MS4610: Introduction to Data Analytics

Data Contest 2020

Project - Loan Default Prediction

GROUP - 20

ME17B112 | ED17B014 | ME17B106 | ME17B120 | ME18B062|ED17B042 | ME17B043

Introduction

Our objective is to predict whether the loan by an applicant will default or not. Following are the different metrics that were given about each applicant and corresponding loan.

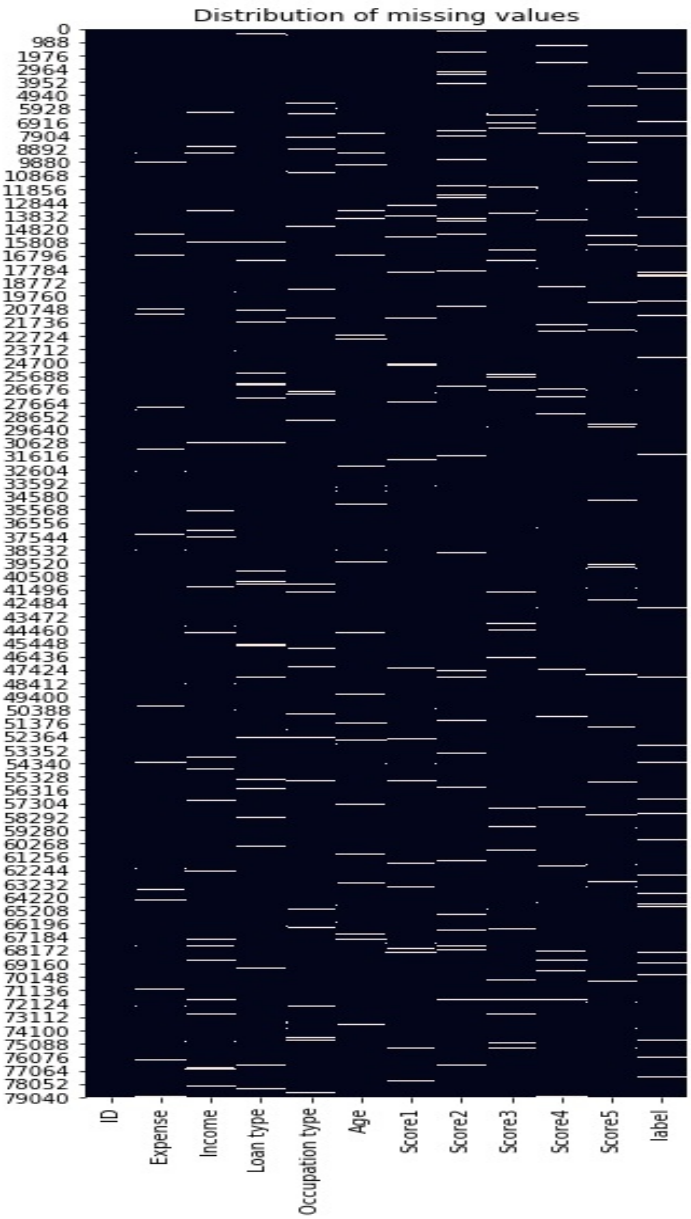
1. ID: A unique identifier for every financial loan that is being considered.
2. Loan type: Type of loan taken.
3. Occupation type: Occupation of the customer.
4. Income: A continuous variable that is indicative of the annual income of the customer. This is not the exact income value.
5. Expense: A continuous variable that is indicative of the annual expenditure of the customer. This is not the exact expense value.
6. Age: Age of customer – Value of ‘0’ is considered as below 50, and the value of ‘1’ is considered as above 50.
7. Score1, Score2, Score3, Score4, Score5: Represents five different metrics calculated by the organization about the customer and the loan that is being considered.
8. Label: says whether the loan is defaulted or not.

EDA

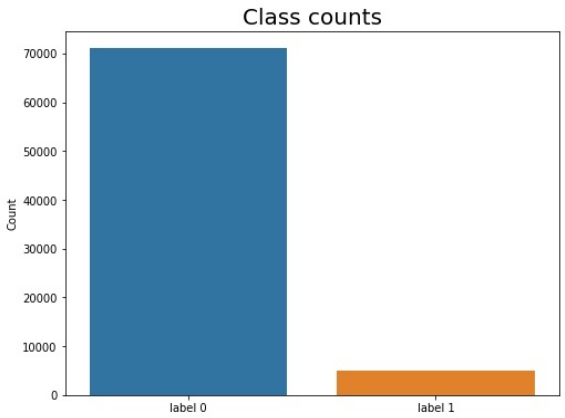
Here we explored the dataset statistically and graphically. The following are the significant areas we analyzed.

1. Quantity and distribution of missing values in the dataset
2. Correlation of features
3. Distribution of features
4. Order of feature importance of given features

Distribution of NaN



We can see that the missing values are distributed throughout the features and not concentrated on some areas. They can be filled by using appropriate imputation functions, which we will discuss later in this report.

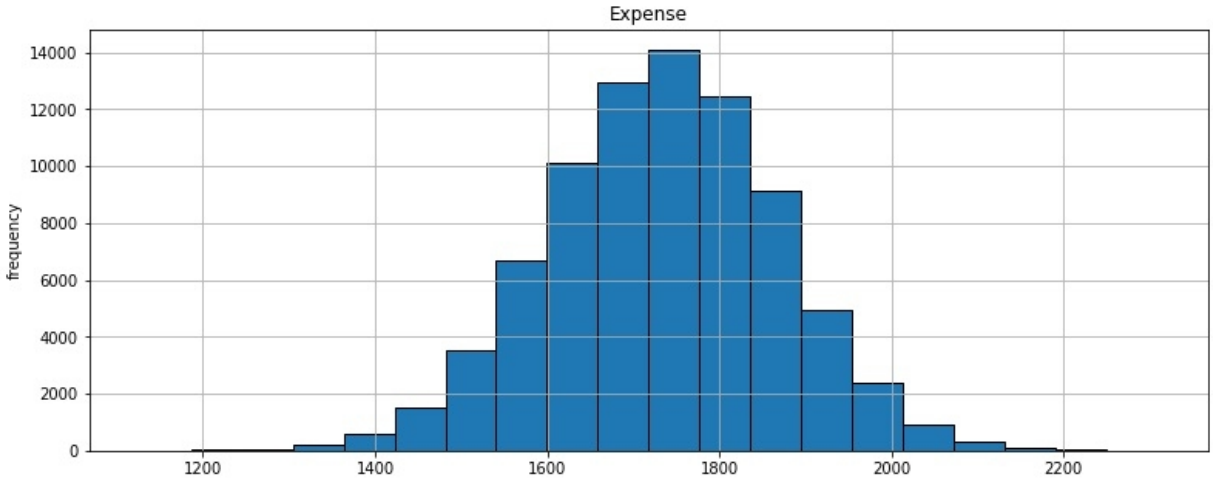


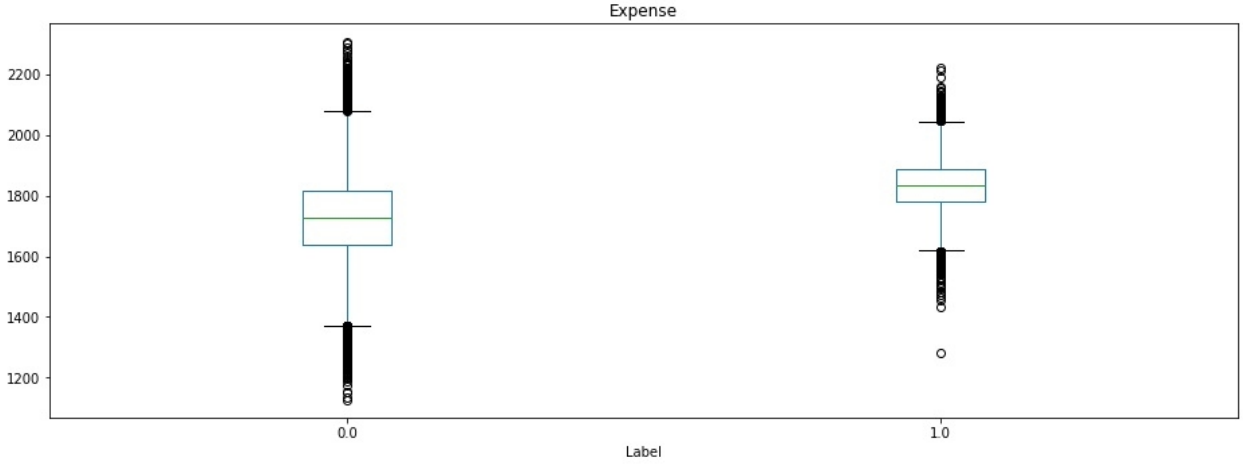
From the above plot, we can see there is a significant imbalance between the two classes. There are 71064 ‘0 label class’ points, while there are only 5033 ‘ 1 label’ class points.

