1 How To Use This Document

Highly regulated industries, such as banking and insurance, must comply with government regulations for model validation before a model can be put into production. This includes creating robust model development documentation. DataRobot automates the generation of model documentation, expediting the process required for regulatory compliance and following best practice for reducing model risk.

This document is split into two components: those sections that are automatically produced by DataRobot and those that require further input by the user. The sections in blue italicized font include specific instructions for the documenter and require additional user input of organization-specific information, such as business use cases, data sources, and implementation details. Once the sections are complete, remove the instructions. The remaining sections in non-blue italicized font are automatically populated by DataRobot and require no further input.

Copyright ©2024, DataRobot, Inc.



Table of Contents

* 1 How To Use This Document
* 2 OVERVIEW AND SUMMARY
* 2.1 Model Summary and Description
* 2.2 Model Results
* 2.3 Model Version Control
* 3 MODEL DATA OVERVIEW
* 3.1 Model Features Summary
* 3.1.1 Model Features and Summary Statistics
* 3.2 Feature Association
* 3.3 Feature Impact Chart
* 3.4 Feature Impact Table
* 3.5 Data Partitioning
* 3.6 Missing Values
* 4 MODEL PERFORMANCE
* 4.1 Model Performance Overview
* 4.2 Validation Testing and Stability
* 4.3 Sensitivity Analysis
* 4.4 Lift Chart (CUSTOMIZED COMPONENT)
* 4.5 Lift Chart
* 4.6 ROC Curve
* 4.6.1 ROC Curve
* 4.6.2 Confusion Matrix
* 4.6.3 Prediction Distribution
* 5 THEORETICAL FRAMEWORK AND METHODOLOGY
* 5.1 DataRobot Modeling Overview
* 5.2 Model Methodology
* 5.2.1 Custom Learning Classifier
* 5.3 Literature Review
* 5.4 Challenger Model Performance

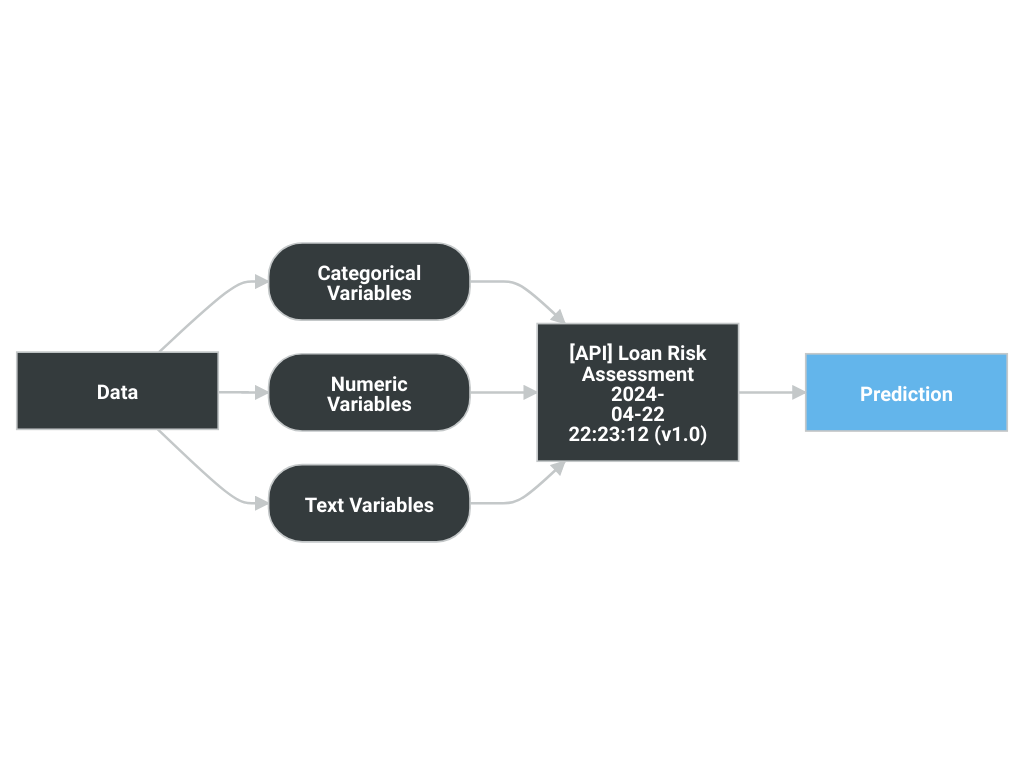
2 OVERVIEW AND SUMMARY

Insert section information here.

2.1 Model Summary and Description

The particular model referenced in this document: [API] Loan Risk Assessment 2024-04-22\_22:23:12 (v1.0). This model was developed in a project created with v501ca4b03ce2dabc of DataRobot. This model is denoted within DataRobot by the Project ID: 6626e3528e3046e0cc7dad12 and the Model ID: 6626e3538e3046e0cc7dad14. The project was created on 2024-04-22 22:23:14.

The model development workflow process (i.e., the model blueprint) is detailed in the figure below.



A Blueprint represents the high-level end-to-end procedure for fitting the model, including any preprocessing steps, algorithms, and post-processing. It illustrates the many steps involved in transforming input predictors and targets into a model. Each element (or, “node”) in a blueprint can represent multiple steps.

The following elements connect to create the blueprint:

* [API] Loan Risk Assessment 2024-04-22\_22:23:12 (v1.0)

2.2 Model Results

DataRobot runs performance testing during the model development process to evaluate model results and reliability. The validation, cross-validation, and holdout (if applicable) out-of-sample performance scores are presented below, as well as the number of observations for each partition. The performance metric used for this project was LogLoss and the project included a total of 10,000 observations. An asterisk (\*) next to a score, whether validation or holdout, indicates that DataRobot used in-sample predictions to derive the score. (In-samples predictions are those that include data from the validation or holdout partitions due to sample size used to build the model.)

|  |  |
| --- | --- |
| Scoring Type | Score (LogLoss) |
| holdout | 0.4014 |
| validation | 0.4095 |

2.3 Model Version Control

DataRobot handles model and project version control automatically by tagging each model on the Leaderboard with a unique Model ID. The Model ID represents a single instance of a model type, feature list, sample size, and set of tuning parameter values. DataRobot also maintains unique Project IDs for each project, allowing accessibility to all models built for the project dataset. DataRobot's version control allows for reproducibility and traceability of the models it creates, which greatly increases the auditability of the model development process.

