Instituto Politécnico de Lisboa
Instituto Superior de Engenharia de Lisboa

Machine Learning and Data Mining
Project A & A1: MedKnow Data Report

October 2024

Authors:
Adam Szokalski 53153
Adrian Osędowski 53015

Group:
AMD_D_06

Contents

1 Domain Description **2**

1.1 Data Overview . . . 2

1.2 Data Features . . . 2

1.3 Assumed Daily Operations of MedKnow . . . 3

2 Project Structure **3**

2.1 MedKnow . . . 3

2.1.1 Database . . . 3

2.1.2 REST API . . . 6

2.2 SoftKnow . . . 7

2.2.1 MLFlow . . . 7

2.2.2 REST API . . . 7

3 Project A1 **9**

3.1 Domain Description . . . 9

3.2 Project structure . . . 9

3.2.1 FunghiData . . . 9

3.2.2 SoftKnow . . . 10

3.3 Orange DM project . . . 11

4 Conclusion **11**

5 Code **11**

1 Domain Description

1.1 Data Overview

The MedKnow database encompasses data related to lens prescription activities. It includes patient demographics, specifically age, and various medical parameters. Some measurements are categorized into three distinct values, while others are binary.

1.2 Data Features

Each provided record is characterized by the following set of features:

1. **Age** (`age`)
   The age category of the patient's eyes, quantified using the following values:
   - *Young* - The patient's eyes are not yet aged and may not have deteriorated.
   - *Presbyopic* - Natural aging around the 40s renders the patient's eyes unable to focus on nearby objects [1].
   - *Pre-presbyopic* - The patient is experiencing the symptoms of presbyopia prematurely, before the 40s [1].
2. **Prescription** (`prescription`)
   The type of vision deterioration, denoted by the following values:
   - *Myope* - Nearsightedness; distant objects appear blurry while nearby objects remain clear [2].
   - *Hypermetrope* - Farsightedness; nearby objects appear blurry while distant objects remain clear [2].
3. **Astigmatic** (`astigmatic`)
   Indicates whether the patient has astigmatism, which causes blurry vision. Although there are many types of astigmatism [5], the measure only has two values:
   - *Yes* - The patient has astigmatism (any type).
   - *No* - The patient does not have astigmatism.
4. **Tear Rate** (`tear_rate`)
   The amount of fluid produced by the eye. It is generally not recommended to wear contact lenses for individuals whose tear rate is not at a normal level [4]. The dataset categorizes tear rate into two values:
   - Normal - Tear rate is within the regular range.
   - Reduced - Eye produces less fluid than it should.
5. **Lenses** (`lenses`)
   The type of lenses prescribed by a doctor based on the given features. There are three possible values:
   - *Hard* - Made of rigid gas-permeable plastic that ensures stiffness [3].
   - *Soft* - Typically made of silicone hydrogel [3].
   - *None* - Contact lenses are not recommended.

1.3 Assumed Daily Operations of MedKnow

The data suggests that MedKnow's daily operations primarily consist of:

1. Registering patients and scheduling appointments.
2. Interviewing patients.
3. Testing patients' vision and measuring their features (age, prescription, astigmatic condition, tear rate).
4. Prescribing the optimal type of lenses for the patient. The doctor's expertise and knowledge are crucial in this step.

2 Project Structure

The project is divided into two logical parts: **MedKnow** and **SoftKnow**.

- **MedKnow**: Comprises a database and an API to add records and generate datasets.
- **SoftKnow**: Responsible for training and deploying models based on datasets obtained from MedKnow. It utilizes MLFlow to manage the machine learning lifecycle. The project is automatically deployable using Docker (with `Dockerfile` and `docker-compose.yaml`).

There are four containers: `medknow`, `postgres`, `softknow`, and `mlflow`, with two containers dedicated to each logical component. The REST API is developed using FastAPI, a Python framework. The database used is PostgreSQL.

