Towards Explainable AI: Design and Development for Explanation of Machine Learning Predictions for a Patient Readmittance Medical Application

Sofia Meacham[1], Georgia Isaac[1], Detlef Nauck[2(✉)], and Botond Virginas[2]

[1] Bournemouth University, Poole, UK
[2] British Telecom, Adastral Park, Ipswich, UK
detlef.nauck@bt.com

Abstract. The need for explainability of AI algorithms has been identified in the literature for some time now. However, recently became even more important due to new data protection act rules (GDPR 2018) and due to the requirements for wider applicability of AI to several application areas. BT's autonomics team has recognized this through several sources and identified the vitality of AI algorithms explainability to ensure their adoption and commercialization. In this paper, we designed and developed a system providing explanations for a prediction of patient readmittance using machine learning. The requirements and the evaluation were set by BT through their projects with real-customers in the medical domain. A logistic regression machine learning algorithm was implemented with explainability "hooks" embedded in its code and the corresponding interfaces to the users of the system were implemented through a web interface. Python-based technologies were utilized for the implementation of the algorithm (Scikit-learn) and the web interface (web2py), and the system was evaluated through thorough testing and feedback. Initial trade-off analysis of such an approach that presents the overhead introduced by adding explainability versus the benefits was performed. Lastly, conclusions and future work are presented, considering experimentation with more algorithms and application of software engineering methods such as abstraction to the aid of explainable AI, leading further along to "explainability by design".

Keywords: Explainable AI · Machine learning · Scikit-learn · Human-centered computing · Human–computer interaction

1 Introduction

This paper discusses the design and implementation of a medical-domain system for prediction of diabetic patient readmittance from the viewpoint of 'Explainable AI', a concept used to provide explanations and understandability of the corresponding machine learning algorithm's decisions. The dataset used to develop said system was provided by the UCI Machine Learning Repository [1, 2], first collected and donated by Strack et al. [3]. The dataset details diabetic patient admission encounters and

K. Arai et al. (Eds.): CompCom 2019, AISC 997, pp. 939–955, 2019.
https://doi.org/10.1007/978-3-030-22871-2_67

encourages the prediction of the binary attribute 'Readmitted' as either 'readmitted' or 'not readmitted'. The client for this research was BT's research labs at Adastral Park, Ipswich, UK. BT initiated the need for explainability in their machine learning algorithms and holds several clients in the health and other application sectors.

From extensive background study detailed in Sect. 2, it has been recognized that algorithms of the future, such as AI and classification/prediction algorithms, will need to explain themselves in order to ensure their adoption. Also, different types of algorithms are amenable to different approaches that need to be investigated.

In this paper, we took the bottom-up approach as regards to the explainability problem by starting from a specific case study that includes a specific classification algorithm (logistic regression) and attempts to make it explainable to users ranging from machine learning experts to domain experts (medical professionals in this case).

A systems engineering approach was taken to design the system at high-level of abstraction prior to detailed implementation. Specifically, UML semi-formal notations such as Class diagrams were used to define the system structure and functionality. Detailed implementation that consisted in integrating the web2py web framework with data science algorithms taken from the Scikit learn library followed. The developed web interface environment enabled experimentation with explainability aspects of the algorithm with promising results. It was observed that if "hooks" are positioned in specific points of the algorithm presenting relevant information to the user following would significantly improve explainability without compromising the overall algorithm performance.

The remainder of the paper will cover in Sect. 2.1 a background study on 'Explainable AI' research and in Sect. 2.2 explainability for several types of classification algorithms. In Sect. 3, the case study of an explainable algorithm for the medical domain is detailed. Specifically, in Sect. 3.1 the system requirements are presented, in Sect. 3.2 the system architecture and in Sects. 3.3 and 3.4 the design and implementation of the algorithm with the viewpoint of explainability are provided. Section 4 presents the evaluation of the proposed approach. Specifically, in Sect. 4.1 the classifier accuracy is detailed, in Sect. 4.2 the functional correctness is presented, in Sect. 4.3 the overhead to performance due to explainability is details and in Sect. 4.4 the client-machine learning expert feedback is presented. Section 5 offers conclusion and suggestions for future work and research directions.

2 Background Study: Explainable AI (XAI)

2.1 Explainable AI

'Explainable AI' (XAI) is a concept established in 2016 by DARPA, defining the practice of improving understandability, trustability, and manageability of emerging artificial intelligence systems [4]. Launchbury describes the evolution of artificial intelligence, starting with describing; initially, artificial intelligence was able to provide a description of data using sets of logic rules to represent knowledge in limited domains, though had no learning capability, only data perception and reasoning. Next, AI evolved to have the ability to predict and classify big data through training and

testing of statistical models, however this is limited by its minimal ability to provide reasoning, leading to improvements in AI perception and learning, but little in terms of abstraction and reasoning. Launchbury introduces XAI as the next step in AI evolution; a stage in which developers construct explanatory models for use in real world situations, with the hope of facilitating natural communication between machines and people. In this evolution, systems can learn and provide reasoning, also improving perception and abstraction [5].

