DOI: 10.5281/zenodo.15519378

A WEB APPLICATION TOOLS COMPRISING UTILIZATION OF AN ARTIFICIAL INTELLIGENCE

Pavol Sojka[35]

Abstract

Nowadays, many data sources can bring additional information to their owners. However, few can evaluate these resources due to a lack of knowledge in scientific fields such as statistics and mathematics. Our paper aims to partially solve this kind of problem, and we have created a tool that implements steps to facilitate this process. In our previous work, we implemented a data cleaning tool. The second step is to select a suitable model according to the given data in the process of automatically generating a suitable model, and the third step is to apply the generated model to the data. In the first chapter, we describe the problem at the overall level, in the second chapter, we present the methodology used, in the third chapter, we describe the project itself and the obtained models, the use of the generated model with assistance of artificial intelligence to describe the best-ranked model, and the last chapter contains the conclusion.

Keywords

Python, web application, scikit-learn, dataset, artificial intelligence

1 Introduction

In recent years, we have witnessed the rapid rise of new technologies and the upgrading of existing ones; thus, they have generated massive data. There are plenty of solutions available on the market, some of them free and some of them paid. This paper will not describe these applications because of the wide range, which is not our paper's aim. For example, more about free and paid solutions can be found in Mueller et al. (2019), Baumer et al. (2017), or Gearheart (2020). These authors cover the two most widely used languages in data science R and Python while the latter discusses a commercial software solution (SAS). Other options, such as IBM SPSS, are also available.

A similar objective was addressed by Sojka (2023), who developed a web-based tool to support data cleansing, model selection, and basic analysis for users without deep technical expertise. His work also emphasizes the importance of automating key stages of the data science workflow using Python-based AutoML frameworks, aligning closely with our focus on usability and accessibility. Further in the text, we will introduce some of the most commonly used technologies, but our primary goal is to use Python with specialized libraries to create our application. Our aim was to create and implement an application in which a user with standard capabilities can upload a dataset and perform basic operations such as data cleansing and initial analysis, followed by saving the processed dataset, generating models, and testing them

[35] University of Economics in Bratislava, Faculty of Economic Informatics, Department of Applied Informatics, Dolnozemská cesta 1, 852 35 Bratislava, pavol.sojka@euba.sk.

(Qinglei et al., 2017). Fundamentally, the objective of data science is to provide consistent and clean data; however, in real-world scenarios, this is rarely achievable due to the presence of duplicates, erroneous entries, and abnormal or inconsistent values.

1.1 First steps in data science

As mentioned above, Qinglei et al. describe preparing data for another evaluation. Without these important steps, the final result could be distorted; therefore, the decision could be wrong, and the whole process would be a waste of time. The first steps taken on the provided dataset should cover data deduplication, deleting erroneous ones, finding missing data, and correcting them with either neutral or average numbers. There could also be other methods, but in our paper, we use three forms of replacing missing data: providing zero values, averaging, and deleting whole rows of data, because no other methods are suitable. The user can try all the methods independently to determine which suits his/her needs. After processing all the data, we can choose the appropriate model applied to our dataset. Choosing the proper one, we describe it later in our work.

1.2 Python environment

Python is a dynamic interpreted language. It is sometimes classified among the so-called scripting languages, but its possibilities are more significant. Python was designed to allow the creation of large, full-fledged applications (including a graphical user interface—see, for example, wxPython, which uses wxWidgets, or PySide and PyQT for Qt, or PyGTK for GTK+).

Python is a hybrid (or multi-paradigmatic) language, that is, it allows you to use not only an object-oriented paradigm when writing programs, but also a procedural and, to a limited extent, a functional one, depending on what suits you or is best for the given task. Because of this, Python has excellent expressive capabilities. The program code is short and easy to read compared to other languages.

