**Contents**

1. Executive Summary
2. Initial Data Exploration
3. Splitting and Feature Engineering Results
4. Training, Testing & Evaluation

* KS Plot
* ROC curve for Logistic Regression
* Lifi Chart

1. Operational Metrics Computation & Scores Transformation
2. Conclusion
3. Appendix

# 

# MVP Executive Summary

This project report presents an analysis based on a scenario when a financial institution examines a request for a loan, assessing the risk of default to determine whether to grant it. For example, with simulated data (containing the borrower's financial history and information about the requested loan), a small personal loan financial institution can make use of predictive analytics to help decide whether or not to grant a loan for each borrower, thereby reducing the number of loans it offers to those borrowers most likely to default, increasing the profitablity of its loan portfolio.

This analysis pre-processes data (cleaning and feature engineering), trains prediction models, and performs scoring on an HDInsight Spark cluster with Microsoft R Server. HDInsight is a cloud Spark and Hadoop service for the enterprise. HDInsight is also the only managed cloud Hadoop solution with integration to Microsoft R Server. This analysis uses historical data to predict a model for future loans in R Server for HDInsight. It chooses to train a Logistic Regression, a standard model used in the Credit Score industry (contrary to more complex models such as random forests or neural networks, it is easily understandable through the simple formula generated during the training).

The steps (described below), each create a function to perform their task. The individual steps 1-4 are described in more detail below. The following scripts are then used to execute the steps.

* In creating the model and score test data, **development\_main.R** script is run, invoking steps 1-4 described further below.

The default input for this script generates 10,000 rows for training models, and will split this into train and test data. After running this script, the data files are in the /var/RevoShare/<username>/LoanCreditRisk/dev/temp directory.

Models are stored in the /var/RevoShare/<username>/LoanCreditRisk/dev/model directory.

The Hive table ScoresData contains the the results for the test data.

Finally, the model is copied into the /var/RevoShare/<username>/LoanCreditRisk/prod/model directory for use in production mode.

* After completing the model, **production\_main.R** script is run next, invoking steps 1, 2, and 3 using the production mode setting. **production\_main.R** uses the previously trained model and invokes the steps to process data, perform feature engineering and scoring. The input to this script defaults to 22 applicants to be scored with the model in the prod directory. After running this script the Hive table ScoresData\_Prod now contains the scores for these applicants.
* Once all the above code has been executed, a PowerBI dashboard, as an example, can then be used to visualize the scores created from the model.

Below is a summary of the individual steps 1-4 invoked, when running the main scripts.

* The first few steps prepare the data for training.
  + **step1\_preprocessing.R**: Uploads data and performs preprocessing steps -- merging of the input data sets and missing value treatment.
  + **step2\_feature\_engineering.R**: Creates the label isBad based on the status of the loan, splits the cleaned data set into a Training and a Testing set, and bucketizes all the numeric variables, based on Conditional Inference Trees on the Training set.
* **step3\_train\_score\_evaluate.R:** Trains a logistic regression classification model on the training set, and saves it. In development mode, this script then scores the logisitic regression on the test set and evaluates the tested model. In production mode, the entire input data is used and no evaluation is performed.
* **step4\_operational\_metrics.R:** Computes the expected bad rate for various classification decision thresholds and applies a score transformation based on operational metrics.
  + - * After step4, the development script runs **copy\_dev\_to\_prod.R** to copy the model information from the **dev** folder to the **prod** folder for use in production or web deployment.
* **deployment\_main.R** creates a web service and test it on the edge node.

With the predictions created, as an example, Power BI Dashboard, can be used to examine the test data to find an appropriate score cutoff, then use that cutoff value for a new set of loan applicants.