With the nature of machine learning classification algorithms being black-box and mostly uninterpretable, the integration of explainability through XAI into these algorithms should be imperative, especially in high-risk situations where the algorithm's result must be trusted enough to avoid repercussions. It was determined that two systems of XAI implementation can be used, being; ante-hoc and post hoc. Ante-hoc systems encompass machine learning algorithms that already hold a level of interpretability through 'glass-box' approaches, such as the decision tree or linear regression algorithms [6]. Post hoc systems alternatively provide explanations for uninterpretable, black-box systems such as neural networks, where their workings and decisions are undeterminable without the intervention of a separate tool such as LIME (Local Interpretable Model-Agnostic Explanations) [6, 7]. In both ante-hoc and post hoc systems, explainability is used to enable and inform domain experts to make their own decisions based on the outcomes of machine learning algorithms.

2.2 Types of Classification Algorithms and Their Explainability

Implementation of a classification system of any type requires thorough investigation into varying machine learning algorithms to ensure that the correct and most accurate algorithm is chosen for the task. In this case, as the system must not only classify but also explain and provide reasoning, there are further implications which must also be considered during investigation.

One such implication is the interpretability of the algorithm – or the ease at which the mapping of an output to descriptors can be understood by a user. These descriptors could be used to form the basis of explanations for the algorithm's decision [6]. A notable understanding though is that usually the more complex an algorithm, the less interpretable it is, even though the more complex algorithms tend to be more accurate [8]. This introduces a trade-off and begs the question: Do we require an incredibly accurate algorithm with limited to no explanations, or a reasonably accurate algorithm with sufficient explanations? PwC define a gap analysis approach for understanding and working with this trade-off. An algorithm's ability to explain and an organisation's readiness are measured on scales from low to high. A required level along with a current level are also defined on each scale. This method of analysis proves useful in determining which aspects of machine learning the organization should focus on. For instance, if they are comfortably past the required level of explainability, then this should indicate that they are able to trade-off some of their model explainability for accuracy instead [8]. In a business setting, a method such as gap analysis should be considered when working with minimizing negative impacts of the explainability-accuracy trade-off.

Beyond the main categorization of classification algorithms in ante-hoc and post hoc explainability types, current attempts in literature focus on reviewing (from the explainability perspective) the following commonly used algorithms:

Decision Tree Algorithms. The decision tree algorithm is considered one of the simplest and most interpretable white-box machine learning algorithms available, as it can be compared to the human decision-making process. Compared to the performance of a logistic regression model, it can be seen to underperform slightly, however due to its simplicity it can be used to provide helpful visual explanations for its results [9]. Though simple and interpretable, the decision tree algorithm is particularly susceptible to noisy data, for instance if two instances of data have the same values for some attributes but differing classification results and is also prone to overfitting.

Neural Networks. These are black-box algorithms, known to provide good accuracy but limited to no interpretability [10]. Neural networks require much lengthier training times in comparison to simpler models, owing to their complex nature. They contain considerably more connections and subtle properties when it comes to interactions between attributes. Mixing this with the workings of hidden layers alone being notoriously difficult to decipher, the overall interpretability of the algorithm suffers [8]. Due to the complexity of neural networks, extensive amounts of data are usually required to train a reliable model.

Logistic Regression. Similarly to decision trees, logistic regression is considered a simple, white-box algorithm which can provide interpretable explanations of results. Its interpretability stems from the ability to easily extract and understand the workings of the algorithm, for instance, through its coefficients [11]. Using these, the relationship between each attribute and the result can be visualized, providing an initial step towards explainability.

3 Case Study: Explainable Logistic Regression Algorithm for the Medical Domain

3.1 System Requirements

The system was designed and developed as a project at Bournemouth University in collaboration with an industrial partner, BT. BT's autonomics team provided the requirements for this project as they face the explainability problem in several of their projects with real clients. As part of their attempts to commercialize their autonomic algorithms and generally their classification algorithms adoption, explainability of the algorithm's decision making have arose several times. A medical application domain prediction algorithm was chosen for this project due to the criticality of the risk taken through classification decisions for these types of applications.

The main objective of the system was to predict patient readmission from the dataset suggested by the client and provided by UCI, and to give explanations for the given predictions.

As mentioned in the background study, there are two types of explainability in literature: ante-hoc where the explainability is inserted between the steps of the algorithm and post hoc where the algorithm is so complicated that it can't be stopped without serious compromising and explainability is inserted at the end of the algorithm, usually through applying external tools.

For this case study, an ante-hoc system with the aim of providing explainability of machine learning algorithm decisions for a medical application was designed and developed. Several technologies can be used towards this system with most cases incorporating interface design and web technologies. Interface design and web technologies are the most common approaches used to enable the communication between machines and users forming the backbone of any human–computer interaction. As explainability and specifically to domain experts in our case lies in this human–computer interaction point, it is apparent that interface design and web technologies should be incorporated.