Users may also export Scoring Code for a DataRobot model in Java. You can download both a pre-compiled .jar file (with all dependencies included), plus the source code. Scoring Code is easy to deploy, test, and maintain on a variety of platforms, and you can inspect the generated Java code for complete transparency. DataRobot Scoring Code employs advanced features to ensure that predictions computed using generated Java code are the same as predictions computed inside DataRobot.

3 MODEL DATA OVERVIEW

Insert section information here.

3.1 Model Features Summary

Below is a summary of modeling features provided from the user defined training data, and attached to this model.

The Model Features and Summary Statistics table provides a brief overview of the summary statistics of model features. This includes Feature Name, variable type (Var Type), number of unique values (Unique), Number of missing values (Missing), Mean, Standard Deviation (Std Dev), Median, Minimum Value (Min), Maximum Value (Max) and Assessment of target leakage risk (Target Leakage).

3.1.1 Model Features and Summary Statistics

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Feature Name | Var Type | Unique | Missing | Mean | Std Dev | Median | Min | Max | Target Leakage |
| loan\_amnt | Numeric | 521 | 0 | 11028.42 | 7449.32 | 9600.0 | 500.0 | 35000.0 | Low |
| funded\_amnt | Numeric | 666 | 0 | 10752.49 | 7164.62 | 9225.0 | 500.0 | 35000.0 | Low |
| term | Categorical | 2 | 0 | N/A | N/A | N/A | N/A | N/A | Low |
| int\_rate | Numeric | 349 | 0 | 12.13 | 3.73 | 11.86 | 5.42 | 23.91 | Low |
| installment | Numeric | 5430 | 0 | 321.38 | 210.36 | 275.07 | 15.69 | 1276.6 | Low |
| grade | Categorical | 7 | 0 | N/A | N/A | N/A | N/A | N/A | Low |
| sub\_grade | Categorical | 35 | 0 | N/A | N/A | N/A | N/A | N/A | Low |
| emp\_title | Text | 6645 | 482 | N/A | N/A | N/A | N/A | N/A | N/A |
| emp\_length | Categorical | 11 | 212 | N/A | N/A | N/A | N/A | N/A | Low |
| home\_ownership | Categorical | 5 | 0 | N/A | N/A | N/A | N/A | N/A | Low |
| annual\_inc | Numeric | 1642 | 1 | 68325.71 | 48948.908 | 58000.0 | 4080.0 | 900000.0 | Low |
| verification\_status | Categorical | 3 | 0 | N/A | N/A | N/A | N/A | N/A | Low |
| pymnt\_plan | Categorical | 2 | 0 | N/A | N/A | N/A | N/A | N/A | N/A |
| url | Text | 8000 | 0 | N/A | N/A | N/A | N/A | N/A | N/A |
| desc | Text | 5405 | 2591 | N/A | N/A | N/A | N/A | N/A | N/A |
| purpose | Categorical | 14 | 0 | N/A | N/A | N/A | N/A | N/A | Low |
| title | Text | 4682 | 4 | N/A | N/A | N/A | N/A | N/A | N/A |
| zip\_code | Categorical | 692 | 0 | N/A | N/A | N/A | N/A | N/A | Low |
| addr\_state | Categorical | 49 | 0 | N/A | N/A | N/A | N/A | N/A | Low |
| dti | Numeric | 2490 | 0 | 13.28 | 6.78 | 13.3 | 0.0 | 29.99 | Low |
| delinq\_2yrs | Numeric | 10 | 5 | 0.15 | 0.509 | 0.0 | 0.0 | 11.0 | Low |
| earliest\_cr\_line | Date | 450 | 5 | 1997-07-10 | 2785.46 days | 1998-08-01 | 1969-03-01 | 2068-12-01 | Low |
| inq\_last\_6mths | Numeric | 17 | 5 | 1.06 | 1.45 | 1.0 | 0.0 | 25.0 | Low |
| mths\_since\_last\_delinq | Numeric | 89 | 5072 | 35.82 | 22.38 | 34.0 | 0.0 | 120.0 | Low |
| mths\_since\_last\_record | Numeric | 90 | 7323 | 61.72 | 46.11 | 86.0 | 0.0 | 119.0 | Low |
| open\_acc | Numeric | 36 | 5 | 9.34 | 4.53 | 9.0 | 1.0 | 39.0 | Low |
| pub\_rec | Numeric | 4 | 5 | 0.0608 | 0.25 | 0.0 | 0.0 | 3.0 | Low |
| revol\_bal | Numeric | 6699 | 0 | 14168.41 | 25890.97 | 8545.5 | 0.0 | 1207359.0 | Low |
| revol\_util | Numeric | 1022 | 19 | 48.26 | 28.305 | 48.3 | 0.0 | 108.8 | Low |
| total\_acc | Numeric | 71 | 5 | 22.049 | 11.63 | 20.0 | 1.0 | 90.0 | Low |
| initial\_list\_status | Categorical | 1 | 0 | N/A | N/A | N/A | N/A | N/A | N/A |
| policy\_code | Numeric | 1 | 0 | 1.0 | 0.0 | 1.0 | 1.0 | 1.0 | N/A |
| loan\_status | Numeric | 2 | 0 | 0.13 | 0.33 | 0.0 | 0.0 | 1.0 | N/A |
| Partition | Categorical | 2 | 0 | N/A | N/A | N/A | N/A | N/A | Low |

The last column in this table is an assessment of target leakage risk. DataRobot automatically tests for target leakage on a per-feature basis during the Autopilot process. Target leakage, sometimes called data leakage, occurs when a model is trained using a dataset that includes information that would not be available at the time of prediction. This can produce overly optimistic model performance results during training, given a feature will near-completely describe the target (e.g., the number of late payments on a loan as a predictor for loan default at loan application date.)

