ReneWind (Case Study)

Project 6: Model Tuning

Date: February 5th, 2022.

By: Ijeoma Ejem

Contents / Agenda

EXECUTIVE SUMMARY

Executive Summary

	Conclusion	Recommendation
1	Ciphered attributes make it difficult to explore patterns in the data that could lead to stronger insights and solutions (preventive measures).	Partnering with a wind turbine manufacturer in some fashion to gain access to some ciphered attributes will benefit the research process and could lead to further reduced maintenance costs and smoother production processes.
2	Some overfitting was identified across most of our evaluated models. This could lead to higher inspection costs despite reduced replacement or repair costs.	These instances should be weighed out by looking further into the numbers. E.g. How much does a machine inspection, repair and replacement cost per visit? How could the number of positive and negative predictions affect overall maintenance costs long-term based on different model predictions?
3	The undersampled XGBoost Classifier model produced the best recall score (~0.887) on the test data. False negatives are expected to be minimized by 88.7%.	This model should be re-evaluated and re-tested over time based on newly generated data following productionisation of the current model.

PROBLEM AND SOLUTION OVERVIEW

Business Problem and Solution Overview

Problem:

- As the effort to reduce the environmental impact of energy production increases, renewable energy sources play an increasingly important role. The U.S Department of Energy has put together a guide to achieving operational efficiency using predictive maintenance practices.
- ReneWind is working on improving the machinery/processes involved in the production of wind energy using machine learning and has collected data of generator failure of wind turbines using sensors.

Solution:

- Using exploratory data analysis (EDA) to explore significant factors that influence failure patterns and component failure, if possible.
- Building and testing various classification models to find the best one that will identify failures early so that generators can be repaired before failing/breaking, which will reduce overall maintenance costs.
- Analyzing results from the data and providing business insights and recommendations to help ReneWind accomplish its mission of achieving operational efficiency.

DATA OVERVIEW

Data Overview

- There are 20,000 rows (observations) and 41 columns (attributes) in the training data.
- Our test data consists of 5,000 rows (observations) and 41 columns (attributes).
- All datatypes are accurately represented; there are 40 float values and 1 integer (target) in the dataset.
- Attributes in the data have been ciphered for confidentiality so analysis and insights from data are limited.
- From the statistical summary of our numerical data, the following is deduced;
  - All attributes have a count of 20,000 with the exception of V1 and V2.
  - More attributes have negative average and median values than positive.
  - The attribute with the largest standard deviation is V32 (5.500).
  - Based on the minimum values, maximum values and everything in between, the range of values across all attributes is somewhere between -20.374 and 23.633.
- There are 36 missing values in training data and 11 missing values in test data.
- There are 0 duplicate values in the data.

DATA PREPROCESSING

Data Preprocessing Overview

Outlier Detection and Treatment:

From the EDA, we identified many outliers in the data with both negative and positive values. However, due to the attributes being ciphered we can not verify the accuracy of these values, and as a result, we are unable to treat the outliers.

Data Preparation for Modeling:

Since there are some missing values, we used an imputer function to replace them with the median values of the respective attributes. We successfully transformed the train, validation and test sets without data leakage.

Our test data has already been separated from the training data so there's no need to split the data into train, validation and test sets. Instead, we split only our training data into train and validation set with a validation set size of 25%. After the above steps, there were 15,000 observations in the training set and 5,000 observations in the validation set.

Lastly, we defined a function to output different metrics and defined our scorer (recall) to be used for cross-validation and hyperparameter tuning.

MODEL BUILDING

Model Building Criterion

Model can make wrong predictions as:

1. Model predicts that a machine will fail but in reality, the machine will not fail.
2. Model predicts that a machine will not fail but in reality, the machine will fail.

Which case is more important?

The second prediction is more important because:

- If a model predicts that a machine will have no failure when there will be a failure, it could increase maintenance costs.
- The cost of repairing a machine is much less than the cost of replacing it, and the cost of inspection is less than the cost of repair.

How to reduce the losses?

- We need to choose the metric which will ensure that the maximum number of machine failures are predicted correctly by the model.
- Recall metric will ensure higher chances of minimizing false negatives that increase maintenance costs.

Confusion Matrix Overview

True Positives (TP): The machine should fail and the model predicted that it will fail.

True Negatives (TN): The machine should not fail and the model predicted that it will not fail.

False Positives (FP): The machine should not fail and the model predicted that it will fail.

False Negatives (FN): The machine should fail and the model predicted that it will not fail.

Cross-Validation & Validation Assessment Overview

Validation performance will provide an estimate of our model's performance on the validation set. We use validation performance when fine-tuning the hyperparameters of our models to get the best possible performance on our dataset.

Cross-validation uses multiple iterations to train and validate our model on different subsets of the data. This will provide a more robust assessment of our model’s performance and an estimate of the model's ability to generalize to new unseen data.

For this reason, a good cross-validation performance is preferred over a good validation performance, as the former indicates better generalization ability. However, both are still very important to evaluate so we assessed the cross-validation performance on the training set and validation performance on the validation set across all models.

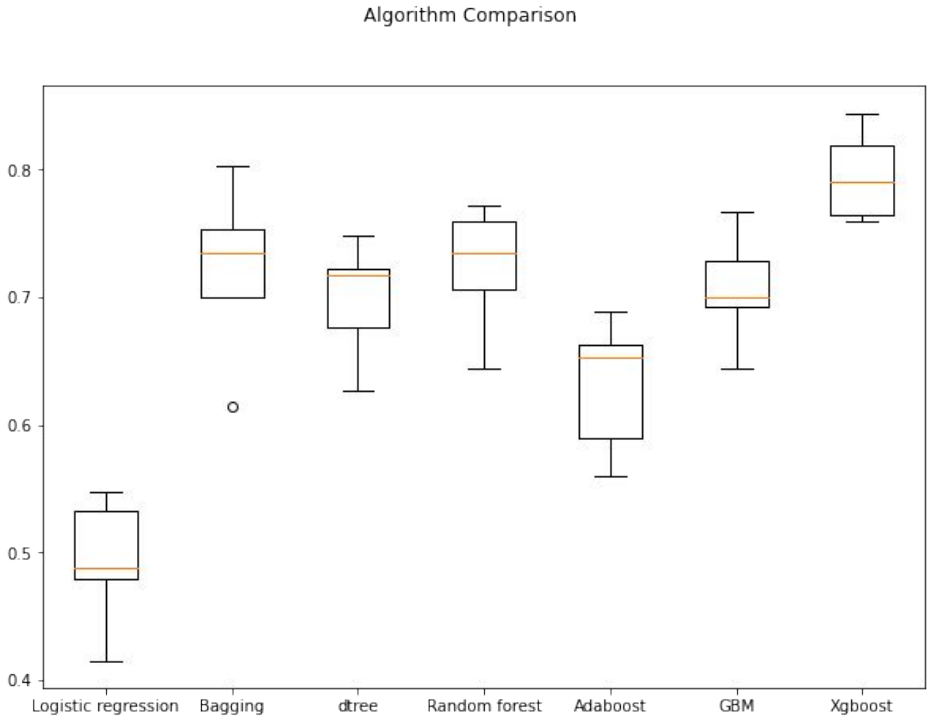
Model Building (Original Data)

Cross-Validation Performance:

Logistic regression: 0.4927
Bagging: 0.7210
dtree: 0.6982
Random forest: 0.7235
Adaboost: 0.6309
GBM: 0.7066
Xgboost: 0.7956

Validation Performance:

Logistic regression: 0.4820
Bagging: 0.7302
dtree: 0.7050
Random forest: 0.7266
Adaboost: 0.6762
GBM: 0.7230
Xgboost: 0.8201



A boxplot of the CV scores of all models listed

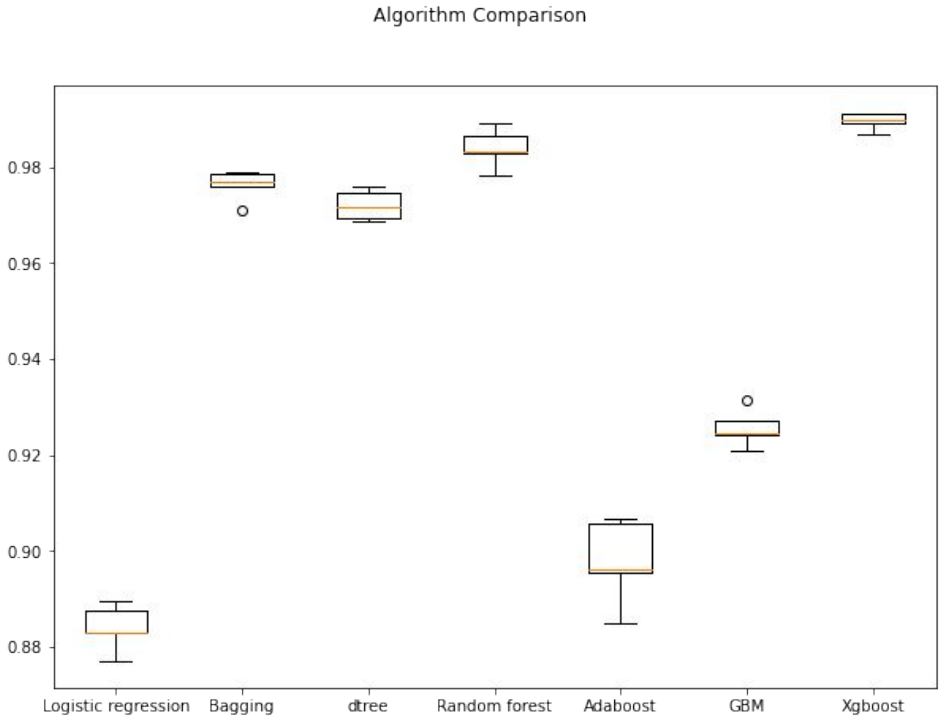
Model Building (Oversampled Data)

Cross-Validation Performance:

Logistic regression: 0.8839
Bagging: 0.9762
dtree: 0.9720
Random forest: 0.9839
Adaboost: 0.8978
GBM: 0.9256
Xgboost: 0.9895

Validation Performance:

Logistic regression: 0.8489
Bagging: 0.8345
dtree: 0.7769
Random forest: 0.8489
Adaboost: 0.8561
GBM: 0.8776
Xgboost: 0.8669



