TDS project - Tools for automating and improving feature selection

Yuval Uner

April 2024

1 Abstract

In predictive modeling with tabular data, not all features are equally informative, and some may even be harmful to the model. This project addresses the challenge of feature selection by introducing a suite of automated algorithms designed to identify and retain only the most relevant features. Evaluations conducted on multiple datasets demonstrate the efficacy of these algorithms, with most achieving competitive or superior performance compared to existing methods.

2 Problem Description

Feature selection is one of the most relevant processes in any methodology for creating a statistical learning model. The focus of feature selection is to select a subset of variables from the input which can efficiently describe the input data while reducing effects from noise or irrelevant variables and still provide good prediction results.

Performing feature selection well can help create more accurate, easier to understand and faster to train models while also making the data itself easier to interpret. On the other hand, poor feature selection may yield exactly the opposite effects.

However, performing feature selection well may be a difficult process, as finding and selecting a globally optimal subset is too computationally expensive, at $O(2^n)$ (where n is the number of features), and manual analysis or trial and error are impractical for significant number of features. Therefore, algorithms that can automatically perform feature selection and find sub-optimal yet close to optimal subsets in reasonable time are essential for this part of the data science pipeline.

3 Solution Overview

3.1 General approaches to feature selection

In this project, the focus was on two of the main approaches to feature selection: filter methods and wrapper methods.
Filter methods use variable ranking techniques to assign a score to each feature, selecting either all features above some threshold or a a set number of the features with the best scores.
Wrapper methods use a predictive model as a black box, and its objective function as a way to score subsets of features, employing heuristic search methods to find said subsets.

3.2 Filter methods

3.2.1 Weighted sum of correlation and mutual information

The first filter algorithm implemented uses a weighted sum between two commonly used criteria for feature selection, correlation and mutual information between each feature and the target variable.
The score of each feature is computed as $\alpha * corr(X_i, y) + (1 - \alpha) * MI(X_i, y)$, where X_i is the feature, y is the target variable and α is the weight, which can be set by the user.
Additionally, the user can set their desired number of features and minimum score threshold.
The algorithm also has an auto-optimization feature, where the user can automatically run a grid search on α and the number of features n, as well as optionally the threshold, to find values that are as close to optimal as possible.

3.2.2 Variance Inflation Factor

The second filter algorithm uses Variance Inflation Factor (VIF) as its criteria. VIF gives a measure of how much variance a feature adds to a model due to multicollinearity with other features.
High VIF scores, typically 5 or higher, indicate high multicollinearity.
This algorithm allows the user to remove all features with a VIF score above or below a threshold or choose to keep n features with the best VIF scores.

3.3 Wrapper methods using deep reinforcement learning

Feature selection can be described as a prediction problem, where a model needs to predict which features will maximize an objective function. To keep feature selection and predictions on data as separate and thus interpretable, reinforcement learning was used (as direct gradient computation is not possible in this case). Both algorithms described in this subsection were implemented using the Stable-Baselines3 [3] library.

In the case of both algorithms, one episode was considered as 1 pass over the data (1 epoch), the step reward was the negative of the downstream model's objective function, and the action space is a binary vector of length n, the number of features.

3.3.1 Linear agent

The linear agent uses a batch of data, of a user specified size b, to predict a subset of features.

It uses a linear neural network to extract features from the data (the feature extractor), followed by linear neural networks for computing the action and value.

The environment is defined such that the observation space is a batch of data (1 dimensional $b * n$ vector).

At each step, a downstream model is trained on the training data with the selected subset of features.

This agent can use either the A2C or PPO policies.

3.3.2 Sequential agent

The sequential agent uses a clustering algorithm, either MeanShift or DBSCAN, to separate the data into sorted sequences by one of its features, for example "age" or "year built".

It can use either a linear network or a self-attentive bi-directional LSTM as its feature extractor, and can use either the A2C, PPO or RecurrentPPO policies.

