Project report - Customer Transaction Prediction

Darshan Bhansali

Aim

In this project, we need to identify which customers will make a specific transaction in the future, irrespective of the amount of money transacted.

The Datasets

- Two datasets provided (“train.csv”, “test.csv”)

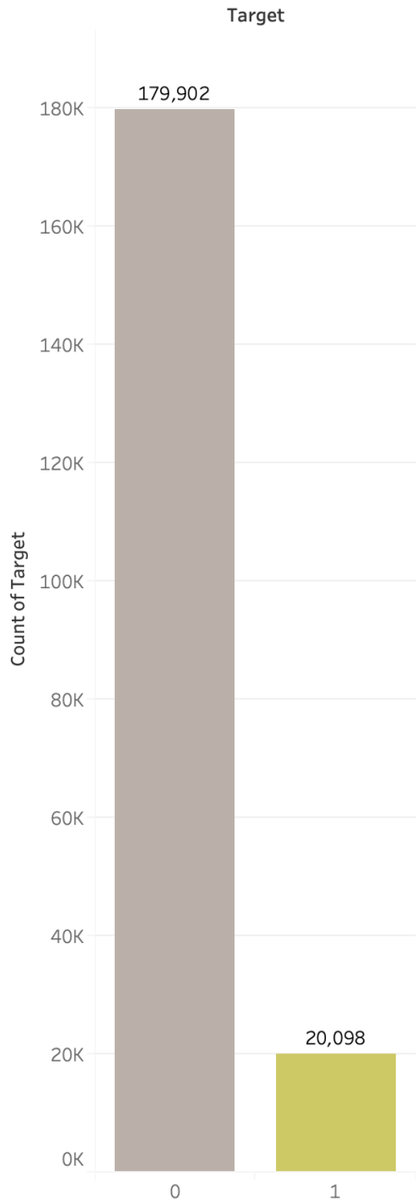
- Structure of train dataset:
  - ID variable : “ID_code”
  - Feature variables: 200 independent features (Var_0 - Var_199), 1 dependent variable (‘target’)
  - Observations : 200,000

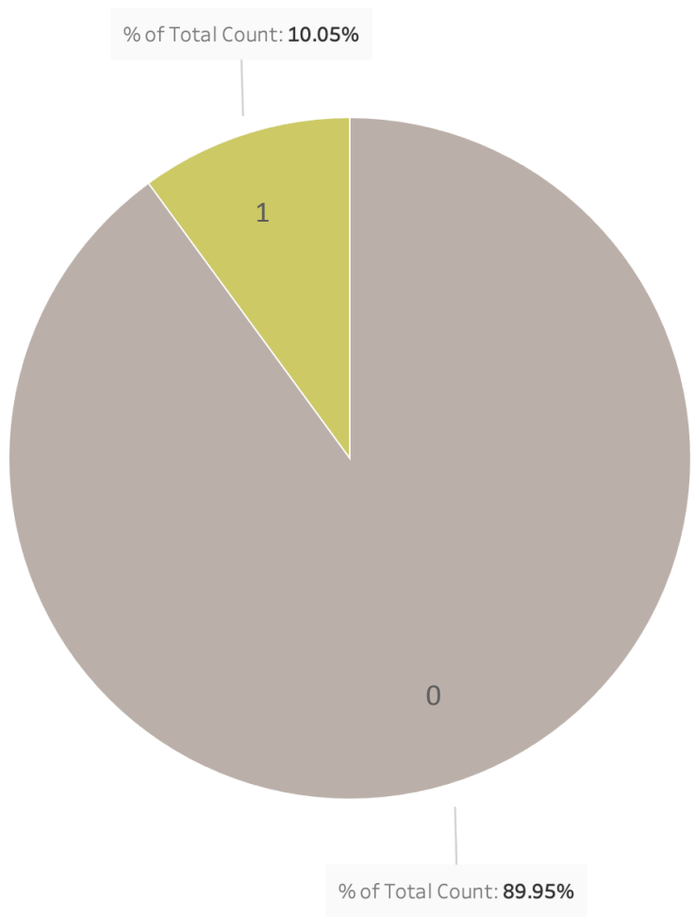
- Structure of test dataset:
  - ID variable : “ID_code”
  - Feature variables: 200 independent features (Var_0 - Var_199)
  - Observations : 200,000

Overview of the data

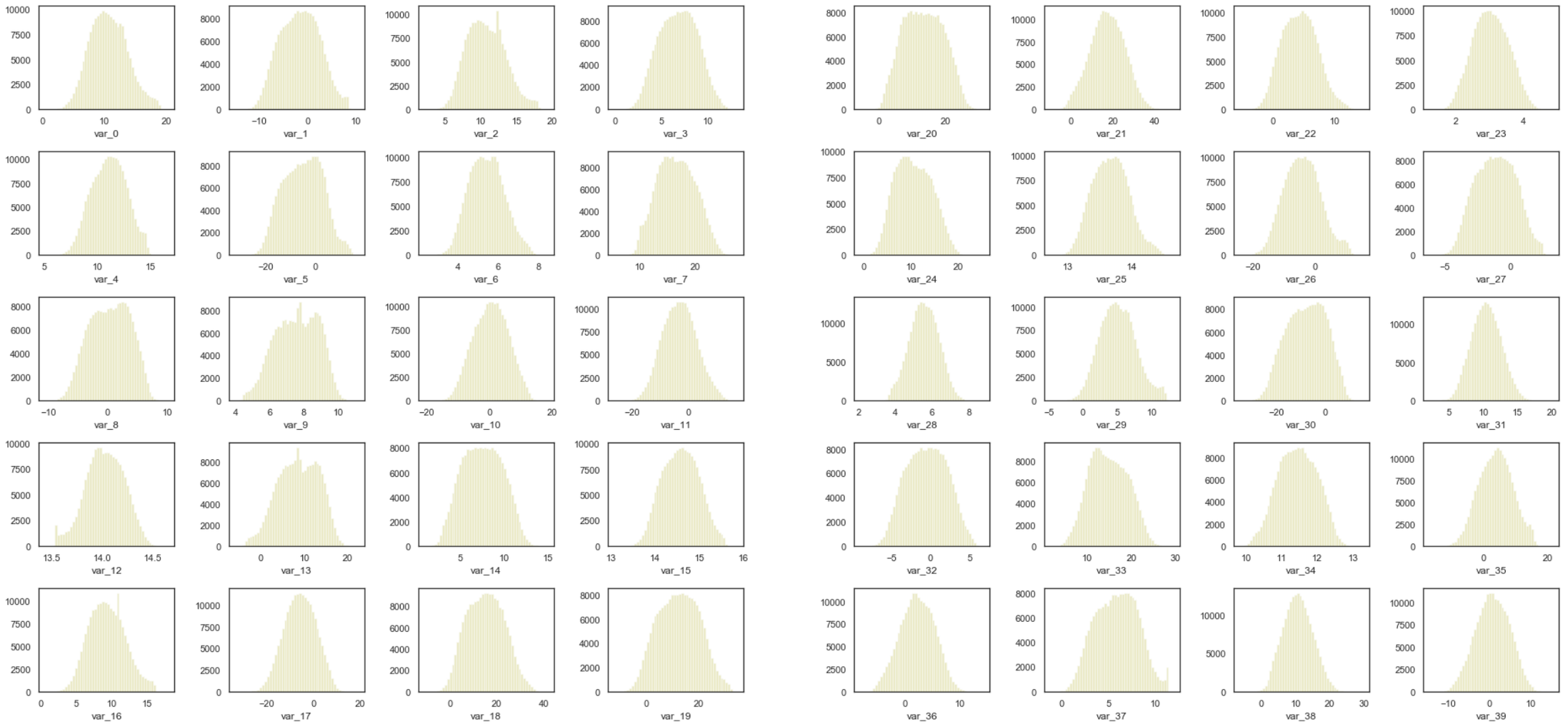
- We'll see the following:
  - Distribution of the target variable in the training data
  - Correlation heat map of the independent features of the train dataset
  - Correlation heat map of the independent features of the test dataset

