

Report format

Table of contents

1	Introduction	1
2	Exploratory data analysis	1
3	24 logistical models (Individual models then ensembles, in alphabetical order)	2
4	Ensembles and individual model plots	6
5	Strongest evidence based data:	8
6	Five strongest evidence based recommendations	8
7	Conclusions	8

1 Introduction

Statement of the problem from the customer's perspective

History of the problem, previous results

2 Exploratory data analysis

- Head of the data
 - Discuss the characteristics of each feature.
- y (predictor) vs target variables (insert plot here)
 - Discussion of y vs target variables

- Boxplots of the numeric data (insert plot here)
 - Discussion of boxplots of the numeric data
- Histograms of each numeric column (insert plot here)
 - Discussion of histograms of each numeric column
- Data summary (insert table here)
 - Discussion of the data summary
- Outliers in the data (insert outliers data here)
 - Discussion of outliers in the data
- The stories in the exploratory data analysis

3 24 logistical models (Individual models then ensembles, in alphabetical order)

One paragraph summary about statistical modeling

- Bagging


```
bagging_train_fit <- ipred::bagging(formula = y ~ ., data = train)
```
- BayesGLM


```
bayesglm_train_fit <- arm::bayesglm(y ~ ., data = train, family = gaussian(link = "identity"))
```
- BayesRNN


```
bayesrnn_train_fit <- brnn::brnn(x = as.matrix(train), y = train$y)
```
- BoostRF (Random Forest) (tuned)


```
boost_rf_train_fit <- e1071::tune.randomForest(x = train, y = train$y, mtry = ncol(train) - 1)
```
- Cubist


```
cubist_train_fit <- Cubist::cubist(x = train[, 1:ncol(train) - 1], y = train$y)
```
- Earth


```
earth_train_fit <- earth::earth(x = train[, 1:ncol(train) - 1], y = train$y)
```

- Elastic (uses best model)
`best_elastic_model <- glmnet::glmnet(x, y, alpha = 0, lambda = best_elastic_lambda)`
- GAM (Generalized Additive Models) (uses smoothing splines)
`f2 <- stats::as.formula(paste0("y ~", paste0("gam::s(", names_df, ")", collapse = "+")))`
`gam_train_fit <- gam(f2, data = train1)`
- Gradient Boosted
`gb_train_fit <- gbm::gbm(train$y ~ ., data = train, distribution = "gaussian", n.trees = 100, shrinkage = 0.1, interaction.depth = 10)`
- K-Nearest Neighbors (Optimized)
`knn_train_fit <- e1071::tune.gknn(x = train[, 1:ncol(train) - 1], y = train$y, scale = TRUE, k = c(1:25))`
- Lasso (uses best model)
`best_lasso_lambda <- lasso_cv$lambda.min`
`best_lasso_model <- glmnet(x, y, alpha = 1, lambda = best_lasso_lambda)`
- Linear (tuned)
`linear_train_fit <- e1071::tune.rpart(formula = y ~ ., data = train)`
- Neuralnet
`neuralnet_train_fit <- nnet::nnet(train$y ~ ., data = train, size = 0, linout = TRUE, skip = TRUE)`
- PCR (Principal Components Regression)
`pcr_train_fit <- pls::pcr(train$y ~ ., data = train)`
- PLS (Partial Least Squares)
`pls_train_fit <- pls::plsr(train$y ~ ., data = train)`
- Ridge
`best_ridge_lambda <- ridge_cv$lambda.min`
`best_ridge_model <- glmnet(x, y, alpha = 0, lambda = best_ridge_lambda)`
- RPart
`rpart_train_fit <- rpart::rpart(train$y ~ ., data = train)`
- SVM (Support Vector Machines) (tuned)
`svm_train_fit <- e1071::tune.svm(x = train, y = train$y, data = train)`

- Tree

```
tree_train_fit <- tree::tree(train$y ~ ., data = train)
```

- XGBoost (optimized)

```
xgb_model <- xgboost::xgb.train(data = xgb_train, max.depth = 3, watchlist = watchlist_test, nrounds = 70)
xgb_model_validation <- xgboost::xgb.train(data = xgb_train, max.depth = 3, watchlist = watchlist_validation, nrounds = 70)
```

- Ensemble Bagged Random Forest (tuned)

```
ensemble_bag_rf_train_fit <- e1071::tune.randomForest(x = ensemble_train, y = ensemble_train$y_ensemble, mtry = ncol(ensemble_train) - 1)
```

- Ensemble Bagging

```
ensemble_bagging_train_fit <- ipred::bagging(formula = y_ensemble ~ ., data = ensemble_train)
```

- Ensemble BayesGLM

```
ensemble_bayesglm_train_fit <- arm::bayesglm(y_ensemble ~ ., data = ensemble_train, family = gaussian(link = "identity"))
```

- Ensemble BayesRNN

```
ensemble_bayesrnn_train_fit <- brnn::brnn(x = as.matrix(ensemble_train), y = ensemble_train$y_ensemble)
```

- Ensemble Boosted Random Forest (tuned)

```
ensemble_boost_rf_train_fit <- e1071::tune.randomForest(x = ensemble_train, y = ensemble_train$y_ensemble, mtry = ncol(ensemble_train) - 1)
```

- Ensemble Cubist

```
ensemble_cubist_train_fit <- Cubist::cubist(x = ensemble_train[, 1:ncol(ensemble_train) - 1], y = ensemble_train$y_ensemble)
```