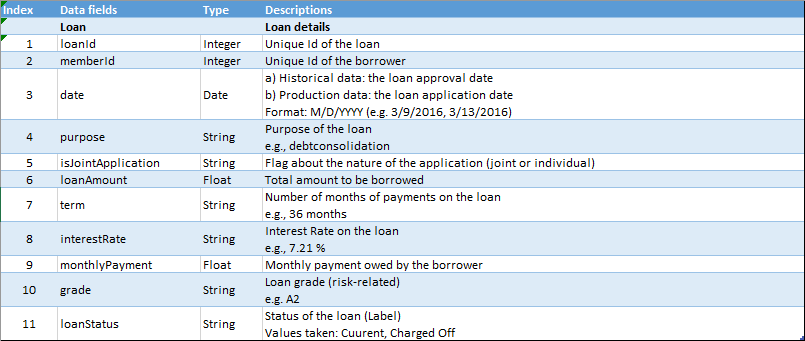
# Initial Data Exploration

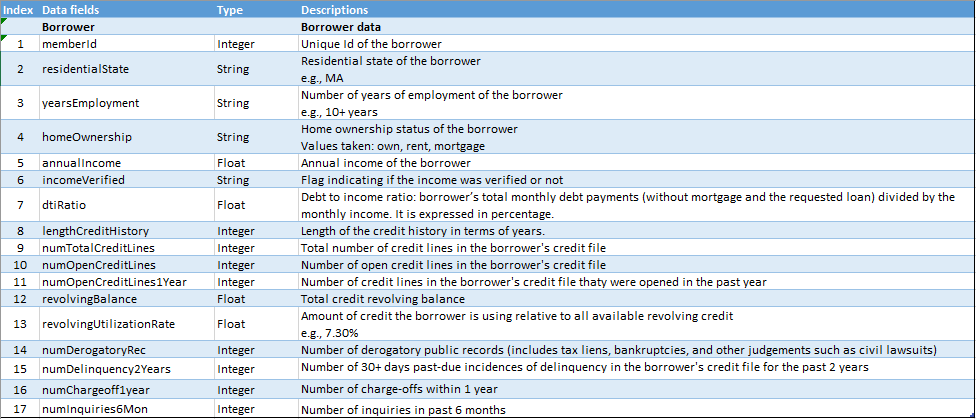
This Loan Credit Risk project has the following CSV files, consisting the following:

Loan.csv and Borrower.csv: data sets of Loan and Borrower (respectively), with 100K rows of the simulated data used to build the end-to-end Loan Credit Risk solution.

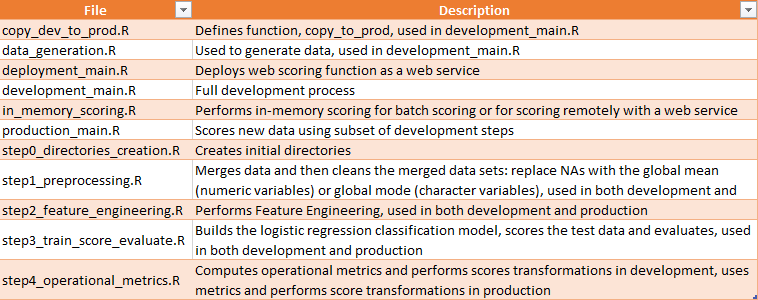
Loan\_Prod.csv and Borrower\_Prod.csv: data sets of Loan and Borrower (respectively), with 22 rows of the simulated data used in the Production pipeline.

The schema and description of the input tables and variables are as follows:





R codes/scripts/files used are as follows:



Firstly, there will be creating or cleaning of intermediate directories both on the edge node and HDFS. These directories will hold all the intermediate processed data sets in subfolders and will be performed using R-scripts in step0\_directories\_creation.R file.

Next, there will be merging and cleaning of data. In order to speed up the computations in this step and the following ones, the input data is first converted to .xdf files stored in HDFS. Characters are converted to factors at the same time. The xdf files are then merged with the rxMerge function, which writes the xdf result to the HDFS directory “Merged”. Finally, the missing values are filled in the merged table. Missing values of numeric variables are filled with the global mean, while character variables are filled with the global mode, achieved by the following way:

* Use rxSummary function on the HDFS directory holding the merged table xdf files. This will give the names of the variables with missing values, their types, the global means, as well as counts table through which the global modes are computed.
* These statistics information are saved to be used for the Production or Web Scoring stages, in the directory LocalModelsDir.
* If no missing values are found, the merged data splits are copied to the folder “MergedCleaned” on HDFS, without missing value treatment.

If there are missing values, the following are performed:

* + - * Compute the global means and modes for the variables with missing values by using the rxSummary results.
      * Define the “Mean\_Mode\_Replace” function which will deal with the missing values. It will be called in rxDataStep function which acts on the xdf files of “Merged”.
      * Apply the rxDataStep function.

Eventually, there are the cleaned splits of the merged table, “MergedCleaned” on HDFS.

Input:

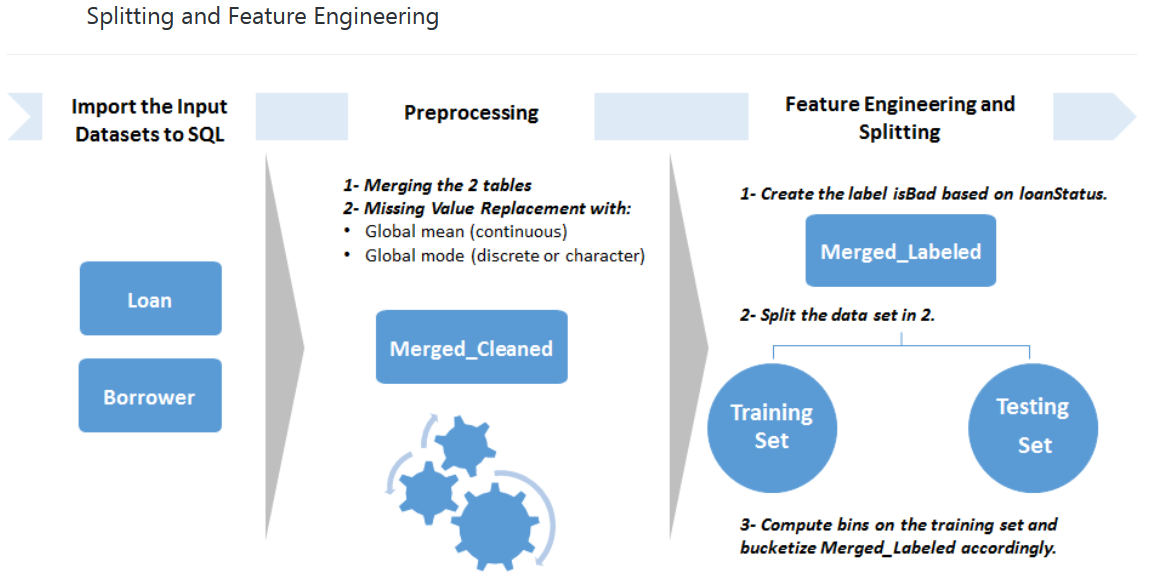
* + - * Working directories on the edge node and HDFS.
      * 2 Data Tables: Loan and Borrower (paths to csv files)