2.1 MedKnow

2.1.1 Database

The database consists of tables that store information about appointments, measurements, doctors, and patients. Additionally, there are dictionary tables that help convert string values to quantized numeric values.
The `Measurements` table contains information about all features except for `lens_type`, which is prescribed by a doctor during an appointment. It also records the measurement date, its expiration date, and the ID of a previous measurement. We decided to separate measurements from appointments because measurements can be conducted by a nurse during a separate visit and can be valid for more than one appointment.

Column	**Data Type**	**Constraint**
measurement_id	SERIAL	Primary Key
measurement_date	DATE	Not Null
expiration_date	DATE	Not Null
age_id	INTEGER	Foreign Key (ages.age_id)
tear_rate_id	INTEGER	Foreign Key (tear_rates.tear_rate_id)
last_measurement_id	INTEGER	Foreign Key (measurements.measurement_id)
astigmatic	BOOLEAN	Not Null
prescription_type_id	INTEGER	Foreign Key (prescription_types.prescription_type_id)

Table 1: Measurements Table

The `Appointments` table contains information about patients, doctors, visit dates, the doctor's decision on which lens type to use, and the measurement used.

Column	**Data Type**	**Constraint**
patient_id	INTEGER	Foreign Key (patients.patient_id)
doctor_id	INTEGER	Foreign Key (doctors.doctor_id)
appointment_date	DATE	Not Null
lens_type_id	INTEGER	Foreign Key (lens_type.lens_type_id)
used_measurement_id	INTEGER	Foreign Key (measurements.measurement_id)
Primary Key: (patient_id, appointment_date)		

Table 2: Appointments Table

The `Patients` table stores data about the patient's name, surname, birth date, and the ID of the previous measurement.

Column	**Data Type**	**Constraint**
patient_id	SERIAL	Primary Key
name	VARCHAR	Not Null
surname	VARCHAR	Not Null
birth_date	DATE	Not Null
last_measurement_id	INTEGER	Foreign Key (measurements.measurement_id)

Table 3: Patients Table

The `Doctors` table stores information about doctors' names and surnames.

Column	**Data Type**	**Constraint**
doctor_id	SERIAL	Primary Key
name	VARCHAR	Not Null
surname	VARCHAR	Not Null

Table 4: Doctors Table

The following tables store the names of the quantized feature values.

Column	**Data Type**	**Constraint**
age_id	SERIAL	Primary Key
name	VARCHAR	Unique

Table 5: Ages Table

Column	**Data Type**	**Constraint**
tear_rate_id	SERIAL	Primary Key
name	VARCHAR	Not Null, Unique

Table 6: Tear Rates Table

Column	Data Type	Constraint
lens_type_id	SERIAL	Primary Key
name	VARCHAR	None

Table 7: Lens Type Table

Column	Data Type	Constraint
prescription_type_id	SERIAL	Primary Key
name	VARCHAR	None

Table 8: Prescription Types Table

An Entity-Relationship (ER) diagram of the database is presented below:

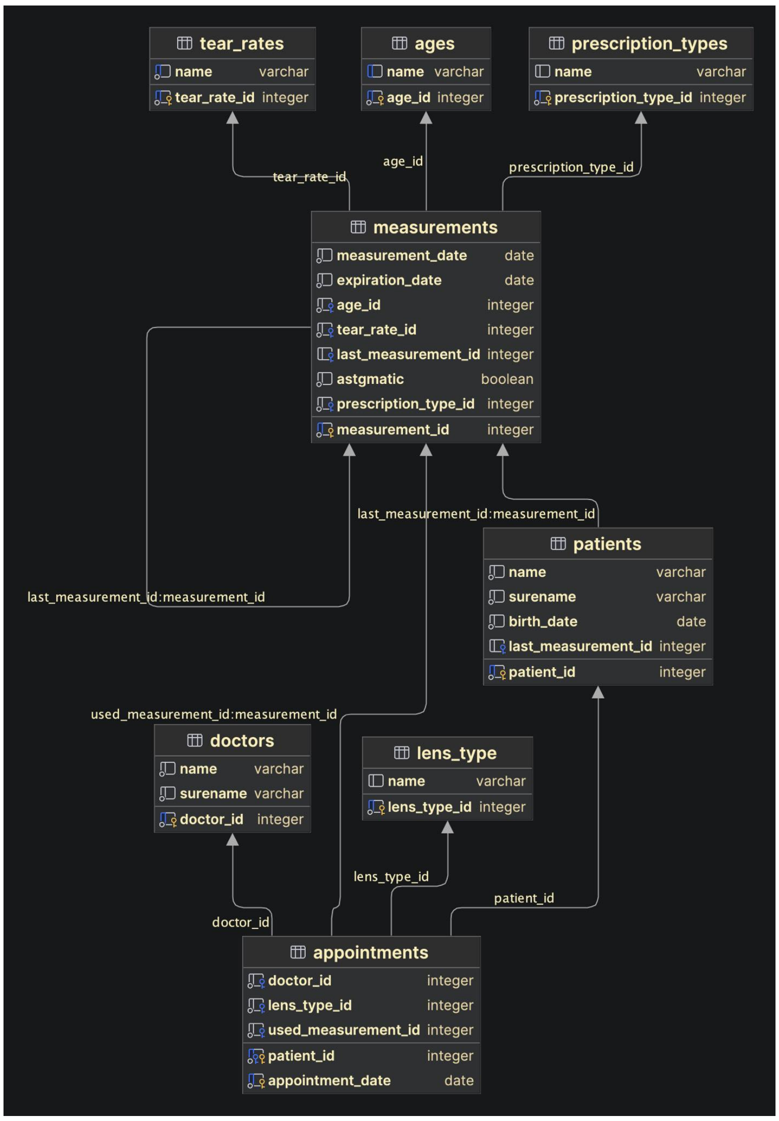

Figure 1: Entity-Relationship Diagram of the Database

2.1.2 REST API

The most important endpoints are outlined below. The `Generate Dataset` endpoint is utilized by SoftKnow to retrieve datasets for training. Additionally, there are endpoints responsible for adding records to the `Appointments` and `Measurements` tables.

- **Generate Dataset:**
  - **Method:** GET
  - **Route:** `/generate_dataset`
  - **Response:**

```
{
    "data": [
        {
            "astigmatic": <bool>,
            "age": <int>,
            "tear_rate": <int>,
            "prescription": <int>,
            "lens_type": <int>
        },
        ...
    ]
}
```

- **Add Measurement:**
  - **Method:** POST
  - **Route:** `/add_measurement`
  - **Request Body:**

```
{
    "measurement_date": "<date>",
    "expiration_date": "<date>",
    "age": "<string>",
    "tear_rate": "<string>",
    "astigmatic": <boolean>,
    "prescription": "<string>",
    "last_measurement_id": <int>
}
```

  - **Response:** (HTTP 201 Created)

```
{
    "measurement_id": <measurement_id>,
    "status": "success"
}
```

- **Add Appointment:**

  - **Method:** POST
  - **Route:** `/add_appointment`
  - **Request Body:**

```
{
    "patient_id": <int>,
    "doctor_id": <int>,
    "appointment_date": "<date>",
    "lens_type": "<string>",
    "used_measurement_id": <int>
}
```

  - **Response:** (HTTP 201 Created)

```
{
    "patient_id": <patient_id>,
    "appointment_date": "<date>",
    "status": "success"
}
```

2.2 SoftKnow

SoftKnow is responsible for training datasets obtained from MedKnow using the OneR and ID3 methods. The project is structured so that each method has its own class, and they are invoked by users through the API. Additionally, there is integration with MLFlow, a version control system for machine learning models.

2.2.1 MLFlow

MLFlow is accessed via a web browser with a user-friendly interface. It displays all trained models along with metrics such as accuracy, precision, recall, and F1 score. In this project, MLFlow is also used to retrieve the latest model for making predictions.