A distinctive feature of the Python language is productivity in terms of the speed of writing programs. This applies to both the most straightforward programs and pervasive applications. In the case of simple programs, this property is manifested mainly by the brevity of the notation. For large applications, productivity is supported by features used in large-scale programming, such as native support for namespaces, use of exceptions, standard tools for writing tests (unit testing), and more. High productivity is associated with the availability and ease of use of a wide range of library modules, enabling easy solutions to tasks from many areas (Jaworski et al., 2021).

Considering all the conditions that must be met, we decided to use *Python with all the packages provided. This language also has broad support from the developer community and is widely spread in the* commercial sphere.

1.3 Streamlit library

Streamlit is a free, open-source, all-python framework that enables data scientists to quickly build interactive dashboards and machine learning web apps with no required front-end web development experience. If you know Python, then you are all equipped to use *Streamlit* (Li, 2022). It is a cost-free product, consisting of all the necessary tools to build interactive web pages with charts, maps, sliders, and other valuable tools. You can create dashboards and other interactive content with a few lines of code. It employs its web server, so the application is ready.

1.4 Pandas and matplotlib libraries

In our application, we used, amongst others, a package called *Pandas* and *Matplotlib*. *Pandas* is an open-source Python package that provides some data analysis tools. The package has several data structures that can be used for various data manipulation tasks. It also has a variety of methods that can be used to analyze data, which is useful when working on data science and machine learning problems in Python. This package is widely used in data analysis and for working with external data like Excel worksheets, JSON (JavaScript Object Notation) files, CSV (comma-separated values) files, etc. The package contains extended capabilities for working with rows and columns altogether, without having to use loops; functions are used instead, so that we can call this type of programming paradigm functional programming. As mentioned, Python is a mix of procedural, object-oriented, and functional programming.

Matplotlib is a plotting library available for the Python programming language as part of NumPy, a numerical data processing resource. It uses an object-oriented API to insert plots into Python applications (Nelli, 2018). Our application displays a histogram plot.

2 Methodology

Based on the literature review, we have chosen the framework mentioned in the text. According to Richards, "Streamlit shortens the development time for the creation of data-focused web applications, allowing data scientists to create web app prototypes using Python in hours instead of days." (Richards, 2021). According to Nokeri, "Secure web apps and deploy them to cloud platforms" (Nokeri, 2021), we also decided to deploy our application to the cloud infrastructure in Oracle Cloud. Now, let us describe the steps to fulfill our goal.

1. We decided to develop an application allowing users to upload their data files and perform basic cleaning tasks. This approach is called „data cleansing, " and according to Nauman, „Data cleansing is a complex set of tasks that takes as input one or more datasets and produces as output a single, clean dataset " (Nauman et al., 2022). This process takes a vast amount of time, so we decided to reduce this amount of time by reducing some tasks involved in data cleansing. After data cleansing, we will try to find an appropriate classification model to apply to our dataset.
2. After some research, we decided to use web technology due to its widespread use and ease of use with web browsers.
3. We used a web framework, Streamlit, based on *Python* language technology.
4. Developing source code to implement the application.
5. Testing and deploying on the local server.
6. Final application deploying on Oracle cloud infrastructure on a Linux virtual machine.

3 Environment preparation

3.1 Prerequisites

We needed to take mandatory steps to achieve the final goal. First, we must download the Streamlit libraries from the project's home website. On the website, some installers are suitable for various operating systems and architectures. After choosing the appropriate one, you are

recommended to follow the instructions to finish the installation process. To achieve this task, implementors should know how to administer operating systems, either Windows, Linux, or macOS. When installation is successfully done, we can start the Streamlit framework. If everything works fine, we get a link and port on which the web server listens. The user is then redirected to the web browser, where the application will reside when completed. The steps described above are necessary; otherwise, implementation will not be possible.

