vertispine

https://vertispine.vercel.app/

British Orthopaedic Association x stryker

AI in Orthopaedics Hackathon 2024 submission

Jan Drmota

overview

1. Objectives
2. Code availability
3. Project structure
4. Environment setup
5. Two target classification
6. Three target classification
7. Implementation in orthopaedic outpatient clinical setting
8. Conclusions
9. Team

objectives

UC Irvine Machine Learning Repository

Using the Vertebral Column dataset:

1. Create a machine learning algorithm using 6 features to classify patients into 2 targets (normal or abnormal).

2. Create a machine learning algorithm using 6 features to classify patients into 3 targets (normal, hernia or spondilolysthesis).

code availability

GitHub repository available at:

https://github.com/jdrmota/vertispine-jupyter

Clone with URL in command line:

```
git clone https://github.com/jdrmota/vertispine-jupyter.git
```

Or clone with password-protected SSH key from command line:

```
git clone git@github.com:jdrmota/vertispine-jupyter.git
```

If you don't have git or a GitHub account

Alternatively download .zip and unzip from: https://github.com/jdrmota/vertispine-jupyter/archive/refs/heads/main.zip

Attached to submission email as well!

project structure

vertispine-jupyter/	root directory
requirements.txt	list of required libraries
vertispine2C.ipynb	Jupyter library for 2 target classification
vertispine3C.ipynb	Jupyter library for 3 target classification
.gitignore	file ignores for git
README.md	instructions
data/	data folder
column_2C_weka.arff	data file for 2 target classification
column_3C_weka.arff	data file for 3 target classification

environment setup: libraries

The list of necessary libraries are in the file `requirements.txt`

```
scikit-learn==1.5.1
matplotlib
numpy
pandas
scipy
joblib
```

Next slide explains installation process.

If those fail, manually install them with: `pip install scikit-learn==1.5.1 matplotlib numpy pandas scipy joblib`

environment setup: prerequisites

Python must be installed, I used Python3.

Jupyter notebooks are easiest to view using Anaconda Navigator, can be found at:
https://www.anaconda.com/download

ANACONDA

environment setup: running code

1. Jupyter via Anaconda navigator (recommended)

The necessary libraries will be installed with the code in the Jupyter notebook.
Select `Run > Run All Cells`. You may need to restart the kernel and re-run all cells after installation.
No further manual installation is required.

2. Command line

If you are running the code from the command line, you will need to install the required libraries with the following command (in the root directory), ensure pip is installed (comes with Python 3.4 or later), else install from: https://pip.pypa.io/en/stable/installation/

pip:

```
pip install -r requirements.txt
```

pip3:

```
pip3 install -r requirements.txt
```

two target classification

`vertispine2C.ipynb`

Extensive analysis is done in Jupyter notebook. Please see Markdown provided.

Code flow:

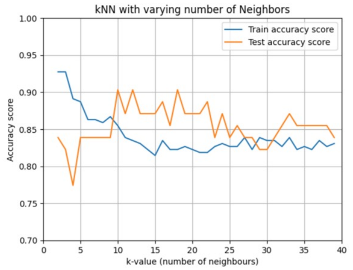
1. Import dataset
2. Pre-process data
3. **80/20** Train/test split
4. **Hyperparametric tuning** KFold GridCV
5. Evaluation of final model

two target: analysis

`vertispine2C.ipynb`

test/train split

Please see Jupyter notebook



Iterations of the kNN with different number of neighbours was tested for the best performing option. This gave the following three best options:

```
90% test accuracy with number of neighbours =
=   10
=   12
=   18
```

Hyperparametric tuning

```
Best Parameters: {'algorithm': 'auto', 'metric': 'euclidean', 'n_neighbors': 2, 'weights': 'distance'}
```

Hyperparametric tuning with k-fold split and cross-validation scoring both accuracy and f1-score gave us the above best parameters with a final accuracy of 82.26%.

two target: evaluation

`vertispine2C.ipynb`

- Simple train/test split had an accuracy of ~90%
  - However, this is only from one run that may cause overfitting/underfitting

- Hyperparametric tuning with cross-validation (scoring both accuracy and f1-scores) was therefore done to ensure the algorithm performs the best, free from dataset biases. This achieved an 82.26% accuracy which is still considered good.

- Further improvements could include:
  - Trialing different machine learning algorithm (e.g. Random Forest Classifier, Support Vector Machine, etc.)
  - Increasing dataset size
  - Further parameter tuning and testing

three target classification

`vertispine3C.ipynb`

Extensive analysis is done in Jupyter notebook. Please see Markdown provided.

Code flow:

1. Import dataset
2. Pre-process data
3. Scale data
4. **Hyperparametric tuning** KFold GridCV
5. Evaluation of final model
6. Saving and deploying

three target: analysis

`vertispine3C.ipynb`

Hyperparametric tuning

```
Best Parameters: {'algorithm': 'auto', 'metric': 'euclidean', 'n_neighbors': 13, 'weights': 'distance'}
```

Hyperparametric tuning with k-fold split and cross-validation scoring both accuracy gave us the above best parameters with a final accuracy of 79.03%.

three target: evaluation

`vertispine3C.ipynb`

- Hyperparametric tuning with cross-validation (scoring accuracy) achieved an 79.03% accuracy which is considered good.