Another advantage of using MLFlow is its ability to store all models in a unified format, which simplifies deployment. MedKnow can export these models for independent deployment or utilize the provided API. Furthermore, MLFlow is easily extensible, allowing the addition of more models quickly.

2.2.2 REST API

There are two primary endpoints. The first is responsible for training all models without requiring user input parameters. The second endpoint handles making predictions using the previously trained model. It predicts the lens type based on features and the chosen algorithm name (ID3 or OneR) provided by the user. The endpoints are detailed below:

- **Train Model:**
  - **Method:** GET
  - **Route:** `/train_model`
  - **Description:**
    1. Fetches the dataset from the endpoint `{MEDKNOW_API_URL}/generate_dataset`.

2. Converts the dataset to a Pandas DataFrame and applies preprocessing using the `preprocess` function.
3. Trains each model defined in the `models` list and stores their training results.

  – **Response:**

```
{
    "ID3": {
        "version": <string>,
        "accuracy": <float>,
        "precision": <float>,
        "recall": <float>,
        "f1": <float>
    },
    "OneR": {
        "version": <string>,
        "accuracy": <float>,
        "precision": <float>,
        "recall": <float>,
        "f1": <float>
    }
}
```

- **Predict:**
  – **Method:** POST
  – **Route:** `/predict/{model_name}`
  – **Path Parameter:**
    * `model_name` (`"ID3"` or `"OneR"`)
  – **Request Body:**

```
{
    "astigmatic": <bool>,
    "age": <int>,
    "tear_rate": <int>,
    "prescription": <int>
}
```

  – **Description:**
    1. Fetches the latest version of the specified model from MLFlow.
    2. Applies the model's prediction method to the input data.
  – **Response:**

```
{
    "prediction": <predicted_value>
}
```

3 Project A1

3.1 Domain Description

The dataset is described in the provided `dataset_description.txt` file hence we don't need to research the meaning of the data ourselves. We studied the dataset and the provided description to understand it's format and suggest the proper preprocessing. We noticed two things:

- Some data in the dataset was missing. We filled the missing data with the value "Unknown" for the records where the missing value was not in the class column. If the class value was missing - we dropped the row.
- Provided data is text-valued, categorical and discreet - we decided to use label encoding to make the dataset more versitale and working with the previously implemented ID3 classifier.

3.2 Project structure

For the A1 Project we extended upon the previous implementation. We developed a new service to handle the needs of the **FunghiData** institute - serving the *dataset* and the *tab file*. We also generalized the **softknow** service to handle many clients and data sources. Finally we modified the OneR method to save the resulting rule in the given format.

3.2.1 FunghiData

This part is responsible for processing data from **.csv** file to proper format. The `Generate Dataset` endpoint is utilized by SoftKnow to retrieve datasets for training and `Generate Tab File` returns dataset in proper format for Orange DM.

- **Generate Dataset:**
  - **Method:** GET
  - **Route:** `/generate_dataset`
  - **Response:**

```
{
    "data": [
        {
      "class": <string>,
      "cap-shape": <string>,
      "cap-surface": <string>,
      "cap-color": <string>,
      "bruises": <string>,
      "odor": <string>,
      "gill-attachment": <string>,
      "gill-spacing": <string>,
      "gill-size": <string>,
      "gill-color": <string>,
      "stalk-shape": <string>,
      "stalk-root": <string>,
      "stalk-surface-above-ring": <string>,
      "stalk-surface-below-ring": <string>,
      "stalk-color-above-ring": <string>,
      "stalk-color-below-ring": <string>,
      "veil-type": <string>,
      "veil-color": <string>,
      "ring-number": <string>,
      "ring-type": <string>,
      "spore-print-color": <string>,
      "population": <string>,
      "habitat": <string>
    },
        ...
    ]
}
```

- **Generate Tab File:**
  - **Method:** GET
  - **Route:** `/generate_tab_file`
  - **Response:** (HTTP 200 OK)
    The response returns the generated file for download.

