MetaLogic consulting

Use Case: Warranty Fraud Detection

White Papers

OBJECTIVE

The goal of this project was to build machine learning models for an electronics manufacturer that use customer profile data, geographical data, product data, and warranty claims data to predict whether a warranty claim is genuine or fraudulent. As experts expect around 5-15% of warranty claims to be fraudulent, a successful warranty fraud detection algorithm could help companies save millions of dollars annually.

DATA

To carry out our objective, we extracted a dataset with 11,917 observations and 20 seemingly significant features. Each observation represented a warranty claim that was either genuine or fraudulent. As expected, the target variable (label) was imbalanced, since around 8% of the observations were fraudulent and approximately 92% of the observations were genuine.

PREPROCESSING

Preprocessing steps included handling missing values, encoding categorical features, dropping unnecessary columns, standardizing redundant values, feature engineering, and encoding the outcome variable (label). The reason that we encoded the categorical string features into dummy numerical features was that machine learning models tend to perform better when text classes are translated into binary inputs.

Furthermore, before moving on to the modeling section of our project, we performed a holistic exploratory data analysis to understand the characteristics and nature of the data.

MODEL BUILDING AND EVALUATION #1: BENCHMARK

Although the dataset was imbalanced, as it had a clear majority of warranty observations that were genuine, we first created a set of benchmark models in which we did not equalize the label classes by oversampling or undersampling the data. In addition, for these benchmark models, we did not perform feature selection techniques and, therefore, used all 20 features to predict the legitimacy of a warranty claim.

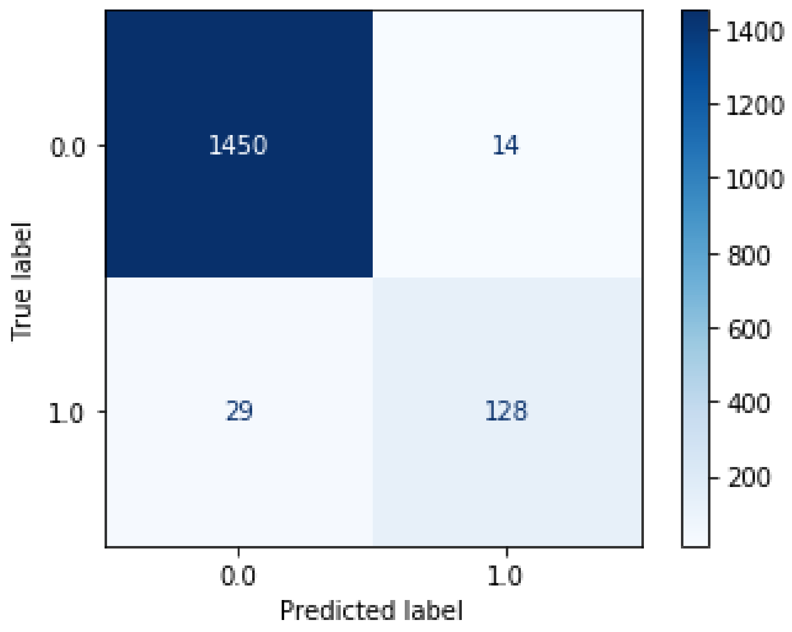
Next, we randomly split the data frame into a training set (80%) and a testing set (20%). Subsequently, we built the following machine learning algorithms on the training set: (1) Logistic Regression, (2) Support Vector Machines, (3) Multi-Layer Perceptron (Artificial Neural Networks), (4) Random Forests, and (5) Gradient Boosting.

For each of the five algorithms, we tested hundreds of model iterations by tuning the model hyperparameters in each iteration. To select the best model within each algorithm, we performed cross-validation with k=5 and selected the model with the highest cross-validation classification accuracy. Therefore, at the end of the first model building stage, we chose a total of five models from a set of over thousand models. More specifically, we chose the best Logistic Regression model, the best Support Vector Machines model, the best Multi-Layer Perceptron model, the best Random Forests model, and the best Gradient Boosting model.

Next, we used each of these models to predict the outcome of the observations in the testing set. Note that this step emulates predicting real data, as these models have never seen these observations before. After predicting each observation in the testing set, we compared the predictive accuracies of the five models. The metrics that we used were overall classification accuracy, precision, and recall.