DataRobot tests for target leakage risk using Alternating Conditional Expectation (ACE) to measure the association between each feature and the target; the ACE score is normalized using the project optimization metric so that its value is in the range [0,1]. DataRobot converts this score into a target leakage risk category using the thresholds below. For features at high risk of target leakage, consider removing them from the training data set. For features at moderate risk for target leakage, subject matter expertise should be applied to determine if they are safe to include.

The thresholds for target leakage risk are based on a normalized ACE score:

* High risk: > 0.975, consider removing
* Moderate risk: > 0.85, apply subject matter expertise
* Low risk: < 0.85, no action

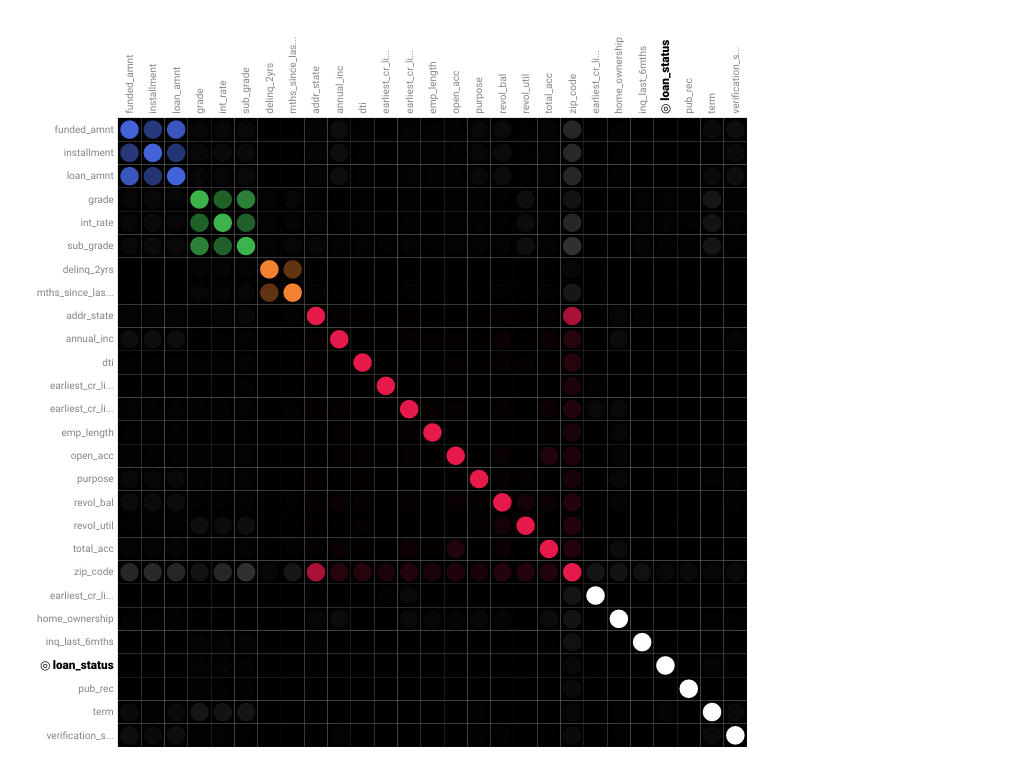
3.2 Feature Association

DataRobot’s Feature Association Matrix is populated by default using the user defined training data. The Feature Association Matrix provides information on association strength between pairs of numeric and categorical features that are visually denoted by the opacity of the color (that is, num/cat, num/num, cat/cat, where lighter shades indicate weaker association and vice versa) and feature clusters. Clusters, families of features denoted by color on the matrix, are features partitioned into groups based on their association structure.

Some of the noted benefits of the Feature Association Matrix include:

* Understand the strength and nature of associations within the data;
* Detect families of pairwise association clusters; and,
* Identify clusters of high-association features prior to model building.

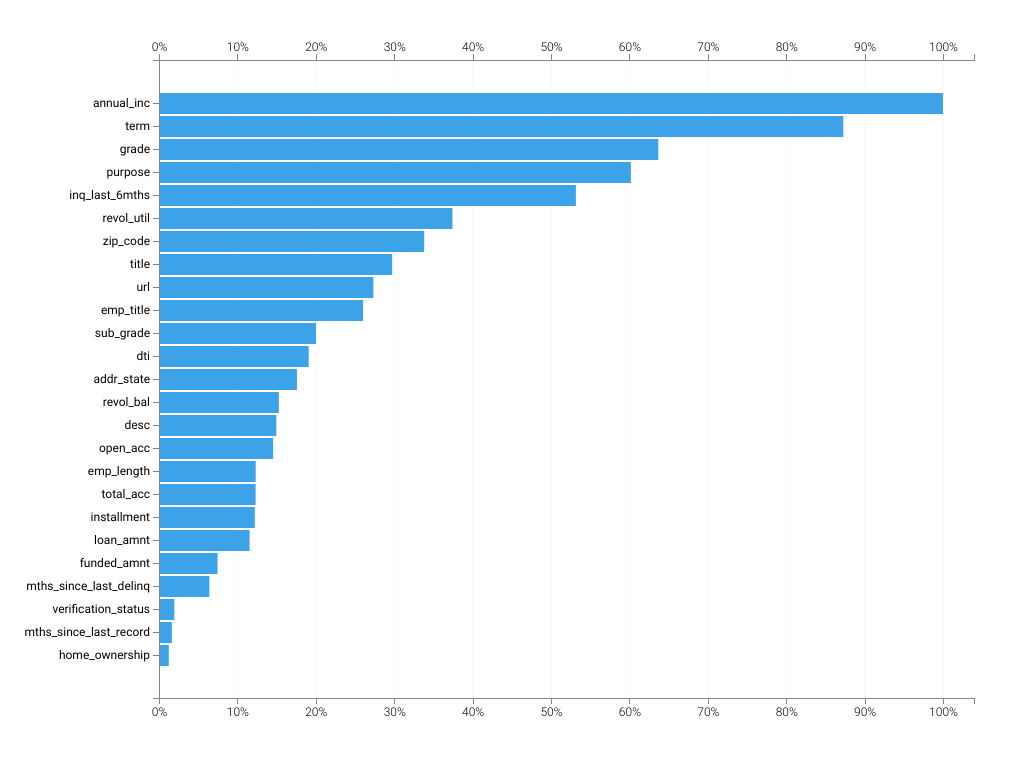
The Feature Association Matrix lists up to the top 50 features, selected by Importance Score, on both the X and Y axes, where the intersection of a feature pair provides an indication of their level of association. By default, the matrix displays by the Mutual Information values and sorts by the cluster.



