TALLINN UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology
Department of Software Science

Artur-Aleksander Pärnoja 193348 IAIB

APPLICATION FOR THE ANALYSIS OF DRAWING AND WRITING TESTS

Bachelor's thesis

Supervisors: Sven Nõmm, PhD
Elli Valla, MSc

Tallinn 2022

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Artur-Aleksander Pärnoja 193348 IAIB

RAKENDUS JOONISTAMISE JA KIRJUTAMISE TESTIDE ANALÜÜSI JAOKS

Bakalaureusetöö

Juhendajad: Sven Nõmm, PhD
Elli Valla, MSc

Tallinn 2022

Author's declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Artur-Aleksander Pärnoja

06.05.2022

Abstract

The present thesis is devoted to developing the application for machine-learning-based analysis of the drawing and writing tests used in computer-aided diagnostics of Parkinson's disease. In recent years, machine-learning-based analysis of drawing and writing tests has resulted in highly accurate models supporting the diagnostics of Parkinson's disease and other neurodegenerative disorders. Also, this approach gained popularity in related areas such as psychiatry and ergonimics of the work environment. Nevertheless, to the author's best knowledge, there is no proper application allowing medical practitioners to use these results. This situation has resulted in the gap between research achievements on the one academic side and the inability to apply them to everyday medical practice.

To overcome this gap, the author of the thesis proposed an application developed for the Windows OS which allows a practitioner to choose a previously trained machine learning model and apply it to the given test. The graphical user interface allows the practitioner to observe the drawing itself and evaluate the kinematic, pressure, and geometric features describing the test. Then chosen Machine-Learning model may be applied to estimate the presence of the symptoms of Parkinson's disease. As a final step, the application invokes model agnostic description explainers LIME and SHAP, which provide the information on which features were most important in making classification predictions.

The proposed application is developed using the PyQt5 framework. It requires one to submit the testing data and use the library of pre-trained models produced by the research group in TalTech.

It was recently tested by two practitioners whose feedback was positive. They also provide feedback about the developments they would expect from the following versions of the application.

Annotatsioon

Käesolev lõputöö on pühendatud Parkinsoni tõve diagnostikas kasutatavate joonistus- ja kirjutamistestide analüüsi rakenduse väljatöötamisele. Viimastel aastatel on joonistus- ja kirjutamistestide masinõppepõhine analüüs andnud tulemuseks väga täpsed mudelid, mis toetavad Parkinsoni tõve ja teiste neurodegeneratiivsete häirete diagnostikat. Samuti saavutas see lähenemine populaarsuse seotud valdkondades, nagu psühhiaatria ja töökeskkonna ergonoomika. Sellegipoolest ei ole autorile teadaolevalt rakendust, mis võimaldaks arstidel neid tulemusi kasutada. Selline olukord on toonud kaasa lõhe ühelt poolt teadussaavutuste ja suutmatuse vahel neid igapäevases meditsiinipraktikas rakendada.

Selle lünga ületamiseks pakkus lõputöö autor välja Windows OS-i jaoks arendatud rakenduse, mis võimaldab arstidele valida eelnevalt treenitud masinõppe mudeli ja rakendada seda antud testis. Graafiline kasutajaliides võimaldab arstil jälgida testi joonist ning hinnata testi kirjeldavaid kinemaatilisi, surve- ja geomeetrilisi tunnuseid. Seejärel võib Parkinsoni tõve sümptoomide esinemise hindamiseks rakendada valitud masinõppe mudelit. Viimase sammuna kutsub rakendus välja mudeliagnostilised selgitajad LIME ja SHAP, mis annavad teavet selle kohta, millised tussused olid klassifitseerimisennustuste tegemisel kõige olulisemad.

Käesolev rakendus on välja töötatud PyQt5 raamistiku abil. Selle kasutamiseks tuleb laadida testimisandmed ja kasutada TalTechi uurimisrühma poolt koostatud eeltreenitud mudelite kogumi.

Rakendust katsetasid hiljuti kaks arsti (psühhiaater ja neuro-füsioloog), kelle tagasiside oli positiivne. Samuti andsid nad tagasisidet selle kohta, milliseid arenguid nad rakenduse järgmistelt versioonidelt ootaksid.

List of abbreviations

OS	**Operating System**
AI	**Artificial Intelligence**
XAI	**Explainable Artificial Intelligence**
PDF	**Portable Document Format**
PD	**Parkinson's Disease**
HC	**Healthy Control**
JSON	**JavaScript Object Notation**
LIME	**Local Interpretable Model-agnostic Explanation**
SHAP	**SHapley Additive exPlanations**
GUI	**Graphical User Interface**

Contents

Chapter 1

Introduction

Parkinson's disease is a neurological disorder that causes unintended and uncontrollable movements. Parkinson's disease's main symptoms are shaking, stiffness, and difficulty in balance and coordination. The most notable signs of Parkinson's disease occur when nerve cells in the basal ganglia, an area of the brain that controls movement, become impaired and/or die. Currently there is no cure for Parkinson's disease [3].

Rapidly aging population of the European countries causes a growing number of patients suffering from the neurodegenerative diseases, which in turn results in growing pressure on the medical personnel, therefore special monitoring and analyzing system is needed. The main goal of the current thesis is to develop a simple and easy to use application, which will support the work and reduce the pressure on the medical personnel. The analyzing system will speed up the Parkinson's disease diagnostic process, by means of machine learning models and graphical data representation.

The topic of this thesis is in trend of the recent developments in this area [7, 1, 6, 13]. Also, a lot of research were conducted on the application of XAI methods in different medical fields [15, 16, 14], which prove that the XAI can be used for medical purposes. Even though much research was conducted on the topic of the use of machine learning for the diagnosis of Parkinson's disease, no application with graphical user interface and ability to apply machine learning classifiers was provided. The one that could allow to apply various classification models for digital test analysis and process, visualize, and explain the data and results acquired from the analyzed test. All those problems will be solved in the current work.

The application for digital drawing and writing tests analyzing will be done as a standalone Windows desktop application on PyQt5 framework. The application will provide an ability to use classification models library and choose which model to apply on a test. The classification models creation and training are not a part of the current thesis. All models were previously developed by the work group. The data from the digital test will be presented to the user by means of interactive graphs, which will provide a better overview of an acquired information. The main novelty component of this work is a use of Explainable Artificial Intelligence (Explainable AI). The Explainable AI will explain the results provided by the classification model to the application user. The usage of Explainable AI is intended to increase the user's trust level to the machine learning model.

