State Farm Take Home Assessment

**1. Dataset:**

1.1 Introduction

The data set given is comprised of 100 features for both the training and test set. The features match in name, and datatype and are assumed to be the same. Various features are missing up to 80% of their data. Individual data points that are missing many features are assumed to be insignificant for this project.

1.2 Data Cleaning

Looking at the target variable ‘y’, we can quickly conclude that about 15% of the data are positive classes and 85% are negative classes. Rebalancing this data set is done late in the project but demonstrates some key strengths and weaknesses between the two chosen modeling methods. This will be discussed in the modeling section.

As a basic requirement for evaluating any model, sparse features must be dealt with. In my quick-and-dirty approach, I created a transformation pipeline that imputes with the mean of the feature. Then the object (string) features were imputed with the most frequently type of feature. Lastly the object features were one-hot encoded.

On top of these cleaning approaches, some sparse features (>50% missing data) were dropped. This was adjusted as the modeling was done.

1.3 Summary

There are a few details that weren’t cared for in the data cleaning procedure that I would like to address here. The most obvious is the simplicity of one-hot encoding most string features. I did find that there were some “yes” and “no” columns that could easily be represented by a single binary feature, instead of two inversely correlated binary features. This is not an issue since the model will correctly balance these feature weights in a logistic regression. And (I assume but am not completely certain) that the second model I use (XGBoost) is sophisticated enough to handle the extra bloat.

The process of creating models is an iterative process, and the flow of this report (and accompanying notebook) does not reflect that fact. I did go back and forth a few times to check if leaving in more (or less) sparse features would affect the performance of the model, but I think a deeper dive into the individual features would help me understand where to go next.

Also worth noting is that I completely left out any feature engineering on the numerical columns.

**2. Modeling**

2.1 Introduction

For both models, standard procedure for testing models applies. The “training” data provided is split between a training, and validation set (0.75, 0.25 respectively). For any hyperparameter tuning, k-fold cross-validation is implemented to prevent overfitting to the hyperparameters. Lastly, for a good set of parameters, the model is then trained on the full data set provided before predicting on the final test set.

2.2 Logistic Regression

A benefit of using the logistic regression is that it is easy to implement and understand. However, it assumes a linearity between the dependent and independent variables (per its generalized linear model class). Meaning we would be required to spend more time feature engineering it than using an alternative, non-glm model. Additionally, leaving the dataset unbalanced can greatly affect the model’s performance.

2.2 XGBoost Classification

XGBoost classifiers are great classifiers because they are sophisticated enough to capture nonlinear relationships between the independent and dependent variable. They are unfortunately poor extrapolators. This means if the model receive inputs that are outside the range of the parameters it was trained for, it can quickly become unreliable.

**3. Summary**

A comparison of the final models can be summarized below using the AUC metric.

|  |  |
| --- | --- |
| Logistic Regression | 0.63 |
| XGBoost | 0.58 |

Based on the AUC metric, I’m inclined to believe the logistic regression model will perform best, although I did not believe so to begin with.

Lastly, given a non-technical business partner, I would attempt to translate the difference in the model performance to some dollar value. Given that I know nothing about what data this is or how I can tie it to auxiliary data, I would use a confusion matrix to simply show how well the each model correctly classifies our data.