3.2.2 SoftKnow

- **Preprocessing**

  Label encoding is performed for each feature, because data is passed as string values, differently than for MedKnow, where it was stored in database already encoded as dictionary entity keys.

- **Rest API**

  The **Train Model** and **Predict** endpoints were modified to serve different routes for each client:

  - `/train_model/{client}`,
  - `/predict/{client}/{model_name}`.

  so now it is possible to choose between **funghidata** and **medknow**, routes.

- **Output file**

  The training of the OneR method has been modified so it logs a new artifact in MLFlow: **oneR_OUTPUT.txt**. This is the file explaining the classification rule in the requested format. It can be found for each training by navigating to:

  ```
  MLFlow (localhost:5001) > fungidata.OneR > latest run > Artifacts > oneR_OUTPUT.txt
  ```

  The output file we generated is as follows:

  ```
  (odor, ALMOND, EDIBLE) : (0, 137)
  (odor, ANISE, EDIBLE) : (0, 128)
  (odor, CREOSOTE, POISONOUS) : (0, 60)
  (odor, FISHY, POISONOUS) : (0, 174)
  (odor, FOUL, POISONOUS) : (0, 648)
  ```

```
(odor, MUSTY, POISONOUS) : (0, 15)
(odor, NONE, EDIBLE) : (33, 1123)
(odor, PUNGENT, POISONOUS) : (0, 73)
(odor, SPICY, POISONOUS) : (0, 167)
```

it can also be found on the repo in `extras` directory

3.3 Orange DM project

We used a **File** widget to load data in **.tab** format. Then in **Data Table** widget we can see total number of instances and in **Pivot table** number of instances of each class selecting one in menu. The **Tree** and **Random Forest** widgets are responsible for classification, then we can visualize data using **Tree Viewer** and **Pythagorean Forest**. The **Test and Score** widgets are used to validate data and calculate accuracy and other metrics, there is possibility to choose different methods, such as Cross validation. The **Predictions** widget shows predicted value for each row and also calculates performance scores.

4 Conclusion

The integration of MLFlow provided the capability to precisely compare algorithms. After training multiple versions of both the ID3 and OneR algorithms, it was observed that ID3 consistently outperforms OneR across all metrics and versions. ID3 delivers better accuracy, precision, recall, and F1 scores, indicating it is a more reliable model for the given data. Although ID3 exhibited tendencies to overfit in certain configurations, the limited size of the dataset restricts comprehensive testing and optimization. In summary, ID3 is definitively the better choice for MedKnow's daily operations.

The extension of a project about analyzing a dataset with information regarding the edibility of mushrooms provides a good classifier using a 1R method. The accuracy of almost 99 % was archived.

5 Code

The code for this project is available on GitHub: https://github.com/aszokalski/AMD-ProjectA

References

[1] *Pre-Presbyopia: When Your Eyes Become Older Than Your Age*, February 1, 2022. Accessed on September 30, 2024. [Online]. Available: `https://visionscienceacademy.org/pre-presbyopia-when-your-eye-become-older-than-your-age/`

[2] *Myopia vs. Hyperopia: What Are the Differences?*, July 20, 2023. Accessed on September 30, 2024. [Online]. Available: `https://riverheightseyecare.com/myopia-vs-hyperopia-what-are-the-differences/`

[3] *Hard Contact Lenses vs. Soft Contact Lenses.* Accessed on September 30, 2024. [Online]. Available: `https://trueeye.com/hard-contact-lenses-vs-soft-contact-lenses/`

[4] *Tear Exchange and Contact Lenses: A Review.* Accessed on September 30, 2024. [Online]. Available: `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4314619/`

[5] Mayo Clinic. *Astigmatism - Symptoms and Causes.* Accessed on September 30, 2024. [Online]. Available: `https://www.mayoclinic.org/diseases-conditions/astigmatism/symptoms-causes/syc-20353835#`