Boxplot of CV scores from oversampled models

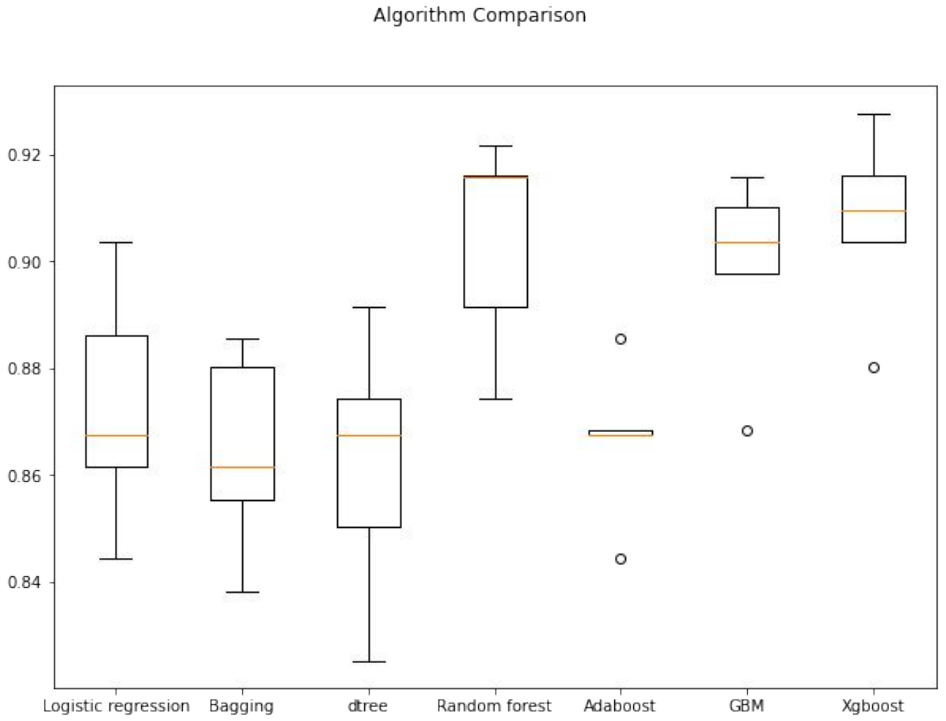
Model Building (Undersampled Data)

Cross-Validation Performance:

Logistic regression: 0.8726
Bagging: 0.8641
dtree: 0.8617
Random forest: 0.9038
Adaboost: 0.8666
GBM: 0.8990
Xgboost: 0.9074

Validation Performance:

Logistic regression: 0.8525
Bagging: 0.8705
dtree: 0.8417
Random forest: 0.8920
Adaboost: 0.8489
GBM: 0.8884
Xgboost: 0.9028



Boxplot of CV scores from undersampled models

Model Performance Evaluation

ORIGINAL DATA:

We compared the models built on original data with those built using oversampling or undersampling technique. While, the performance scores of our selected models are doing poorly on original data, it is useful to see how the use of both techniques significantly improves model performance.

OVERSAMPLED DATA:

When oversampling was applied, it produced better cross-validation performances on the training set but worse validation performances on the validation set than the undersampled data. Logistic Regression model appears to be the poorest performing model among the rest in cross-validation.

UNDERSAMPLED DATA:

Our decision tree model proved to be the weakest performing model in the undersampled data across both cross-validation and validation performance.

Process Summary & Conclusions

After evaluating the models built on original data and those built with oversampling or undersampling technique, we further tested our logistic regression and decision tree models on their strongest performing technique.

Due to the performance differences between cross-validation and validation tests across the oversampled and undersampled models, we used random search CV to find the best parameters for the respective models and tuned all models on both techniques. After which, we compared oversampling and undersampling recall scores of each model and selected the best performing sampling technique to evaluate against other models.i.e. Undersampling XGBoost and evaluating it against Oversampled Bagging Classifier and other models.

The best performing technique on our selected models are as follows;
Bagging Classifier (Oversampled), Decision Tree (Undersampled), Random Forest (Undersampled), AdaBoost Classifier (Undersampled), Gradient Boost Classifier (Undersampled), and XGBoost Classifier (Undersampled).

HYPERPARAMETER TUNING

Bagging Classifier Tuned with Oversampled Data

Observation:

The oversampled bagging classifier model was evaluated against the best performing models and was found to not perform as well as other models. While the recall scores here are higher than any other metric in this validation set, the extreme low precision and F1 score indicates overfitting in the data. While, a high recall score will cut repair and replacement costs, any significant reduction in the ability of the model to capture enough precision will simultaneously increase inspection costs significantly.

Training Set:

	Accuracy	Recall	Precision	F1
0	0.851	0.912	0.813	0.860

Validation Set:

	Accuracy	Recall	Precision	F1
0	0.782	0.871	0.187	0.308

Decision Tree Tuned with Undersampled Data

Observation:

Our undersampled decision tree model proved to be one of the best performing models with a recall score of (~0.888) on validation set. The recall score here is also the best performing metric in both training and validation set. It is important for us to the mindful of overfitting in the data as precision and F1 scores here are significantly low even when compared to our oversampled bagging classifier model.

Training Set:

	Accuracy	Recall	Precision	F1
0	0.764	0.909	0.705	0.794

Validation Set:

	Accuracy	Recall	Precision	F1
0	0.609	0.888	0.114	0.202

Random Forest Tuned with Oversampled Data

Observation:

The oversampled random forest produced the lowest recall score of our selected models. However, this is the model with the least overfitting we have produced so far. Since our priority is a high recall score this model appears to be the least favourable option in that regard.

Training Set:

	Accuracy	Recall	Precision	F1
0	1.000	0.999	1.000	1.000

Validation Set:

	Accuracy	Recall	Precision	F1
0	0.988	0.863	0.920	0.891

Random Forest Tuned with Undersampled Data

Observation:

We compared the oversampled random forest with the undersampled random forest performance results and found that despite the undersampled random forest having a higher recall score than the other, a difference of (~0.022), there is more overfitting evident in the undersampled performance results.
While, there is less overfitting in this model sample than other models we have evaluated so far, we still need to be mindful of how the overfitting can affect ReneWind's overall maintenance costs.

Training Set:

	Accuracy	Recall	Precision	F1
0	0.961	0.933	0.989	0.960

Validation Set:

	Accuracy	Recall	Precision	F1
0	0.938	0.885	0.468	0.612

AdaBoost Tuned with Undersampled Data

Observation:

The model built on the undersampled adaboost classifier produced a recall score of (~0.878) on validation set, which falls in the bottom 3 of recall performance score across our 7 selected models. We tested and evaluated stronger models than adaboost as we already expected models like xgboost to improve on the scores produced so far, based on our cross-validation and validation performance results.

Training Set:

	Accuracy	Recall	Precision	F1
0	0.950	0.916	0.982	0.948

Validation Set:

	Accuracy	Recall	Precision	F1
0	0.931	0.878	0.438	0.584

Gradient Boost Tuned with Undersampled Data

Observation:

This undersampled gradient boost model improved only slightly on the oversampled bagging classifier model across recall score and other metrics but it still performing poorly compared to better performing models like the undersampled decision tree and random forest. We have one last model to evaluate before final model selection.

Training Set:

	Accuracy	Recall	Precision	F1
0	0.995	0.992	0.998	0.995

Validation Set:

	Accuracy	Recall	Precision	F1
0	0.919	0.871	0.395	0.544

XGBoost Tuned with Undersampled Data

Observation:

Our final model, undersampled XGBoost Classifier, produced the highest recall score (~0.906) seen so far on the validation set. This model indicates some overfitting as the precision and F1 scores are significantly lower in validation than training.

Based on these results, we can now compare all models and make a selection based on the chosen metric and overall performance evaluation.

Training Set:

	Accuracy	Recall	Precision	F1
0	0.998	1.000	0.996	0.998

Validation Set:

	Accuracy	Recall	Precision	F1
0	0.879	0.906	0.303	0.454

PERFORMANCE EVALUATION & MODEL SELECTION

Model Performance Summary

Training:

	Bagging Classifier Tuned with Oversampled Data	Decision Tree Tuned with Undersampled Data	Random Forest Tuned with Oversampled Data	Random Forest Tuned with Undersampled Data	AdaBoost Classifier Tuned with Undersampled Data	Gradient Boosting Tuned with Undersampled Data	XGBoost Tuned with Undersampled Data
Accuracy	0.851	0.764	1.000	0.961	0.950	0.995	0.998
Recall	0.912	0.909	0.999	0.933	0.916	0.992	1.000
Precision	0.813	0.705	1.000	0.989	0.982	0.998	0.996
F1	0.860	0.794	1.000	0.960	0.948	0.995	0.998

Validation:

	Bagging Classifier Tuned with Oversampled Data	Decision Tree Tuned with Undersampled Data	Random Forest Tuned with Oversampled Data	Random Forest Tuned with Undersampled Data	AdaBoost Classifier Tuned with Undersampled Data	Gradient Boosting Tuned with Undersampled Data	XGBoost Tuned with Undersampled Data
Accuracy	0.782	0.609	0.988	0.938	0.931	0.919	0.879
Recall	0.871	0.888	0.863	0.885	0.878	0.871	0.906
Precision	0.187	0.114	0.920	0.468	0.438	0.395	0.303
F1	0.308	0.202	0.891	0.612	0.584	0.544	0.454

Final Model Selection

RESULT:

Our chosen metric is recall because we want to minimize false negatives, which will ensure that the maximum number of machine failures are predicted correctly by the model. The best recall score across all evaluated models was produced by the undersampled XGBoost Classifier model. We will chose this model despite some overfitting in the model.

Final Model Performance Evaluation

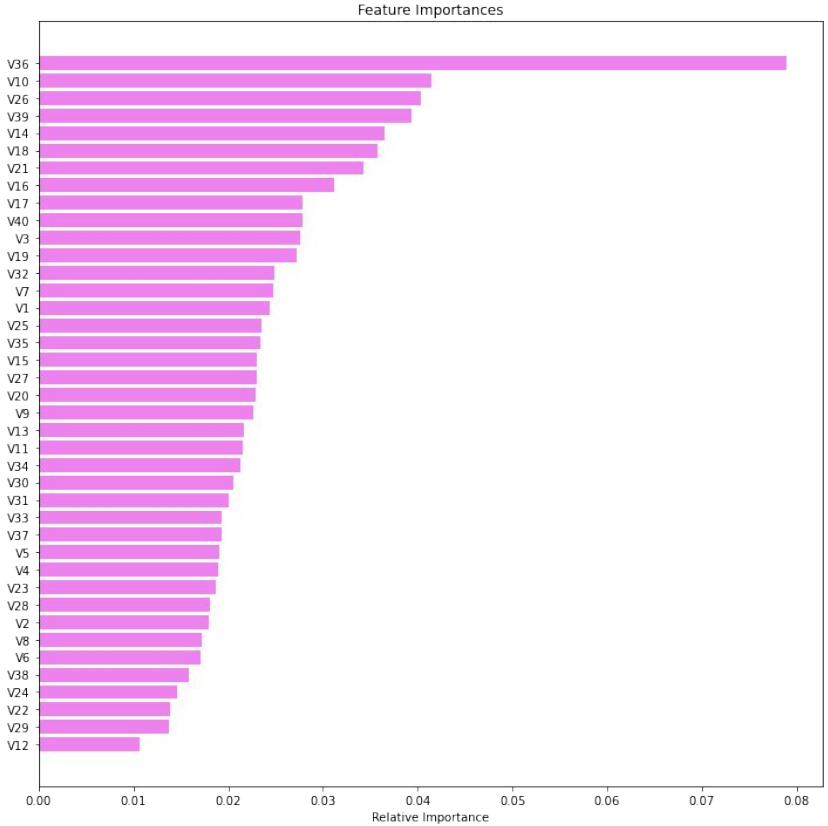
Undersampled XGBoost Classifier on Test Set:

With a recall score of (~0.887) our final model will minimize false negative predictions 88.7% of the time.