The following are some general takeaways from looking at the matrix above:

* Each dot represents the association between two features (a feature pair), where the opacity of the color denotes the pair-wise strength of association.
* Each cluster is represented by a different color.
* The opacity of color indicates the level of association 0 to 1, between the feature pair. Levels are measured by the set metric, either mutual information or Cramer’s V.
* Shaded gray dots indicate that the two features, while showing some association, are not in the same cluster.
* White dots represent features that were not categorized into a cluster.
* The target feature, if present, is indicated by two small concentric circles next to the feature name.

3.3 Feature Impact Chart



3.4 Feature Impact Table

|  |  |  |
| --- | --- | --- |
| Feature Name | Impact Normalized | Impact Unnormalized |
| annual\_inc | 1.0 | 0.0267 |
| term | 0.8728 | 0.0233 |
| grade | 0.6368 | 0.017 |
| purpose | 0.6019 | 0.0161 |
| inq\_last\_6mths | 0.5317 | 0.0142 |
| revol\_util | 0.3743 | 0.01 |
| zip\_code | 0.3382 | 0.009 |
| title | 0.2974 | 0.0079 |
| url | 0.2734 | 0.0073 |
| emp\_title | 0.2603 | 0.007 |
| sub\_grade | 0.2003 | 0.0054 |
| dti | 0.1909 | 0.0051 |
| addr\_state | 0.1759 | 0.0047 |
| revol\_bal | 0.1529 | 0.0041 |
| desc | 0.1496 | 0.004 |
| open\_acc | 0.1455 | 0.0039 |
| emp\_length | 0.1233 | 0.0033 |
| total\_acc | 0.1232 | 0.0033 |
| installment | 0.1222 | 0.0033 |
| loan\_amnt | 0.1155 | 0.0031 |
| funded\_amnt | 0.0746 | 0.002 |
| mths\_since\_last\_delinq | 0.0642 | 0.0017 |
| verification\_status | 0.0195 | 0.0005 |
| mths\_since\_last\_record | 0.0164 | 0.0004 |
| home\_ownership | 0.0125 | 0.0003 |
| delinq\_2yrs | 0.0016 | 0.0 |
| initial\_list\_status | 0.0 | 0.0 |
| int\_rate | 0.0 | 0.0 |
| policy\_code | 0.0 | 0.0 |
| pub\_rec | 0.0 | 0.0 |
| pymnt\_plan | 0.0 | 0.0 |
| earliest\_cr\_line | -0.0033 | -0.0001 |

3.5 Data Partitioning

The data partitions were defined by the user according to the column Partition.

3.6 Missing Values

No data about missing values handling is available. Some model types do not provide this information. Also older versions of DataRobot have not gathered this information, if this is a pre-upgrade model - please retrain it to have access to missing values information.

4 MODEL PERFORMANCE

Insert section information here.

4.1 Model Performance Overview

As an additional layer of model validity, DataRobot not only evaluated the statistical metrics underlying the model, but also performed testing on out-of-sample records.

The performance metric used for this project was LogLoss. The model performance results are presented below for out-of-sample testing:

|  |  |
| --- | --- |
| Scoring Type | Score (LogLoss) |
| holdout | 0.4014 |
| validation | 0.4095 |

4.2 Validation Testing and Stability

To find patterns in a dataset from which it can make predictions, an algorithm must first learn from a historical example – typically from a historical dataset that contains the output variable you want to predict. However, if a model is trained too closely on its training data then it may be overfit. Overfitting is a modeling error that occurs when a model is too closely fit to training data and therefore performs poorly on out-of-sample data (data that was not used to train the model). Overfitting generally results in an overly complex model that explains idiosyncrasies and random noise in the training data, rather than the underlying trends that the model was intended to capture. To avoid overfitting, the best practice is to evaluate model performance on out-of-sample data. If the model performs very well on in-sample data, (the training data) but poorly on out-of-sample data, that may be an indication that the model is overfit.

DataRobot uses standard modeling techniques to validate model performance and ensure that overfitting does not occur. DataRobot used a robust model k-fold cross-validation framework to test the out-of-sample stability of a model's performance. In addition to cross-validation partitioning, DataRobot uses a holdout sample to further test out-of-sample model performance and ensure the model is not overfit.

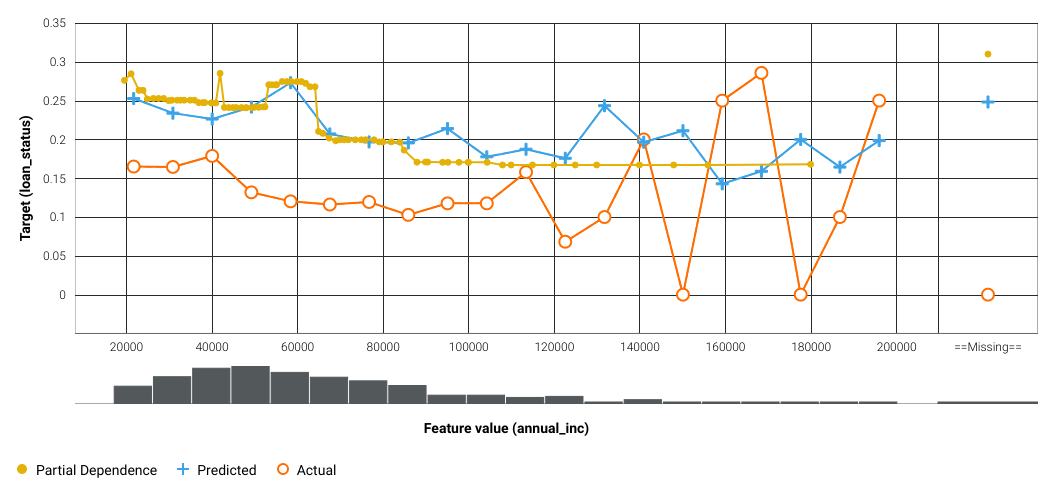
The following procedure was used during development to insure that overfitting did not occur:

* A Training, Validation, and Holdout modeling approach uses a training data partition to train the model, a validation partition to assess model performance, and a holdout partition assess the generalization error of the final model to ensure the model is not biased or overfit.
* This project did not use a holdout partition.
* A validation partition consisting of all observations with the value of "V" for the feature: "Partition"
* The remaining observations, consisting of all observations with the value of "T" for the feature "Partition", were used to train the models

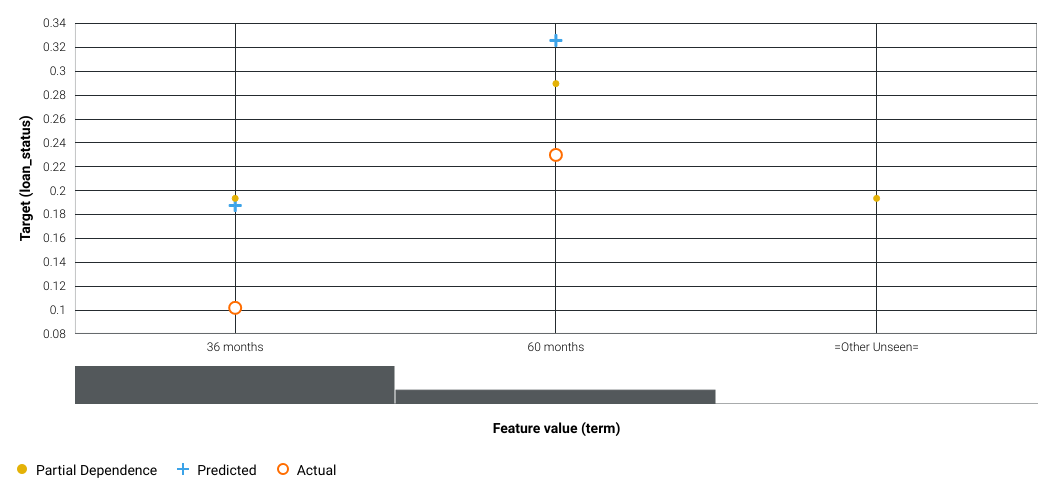
4.3 Sensitivity Analysis

Partial dependence plots show the average partial relationship between a set of predictors and the predicted response. The partial dependence plots below capture the top features in our model, as measured by Feature Impact.

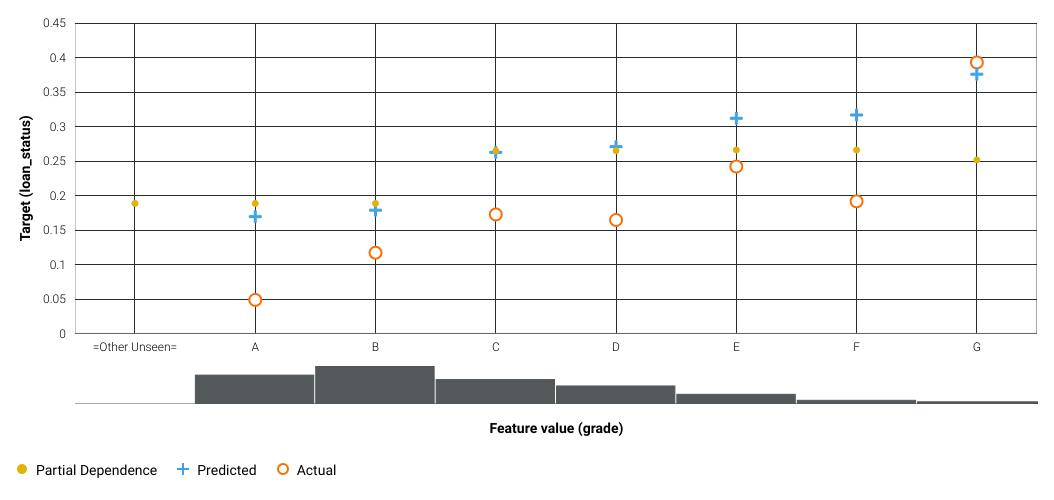
annual\_inc



term



grade



The orange circles depict, for the selected feature, the average target value for the aggregated feature values. The blue crosses depict, for the selected feature, the average prediction for a specific value. From the graph you can see that DataRobot also averages the predicted feature values. Comparing the actual and predicted points can identify segments where model predictions differ from observed data. This typically occurs when the segment size is small. In those cases, for example, some models may predict closer to the overall average.