As a result of this work the application for the analysis of digital drawing and writing tests was created. The application is intended to be used for medical purposes and allows the practitioners (medical doctors or nurses) to apply classification models on digital tests and obtain fast and accurate diagnosis. By the means of Explainable AI, the explanation for the diagnosis is provided to the practitioner. The application obtains all necessary functionality for producing understandable and reliable diagnosis and provides the ability to build a full report of conducted analysis and save it to Portable Document Format (PDF).

The rest of the thesis is organised as follows. Section 2 presents the necessary information on PD and the testing techniques used to diagnose it. Formal problem statement and tasks necessary to solve it are presented in Section 3. The proposed solution is described in Section 4. Section 5 presents main results of the thesis. It follows by the Section 6 where limitations and possible improvements are discussed. Conclusions are drawn in the last section.

Chapter 2

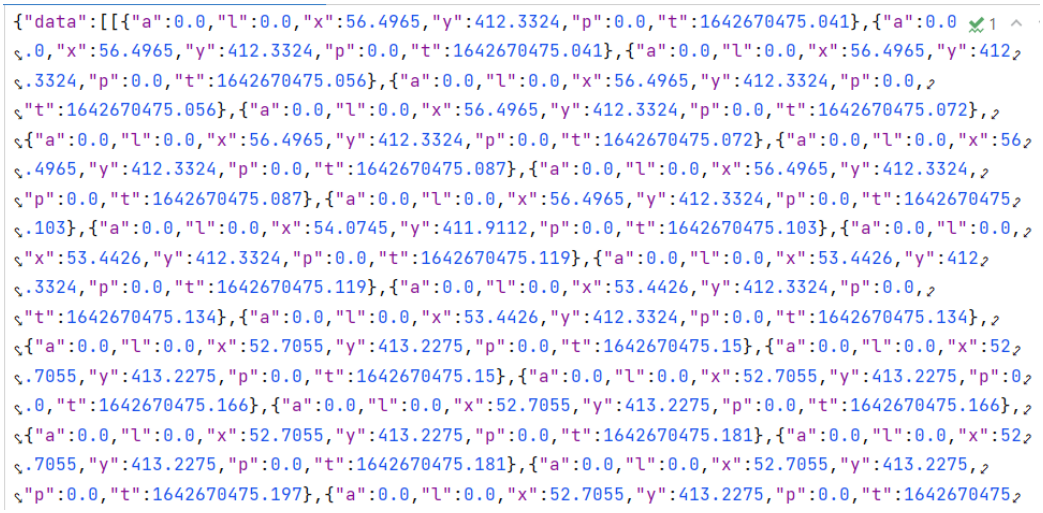
Background

Fine motor tests, such as Luria's alternating series drawing tests, are a very widespread and effective method of Parkinson's disease diagnostics. For a long time, Parkinson's disease drawing and writing tests were conducted with the use of simple pencil and paper. Then, with the evolution of technologies, it became possible to conduct such tests with a tablet computer. Starting with a seminal paper [5, 1] suggested a computational procedure for movement analysis in handwriting. [5] researches made it possible to use machine learning approach for digital drawing and writing test analysis. Later [1] has extended the procedure to include more features describing the kinematics of stylus tip motions.

[7, 13] continued working in this direction and introduced special software for iPad and Surface platforms, which is intended to collect information about patient's handwriting. Unlike [1], where a digital table was used, [13] presented the results based on the tests acquired using Tablet PC. This approach allows the tested subject to see the feedback of their drawing and therefore better mimics original pencil and paper. After that, the vast experiment was conducted, where many participants were involved including people with confirmed Parkinson's disease. As a result of the experiment participants handwriting data was collected. Acquired information then was processed and used for training classification models, which allows diagnosing Parkinson's disease by digital drawing and writing tests.

It is planned that in the future, when current work will be finished, Parkinson's disease diagnostics will begin with a patient doing a digital drawing or

writing test on a tablet computer, which will save the patient's handwriting data to a JavaScript Object Notation (JSON) file. The file contains data about *X-coordinate* (x), *Y-coordinate* (y), *altitude* (l), *azimuth* (a), *pressure* (p) and *time* (t) for every fraction of a second of the test (Figure 2.1).



```
{"data":[[{"a":0.0,"l":0.0,"x":56.4965,"y":412.3324,"p":0.0,"t":1642670475.041},{"a":0.0
.0,"x":56.4965,"y":412.3324,"p":0.0,"t":1642670475.041},{"a":0.0,"l":0.0,"x":56.4965,"y":412
.3324,"p":0.0,"t":1642670475.056},{"a":0.0,"l":0.0,"x":56.4965,"y":412.3324,"p":0.0,
"t":1642670475.056},{"a":0.0,"l":0.0,"x":56.4965,"y":412.3324,"p":0.0,"t":1642670475.072},
{"a":0.0,"l":0.0,"x":56.4965,"y":412.3324,"p":0.0,"t":1642670475.072},{"a":0.0,"l":0.0,"x":56
.4965,"y":412.3324,"p":0.0,"t":1642670475.087},{"a":0.0,"l":0.0,"x":56.4965,"y":412.3324,
"p":0.0,"t":1642670475.087},{"a":0.0,"l":0.0,"x":56.4965,"y":412.3324,"p":0.0,"t":1642670475
.103},{"a":0.0,"l":0.0,"x":54.0745,"y":411.9112,"p":0.0,"t":1642670475.103},{"a":0.0,"l":0.0,
"x":53.4426,"y":412.3324,"p":0.0,"t":1642670475.119},{"a":0.0,"l":0.0,"x":53.4426,"y":412
.3324,"p":0.0,"t":1642670475.119},{"a":0.0,"l":0.0,"x":53.4426,"y":412.3324,"p":0.0,
"t":1642670475.134},{"a":0.0,"l":0.0,"x":53.4426,"y":412.3324,"p":0.0,"t":1642670475.134},
{"a":0.0,"l":0.0,"x":52.7055,"y":413.2275,"p":0.0,"t":1642670475.15},{"a":0.0,"l":0.0,"x":52
.7055,"y":413.2275,"p":0.0,"t":1642670475.15},{"a":0.0,"l":0.0,"x":52.7055,"y":413.2275,"p":0
.0,"t":1642670475.166},{"a":0.0,"l":0.0,"x":52.7055,"y":413.2275,"p":0.0,"t":1642670475.166},
{"a":0.0,"l":0.0,"x":52.7055,"y":413.2275,"p":0.0,"t":1642670475.181},{"a":0.0,"l":0.0,"x":52
.7055,"y":413.2275,"p":0.0,"t":1642670475.181},{"a":0.0,"l":0.0,"x":52.7055,"y":413.2275,
"p":0.0,"t":1642670475.197},{"a":0.0,"l":0.0,"x":52.7055,"y":413.2275,"p":0.0,"t":1642670475
```


Figure 2.1: A part of a JSON file with handwriting data

Then the medical doctor, or nurse will load the obtained file to developed application and choose a classification model to be applied for analysis. The application algorithms will read, process, and analyze the data from the file. As a result, the application user will be provided with handwriting data, graphical representation, diagnosis, and explanation for how the chosen model arrived at this decision. Explanation of model's decisions will be the main novelty of current work and will be implemented by means of LIME and SHAP algorithms. After the analyzing step is finished, the full report could be generated and saved as a PDF file (Figure: 2.2).

