Unit 5 Assignment: Comparing Ensemble Models

Juliana Noronha

University of Maryland, Global Campus

DATA 660: Advanced Topics in Machine Learning

Dr. Ed Herranz

April 15, 2025

Introduction

This report compares the performances of three types of ensemble models (bagging, boosting and stacking) and two SVM implementations (linear and nonlinear) against the same dataset. The implementation details of the SVMs were included in another assignment (Unit 3) and will thus not be discussed here. The analysis is based on the Bank Marketing dataset, which contains structured data collected from direct telemarketing campaigns run by a Portuguese bank. The objective is to predict whether a client will subscribe to a bank term deposit.

The dataset has 16 features and one target variable - "Subscribed deposit", which can take the values of yes or no. The dataset has 4521 observations, with just 521 of those taking the value of "yes" for the target variable. The remaining features are primarily categorical with varying numbers of levels (e.g. the "Job" column can take one of 12 values, while the "Contact type" column has two available values). There are three numeric features, which are age, last contact day and number of contacts. EDA shows that there is minimal collinearity among the feature columns, but does highlight some potentially concerning outliers. For example, average credit balance shows that there is a small contingent of people with very large average credit balances (Figure 1). Finally, the distributions of the feature variables against the target variable (Subscribed deposit) are plotted, which yields no striking findings (Figure 2). However, the plots of Age and Average Credit Balance could indicate that those who ultimately subscribe tend to be older on average and tend to have higher average credit balances.

For preprocessing, outliers are handled by capping values at 1.5 times the IQR, in either direction. The class imbalance is addressed by applying SMOTE (Synthetic Minority Over-sampling Technique), which takes training set from 3616 instances of nos and 50 instances of yeses to 3193 instances of both yeses and nos. There are no missing values or duplicated rows

in this particular dataset. The numeric features are standardized and the categorical features are encoded prior to model fitting.

Implementation details

Bagging Implementation

For the bagging implementation, a Random Forest model was used. On first pass, a simple random forest is created, setting the number of estimators at 100 trees. This resulted in a sensitivity of 0.17, a specificity of 0.98, an accuracy of 0.89 and a balanced accuracy of 0.58. To achieve an improved model, hyperparameter tuning using grid search and cross validation was used. Different numbers of estimators (50, 100, 200), max depths (0, 10, 20), minimum samples required for a split (2, 5, 10), minimum samples required per leaf (1, 2, 4) and the number of features to use for each split (square root of n samples or log base 2 of n samples) were all considered. Since this company would likely prefer a model that accurately identifies those who will subscribe, recall (i.e. sensitivity) was used as the scoring metric. The results of GridSearchCV concluded that a max depth of 20, using the square root of n samples for the max number of features, one sample per leaf, two samples per split and 200 estimators constituted an optimal set of hyperparameters. This tuned model resulted in a sensitivity of 0.11, a specificity of 0.99, an accuracy of 0.90 and a balanced accuracy of 0.55. The already high specificity increased by 1% point, but there is little change to the overall model performance.

Boosting Implementation

For the boosting implementation, a Gradient Boosting Classifier model was used, again with 100 estimators on first pass. This resulted in a sensitivity of 0.21, a specificity of 0.97, an accuracy of 0.89 and a balanced accuracy of 0.59. As before, hyperparameter tuning was carried out using GridSearchCV. Number of estimators (50, 100, 200), learning rate (0.01, 0.1, 0.2) and

max depth (3, 4, 5) were all evaluated. The optimal set of hyperparameters were: 100 estimators, 0.2 learning rate and 5 max depth. The tuned model resulted in 0.72 sensitivity, 0.39 specificity, 0.43 accuracy and 0.56 balanced accuracy. The sensitivity has dramatically improved in the tuned model while maintaining a similar balanced accuracy. The overall accuracy has predictably taken a hit, given the class imbalance still present in the test set.

Stacking Implementation

Finally, for the stacking implementation, the optimal random forest model, optimal gradient boosting model and a decision tree were used as base learners, with a final estimator using logistic regression to return probabilities of each class between 0 and 1. This model returned 0.20 sensitivity, 0.97 specificity, 0.88 accuracy and 0.59 balanced accuracy. This performance is almost exactly the same as the Gradient Boosting model on its own.