The implementation decisions for interface design and web technologies were focused on the consistent use of Python as the backbone of the development. As Python is a strong language when it comes to compute-intensive tasks and is considered a leading language for machine learning [12], and the Scikit-learn library is both widely used and one of the top-performing machine learning libraries in terms of time efficiency [12], both technologies were chosen. Previous works have identified the ability of the Python web framework Web2py to work effectively with the Python machine learning library Scikit-learn [13], therefore this was the primary web technology chosen to develop the system.

The system described in this paper utilises a logistic regression algorithm in order to classify patients as regards to their probability of readmission. Several algorithms were evaluated from their capability to solve the case study problem and their explainability. The decision tree algorithms, however, simple and interpretable they maybe, were not chosen due to their susceptibility to noise data and proneness to overfitting. The neural networks are mostly categorized to post hoc explainability due to their complexity and their requirement for enormous amount of data. The logistic regression was the best choice as in terms of the medical dataset as enough data exists to train the logistic regression model efficiently, and resulting probabilities provide effective insight into how changes in their medical situation, for instance the number of inpatient procedures they have had, could affect their likelihood of being readmitted. Logistic regression also provides improved interpretability in comparison to more complex algorithms such as neural networks.

3.2 System Architecture

The high-level proposed system architecture is depicted in Fig. 1.

In this figure, the Model View Controller architectural pattern followed by web2py web framework is the center of the web interface design. In this model, the data stored in the model are manipulated by the functions of the Controller and are being viewed by the user.

In our proposed system architecture, this MVC model is enriched and extended with data science algorithms from the Scikit learn library which are interacting at two

points: as part of the controller where the classification algorithm is being called to operate on data; as part of the model where we have integrated an existing data set and with the additional data stored/retrieved by the user.

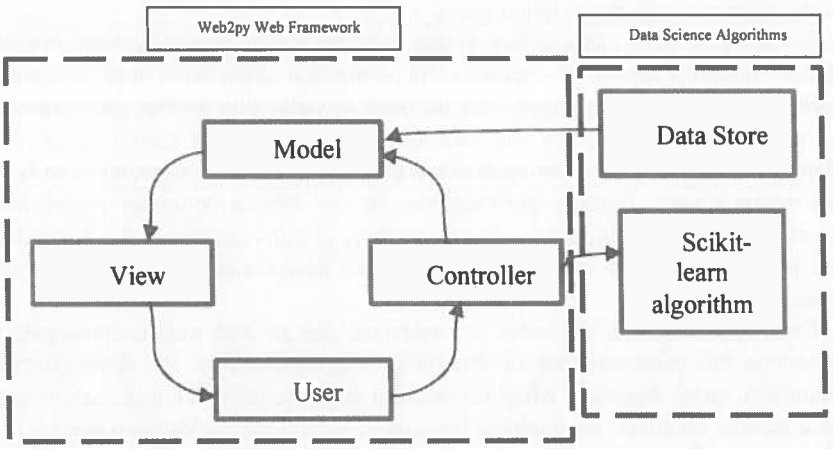

Fig. 1. System architecture diagram

This architecture is reproducible and can be applied to create web interfaces that use data science algorithms in several application contexts from explainable AI to intelligent interfaces.

3.3 Explainable Algorithm Ante-Hoc Design

The class diagram in Fig. 2 details the architecture of the classification system.

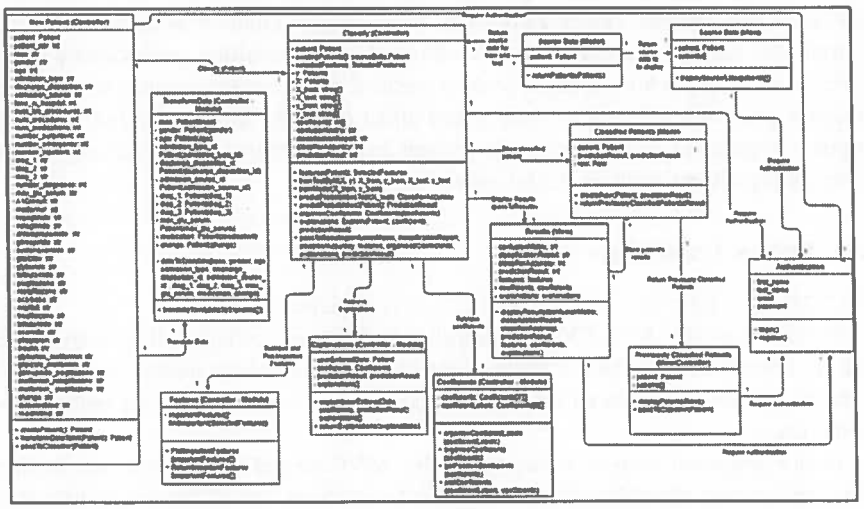

Fig. 2. UML class diagram for classification algorithm

Classification using the *Classify class* itself requires the user to be logged in, defined by the *Authentication class*, and a new patient to be created, defined by the *New Patient class*. Once the patient is created, the patient data is transformed to ensure all data points are integers, defined by the *Transform Data class*.

