A BINARY CLASSIFIER FOR DEFECT PREDICTION: EXAMPLE USE CASE WITH WINE ID

Kevin Geidel

MSDS 422: Practical Machine Learning

Northwestern University

June 2, 2024

1 Executive summary

Given the rising prevalence of data science teams in industry and the widespread adoption of the *Team Data Science Process (TDSP)* methodology, classifier *Machine Learning* (ML) models will see deployment in more and more use cases. Even small teams will be deploying data science tools that are tailored for many different settings throughout various enterprises (Hyatt, 2024).

The concept is explored with a use case that seeks to deploy a binary classifier for a defect prediction application. Using the `wine` data set (Vanschoren, 2014) from `openml.org` we have 14 features for 178 records that include labels on if the sample is wine or not. *Exploratory Data Analysis (EDA)* is conducted with the aim of selecting, training and tuning four models to classify, for now, a sample as wine or not. This would be extended to a model that seeks to predict samples that run a high probability of being defective or non-conforming. Emphasis is on turning this wine classifier deployment into a DRY (*Don't Repeat Yourself*), generic (but still flexible) and easy to deploy tool that data scientists can use to expedite routine data exploration, data wrangling and data mining.

All four models took very little ($<$0.3 seconds) to both fit and predict. This is possibly due, in part, to the small data set. The *RandomForestClassifier* performed the best of the four. The accuracy, precision, recall and f1 scores were 0.96, 0.9, 0.9 and 0.9 respectively.

2 Research design

There are a variety of options for ML binary classifiers (Geron, 2023, ch 3). Is there a particular algorithm that should be considered when the target is conformance vs non-conformance from a quality assurance perspective? What steps should be taken to tune a binary classifier for maximal accuracy, precision and recall? The following is an ML project meant to test some best practices for deploying these classifiers.

The study will create a work flow that mirrors the TDSP ("What is the Team Data Science Process?", 2024). Various stages in the TDSP (which is illustrated in figure 1) are represented by data models and have methods for applying standard work to these structures, moving them along the path to the next stage. The benefits of using customized libraries to conduct ML projects are shortened development cycles, more robust/stable applications, greater interoperability amongst team members and more time to focus on making models that perform better and provide more actionable conclusions (Geron, 2023). The code for the

project (and this paper) can be found in the `GitHub` repository located at `https://github.com/kgeidel/MSDS-422-final-project` (Geidel, 2024).

Performance of four different classifier models and various techniques will be measured and compared. Various transformations and parameters are used. All four trials will be evaluated using the same metrics. Several quantitative performance metrics are industry standards for evaluating ML effectiveness (Powers, 2007). Table 1 lists the metrics used.

After cursory *Exploratory Data Analysis* (EDA) and the creation of three engineered features, a train/test split of 15% is used to prevent target and data leakage. Our target label is separated from features and a preprocessing pipeline is developed. All four models are fit to the cleaned training data. Cross-validation is used to find average performance metrics against other folds of the training data and then the test data itself.

3 Exploratory Data Analysis (EDA)

Listing 1 at the end of the appendix includes the twenty-nine cells of Jupyter notebook code and their output. EDA begins in cell 2 by loading the data set which was downloaded in the `arff` format. There are 178 records with 13 features plus a label, the binary class that I encode to *isWine*. Cell 3 lists the features and their data types. We see *binaryClass* is the only categorical label and all records contain values for all features. In cells 4 and 5 we encode the *binaryClass* into our numeric target *isWine*.

The data set structure is further probed in cell 6, which describes each feature. We see a glimpse at the distribution of each feature. In cell 7 we confirm no missing values which removes the need for an imputer. Visualization begins in cell 8, we create histograms of each feature. Some, particularly *Malic Acid*, *Ash*, *color intensity* and *Proline* seem to have normal distributions (although skewed.) Others appear to have multi-modal distributions (such as *alcohol* and the *OD280/OD315 dilution*.) Numeric descriptions continue in cell 9, finding Pearson correlation coefficients for each variable in relation to our target. The strongest correlation is a negative correlation between *alcohol* and the target at -0.7264. *Color intensity* and *proline* also have strong correlations.