	Accuracy	Recall	Precision	F1
0	0.874	0.887	0.294	0.442

Feature Importance Evaluation

The most important feature in our final model is V36.



PRODUCTIONISATION OF FINAL MODEL

Pipeline Overview

Data Preparation:

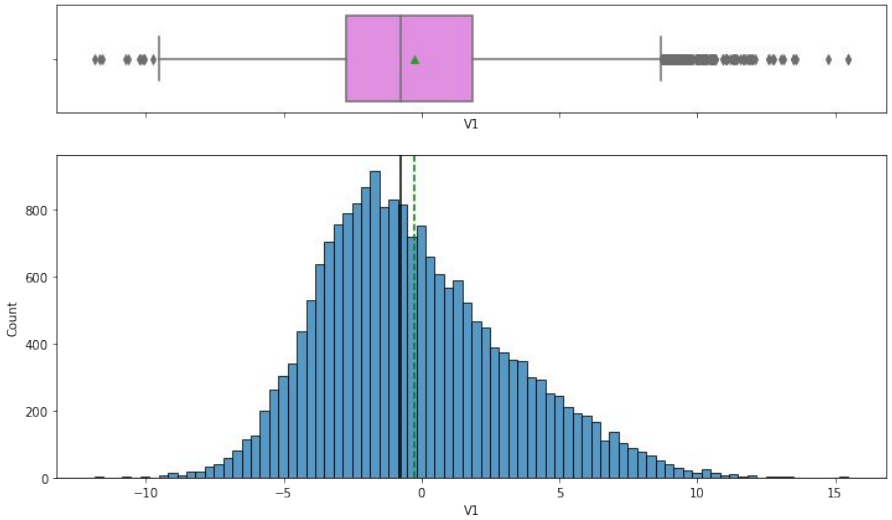
Since we only have one datatype in the data, we did not need to use column transformer. We also did not need to split the data so we simply built the pipeline with two simple steps;

1.) transform the data using median imputer (to replace missing values in the data with median values), and
2.) fit the undersampled XGBoost classifier model.