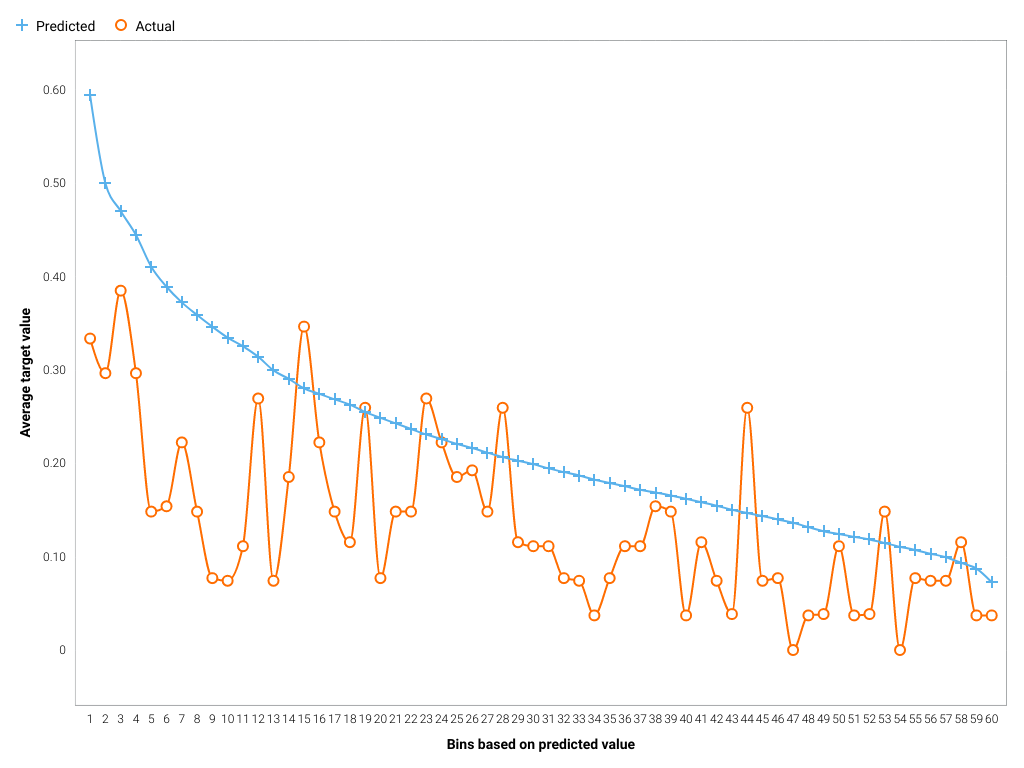
The yellow partial dependence data points depict the marginal effect of a feature on the target variable after accounting for the average effects of all other predictive features. It indicates how, holding all other variables constant, the value of this feature affects prediction. DataRobot holds constant the values of all columns in the sample except the feature of interest. The value of the feature of interest is then reassigned to each possible value, calculating the average predictions for the sample at each setting. These values help determine how the value of each feature affects the target. The shape of the yellow data points describes the model's view of the marginal relationship between the selected feature and the target.

4.4 Lift Chart (CUSTOMIZED COMPONENT)

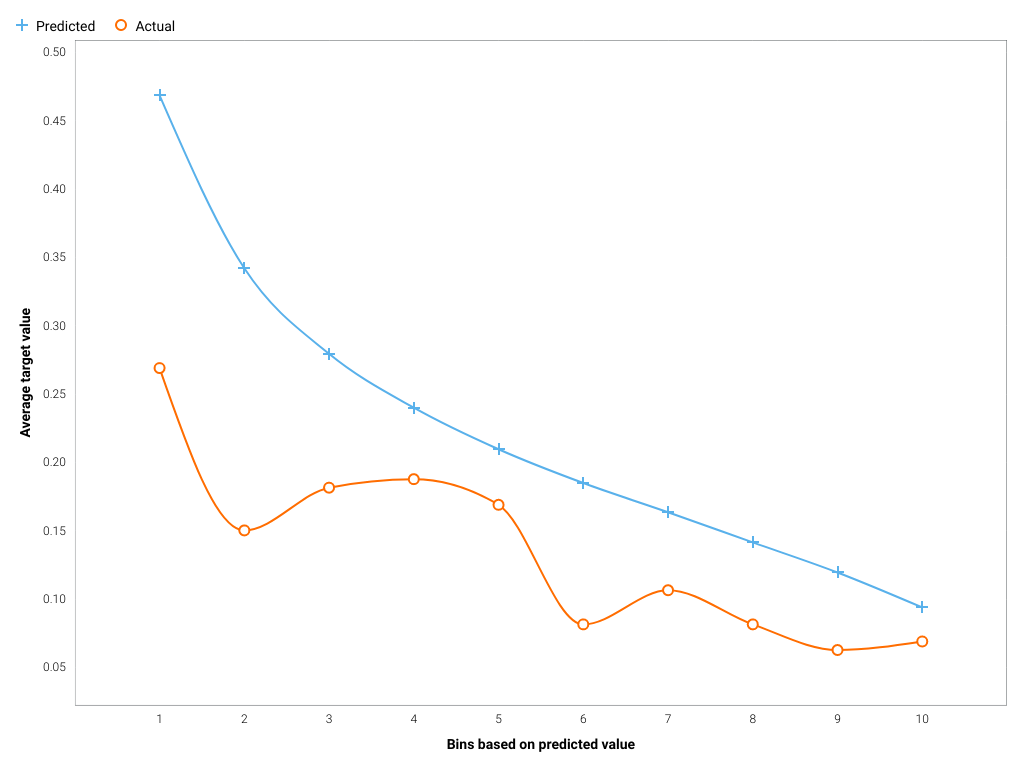
The Lift Chart sorts and groups numeric feature values into equal-sized bins, depicting how well a model segments the target population and how capable it is of predicting the target.

The lift chart can be customized using tags, for example, "{ { lift\_chart | reverse=True;bins=60;source=validation} }".

The following chart shows the lift chart of the validation data used for model building in reverse order and 60 bins.