3.2 Appropriate dataset selection

Our application is now entirely suitable for all datasets. We aimed to use this application on datasets that mostly contain numerical values organized in rows and columns. So, we prefer a standard form of dataset, known widely as an Excel sheet or a LibreOffice sheet, but exported to CSV format. Other types of datasets that mostly contain string values are not suitable. For categorical values, we implement various encoders that employ algorithms that convert categorical values to groups of numbers, allowing them to function correctly. We used the sonar.csv dataset for this type of machine learning computation, retrieved from GitHub, to maintain a proper approach to our project. Dataset sonar.csv contains data describing whether echoed back acoustic signals belong to mine (M) or rock (R) reflection patterns.

3.3 Appropriate model selection

Our project is divided into several parts. The first is data cleansing and a simple, brief graphical analysis based on histogram plots. The second part is model selection; the final part comprises model testing. Testing selected models will not be included in this paper.

Choosing the appropriate model is a very complex and challenging process. It comprises choosing the suitable type of estimator, an appropriate algorithm, selecting hyperparameters, their values, their different combinations, and so on. The machine learning industry is thus very complex and challenging for one person to understand. When you look at machine learning algorithms, there is no unique solution or one approach that fits all. Several factors can affect your decision to choose a machine learning algorithm (Harlalka, 2018). Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns, and make decisions with minimal human intervention (Gradillas, 2021).

Our application is written as a web application with a straightforward and intuitive interface, so any user with minimal knowledge of statistical methods or machine learning algorithms can use it for their data. We decided to implement classifier estimators only with prepared classifiers in the training dataset; this is called supervised learning. Other types of learning we did not use in our work are unsupervised and reinforcement learning.

For the complexity of such an application, we implemented a classification algorithm only, but the implementation of regression algorithms is planned for future implementation. Our application offers data cleansing, model search, and application of models found by automated machine learning. The user must provide the file in CSV (comma-separated values) with the pre-classified dataset to train the model. Subsequently, the dataset is uploaded to the server, and the search for the model will start. Then we can try our newly found model on our dataset to verify its precision.

At this point, we must remember that this part is very sensitive because we must choose the right amount of time to perform computations. We tested auto-selected models within the five-minute limit, as we trained models on small datasets, and we were limited by the hardware resources given to the server. This type of computation depends on the number of computer

processors and the size of the operational memory. We also tested our computations on a notebook with eight cores and sixteen gigabytes of memory, but the server we used has one core and six gigabytes of memory. Therefore, we have to choose parameters to avoid unreasonable time to get results. Our solution is prepared for server usage because of the web application type of service. We recommend using our application in a server environment with a minimum of four cores and eight gigabytes of memory to get reasonable results in a reasonable timespan for larger datasets. We set the time to a fixed value (for us it was five minutes), which is the same for every hardware configuration, but here applies the clause that the more efficient hardware, the more tested algorithms (figure 1, bolded text).

Figure 1: Finding the best model

```
auto-sklearn results:
  Dataset name: c1241e23-271b-11ee-80b0-00155de7a23f
  Metric: accuracy
  Best validation score: 0.913043
  Number of target algorithm runs: 128
  Number of successful target algorithm runs: 113
  Number of crashed target algorithm runs: 14
  Number of target algorithms that exceeded the time limit: 1
  Number of target algorithms that exceeded the memory limit: 0

Accuracy: 0.7391304347826086

{66: {'model_id': 66, 'rank': 1, 'cost': 0.08695652173913049, 'ensemble_weight': 1.0,
'data_preprocessor':
<autosklearn.pipeline.components.data_preprocessing.DataPreprocessorChoice object at
0x7fc011c5eb80>, 'balancing': Balancing(random_state=1, strategy='weighting'),
'feature_preprocessor':
<autosklearn.pipeline.components.feature_preprocessing.FeaturePreprocessorChoice
object at 0x7fc0116867f0>, 'classifier':
<autosklearn.pipeline.components.classification.ClassifierChoice object at
0x7fc00e6c63a0>, 'sklearn_classifier': LinearSVC(C=493.6813394002163,
class_weight='balanced', dual=False,
    intercept_scaling=1.0, random_state=1, tol=0.05050518248614478)}}
```