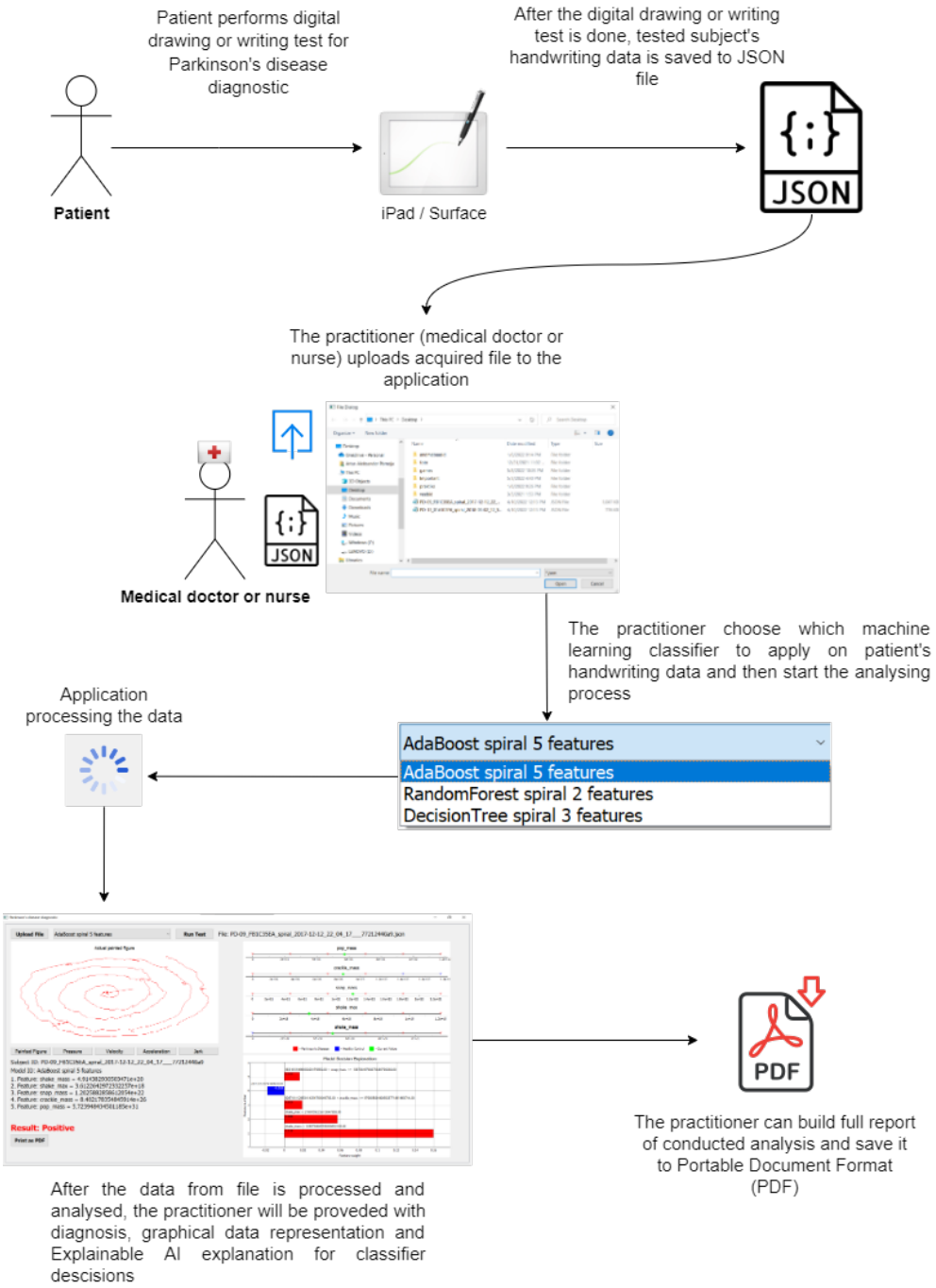


Figure 2.2: Application of drawing tests for Parkinson's disease diagnostics, full workflow

Chapter 3

Problem Statement

A very insufficient number of applications with graphical user interfaces, which allow to apply machine learning models for data analysis and classification, are done. None of these applications applies Explainable Artificial Intelligence for classification model decisions explanation. The lack of data representation and absence of understandable and reliable explanation for acquired results makes existing applications not suitable for use in medical purposes.

3.1 Functional Requirements

The practitioners (medical doctors or nurses) require a simple and easy to use application, which will allow them to conduct Parkinson's disease diagnostics and obtain accurate well-grounded results. The practitioner should be provided with the option to choose between available classification models. After the model is applied the graphical representation of the patient's handwriting data and demonstration of the position of data point along each feature are presented to the practitioner. For model decision explanation the two most known interpreters (LIME and SHAP) should be used. The application user should be able to switch between interpreters after the results are acquired. All graphs along with other provided information could be formatted and converted into the PDF format.

3.2 Non Functional Requirements

The application should be implemented as the stand-alone Windows desktop application, as one of the practitioners' requirements. While the application algorithms perform the analysis of the data, the application user should see loading animation.

Available classification models should have understandable and descriptive names, which contain classification algorithm name (AdaBoost, RandomForest, etc.), test name (Spiral, Sentence, etc.) and the number of features used for training. The classification algorithm in the model's name will tell the user about which algorithm will perform diagnostic. The test name is needed for the practitioner to avoid the situation when the model is applied not on corresponding test. There could be several models provided, which use the same classification algorithm and are intended for the same test, but which were trained on different number of features, which is why the number of features should be used in the model's name.

For representing the limits of the features and demonstration of the data point position along other features the application should automatically display one-, two- or three- dimensional scatter plot. The dimensionality of the graph should be valid according to number of training features of the selected model.

3.3 Main Steps

To achieve the main goal of this thesis next steps should be done:

1. Find framework with tools for work with scientific data and ability to display various graphs (two- and three-dimensional; scatter plots; bar charts; box plots).
2. Build initial (skeleton) application with chosen framework.
3. Add file dialog system for reading JSON data files.
4. Remake existing feature engineering function for working with a single data file and integrate it to the application.

5. Add all needed elements (buttons, text labels, drop-down select, graph sections) and their functionality to the application.