At each step, the agent predicts features for one sequence, then trains a model on that sequence with the selected features.

3.4 Wrapper methods using SHAP values

The algorithms discussed in this section are based on the algorithm presented in [2], an iterative algorithm that uses SHAP values remove the feature determined to have the highest negative influence on the model.

At the end, it returns the subset of features which got the best score.

In this project, two variants of the algorithm were implemented, each expanding the algorithm's exploration of the search space.

Additionally, an auto-optimization method utilizing grid search to find good values of the algorithm's hyperparameters, q_{low} and q_{high} was implemented.

3.4.1 Branching variant

In this variant, the algorithm creates 2 branches at each step, one where the feature with the highest negative influence is removed, and one where the feature with the second highest negative influence is removed.

To prevent infinite loops, the algorithm maintains a closed list of all previously visited subsets of features.

3.4.2 Backtracking variant

In this variant, the algorithm can add 1 previously removed feature in each iteration, taking the subset which maximizes model performance to the next iteration.

To prevent infinite loops, the algorithm maintains a closed list of all previously visited subsets of features.

4 Experimental Evaluation

4.1 Datasets

The algorithms were evaluated on 4 datasets:

- House price dataset. The regression dataset used in class, with the same features as selected in the introductory lecture.
- Life Expectancy (WHO) dataset.. A regression dataset.
- Creditability - German Credit Data. A classification dataset.
- Titanic dataset, imported via the seaborn library. A classification dataset.

Regression datasets were evaluated using MSE, R^2, MAE, Mean Average Precision Error (MAPE) and Max Error (ME). Classification datasets were evaluated using log loss (LL), accuracy (acc), precision (pre), recall (rec) and F1 score.

All datasets went through the same preprocessing - categorical columns were 1-hot encoded, missing values were replaced with the mean of the column, and pairs of features with collinearity above 0.9 were removed, keeping the one with a higher correlation to the target variable. Finally, the data was split into a train, validation and test set.

The models used for evaluation were sklearn's linear regression and logistic regression.

4.2 Results - filter methods

The filter methods were compared to a baseline of a model without feature selection, as well as sklearn's filter methods: variance threshold (VT) with a threshold of 0.8 and select K best (sKb) with the criteria being Mutual Information, and the rest set to the default settings.

Algorithm	House prices						Life expectancy					
	MSE	R^2	MAE	MAPE	ME	n features	MSE	R^2	MAE	MAPE	ME	n features
Weighted sum	1.747670e+09	0.808718	24824.893832	0.143633	267826.667607	12	16.619741	0.818799	2.912961	0.045422	15.386732	13
VIF	1.903587e+09	0.791653	25041.658850	0.143412	303227.198129	9	16.702660	0.817895	2.933860	0.045631	14.948111	18
sklearn VT	1.920542e+09	0.789798	25147.118687	0.143930	302285.450520	7	16.547799	0.819583	2.949985	0.046042	14.321200	16
sklearn sKb	1.824412e+09	0.800319	24561.211463	0.144128	291729.031365	10	17.121958	0.813323	2.963201	0.046342	14.321200	10
None	1.824412e+09	0.803238	25263.788761	0.143753	264032.552235	24	16.702660	0.817895	2.933860	0.045631	14.948111	18

Algorithm	Creditability						Titanic					
	LL	acc	pre	rec	F1	n features	LL	acc	pre	rec	F1	n features
Weighted sum	7.929604	0.78	0.812500	0.902778	0.855263	32	5.606791	0.844444	0.805556	0.805556	0.805556	13
VIF	7.929604	0.78	0.804878	0.916667	0.857143	39	6.407761	0.822222	0.763158	0.805556	0.783784	13
sklearn VT	9.010913	0.75	0.758242	0.958333	0.846626	10	12.014551	0.666667	0.750000	0.250000	0.375000	2
sklearn sKb	9.010913	0.75	0.764045	0.944444	0.844720	10	6.808246	0.811111	0.756757	0.777778	0.767123	10
None	8.650477	0.76	0.800000	0.888889	0.842105	43	5.206305	0.855556	0.810811	0.833333	0.821918	27

As can be seen, in all but the Life expectancy dataset, weighted sum got better results than the sklearn algorithms, and always got results that are either slightly below or above the results of applying no feature selection (below indicates all features are at-least somewhat relevant, above indicates some features may have been noise). In the one case it did slightly worse, it was still very close to the best performing algorithm, and achieved those results with less features.