Output:

* + - * The statistics summary information saved to the local edge node in the LocalModelsDir folder.
      * Cleaned raw data set MergedCleaned on HDFS in xdf format.

The above are all performed in step1\_preprocessing.R file.

# Splitting and Feature Engineering Results



For feature engineering, new features are designed:

* Categorical versions of all the numeric variables. This is done for interpretability and is a standard practice in the Credit Score industry.
* isBad: the label, specifying whether the loan has been charged off or has defaulted (isBad = 1) or if it is in good standing (isBad = 0), based on loanStatus.

This is done by following these steps:

1. The label isBad is created with rxDataStep function. Outputs are written to the HDFS directory “MergedLabeled”. At the same time the variable hashCode are created with values corresponding to hashing loanId to integers. It is used for splitting. This hashing function ensures repeatability of the splitting procedure.
2. The data set is split into a training and a testing set. This is done by selecting randomly a proportion (equal to the user-specified splitting ratio) of the MergedLabeled data. The output is written to the “Train” directory.
3. The bins are computed that will be used to create the categorical variables with smbinning. Because some of the numeric variables have too few unique values, or because the binning function did not return significant splits, it is decided to manually specify default bins in case smbinning does not return the splits. These default bins have been determined through an analysis of the data or through running smbinning on a larger data set. Those cutoffs are saved in the directory LocalModelsDir, to be used for the Production or Web Scoring stages.

The bins computation is optimized by running smbinning in parallel across the different cores of the server, through the use of rxExec function applied in a Local Parallel (localpar) compute context. The rxElemArg argument it takes, is used to specify the list of variables (here the numeric variables names) needed to apply smbinning on.

1. The variables are bucketized based on the computed/specified bins with the function bucketize, wrapped into an rxDataStep function. The final output is written into the HDFS directory **MergedFeatures**, in xdf format.

Finally, the newly created variables are converted from character to factors, and the variable information is saved in the directory LocalModelsDir, to be used for the Production or Web Scoring stages. The data with the correct variable types is written to the directory “MergedFeaturesFactors” on HDFS.

Input:

(assuming the cleaned data, MergedCleaned is already created there by the above previous step/section).

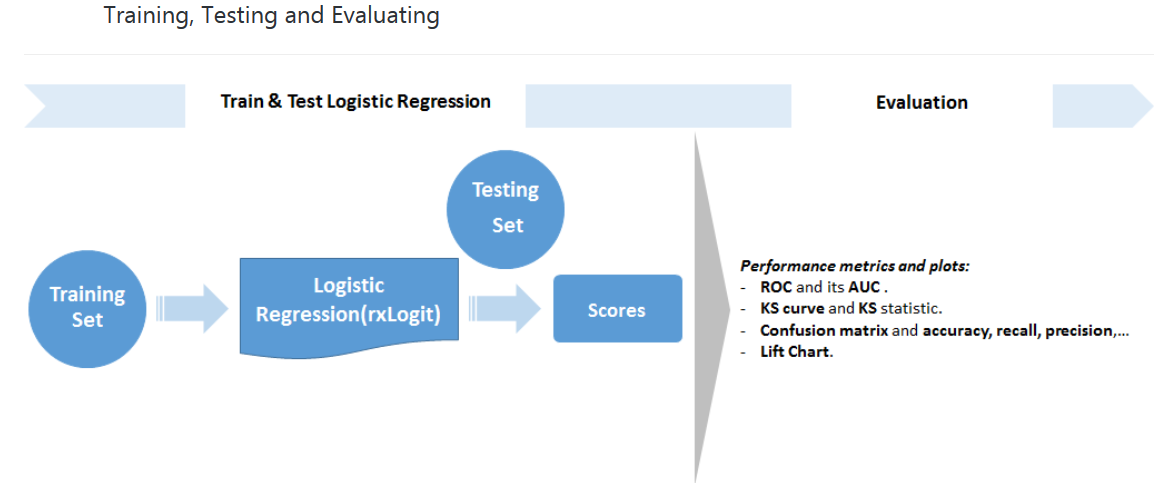
* + - * Working directories on the edge node and HDFS.
      * The splitting ratio, corresponding to the proportion of the input data set that will go to the training set.

Output:

* + - * Cutoffs saved to the local edge node in the LocalModelsDir folder.
      * Factor information saved to the local edge node in the LocalModelsDir folder.
      * Analytical data set with correct variable types MergedFeaturesFactors on HDFS in xdf format.