6. Add loading animation and multi-threading logic.

7. Add two- and three-dimensional graphs.

8. Add LIME and SHAP interpreters to the application for explaining machine learning model predictions.

9. Add logic to convert all information (including graphs) into the PDF format.

Chapter 4

Solution

4.1 Framework

To build the stand alone Windows desktop application, *JavaFx*, *PyQt5* and *PyQt6* were chosen as possible frameworks for implementation of the application.

- *JavaFX* is an open source, client application platform for desktop, mobile and embedded systems built on Java language. It provides fully featured toolkit for developing rich client applications [2].
- *PyQt* connects the Qt C++ cross-platform framework with the Python language, it is a GUI module. Qt is more than a GUI toolkit, which is why it features abstractions of network sockets or threads, along with Unicode, SQL, databases, SVG, OpenGL, XML, an operational we browser, a service system and a vast array of GUI widgets [8]. *PyQt5* and *PyQt6* are two different versions of *PyQt*.

For implementation of current application, the *PyQt5* framework was chosen, because there are many scientific libraries and tools (scikit-learn, pandas, NumPy, Matplotlib) provided for Python language. There are much less scientific libraries for Java, and they are more complex in use. In previous studies on this topic the feature engineering method was developed on Python language [13]. This method should be used in current work because it provides DataFrame with features based on which the classification model makes prediction. *PyQt* framework provides tools for showing different types of two- and three- dimensional interactive graphs, meaning user

can scale and move over them using mouse. The interactive graphs give user better overview and are more understandable in comparison with static graphs that *JavaFX* offers. The *PyQt* framework fifth version was chosen over sixth, because it is older and there are fewer bugs than in the newer sixth version, which makes it more reliable. Also, there are more materials and tutorials available for PyQt5 than for PyQt6.

4.2 Skeleton Application

Initial (skeleton) application is the basis for building future application. Usually this is just a group of initial classes and functions, necessary for application work. Skeleton application usually has minimal functionality from back-end side and empty application's main window from front-end side. It will be filled with content and functional logic on future development stages. Building a skeleton application is a crucial step in the development process because it gives first representation and overview of used framework and its capabilities. At this stage of development underlying potential problems can be discovered, which can result in change of a framework. Skeleton application also gives developer an understanding and ability to plan future design of the application.

4.3 File Dialog System

For reading files with patient data, the file dialog system was implemented in the application with use of the *QFileDialog* package. File dialog system allows the application user to choose a file with patient digital drawing or writing test data and then load it to the application with familiar Windows design, within two mouse clicks. (Figure 4.1) It is only possible to choose a file with the *.json* extension from the file dialog window. This is done to prevent situations when a user can accidentally upload the wrong file, as the application only accepts *.json* files. After the data from the file was read, the data processing part can begin.

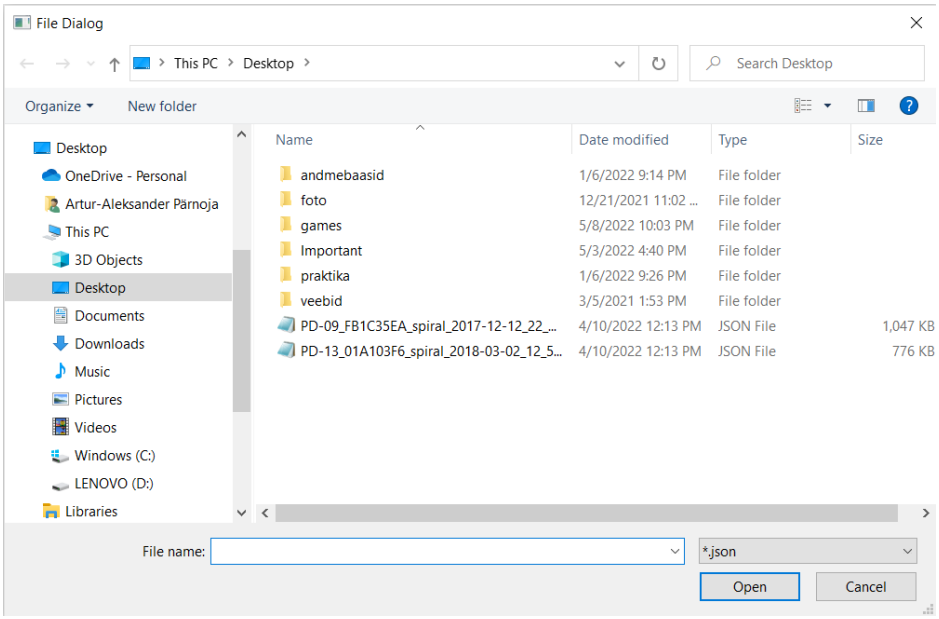


Figure 4.1: File dialog window, for uploading JSON file with patient's handwriting data to the application

4.4 Data Processing Function

There is a feature engineering function, made by [13] in her thesis bachelor, which provides extraction of kinematic and pressure features from raw data provided by JSON files. The best features obtained by using this function were used to train classification models, which then were used in the current thesis. It was important to use the same modified function for obtaining the same features from the patient's handwriting data, for better accuracy. [13] feature engineering function was made for getting features for classification model training. It accepts path to directory with digital drawing test files and produces excel files with calculated features. In the current application the path to a single file will be provided and features should be returned as *pandas.DataFrame*. Because of these factors [13] feature engineering function was redone in the current thesis to fit the application requirements. Some graphs in the application will be built on features data produced by [13] modified function and the classification model will be applied on the obtained dataset with features for getting the diagnosis.

4.5 Application Content

After the data processing part is done, the main content of the application could be added to the graphical user interface (GUI) along with all necessary logic in back-end part. GUI main content in current thesis is following.

- *buttons*, for user interactions with the application
- *text labels*, to provide the application user with all needed information
- *select element*, for classification model selection
- *graph containers*, for representing graphs

In back-end part the logic for reading JSON file and classification model selection was added along with button click actions handling. The logic for automatic data processing after file uploading was also added to the back-end.

4.6 Multithreading

The JSON file with patient's data which the application user loads to the application may contain from approximately 2000 to approximately 4000 data entries. Processing such a big amount of data may take some time, which may result in GUI freeze and bad user experience. To solve this problem multithreading logic was added to the application. Multithreading was implemented through PyQt's *QThread* class.

When data processing or some other long-running process starts, the application user sees loading animation (Figure 4.2).
This is achieved by running long-running tasks and processes in a separate thread (worker thread), while the main thread shows the user the loading animation. Once the long-running task is finished, the worker thread finishes its work, animation stop, and the user can use the application again without experiencing GUI freezes.

Figure 4.2: Loading animation, which application user sees while application performing a long-running process

4.7 Graphs

There are three graph sections in the application. First graph section (on the left side) contains line charts of main features change over time. Second graph section (right topmost) contains one-, two- and three- dimensional scatter plots for displaying parameters limits and position of the data point along each feature. Third graph section (right bottom most) contains a model decision explanation bar chart.