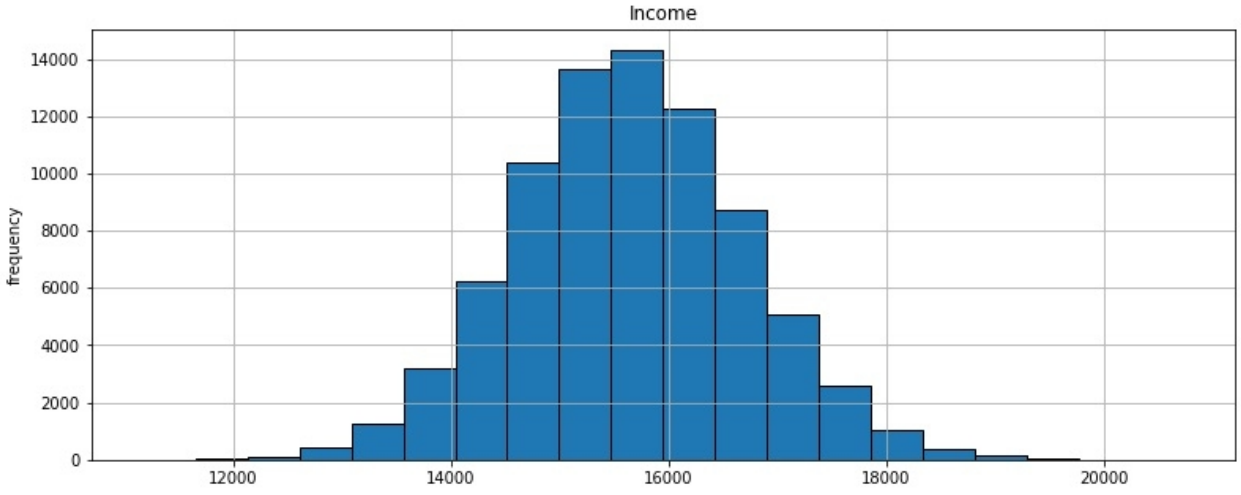
	column	unique values	% of total
0	ID	76097	100.000000
6	Score1	74252	97.575463
8	Score3	74243	97.563636
2	Income	74224	97.538668
9	Score4	74204	97.512386
10	Score5	74200	97.507129
7	Score2	74154	97.446680
1	Expense	74145	97.434853
4	Occupation type	3	0.003942
3	Loan type	2	0.002628
5	Age	2	0.002628
11	label	2	0.002628

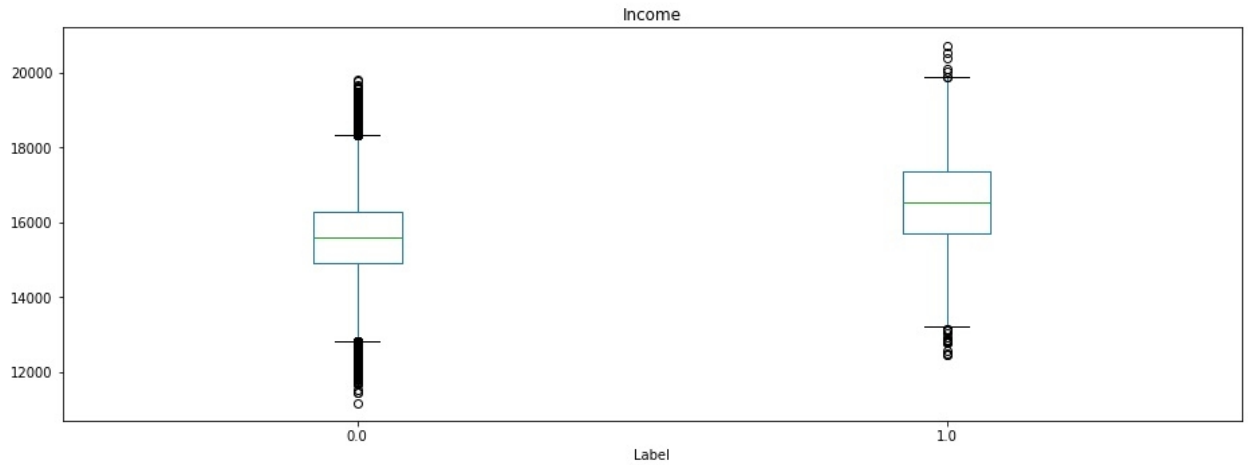
Occupation, loan type, age, and label are categorical as already known.

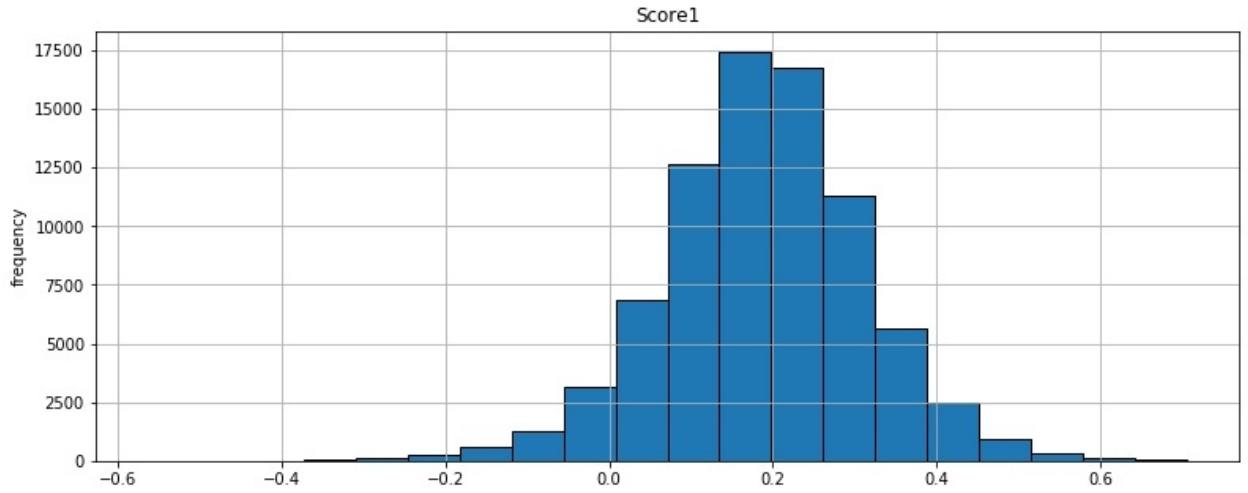
Rest has enough unique values to be considered as continuous variables.