Source: own elaboration

3.4 Python code snippet

When a user wants to use the model auto selection feature, he/she must upload the source CSV data file to the server. As the application creators, we can tune model outcomes with so-called hyperparameters. This part of model creation is called hyperparameter tuning, and only the application creators can now manipulate them. In future versions of the application, we can allow users to modify some parameters to a limited extent, like entering other times as we preset in our application (five minutes). This should be done carefully because entering a higher value for the time limit can cause a massive workload to be applied to the server. Figure 2 shows a

code snippet showing automated machine learning Python code with hyperparameters in bolded font.

Figure 2: Python code snippet with parameters

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=1)
model = AutoSklearnClassifier(time_left_for_this_task=5*60, per_run_time_limit=30,
n_jobs=8, ensemble_kwargs={'ensemble_size': 1})
model.fit(X_train, y_train)
```

Source: own elaboration

As shown in Figure 3, we are searching for the best classification model, and we can tune some arguments to achieve our goal. Beyond other possible parameters, we are using time_left_for_this_task, where we set the overall time to run the whole program. We can say that the more time we spend running the program, the better the results will be. In our research, we chose a reasonable time to be set to five minutes, because our dataset was not big enough to set a higher value. We performed tests with the newly generated model on the same dataset but without classification labels, and the model classified the same data as the original dataset with pre-labeled data rows. This can be considered the best result to achieve. We would have obtained highly precise classifications if we had a similar dataset. We must consider that the final model could be overfitted if we strongly push for the highest accuracy of the model beyond a reasonable level. Another parameter is ***per_run_time_limit***, which limits the run time for one tested model, and we set it to thirty seconds, after which the cycle is terminated. Another cycle with different parameters is used. The final parameter in our script is ***ensemble_size***, which means several finally generated models ordered from the best to the worst. The figure below shows the three best models generated after we set the number to three.

From the given result (Fig. 3), we see the first three models ordered from the first *RandomForestClassifier* as the best one, and the second and third ones are *MLPClassifier* (Multi-layer Perceptron classifier) as the best models to use with our training dataset. Second and third places use the same classifier and therefore are distinguished by different arguments called hyperparameters enclosed in parentheses that *AutoSklearnClassifier* chose as the best ones for our scenario.

Figure 3: Finding the best three models

```
auto-sklearn results:
  Dataset name: fd2992cc-28db-11ee-804a-00155de7adf2
  Metric: accuracy
  Best validation score: 0.913043
  Number of target algorithm runs: 139
  Number of successful target algorithm runs: 120
  Number of crashed target algorithm runs: 18
  Number of target algorithms that exceeded the time limit: 1
  Number of target algorithms that exceeded the memory limit: 0