Distribution of target variable in training data

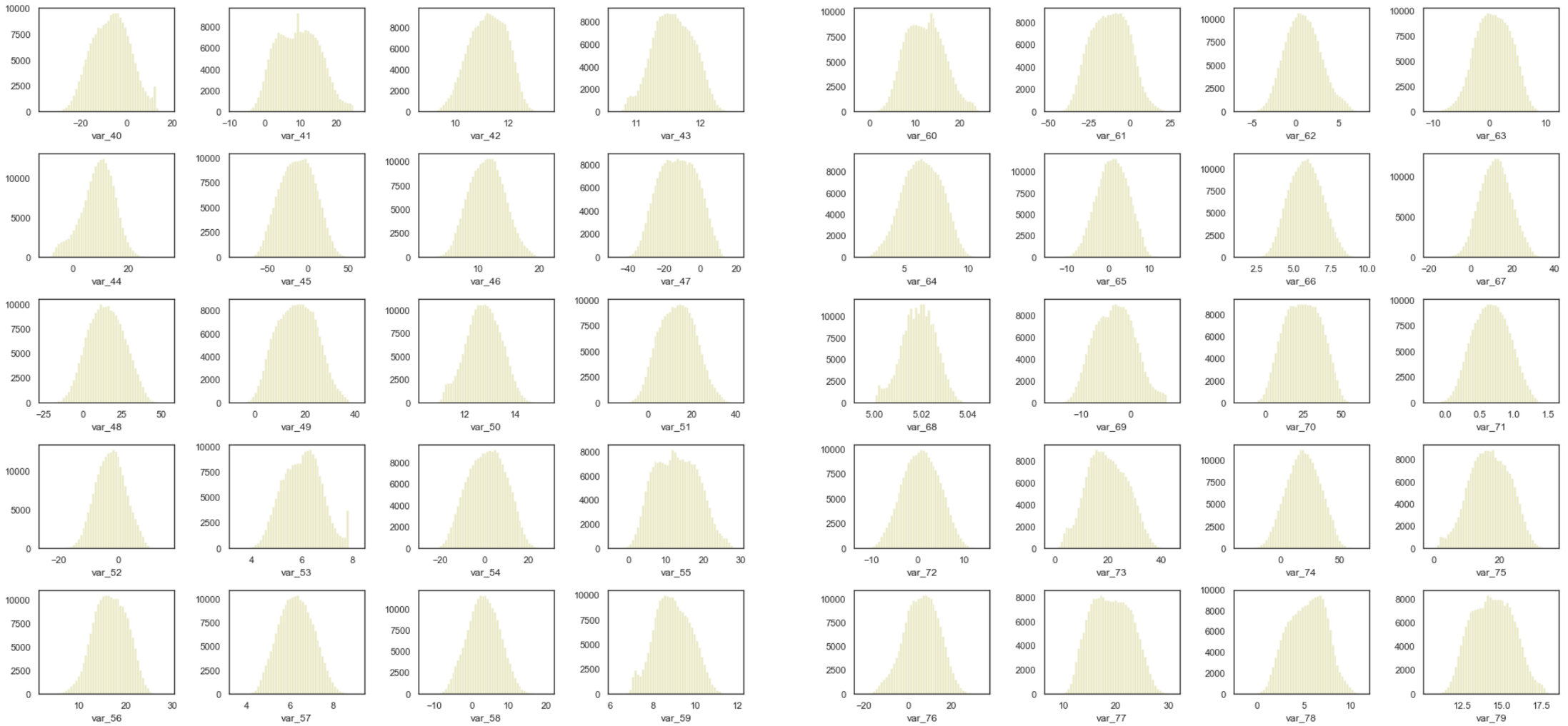




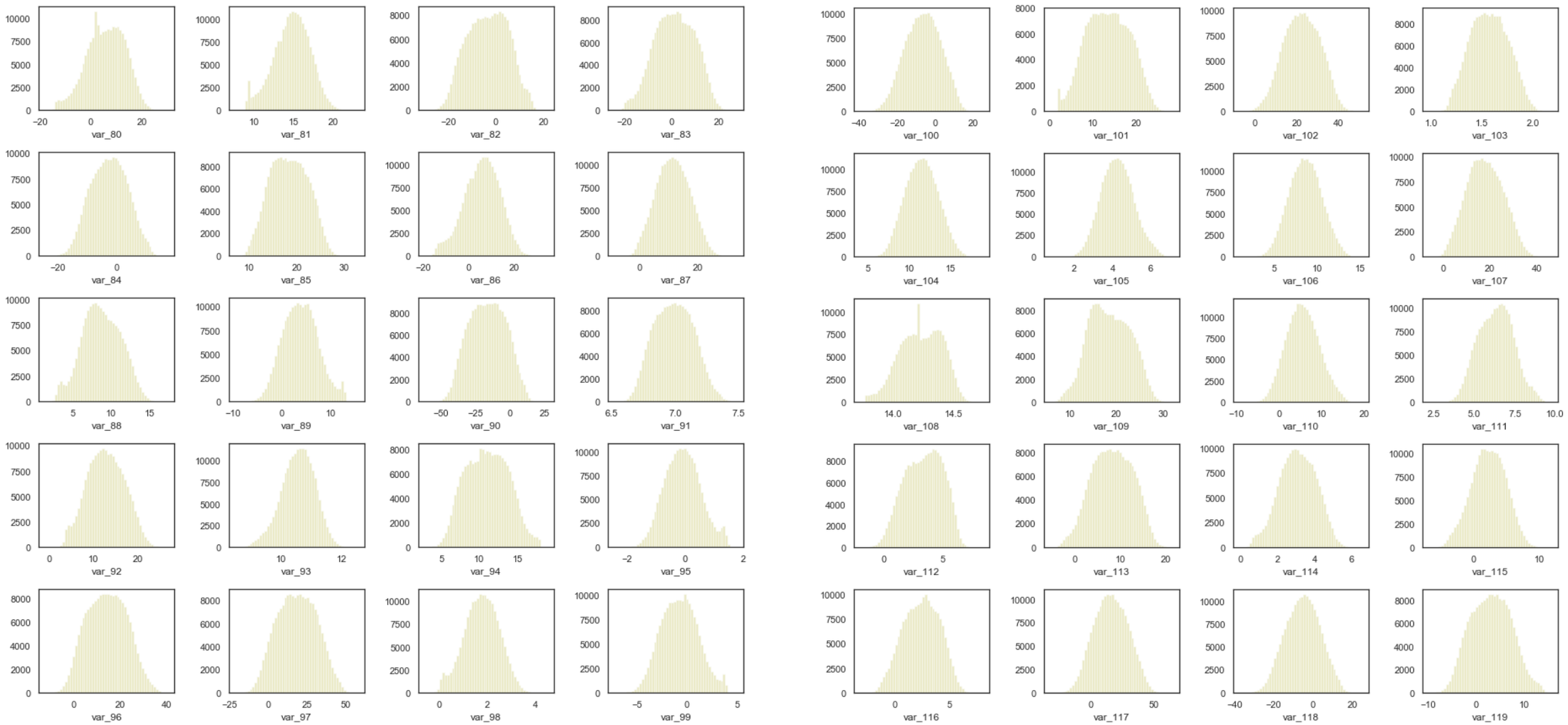
