. Kondor, and K. M. Borgwardt, "Graph kernels," *J. Mach. Learn. Res.*, vol. 11, pp. 1201–1242, Mar. 2010.

[53] D. Haussler, "Convolution kernels on discrete structures," Univ. California Santa Cruz, Santa Cruz, CA, USA, Tech. Rep. UCS-CRL-99-10, 1999.

[54] P. Yanardag and S. V. N. Vishwanathan, "Deep graph kernels," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Sydney, NSW, Australia, Aug. 2015, pp. 1365–1374.

[55] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *Proc. 33nd Int. Conf. Mach. Learn. (ICML)*, New York, NY, USA, Jun. 2016, pp. 2014–2023, 2016.

[56] S. Ivanov and E. Burnaev, "Anonymous walk embeddings," in *Proc. 35th Int. Conf. Mach. Learn.*, in Proceedings of Machine Learning Research, vol. 80, J. Dy and A. Krause, Eds., 2018, pp. 2186–2195.

[57] S. Ivanov, N. Durasov, and E. Burnaev, "Learning node embeddings for influence set completion," in *Proc. IEEE Int. Conf. Data Mining Workshops (ICDMW)*, Nov. 2018, pp. 1034–1037.

[58] H. Farbmacher, L. Loew, and M. Spindler, "An explainable attention network for fraud detection in claims management," *J. Econometrics*, 2020.

[59] M. A. Montemurro, "Beyond the Zipf–Mandelbrot law in quantitative linguistics," *Phys. A, Stat. Mech. Appl.*, vol. 300, nos. 3–4, pp. 567–578, Nov. 2001.

[60] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 3133–3181, 2014.

[61] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "LightGBM: A highly efficient gradient boosting decision tree," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 3146–3154.

[62] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[63] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Jan. 2011.

[64] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," 2019, *arXiv:1907.10902*.

[65] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, Apr. 2017, pp. 506–519.

[66] I. Fursov, A. Zaytsev, N. Kluchnikov, A. Kravchenko, and E. Burnaev, "Differentiable language model adversarial attacks on categorical sequence classifiers," 2020, *arXiv:2006.11078*.

[67] W. E. Zhang, Q. Z. Sheng, A. Alhazmi, and C. Li, "Adversarial attacks on deep learning models in natural language processing: A survey," 2019, *arXiv:1901.06796*.

[68] W. Wang, B. Tang, R. Wang, L. Wang, and A. Ye, "A survey on adversarial attacks and defenses in text," 2019, *arXiv:1902.07285*.

[69] A. Saltelli, S. Tarantola, F. Campolongo, and M. Ratto, *Sensitivity Analysis in Practice: A Guide to Assessing Scientific Models*. Hoboken, NJ, USA: Wiley, 2004.

IVAN FURSOV was born in Chelyabinsk, Russia. He received the master's degree from the Skolkovo Institute of Science and Technology, in 2020. In the master's thesis, he proposed a new adversarial attack on sequence classifiers. He is currently a Deep Learning Research Engineer at Tinkoff and continues to work on new approaches in adversarial attacks on NLP models.

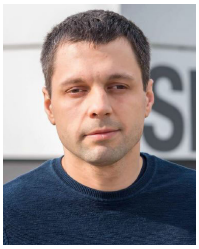
ELIZAVETA KOVTUN was born in Moscow, Russia, in 1999. She received the B.Sc. degree in applied mathematics and physics from the Moscow Institute of Physics and Technology, Moscow, in 2020. She is currently pursuing the M.Sc. degree in information science and technology with the Skolkovo Institute of Science and Technology, Moscow. Since 2020, she has been a member of the Laboratory of Applied Research Skoltech-Sberbank. Her research interest includes the application of deep learning in industry.

RODRIGO RIVERA-CASTRO is currently a Research Engineer with the Skolkovo Institute of Science and Technology. He is also pursuing his doctoral research in the area of unsupervised methods for learning event sequences and time series. His research interests include extracting rich representations from temporal sequences to enable downstream tasks and power applications in finance, marketing, supply chain, and the web.

ALEXEY ZAYTSEV was born in Kharkiv, Ukraine. He received the graduate degree from MIPT, in 2012, and the Ph.D. degree in mathematics from IITP RAS, in 2017. In his master's thesis, he proposed a modification of the Bayesian approach for linear regression that allows an automated feature selection. He is currently an Assistant Professor at Skoltech. His research interests include the development of new methods for sequential data, Bayesian optimization, and embeddings for weakly structured data.

RASUL KHASYANOV received the M.Sc. degree from the Skolkovo Institute of Science and Technology, in 2019. His research interest includes application of machine learning in industry.

MARTIN SPINDLER received the Ph.D. degree in economics and mathematics from the University of Munich, in 2012. He was Visiting Scholar at the Boston College and MIT. He is a Professor of statistics and data science at the University of Hamburg. His research interests include machine learning, deep learning, and applications to business-related and economic problems.



EVGENY BURNAEV received the M.Sc. degree from the Moscow Institute of Physics and Technology, in 2006, and the Ph.D. degree from the Institute for Information Transmission Problem, in 2008. He is currently an Associate Professor with the Skolkovo Institute of Science and Technology, Moscow, Russia. His research interests include Gaussian processes for multi-fidelity surrogate modeling and optimization, deep learning for 3-D data analysis, manifold learning, online learning for prediction, and anomaly detection.