PyQtGraph library was used for implementing graphs. This library allows the creation of different two- and three- dimensional, interactive graphs. In the second graph section the graph dimensionality is chosen automatically according to the number of features used for model training. If the predictive model was trained on two or three features, there will be a two- or three-dimensional scatter plot accordingly. If the predictive model was trained on four or more features, the graph will be presented as one-dimensional graphs, for each feature (Figure: 4.3).

The second section graphs themselves, represent a grid of feature points which was generated from features minimum and maximum values. On grid generation the classification model defines each point class: Parkinson's Disease (PD) or Healthy Control (HC). Then points obtain color according to their class: red for PD and blue for HC. This gives the user a good overview of features limits and borders between PD and HC values. Then the current test feature point is added to the graph and colored green. This presents the position of current features among all features in the grid (Figure: 4.4).

If there are more than three features, then instead of a grid of points in second graph section the user will be presented with a one-dimensional graph for each

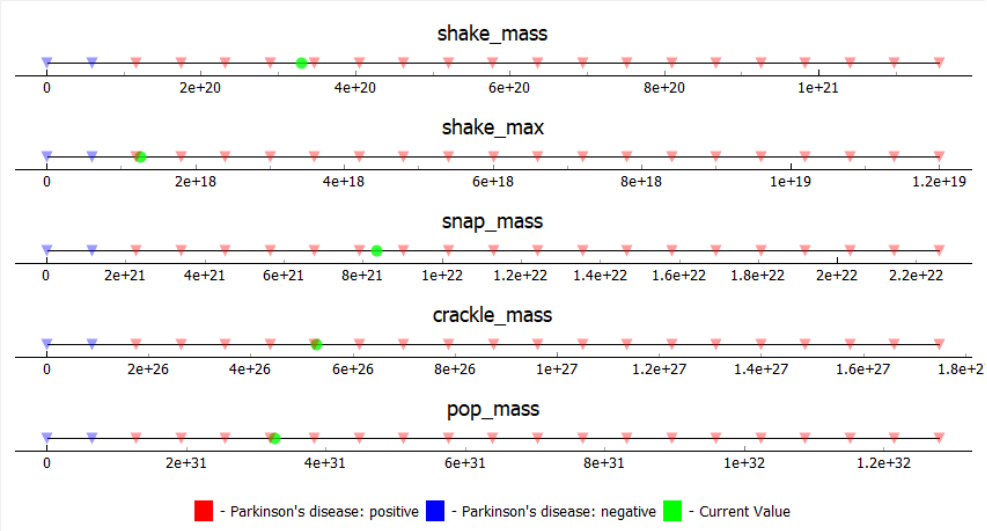


Figure 4.3: Five one-dimensional graphs, representing the border between Parkinson's disease and healthy controls for features, obtained with five dimensional model

feature correspondingly. The principle of points generation and coloring is the same as for two- and three- dimensional graphs.

4.8 Explainable Artificial Intelligence

Explainable artificial intelligence (XAI) is artificial intelligence (AI) which provides explanations of its own solutions. This allows users to understand how AI makes decisions and removes the concept of black box, when even AI creators cannot explain why AI made specific decision [10].

Main novelty component of the current thesis is XAI which is presented by use of methods that allow to get explanations for classification model decisions. A good explanation uses believable model features, shows that the training data is relevant for the given situation, and distinguishes the result from other outcomes [11].

The explanation methods are needed, because they will provide verification of the accuracy and reliability of different models in making predictions about Parkinson disease presence. This will make developed application safe to use

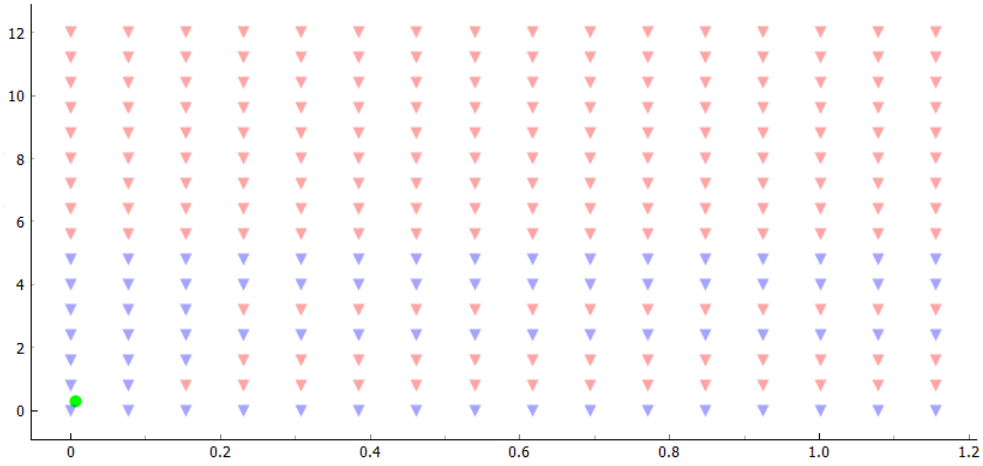


Figure 4.4: Two-dimensional grid of feature data points, representing the border between Parkinson's disease and healthy controls for features, obtained with two dimensional model

and improve user experience by making model decisions understandable and helping the application user to trust the AI.

When an analysis is based on 2 or 3 parameters, it is possible to show on the graph the location of the patient's parameters in relation to the decision boundary. Perhaps, according to the local distribution, some of the parameters will be close to the boundary, but this does not always mean that this parameter will play the largest role in the machine learning model decision. Therefore, it was decided to apply Explainable AI in addition to graph that represents the decision boundary.

Modern Explainable AI methods provide good explanation results but are designed for computer scientists instead of medical experts. However, as the use of Explainable AI in medicine is novel for Estonia, for starters it was decided to use existing Explainable AI methods in current work. Then, after receiving the feedback from practitioners, it will be possible to improve the current system if needed.

In current work Local Interpretable Model-agnostic Explanation (LIME) and SHapley Additive exPlanations (SHAP) interpreters will be used to provide explanations for results acquired by classification models.

- *LIME* is the interpreter, that supports explaining individual predictions for text classifiers or classifiers that act on tables (numpy arrays of numerical or categorical data) or images [9].

- *SHAP* is a game theoretic approach to explain the output of any machine learning model. It connects optimal credit allocation with local explanations using the classic Shapley values from game theory and their related extensions [4].