VIF also produced satisfactory results, with performance either similar or slightly below weighted sum, but still better than the sklearn algorithms in 2 of 4 datasets. The worse results may be due to VIF being not quite as fitting of a criterion for filter methods, as it does not reflect a direct relation to the target, but rather a negative influence on the model.

4.3 Results - wrapper methods using deep reinforcement learning

The linear agent is compared to a baseline without feature selection, and to sklearn's Sequential Feature Selection (SFS). All agents used the A2C policy.

Algorithm	House prices						Life expectancy					
	MSE	R^2	MAE	MAPE	ME	n features	MSE	R^2	MAE	MAPE	ME	n features
Linear agent	1.849943e+09	0.797525	25484.873657	0.142152	269775.219994	13	17.864656	0.805226	2.993281	0.046869	16.622377	14
SFS	1.818245e+09	0.800994	25181.713719	0.139444	265985.917502	12	17.269814	0.811711	2.984361	0.046360	15.594541	9
None	1.818245e+09	0.800994	25181.713719	0.139444	265985.917502	24	16.702660	0.817895	2.933860	0.045631	14.948111	18

Algorithm	Creditability						Titanic					
	LL	acc	pre	rec	F1	n features	LL	acc	pre	rec	F1	n features
Linear agent	7.569167	0.79	0.800000	0.944444	0.866242	28	6.007276	0.833333	0.783784	0.805556	0.794521	18
SFS	7.929604	0.78	0.804878	0.916667	0.857143	21	6.007276	0.833333	0.783784	0.805556	0.794521	13
None	8.650477	0.76,0	800000	0.888889	0.842105	43	5.206305	0.855556	0.810811	0.833333	0.821918	27

Results-wise, the linear agent performed worse than sklearn's SFS on all but the creditability dataset, but even then it chose a larger subset of features to achieve said result.

Generally, this algorithm did not lead to an improvement. This could be due to a myriad of reasons - improper use of RL algorithms for the scenario, not enough data to train the agent, improper set-up of the environment or reward computation, bad hyper-parameters for the RL algorithm, the neural network may not have had a fitting architecture, overfitting or underfitting on the training data, too few or too many training steps, or possibly other reasons.

The sequential agent was evaluated by the average, best and worst metrics on collections of models, where each model was trained over 1 of the sequences of the data. It was compared to the same sequences but without feature selection, as well as to SFS and no feature selection on the un-sequenced data. All agents used the A2C policy with a recurrent network.

Algorithm	House prices						Life expectancy					
	MSE	R^2	MAE	MAPE	ME	n features	MSE	R^2	MAE	MAPE	ME	n features
Sequencing and feature selection, avg	1.244608e+09	0.729320	20468.389875	0.123088	115741.743176	17	15.169477	0.820941	2.891378	0.044480	10.633530	15
Sequencing and feature selection, best	2.671738e+08	0.842390	12008.441447	0.091809	44460.767274	17	10.299113	0.858475	2.282176	0.035207	9.280224	15
Sequencing and feature selection, worst	2.768463e+09	0.591246	29037.955614	0.184504	250863.424557	17	19.270130	0.758802	3.341740	0.052951	13.419964	15
Sequencing and no feature selection, avg	1.219171e+09	0.731051	21030.464421	0.124247	110643.543849	24	15.330681	0.818877	2.914573	0.044863	10.609097	18
Sequencing and no feature selection, best	2.740208e+08	0.839982	12094.926858	0.091211	43822.696882	24	10.700767	0.859795	2.382483	0.036765	9.133920	18
Sequencing and no feature selection, worst	2.768463e+09	0.601062	31167.329215	0.175765	247530.791648	24	19.493988	0.757144	3.359446	0.053249	13.494354	18
SFS	1.818245e+09	0.800994	25181.713719	0.139444	265985.917502	12	17.269814	0.811711	2.984361	0.046360	15.594541	9
None	1.797745e+09	0.803238	25263.788761	0.143753	264032.552235	24	16.702660	0.817895	2.933860	0.045631	14.948111	18