- Further improvements could include:
  - Trialing different machine learning algorithm (e.g. Random Forest Classifier, Support Vector Machine, etc.)
  - Increasing dataset size
  - Further parameter tuning and testing
  - Using further scoring parameters for evaluation (e.g. f1-score, etc.)

implementation: steps to implement

Development of machine learning algorithm	Automated data ingestion from PACS and preprocessing
Front-end interface for clinician to interact with ⬌ Back-end integration with EPR/PACS	Seamless user-friendly integration with current IT systems
Deploy model on hospital/trust servers as cloud-based API	Adapt to hospital system requirements
Upskilling of surgeons and technical support	Interactive workshop sessions + e-module tutorials
Pilot study	Analysis - quantitative: performance & qualitative: experience
Regulatory compliance: GDPR, UKCA, IOC, MHRA, ethics	Ensuring data safety & security + confidentiality
Launch	★
Continuous monitoring and training of algorithm	Flag abnormal cases/missed cases to support clinical decision making

implementation: outpatient setting

1. Patient has X-ray images taken

2. Images uploaded to EPR/PACS

EPR
PACS

3. Web-app interacts with images, collecting parameters

vertispine

Machine learning model classifying patients based on radiological findings

- Reducing clinic time
- Ensuring diagnosis verification
- Guiding clinical decision making

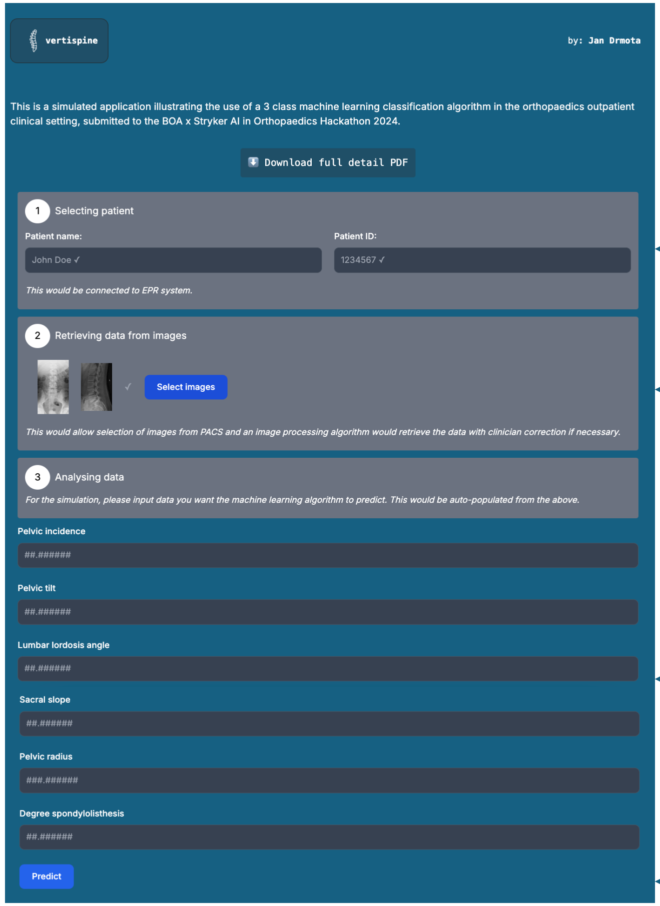
4. Clinical decision making

Initially acting as a support tool flagging missed abnormalities/confirming clinician's diagnosis. Once algorithm accuracy exceed surgeon accuracy can be considered as first-line diagnostic tool.

- Surgical management
- Steroid injection
- Analgesics
- Physiotherapy

implementation: simulation

https://vertispine.vercel.app/



Web application available at: https://vertispine.vercel.app/

Patient selected from connected EPR (simulated)

Patient images retrieved and data collected (simulated)

Input real-world patient parameters (currently manual)

Predict with 3 target machine learning algorithm

To be automated in the future from measurements made on PACS or from machine learning algorithm that can calculate from images.

Expected output from back-end API:

implementation: web-based app

https://vertispine.vercel.app/

https://vertispine.vercel.app/ Check it out!

Front-end

Tailwind CSS

React-based web application with Tailwind styling that would be connected to EPR and PACS to collect data from radiological images.
Currently data has to be inputted manually.

REST API: HTTP GET request

https://vertispine.me/?pelvic_incidence=63.027817&pelvic_tilt=22.552586&lumbar_lordosis_angle=39.609117&sacral_slope=40.475232&pelvic_radius=98.672917&degree_spondylolisthesis=-0.254400

JSON return

{"prediction":["Hernia"]}

Back-end

Python Flask micro web framework predicting target with the trained kNN machine learning algorithm loaded from sklearn.pipeline file from Jupyter output served with Nginx web server returning JSON prediction.

Hosted with:

Next.js web application served with Vercel with automatic builds from GitHub branch pushes.

Ubuntu droplet on DigitalOcean via domain https://vertispine.me hosting server.

conclusions

Drawn from analysis done in Jupyter notebooks

- kNN is a feasible algorithm choice for this task with high accuracy results
- The main limitation currently is relatively small dataset size
- The computational speed with a larger dataset may require an alternative to kNN
- Implementation in orthopaedic outpatient setting is realistically possible
  - Further evaluation studies of algorithm performance vs. clinician performance are necessary to establish clinical relevance
- Currently, algorithm can assist clinical decision making by flagging missed abnormalities

team

Jan Drmota

https://www.linkedin.com/in/jandrmota/