Accuracy: 0.782608695652174
```

*{14: {'model_id': 14, **'rank'**: 1, 'cost': 0.15217391304347827, 'ensemble_weight': 0.3333333333333333, 'data_preprocessor': <autosklearn.pipeline.components.data_preprocessing.DataPreprocessorChoice object at 0x7f5b4c0b9fa0>, 'balancing': Balancing(random_state=1), 'feature_preprocessor': <autosklearn.pipeline.components.feature_preprocessing.FeaturePreprocessorChoice object at 0x7f5b4be94700>, 'classifier': <autosklearn.pipeline.components.classification.ClassifierChoice object at 0x7f5b4c0af970>, 'sklearn_classifier': **RandomForestClassifier**(criterion='entropy', max_features=15, n_estimators=512,*

*n_jobs=1, random_state=1, warm_start=True)}, 53: {'model_id': 53, **'rank'**: 2, 'cost': 0.13043478260869568, 'ensemble_weight': 0.3333333333333333, 'data_preprocessor': <autosklearn.pipeline.components.data_preprocessing.DataPreprocessorChoice object at 0x7f5b4bfbe730>, 'balancing': Balancing(random_state=1, strategy='weighting'), 'feature_preprocessor': <autosklearn.pipeline.components.feature_preprocessing.FeaturePreprocessorChoice object at 0x7f5aecffec70>, 'classifier': <autosklearn.pipeline.components.classification.ClassifierChoice object at 0x7f5aecffe790>, 'sklearn_classifier': **MLPClassifier**(alpha=8.440979695014983e-05, beta_1=0.999, beta_2=0.9,*

hidden_layer_sizes=(157, 157, 157),

learning_rate_init=0.0002774746616573728, max_iter=128,

n_iter_no_change=32, random_state=1, validation_fraction=0.0,

*verbose=0, warm_start=True)}, 79: {'model_id': 79, **'rank'**: 3, 'cost': 0.08695652173913049, 'ensemble_weight': 0.3333333333333333, 'data_preprocessor': <autosklearn.pipeline.components.data_preprocessing.DataPreprocessorChoice object at 0x7f5b4bfacc70>, 'balancing': Balancing(random_state=1, strategy='weighting'), 'feature_preprocessor': <autosklearn.pipeline.components.feature_preprocessing.FeaturePreprocessorChoice object at 0x7f5aed24d1c0>, 'classifier': <autosklearn.pipeline.components.classification.ClassifierChoice object at 0x7f5aed44f700>, 'sklearn_classifier': **MLPClassifier**(activation='tanh', alpha=2.89664380675713e-05, beta_1=0.999,*

beta_2=0.9, hidden_layer_sizes=(50, 50, 50),

learning_rate_init=0.00019621972402090513, max_iter=256,

n_iter_no_change=32, random_state=1, validation_fraction=0.0,

verbose=0, warm_start=True)}}

Source: own elaboration

4 Downloading final results

The web application returns the first best model in a brief information window. It offers a button to download all the models with their hyperparameters ranked from one to the

maximum models defined in the parameter ensemble_size. A web application generates a binary file in the background with serialized models used to create the best models. This data is subsequently deserialized and loaded into the application in the final step, which consists of deploying the model on another dataset without knowing its classification hyperparameters, using the scikit-learn package.

5 Using artificial intelligence to describe a first-ranked model

We used Google Gemini's artificial intelligence engine to retrieve the model description. Our application offers a new, intuitive interface to interact with artificial intelligence to obtain a detailed description of our first-ranked model with pros and cons, tested on our given dataset.

Figure 4: Python code query to Google Geimini AI model