All the above are performed by step2\_feature\_engineering.R file.

# Training, Testing & Evaluation



Here, in Training, Testing & Evaluation, the following are performed:

The xdf files are split in MergedFeaturesFactors into a training set, “Train”, and a testing set “Test”. This is done through rxDataStep functions, according to the splitting ratio defined and used in above-mentioned step/section (initial data exploration) and using the same hashCode created in above-mentioned step/section (initial data exploration).

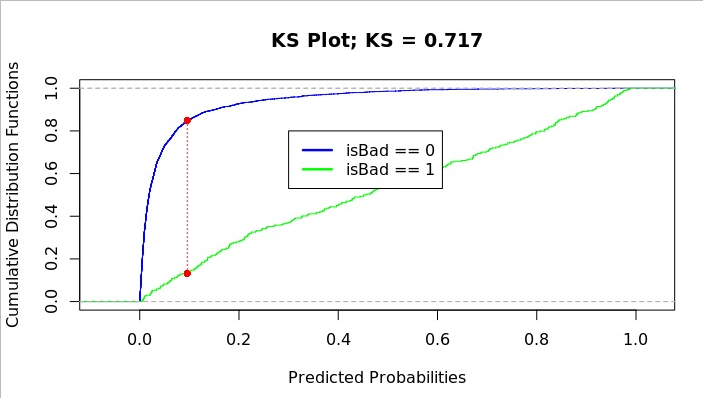
A Logistic Regression is trained on Train, and it is saved on the local edge node in the LocalModelsDir folder. It would be used in the Production or Web Scoring stages later.

Training a Logistic Regression for loan credit risk prediction is a standard practice in the Credit Score industry. Contrary to more complex models such as random forests or neural networks, it is easily understandable through the simple formula generated during the training. Also, the presence of bucketed numeric variables helped to understand the impact of each category and variable on the probability of default. The variables used and their respective coefficients, sorted by order of magnitude, are stored in the data frame Logistic\_Coeff returned by the step 3 function.

Finally, predictions on the testing set are computed, as well as performance metrics in the following:

# KS Plot

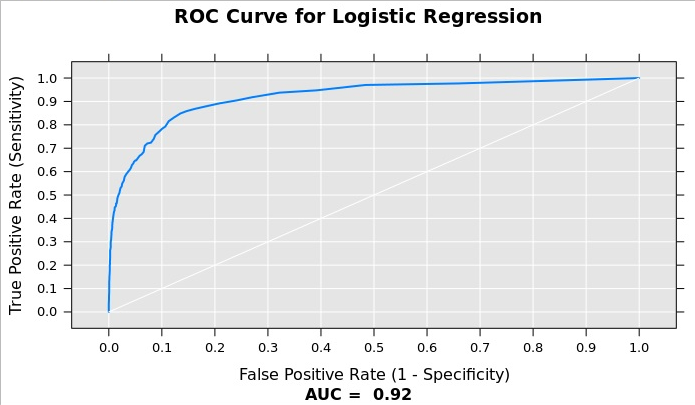
* **KS** (Kolmogorov-Smirnov) statistic. The KS statistic is a standard performance metric in the credit score industry. It represents how well the model can differenciate between the Good Credit applicants and the Bad Credit applicants in the testing set. The KS plot which corresponds to two cumulative distributions of the predicted probabilities are drawn. One is a subset of the predictions for which the observed values were bad loans (is\_bad = 1) and the other concerning good loans (is\_bad = 0). KS is the biggest distance between those two curves.



* Various classification performance metrics are computed on the confusion matrix. These are dependent on the threshold chosen to decide whether to classify a predicted probability as good or bad. Here, the point of the x axis in the KS plot where the curves are the farthest possible is used as a threshold.

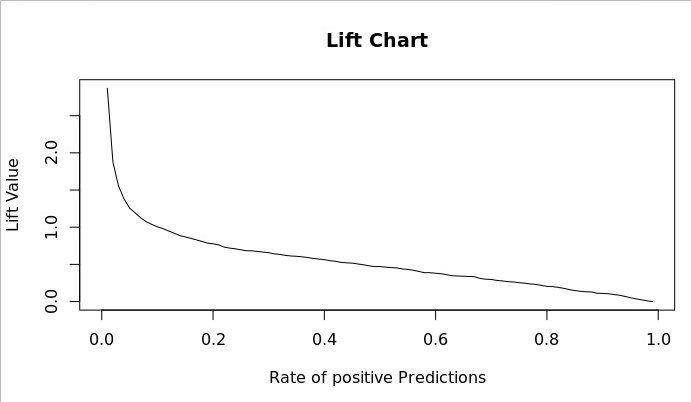
# ROC curve for Logistic Regression

* + - * **AUC** (Area Under the Curve) for the ROC. This represents how well the model can differenciate between the Good Credit applicants from the Bad Credit applicants given a good decision threshold in the testing set. The ROC, representing the true positive rate in function of the false positive rate for various possible cutoffs are drawn.