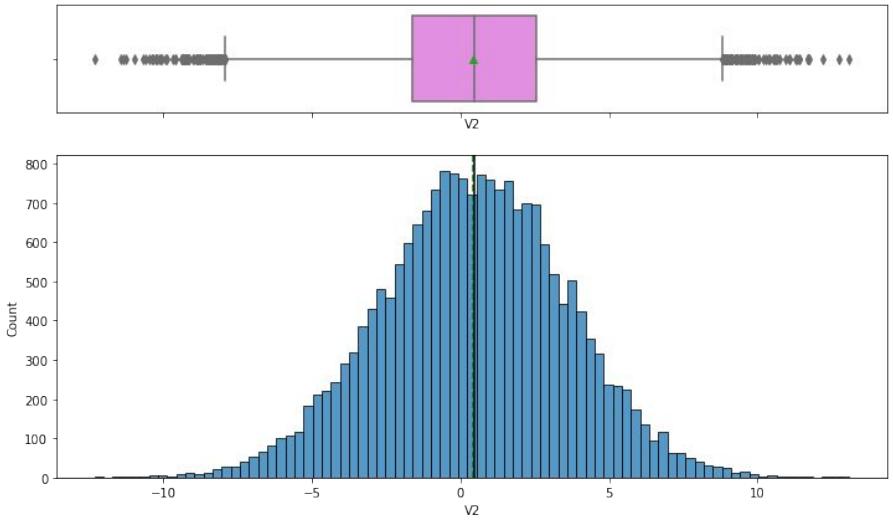
APPENDIX: EDA RESULTS

EDA Results

UNIVARIATE ANALYSIS



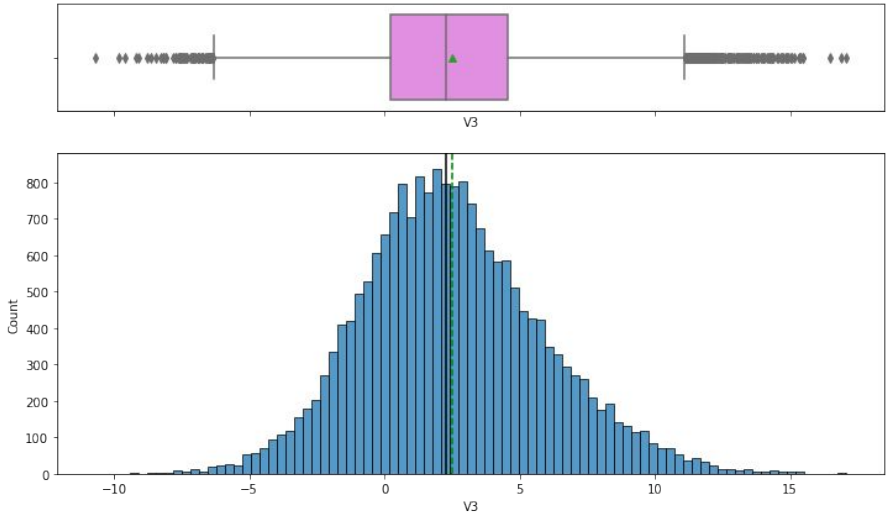
V1



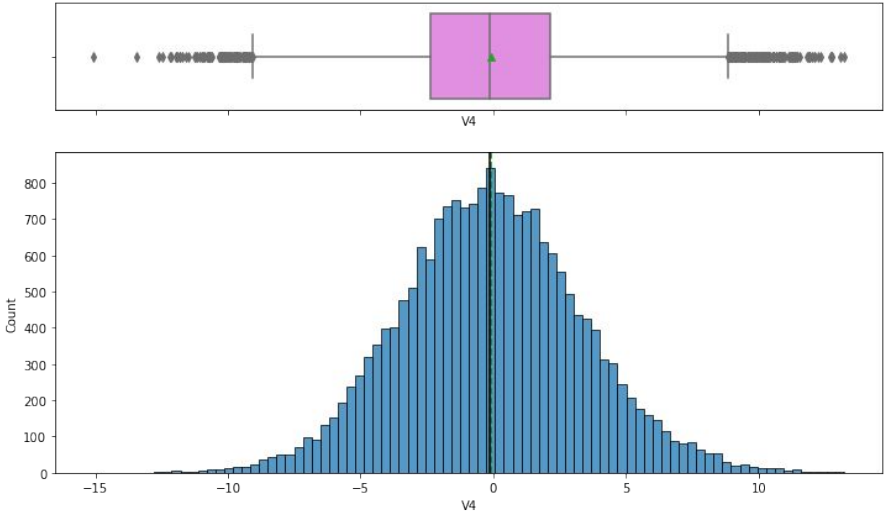
V2

EDA Results

UNIVARIATE ANALYSIS



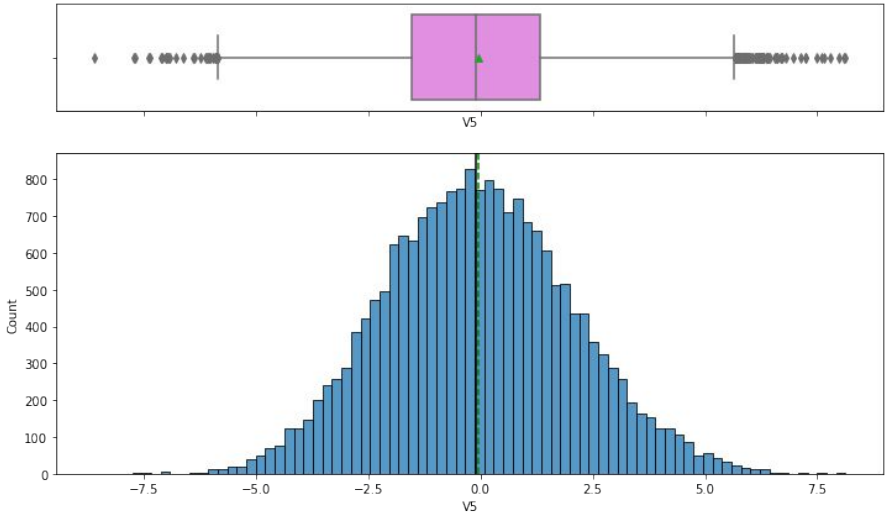
V3



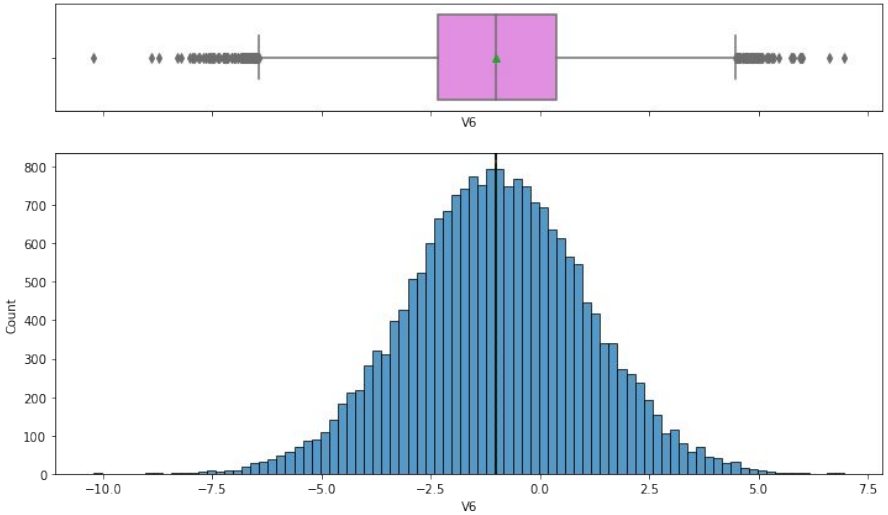
V4

EDA Results

UNIVARIATE ANALYSIS



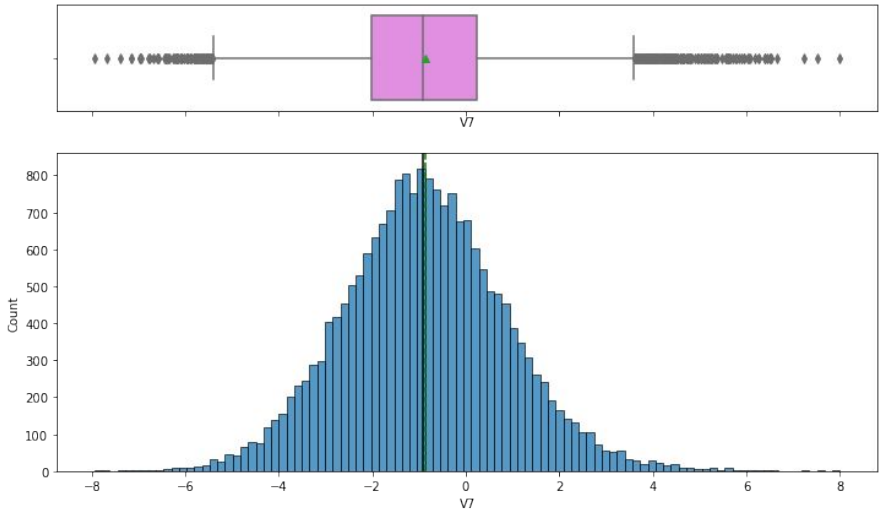
V5



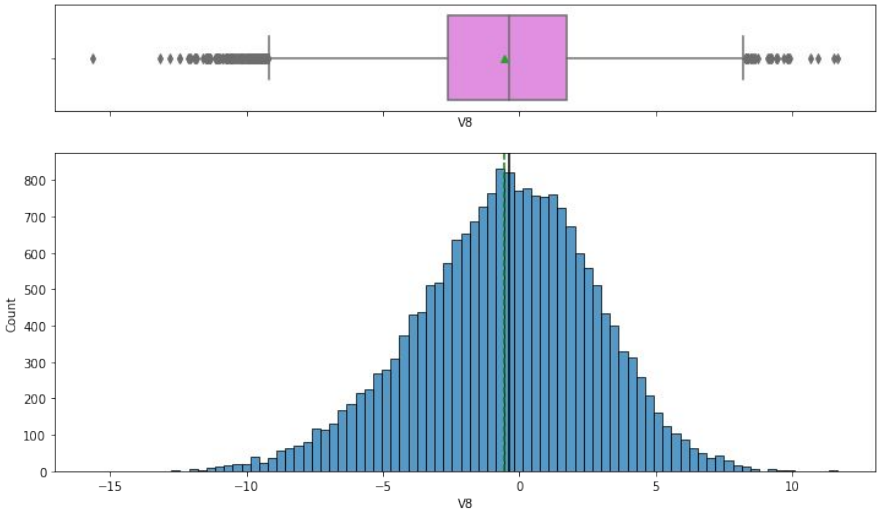
V6

EDA Results

UNIVARIATE ANALYSIS



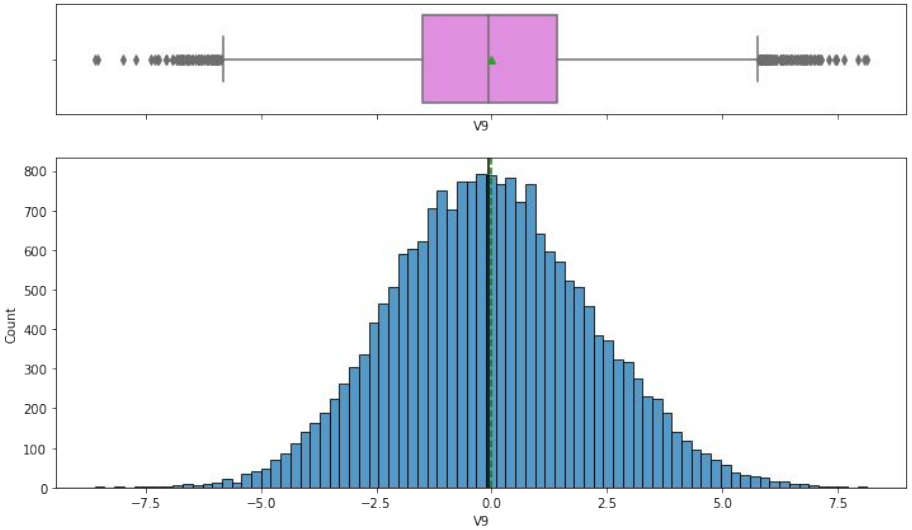
V7



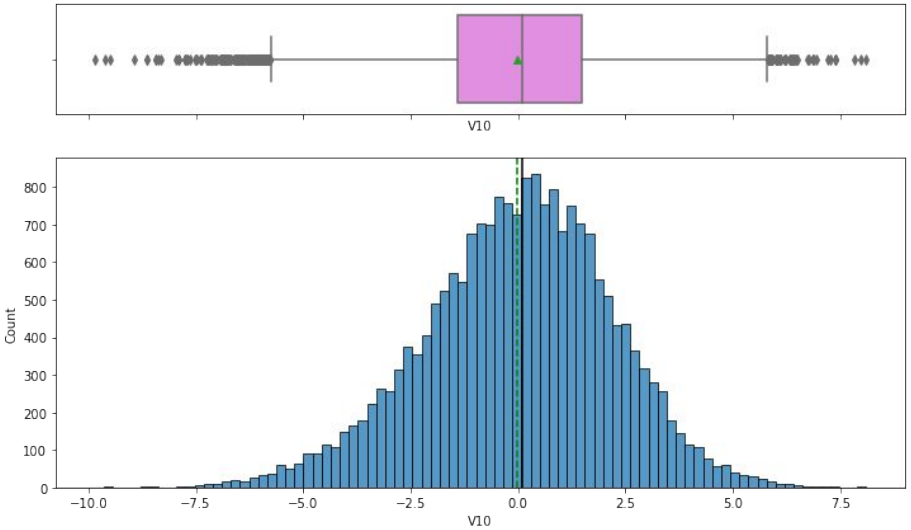
V8