Algorithm	Creditability						Titanic					
	LL	acc	pre	rec	F1	n features	LL	acc	pre	rec	F1	n features
Sequencing and feature selection, avg	10.376203	0.712121	0.728337	0.950000	0.809917	24	3.474087e+00	0.903614	0.863636	0.887097	0.875000	17
Sequencing and feature selection, best	6.007276	0.833333	0.885246	1.000000	0.892562	24	2.220446e-16	1.000000	1.000000	1.000000	1.000000	17
Sequencing and feature selection, worst	14.745131	0.590909	0.571429	0.900000	0.727273	24	6.948174e+00	0.807229	0.727273	0.774194	0.750000	17
Sequencing and no feature selection, avg	7.569167	0.692890	0.729129	0.883333	0.787042	43	3.256957e+00	0.909639	0.875000	0.887097	0.880952	27
Sequencing and no feature selection, best	7.393570	0.794872	0.879310	0.916667	0.864407	43	2.220446e-16	1.000000	1.000000	1.000000	1.000000	27
Sequencing and no feature selection, worst	14.745131	0.590909	0.578947	0.850000	0.709677	43	6.513913e+00	0.819277	0.750000	0.774194	0.761905	27
SFS	7.929604	0.78	0.804878	0.916667	0.857143	21	6.007276e+00	0.833333	0.783784	0.805556	0.794521	13
None	8.650477	0.760000	0.800000	0.888889	0.842105	43	5.206305e+00	0.855556	0.810811	0.833333	0.821918	27

First, observe the comparison between the metrics on sequenced data, with and without feature selection. From this, we can see that the agent would always either outperform or produce results very close to those produced with the full feature set.

However, when compared to the cases where sequencing was not used, the agent falls in performance in

all but the life expectancy dataset, where it seems to achieve better average performance (not counting the titanic dataset, as it seems that one of the sequences had 100% accuracy, which skewed the results). Additionally, the agent always picked the same number of features on all sequences, which is not necessarily desired behavior, and may be indicative of overfitting.
As such, it seems that the method's main downfall is the sequencing method, specifically MeanShift clustering on 1 feature, which is likely not a good fit for most data. Possible overfitting on the training data may have also hurt performance.
Overall, it seems that the method works well on sequenced data, and can potentially lead to general improvements if the above problems are fixed.

4.4 Results - wrapper methods using SHAP values

In this section, Base refers to the algorithm implemented in [2], while Branching and Backtracking refers to the mentioned variants. These methods were compared to sklearn's SFS, with the direction set to backwards (sequentially remove instead of sequentially add, like the implemented algorithms). All variants used the same values of q_{low} and q_{high}, set to 0.15 and 0.85 respectively.

Algorithm	House prices						Life expectancy					
	MSE	R^2	MAE	MAPE	ME	n features	MSE	R^2	MAE	MAPE	ME	n features
Base	1.791899e+09	0.803878	24631.900881	0.140530	266606.333915	19	19.549114	0.786861	3.136313	0.048357	17.225196	12
Branching	1.783826e+09	0.804761	24881.461113	0.139156	263939.340679	16	16.901200	0.815730	2.949745	0.046016	15.366151	13
Backtracking	1.969068e+09	0.784487	25759.967586	0.147003	285172.612390	12	16.839464	0.816403	2.931861	0.045565	15.380591	14
SFS	1.876361e+09	0.794633	25187.719956	0.138601	271365.350606	12	17.269814	0.811711	2.984361	0.046360	15.594541	9
None	1.797745e+09	0.803238	25263.788761	0.143753	264032.552235	24	16.702660	0.817895	2.933860	0.045631	14.948111	18