LIME and SHAP are two of the most frequently used interpreters, which provide explanations for machine learning models decisions. These interpreters show how the machine learning model was evaluating features and provide the application user with information about which feature value (according to the model) was pointing to Parkinson's disease presence and which not. Interpreters also show the weights of different features. LIME and SHAP provide understandable graphical explanations, which gives user a good overview of how the machine learning model arrived at specific diagnosis.

XAI's explanation is displayed in the third graph section as a horizontal bar chart (Figure: 4.5 4.6).

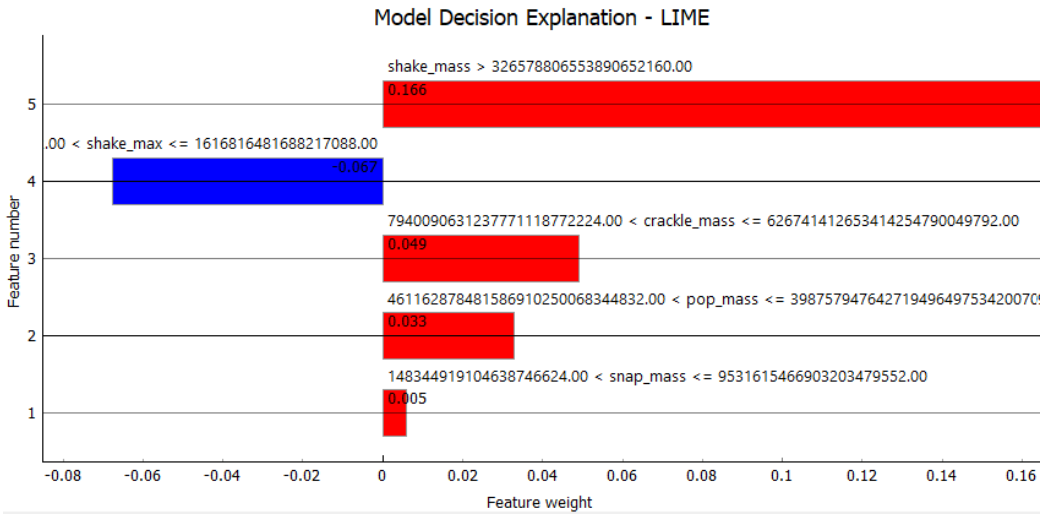


Figure 4.5: Explanation for machine learning classification decisions provided by LIME

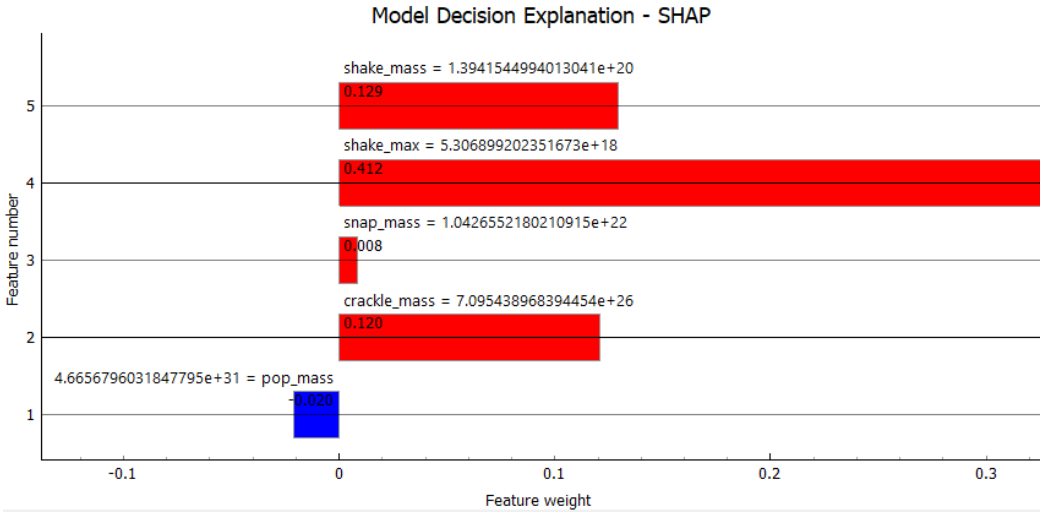


Figure 4.6: Explanation for machine learning classification decisions provided by SHAP

The application user will be able to choose between LIME and SHAP interpreters, which provides more confidence in model diagnosis and allows to compare different explanations.

4.9 PDF Report

The logic, which allows user to download a full Portable Document Format (PDF) report was added to the application. The report includes all information what the user sees in the application, including model name, all graphs, features values, diagnosis, and other data from text labels. This will allow user to save the report on the computer or tablet computer for the future or print it on the paper.

Chapter 5

Results

As a result of the current thesis the application which provides the interface for using machine learning models for the analysis of drawing and writing tests was created. The application is implemented as a desktop application for Windows platform on Python language of version 3.7 with use of scikit-learn, pandas, NumPy, PyQtGraph, LIME and SHAP libraries. Git is used as a version control system and Gitlab is used as a web repository. The application is built on PyQt5 framework.

The application provides the user an option to choose models from models' library, obtain classification results, see main features values change over time through line charts, demonstrate position of data point along each feature and ability to see features limits through one-, two- and three-dimensional scatter plots, apply two most frequently used interpreters (LIME or SHAP) on model prediction result, and get explanation for how it was obtained. All this information could be formatted and converted into the PDF format.

5.1 Workflow

The application workflow stands as follows.

1. Application user loads file with tested person digital drawing test data by clicking on *'Upload File'* button and choosing needed file from the file dialog (Figure: 4.1).

2. After the file is loaded, the user can choose between available machine learning models from model drop-down menu (Figure: 5.1).

3. After the model is chosen, user can apply model on loaded data and get results by clicking *'Run Test'* button.

4. After the results are acquired, the diagnosis put by machine learning classifier will be presented to user. The graphs with main features change over time and data point position according to other data points will be demonstrated. For the main features change over time graphs user can select features by pressing buttons with feature names below graph. Also, this graph represents real painted by patient figure. (Figure: 5.2)

5. Application user can interact with graphs, zoom in and out by scroll wheel, also move through graphs axis with use mouse left click hold and mouse move in needed direction.

6. After all, above steps are done, user can either generate full report in PDF format, by clicking *Print as PDF*, or load new file by clicking *'Upload File'* button and continue from *step 2*, or select another model from model drop-down menu and continue from *step 3*.