Scatter plots are created in cell 10, showing how each variable moves with the others. A few interesting combinations are chosen for further analysis. *Color intensity* over *Flavanoids* had two very distinct groupings in the scatter plots. *Alcohol* over *Color intensity* has a tightly bound linear section with a positive correlation and then another cluster the appears to be a cluster with zero slope. *Alcohol* over *Flavanoids* has

a hooked shape in the scatter plot with two ostensibly distinct clusters. Cell 11 generates these plots much larger for examination.

Cell 12 compares the distribution of these features but with targets (*isWine* true/false) separated. *Alcohol* is a promising feature, the interquartile ranges of the target true and target false distributions do not over lap at all. The other distributions are interesting but as these features stand on their own do not appear to have much predictive power.

Two features are engineered in cell 13. *color_per_flavanoid* is defined as the ratio of *color intensity* over *flavanoids*. We also take the natural log of the *flavanoid* attribute. The new features are put into histograms (separated by target.) The ratio *color_per_flavanoid* jumps out as another promising feature as the target true and target false distributions seem distinct.

4 Data pipeline and ML preprocessing

Development of the data pipeline begins in cell 14. The data set is divided into training and test (15%.) Labels are striped from the training features and kept to the side for fitting. Cell 15 is meant to drop records with missing values that could not be imputed. We confirmed above that this does not apply to any records in the data set but a more abstract pipeline would do well to take these steps. In a similar vein, two separate pipelines are created for numeric and categorical features, even though we only require the former in this particular instance. Cell 16 creates the numeric pipeline: a simple imputer utilizing the median strategy and a standard scaler. Cell 17 creates the categorical pipeline: an ordinal encoder, a simple imputer (using "most frequent") and one hot encoding that ignores unknown label values.

In cell 18 these are combined into the preprocessing pipeline. We process the training data in cell 19 and confirm it is ready for model fitting.

5 ML engineering, evaluation and deployment

Four models commonly used for classification are testing and compared. They are the *Stochastic Gradient Descent* (SGD) optimization, the *Random Forest Classifier*, the *Support Vector Machine* (SVM) and the *K-Neighbors Classifier*. Cells 20, 21, 22 and 23 create and fit these four models respectively.

Final evaluation is against the test split of the data set. The target is removed from the test data the features are transformed via the preprocessing pipeline in cell 24. Deployment, in this bench test comparison, entails predicting on the test records and calculating performance metrics (cells 25-28.) Results are outlined in section 6.

6 Findings

The four models tested performed very similarly on the `wine` data set. With only 178 total records (the splits were, obviously, smaller still) time was not a significant factor. All models fit and predicted very quickly. The *RandomForestClassifier* took the "longest" to both fit (0.1 seconds) and predict (0.3 seconds.) Performance against the test data was also similarly impressive across all models.

Cell 29 contains a dataframe that outlines the results. The *RandomForestClassifier* had the best overall performance. The precision, recall and f1 scores were higher for this model. The results were encouraging but may be indicating possible problems with over-fitting or testing bias. This is an issue that additional records and another look at employed techniques would help alleviate.

7 Conclusions

Additional evaluation among folds of the training data would further inform my comparison of the models. This would also hep rule out leakage and bias. Additionally, larger data sets could prove naturally more interesting and require more advanced strategies.

This is a practical choice for deploying classifiers in industry. In fact, this project topic was inspired by a ML tool I developed at my job. We had not performed evaluation on multiple models before selecting a *RandomForestClassifier*. I am relieved to see this model perform well here in this experiment. I got lucky that time, but this type of analysis will be an integral part of my ML deployments in the future.