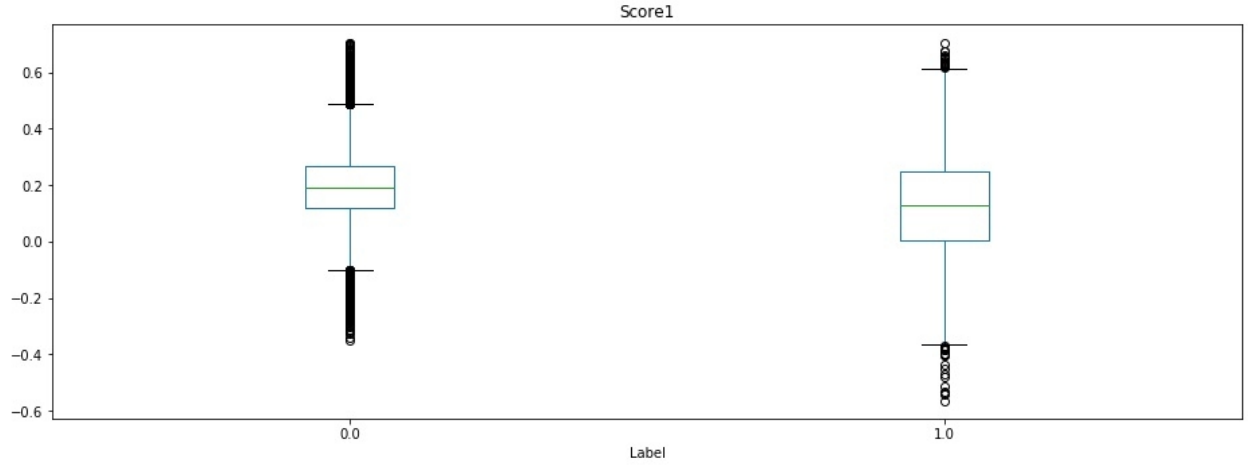

Expense
frequency
14000
12000
10000
8000
6000
4000
2000
0
1200
1400
1600
1800
2000
2200

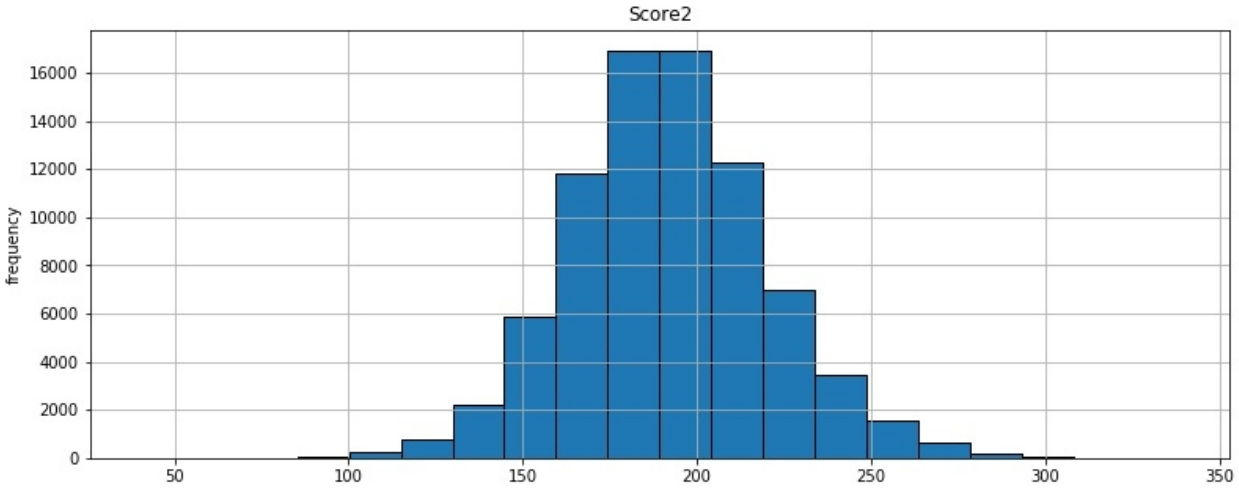

Expense
2200
2000
1800
1600
1400
1200
0.0
1.0
Label

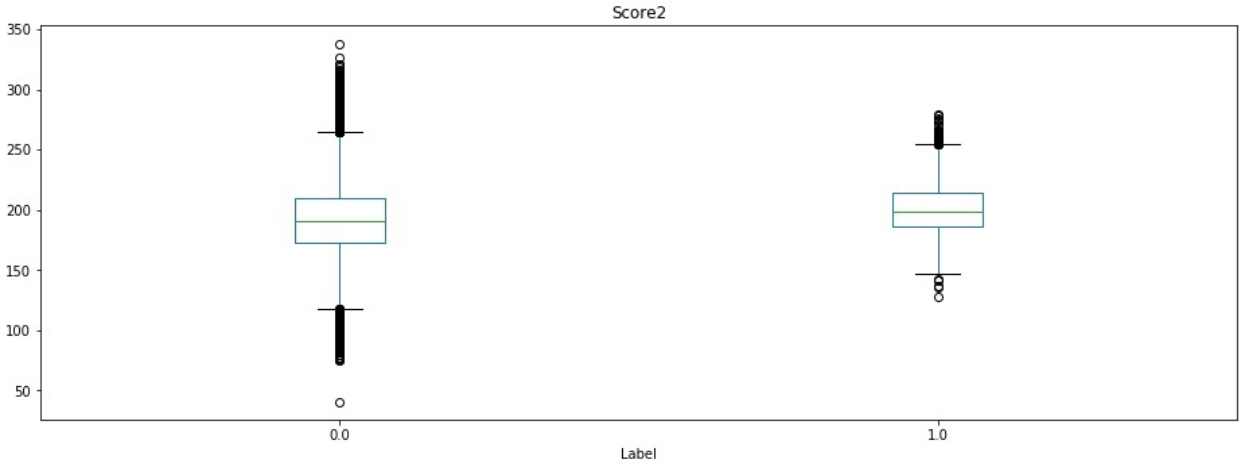

Income
frequency
14000
12000
10000
8000
6000
4000
2000
0
12000
14000
16000
18000
20000

