Sean Conroy July 29, 2019

State Farm Project: Discussion and Comparison of Models

All machine projects begin with pre-processing, and nearly identical pre-processing techniques were employed by both models. Null values were removed as follows:

- Null values for numeric columns were replaced with the average value of that column
- Null values for categorical columns were replaced with the mode value of that column
- Rows with null values for the categorical column “region” were removed entirely (7 rows)

In addition, some data cleanup was required, which is specified below:

- For the X35 and X68 columns, which is the day of the week or month of the year, misspelled values were replaced and normalized
- All categorical columns were re-coded as 1’s and 0;s using the “one-hot-encoding” technique

Other pre-processing:

- The balance of the target column “y” was checked, and it was found that 20% of the target was 1’s. This was decided to be insufficiently imbalanced as to require more advanced sampling models.
- The correlation of all columns was evaluated, and no highly correlated columns were found.
- For the Support Vector Machine model, the final pre-processing step was to scale all the columns except the target using Python’s standard scaler, which transforms each feature so that it’s mean is 0 and standard deviation is 1.
- A training- test split of 80% - 20% was used

As this is a binary classification problem, there are a number of excellent algorithms that can be attempted, including random forest, logistic regression, neural networks. A powerful gradient-boosting algorithm (XGBoost) was selected, along with an advanced linear model (support vector machines).

Model 1: XGBoost

Model 1 utilized the XGBoost algorithm. This tree-based algorithm is relatively straight-forward to deploy "out of the box", and has excellent performance even using default hyperparameters. This is a great first-pass algorithm for use with unfamiliar datasets, and due to its potential to be parallelized, its computational time is very low on a powerful machine. The tuning of hyperparameters is important, specifically the maximum tree depth, number of trees, min child weight, and learning rate. Optimum values were found using grid searches, and employed in the final model. The highest accuracy obtained with the XGboost model was 97.9%, and it was obtained using the following hyper-parameters:

- Max_depth: 7
- Min_child_weight: 1
- N_estimators: 500
- Learning_rate: 0.1

Model 2: Support Vector Machine

The support vector machine is a linear separation algorithm that also can give excellent performance for classification problems, even those non-linear in nature. The algorithm is not directly parallelizable, and thus can be slower. There are three main hyper-parameters, kernel type, C (the penalty parameter of the error term, which controls how smooth the decision boundary is), and gamma (the kernel coefficient, which defines "how far the influence of a single training example reaches"[1]). For non-linear problems, the Radial Basis Function (RBF) kernel is recommended. After tuning, an accuracy of 98.35% on the training test set was achieved, using the following hyper-parameters:

- Kernel: RBF
- C: 1.0
- Gamma: 'auto'

Other comments:

Given that I do not have any intuition about the dataset itself, feature generation or feature engineering beyond simple scaling and pre-processing was not possible. If given more time and resources, I would attempt to try a neural network as well as a variety of simple mathematical combinations of the features to enhance accuracy. Both XGBoost and Support Vector Machines have been deployed frequently in production environments successfully, but XGBoost is more adaptable to additional data fed to the model, and thus would be the preferred choice.

[1] https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html