# Lift Chart

* + - * **The Lift Chart**. The lift chart represents how well the model can perform compared to a naive approach. For instance, at the level where a naive effort could produce a 10% rate of positive predictions, a vertical line on x = 0.10 is drawn and the lift value is read where the vertical line crosses the lift curve. If the lift value is 3, it means that the model would produce 3 times the 10%, ie. 30% rate of positive predictions.



Input:

(assuming the analytical data set, “MergedFeaturesFactors”, is already created there by the above previous step/section (Splitting and Feature Engineering).)

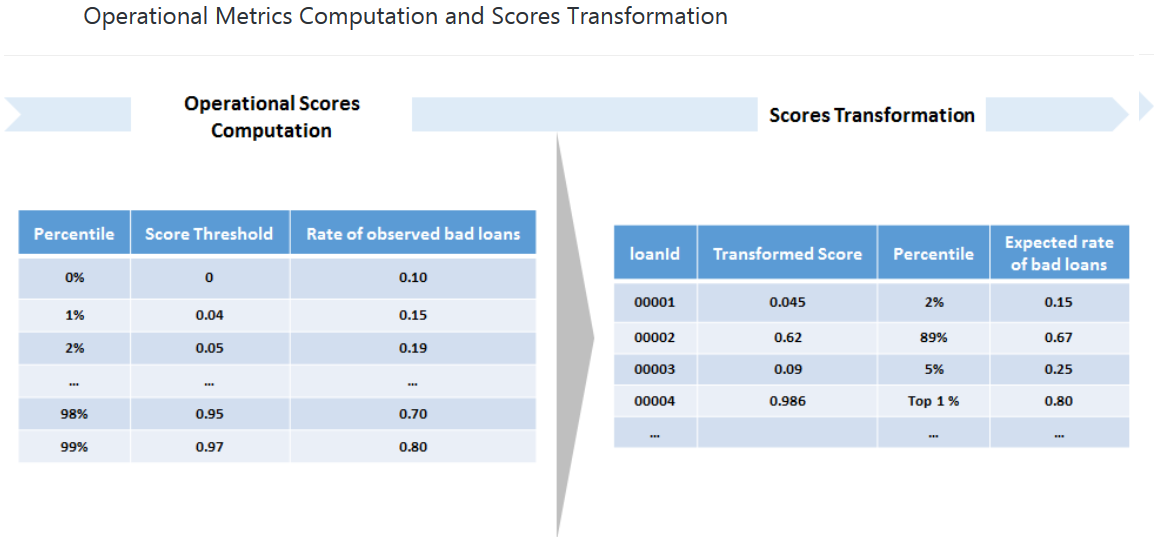
* + - * Working directories on the edge node and HDFS.
      * The splitting ratio, corresponding to the proportion of the input data set that will go to the training set. It should be the same as the one used in step/section (Splitting and Feature Engineering).

Output:

* + - * Logistic Regression model saved to the local edge node in the LocalModelsDir folder.
      * Logistic Regression formula saved to the local edge node in the LocalModelsDir folder.
      * Prediction results given by the model on the testing set, “PredictionsLogistic” on HDFS in xdf format.

All the above are performed by step3\_train\_score\_evaluate.R file.

# Operational Metrics Computation & Scores Transformation



Here, two functions compute\_operational\_metrics, and apply\_score\_transformation are created.

The first, compute\_operational\_metrics will:

1. Apply a sigmoid function to the output scores of the logistic regression, in order to spread them in [0,1] and make them more interpretable. This sigmoid uses the average predicted score, which is saved to SQL in case you want to run a production stage through SQL after a development stage with R.
2. Compute bins for the scores, based on quantiles (we compute the 1%-99% percentiles).
3. Take each lower bound of each bin as a decision threshold for default loan classification, and compute the rate of bad loans among loans with a score higher than the threshold.

It outputs the data frame Operational\_Metrics, which is saved to the local edge node in the LocalModelsDir folder, for use in the Production and Web Scoring stages.

The second, apply\_score\_transformation will:

1. Apply the same sigmoid function to the output scores of the logistic regression, in order to spread them in [0,1].
2. Asssign each score to a percentile bin with the bad rates given by the Operational\_Metrics table.

Input:

(assuming the predictions on the testing set, PredictionsLogistic, are already created there by the above previous step/section (Training, Testing and Evaluating).)

* + - * Working directories on the edge node and HDFS.

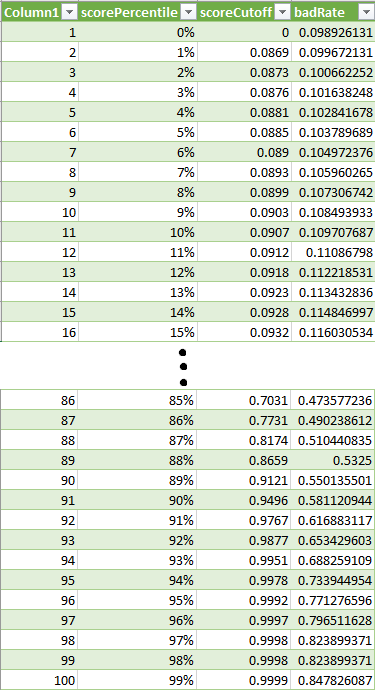
Output:

* + - * Average of the predicted scores on the testing set of the Development stage, saved to the local edge node in the LocalModelsDir folder.
      * Operational Metrics saved to the local edge node in the LocalModelsDir folder.
      * Scores on HDFS in xdf format. It contains the transformed scores for each record of the testing set, together with the percentiles they belong to, the corresponding score cutoff, and the observed bad rate among loans with a higher score than this cutoff.
      * ScoresData on HDFS in Hive format for visualizations in PowerBI.