The classifier is then built by the *Classify class* using data returned by the *Source Data (model) class*, also displayed on a web page, defined by the *Source Data (view) class*. With a built classifier and transformed data, the *Classifier class* can classify the new patient.

During classification, a confusion matrix and classification report are produced. The *Features class* is used to find, plot, and return the important features. As well as this, logistic regression coefficients are retrieved, and both the label and value of each positive coefficient are retrieved, plotted, and returned, as defined by the Coefficients class.

Once classified, both the results and coefficients are used to determine the relevant explanation for each attribute, as defined by the *Explanations class*.

Finally, the patient's data and results are inserted into the database, defined by the *Classified Patients (Model)* all of which, when requested, can be viewed on the web page defined by the *Previously Classified Patients (View Controller) class*. The patient's classification results, along with the confusion matrix, classification report, plotted feature importances, plotted coefficients, and explanations for each coefficient are displayed on the results page, as defined by the Results class.

The following sections detail all major points where explainability was added and presented to the user and the corresponding interface will be depicted in figures.

3.4 Explainable Algorithm Ante-Hoc Implementation

Development of the machine learning classifier was carried out following the CRISP-DM methodology [14]. Both the data understanding and data preparation stages were carried out to ensure insufficiencies in the dataset, such as noise, missing data, and incorrect datatypes, were resolved. These stages led to a reduction in dataset size from 100,000 instances to 69,374.

With a prepared dataset, a logistic regression model was developed in Python using the Scikit-learn library and was chosen based on both its interpretability and suitability to the dataset. Due to its interpretability, it was possible to implement explainability using the ante-hoc method, therefore useful information could be extracted from the classifier wherever it was required. For instance, once the classifier was trained and tested using 70% and 30% of the dataset respectively, the following pieces of information were extracted:

- Probabilities of both Readmitted and Not readmitted
- A confusion matrix representation of model performance
- Positive coefficients determined by the model

These pieces of information could be extracted as and when required, owing to the ante-hoc implementation method used, allowing the creation of an "explainable" interface.

Displaying the Classifier's Results and Accuracy Metrics. As the final user's (domain expert's - doctors in this case) understanding of the overall result was paramount and considering interface design research, a progress bar-styled visualisation was used to display the results. As Wainer suggests, to display data accurately it is important to choose a method which clearly demonstrates both the order and magnitude of the resulting numbers [15]. Figure 3 demonstrates that this rule was followed, with each segment of the bar representing the magnitude of both the 'Readmitted' and 'Not Readmitted' results.

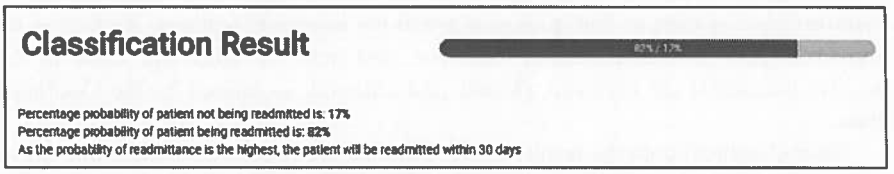

Fig. 3. Classification result progress bar

Additionally, the user is presented with a section detailing the classifier's accuracy. This provides an overall accuracy score achieved by the classifier when comparing classified test data against train data, and a confusion matrix to visualise the classifier's correct and incorrect predictions. This section can be seen in Fig. 4. This provides information explaining the performance of the algorithm and it refers to machine learning experts rather than the final users.

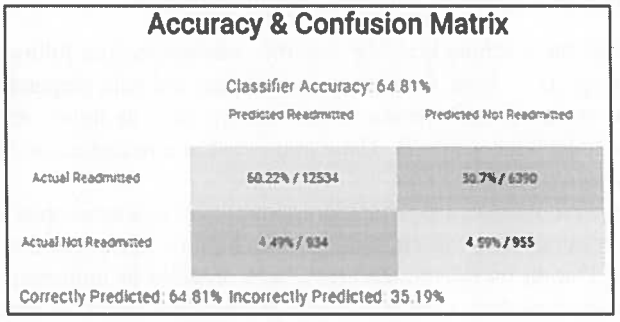

Fig. 4. Confusion matrix for logistic regression classifier, as displayed on results interface

Understanding a classifier's performance is crucial when identifying methods of improvement requiring a suitable visualisation method, a simple and common one being the confusion matrix [11]. The confusion matrix provides the data needed to calculate various metrics; from a simple accuracy metric, to others such as precision, recall, and accuracy [16]. From the client's perspective, inclusion of the confusion matrix provides the user with a basic accuracy measurement, useful for users with limited machine learning experience, and more detailed metrics if required.

The confusion matrix in Fig. 4 demonstrates that the classifier is weak at predicting actual 'Not Readmitted' patients as 'Not Readmitted', and commonly predicts 'Readmitted' patients as 'Not Readmitted', therefore only receives an overall accuracy of $\sim$64%. Reasoning for this is further detailed in Sect. 4.1.

Next, a graph displaying feature importance is presented to the user and produced using a Decision Tree algorithm to demonstrate a method for providing feature information. This graph can be seen in Fig. 5.

