CS577 - Machine Learning – Assignment 5

Alexandros Angelakis csdp1363, angelakis@csd.uoc.gr

Model Selection (Programming)

For my model selection procedure, I chose two classifiers: logistic regression and random forest, using the sklearn implementations. Each classifier had three hyperparameters to be tuned, with each hyperparameter having three possible values. Additionally, I ran the classification models both with and without data standardization. To determine the best model with the optimal hyperparameters, I evaluated their performance using the ROC AUC metric. For more details about the configurations, please refer to my code. It is highly explanatory.

Exploratory Data Analysis

In our dataset, the majority of the features are categorical. With a categorical threshold set to 12, there are 45 categorical features and 5 continuous features. By adjusting the value of the categorical threshold, you can increase or decrease the number of continuous features. Classifiers that perform well with mostly categorical data include tree-based classifiers (e.g., Decision Trees, Random Forests, Gradient Boosting), Naïve Bayes classifiers, Logistic Regression and others.

In Tables 1, 2, and 3, we can see the features with the highest values of correlation, coefficient of variation ratio (CV), and dominance ratio. The coefficient of variation ratio is calculated only for continuous data, while the dominance ratio is computed exclusively for categorical data. A low CV indicates that the data are more stable, whereas a high CV suggests that the data are less stable or noisier. A high dominance ratio (close to 1) signifies significant imbalance in the data (these values represent only the five highest values across all available features).

Feature	Correlation
5	0.407340
4	0.299083
10	0.272154
0	0.267631
12	0.236315

Table 1: Correlation with Target.

Feature	Std/Mean
13	2.199363
3	0.784114
23	0.536183
17	0.530758
6	0.525345

Table 2: Std/Mean Ratio.

Feature	Dominance
11	0.947704
10	0.915871
8	0.854934
1	0.735789
9	0.686221

Table 3: Dominance Ratio.

Statistic	5	4	13	6	11	10
Count	2199	2199	2199	2199	2199	2199
Mean	2.054116	4.697135	2.334243	5.702137	1.470668	1.757162
Std	1.183477	2.978715	5.133847	2.995591	2.004474	2.498801
Min	1	1	1	3	1	1
25%	1	2	1	3	1	1
50%	2	5	1	5	1	1
75%	3	8	1	8	1	1
Max	7	10	39	14	10	10

Table 4: Summary statistics for features 5, 4, 13, 6, 11, and 10.

The exercise requires us to draw histograms for five features; however, I will select six features instead: two from each table. From the correlation table, I will select the two features with the highest correlation with the target. From the coefficient of variation ratio table, I will choose the features with the highest and lowest CV. Finally, from the dominance table, I will select the two features with the highest dominance (imbalance).

As shown in Figure 1, the selected features from the dominance ratio table are indeed imbalanced, with their values concentrated into two categories: one with a large number of occurrences and another with very few.

The selected features from the coefficient of variation ratio table also demonstrate the expected behavior. For a high CV (indicating that the standard deviation is large compared to the mean), the histograms show high variability and less stability. This is evident in Feature 13, where the wide spread of values creates significant dispersion relative to the mean. Conversely, for a low CV, the standard deviation is small compared to the mean, suggesting greater stability in the data. This is confirmed for Feature 6, where the spread is limited, and variability is lower.

The selected features from the correlation table do not provide any meaningful insights, making their histograms uninformative and not particularly useful.