All the above are performed by step4\_operational\_metrics.R file.

The **modeling\_main.R** script uses the Operational\_Metrics table to plot the rates of bad loans among those with scores higher than each decision threshold. The decision thresholds correspond to the beginning of each percentile-based bin.

Below picture shows the Operational\_Metrics table:



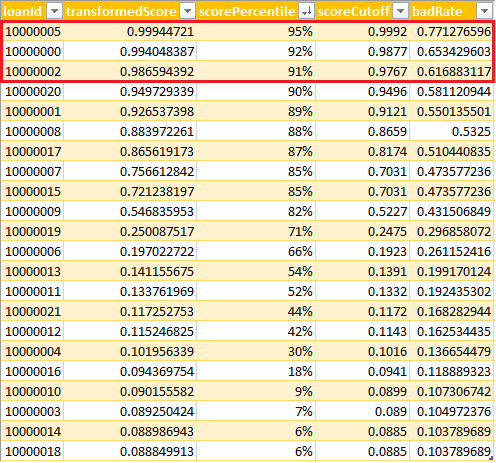
For example, if the score cutoff of the 91th score percentile is 0.9767, and a bad rate of 0.6169 is read. This means that if 0.9767 is used as a threshold to classify loans as bad, a bad rate of 61.69% is resulted. This bad rate is equal to the number of observed bad loans over the total number of loans with a score greater than the threshold.

# Conclusion

In conclusion, Microsoft R Server on HDInsight Spark clusters provides distributed and scalable machine learning capabilities for big data, leveraging the combined power of R Server and Apache Spark. This analysis demonstrates how to develop machine learning models for loan credit risk (including data processing, feature engineering, training and evaluating models), deploys the models as a web service (on the edge node) and consumes the web service remotely with Microsoft R Server on Azure HDInsight Spark clusters.

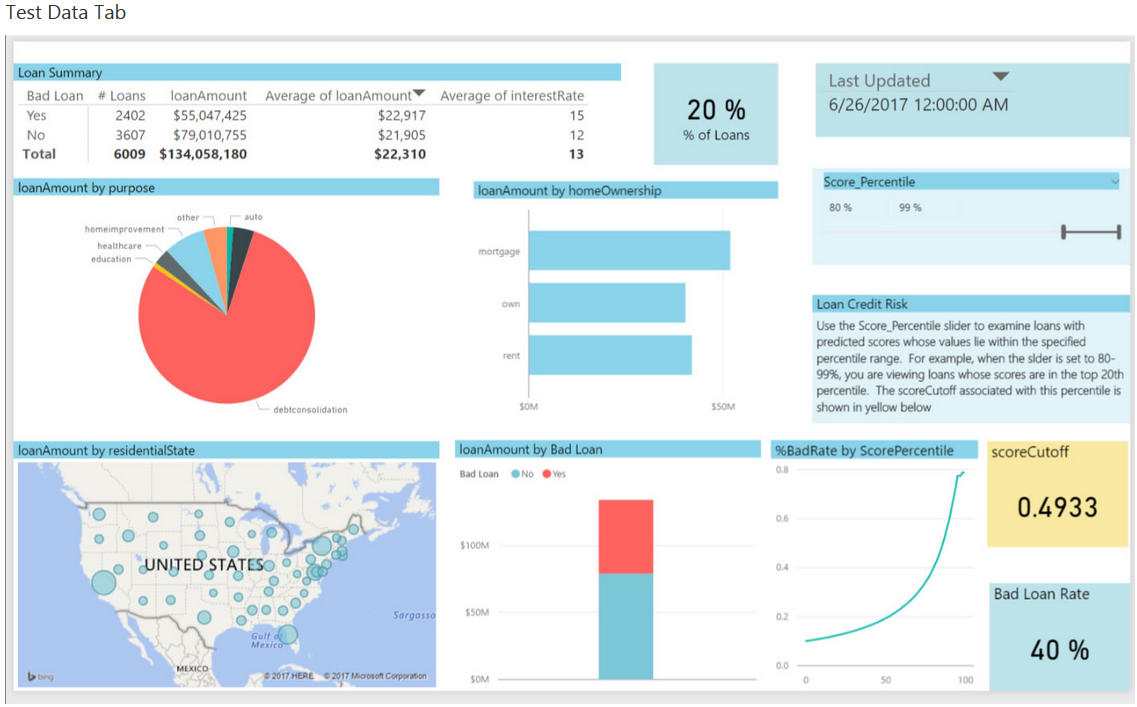
Hive tables are saved containing the predicted scores during both development and production.