- Ensemble Earth

```
ensemble_earth_train_fit <- earth::earth(x = ensemble_train[, 1:ncol(ensemble_train) - 1], y = ensemble_train$y_ensemble)
```

- Ensemble Elastic (uses best model)

```
ensemble_best_elastic_lambda <- ensemble_elastic_cv$lambda.min
```

```
ensemble_best_elastic_model <- glmnet(ensemble_x, ensemble_y, alpha = 0, lambda = ensemble_best_elastic_lambda)
```

- Ensemble Gradient Boosted

```
ensemble_gb_train_fit <- gbm::gbm(ensemble_train$y_ensemble ~ ., data = ensemble_train, distribution = "gaussian", n.trees = 100, shrinkage = 0.1, interaction.depth = 10 )
```

- Ensemble K-Nearest Neighbors (optimized)

```
ensemble_knn_model <- e1071::tune.gknn(x = ensemble_train, y = ensemble_train$y_ensemble, k = c(1:25), scale = TRUE)
```

- Ensemble Lasso (uses best model)

```
ensemble_best_lasso_lambda <- ensemble_lasso_cv$lambda.min
```

```
ensemble_best_lasso_model <- glmnet(ensemble_x, ensemble_y, alpha = 1, lambda = ensemble_best_lasso_lambda)
```

- Ensemble Linear (tuned)

```
ensemble_linear_train_fit <- e1071::tune.rpart(formula = y_ensemble ~ ., data = ensemble_train)
```

- Ensemble Random Forest (tuned)

```
ensemble_rf_train_fit <- e1071::tune.randomForest(x = ensemble_train, y = ensemble_train$y_ensemble, data = ensemble_train)
```

- Ensemble Ridge

```
ensemble_best_ridge_lambda <- ensemble_ridge_cv$lambda.min
```

```
ensemble_best_ridge_model <- glmnet(ensemble_x, ensemble_y, alpha = 0, lambda = ensemble_best_ridge_lambda)
```

- Ensemble RPart

```
ensemble_rpart_train_fit <- rpart::rpart(ensemble_train$y_ensemble ~ ., data = ensemble_train)
```

- Ensemble Support Vector Machines (tuned)

```
ensemble_svm_train_fit <- e1071::tune.svm(x = ensemble_train, y = ensemble_train$y_ensemble, data = ensemble_train)
```

- Ensemble Trees

```
ensemble_tree_train_fit <- tree::tree(ensemble_train$y_ensemble ~ ., data = ensemble_train)
```

- Ensemble XGBoost (optimized)

```
ensemble_xgb_model <- xgboost::xgb.train(data = ensemble_xgb_train, max.depth =
3, watchlist = ensemble_watchlist_test, nrounds = 70) ensemble_xgb_model_validation
<- xgboost::xgb.train(data = ensemble_xgb_train, max.depth = 3, watchlist = ensem-
ble_watchlist_validation, nrounds = 70)
```

- The stories in the models

4 Ensembles and individual model plots

- Most accurate model results:
 - Predicted vs actual
 - * Discussion of most accurate predicted vs actual
 - Residuals
 - * Discussion of residuals from the most accurate model
 - Histogram of residuals
 - * Discussion of the histogram of residuals from the most accurate model
 - Q-Q plot
 - * Discussion of the Q-Q plot from the most accurate model
 - Mean accuracy for train, test, validation, and holdout (mean of test and validation) sets
 - Barchart of model accuracy
 - * Measures mean Root Mean Squared Error, from low to high
 - * Includes 1 standard deviation error bars
 - Insert model accuracy numbers with standard deviations
 - Insert model accuracy bar chart with 1 standard deviation error bars here
- Bias
 - `bias` computes the average amount by which `actual` is greater than `predicted`.
 - If a model is unbiased `bias(actual, predicted)` should be close to zero. Bias is calculated by taking the average of `(actual - predicted)`.
 - Insert bias plot here

- Holdout / Train
 - Calculates the mean of the holdout RMSE / mean of the train RMSE across the samples. Closer to one is better.
 - Insert table for holdout / train numbers
 - Insert plot for holdout / train chart here
- Duration
 - Calculates the mean duration for each model
 - Insert barchart for duration here
- t-test with p-values
 - Performs one and two sample t-tests on vectors of data
 - Insert t-test numbers
 - Insert plot for t-test results
- Barchart of Kolmogorov-Smirnov test by model
 - `stats::ks.test(x = y_hat_bag_rf, y = c(trainy, validationy), exact = TRUE)$statistic`
 - Tests if the holdout data from each of the 40 models came from the same distribution as the raw data. If `y` is numeric, a two-sample (Smirnov) test of the null hypothesis that `x` and `y` were drawn from the same distribution is performed.
 - Lines are given for $p = 0.05$ and $p = 0.10$,
- Variance Inflation Factor report
 - Calculates variance-inflation and generalized variance-inflation factors (VIFs and GVIFs) for linear, generalized linear, and other regression models.
- Correlation of the data
 - Presents a table of the correlation of the data.
- Correlation of the ensemble
 - Presents a table of the correlation of the ensemble. This can be modified in the function call, such as `remove_ensemble_correlations_greater_than = 0.98` (or whatever value is most useful)
- Summary report
- Function call

- Warnings or errors
- The stories in the plots

5 Strongest evidence based data:

- Most accurate models with error ranges
- Strongest predictor with error ranges
- The stories of the strongest evidenced based data

6 Five strongest evidence based recommendations

7 Conclusions