EDA Results

UNIVARIATE ANALYSIS



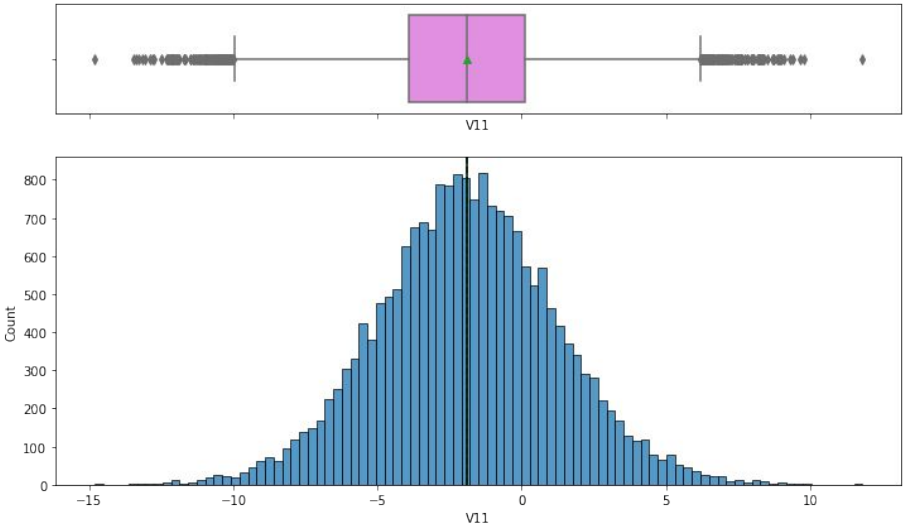
V9



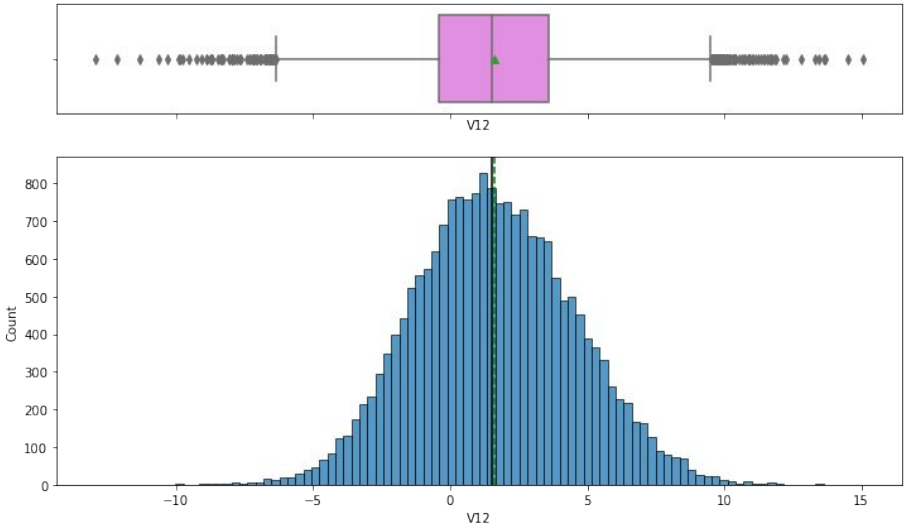
V10

EDA Results

UNIVARIATE ANALYSIS



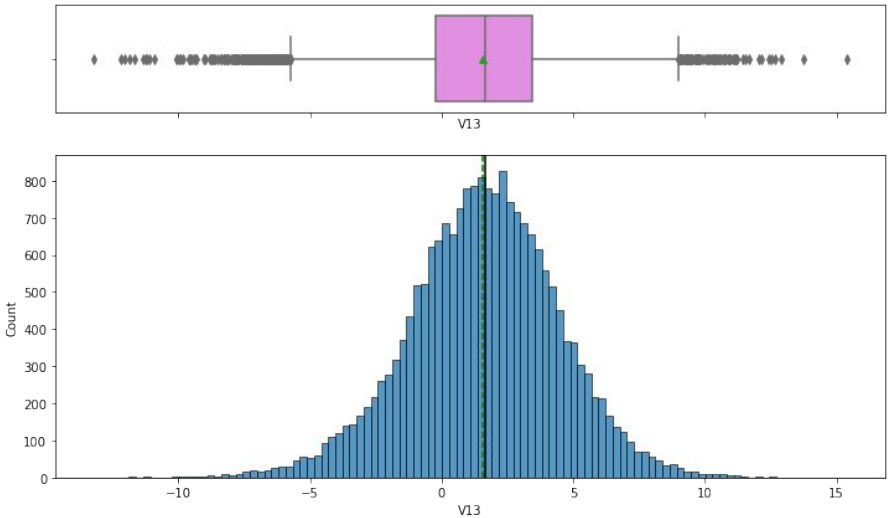
V11



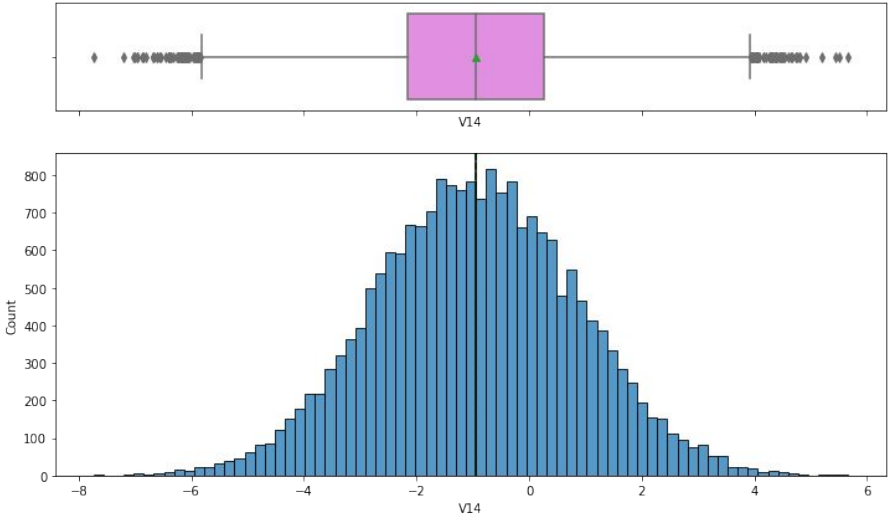
V12

EDA Results

UNIVARIATE ANALYSIS



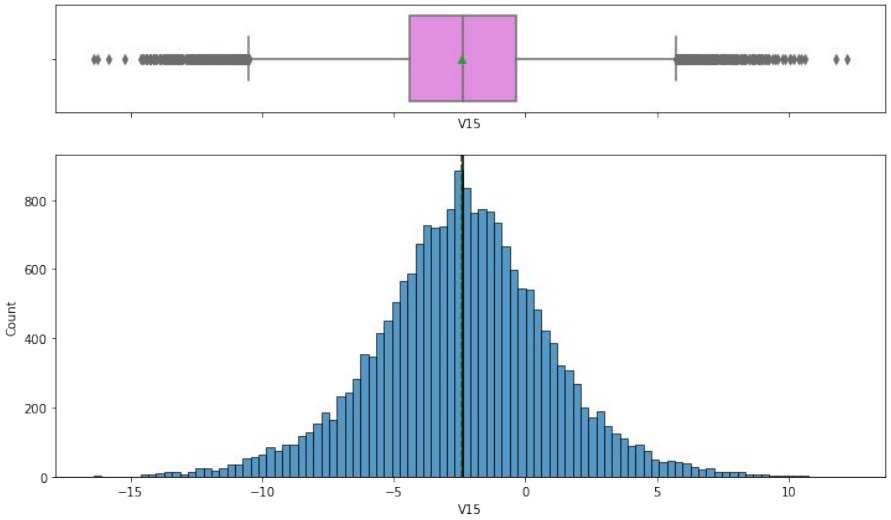
V13



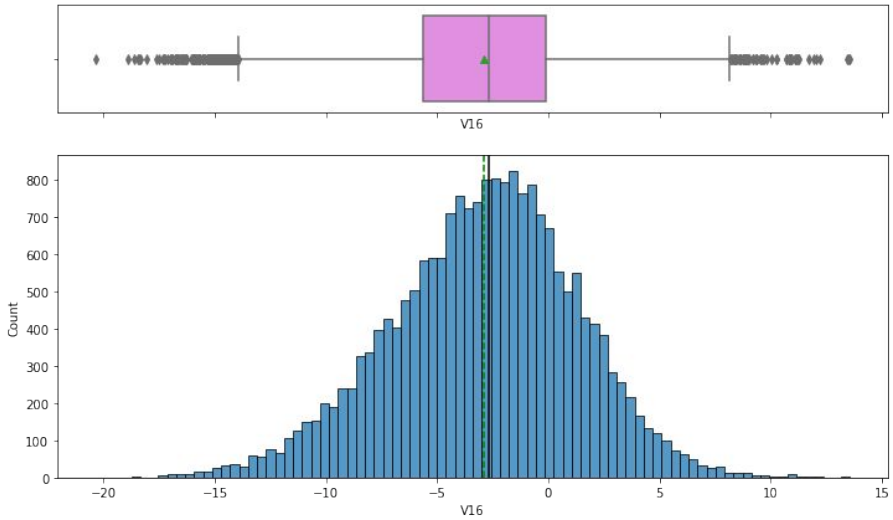
V14

EDA Results

UNIVARIATE ANALYSIS



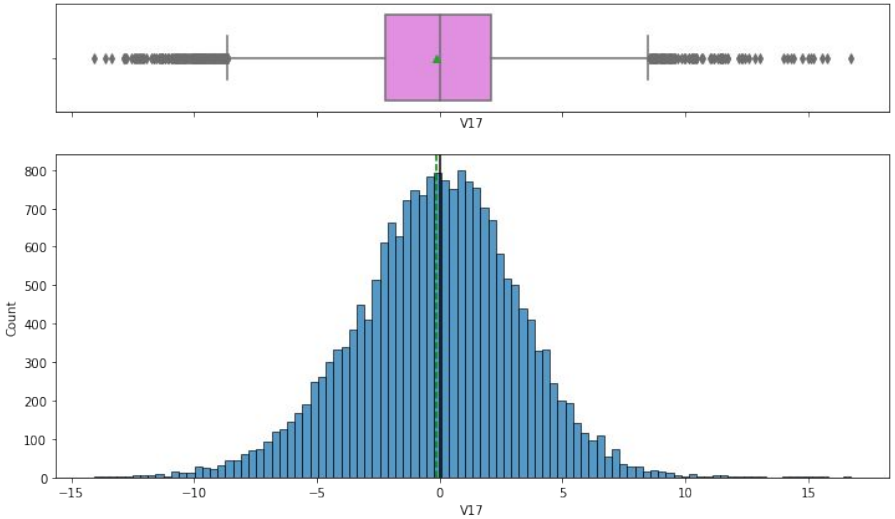
V15



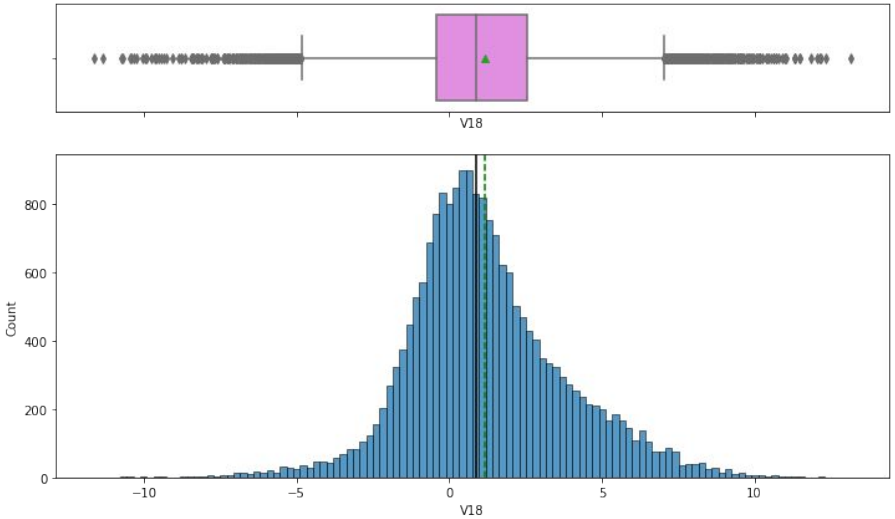
V16

EDA Results

UNIVARIATE ANALYSIS



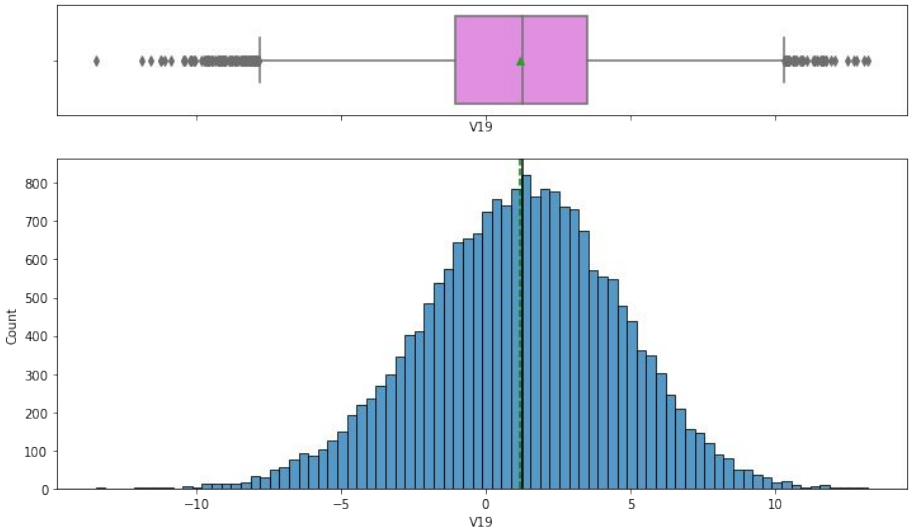
