Notes on competing ML models POC

Sanddep RajputML notes b

Hi Sophia,

The scope of the ML model changed a lot from the time you were involved—and the eventual models were built on the DNB data that you never received—but I wanted to share a common finding with you.

After project completion, I trained and validate two competitor models to logistic regression—Random Forest and XGBoost. In this experiment, we had ~40 predictors and used identical datasets for training and testing (70-30 split). All imputation was done in the same way—by using the median of valid values. The only factor that varied was the model “architecture”.

In these cases, we pick the model with best test set AUC. You can see that there’s not much difference between the three models that produce values between 0.662 and 0.683. XGB in particular had a colossal drop in AUC from train to test, meaning overfitting.

Logistic Regression ROC-AUC:  0.7128197 [Train],  **0.6832135 [Test]**

Random Forest       ROC-AUC:  0.7309099 [Train],  **0.6757280 [Test]**

XGBoost             ROC-AUC:  0.9993046 [Train],  **0.6620616 [Test]**

The model we delivered the client only had 8 predictors—all  of them statistically significant and “palatable”—meaning their coefficient had the correct sign and their internal “quality” team will permit the model to be used for decisioning. That took a lot of personal expertise and experimentation, but returned the AUC that is not any worse than the “auto” models using 40 predictors.

**CURATED**

Logistic Regression ROC-AUC:  0.6868110 [Train],  **0.6793906 [Test]**

**In a nutshell:**

When the number of predictors is small (under 100) and the understanding of data, domain & business problem is necessary (or otherwise excellent), newer black-box algorithms seldom outperform other ML algorithms. That point is further sharpened when

1. Many of the predictors are categorical (or binary) and not continuous, and
2. There are less than 10,000 examples of each class (as in classification)

Many data scientists misunderstand why Stochastic Boosting algorithms are popular. They assume them to be superior and more prestigious, but that’s totally misguided. The reason is much simpler: when the training data is enormous and already stored on a distributed system, stochastic gradient boosting is the *only* realistic possibility! Because that shifts the computation to the nodes where the data is—and further shortens each training cycle by not using the Hessian.

Regards,

Sandeep

Here’s the code with hyperparameters I picked based on common wisdom.

import statsmodels.api as sm

logreg2 = sm.Logit(yy\_trn, sm.add\_constant(XX\_trn)).fit()

from sklearn.ensemble import RandomForestClassifier

woods = RandomForestClassifier(max\_depth=3, n\_estimators=200, min\_samples\_split=5)

woods.fit(XX\_trn, yy\_trn)

import xgboost as xgb

xtr = xgb.XGBClassifier(objective='binary:logistic')

xtr.fit(XX\_trn, yy\_trn)