Algorithm	Creditability						Titanic					
	LL	acc	pre	rec	F1	n features	LL	acc	pre	rec	F1	n features
Base	8.650477	0.76	0.807692	0.875000	0.840000	30	6.007276	0.833333	0.783784	0.805556	0.794521	15
Branching	8.290040	0.77	0.788235	0.930556	0.853503	7	5.606791	0.844444	0.805556	0.805556	0.805556	12
Backtracking	9.010913	0.75	0.776471	0.916667	0.840764	14	5.606791	0.844444	0.823529	0.777778	0.800000	10
SFS	7.569167	0.79	0.814815	0.916667	0.862745	22	5.606791	0.844444	0.805556	0.805556	0.805556	14
None	8.650477	0.76	0.800000	0.888889	0.842105	43	5.206305	0.855556	0.810811	0.833333	0.821918	27

First, it can be easily be seen that the branching variant always improved on the results of the base method. It also beat sklearn's SFS in all datasets but the creditability dataset in terms of metrics, but even then the metrics were very close, while the branching variant used less than a third of the features, which may be more desirable, especially from an explainability standpoint. Its results were also always either close to or improved compared to when using all of the features.
The backtracking method got the best results on 2 datasets, and the worst results on 2 datasets. While the backtracking method is a powerful tool for exploring the search space and accounting for changes in

the influence of features when other features are removed, it is also more vulnerable to overfitting as a result. Where it had the worst performance, it likely overfitted on the (fairly small) validation set, thus leading to worse results on the test set.
Overall, the branching variant seems to be a general improvement, while the backtracking variant seems to provide the biggest improvement so long as it can avoid overfitting and work the worst when it does not.

5 Related Work

Numerous tools and algorithms exist for addressing automated feature selection, offering a diverse array of methodologies and techniques. The sklearn library, for instance, provides a comprehensive suite of feature selection algorithms [4], including those utilized for comparison in this project.
The main source of inspiration for this project is the paper "A survey on feature selection methods" [1], where many algorithms for feature selection as well as their principles are discussed. These include the principles behind filter methods and wrapper methods, as well as SFS And SFFS algorithms, all principles used and implemented in this project. Another paper that was a big source of inspiration for this project is "A feature selection method based on Shapley values robust for concept shift in regression"[2], which presents the algorithm for which 2 variants were developed in this project.
The solution presented in this project is different in that it uses different metrics for the filter methods than those presented in [1] and [4], introduces RL based methods for feature selection using linear networks and sequence modeling, and introduces variants to [2] based on the principles seen in [1].

6 Conclusion

Throughout this project, a diverse range of feature selection algorithms were developed and rigorously tested across multiple datasets. The experimental outcomes demonstrated the impact that feature selection techniques can have on the performance of predictive models. Notably, in most cases the model's performance was either improved or relatively unchanged after feature selection, underscoring the effectiveness of automated feature selection in optimizing model performance and understanding data.
Overall, this project has taught me the importance of feature selection as well as the principles behind performing effective feature selection, and has highlighted the role that proper feature selection can play in predictive tasks and understanding of the models and data.
Additionally, I learnt reinforcement learning for this project, which gave me a chance to dive into this vast field and broadening my understanding of machine learning.

References

[1] Chandrashekar, Girish, and Ferat Sahin. ”A survey on feature selection methods.” Computers & electrical engineering 40.1 (2014): 16-28.

[2] Sebastián, Carlos, and Carlos E. González-Guillén. ”A feature selection method based on Shapley values robust to concept shift in regression.” arXiv preprint arXiv:2304.14774 (2023).

[3] Stable-Baselines3, a RL library

[4] sklearn feature selection