V17



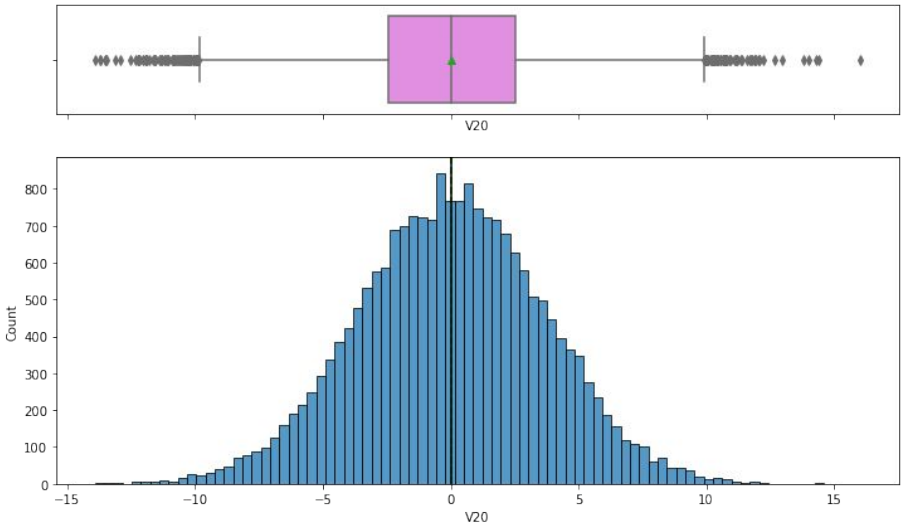
V18

EDA Results

UNIVARIATE ANALYSIS



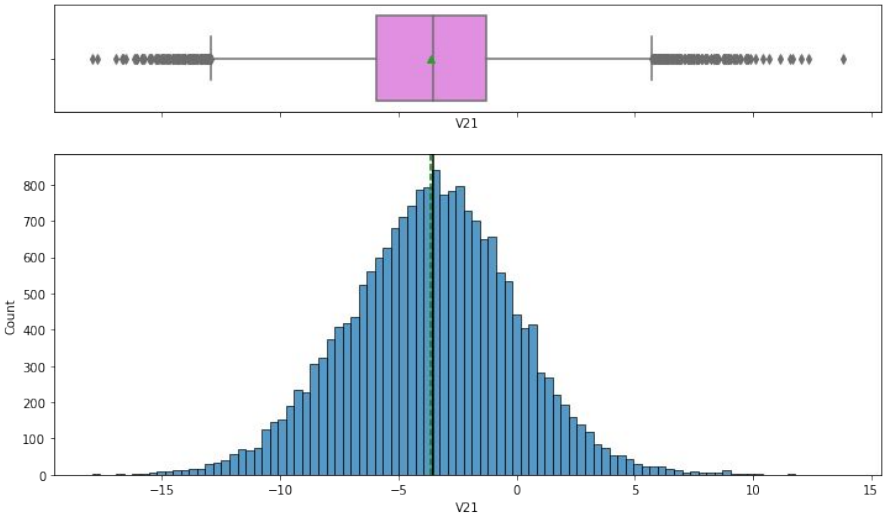
V19



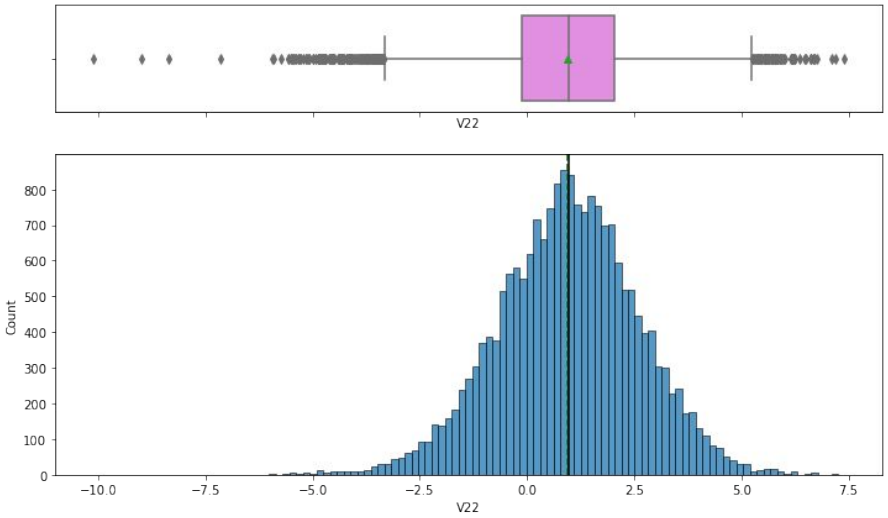
V20

EDA Results

UNIVARIATE ANALYSIS



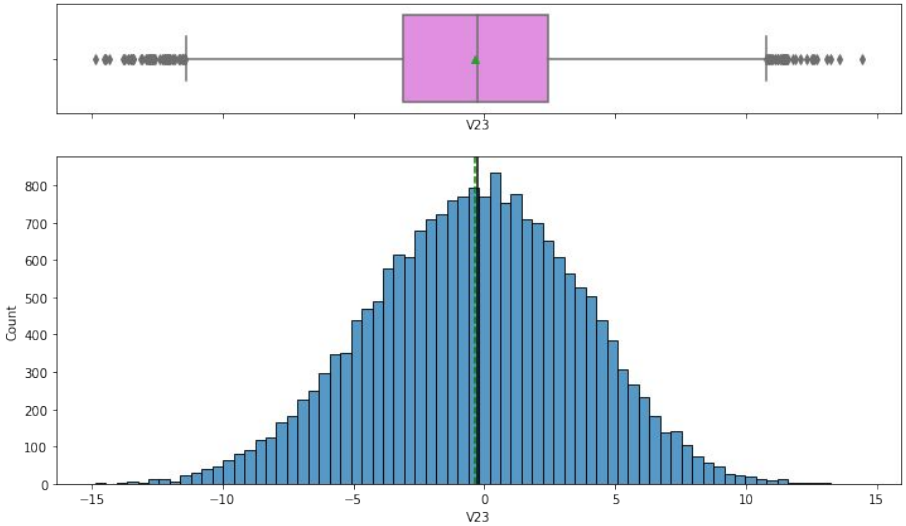
V21



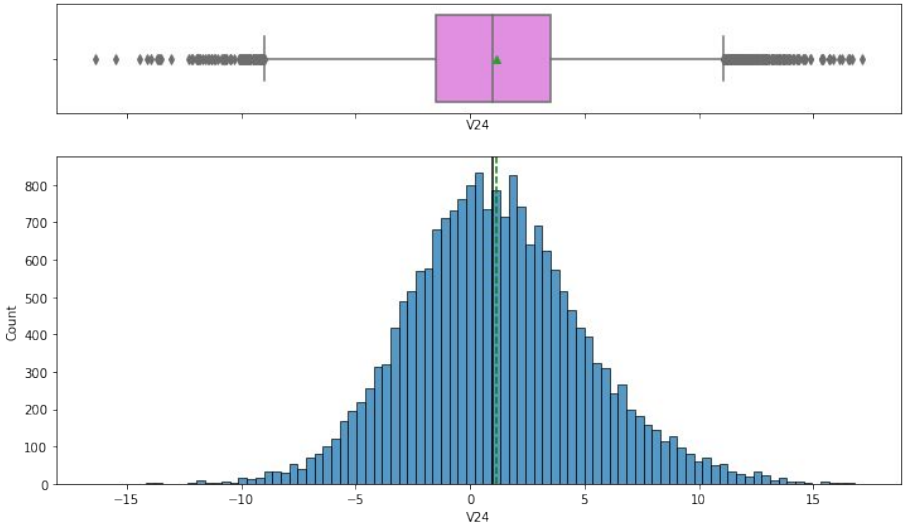
V22

EDA Results

UNIVARIATE ANALYSIS



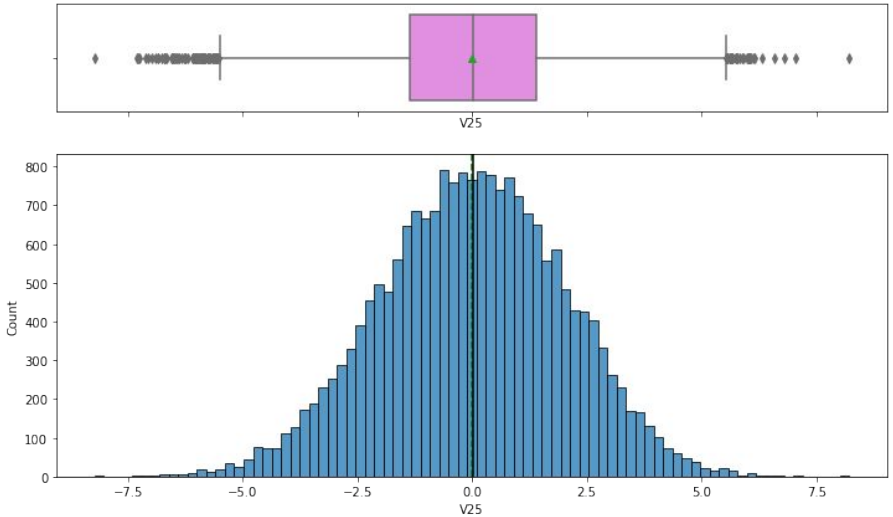
V23



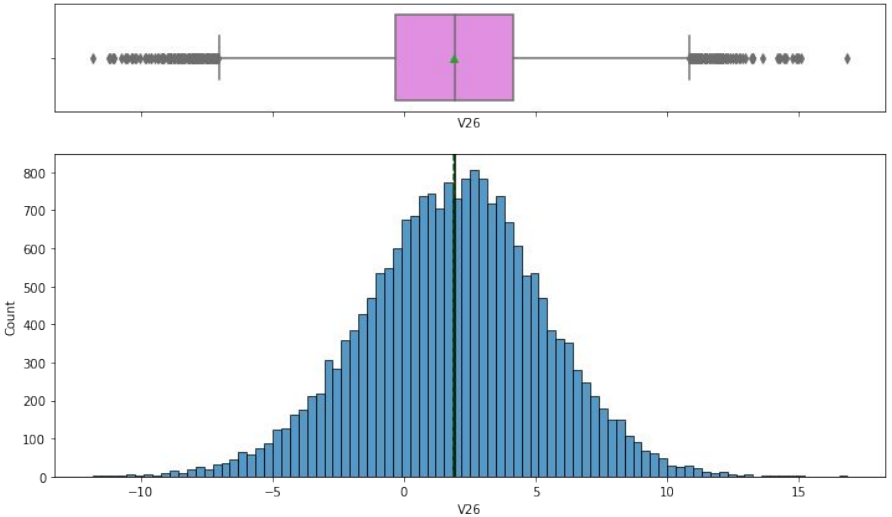
V24

EDA Results

UNIVARIATE ANALYSIS



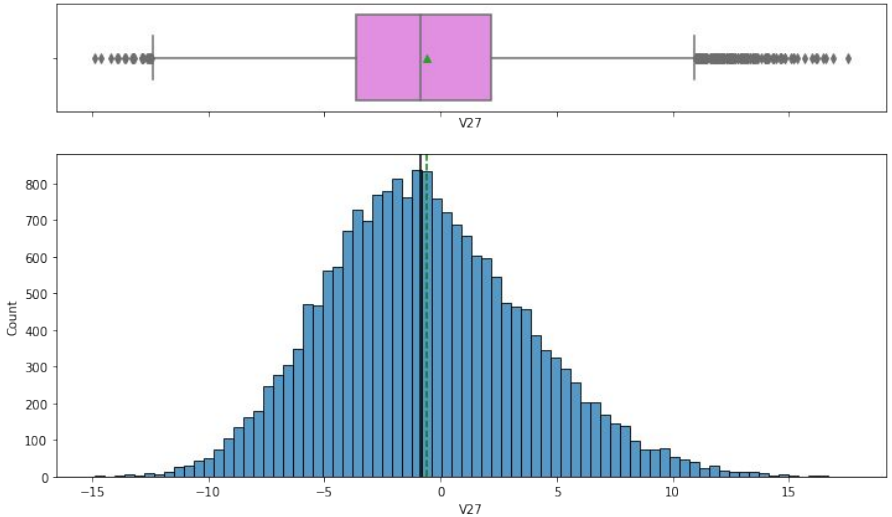
V25



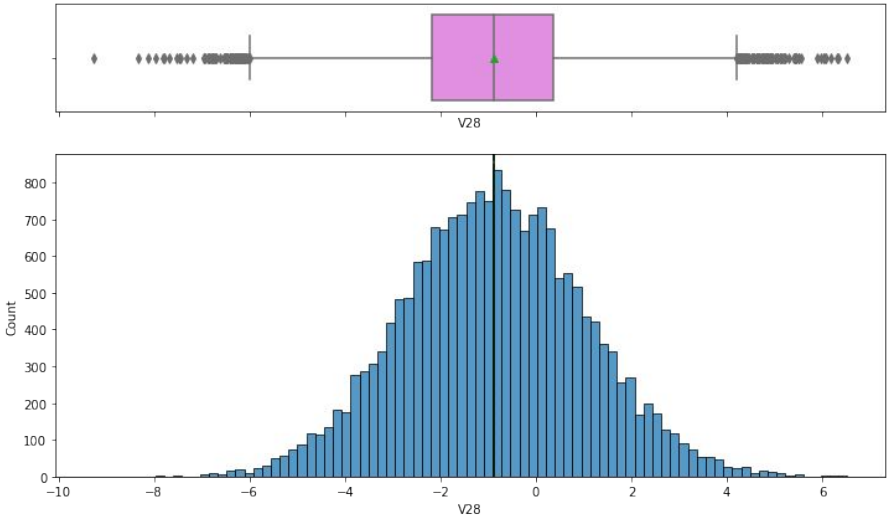