The best overall model was the Gradient Boosting model with a learning rate of 0.1, max depth of 5, and number of estimators of 250 (amongst 20+ other hyperparameters). The results were as follows:

- Accuracy: 97.3%
- Precision: 90.1%
- Recall: 81.5%



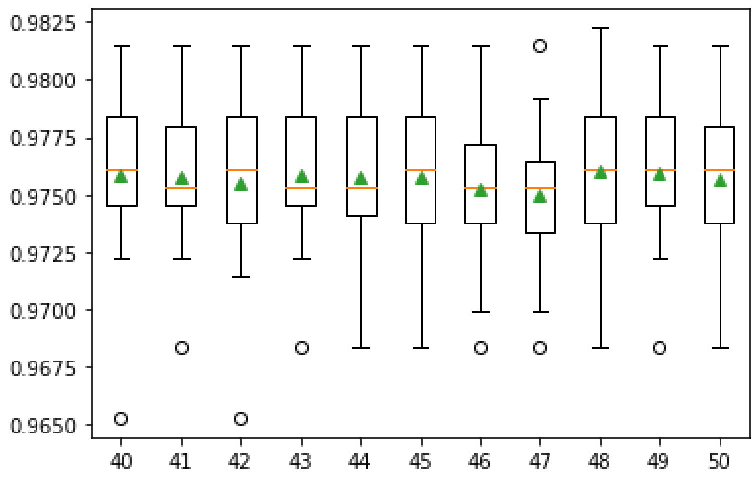
MODEL BUILDING AND EVALUATION #2: FEATURE SELECTION

In the second model building phase, we added an extra layer of complexity relative to the benchmark models. To be specific, we performed feature selection to isolate the most significant features in terms of predictive power. In this phase, we did not address the imbalanced nature of the data frame.

To conduct feature selection, we used a wide variety of techniques on the training data. It is crucial to use feature selection techniques only on the training data and not the entire dataset to avoid overfitting the models to the unseen testing data. First, we looked at the variable importance from the Random Forest model previously trained. Second, we performed the following feature selection techniques: F-Classif, Chi-square, and F-Regression. All feature selection techniques were consistent, as they selected very similar top features. Third, we used the ExtraTreesClassifier to perform feature selection. Fourth, we used the Recursive Feature Elimination (RFE) technique. As we found the cross-validation accuracy to be higher (with Random Forests) using the features given by the RFE technique, we proceeded with the RFE method.

The next step was to evaluate the optimal number of features to select using RFE. Accordingly, we iteratively added features based on their predictive power, starting from the top 1 feature until all the features were added. For each iteration, we logged the cross-validation classification accuracy. It is important to note that we had a total of 94 features, as opposed to the initial 20 features, as we performed feature engineering and created dummy variables for categorical features.

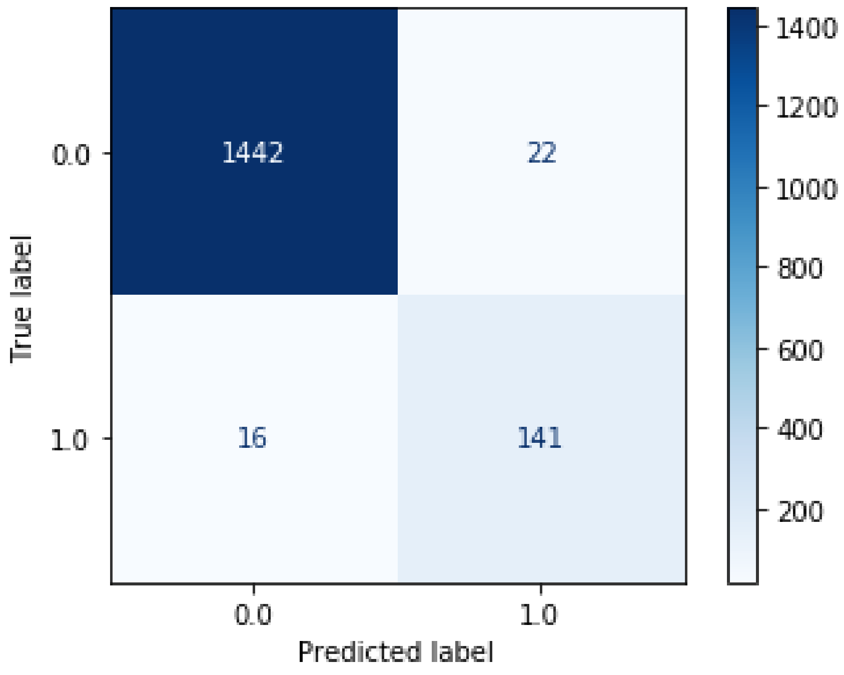
The figure below (one of 10 figures that we used to select the optinal number of features) shows that using 48 variables yielded the highest cross-validation accuracy.



Next, we performed the same steps as the previous phase: we built hundreds of iterations of each of the five machine learning algorithms highlighted above (to tune hyperparameters), selected the best model in terms of cross-validation accuracy within each algorithm class, and evaluated the top five models on the testing set.