Results and comparison

A summary of all model evaluation metrics can be found in Figure 3. Among the ensemble models, the tuned Gradient Boosting performed best and the tuned Random Forest performed worst with respect to sensitivity. No model, including the tuned versions of Random Forest and Gradient Boosting, produced balanced accuracy scores greater than 0.60, all hovering between 0.55 and 0.59. Previously, a non-linear SVM was fit to the same data, which resulted in 0.33 sensitivity, 0.90 specificity, 0.84 accuracy and 0.61 balanced accuracy. The sensitivity of the non-linear SVM actually exceeds that of most of the ensemble models as well, with the added benefit that an SVM is easier to implement and interpret (depending on the scale at which such a model would be deployed). Additionally, the non-linear SVM produced a slightly higher balanced accuracy. However, the non-linear SVM's sensitivity is still more than half that of the tuned Gradient Boosting model's sensitivity.

References

3.4. Metrics and scoring: quantifying the quality of predictions. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/model_evaluation.html#scoring-parameter

GradientBoostingClassifier. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

RandomForestClassifier. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

StackingClassifier. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.StackingClassifier.html

Appendix

Figure 1: Pairplot

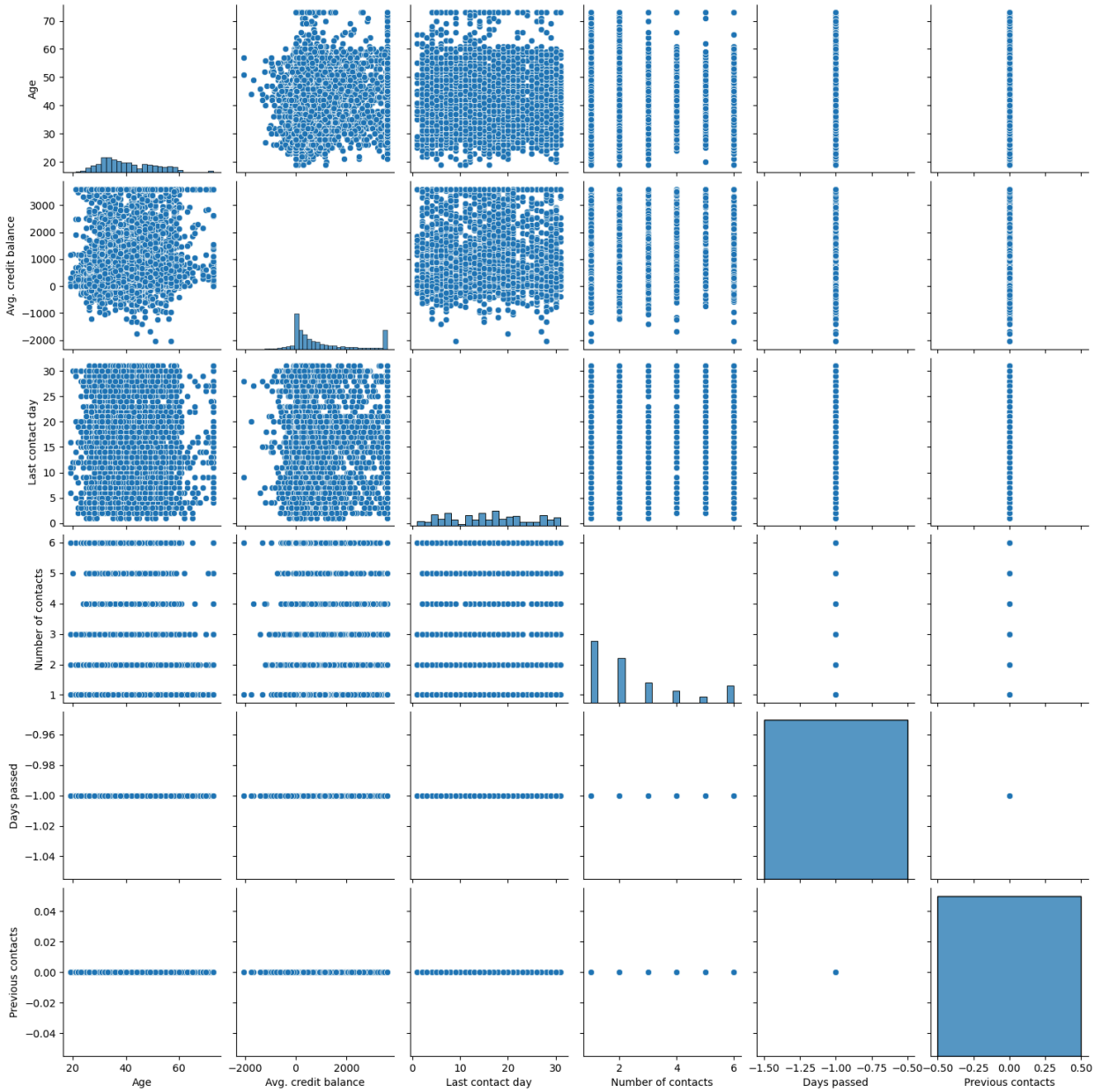

Figure 2: Charts of features against target (Subscribed deposit)