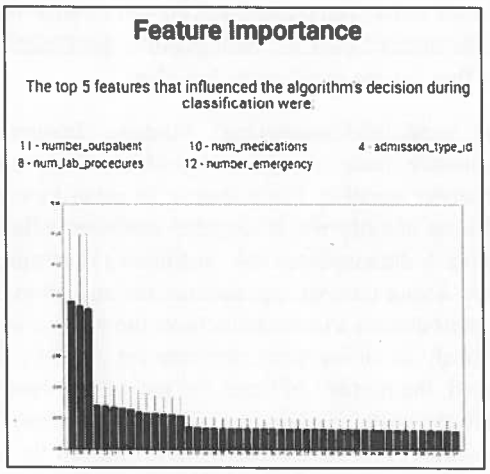

Fig. 5. Dataset feature importance, as displayed on results interface

Scikit learn's *ExtraTreesClassifier* was used to calculate a list of important features based on the impact each attribute from the dataset has on the overall prediction result. A similar activity was carried out by Saabas involving interpretability improvement of random forest classifications through the development of a new Python library, *treeinterpreter*. In his study, he extracted a figure for each attribute defining its contribution to the result, both for individual predictions and for the classifier as a whole [17]. Although produced through the *ExtraTreesClassifier* library instead of *treeinterpreter* a similar result is received, with an ordered list of each attribute's contribution to the overall classifier's result outputted. These results were plotted on a bar chart for easy visualisation of attribute contributions as shown in Fig. 5. This provided a level of explanation for the model as a whole, as the user can determine which features could have affected the results in what way, however it does not focus on specific explanations or reasoning for individual classifications.

Implementing Explainability for Individual Classifications. As the logistic regression algorithm uses coefficients to determine the influence of attributes on the result likelihoods, these coefficients must be retrieved and utilised to implement explainability for individual classifications. The coefficients were first extracted using the line of code: *coefficients = X.columns, np.transpose(model.coef_)*. Next, a

coefficientsHelper module was developed which enabled the coefficients' attribute labels and coefficients to be received separately. It also contained a function used to retrieve only positive coefficients from the model, allowing the attributes which would most affect a likelihood of Readmitted to be easily identified and presented to the user. The *coefficientsHelper* module also contained a function which, using the retrieved coefficients' attribute labels and coefficients, would produce a bar chart plotted using the Matplotlib library.

Combining these functionalities, it was possible to develop a new module which would consider the coefficients and the user-entered patient data for each attribute. This module, *explainClassification*, determined sets of informative phrases, one of which, depending on the user-entered data for each positive coefficient, would be displayed upon hovering over bars on the coefficients bar chart.

Implementing the 'explainClassification' Module. Implementation of the *explainClassification* module made use of the algorithm's domain decisions, but also of domain knowledge where possible. For instance, in using its own knowledge on the dataset the algorithm can identify which supplied attributes influence the classification result and how. Figure 6 demonstrates the coefficients determined by the algorithm, plotted on a bar chart. Using this, we can see that the algorithm believes that some of the most influential attributes on a decision include; the number of times the patient has been admitted previously as an inpatient, whether the patient's diabetes medication/s has/have been changed, the number of times the patient has been admitted previously as an emergency, and the value of their last glucose serum test. The algorithm determines whether a patient will be readmitted or not based on the values of these coefficients. For instance, if a patient has had more inpatient visits than emergency and had an elevated glucose serum test result (>200 mg/dL or >8.59 A1C), the probability that they would be readmitted in the near future would increase. These most influential attributes are passed through to the *explainClassification* module, once the overall result is determined, in order to decide which set of statements inside the module to use to provide correct explanation statements. The code used to make this decision is shown in Fig. 6.

```
if notReadmitted > readmitted:
        resultStr = "As the probability of non-readmittance is the highest,
                     the patient will not be readmitted within 30 days"
        result = "Not readmitted within 30 days"
        explain = explainClassification.explainNotReadmitted(positiveCoefficients)

    else:
        resultStr = "As the probability of readmittance is the highest,
                     the patient will be readmitted within 30 days"
        result = "Readmitted within 30 days"
        explain = explainClassification.explainReadmitted(positiveCoefficients)
```

Fig. 6. Decision statements determining how to use *explainClassification* module to retrieve correct explanation statements