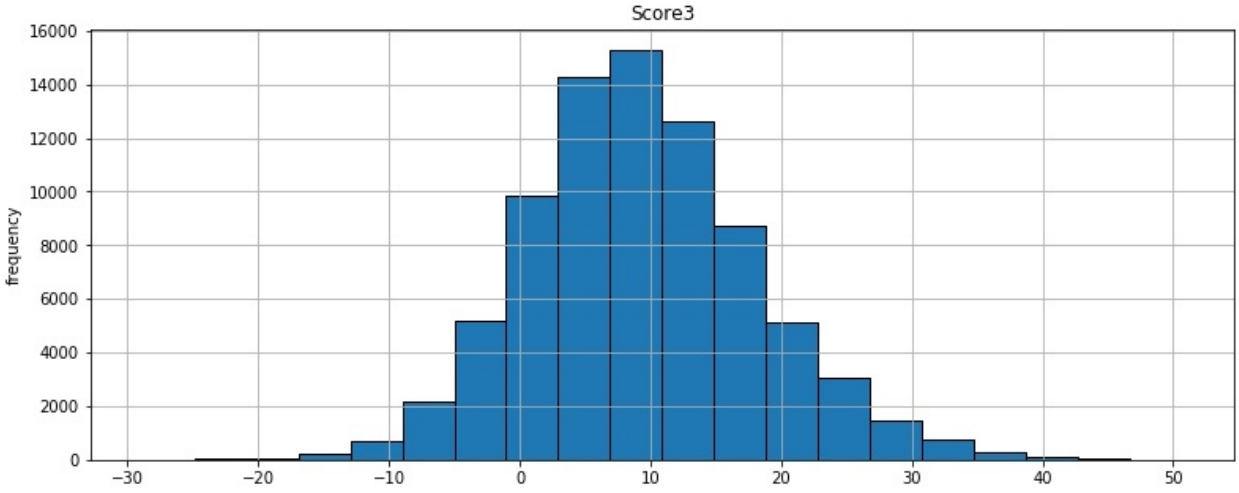

Income
20000
18000
16000
14000
12000
0.0
1.0
Label

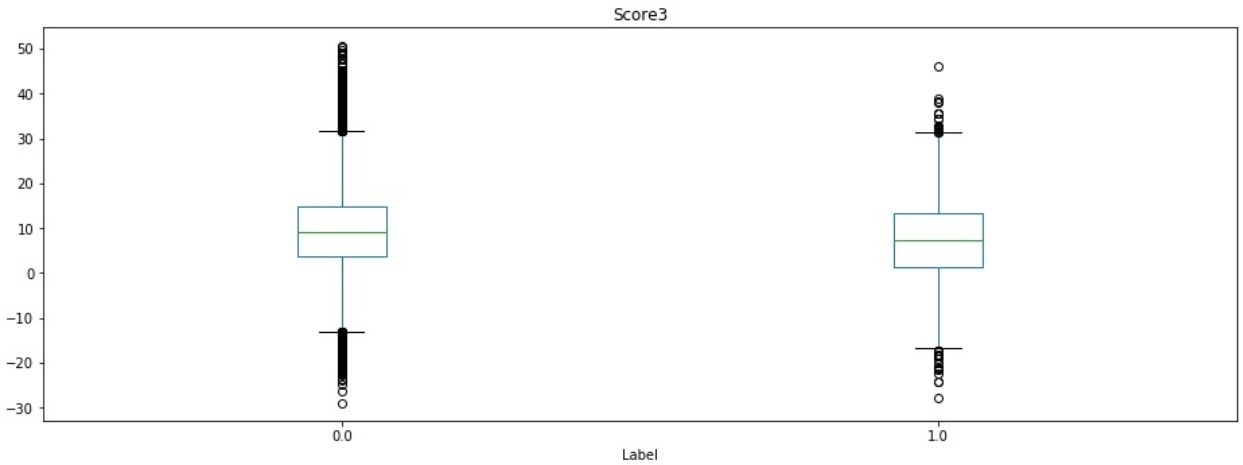

Score1
17500
15000
12500
10000
7500
5000
2500
0
frequency
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6

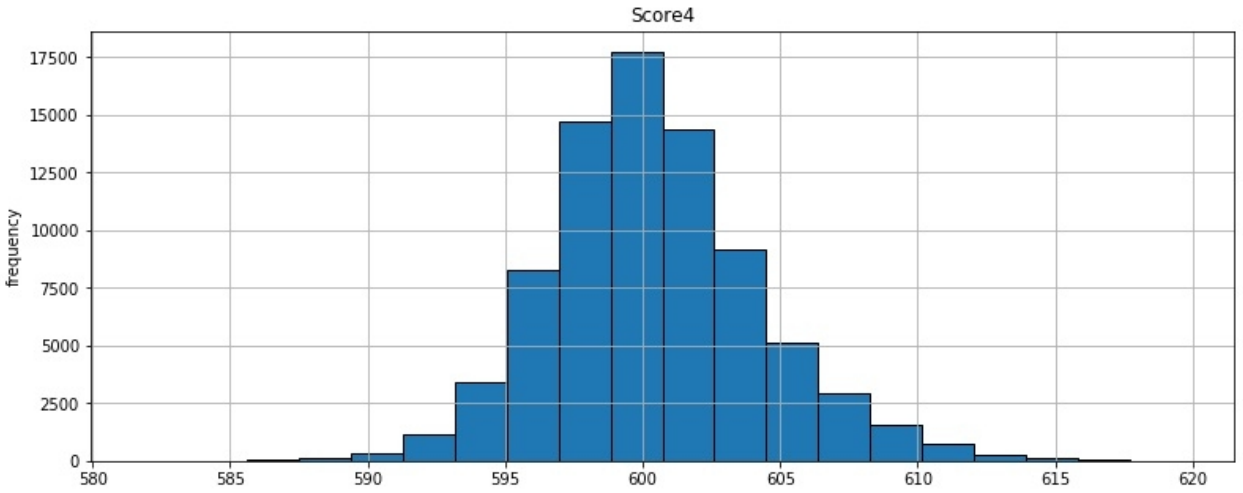

Score1
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0.0
1.0
Label

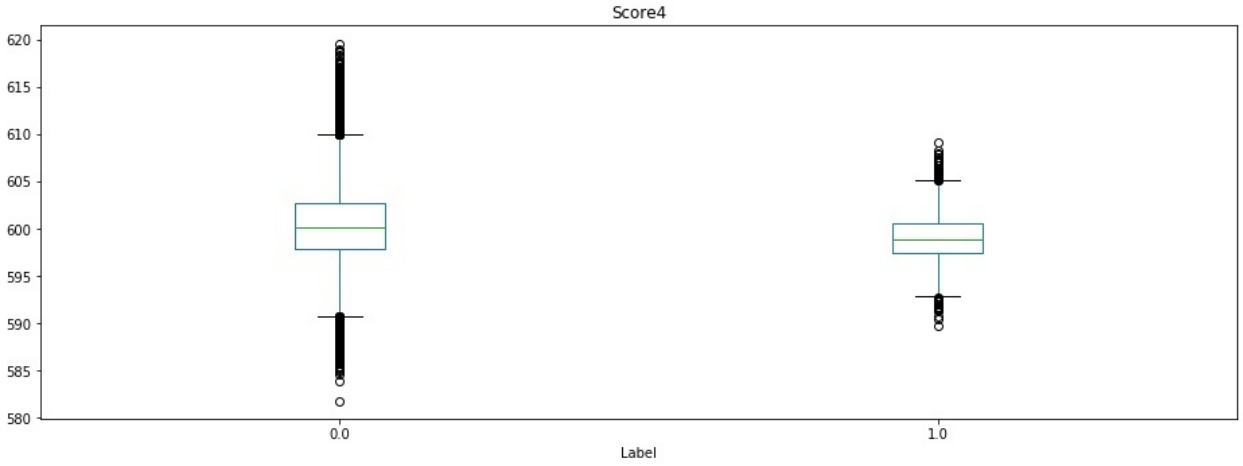

Score2
frequency
16000
14000
12000
10000
8000
6000
4000
2000
0
50
100
150
200
250
300
350

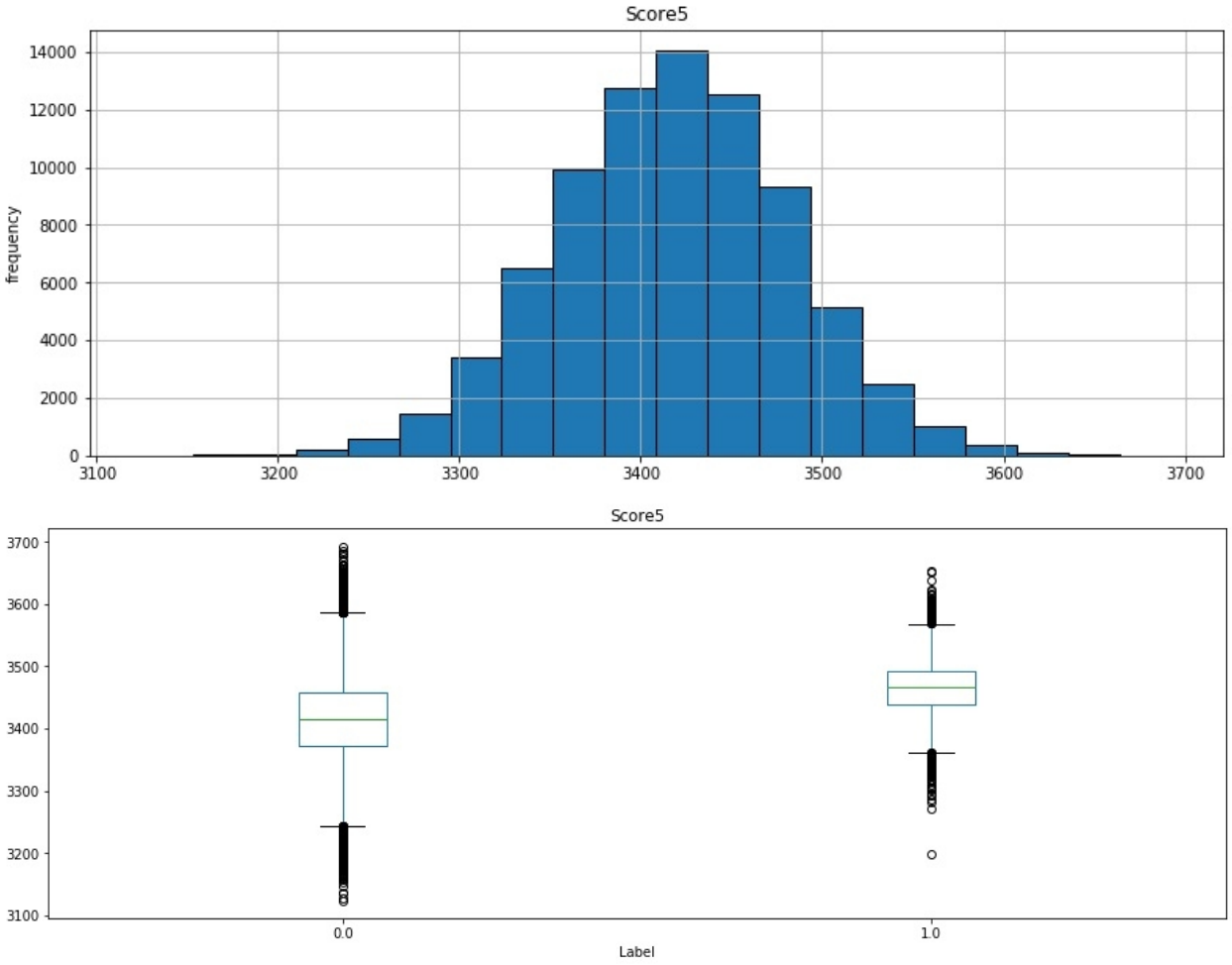

Score2
350
300
250
200
150
100
50
0.0
1.0
Label


Score3
frequency
16000
14000
12000
10000
8000
6000
4000
2000
0
−30
−20
−10
0
10
20
30
40
50


Score3
50
40
30
20
10
0
−10
−20
−30
0.0
1.0
Label


Score4
17500
15000
12500
10000
7500
5000
2500
0
frequency
580
585
590
595
600
605
610
615
620

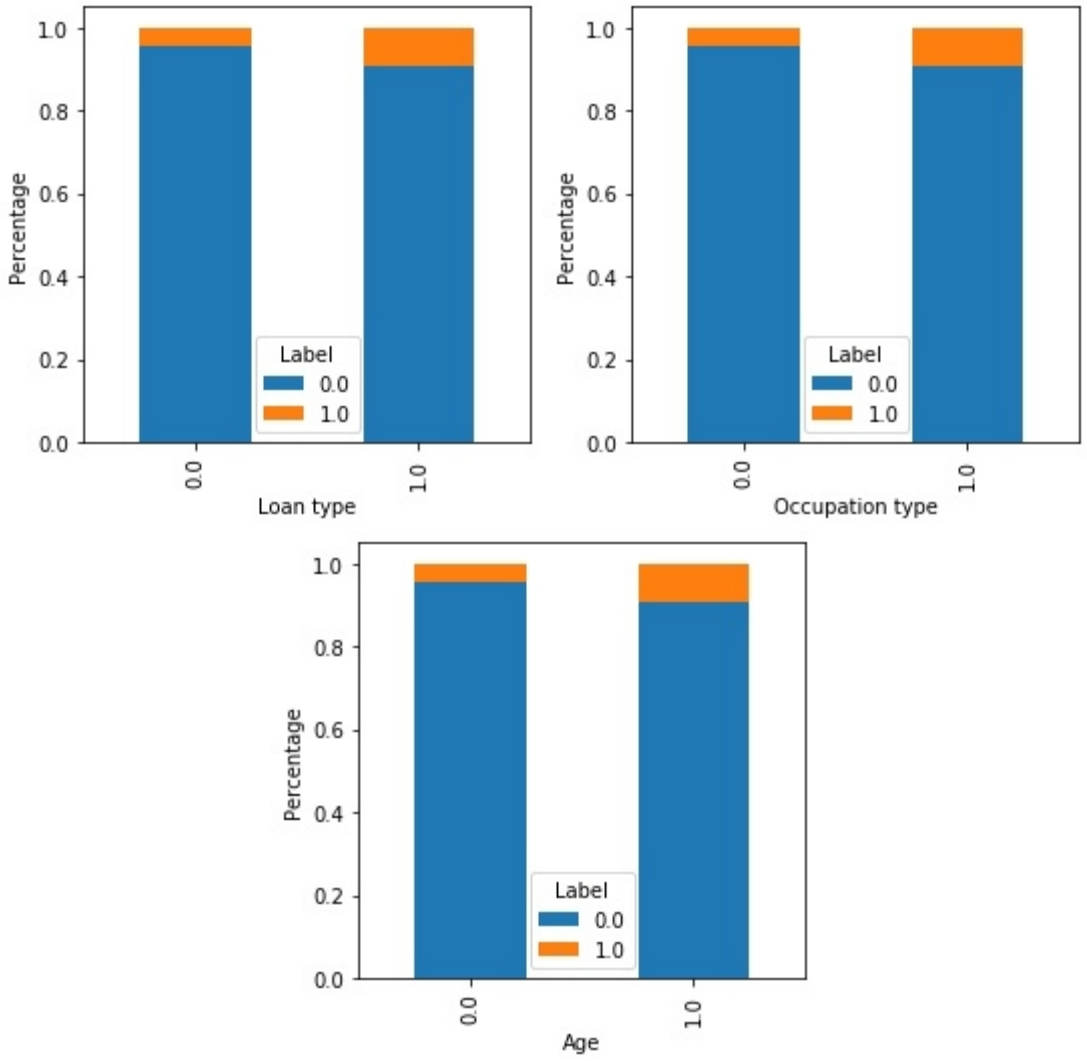

Score4
620
615
610
605
600
595
590
585
580
0.0
1.0
Label



From the histogram plot, we can see that all the continuous features usually are distributed and are not skewed with respect to the mean value.

In the box plot, we can see many points lying outside the cutoff hence we will not consider them as outliers except for the ones farthest among them.

E.g. in the score 5 box plot, score5= 3200 and Label=1 is an outlier. We can remove that data point.

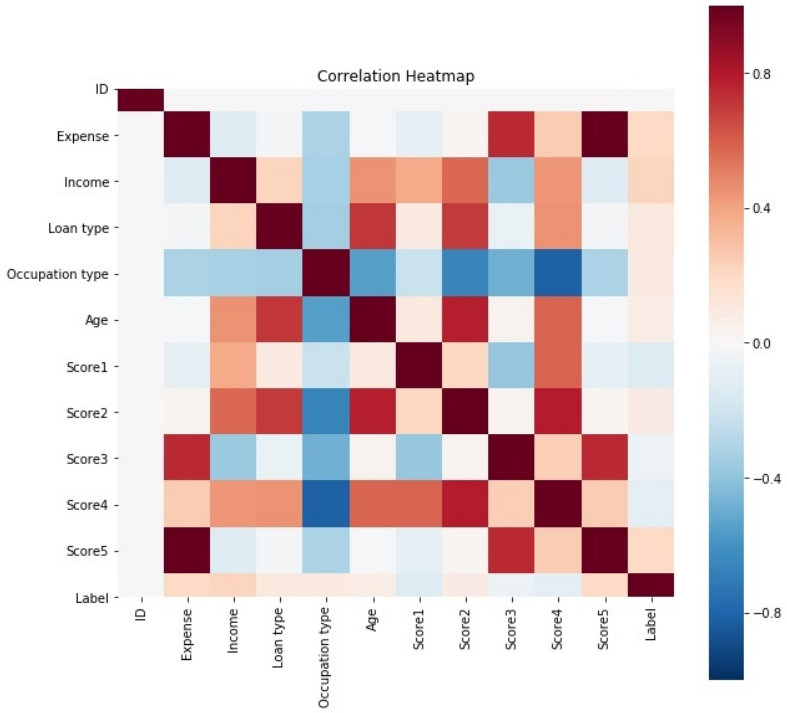


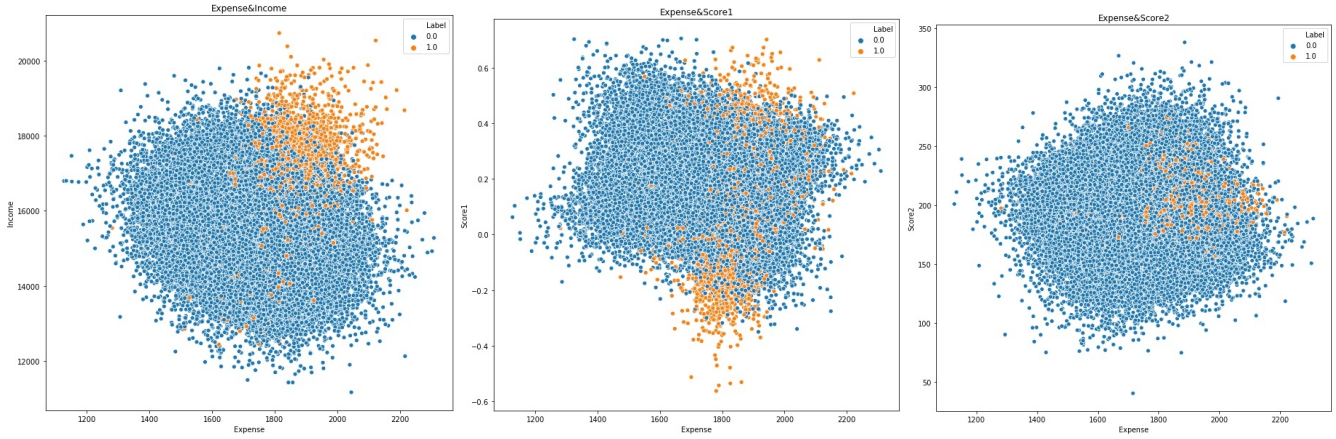
loan type 1 has more proportion for label 1

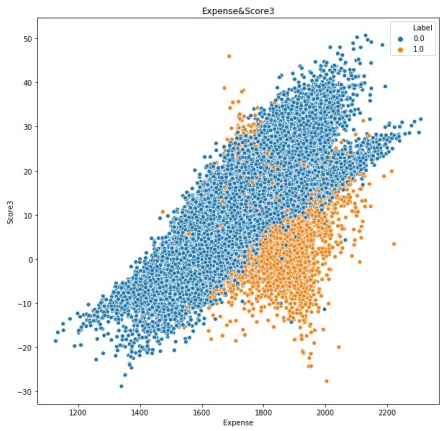
The proportion of occupation type 2 is higher for the label 1

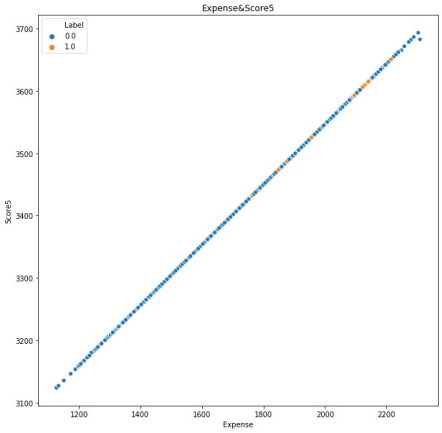
we can consider age, occupation type, and loan type for model training as they show an increased chance for a particular class happening for specified categorical values.

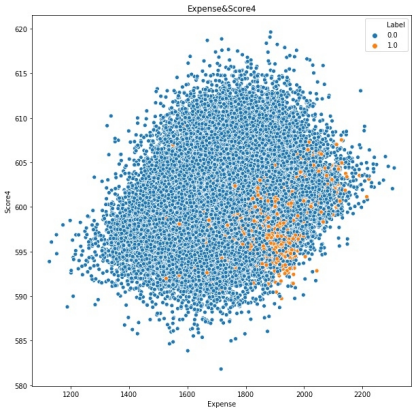
Here, we will see the correlation between features so that redundant features can be avoided. This will help solve multicollinearity and overfitting issues to an extent. We have shown correlations through a correlation heatmap, as shown below.

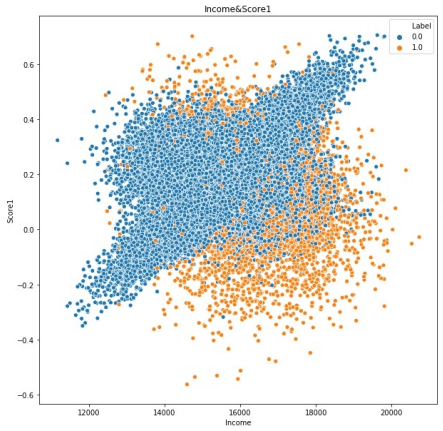




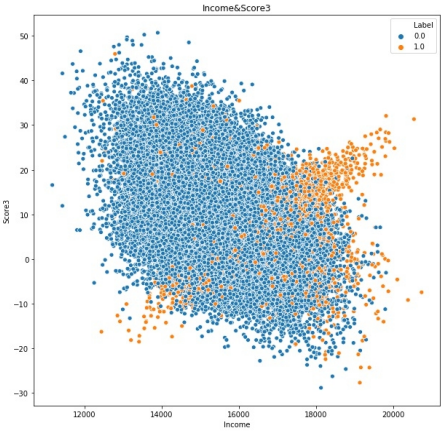

Expense&Score3
Label
0.0
1.0
Score3
Expense

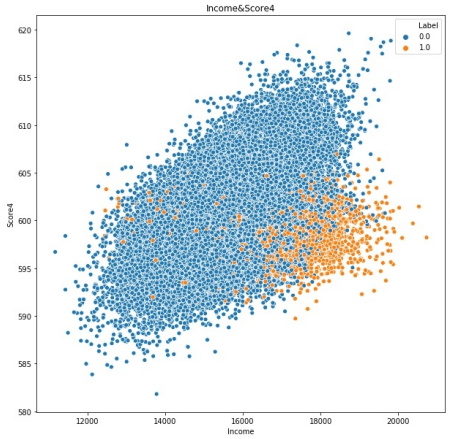

Expense&Score5
Label
0.0
1.0
Score5
Expense

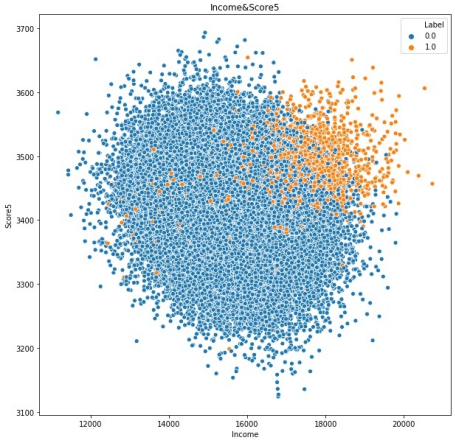

Expense&Score4
Label
0.0
1.0
Score4
Expense

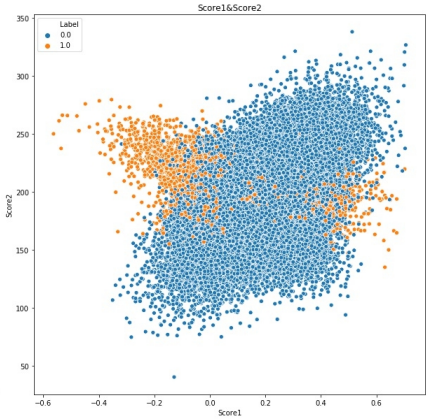

Income&Score1
Label
0.0
1.0
Score1
Income

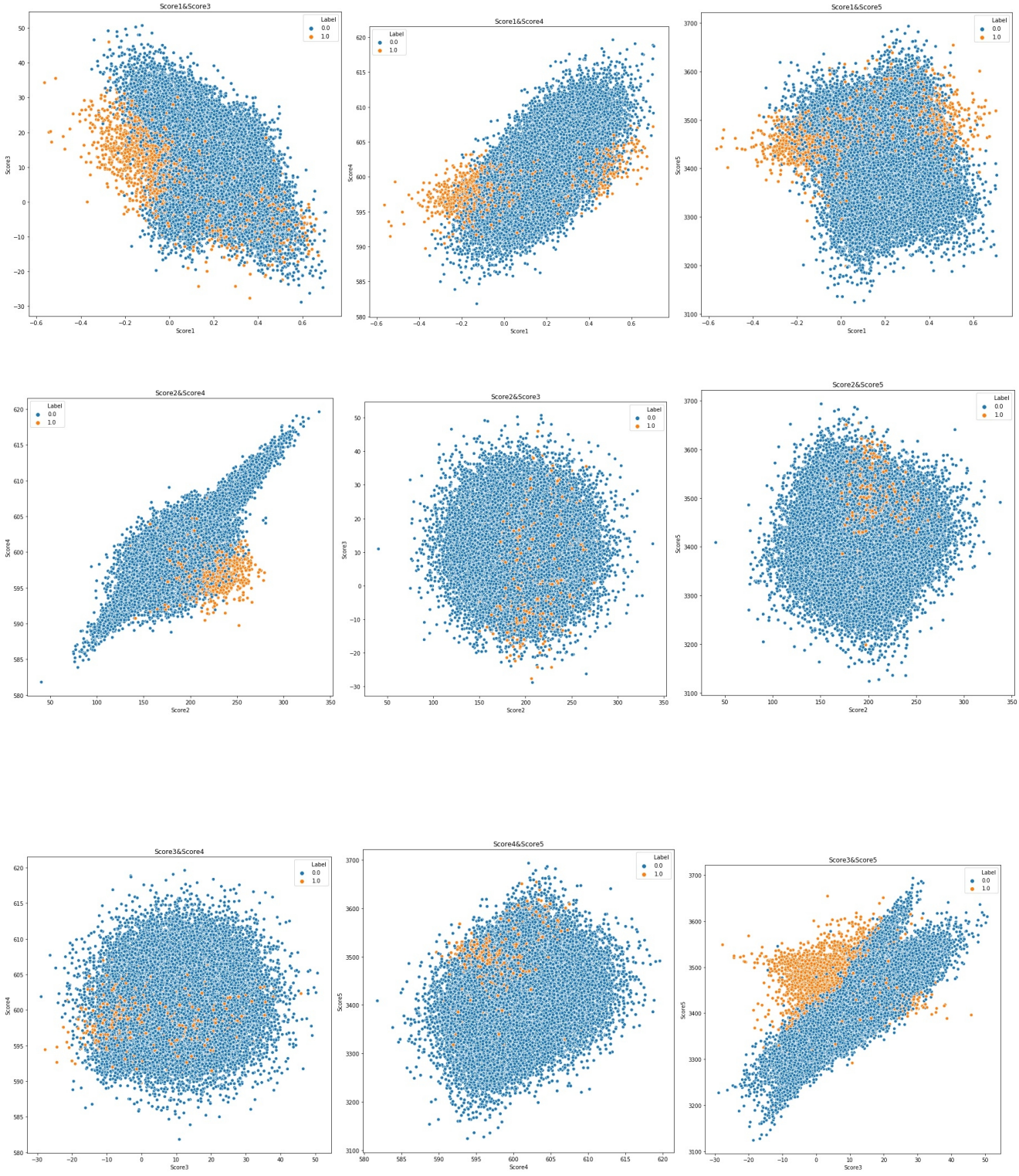

Income&Score2
Label
0.0
1.0
Score2
Income


Income&Score3
Label
0.0
1.0
Score3
Income

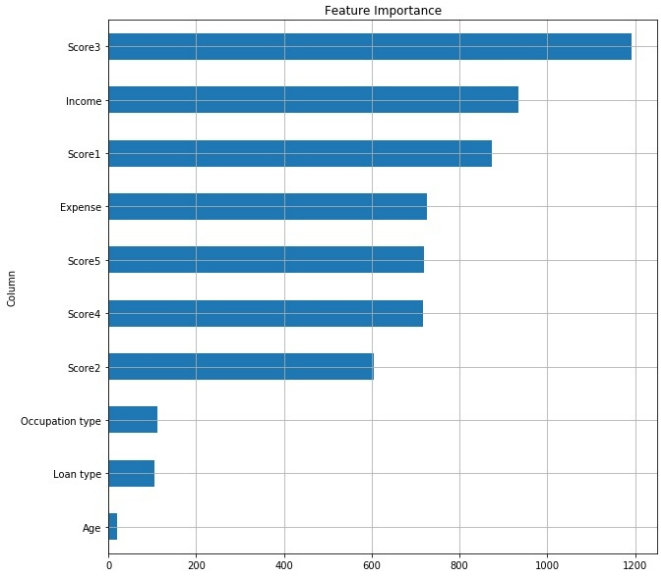

Income&Score4
Label
0.0
1.0
Score4
Income


Income&Score5
Label
0.0
1.0
Score5
Income

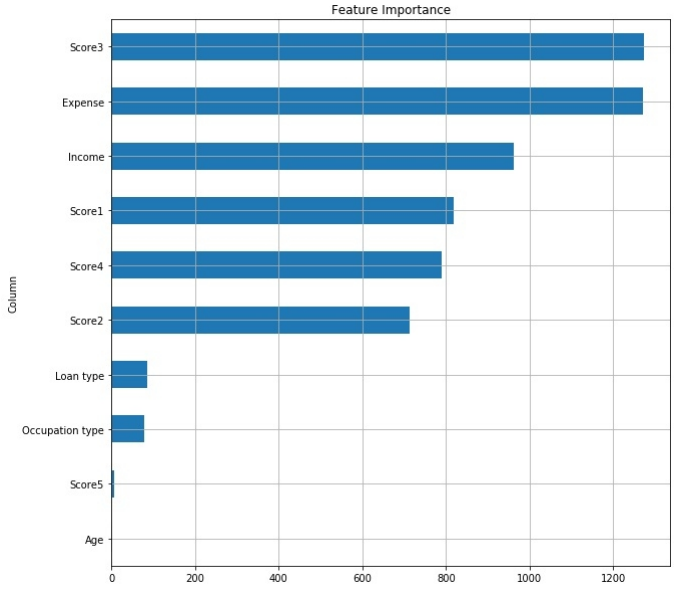

Score1&Score2
Label
0.0
1.0
Score2
Score1



From the scatter plots, we can see the correlation between score5 and expense; hence we can drop one among these two features.



Feature importance before imputation



Feature importance after imputation

we fitted a LightGBM model on the dataset and analyzed feature importances of all features before and after imputation.

Data Cleaning

Missing Value Imputation

A Variant of MICE (Multiple Imputation by Chained Equations):

This is an iterative algorithm that uses a regression framework to predict the missing values of (numerical) features.

Step 1:

Impute the missing values in each variable with temporary "placeholder" values such as the mean or median

Step 2:

Set back to missing the “place holder” imputations for one of the variables. This way, the current data copy contains missing values for only this variable, but not for others.

Step 3: Regress the chosen variable on all other (independent) variables via a linear regression model. Now make predictions for the missing values for the chosen variable.

Step 4: Repeat Steps 2-3 separately for each variable that has missing data.

Step 5: Repeat steps 2-4 multiple rounds till the predictions of the value for missing values do not change too much.

MissForest Imputation:

This is a machine learning based imputation technique. It uses random forest algorithm for the task. It is based on an iterative approach, and at each iteration, the generated predictions are better.

Benefits:

- Doesn’t require extensive data preparation — as a Random forest algorithm can determine which features are important
- Doesn’t require any tuning — like K in K-Nearest Neighbors
- Doesn’t care about categorical data types — Random forest knows how to handle them

Algorithm

Step 1: The missing values are filled in using median/mode imputation.