Appendix

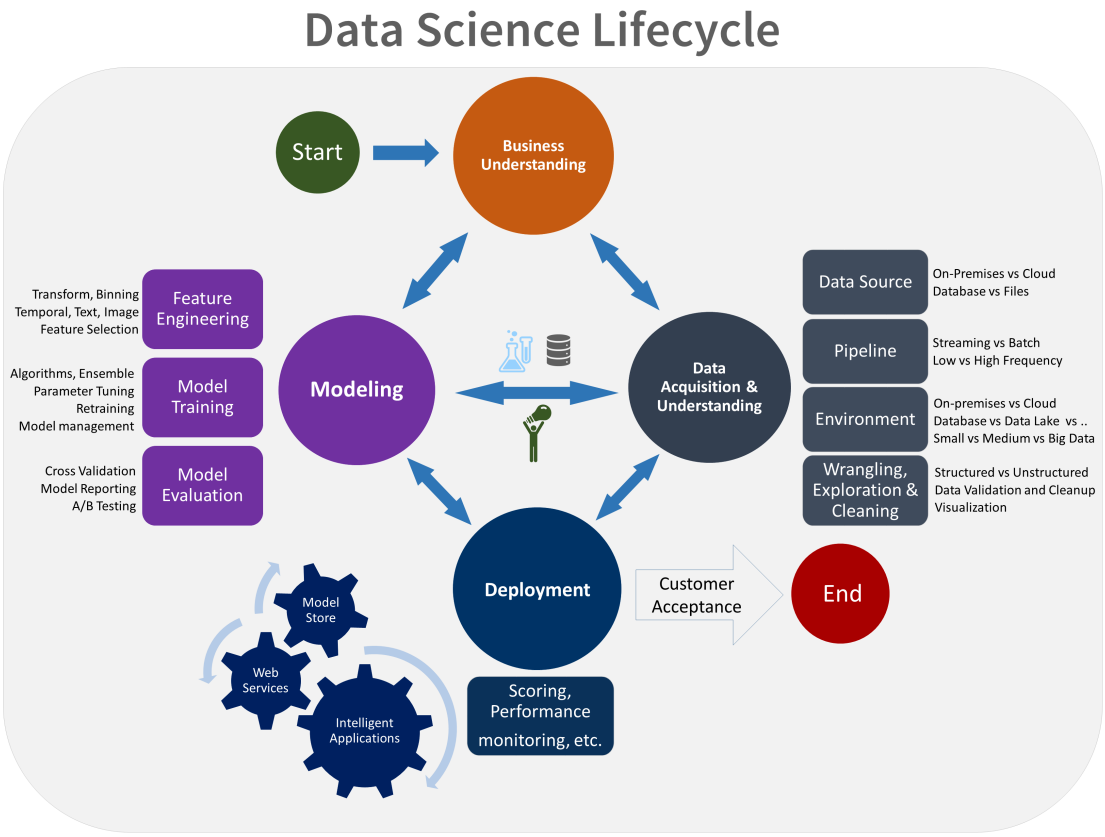

Figure 1: "Data life cycle" ("What is the Team Data Science Process?", 2024)

Metric	Calculations*	Description
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$	Fraction of predictions made correctly
Precision	$\frac{TP}{TP+FP}$	Fraction of positive predictions made correctly
Recall	$\frac{TP}{TP+FN}$	Fraction of actual positives correctly identified

*TP=True Positives, TN=True Negatives, FP=False Positives, FN=False Negatives

Table 1: Performance metrics used on each of the four trials.

References

Geidel, K. (2024). MSDS-422-final-project. Northwestern University. https://github.com/kgeidel/MSDS-422-final-project.

Geron, A. (2023). *Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow* (3 ed.). O'Reilly.

Hyatt, J. (2024). Django + postgres: A data science juggernaut. Master's thesis, Syracuse University.

Powers, D. (2007). Evaluation: From precision, recall and f-factor to roc, informedness, markedness and correlation. *School of Informatics and Engineering*.

Vanschoren, J. (2014). wine. https://www.openml.org/search?type=data&sort=runs&status=active&id=973.

"What is the Team Data Science Process?" (2024). Azure Architecture Center. https://learn.microsoft.com/en-us/azure/architecture/data-science-process/overview.