V26

EDA Results

UNIVARIATE ANALYSIS



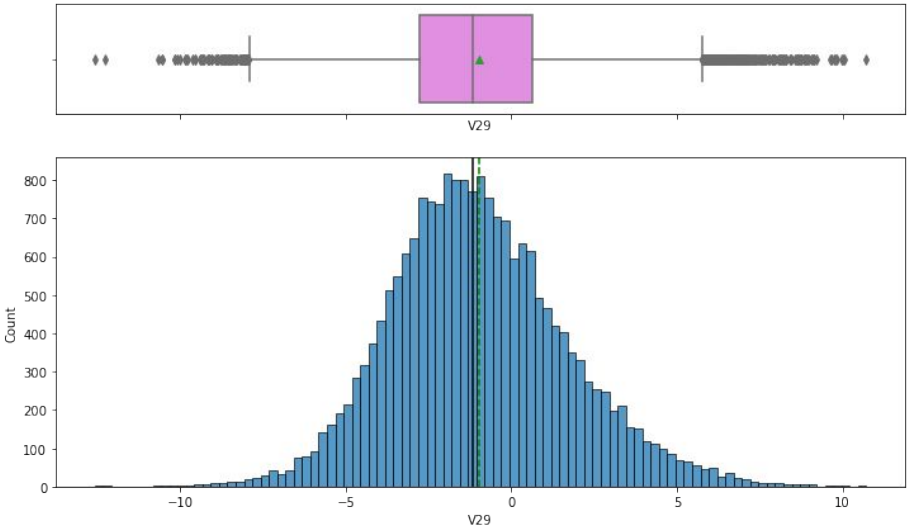
V27



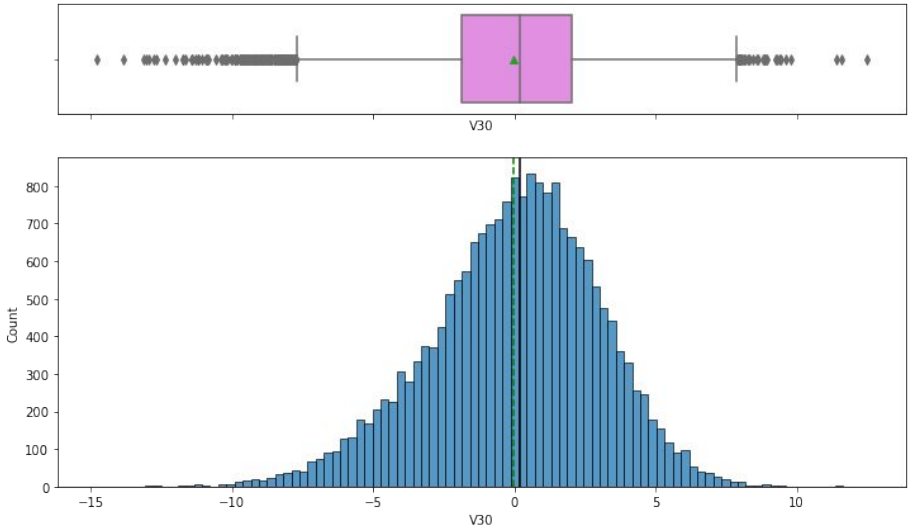
V28

EDA Results

UNIVARIATE ANALYSIS



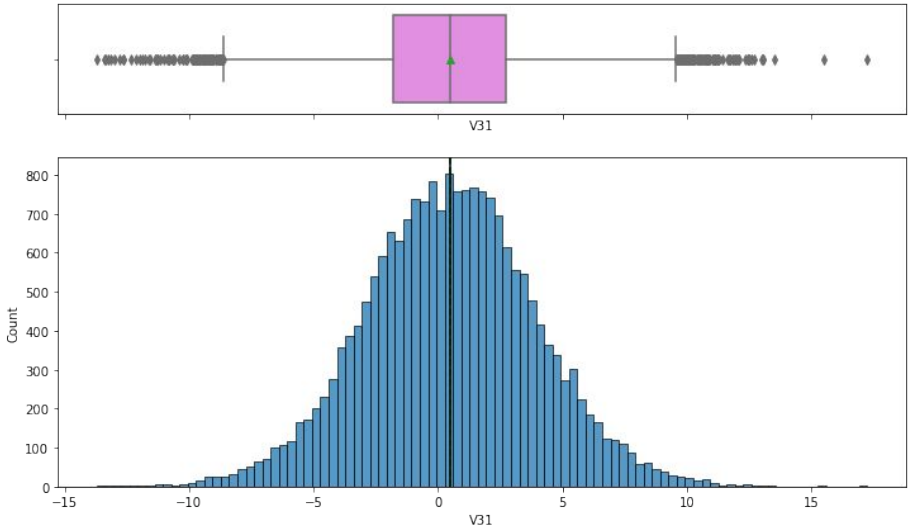
V29



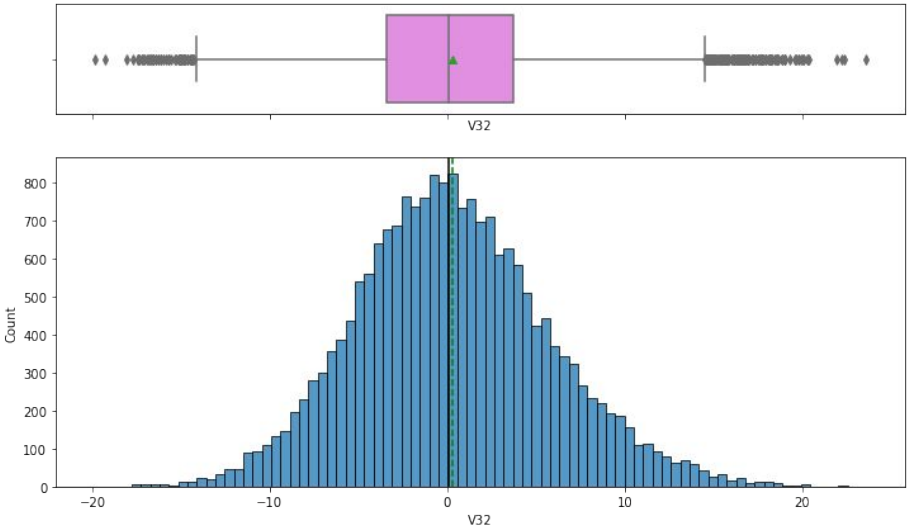
V30

EDA Results

UNIVARIATE ANALYSIS



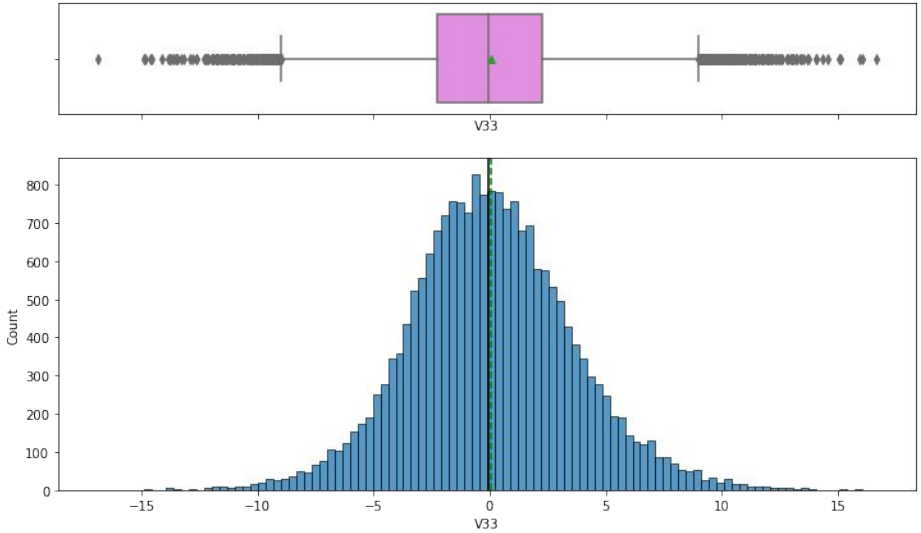
V31



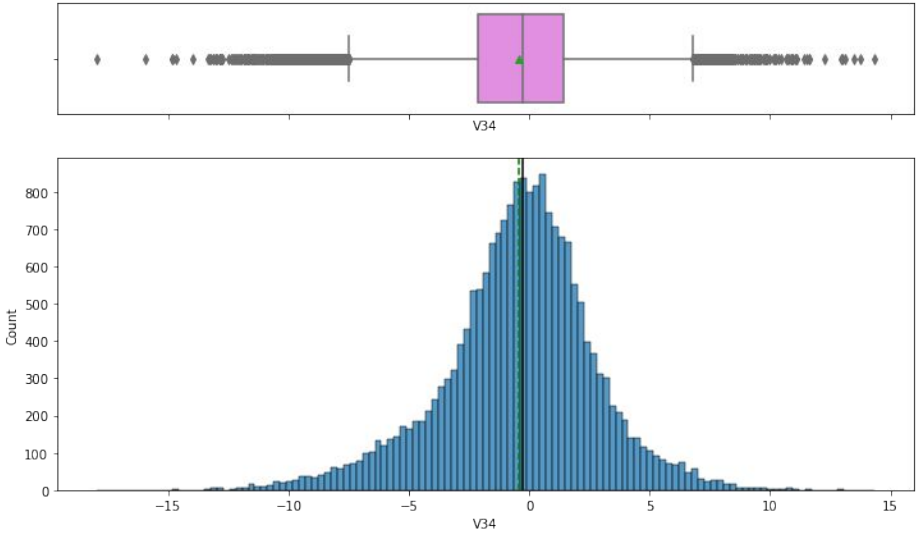
V32

EDA Results

UNIVARIATE ANALYSIS



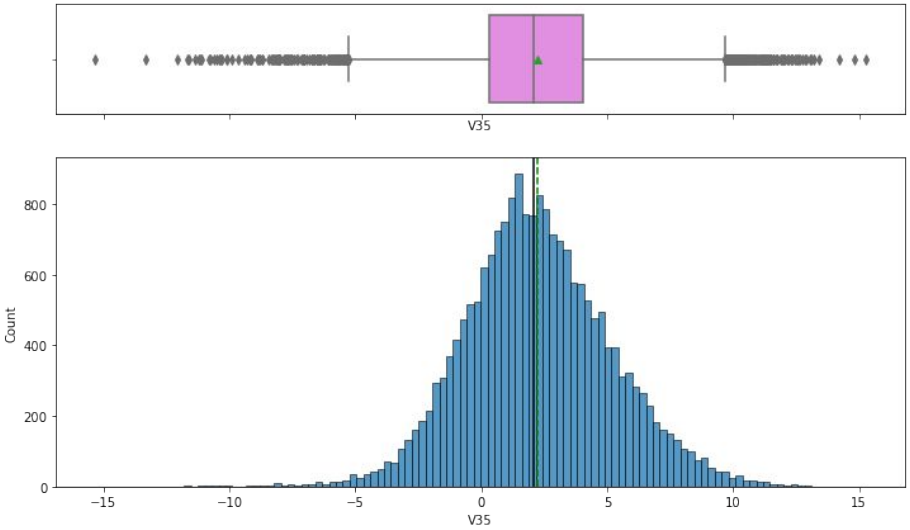
V33



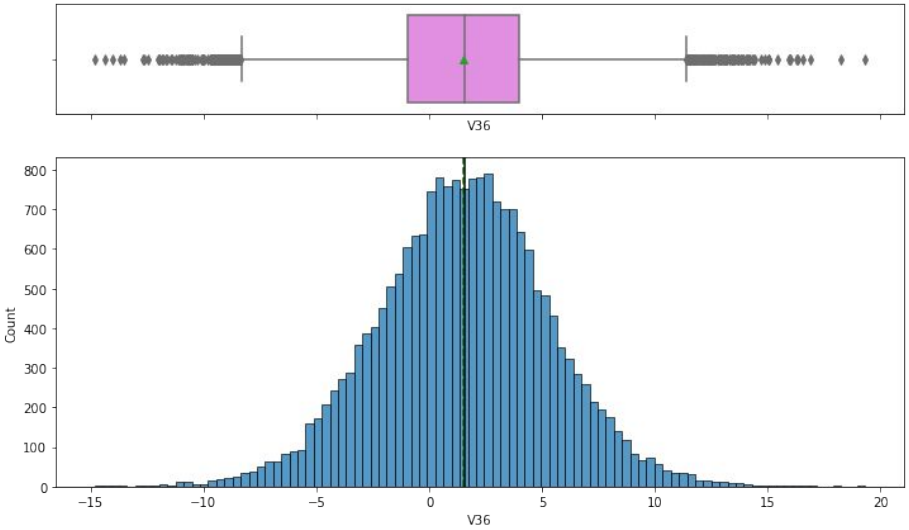
V34

EDA Results

UNIVARIATE ANALYSIS