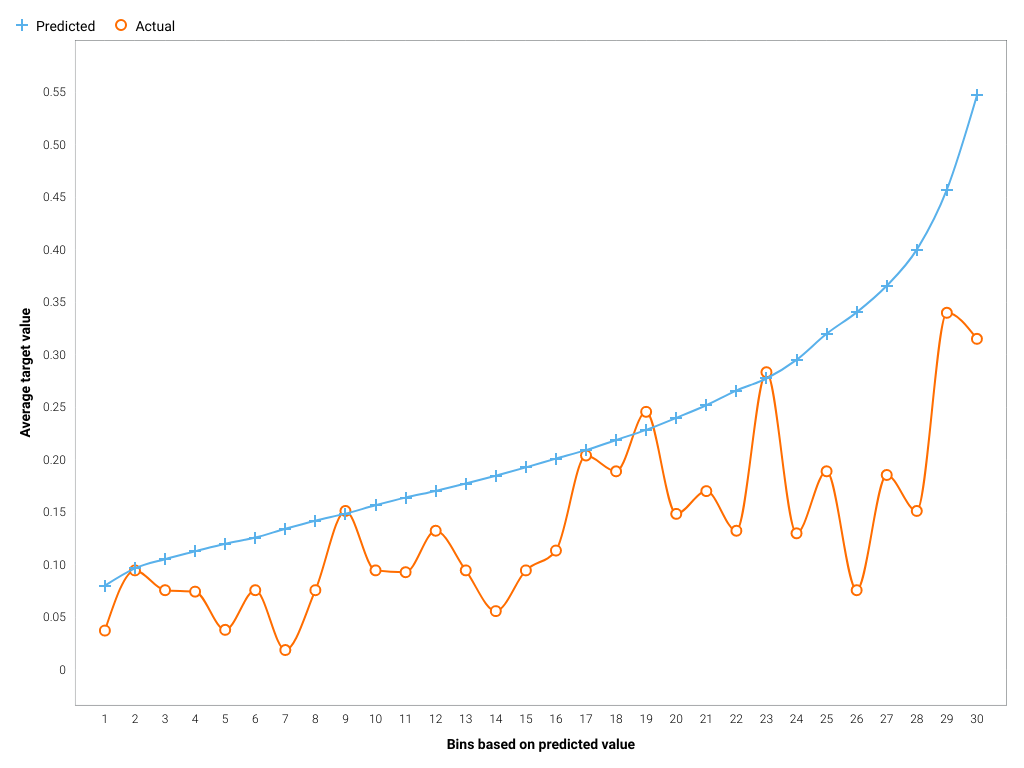


The following chart shows the lift chart of the validation data used for model building in reverse order and 10 bins.



4.5 Lift Chart

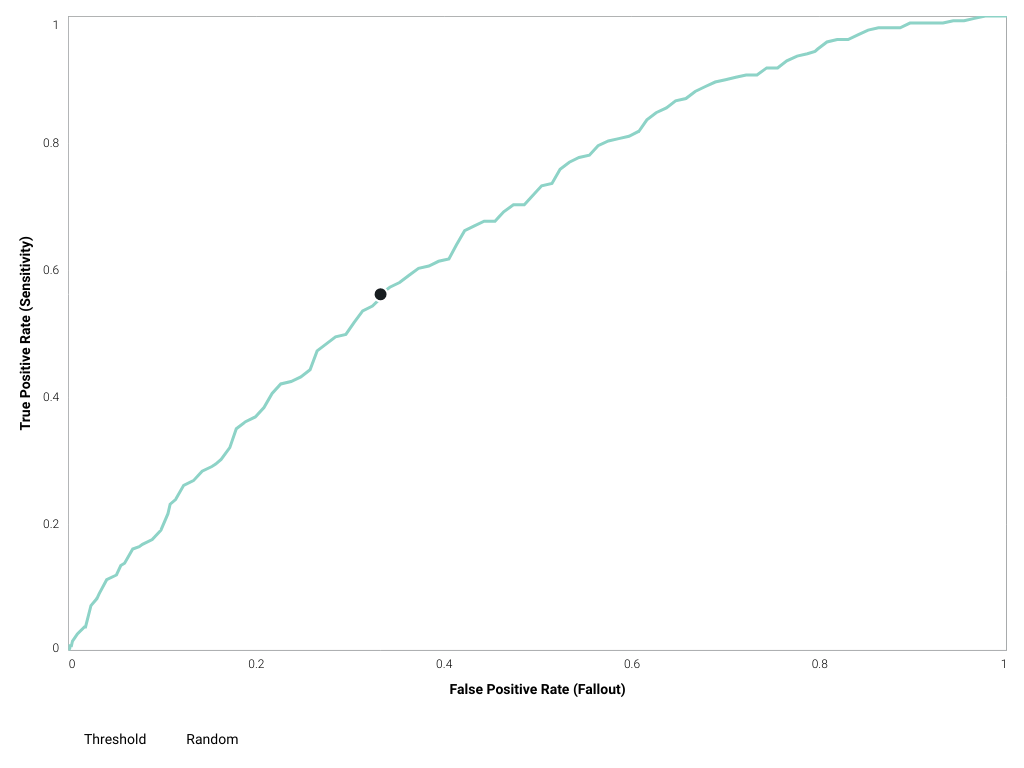
The Lift Chart sorts and groups numeric feature values into equal sized bins, depicting how well a model segments the target population and how capable it is of predicting the target. This helps the user to visualize model accuracy for each bin. The chart is sorted by predicted values -- lowest to highest predictions, for example -- which provides transparency to the model performance for different ranges of values of the target variable. Looking at the Lift Chart, the left side of the curve indicates where the model predicted a low score on one section of the population while the right side of the curve indicates where the model predicted a high score. The model Lift Chart is presented in the figure below.



The points on the Lift Chart indicate the average percentage in each bin. The "Predicted" blue line displays the average prediction score for the rows in that bin. The "Actual" orange line displays the actual percentage for the rows in that bin. In general, the steeper the Actual line is, and the more closely the Predicted line matches the actual line, the better the model. A close relationship between these two lines is indicative of the predictive accuracy of the model; a consistently increasing line is another good indicator of satisfactory model performance.