In this analysis, the below Scores table shows the scenario that if the score cutoff is at the 91th score percentile. This means 3 out of the 22 potential loans will be rejected based on this criteria (with a bad rate of 61.69%). This bad rate is equal to the number of observed bad loans over the total number of loans with a score greater than the threshold.



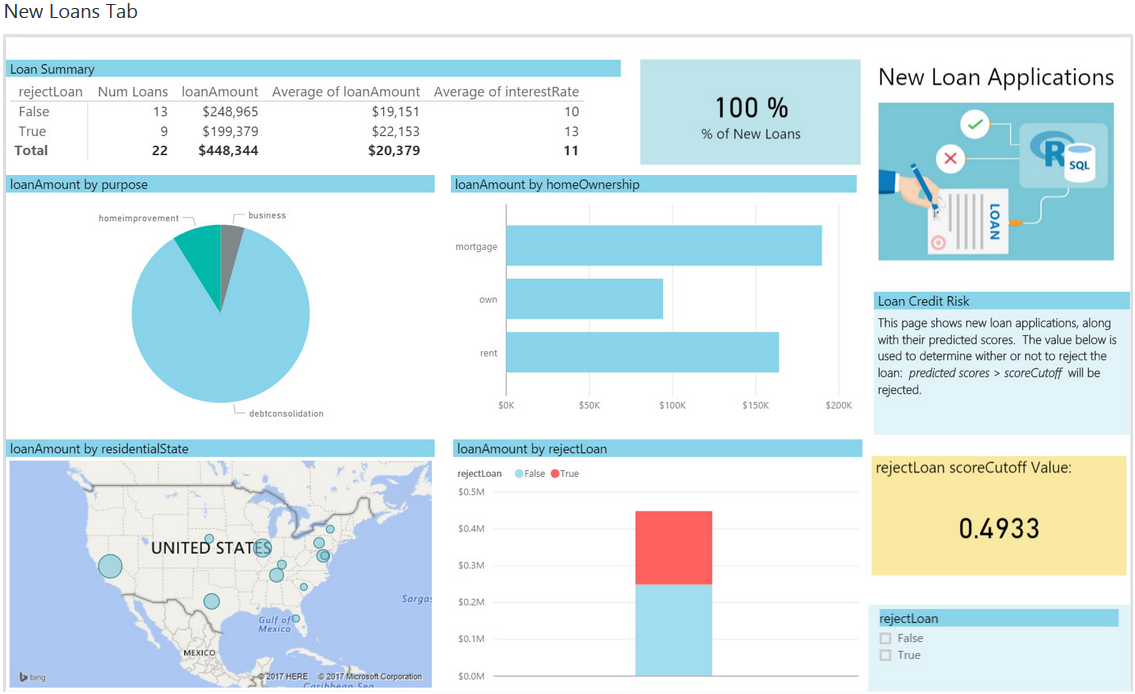
Moving forward, as an example, this data can then be visualized in Power BI.

For example, the PowerBI dashboard will allow the visualization and use these predicted scores to aid in deciding when to approve a loan. There can be two different tabs: the Test Data tab will allow exploration of the scores in the test data in order to decide on a cutoff value to use in the decision to reject a loan. The Prod Data tab will show new potential loans in the production pipeline where the results can be viewed using this cutoff value.



In the example, the output scores from the model have been binned according to percentiles: the higher the percentile, the more likely the risk of default. On the Test Data tab, the slider (at the top right) can be used to examine loans in the test data that correspond to these percentiles; for example, to find a suitable level of risk for extending a loan. The slider is set to show the top 20% of scores (Score Percentile from 80-99%). The box in yellow can be used to show the corresponding cutpoint that can be used to classify these loans as bad, new loans with predicted scores higher than this number will be rejected. The default value of .4933 (corresponding to the top 20% in the training data) is used in the Loan Summary tab for this example.