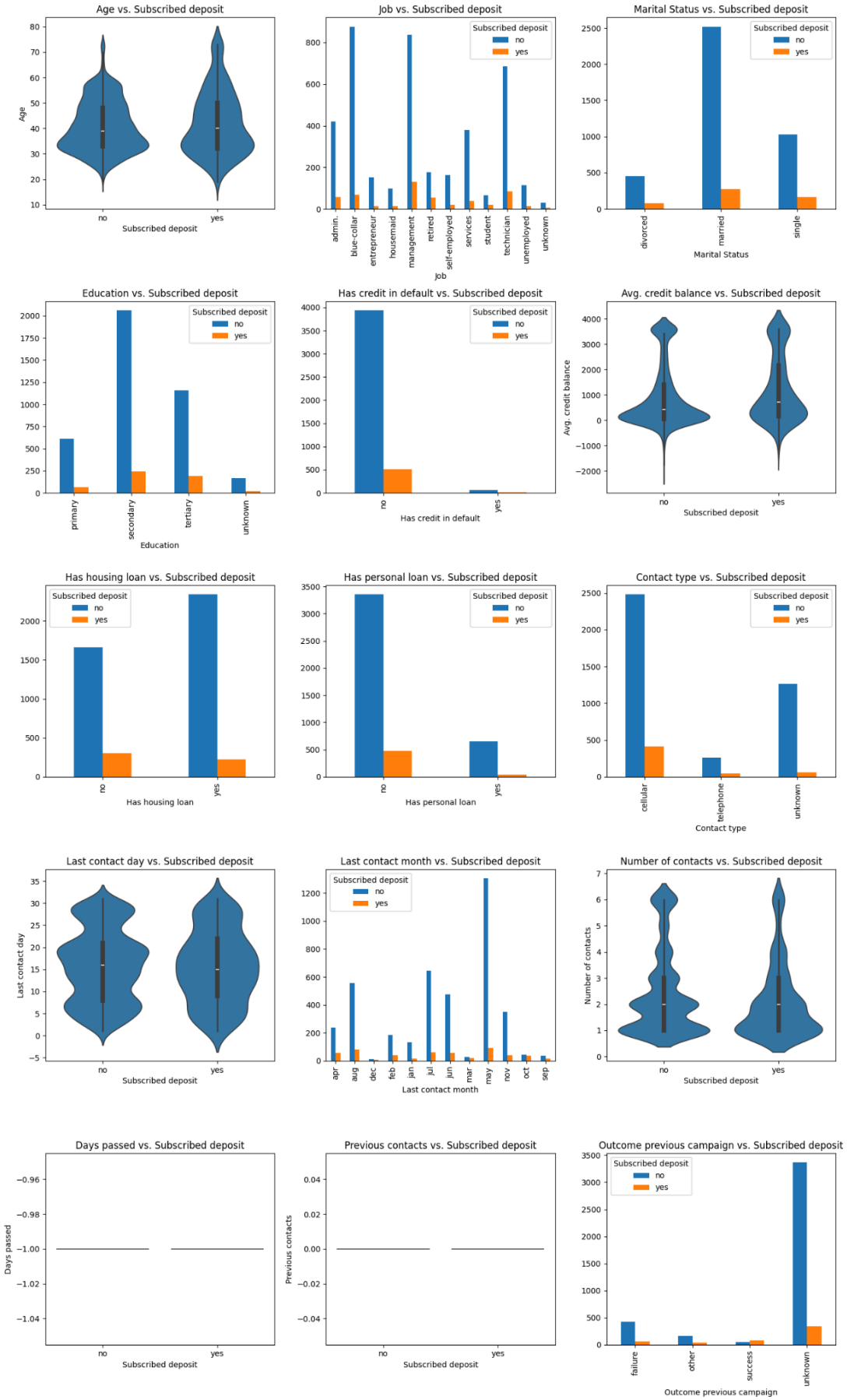

Figure 3: All Model Evaluation Results