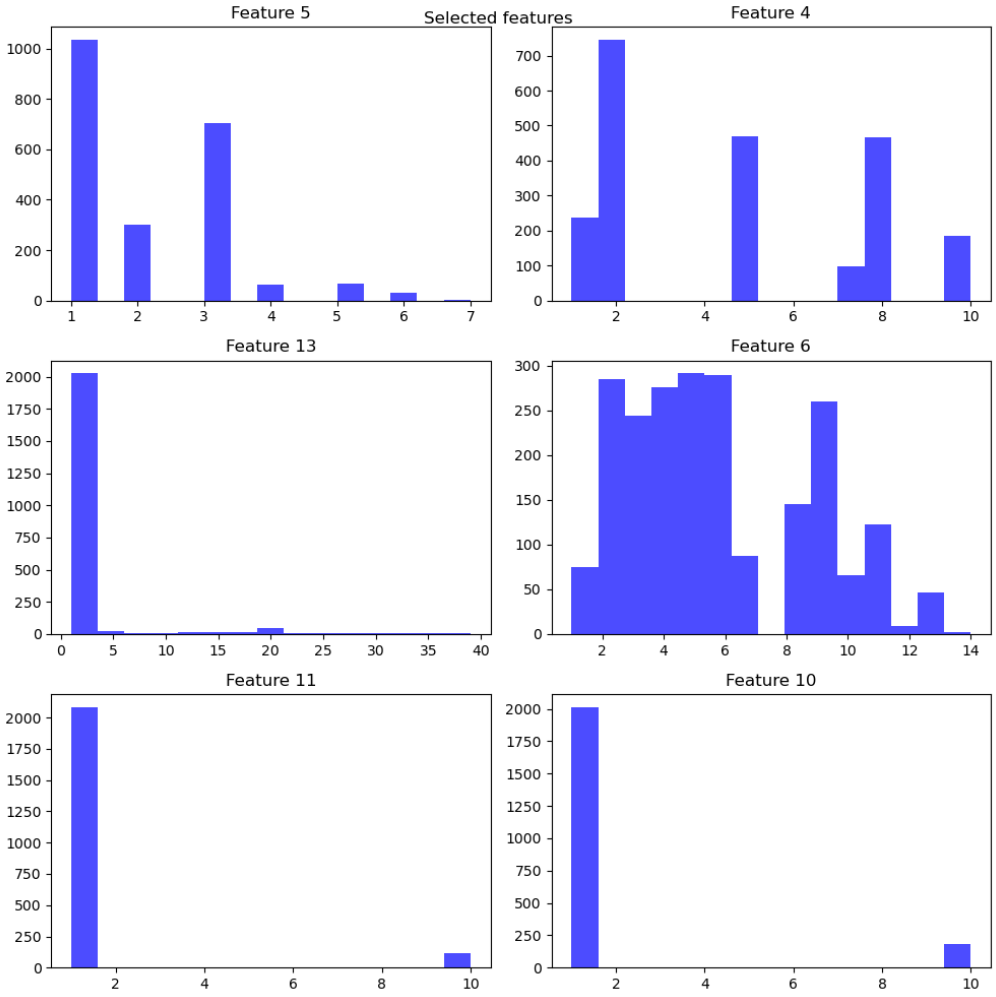

Figure 1: Histograms of the selected features

Model Selection

I implemented the function `create_folds(arguments)` to split the data matrix into k stratified folds. For this function, and this function only, I utilized the `sklearn` implementation to generate the k stratified splits.

For the cross-validation step, I implemented the function `CV(arguments)`, which identifies the best configuration and evaluates its performance using the cross-validation procedure. The best configuration is determined by comparing the average ROC AUC across all folds.

Once the best configuration is identified, I construct the final model and train it on the entire dataset. The configurations used to identify the best model are listed in Table 5.

Standardization	**Classifier**	**Hyperparameters**
True	RandomForestClassifier	n_estimators: [100, 200, 500] max_depth: [None, 10, 20, 30] min_samples_leaf: [1, 2, 4] min_samples_split: [2, 5, 10]
False	RandomForestClassifier	n_estimators: [100, 200, 500] max_depth: [None, 10, 20, 30] min_samples_leaf: [1, 2, 4] min_samples_split: [2, 5, 10]
True	DecisionTreeClassifier	max_depth: [None, 5, 10, 15] min_samples_leaf: [1, 5, 10] min_samples_split: [2, 5, 10] max_leaf_nodes: [None, 10, 20, 50]
False	DecisionTreeClassifier	max_depth: [None, 5, 10, 15] min_samples_leaf: [1, 5, 10] min_samples_split: [2, 5, 10] max_leaf_nodes: [None, 10, 20, 50]

Table 5: Configurations used for my experiments

With $k = 5$, the majority of the times I run the code, the best configuration is:

Standardization	True
Model	RandomForestClassifier
Parameters	{'max_depth': 20, 'min_samples_leaf': 4, 'min_samples_split': 10, 'n_estimators': 500}
Average AUC	0.886

Table 6: Best Configuration for the Model from CV

I also tried using nested cross-validation, which provides a way to reduce the bias in combined hyperparameter tuning and model selection. The best configuration this time is:

Standardization	False
Model	`RandomForestClassifier`
Parameters	{'max_depth': 20, 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 200}
Average AUC	0.918

Table 7: Best Configuration for the Model from nested CV

As we can see from Tables 6 and 7, the AUC obtained from nested CV is significantly higher than the AUC obtained from non-nested CV. This is because nested CV allows for better hyperparameter selection. Additionally, this result suggests that the non-nested CV approach may have overfitted during hyperparameter tuning or evaluation.

Computing the out-of-sample performance

Given the hold-out test set, the out-of-sample ROC AUC is approximately 0.878, while the ROC AUC obtained from cross-validation is approximately 0.886. As we can see, the out-of-sample ROC AUC is slightly lower than the cross-validation ROC AUC. The small drop from 0.886 (cross-validation) to 0.878 (out-of-sample test set) indicates that the model generalizes well and is not significantly overfitting to the training data. This close agreement between the two values demonstrates the robustness and reliability of the model's performance.

For the model selected using nested cross-validation, the out-of-sample ROC AUC is approximately 0.880, while the ROC AUC obtained from cross-validation is approximately 0.918. The model performs slightly better than a trivial classifier on unseen data, with a small and expected generalization drop. This indicates that the model is robust and well-optimized.

The ROC curve

The trivial (naive) classifier is a model that always predicts the the majority class of the training data. This results in an AUC of 0.50, with a linear ROC curve.

In Figure 2, we can see the ROC curves of the naive classifier and the best model obtained through non-nested cross-validation. Similarly, in Figure 3, we can see the ROC curves of the naive classifier and the best model obtained through nested cross-validation.

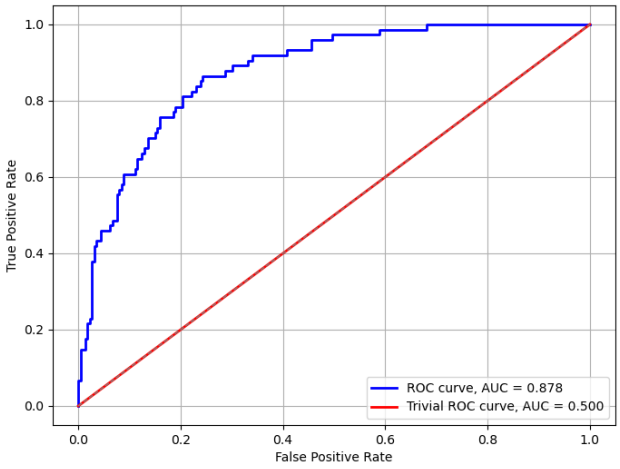

Figure 2: ROC Curves of non-nested CV

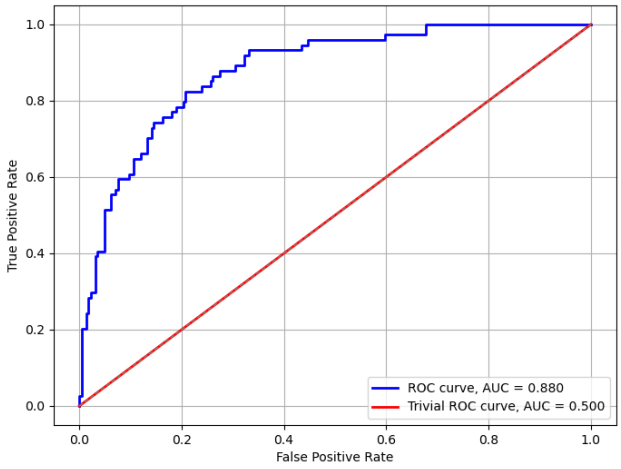

Figure 3: ROC Curves of nested CV