For initial implementation of explainability, simple statements such as "The probability of readmission increases as the patient's glucose serum levels were >200" were provided for each coefficient with the greatest impact on classification result. To further improve these explanations a degree of domain knowledge was also utilised, enabling the provision of explanations which considered the magnitude of change in the classification's result based on the change in data of the positive coefficients. The idea that one unit of change in one attribute will not have the same influence on the classification result as one unit of change in another attribute should be evident, as displayed by the coefficients bar chart, however this could not be integrated into the current explanations without developing the *explainClassification* module's decision statements. These statements were based on observational experimentation carried out by classifying a patient multiple times, each time slightly changing the value provided for one attribute at a time in order to determine its actual influence on the resulting probabilities. As an example, a patient who had neutral values for all positive coefficients provided a 26% probability of being readmitted. If the number of inpatient procedures is increased from 0 to 1, the probability of readmission increases to 32.7%, meaning a singular unit of change for this attribute is 6.7%. In comparison, if the number of inpatient procedures is reduced back to 0 and the number of emergency procedures is increased from 0 to 1, the probability of readmission is 28.8%, therefore a singular unit of change for this attribute causes a 2.8% increase in probability of readmission. The decision statements rely on boundary values determined by the unit changes for each positive coefficient attribute. When values for each positive coefficient fall into a boundary, a suitable statement is provided to the user. For instance, since we know that one unit of change in the number of inpatient procedures has a considerable effect on the classification result, we can determine that if the patient has 0 inpatient procedures, the classification result would not have been affected. However, if the patient has between 1 and 5 inpatient procedures, the readmission probability would have increased slightly, and inpatient procedures greater than 5 would have significantly affected the readmission probability. Similar statements were generated for each attribute with a positive coefficient. An example decision statement used to determine an explanation for a result of Readmitted using a positive coefficient attribute is shown in Fig. 7 below.

JQuery was then used to implement hotspots over each positive coefficient's bar on the bar chart. Upon hover of these hotspots, the statement related to the chosen attribute and the user-entered data for that attribute would display in and information box, providing the user with an explanation on that attribute's effect on the overall probabilities. Figure 8 demonstrates this functionality on a patient who is more likely to be readmitted.

The *explainClassification* module produced results with which the client was pleased, as during the final feedback stage they mentioned that it was easy for them to understand the classifier's result and how the data they entered affected it. Further in its success in providing understandable explanations, a less detailed yet more textual result was achieved when compared to explanations given by the SimMachines tool [18]. SimMachines classifies data and outputs a pie chart allowing the user to select each

```
if userEnteredCoef11 == "> 300" or userEnteredCoef11 == "> 200":

    response = "The patient's glucose serum test result was elevated
               at %s, therefore likelihood of readmission increases
               slightly." % userEnteredcoef11
    explain.append(response)

elif userEnteredCoef11 == "None" or userEnteredCoef11 == "Normal" :

    response = "The patient's glucose serum test result was %s,
               therefore this is unlikely to have affected the
               likelihood of readmission." % userEnteredCoef11
    explain.append(response)

else:
    response = " "
    explain.append(response)
```

Fig. 7. Decision statements within *explainClassification* used to determine correct explanation statements for a Readmitted result with the *max_glucose_serum* attribute

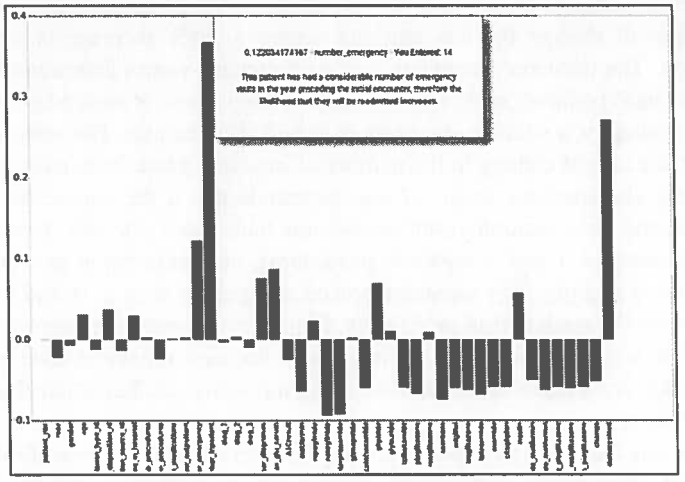

Fig. 8. Bar chart displaying logistic regression coefficients, providing explanations for classification result

segment of interest. This presents each piece of entered data which led to the chosen segment's result. For instance, the first split of the pie chart determines the overall binary result of the classification, the next provides a feature and its value that led to the binary result, and so on. Instead of focusing on each possible result as SimMachines does, the developed system only focuses on the final result and each of the important features. This allowed for the textual explanations behind the users' entered data and its effect on the overall result.

4 Evaluation of the Proposed Approach

4.1 Classifier Accuracy

According to Fig. 4 in Sect. 3.4, the classifier achieves an accuracy of 64.81%. It was identified that this inaccuracy could be due to dataset imbalance, as the confusion matrix was based on the 30% test split of the dataset, however as the emphasis of this research work was explainability of the machine learning algorithm and the implementation of the system was limited to a project timeframe, the accuracy of the classifier and imbalanced dataset were not prioritised, and a $\sim$64% accuracy was deemed satisfactory.

The ROC Curve generated by a classification attempt, shown below in Fig. 9, demonstrates that the classifier is OK at separating readmitted patients from non-readmitted patients. A much more successful classifier would result in a curve following the left-hand border of the graph, indicating a higher true positive rate and more accurate results overall. Similarly, a graph with a larger area under the ROC curve would also indicate a more accurate classifier.