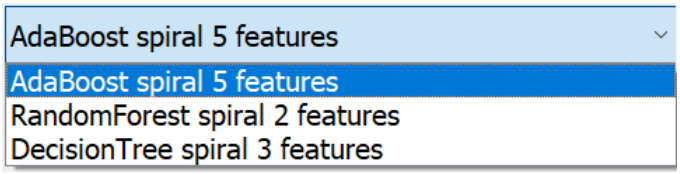


Figure 5.1: Model drop-down menu, which allows to choose machine learning model to be applied on digital drawing test

5.2 Results Validation

Within the current thesis the large number of files with real people handwriting data was provided. These files included handwriting data of people with confirmed Parkinson's disease. The files were marked correspondingly: PD for Parkinson's disease and HC for Healthy controls. These files were used to test the functionality of the application. The application was tested on more than 20 files which included both PD and HC files. In approximately ninety

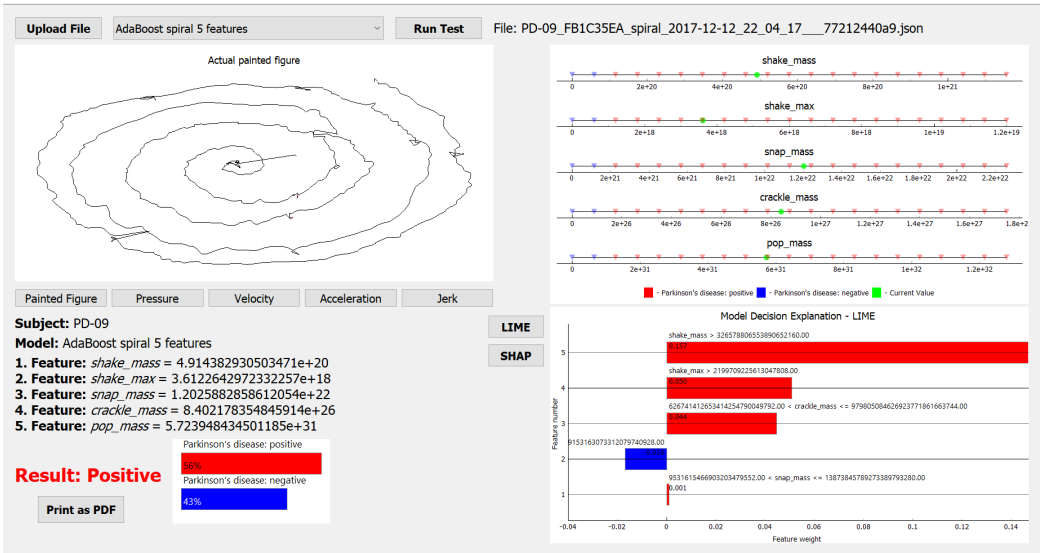


Figure 5.2: Screenshot of the application

percent of cases the application provided correct diagnosis. In another ten percent of cases, it was possible to detect the wrong diagnosis by Explainable AI explanation and by graph, which represents data point position among other features.

To test the application functionality, other classification models were developed and trained on features acquired from feature engineering functions developed by [13] and provided files with people handwriting data. These new machine learning models will come together with the final version of the application. The machine learning models cannot be considered as developed correctly as the training of machine learning classifiers was not the part of current thesis and appropriate testing was not provided.

In terms of current thesis, the application has been tested by one practitioner. To test the application the practitioner uploaded some JSON files with handwriting data to the application. The uploaded files were not used for the application testing before. After examining the results, results explanations, and graphical data representation, provided by the application and XAI, the practitioner gave positive feedback and was satisfied with the work done. Moreover, the practitioner also suggested some improvements. One of the improvements was the improvement to the actual pained figure graph,

which could also show the places where the patient was hovering the stylus. These places show where the patient was hesitating and thinking what to draw next and should be marked with a different color than the rest of the painting. These and some other small design and functionality improvements suggested by the practitioner were taken into account and implemented in the application (Figure: 5.3).

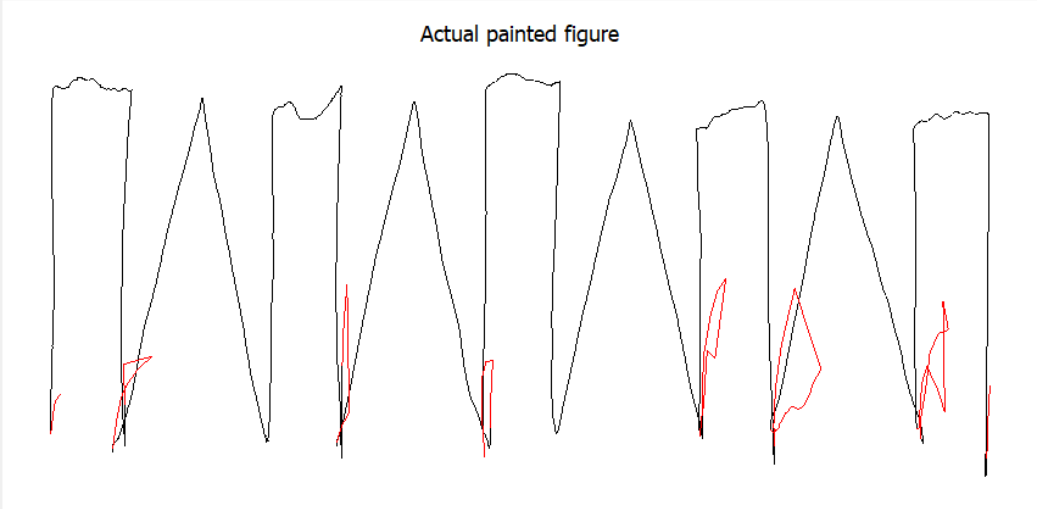


Figure 5.3: Improved figure, which show the places where the patient was hovering the stylus with red color

5.3 GitLab

The full application code can be found at: application for the analysis of drawing and writing tests TalTech gitlab

Chapter 6

Discussion

This work may be seen as the further development of the work bachelor of [12] who developed browser-based application. A web application with use of Flask library like [12] did could be the alternative variant of implementation of current project. Considering the feedback received from practitioners' decision was made to implement this application as the stand-alone Windows desktop application. The main difference from [12] work is the use of Explainable AI for results explanation. Also, the work of [12] is mostly devoted to the comparing different tests and is more designed for AI experts, whereas the present project is more oriented on the analysis of each single test and was created for field experts (medical doctors and nurses).

6.1 Restrictions

Current work has several restrictions. First, the application only accepts patients handwriting data in JSON format.

The classification models should be saved to file with .pkl extension. Each model should have JSON file with the same name as model file has and with content of model name, which will be present to user (*model_name*), list with feature names, which were used for model training (*features*) and minimum/maximum values for each feature in list where on the index of 0 goes feature minimum border value and on the index of 1 goes maximum border value. (Figure: 6.1) For the correct work of application it is recommended to only use models that were trained on minimum of two and maximum of five features.

