Artificial intelligence Project3: Report

Malik DAHMANI

June 22, 2024

Contents

1 Implementation Details

1.1 Preprocessing Techniques

First, data is loaded from Excel files using pd.read_excel. The first unnamed column is removed and I define a list of feature columns based on the data dictionary .

Then, for handling the Missing Data, I first remove all the features with a missing rate of 30% or higher. This threshold was chosen to retain a balance between having sufficient data and not losing too many features. For missing values in the remaining features, I use the Iterative Imputer (IterativeImputer). This technique, based on a form of multiple imputation, iteratively models each feature with missing values as a function of other features.

To continue, for the feature ranking and selection I apply the ReliefF algorithm to rank the importance of features. This is a robust feature selection method that is particularly effective in dealing with feature interactions. I then select the top 50 features based on their importance scores.

Finally, the cleaned and imputed data is split into training and validation sets using an 80-20 split with train_test_split.

1.2 Classifier Architectures and Hyperparameters

I use base classfier such as:

- Gaussian Naive Bayes
- K-Nearest Neighbors
- Logistic regression
- Random Forest

And one Meta-classifier, Support Vector Classifier (SVC), configured with probability=True to enable probability estimates, facilitating the stacking process

1.3 Hyperparameters

The hyperparameters I used are:

- IterativeImputer: random_state=42 for reproducibility.
- LogisticRegression: max_iter=10000 to ensure convergence.
- RandomForestClassifier: random_state=42 for reproducibility.
- SVC: probability=True to enable probability outputs, random_state=42 for reproducibility.

2 Discussion

2.1 Implementation challenge

Handling missing data

Choosing an appropriate threshold for feature cleaning was critical. A too high threshold would remove potentially informative features, while a too low threshold could introduce noise from excessive missing data. Moreover, imputing missing values accurately is crucial, and IterativeImputer was chosen for its effectiveness in handling complex datasets with potential inter-feature relationships.

Feature selection

Selecting the optimal number of features to retain was challenging. The ReliefF algorithm helped rank features, but the choice of the top 50 features was somewhat arbitrary and could be tuned further.

Model complexity

Balancing the complexity and interpretability of the stacking ensemble was a concern. Using SVC as a meta-classifier adds robustness but also computational complexity.

2.2 Key insights

Ensemble learning

Stacking multiple classifiers enhanced predictive performance by leveraging the strengths of different models.

Feature selection

Proper feature selection and imputation techniques significantly impacted model performance, highlighting the importance of robust preprocessing.

Model Interpretability

While the stacking model is powerful, its interpretability decreases compared to simpler models, posing a trade-off between performance and transparency.

Overall, the implemented model demonstrated robust performance in predicting the target variable (on the train set), supported by comprehensive preprocessing and advanced ensemble techniques.