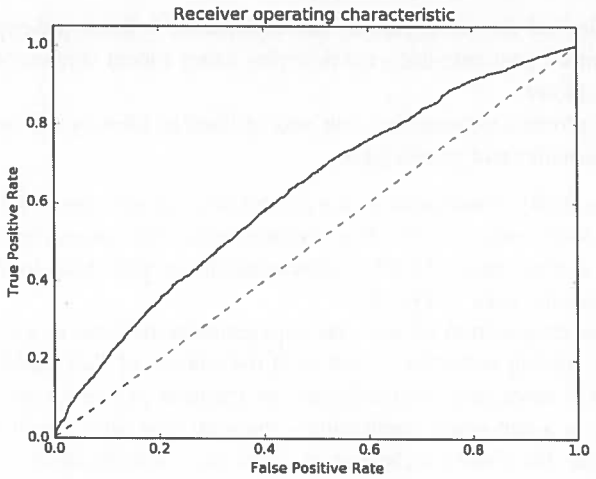

Fig. 9. ROC (Receiver Operating Characteristic) curve plotting the TPR (True Positive Rate) against the FPR (False Positive Rate) at differing thresholds

Again, the inaccuracy of the classifier is most likely due to the unbalanced nature of the dataset, however a satisfactory result in terms of the project's requirements was met.

4.2 Functional Correctness

System evaluation was performed as regards to functional correctness with functional test cases, which covered all aspects of the produced system, including login and registration, patient classification, previously classified patients and source data. From a total of 43 tests only 40 were executable due to reasons such as unimplemented or

partly implemented features. Of the 40 executed tests the system achieved a success rate of 100% (Table 1).

Table 1. Record of test execution and success rates

Executed Tests	40/43 93%	Unexecuted Tests	3/43 7%
Passed Tests (of executed)	40/40 100%	Failed Tests (of executed)	0/40 0%

4.3 Explainability Overhead Experimentation

During implementation of explainability features into the logistic regression classifier, it was identified that a significant overhead exists pre-explainability upon classification using a local version of the system. An experiment into said overhead was undertaken, during which the following was attempted:

- Classification of the same patient was conducted 5 times pre-explainability, and another 5 times post-ante-hoc-explainability using a local version of the system and Internet Explorer.
- Internet Explorer's network console was utilised to identify the loading speeds of both the classifier and results page.

Pre-explainability, classification of a patient took an average of 4 s, resulting in an average total load time of 4.5 s. Post-explainability, the same classification actions resulted in an average time of 11.6 s, increasing the average total load time to 12.6 s. These results can be seen in Fig. 10.

Overall, the introduction of ante-hoc explainability resulted in a 176% increase of overhead. This finding would be crucial as, if the amount of data used for classification exceeded 70,000 rows, or more predictors are required to make a prediction – which are both likely in a real-world application – the total load time would increase further.

It is possible that chosen technologies could have influenced this increase in total load time. For the sake of machine learning systems in risky domains requiring explainability, this observed overhead is a consequence which must be understood and prepared for though careful optimization of both chosen technologies and code if timely results are relied on.

4.4 Client Feedback

Throughout the development of the system, the client (BT Adastral park, data science team) was consulted on a regular basis and expected to provide two types of feedback; written progress feedback throughout the lifetime of the project, and a final survey covering all system functions upon completion. Feedback throughout the project's

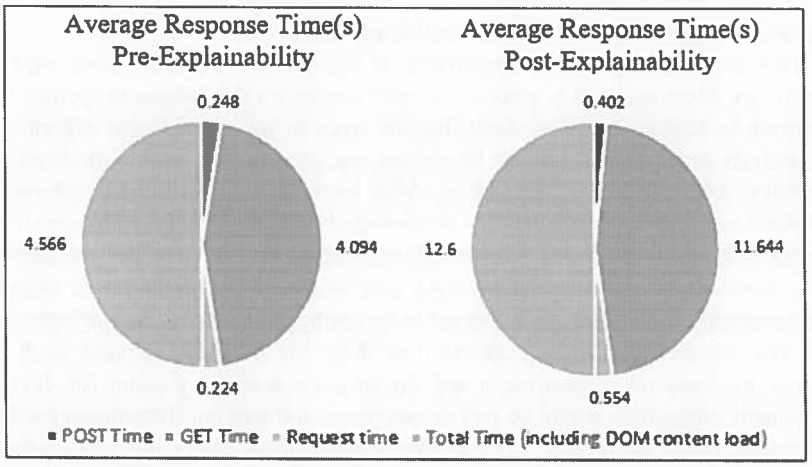

Fig. 10. Pre-explainability and post-explainability average timings

lifetime influenced both changes and completion of the system's features and played an integral role in the evaluation of the system.

Final feedback received from the client was very positive and took the form of an online survey created using MySurveyLab [19]. The client was asked to carry out several tasks which would invoke each of the system's features. Most importantly, they were able to understand certain classification and explainability aspects, such as; classifier performance and accuracy, feature importance, and explanations provided using the coefficients bar chart. Though the client had the ability to provide reasoning for their answers, they showed comfort in their understanding of explanations through only selecting "Yes" where asked whether they understood the above aspects. It can be assumed that, if the client was unsure about what was displayed to them, they would have selected the "Somewhat" option, and provided extra written feedback.

