Introduction

Our goal was building a machine learning model to predict if a forest fire in Portugal had or not intentional cause.

Problem Definition

This is a binary classification problem. The target variable is, of course, intentional\_cause and we want to output the probability of a fire being intentional. We do this by capturing the knowledge from the dataset given to us, that has the following original variables.

Predictive Modelling

The evaluation metric we used was area under the roc curve, which is the trade-off between true positive rate and false positive rate; basically tells us how good the model is at distinguishing the two classes, and the higher the better.

We split the data in train and test and the train in train plus validation.

We used the k-fold cross validation. With k being 10. This way, we have the guarantee that each example will be used at least once for training and other for testing.

Then we created recipes:

fires\_rec <- recipe(intentional\_cause ~ ., data=fires\_train) %>%

step\_rm(id) %>%

step\_rm(municipality) %>%

step\_rm(parish) %>%

step\_rm(lat) %>%

step\_rm(lon) %>%

step\_rm(alert\_hour) %>%

step\_date(alert\_date) %>%

step\_date(extinction\_date) %>%

step\_date(firstInterv\_date) %>%

step\_naomit(everything(), skip = TRUE) %>%

step\_novel(all\_nominal(), -all\_outcomes()) %>%

step\_normalize(all\_numeric(), -all\_outcomes()) %>%

step\_dummy(all\_nominal(), -all\_outcomes()) %>%

step\_dummy(all\_nominal(), -all\_outcomes()) %>%

step\_zv(all\_numeric(), -all\_outcomes()) %>%

step\_corr(all\_numeric\_predictors(), threshold = 0.7, method = "spearman") %>%

step\_bin2factor(all\_outcomes()) %>%

prep()

summary(fires\_rec)

We performed some tunning on the hyperparameters.

Hyperparameter Tuning  
• A hyperparameter is a parameter whose value controls the  
learning process  
• Conduct a search to tune the hyperparameters  
• An experimental methodology is necessary to avoid overfitting

Grid Search  
• define grid  
• choose the best

try

bagging – ensemble of decision trees

SVM taking too much time and seemed to output errors.

Overfitting not occured

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| approach | model | function | engine | parameters | notes |
| Distance | kNN | nearest\_neighbor() defines a model that uses the K most similar data points from the training set to predict new samples. This function can fit classification and regression models. | kknn | * neighbors: # Nearest Neighbors (type: integer, default: 5L) * weight\_func: Distance Weighting Function (type: character, default: ‘optimal’) * dist\_power: Minkowski Distance Order (type: double, default: 2.0) | With dates |
| Probabilistic | Logistic regression | logistic\_reg() defines a generalized linear model for binary outcomes. A linear combination of the predictors is used to model the log odds of an event. This function can fit classification models. | glmnet | * penalty: Amount of Regularization (type: double, default: see below) * mixture: Proportion of Lasso Penalty (type: double, default: 1.0)  1. lasso; 0 – ridge | No dates rec |
| probabilistic | Naive bayes | naive\_Bayes() defines a model that uses Bayes' theorem to compute the probability of each class, given the predictor values. This function can fit classification models. | klaR | * smoothness: Kernel Smoothness (type: double, default: 1.0) * Laplace: Laplace Correction (type: double, default: 0.0) | No dates |
| optimization | Single Layer Neural network | mlp() defines a multilayer perceptron model (a.k.a. a single layer, feed-forward neural network). This function can fit classification and regression models. | nnet | * hidden\_units: # Hidden Units (type: integer, default: none) * penalty: Amount of Regularization (type: double, default: 0.0) * epochs: # Epochs (type: integer, default: 100L) | No dates |
| Ensembles | Random forest | rand\_forest() defines a model that creates a large number of decision trees, each independent of the others. The final prediction uses all predictions from the individual trees and combines them. This function can fit classification, regression, and censored regression models. | ranger | * mtry: # Randomly Selected Predictors (type: integer, default: see below) * trees: # Trees (type: integer, default: 500L) * min\_n: Minimal Node Size (type: integer, default: see below). For classification, a value of 10 is used. | With dates; parallell |
| ensembles | Boosted trees | boost\_tree() defines a model that creates a series of decision trees forming an ensemble. Each tree depends on the results of previous trees. All trees in the ensemble are combined to produce a final prediction. This function can fit classification, regression, and censored regression models. | xgboost | * tree\_depth: Tree Depth (type: integer, default: 6L) * trees: # Trees (type: integer, default: 15L) * learn\_rate: Learning Rate (type: double, default: 0.3) * mtry: # Randomly Selected Predictors (type: integer, default: see below) * min\_n: Minimal Node Size (type: integer, default: 1L) * loss\_reduction: Minimum Loss Reduction (type: double, default: 0.0) * sample\_size: Proportion Observations Sampled (type: double, default: 1.0) * stop\_iter: # Iterations Before Stopping (type: integer, default: Inf) | Parallel required;  With dates |
| Tree-based | Decision tree via cart | decision\_tree() defines a model as a set of if/then statements that creates a tree-based structure. This function can fit classification, regression, and censored regression models. | rpart | * tree\_depth: Tree Depth (type: integer, default: 30L) * min\_n: Minimal Node Size (type: integer, default: 2L) * cost\_complexity: Cost-Complexity Parameter (type: double, default: 0.01) | With dates |