Advanced Machine Learning - Project 2

Michał Gromadzki

June 2024

1 Introduction

This project was devoted to building the most effective model. The model should use as few features as possible and have the highest percentage of correctly classified people to take a certain offer. To efficiently select such model and features the following preprocessing and, more importantly, a custom metric has been implemented.

1.1 Preprocessing

Firstly the entire dataset is split into train and validation sets with proportions of 80:20. Then all features with correlation higher than $TH = 0.9$ are removed using the following algorithm.
Algorithm for removing variables with high correlation:

- Algorithm is iteration-based, in each iteration 1 variable can be removed
- In each iteration:
    1. Calculate the correlation between all variable pairs and take the absolute value of it
    2. Exclude correlation equal to 1, occurs on the diagonal of the correlation matrix
    3. Remove one of the features from the pair with the highest correlation
- Algorithm stops when there are no variable pairs with correlation over TH

This algorithm ensures that the minimal number of required features is removed.

Lastly before passing the data to any model, all features are scaled using StandardScaler from sklearn package.

1.2 Custom Metric

The custom metric used for selecting the best-performing models and feature sets is calculated as follows:

- Calculate the probability of belonging to class 1 for all observations in the validation set
- Select 20% of observations with the highest probability
- Calculate how many observations were correctly selected - $corr$
- Calculate the final value of the metric: $corr \cdot 5 \cdot 10 - 200 \cdot num$, where num is the number of used features

In the second step, the 20% of the validation set is inspired by the original task, where we have to identify 1000 customers out of a possible 5000, which is equal to 20%. As the validation set is 5 times smaller than the test set we need to multiply the value of $corr$ by 5 to maintain the balance between profit from correctly selecting a customer and the cost of features. This is why, in the last step, $corr$ is multiplied not only by 10 but also by 5. Note that this metric directly corresponds to the amount of money a model is expected to earn.

2 Experiments

Six feature selection methods were tested. For each method the following models were tested:

- NN - Neural network
- LR - Logistic Regression
- XGB - XGBoost
- RF - Random Forest
- SVC - Support Vector Machine
- LDA - Linear Discriminant Analysis

Note that for each of the following feature selection methods it is possible to select an exact number of features to choose. This property allows to iterate through the number of selected features, starting from 1 and ending at 25. It is not profitable to select more than 25 features since even achieving 100% accuracy would yield at most $1000 \cdot 10 - 26 \cdot 200 = 4800$ which is average profit.

To select the best-performing model and feature combination values of Custom Metric will be used, and in case of a tie accuracy score will be used.

2.1 Correlation

Feature selection based on correlation works as follows. Firstly, calculate the correlation of all features to the target variable. Then sort all the variables in descending order based on the calculated correlation. Lastly, in each iteration select n first features from the ordered list. In each iteration select one more feature than in the previous one.

2.2 Mean

This feature selection method focuses on selecting features with the biggest difference in the distribution in each of the target variable classes. Firstly, it calculates the means for all features in each target variable class. Then it calculates the absolute value of the difference between both means and divides it by the means of the feature calculated on the entire dataset. Then it iterates through the ordered feature list similarly to the previous method. Note that in this step we perform feature selection before transforming features with StandardScaler.

2.3 SelectKBest

SelectKBest feature selection method selects k best features according to some scoring function, where the user sets k. In this case, the scoring function is ANOVA F-value. Again after each iteration, one more feature is selected than in the previous iteration.

2.4 SelectFromModel

This method selects features based on the feature importance calculated by a provided estimator. In this case, RandomForest was used. Similar to the previous methods in each iteration one more feature is selected. Note that since feature selection is based on an estimator feature scaling was performed before the selection process.

2.5 SequentialFeatureSelector

Sequential Feature Selector adds features to form a feature subset greedily. Due to the high computation complexity of this method, Logistic Regression was used as the estimator. Again, the feature selection process is based on an estimator, hence feature scaling was performed before selection.

2.6 Recursive Feature Elimination

Since this method is computationally expensive firstly the number of considered features is reduced to 50 by selecting the features with the highest mutual information with the target variable. Then the Recursive Feature Elimination is performed. Again, due to high computation complexity, Logistic Regression was used as the estimator and the feature transformation was performed before the selection process.

3 Results

The results of the described feature selection methods have been provided below. Table 1 includes mean accuracy for each model for each method.

	NN	LR	XGB	RF	SVM	LDA
Corr	0.520	**0.529**	0.507	0.513	0.508	**0.529**
Mean	**0.517**	0.515	0.511	0.510	0.498	0.515
KBest	0.521	**0.529**	0.508	0.514	0.508	**0.529**
SFM	0.623	0.510	0.638	0.657	**0.677**	0.510
Seq	0.554	0.513	0.557	**0.566**	**0.566**	0.513
RFE	0.542	0.536	0.530	0.536	**0.550**	0.536

Table 1: Mean accuracy for each model for each method

Table 2 includes max accuracy for each model for each method.

	NN	LR	XGB	RF	SVM	LDA
Corr	**0.550**	0.546	0.533	0.533	0.526	0.546
Mean	0.542	0.531	**0.546**	0.532	0.536	0.532
KBest	0.533	**0.546**	0.528	0.543	0.526	**0.546**
SFM	0.685	0.530	0.663	0.694	**0.724**	0.529
Seq	**0.599**	0.541	0.579	0.586	0.598	0.541
RFE	**0.608**	0.556	0.601	0.601	0.598	0.557

Table 2: Max accuracy for each model for each method

Table 3 includes mean values of Custom Metric for each model for each method.

	NN	LR	XGB	RF	SVM	LDA
Corr	2888	2902	2710	2694	2674	**2904**
Mean	**2728**	2544	2558	2576	2578	2542
KBest	2800	2902	2730	2792	2674	**2904**
SFM	4452	2876	4876	4864	**5214**	2884
Seq	3630	3092	3718	3898	**3906**	3094
RFE	3212	3042	2990	3092	**3516**	3040

Table 3: Mean custom metric for each model for each method

Table 4 includes max values of Custom Metric for each model for each method.

	NN	LR	XGB	RF	SVM	LDA
Corr	5200	4800	5250	**5400**	4950	4800
Mean	**5300**	5250	5200	5100	4750	5250
KBest	4800	4800	5250	**5400**	4950	4800
SFM	6350	5450	6700	6800	**7250**	5450
Seq	6800	6000	6100	6100	**6950**	6000
RFE	**5350**	5200	5050	4850	5150	5200

Table 4: Max custom metric for each model for each method

SVM consistently excels, particularly with the SelectFromModel and Sequential Feature Selection methods, achieving the highest mean and maximum accuracy, as well as custom metric values. Neural Network also performs well, especially in terms of mean Custom Metric. Logistic Regression and Linear Discriminant Analysis show notable accuracy with correlation-based and KBest methods. Random Forest and XGBoost have strong performances in specific cases but exhibit less consistency across all metrics.

4 Final model selection

The best-performing model was SVM trained on a subset of 4 features selected by SelectFromModel feature selection method. To ensure finding the best model and features combination a smaller experiment was conducted. Features were selected the same way as in the 2.4, however limited to 5 features to reduce computation complexity. This reduction in computation complexity allowed for increased search in the model's parameter space. For each feature number, from 1 to 5, 70 models with different hyper-parameters were tested. Moreover, the described experiment was also conducted a second time with the inclusion of interactions of selected variables. Metrics on the validation set and characteristics of the two best-performing models are provided in Table 5.

	num_features	columns	accuracy	CustomMetric	C	kernel	degree	Interactions
1st Model	4	100, 102, 103, 105	0.639	**7400**	0.1	poly	4	No
2nd Model	5	100, 102, 103, 104, 105	**0.679**	7300	0.001	linear	3	Yes

Table 5: Metrics and hyper-parameters for two best-performing models

In the end, the second Model was selected. Even though it has a bit smaller value of Custom Metric it has much greater accuracy. Due to higher accuracy, it is expected that the second model will outperform the first one on the bigger test set. Therefore the selected features are: 100, 102, 103, 104, 105. However, note that the features here are indexed from 0 so in the actual answer all features indexes are increased by 1.

In conclusion, the selected model was SVM with the following hyper-parameters:

- C - 0.001
- kernel - linear
- degree - 3
- gamma - auto

Not mentioned parameters are set to default values. The model was trained on the following features: $101, 103, 104, 105, 106$ and their interactions. It is expected that the model will earn around 7500.

5 Implementation

Each feature selection method was implemented in its .py file. Running each of the files results in creating a single .csv file with results for the selected features selection method. The .csv file contains the following columns: num_features, model, accuracy, metric, method. All helper functions such as data loading and calculation of the custom metric were implemented in utils.py. Lastly, the hyperparameter-tuning and final model is implemented in final.ipynb. For all the randomized operations a set seed of 1337 was used.