More work is required to take this research forward and accumulate feedback from the final users which is this case are the medical domain experts-doctors.

5 Conclusions and Future Work

In this paper, the design and development of web system used to perform classification and explanation of patient readmittance using machine learning has been explored and detailed. Technologies used in both design and development stages are discussed, with the aim of providing the reader with enough information to reproduce this system architecture in their own applications.

The resulting web system successfully provided a facility with which patients can be classified as either 'will be readmitted' or 'will not be readmitted' with satisfactory accuracy for the purposes of this research which focused explainability of the algorithm rather than its performance. More importantly, the system was able to provide the user

with enough explanation with which an understanding of how changing user inputs – patient attributes – can affect the classification result.

This paper was only the beginning of the journey towards more explainable algorithms. More research is required in order to explore algorithmic properties that are amenable to explainability by extending the work to include different algorithms. As the problem is huge and difficult to address, we plan to start with classification-type algorithms and ante-hoc explainability. After working with several algorithms, trends and patterns to extract explainability are expected to emerge. We plan to use a bottom-up approach through working with several algorithms first and generalizing afterwards. Also, well-formed software engineering practices such as model-based design and domain-specific modelling are expected to be applicable and play an important role in this area. Identifying and appropriate describing the different domains such as the medical professional users-doctors and the machine-learning domain for developing explainable algorithms would be part of our research direction. Bringing explainability to the design level using the above methods will enable the final target of Explainability by design for the future trustworthy, accountable and explainable AI systems.

References

1. Dua, D., Karra Taniskidou, E.: UCI Machine Learning Repository. University of California, School of Information and Computer Science, Irvine, CA (2017). http://archive.ics.uci.edu/ml
2. Lichman, M.: UCI Machine Learning Repository. University of California, School of Information and Computer Science, Irvine, CA (2013). http://archive.ics.uci.edu/ml
3. Strack, B., DeShazo, J., Gennings, C., Olmo, J., Ventura, S., Cios, K., Clore, J.: Impact of HbA1c measurement on hospital readmission rates: analysis of 70,000 clinical database patient records. BioMed. Res. Int. **2014**, 1–11 (2014). http://dx.doi.org/10.1155/2014/781670
4. DARPA, Information Innovation Office. Broad Agency Announcement, Explainable Artificial Intelligence (XAI). DARPA, Arlington, VA (2016)
5. Launchburry, J.: A DARPA Perspective on Artificial Intelligence (2017)
6. Holzinger, A., Biemann, C., Pattichis, C., Kell, D.: What do we need to build explainable AI systems for the medical domain? (2017)
7. Ribeiro, M., Singh, S., Guestrin, C.: "Why should I trust you?": explaining the predictions of any classifier. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations (2016). http://aclweb.org/anthology/N16-3020
8. Oxborough, C., Birchall, A., Cameron, E., Townsend, A., Rao, A., Westermann, C.: Explainable AI: driving business value through greater understanding (2018). https://www.pwc.co.uk/audit-assurance/assets/explainable-ai.pdf. Last accessed 27 June 2018
9. Rudd, J., Priestley, J.A.: Comparison of decision tree with logistic regression model for prediction of worst non-financial payment status in commercial credit. Grey Literature from Ph.D. Candidates. 5 (2017). https://digitalcommons.kennesaw.edu/dataphdgreylit/5
10. Gunning, D.: Explainable Artificial Intelligence (XAI) (2016). https://www.darpa.mil/attachments/XAIIndustryDay_Final.pptx. Last accessed 5 July 2018
11. Kuhn, M., Johnson, K. Applied Predictive Modelling. Springer, New York (2013). https://doi.org/10.1007/978-1-4614-6849-3_12

12. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, É.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. (2011). http://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf. Last accessed 5 July 2018
13. Isaac, G., Meacham, S., Hamzeh, H., Stefanidis, A., Phalp, K.: An adaptive E-commerce application using web framework technology and machine learning. In: BCS Software Quality Management (SQM) Conference 2018, London, UK, 26–27 March 2018
14. Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., Wirth, R.: CRISP-DM 1.0, p. 12. SPSS Inc., Chicago (2000) https://the-modeling-agency.com/crisp-dm.pdf. Last accessed 28 June 2018
15. Wainer, H.: How to display data badly. Am. Stat. **38**(2), 137 (1984). https://doi.org/10.1080/00031305.1984.10483186
16. Fawcett, T.: An introduction to ROC analysis. Pattern Recogn. Lett. **27**(8), 861–874 (2006). https://doi.org/10.1016/j.patrec.2005.10.010
17. Saabas, A.: Random forest interpretation with scikit-learn|Diving into data (2015). Blog. datadive.net
18. SimMachines.: Visualisation of TV data and Porsche purchase prediction [image] (2017). https://simmachines.com/focus-areas/media/demo-3/. Last accessed 2 July 2018
19. MySurveyLab.: MySurveyLab.com. 7 Points Ltd., Warsaw, Poland (2018)