```
model = genai.GenerativeModel("gemini-1.5-flash-latest")
txt = f"describe {classifier[0]}"
response = model.generate_content(txt)
print(response.text)
```

Source: own elaboration

Figure 5a: Artificial intelligence model description

`SGDClassifier` in scikit-learn is a linear model that implements stochastic gradient descent (SGD) learning. It is a powerful and versatile algorithm suitable for large datasets because of its efficiency.

In summary, `SGDClassifier` is a powerful and efficient algorithm for large-scale classification tasks, especially when dealing with linear or nearly linear relationships between features and classes. Its flexibility and ability to handle online learning make it a versatile tool in machine learning, but careful hyperparameter tuning and feature scaling are essential for optimal performance.

Source: own elaboration

Figure 5b: Artificial intelligence model description

A RandomForestClassifier is a supervised machine learning algorithm used for classification tasks. It is an ensemble method, meaning it combines multiple decision trees to produce a more accurate and robust prediction than a single decision tree could achieve on its own

In summary, a RandomForestClassifier is a powerful and versatile classification algorithm that combines the strengths of multiple decision trees to achieve high accuracy and robustness. Its ease of use and good performance make it a popular choice for many machine learning applications.

Source: own elaboration

With the knowledge of various models, we can decide which model should be used as the primary one to evaluate our data. Further models should also be considered if needed, or if the first-ranked model does not meet our assumptions according to the given data.

6 Conclusion

Our paper aimed to allow people with less machine learning knowledge to ease the model preparation process. Our secondary goal was also to make an application that would be dataset independent (meaning no unique names for columns and rows), so anyone with any dataset saved in CSV format could upload the dataset to our server. These goals were fulfilled, and the third, much more difficult part remains to process further prepared (cleansed) data. In brief, these steps would comprise feature selection, various model testing, and finally applying a winning model to our dataset. We have already implemented these steps in Python using the auto-sklearn package, specifically designed for these data science tasks. The final step in the creation of our application is to implement a possibility to test the generated model on the user's data by simply uploading it to the server. Subsequently, the saved model (generated model) will be applied to the data using the Python library scikit-learn. After completing the task, there will be an offer to download the file with classification results. With the help of artificial intelligence, we can assess the use of a given model more intensively to gain some knowledge about the provided data.

Comparing alternatives with similar abilities is somewhat tricky because we created this type of application based on our research to use existing solutions with such capabilities bundled in one web application. We could not find that type of solution with a simple user environment. We discovered that users can use similar bundles of products in Python environments such as Anaconda, Conda, and the like, or libraries like *Pandas* or *NumPy*. However, the user's knowledge of how to use these libraries must be sufficient to decide which library should be used first, second, and so on. Our application handles this type of problem as an easy set of steps from the beginning to the end of the webpage, without the user knowing how data is handled internally.

Our concept is in alpha testing of the design to provide a so-called "proof of concept" that allows us to improve some parts of the application. The current state of the application is unsuitable for widespread use because some other technical aspects, like concurrent processing of various datasets, archiving results, and a user logging facility, have not yet been implemented. Further shortcomings represent technical issues like needing more processing time, memory, and hard disk space on the cloud infrastructure. Some calculations need plenty of time to find the optimal solution, so in the future, it would be better to implement graphical processing units (GPUs) instead of central processing units (CPUs). For example, the Python CUDA library handles parallel computations more efficiently than CPU libraries. For now, we are using Oracle's free tier infrastructure, which is sufficient for application development and improvement. However, a better hardware environment must be purchased for real-world testing with high-volume datasets.

7 Resources

Gearheart, J. (2020). End-to-end data science with SAS: A hands-on programming guide. SAS Institute.

Gradillas, R. (2021). Machine Learning Algorithms: How to Choose the Right Kind of Machine Learning Model. Independently Published.

Harlalka, R. (2018, October 3). *Choosing the right machine learning algorithm*. Medium. https://medium.com/hackernoon/choosing-the-right-machine-learning-algorithm-68126944ce1f

Jaworski, M., & Ziadé, T. (2021). Expert Python programming: Master Python by learning the best coding practices and advanced programming concepts. Packt.

Li, S. (2022, January 14). Streamlit hands-on: From zero to your first awesome web app. Medium. Retrieved July 30, 2022, from https://towardsdatascience.com/streamlit-hands-on-from-zero-to-your-first-awesome-web-app-2c28f9f4e214

Mueller, J. P., & Massaron, L. (2019). Python for data science for dummies. Wiley.

Nauman, F., & Herschel, M. (2022). An introduction to duplicate detection. Springer Nature.

Nelli, F. (2018). Python data analytics: With pandas, NumPy, and Matplotlib. Apress.

Nokeri, T. (2022). Web App Development and Real-Time Web Analytics with Python: Develop and Integrate Machine Learning Algorithms into Web Apps. Apress.

Richards, T. (2021). Getting Started with Streamlit for Data Science: Create and deploy Streamlit web applications from scratch in Python. Packt Publishing Ltd.

Sojka, P. (2023). *Model preparation tool based on automatic machine learning framework implemented as a web application*. In *Proceedings of the TIEES 2023 Conference* (pp. 154–161).

Springer Verlag, Singapore. (2018). Data science: 4th International Conference of Pioneering Computer Scientists.