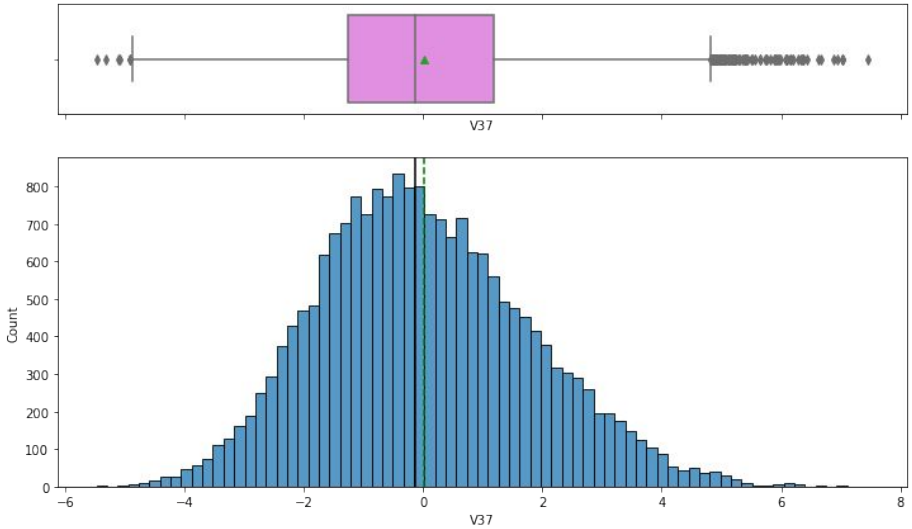
V35



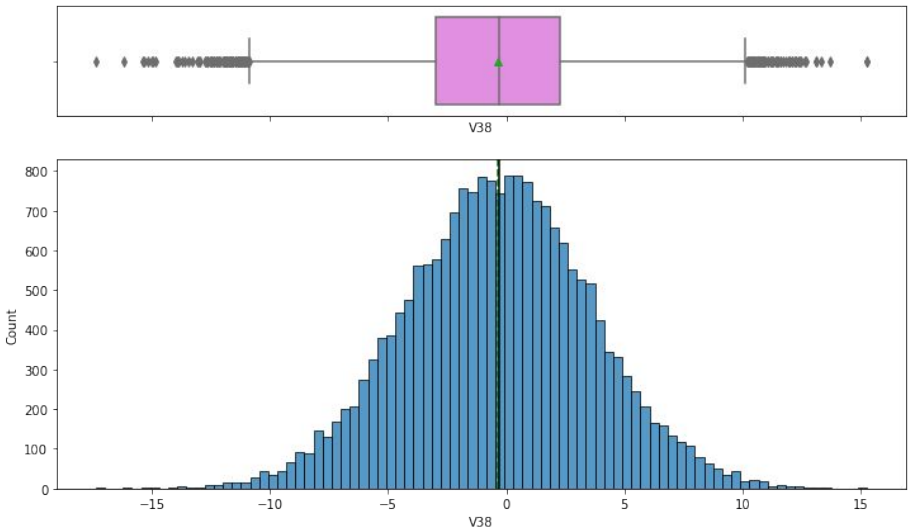
V36

EDA Results

UNIVARIATE ANALYSIS



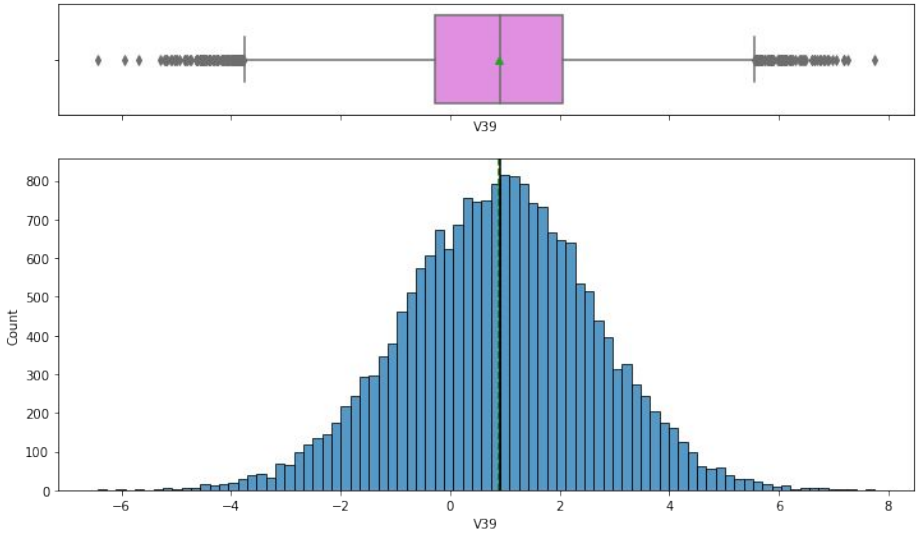
V37



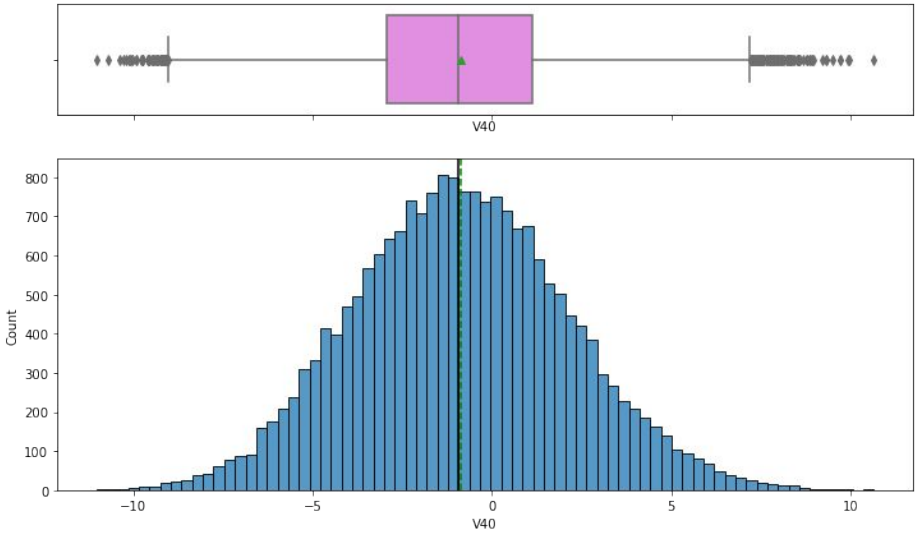
V38

EDA Results

UNIVARIATE ANALYSIS



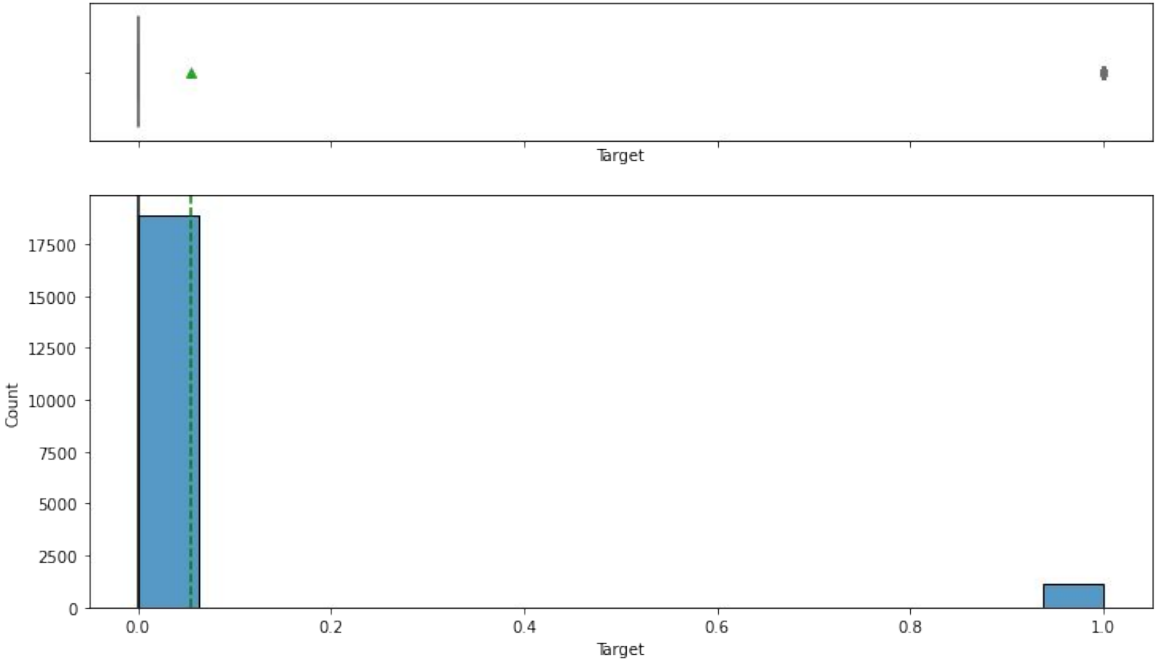
V39



V40

EDA Results

UNIVARIATE ANALYSIS



Target

0 = No failure
1 = Failure

The weight of the data is imbalanced as ‘no failure’ far outweighs ‘failure’ in the data. Hence, the application of undersampling and oversampling techniques to correct the imbalance.

THE END :)