Nature-Inspired Computing Project Report

Ayhem Bouabid
DS-01
Innopolis University
Innopolis, Russian Federation
a.bouabid@innopolis.university

Majid Naser
DS-01
Innopolis University
Innopolis, Russian Federation
m.naser@innopolis.university

Nikolay Pavlenko
DS-01
Innopolis University
Innopolis, Russian Federation
n.pavlenko@innopolis.university

***Abstract*—tbd**

***Index Terms*—tbd**

I. Introduction

Quality data preprocessing is an important prerequisite to creating accurate machine learning models. It solves a wide range of problems, from missing data and data inconsistency to required anonymization. In our project we have decided to focus primarily on one aspect of it, that is, **feature selection**.

Performing feature selection allows us to solve a great number of problems, being especially effective in large datasets containing many features. It improves accuracy of predictions by finding and eliminating spurious relationships, as well as reducing the chances of overfitting. Other effects of feature selection are the improvement of training time for the model through cutting down on unnecessary data, and increase in interpretability, as fewer features have to be analyzed to figure out the dependencies.

However, the main problem that can be solved by feature selection for multiple regression models (and will be addressed specifically in our project) is **multicollinearity**. Multicollinearity is an effect when multiple explanatory variables that are believed to me independent from each other, are in fact, closely interrelated. This can have damaging effects on the accuracy of prediction models, as even a small change in data will lead to unpredictable results. Ideally, feature selections would find such relationships and purge the dataset from them, our model aims to do so as well.

As to the practical applications of our project, we have decided to stick to the original project proposal and test it on a new dataset that would contain various countrywide statistics as features that will try to predict the population growth for a country in a given year. Since we have found no such dataset that would have suited our needs, it was created from scratch, using data provided by the World Bank Open Data site [2]. However, during our work on the project, we have decided to also test other similar datasets, that could be used for a multiple regression model, to figure out if our implemented feature selection process can be used more generally.

NEW INFO ABOUT OTHER DATASETS WILL BE ADDED HERE

II. Related Work

NEW INFO ABOUT SOLVING MULTICOLLINEARITY WILL GO HERE

The idea of feature selection using Nature-Inspired Algorithms is not new - we have based our project around already existent frameworks that implement those algorithms and apply them as optimizers in preprocessing tasks.

One framework that we have used extensively is NiaPy library [6]. They have implemented a large collection of nature-inspired algorithms, and provided a simple interface to use them as optimizers. Their library is also used by the Transaction Fraud Detection project, which is based around the same idea of feature selection with the nature-inspired algorithms and has served as an inspiration behind our own project, so we have decided to follow in their footsteps.

Later, in the process of searching for another library that is compatible with NiaPy implementation of various nature-insipred algorithms, and would provide a friendly and extendable Python interface to use them as optimizers in combination with Pandas and sklearn libraries, we have found the evopreprocess library [3].

That library contains several main modules: `data_sampling`, `data_weighting`, and `feature_selection`. Howeveer, only the last module is relevant for our task, so we will only describe its functionality. The task class for `feature_selection` in evopreprocess is `EvoSampling`, which extends `_BaseFilter` class from scikit-learn. It is important to list the parameters of that class, as they are going to play a big role in the feature selection process:

- `random_seed` - seed state, by default is equal to system time in milliseconds.
- `evaluator` - ML approach used to evaluate the data, expecting a scikit-learn-compatible regressor or classifier.
- `optimizer` - NI optimizer, expecting a NiaPy-copatible method, by default Genetic algorithm

- `n_folds` - number of folds for the cross-validation split into training and validation sets
- `n_runs` - number of runs of the optimizer on each fold
- `benchmark` - evaluation class that measures the quality of data sampling. By default optimizes error rate and F-score for classification problems and mean square error for regression problems
- `n_jobs` - number of optimizers to be run in parallel, by default equal to the number of your CPU's cores

III. Methodology

A. Multicollinearity issue

1) Multicollinearity detection: First problem that we are going to address in our dataset would be multicollinearity. It can be detected in a dataset by finding the VIF - variance inflation factor, calculated for each feature separately, according to the formula below, where R^2 is the coefficient of determination:

$$VIF = \frac{1}{1 - R^2} \tag{1}$$

VIF can normally take positive values, and by checking its value for each column, one could determine whether or not a serious issue is detected. Value of VIF greater than 1 identifies existence of some collinearity, greater than 5 is already a cause for concern, and greater than 10 must be addressed [4].

As our solution to multicollinearity we have decided to combine 2 techniques: clustering and Particle Swarm Algorithm.

2) Clustering: When collinear are present in a dataset, model can gather the same information from one feature, as it can from a group of features. This makes it possible to try and solve multicollinearity by applying clustering with a certain threshold on features, that have been proven to have a correlation [5]. In order to find those correlating features, we could use Spearman correlation, which measures the strength and direction of the monotonic relationship between two variables that can be ordered by transforming their values to ranking in their respective domains and comparing their relationship.

3) Particle Swarm Optimization Clustering: Particle Swarm Optimization is one of the most influential Nature-Inspired algorithms that can optimize a problem by iteratively improving a chosen solution, depending on the elected performance measure [1]. This property makes it possible to use PSO for our clustering problem, where the final choice of solution will be determined by the algorithm and defined fitness function.

B. Feature Selection

In our research, we were able to identify three categories of feature selection methods. Firstly, it is the filter method, which ranks each feature on some statistical metric, and evaluates the ranks afterwards, picking the ones that score the highest. Secondly, it is the wrapper method, which takes a subset of features and trains a model using them. Depending on results of the testing, it adds or removes features from the subset, incrementally improving the performance until user-defined stopping criteria is achieved. Thirdly, it is the embedded method, which is in-built into models themselves - it add a penalizing term to the regression equation (en example of such a model would be Lasso Regression).

Among all categories mentioned, wrapper method has the highest computational costs, but can provide the best dataset that would provide the most accurate results for our model. It can also include nature-inspired algorithms as its estimators, making **wrapped** category of feature selection models our preferred choice.

IV. Github link

$https://github.com/Daru1914/NIC\_Project$

V. Experiments and Evaluation

tbd

VI. Analysis and Observations

tbd

VII. Conclusion

tbd

References

[1] Augusto Luis Ballardini. A tutorial on particle swarm optimization clustering. *arXiv preprint arXiv:1809.01942*, 2018.

[2] The World Bank. World bank open data. https://data.worldbank.org/. Accessed: 2022-12-02.

[3] Sašo Karakatič. EvoPreprocess—Data Preprocessing Framework with Nature-Inspired Optimization Algorithms. *Mathematics*, 8(6), 2020.

[4] Scott Menard. *Applied logistic regression analysis*. Number 106. Sage, 2002.

[5] scikit-learn developers. scikit-learn. https://scikit-learn.org/stable/auto_examples/inspection/plot_permutation_importance_multicollinear.h Accessed: 2022-12-04.

[6] Grega Vrbančič, Lucija Brezočnik, Uroš Mlakar, Dušan Fister, and Iztok Fister Jr. NiaPy: Python microframework for building nature-inspired algorithms. *Journal of Open Source Software*, 3, 2018.

VIII. Funni table

TABLE I
Table Type Styles

Table Head	Table Column Head		
	Table column subhead	***Subhead***	***Subhead***
copy	More table copy[a]		

[a] Sample of a Table footnote.

IX. Funni picture

Fig. 1. Example of a figure caption.