AdaBoost_spiral.json

```
{
    "model_name": "AdaBoost spiral 5 features",
    "features": [
        "shake_mass",
        "shake_max",
        "snap_mass",
        "crackle_mass",
        "pop_mass"
    ],
    "shake_mass": [7765103971069110, 1.15582397963288E+21],
    "shake_max": [291067194747941, 11995876880517800000],
    "snap_mass": [2115859018726470000, 2.25706878759079E+22],
    "crackle_mass": [2.07275542958119E+22, 1.7482357152655E+27],
    "pop_mass": [2.06628484615518E+26, 1.17212795961071E+32]
}
```

Figure 6.1: Example of the content of JSON file with machine learning model info

Files with new models should be placed in *models* folder and files with model info in *json* folder. (Figure: 6.2) Both these folders are in */data/interface* folder from project root. If all is done correct, new model can be chosen from model drop-down menu in the application.

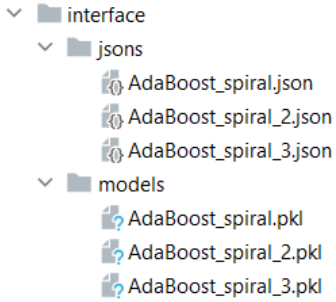


Figure 6.2: Folder structure of machine learning models and JSON files with models info

Although development of machine learning models was not part of the current thesis, to test the application functionality some machine learning models were created. The method (*train_model_best_features*), which was used for model training remains, and is in *application/model_training.py* file. This method allows to create a predictive model along easily and quickly with a

model info file and automatically save them in the correct folders and with correct extensions. Provided method can be used in future developments to allow the application user to generate models from the application's graphical user interface.

Excel files with computed features, generated with use of [13] feature engineering function are needed for XAI explanation methods. These files are containing combined handwriting data for PD and HC. For the use of the application with other type of tests, the *excel* file with combined data should be provided, along with model file and model info file. The *excel* file name should contain the name of the test and should be placed in *data/excels* folder from project root (Figure: 6.3).

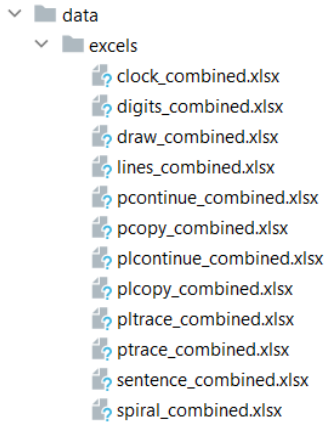


Figure 6.3: Folder structure of excel files with combined handwriting data

Currently only Spiral, Lines, Clock, P, PL, Digits and Sentence writing digital tests were examined, processed, and proved to give reliable results with classification models [7, 13]. Data collected from these digital drawing tests was used for models training which then were used in current thesis. Each model can only be applied on corresponding digital drawing test (spiral model on spiral test etc.) otherwise acquired results cannot be considered as correct. There are more tests which can be used for Parkinson's disease diagnostics, but as far as they were not examined and not used for model training, results produced on classification of these tests cannot be considered as correct or reliable.

Chapter 7

Conclusions

The main goal of this thesis was to create application, which provides the interface for using machine learning models which will allow medical doctors and nurses use the library of trained classification models for performing diagnostics of Parkinson's disease. The developed application should allow to ease the pressure on the medical personnel by providing them with the efficient tool for digital test analysis. As the result of this work an application, which meets all initial demands, was created and now is ready for testing by practitioners.

The application provides the user an option to choose between available models, see main features values change over time, demonstrate position of data point along each feature and ability see features limits, obtain classification results, apply two interpreters (LIME or SHAP) on model prediction result, and get explanation for how the model arrived at the diagnosis. All this information can be formatted and converted into the PDF format.

Main novelty component of this thesis is the use of Explainable Artificial Intelligence for explaining model decisions and increasing the level of user confidence.

The application provide ability to upload other types of machine learning algorithms in addition to the ones already used and can be used as basis for future developments and research.

Bibliography

[1] Peter Drotár, Jiří Mekyska, Irena Rektorová, Lucia Masarová, Zdeněk Smékal, and Marcos Faundez-Zanuy. Evaluation of handwriting kinematics and pressure for differential diagnosis of parkinson's disease. *Artificial Intelligence in Medicine*, 67:39 – 46, 2016.

[2] JavaFx. Javafx definition by javafx web page. `https://openjfx.io`.

[3] Lorraine V Kalia and Anthony E Lang. Parkinson's disease. *The Lancet*, 386(9996):896 – 912, 2015.

[4] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions, 2017.

[5] C. Marquardt and N. Mai. A computational procedure for movement analysis in handwriting. *Journal of Neuroscience Methods*, 52(1):39 – 45, 1994.

[6] Aleksei Netšunajev, Sven Nõmm, Aaro Toomela, Kadri Medijainen, and Pille Taba. Parkinson's disease diagnostics based on the analysis of digital sentence writing test. *Vietnam Journal of Computer Science*, 8(04):493–512, 2021.

[7] S. Nõmm, K. Bardõš, A. Toomela, K. Medijainen, and P. Taba. Detailed analysis of the luria's alternating series tests for parkinson's disease diagnostics. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1347–1352, Dec 2018.

[8] PyQt. Pyqt definition by pyqt web page. `https://pythonpyqt.com/what-is-pyqt`.

[9] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier, 2016.

[10] D. Rothman. *Hands-On Explainable AI (XAI) with Python: Interpret, Visualize, Explain, and Integrate Reliable AI for Fair, Secure, and Trustworthy AI Apps*. Expert insight. Packt Publishing, 2020.

[11] Silvie Spreeuwenberg. What is xai? what makes an explanation good? *Business Rules Journal*, 20(9), 2019.

[12] Elisa Tamm. Associations between fine motor tests among parkinson's disease patients and control groups. `https://digikogu.taltech.ee/et/Item/5052659e-bcf9-4b66-8298-dfda22f9277a`.

[13] Elli Valla, Sven Nõmm, Kadri Medijainen, Pille Taba, and Aaro Toomela. Tremor-related feature engineering for machine learning based parkinson's disease diagnostics. *Biomedical Signal Processing and Control*, 75:103551, 2022.

[14] Bas Van der Velden, Hugo Kuijf, Kenneth Gilhuijs, and Max Viergever. Explainable artificial intelligence (xai) in deep learning-based medical image analysis. 07 2021.

[15] Guang Yang, Qinghao Ye, and Jun Xia. Unbox the black-box for the medical explainable ai via multi-modal and multi-centre data fusion: A mini-review, two showcases and beyond. *Information Fusion*, 77:29–52, 2022.

[16] Jingyang Zhou, Weiwei Cao, Lan Wang, Zezheng Pan, and Ying Fu. Application of artificial intelligence in the diagnosis and prognostic prediction of ovarian cancer. *Computers in Biology and Medicine*, 146:105608, 2022.