The best overall model was the Gradient Boosting model with a learning rate of 1, max depth of 7, and number of estimators of 50 (amongst 20+ other hyperparameters). The results were as follows:

- Accuracy: 97.7%
- Precision: 86.5%
- Recall: 89.8%



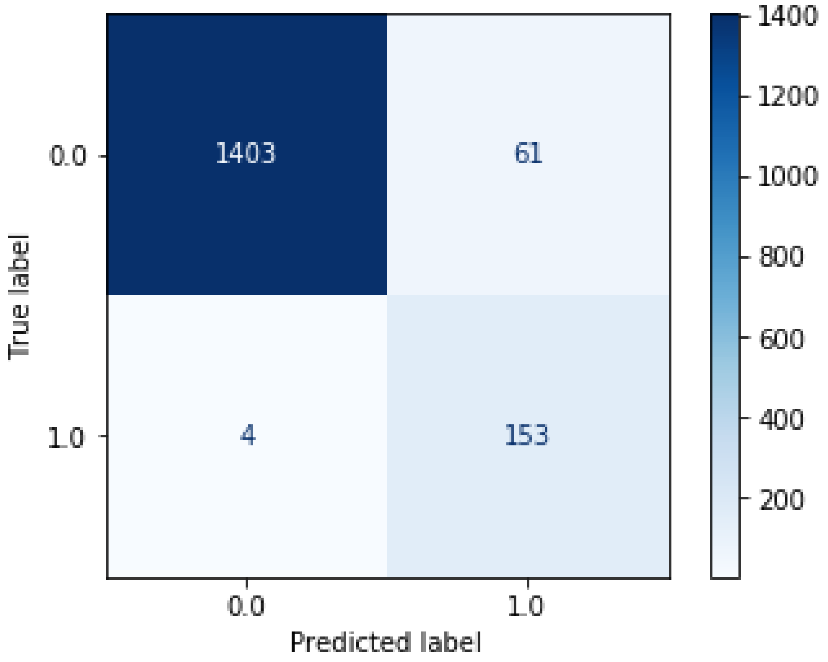
MODEL BUILDING AND EVALUATION #3: SMOTE

In the third model building phase, we addressed the imbalance problem of our dataset by using an oversampling technique called Synthetic Minority Over-Sampling Technique (SMOTE). This technique was essentially used to equalize the two label classes to 50% each. However, in this phase, we used all the variables and not the top 48 variables given by feature selection. It is essential to note that the oversampling was only done for the training set observations and not the testing set records, because we want our models to predict true and not synthetic observations.

As such, we built hundreds of iterations of each of the five machine learning algorithms (while tuning hyperparameters), selected the best model in terms of cross-validation accuracy within each algorithm class, and evaluated the top five models on the testing set.

The best overall model was the Gradient Boosting model with a learning rate of 0.1, max depth of 5, and number of estimators of 250 (amongst 20+ other hyperparameters). The results were as follows:

- Accuracy: 96%
- Precision: 71.5%
- Recall: 97.5%



MODEL BUILDING AND EVALUATION #4: AZURE ML

In the fourth model building phase, we leveraged Microsoft Azure's Machine Learning Studio to recreate the previous machine learning models built. We experimented with various combinations of pipelines: (1) No feature selection and no SMOTE, (2) Feature Selection and no SMOTE, (3) SMOTE and no feature selection, (4) Feature Selection and SMOTE, and (5) Custom Azure ML model. The custom script consisted of an untrained Gradient Boosting model with the optimal hyperparameters, no feature selection, and no oversampling technique. For each of these pipelines, we built hundreds of models based on the five machine learning algorithms outlined above with the objective of tuning hyperparameters to maximize predictive accuracy.

Next, we used each model to predict the observations in the testing set and evaluated their performance.

MODEL SELECTION

The best model in the second phase yielded the highest overall accuracy of 97.7%, whereas the best model in the third phase yielded the highest recall of 97.5%. Since the ultimate objective of the project was to identify as many fraudulent claims as possible, the model with the higher recall was more suited to our goal. However, the downside to picking this model is a lower precision, meaning that it flagged a higher number of genuine claims as fraudulent relative to the other model. In essence, this model correctly detected 97.5% of all the warranty claims that were actually fraudulent.

MODEL DEPLOYMENT

Using the model built in the second phase, we deployed a web service inference pipeline that allowed the end-user to make real-time predictions and provided the option to make batch predictions. Furthermore, we also created applications using Flask and Swagger API, thereby allowing the end-user to make single-instance predictions and/or batch predictions using the best machine learning model.

Finally, we containerized the final model, application, dependencies, libraries, etc. into a Dockerfile, which allowed the end-user to deploy the entire package without technical issues.