Distribution of independent features in training data



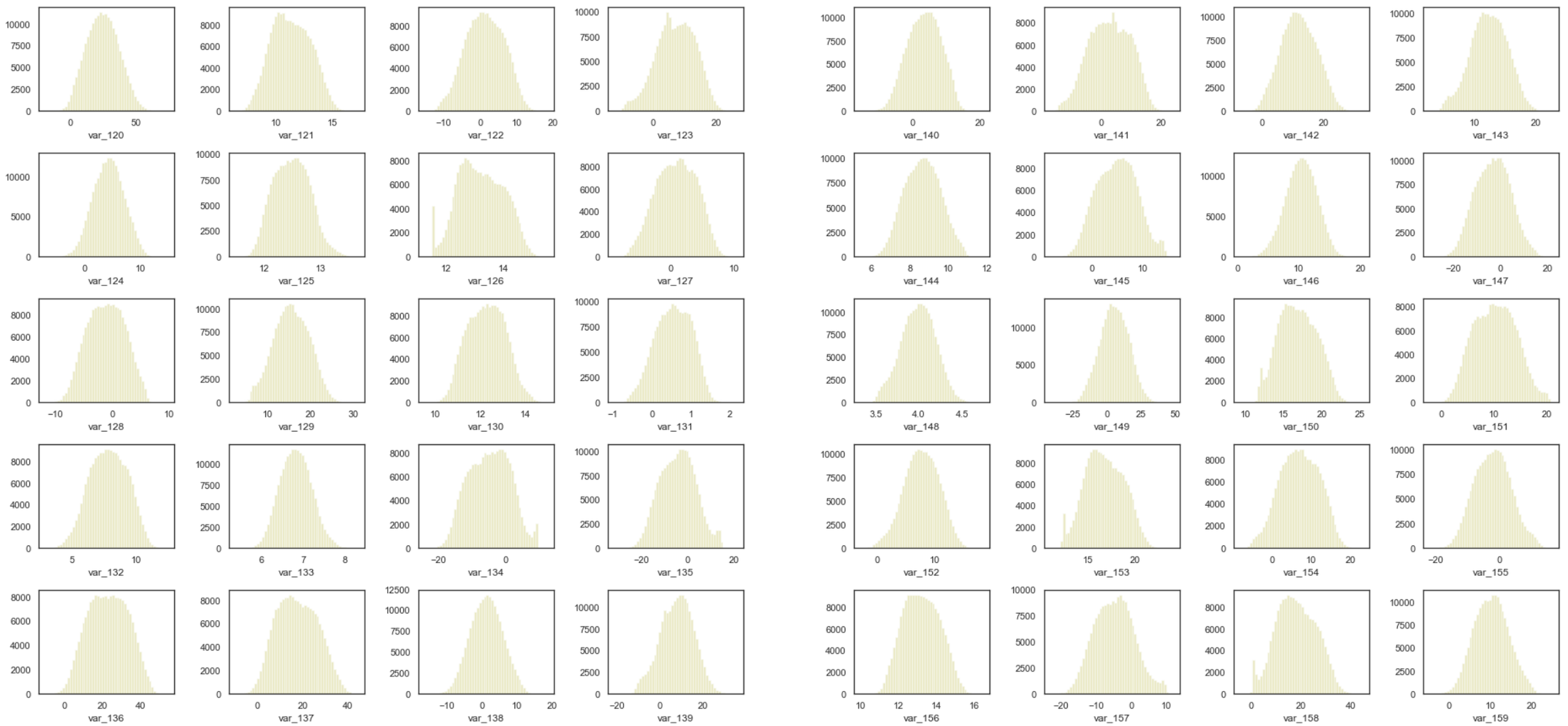
Distribution of independent features in training data



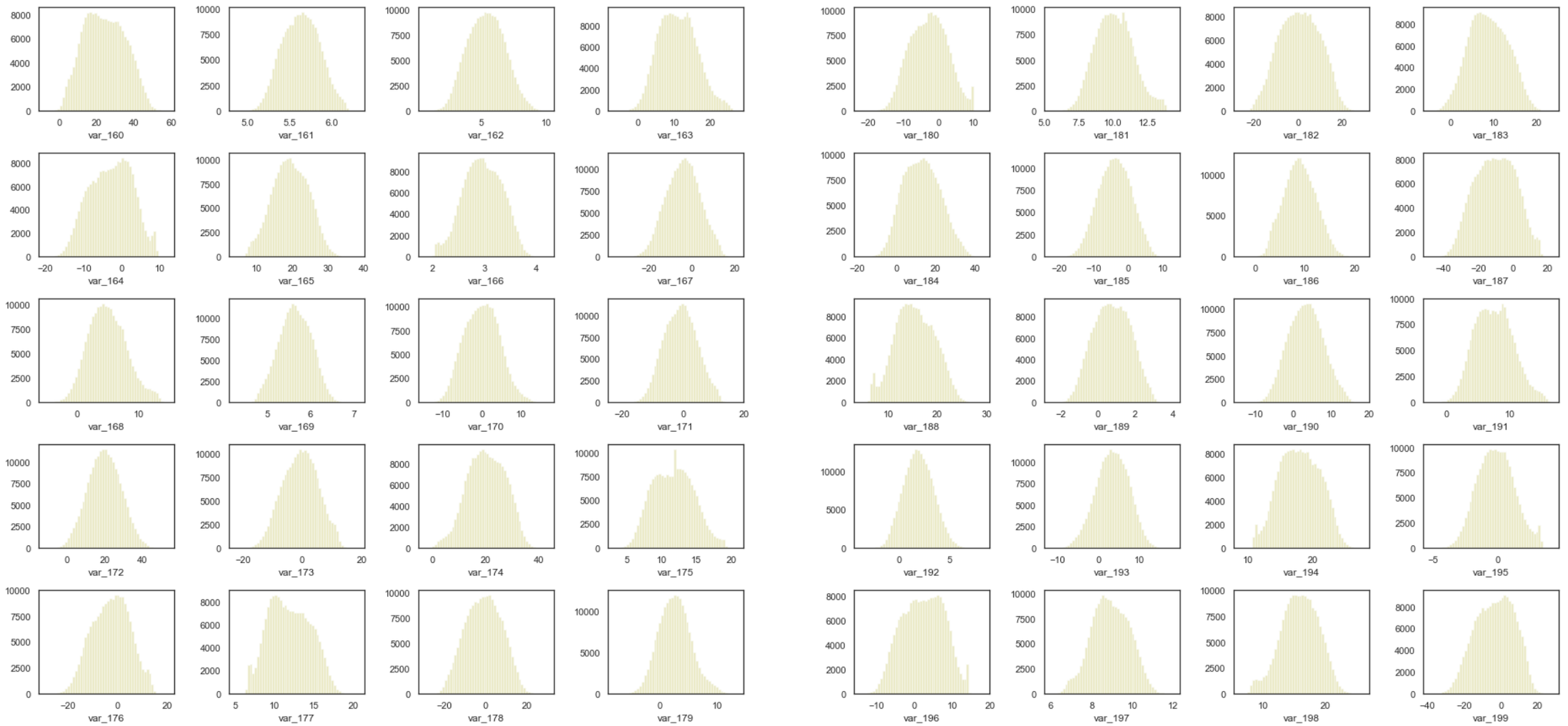
Distribution of independent features in training data



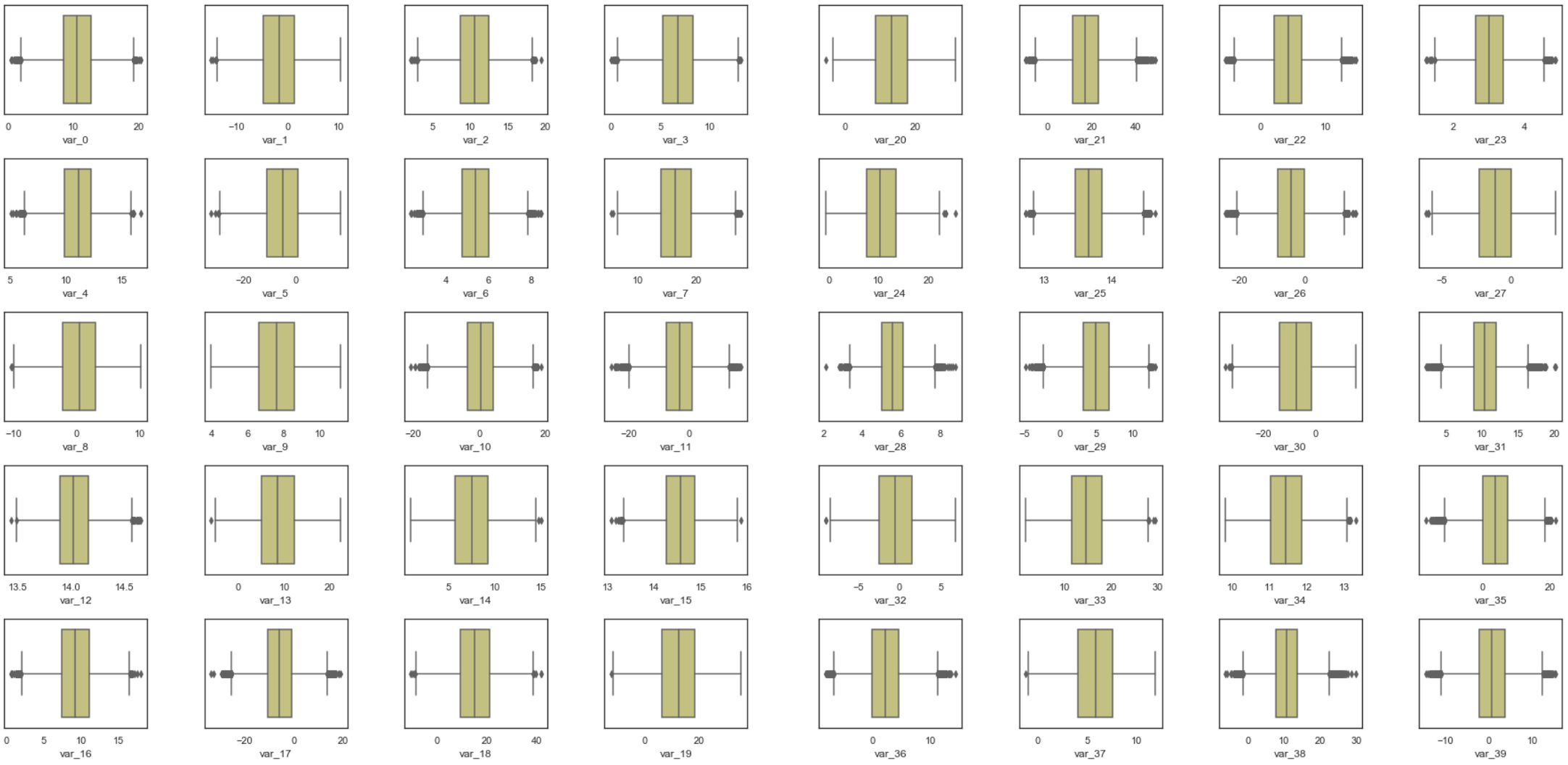
Distribution of independent features in training data



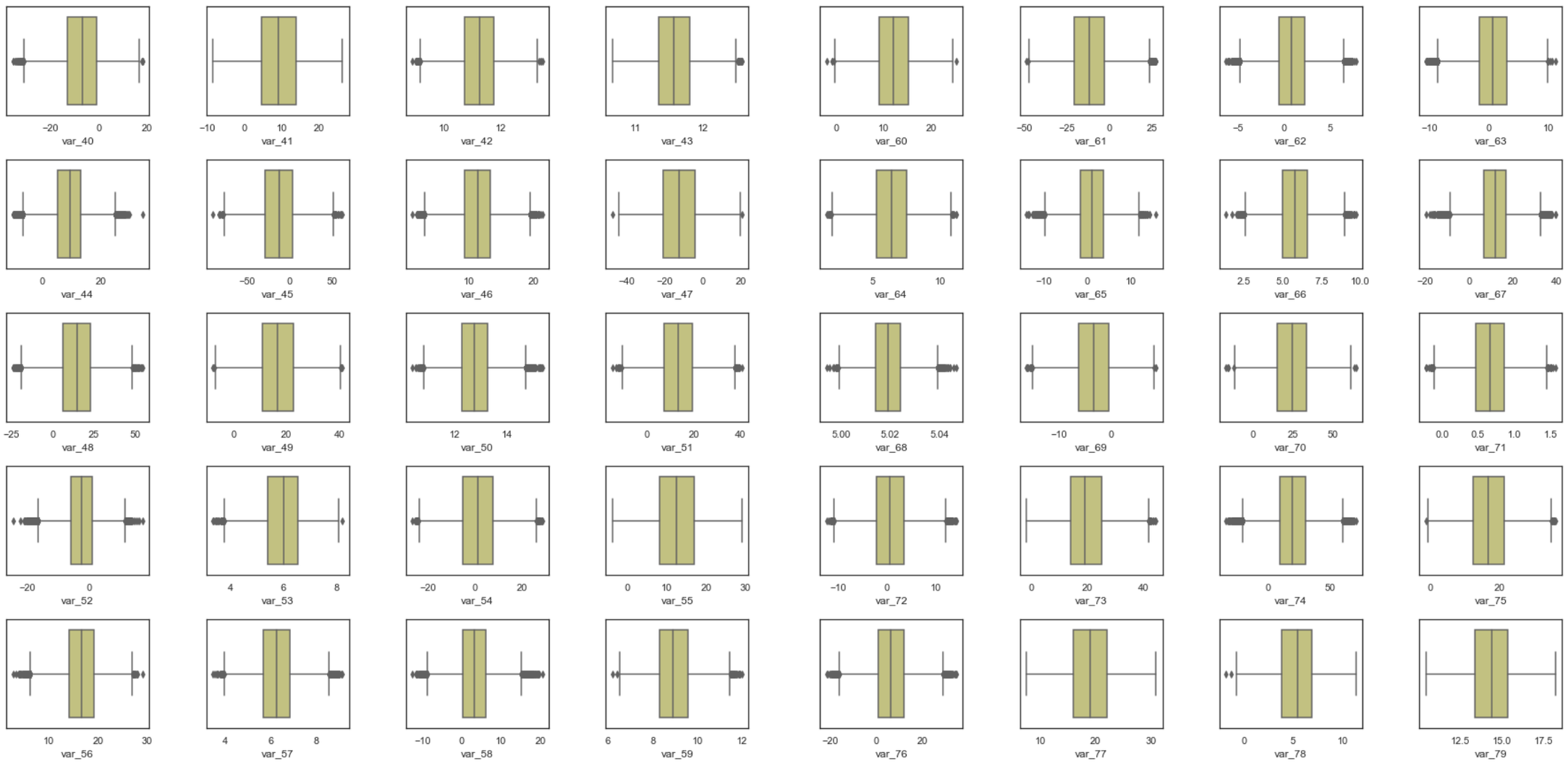
Distribution of independent features in training data



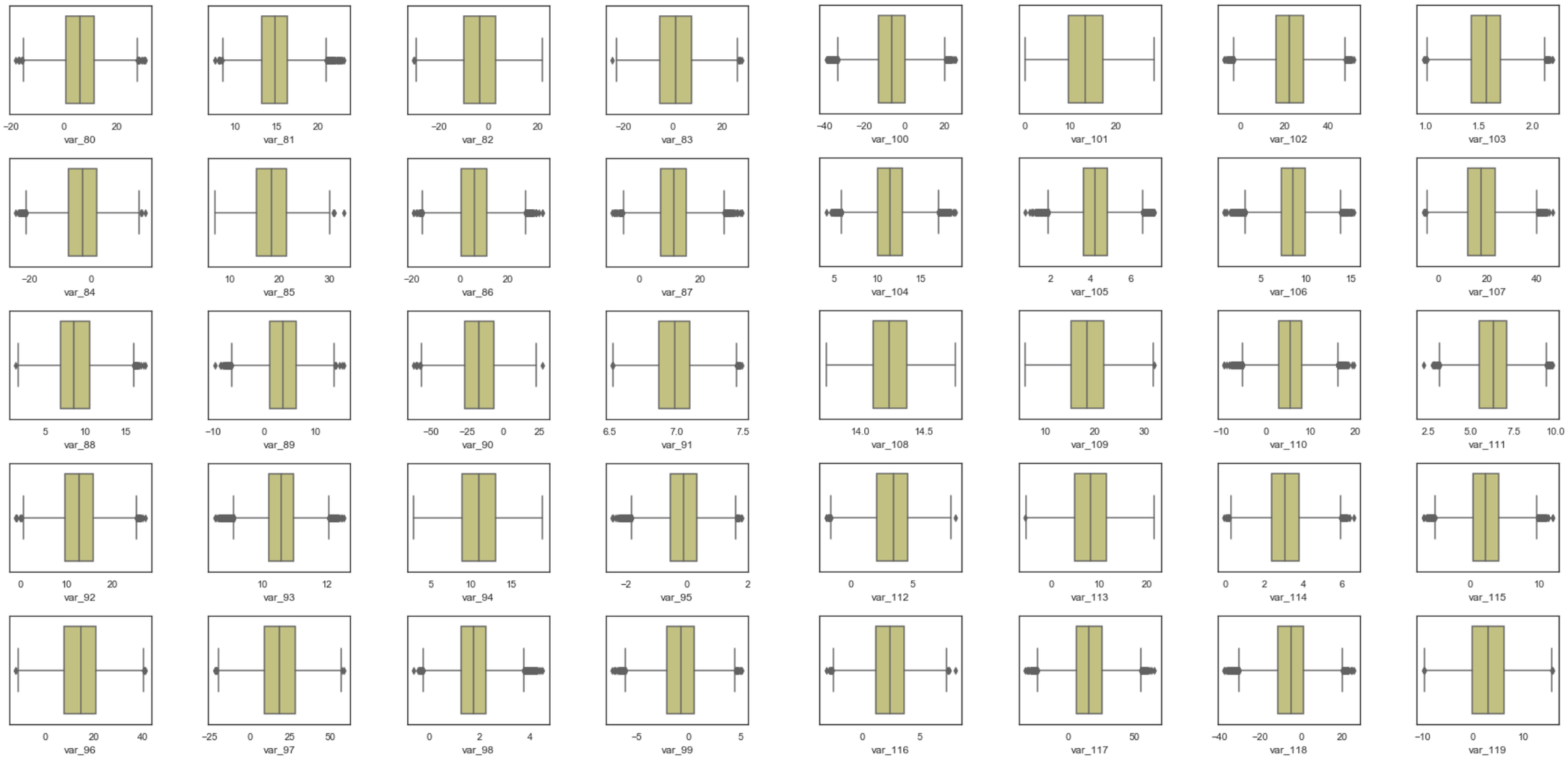
Box plots of independent features in training data



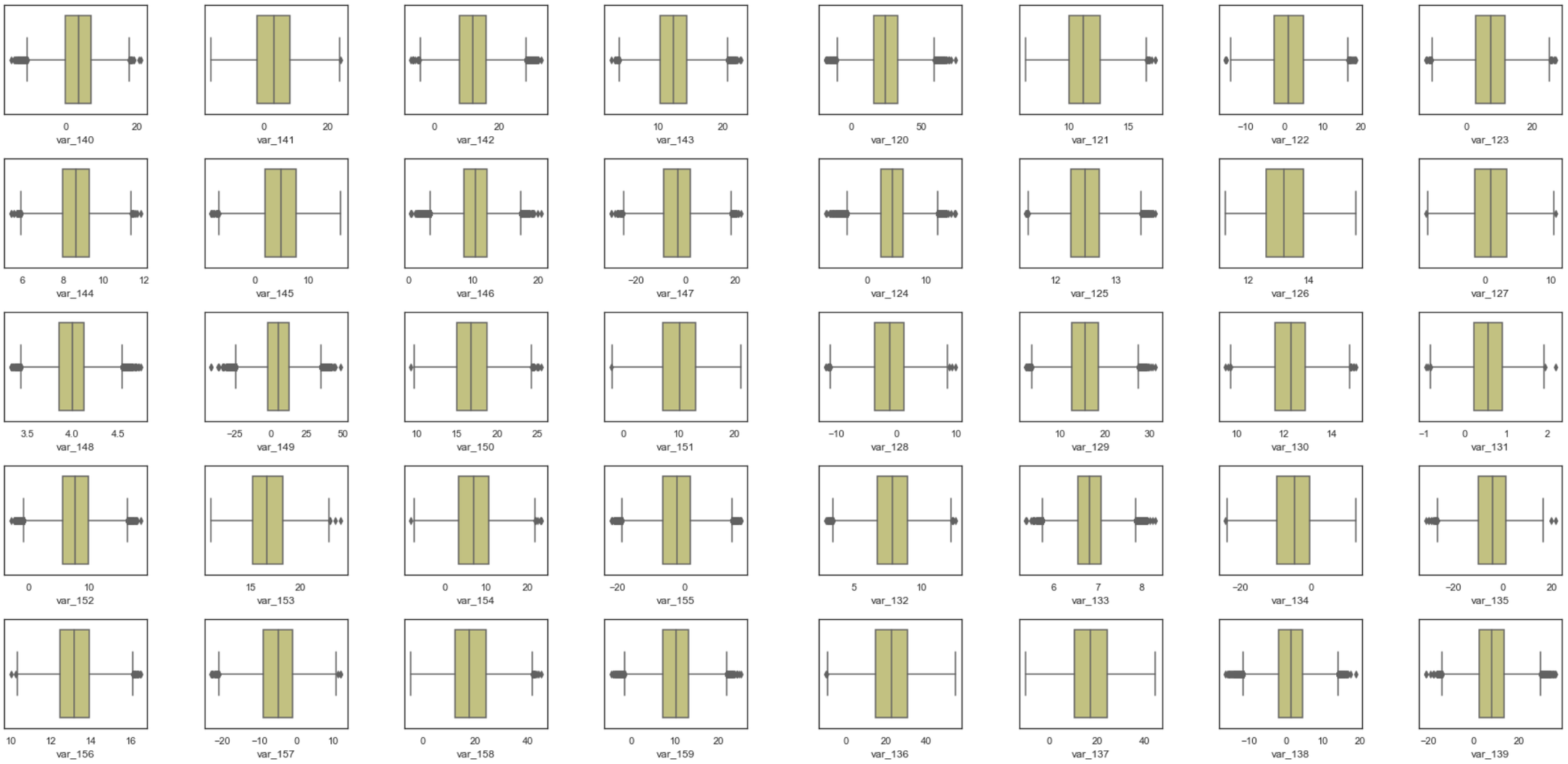
Box plots of independent features in training data



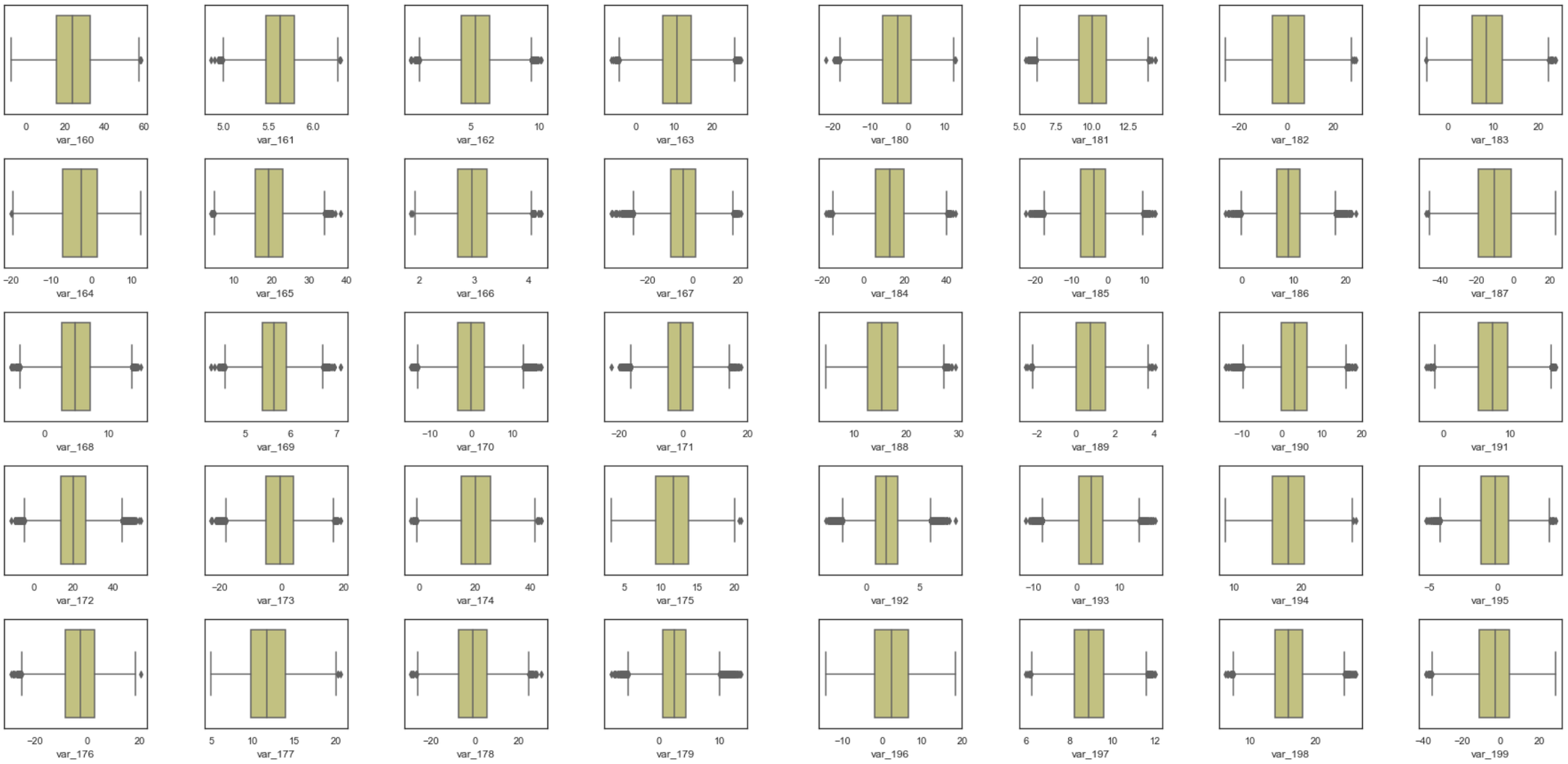
Box plots of independent features in training data



Box plots of independent features in training data



Box plots of independent features in training data

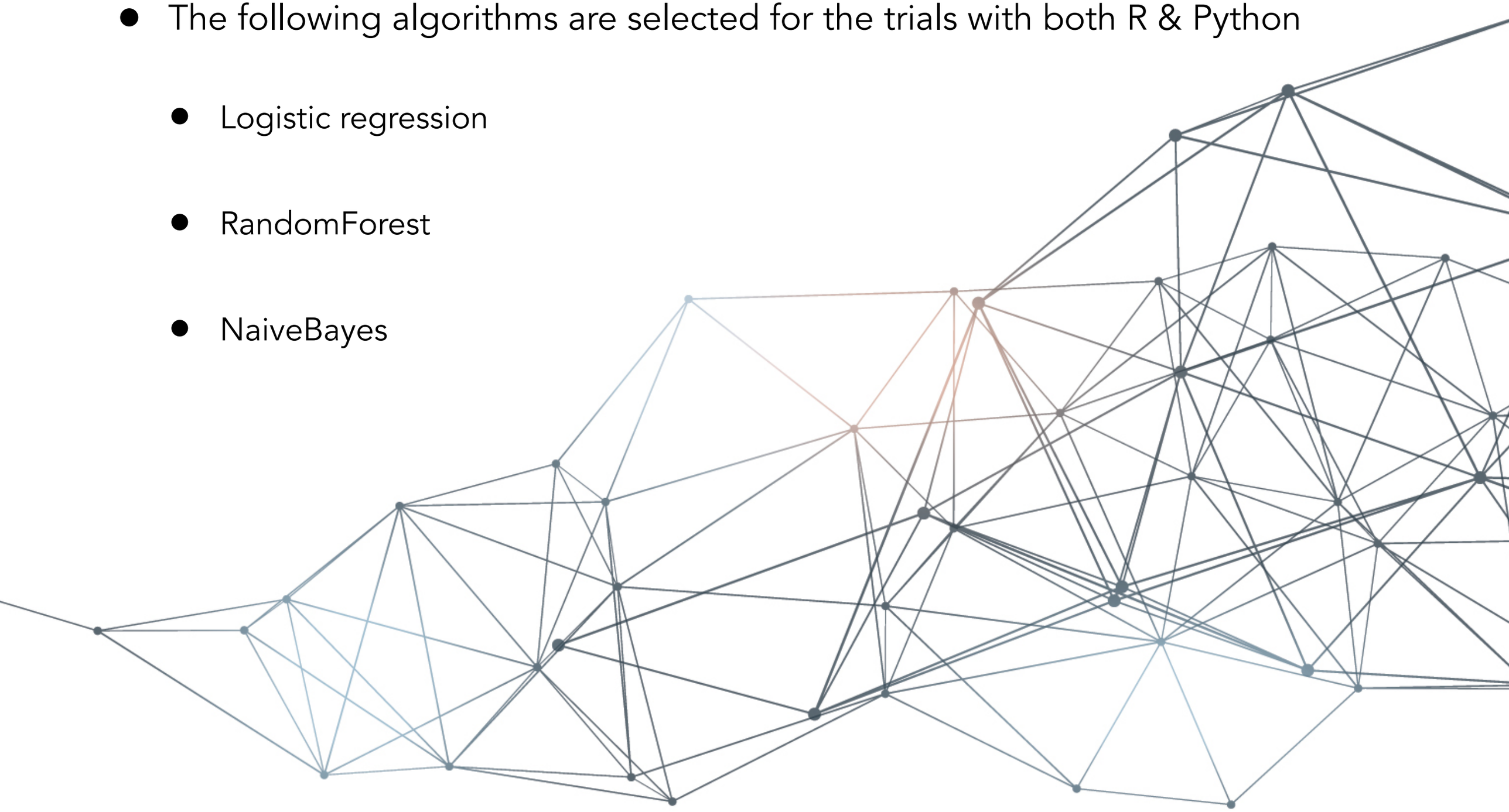


Observation & Preprocessing

- From the study of the distribution & box plots of the features in the training data, we can made the following inferences:
    - Most of the features are normally distributed
    - There is presence of outliers in the data but trials suggested that they do not affect the performance of the models
- The features are distributed over different scales and need to be standardised to streamline the models

Building models

- The following algorithms are selected for the trials with both R & Python
  - Logistic regression
  - RandomForest
  - NaiveBayes



Building models - R

- Trials with train dataset in R script generated the following metrics :

Logistic Regression

	Predicted	
Actual	0	1
0	44713	262
1	4145	879

Accuracy : 91.19%

FNR : 82.5%

AUC : 0.84

RandomForest

	Predicted	
Actual	0	1
0	44965	10
1	4990	34

Accuracy : 89.99%

FNR : 99.32%

AUC : 0.83

NaiveBayes

	Predicted	
Actual	0	1
0	44234	741
1	3191	1833

Accuracy : 92.13%

FNR : 63.52%

AUC : 0.82

Building models - R

- To balance the bias in target class, we retry with oversampled data :

Logistic Regression

Oversampled Data

	Predicted	
Actual	0	1
0	38903	6072
1	1674	3350

Accuracy : 84.50%

FNR : 33.32%

AUC : 0.65

RandomForest

Oversampled Data

	Predicted	
Actual	0	1
0	44958	17
1	4996	28

Accuracy : 89.97%

FNR : 99.44%

AUC : 0.76

NaiveBayes

Oversampled Data

	Predicted	
Actual	0	1
0	36588	8387
1	991	40339

Accuracy : 81.24%

FNR : 19.72%

AUC : 0.64

Building models - R

- NaiveBayes with oversampling provides the best Accuracy : FNR ratio

Logistic Regression

Oversampled Data

Actual	Predicted 0	Predicted 1
0	38903	6072
1	1674	3350

Accuracy : 84.50%

FNR : 33.32%

AUC : 0.65

RandomForest

Oversampled Data

Actual	Predicted 0	Predicted 1
0	44958	17
1	4996	28

Accuracy : 89.97%

FNR : 99.44%

AUC : 0.76

NaiveBayes

Oversampled Data

Actual	Predicted 0	Predicted 1
0	36588	8387
1	991	40339

Accuracy : 81.24%

FNR : 19.72%

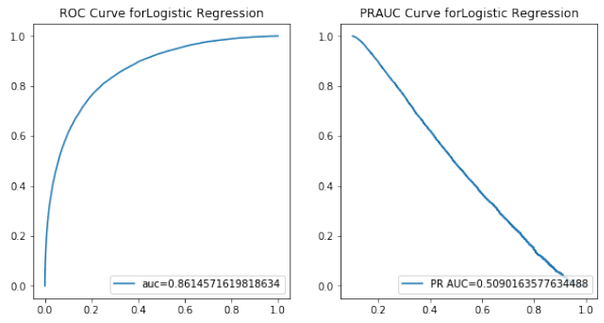
AUC : 0.64

Building models - Python

- Trials with train dataset in python script generated the following metrics :

Logistic Regression

Actual	Predicted 0	Predicted 1
0	44713	262
1	4145	879

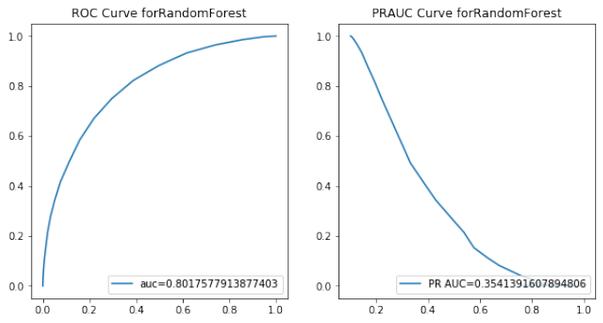


Accuracy : 78.26%

FNR : 22.05%

RandomForest

Actual	Predicted 0	Predicted 1
0	59367	1
1	6622	10

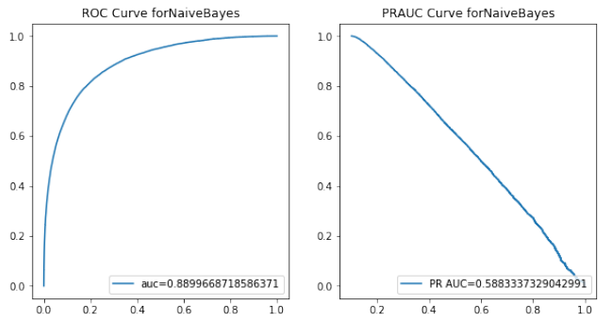


Accuracy : 89.96%

FNR : 99.84%

NaiveBayes

Actual	Predicted 0	Predicted 1
0	44713	262
1	4145	879



Accuracy : 92.16%

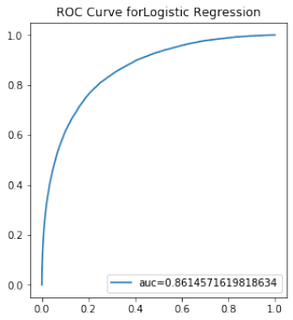
FNR : 63.26%

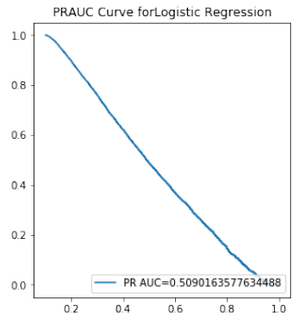
Building models - Python

- Logistic Regression with balanced strata provides the best Accuracy : FNR ratio

Logistic Regression

Actual	Predicted 0	Predicted 1
0	44713	262
1	4145	879





Accuracy : 78.26%

FNR : 22.05%

RandomForest

Actual	Predicted 0	Predicted 1
0	59367	1
1	6622	10

Accuracy : 89.96%

FNR : 99.84%

NaiveBayes

Actual	Predicted 0	Predicted 1
0	44713	262
1	4145	879

Accuracy : 92.16%

FNR : 63.26%