4.6 ROC Curve

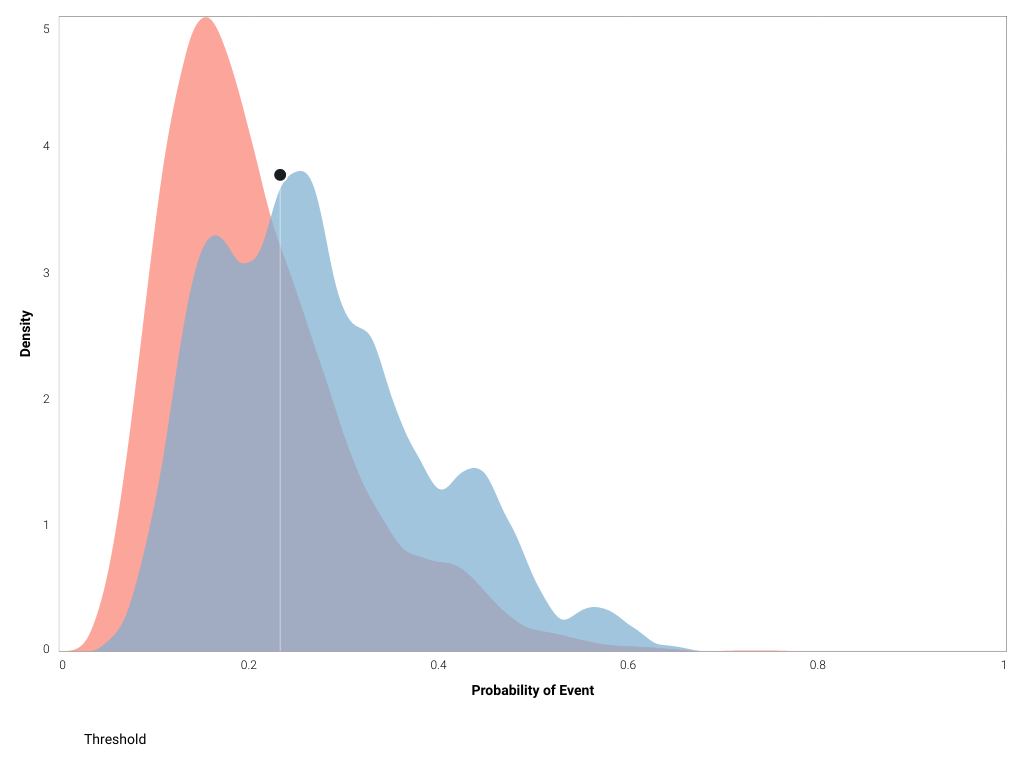
4.6.1 ROC Curve



4.6.2 Confusion Matrix

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| F1 Score | True Positive Rate | False Positive Rate | True Negative Rate | Positive Predictive Value | Negative Predictive Value | Accuracy | Matthews Correlation Coefficient |
| 0.3032 | 0.5613 | 0.3328 | 0.6672 | 0.2077 | 0.9073 | 0.653 | 0.1621 |

4.6.3 Prediction Distribution



5 THEORETICAL FRAMEWORK AND METHODOLOGY

Insert section information here.

5.1 DataRobot Modeling Overview

DataRobot simplifies model development by performing a parallel heuristic search for the best model or ensemble of models, based on both the characteristics of the data and the prediction target. While some machine learning techniques tend to consistently outperform others, it is rarely possible to say in advance which will perform best for a given business problem. Therefore, during the modeling process, DataRobot develops dozens of independent challenger models, exposes the details of how these models were built and how they perform, and enables the user to select the best model for the particular business problem being addressed.

The fundamental workflow within DataRobot for model development is as follows:

* Rapid Data Ingestion: User creates a modeling dataset that includes the prediction target and loads into DataRobot
* Target Selection: User selects the prediction target; DataRobot detects whether the target is categorical or continuous. If the target is categorical, DataRobot selects and builds classification blueprints. If the target is continuous, DataRobot selects and builds regression blueprints. DataRobot also selects an optimization performance metric based on the type of supervised learning problem, which can also be changed by the user
* Automated Data Preparation: DataRobot analyzes the input data and automatically performs advanced preprocessing steps that are discussed in detail in this document. DataRobot also automatically partitions the input dataset into learning, validation and holdout dataset; these can also be defined by the user.
* DataRobot uses information about the selected target variable and predictors to define a set of candidate blueprints for analysis. It then trains models for each blueprint and ranks them on the model Leaderboard based on an out-of-sample validation accuracy score.
* Transparent Model Evaluation and Selection: DataRobot has built-in diagnostic tools to assess model accuracy and performance. Once DataRobot has trained and tested models, users can access them from the Leaderboard. From there, users can review model accuracy and, using built-in model diagnostic tools, understand how each independently built model performs. DataRobot provides many metrics for evaluating model accuracy, such as AUC, Log-Loss and RMSE. DataRobot's Leaderboard actively tracks performance of candidate models using out-of-sample data for comparison purposes.
* Model Deployment and Monitoring: Once the final model is selected, DataRobot provides efficient solutions for deployment (i.e., model implementation) and monitoring. These features enable the model owner to effectively manage model controls in accordance with Model Risk Management standards and policies.

5.2 Model Methodology

The modeling workflow consists of the following elements, which connect to create the blueprint:

* [API] Loan Risk Assessment 2024-04-22\_22:23:12 (v1.0)

The following subsections include details for each node of the modeling blueprint.

5.2.1 Custom Learning Classifier

Custom Tasks enable users to create their own algorithms and use them within DataRobot along side all the models supplied by DataRobot.

Custom Tasks implement the fit and predict methods. Fit is implemented to to run in kubernetes and produces a tarball of model artifacts.

Predict is implemented as a long running web-server application. This application implements a predict route capable of making predictions on the trained task. When started, the contents of the tarball from fit is loaded into the application so that the user’s code can use the artifacts to make predictions.

The predictions from a Custom Classifier’s prediction application are expected to be a list of dictionaries. Each dictionary must contain the class labels mapped to their probabilities. Each row must have a probability for all labels.

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Name | Description | Best Searched |
| dictionary | class\_mapping | INTERNAL PARAMETER, NOT USER-TUNABLE. The class mapping dictionary. This maps string class labels to the numerical class labels used internally within DataRobot. For binary classification, the positive class label is mapped to 1.0 and the negative class label is mapped to 0.0. | None |
| string | version\_id | The ID of the custom model version to use for model training and predictions. | 6626e3538e3046e0cc7dad14 |

5.3 Literature Review

5.4 Challenger Model Performance

As stated by regulatory guidance, comparison with alternative theories and approaches provides guidance for final model selection and is a fundamental component of a sound modeling process.

DataRobot develops dozens of alternative models, exposes the details of how these models were built and how they perform, and enables the user to select the best model for the particular business problem being addressed.

During the model development process, DataRobot considered the following alternative models. The final model was selected based on model performance as well as an analysis of model diagnostics and expert business judgment.

The performance metric used for this project was LogLoss. The model types considered during the model selection process included the following models, which are sorted by the Holdout score.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model Name | Validation Score | Cross Validation Score | Holdout Score | Sample Percentage |