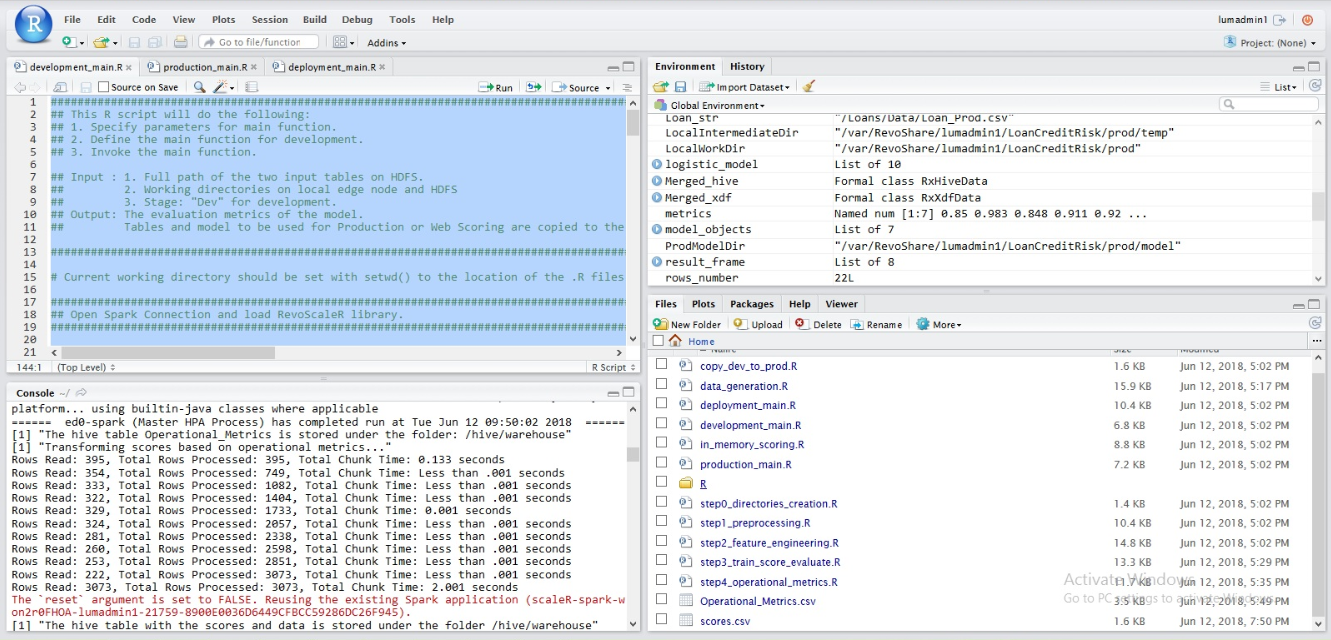
The Loan Summary table divides those loans classified as bad in two: those that were indeed bad (Bad Loan = Yes) and those that were in fact good although they were classified as bad (Bad Loan = No). For each of these 2 categories, the table shows the number, total and average amount, and the average interest rate of the loans. This allows the expected impact of choosing this cutoff value to be seen.



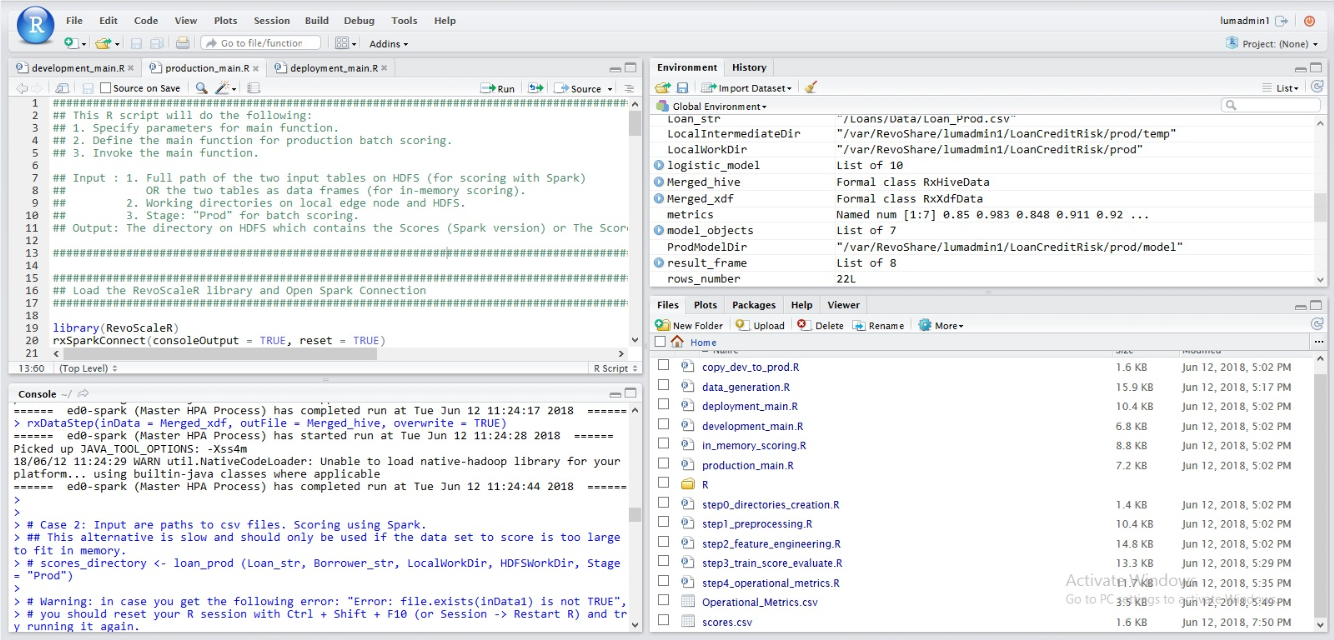
Continuing this example, on the New Loans tab, some scored potential loans can be seen. This example is using 0.4933 as the cutoff value. 9 of the 22 potential loans will be rejected based on this criteria. (With PowerBI Desktop, this cutoff can be changed to a different value.)

# Appendix

Below picture shows the processing of development\_main.R:



Below picture shows the processing of production\_main.R:



Below picture shows the processing of deployment\_main.R:

