Second Home Assignment

Aprendizagem Automática

2022/2023

João Fé
fc60427@alunos.fc.ul.pt
16 WHs

Duarte Raposo
f59099@alunos.fc.ul.pt
16 WHs

Daniel Cabral
fc54790@alunos.fc.ul.pt
16 WHs

1 Methodology

2 Implementation

2.1 Data preprocessing

Just after loading the data from the CSV file, we do three things. First, as the data was loaded from the CSV file, it was also shuffled to make sure the categories are well distributed later on between the training set and the testing set. Second, every continuous feature was normalized. Third, every categorical featured (which are originally represented as decimal numbers) was casted to a string type, just to be sure that these features are not going to be interpreted as continuous values by any code of Scikit Learn that we might execute later. One-hot encoding is done only after feature selection and used only when shown favourable by tests ran.

2.2 Feature selection

For feature selection, a machine learning model was used, namely Random Forest. Because the model does not work well with N/A values. To solve the N/A value issue, these were replaced in the numerical features by their mean, and attributed a ”dummy” class with the value of -1 when categorical. If these injected values resolving the N/A issue ruin a feature, the feature selection Random Forest model should drop it.

The Random Forest Feature Selector non-deterministically returned between 12 and 15 features. As 3 features weren't appearing every time the feature selector was ran, they were discarded, filtering the features to a max of 12.

With this, the features selected were 'nHM', 'F04', 'NssssC', 'nCb', 'F03', 'F02_CN', 'SpMax_L', 'SM6_L','SpPosA_B', 'SpMax_B', 'SM6_B', 'nX'.

2.3 Models training

First of, the data is split into a training set, which uses a 75%-fix-part of the entire dataset, and the remaining 25% were reserved for evaluation.

As for the selection of the model and testing, we selected what models would be interesting to test in our current dataset, and ran some simple tests to see how each model would do without any hyper-parameters. This selection was broad, and even included the MultiLayer-Preceptron, one of which we did not intend on optimizing, but tested out of curiosity on how it would do with the given dataset.

One model that we found out later even getting our attention — the Histogram-Based Gradient Boosting. This model didn't even need any data preparation and showed promising results, but we did not end up optimizing it since it wasn't on our list of models.

Table 1: Tested models and hyperparameters.

Model	Hyper-parameters		
Random Forest	Criterion {gini, entropy, log_loss}	N Estimators {125,126,127,128,129,130}	Maximum depth {Inf.}
SVC	Kernel {linear, poly, rbf}	tol {0.0000001,0.01, 0.1, 0.5}	C {0.5,1,4,6,8,12}
Linear SVC	loss {hinge,square_hinge}	tol {0.0000001,0.01, 0.1, 0.5,1}	C {0.5,1,4,6,8,12}
Gaussian Naive Bayes	Var Smoothing {1e-9,1e-07,1e-06,1e-02}		
Bernoulli Naive Bayes	Alpha {1e-10,1e-8,1e-5,1e-3,0.1}	Binarize {1,1.5,1.75,2,2.25,2.5,2.75,3,4}	
Gradient Boosting	Learning Rate {0.1,0.2,0.5,1}	N Estimators {50,100,150,200,1000}	Maximum depth {2,3,10,30}
K-nearest neighbors	N Neighbors {3,4,5,6,7,8,10}	Weight {uniform, distance}	Algorithm {auto,ball_tree,kd_tree,brute}

With this first round of testing, we decided on optimizing the following models:

- Random Forest Classifier
- SVM (SVC and Linear SVC)
- Naive Bayes (Gaussian and Bernoulli)
- Gradient-Boost
- K-Nearest Neighbours

For each of the models, the optimization was made using a fitness calculation that computes the mean of predefined metrics. As all the metrics used in classification are usually percentages, we always maintained the model with the highest (closest to 100%) fitness score. As for what metrics to use, we decided on having an all-rounded approach that used the Accuracy, F1 Score and Mathews Correlation Score. Some hyperparameters used in the models' optimization can be observed in the Table 1.

3 Results

3.1 Optimized Models Obtained

3.1.1 Random Forest Classifier

Best Hyperparams {'n_estimators': 129, 'criterion': 'log_loss', 'max_depth': None, 'min_samples_split': 2, 'min_samples_leaf': 1, 'min_weight_fraction_leaf': 0, 'max_features': 'log2', 'max_leaf_nodes': None, 'min_impurity_decrease': 0, 'bootstrap': True, 'oob_score': True, 'n_jobs': None, 'random_state': None, 'verbose': 0, 'warm_start': False, 'class_weight': None, 'ccp_alpha': 0, 'max_samples': None}

The **Accuracy** is: **0.9728**
The **Precision** is: **0.9897**
The **Recall** is: **0.9786**
The **F1 score** is: **0.9841**
The **Matthews correlation coefficient** is: **0.8911**

3.1.2 Support Vector Machines

SVM for Classification (SVC)

Best Hyperparams = {'C': 4, 'kernel': 'rbf', 'degree': 3, 'gamma': 'scale', 'coef0': 0, 'shrinking': True, 'probability': False, 'tol': False, 'cache_size': 200, 'class_weight': None, 'verbose': False, 'max_iter': -1, 'decision_function_shape': 'ovr', 'break_ties': False, 'random_state': None}
The **Accuracy** is: **0.9606**
The **Precision** is: **0.9928**
The **Recall** is: **0.9620**
The **F1 score** is: **0.9772**
The **Matthews correlation coefficient** is: **0.8383**

Linear SVM for Classification (LinearSVC)

Best Hyperparams = {'penalty': 'l2', 'loss': 'squared_hinge', 'dual': True, 'tol': 0.1, 'C': 12, 'multi_class': 'ovr', 'fit_intercept': True, 'intercept_scaling': 1, 'class_weight': None, 'verbose': 0, 'random_state': None, 'max_iter': 1000}
The **Accuracy** is: **0.9483**
The **Precision** is: **0.9887**
The **Recall** is: **0.9523**
The **F1 score** is: **0.9702**
The **Matthews correlation coefficient** is: **0.7849**

3.1.3 Naive Bayes

Gaussian Naive Bayes

Best Hyperparams = {'priors': None, 'var_smoothing': 0.01} The **Accuracy** is: **0.9413**
The **Precision** is: **0.9742**
The **Recall** is: **0.9574**
The **F1 score** is: **0.9658**
The **Matthews correlation coefficient** is: **0.7611**

Bernoulli Naive Bayes

Best Hyperparams = {'alpha': 1e-10, 'binarize': 1.75, 'fit_prior': True, 'class_prior': None}
The **Accuracy** is: **0.9150**
The **Precision** is: **0.9608**
The **Recall** is: **0.9405**
The **F1 score** is: **0.9505**
The **Matthews correlation coefficient** is: **0.6505**

3.1.4 Gradient-Boost

Best Hyperparams = {'loss': 'exponential', 'learning_rate': 1, 'n_estimators': 1000, 'subsample': 1, 'criterion': 'squared_error', 'min_samples_split': 2, 'min_samples_leaf': 1, 'min_weight_fraction_leaf': 0, 'max_depth': 10, 'min_impurity_decrease': 0, 'init': None, 'random_state': None, 'max_features': None, 'verbose': 0, 'max_leaf_nodes': None, 'warm_start': False, 'validation_fraction': 0.1, 'n_iter_no_change': None, 'tol': 0.0001, 'ccp_alpha': 0}
The **Accuracy** is: **0.9693**
The **Precision** is: **0.9887**
The **Recall** is: **0.9756**
The **F1 score** is: **0.9821**
The **Matthews correlation coefficient** is: **0.8767**

3.1.5 K-Nearest Neighbours

Best Hyperparams = {'n_neighbors': 4, 'weights': 'uniform', 'algorithm': 'auto', 'leaf_size': 20, 'p': 1, 'metric': 'minkowski', 'metric_params': None, 'n_jobs': None}
The **Accuracy** is: **0.9641**
The **Precision** is: **0.9835**
The **Recall** is: **0.9745**
The **F1 score** is: **0.9790**
The **Matthews correlation coefficient** is: **0.8563**

4 Discussion and Conclusion

With all the testing and model optimizing, we got to the conclusion that the best model we could find was a Random Forest. It achieved the following results displayed in the Confusion Matrix 2.

		True Class	
		Negative	Positive
Predicted Class	Negative	146	13
	Positive	25	957

Table 2: Confusion matrix for Random Forest.

And, computing the metrics, we obtain:

- The **Accuracy** is: **0.9667**
- The **Precision** is: **0.9866**
- The **Recall** is: **0.9745**
- The **F1 score** is: **0.9805**
- The **Matthews correlation coefficient** is: **0.8662**

We also observed that it is not completely deterministic, but the results still remain the best out of all the other models that were tested and optimized.

But, even though this Random Forest Shows excellent results, we need to keep in mind that the complexity of the model still should influence the decision on what we want to use. For several use-cases, this option could change according to the desired Recall/Precision scores we want. Thus, we shouldn't discard all the remaining models, that also showed promising results and are still viable to being picked according to needs.