Bank Term Deposit Subscription Prediction

Introduction

The code and results of predicting whether the bank's credit card customer will subscribe to a term deposit are shown in this report. The report is of the "Code_Sri_Manikanta_Arja.ipynb" notebook, which employs both Random Forest and Support Vector Classifier (SVC) models.

Data Preparation

The code performs the following data preparation operations:

- **Data Loading**: pandas is utilized to load the dataset "bank.csv".
- **Encoding Categorical Variables**:
    - The target variable 'Y' is encoded as binary (Yes: 0, No: 1) with other categorical features ('default','loan','housing').
    - The categorical features ('month', 'poutcome') is encoded with ranking with map.
    - Other categorical features ('job', 'education', 'marital', 'contact',) are one-hot encoded using get_dummies.
- **Feature Scaling**: Numerical features are scaled using StandardScaler.
- **Data Splitting**: The data is split between train and test sets (70% train, 30% test).
- **Handling Class Imbalance**: SMOTE is applied to the training data to address class imbalance.

Model Hyperparameter Tuning

The code both tests and trains Support Vector Classifier (SVC) and Random Forest models.

- **Random Forest**
    - Hyperparameter Tuning:
        - classification__n_estimators: How many trees in the Random Forest. This is tuned because it has a significant impact on how well the model performs. It may make it more accurate to use.
        - The code finds the following values for classification__n_estimators: [100, 150, 200, 250, 300, 350, 400, 450, 500].
        - 5-fold cross-validation is used. This means the training data is divided into five sets. Four sets are utilized to train the model and the fifth one to test it, and the exercise is done five times, where each set is used

once as the validation set. This helps in achieving a valid estimate of how good the model performs.
        - The recall is used as the scoring function during cross-validation. This means that the grid search selects the value of n_estimators that gives the best recall score.
    - Evaluation: A Random Forest model is trained with n_estimators=250 and its performance on the test set is evaluated. The following metrics are reported:
        - Accuracy: 0.8975
        - Recall: 0.9423
        - Precision: 0.9431
        - F1-score: 0.9427

- **Support Vector Classifier (SVC)**
    - Hyperparameter Tuning:
        - classification__kernel: This determines the kind of kernel function to utilize, which controls what shape the decision boundary will have. Different kinds of kernels are able to identify different types of relationships in the data.
        - classification__C: This is the regularization term. It controls the tradeoff between a smooth decision boundary and a low training error. Low C means higher margin, and this may prevent overfitting, whereas a highC concerns well-fitting the training set.
        - GridSearchCV is used, similar to Random Forest.
        - The code searches the following values:
        - classification__kernel: ['linear', 'poly', 'rbf', 'sigmoid']
        - classification__C: [0.001, 0.01, 0.1, 1, 10, 100]
        - 5-fold cross-validation is used.
        - The recall metric is used for scoring.
    - Evaluation: An SVC model is optimized with the best parameters C=0.001 and kernel='poly', and its performance over the test set is:
        - Accuracy: 0.8931
        - Recall: 0.9950
        - Precision: 0.8968
        - F1-score: 0.9434

Choice of Evaluation Metric

Recall is a suitable metric for model evaluation in this case. The rationale is that the bank wants to predict whether a client will subscribe to a term deposit. In such business scenarios, the cost of failing to identify a potential term deposit (false negative) might be higher than the cost of incorrectly classifying a non-subscriber as a subscriber (false positive). Recall measures the ability of the model to find all the actual positive cases, so maximizing recall is crucial to minimize the number of potential subscribers that are missed.

Overfitting avoidance mechanism

- Feature Selection
    - Random Forest Feature Importance: Removed features with low importance to reduce noise and improve generalization.
    - Recursive Feature Elimination : Remove the least important features gradually as the model is trained.

- Support Vector Classifier (SVC) Regularization
    - Random Forest Feature Importance: Removed features with low importance to reduce noise and improve generalization.
    - The C parameter was optimized to prevent overfitting. A lower C value increases the margin but may lead to underfitting.
    - The gamma parameter in the RBF kernel was tuned to avoid an overly complex model.

- Cross-Validation
    - K-Fold Cross Validation was used to ensure the model generalizes well to unseen data.

Results analysis

To decide which model to deploy, we need to consider their performance on the test set. Here's a summary:

- **Random Forest:**
  - Accuracy: 0.8975
  - Recall: 0.9423
  - Precision: 0.9431
  - F1-score: 0.9427

- **Support Vector Classifier (SVC):**
  - Accuracy: 0.8931
  - Recall: 0.9950
  - Precision: 0.8968
  - F1-score: 0.9434

- **Recommendation:**
  - If the aim is to capture as many potential term deposits as possible (i.e., minimize false negatives), SVC is recommended. SVC has significantly higher recall (0.9950) compared to Random Forest (0.9423). That is, SVC is better at capturing the customers who will take term deposits.
  - If the issue is to be more certain in all the positive predictions, then Random Forest is the recommended option. Random Forest is more accurate (0.9384) than SVC (0.9108). This means that when Random Forest predicts that a customer will take term deposits, there are greater chances of being correct.
  - Here, in this specific scenario, the F1-score is very close.
- RFC showed no signs of overfitting after hyperparameter tuning.
- SVC exhibited slight underfitting, as indicated by its lower recall and overall performance compared to RFC.
- Possible reasons for underfitting in SVC:
  - The selected kernel or hyperparameters may not be capturing complex decision boundaries effectively.
  - The regularization parameter C might be too high, causing a simplistic decision boundary.
  - The dataset may have features that are not well-separated in a linear space, making SVC less effective.
  - Feature scaling may have affected SVC's ability to find optimal hyperplanes.