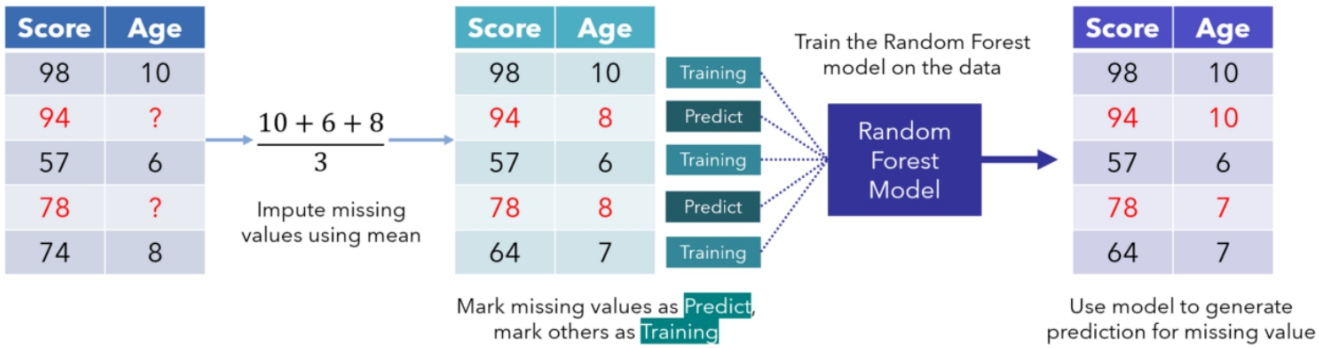
Step 2: Mark the missing values as ‘Predict’ and the others as training rows, which are fed into a Random Forest model trained to predict

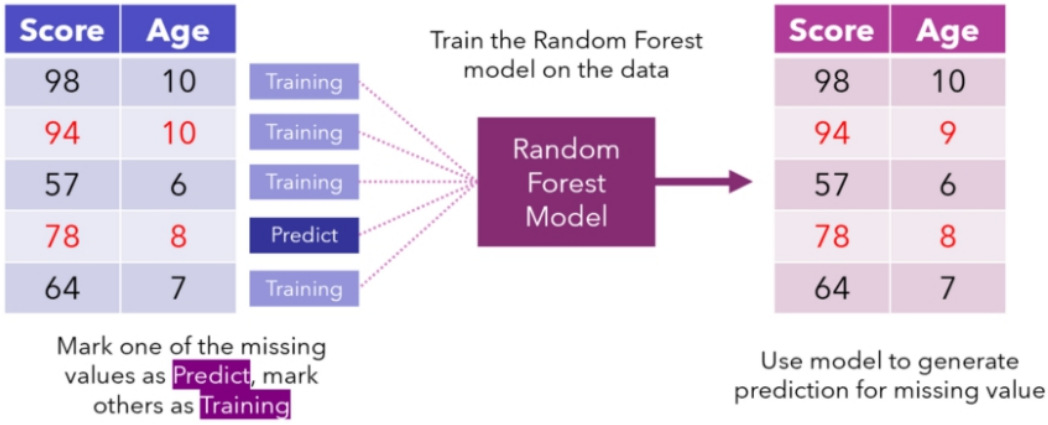
Step 3: Regress the chosen variable on all other (independent) variables via a linear regression model. Now make predictions for the missing values for the chosen variable.

Step 4: This process of looping through missing data points repeats several times, each iteration improving on better and better data.

Step 5: Iterations continue until some stopping criteria is met or after a certain number of iterations has elapsed.

Working of Missing forest on a dummy data set with 2 columns.





The **MissForest imputed data set was used for further feature engineering and modeling** as it dealt with categorical features much better than MICE did.

Feature Engineering

All the indices with null labels are dropped. Features are divided into categorical and numerical features for feature engineering. The numerical (continuous) features are used to create more features.

Polynomial features

New features are engineered using polynomial feature generation functions. Features of degree two are used in this case and it's generated by multiplying each numerical feature with every other one. 21 continuous features are obtained from 6 continuous features.

The categorical and continuous features obtained from polynomial feature generation are combined to obtain the feature set that is to be used for feature selection

Feature Selection

Binning continuous variable

Our target is categorical and we are relying mostly on decision tree models. These models find decision rules to make predictions by computing thresholds for minimizing entropy after split for continuous data. By reducing the number of unique values, we make the splitting procedure easier and more semantically interpretable

For this, we discretize all continuous variables by making quantile cuts on each continuous column. This will help us obtain as many categorical variables. Using these, we will test the performance of CatBoost.

Feature Selection

All the features with high correlation are dropped, eg ‘Score 5’ in the beginning and ‘Score3 * Score4’, ‘ Expense * Score 3 ‘ and ‘Score 1’ etc after polynomial feature generation.

Lgbm feature importance is used to select the most important features from the remaining features

Model Selection

The Data is imputed using Random Forest and it is cleaned using the function ‘impute_functions.py’. The data is now split for training and validation in an 80:20 manner. Due to the high imbalance in the train data (‘Class 0’: ‘Class 1’:: 56812: 4065) after cleaning, we have used SMOTE to resample the data to a 60:40 ratio (Note: The SMOTE resampling is only applied on the train data to avoid incorrect accuracy while testing)(Note: 60:40 ratio was chosen to retain the major class as is as our score metric is weighted f1-score). The Occupation_type parameter is then ‘one-hot encoded’ to use on decision-tree based classifiers.

A variety of models are considered and added to the table. The next step was incorporating the feature selection results. Using the function ‘feature_engineer.py’, we created and added additional features to the data. (The data used is the data after SMOTE as we wanted to avoid performing SMOTE on correlated features). After one-hot encoding the Occupation_type, the data is now once again considered for various models and ensembles. The results have been tabulated as

follows:

	Model name	Feature_Engineered_data(Y/N)	F1-score(weighted)	Accuracy
0	Extra Trees	N	0.982834	0.982917
1	Catboost	N	0.983484	0.983377
2	XGBoost	N	0.981908	0.981800
3	Adaboost	N	0.929381	0.917017
4	Extra Trees	Y	0.981901	0.981997
5	Catboost	Y	0.983268	0.983180
6	XGBoost	Y	0.982178	0.982063
7	Adaboost	Y	0.935067	0.924836
8	LightGBM	Y	0.975043	0.974047
9	Soft Voting Ensemble: ET & Catboost	Y	0.983914	0.983903
10	Soft Voting Ensemble: ET, Catboost & XGB	Y	0.983290	0.983246
11	Hard Voting Ensemble: ET, Catboost and XGB	Y	0.983475	0.983443
12	Stacking Ensemble: ET & catboost	Y	0.983520	0.983509
13	Stacking Ensemble: ET, catboost & XGB	Y	0.984077	0.984100

For the model Extra trees, the default parameters have been used as they were seen to produce the best results. For catboost, we set the parameter ‘depth’ = 10. For XGBoost, we set the parameters 'max_depth' = 17 and 'min_child_weight' = 1. We have only considered grid search for these three models based on their high performance without optimization. The grid search was done in a step by step manner in which each set of parameters was varied until a minimum point is found. We used GridsearchCV to perform the search. The code is attached at the end of the model selection notebook.

From the table, the highest F1-score belongs to the Stacking ensemble using Extra-trees, Catboost and XGBoost. This is the selected model and its classification report is as follows:

```
Stacking Classifier with ET, catboost and XGBoost
              precision    recall  f1-score   support

         0.0      0.991     0.992     0.992     14252
         1.0      0.877     0.872     0.875       968

    accuracy                          0.984     15220
   macro avg      0.934     0.932     0.933     15220
weighted avg      0.984     0.984     0.984     15220
```

Performance with Deep Learning

We select the dataset imputed using the RandomForest technique for this task. The dataset itself contains a majority to minority class ratio of 93.44%:6.56%.

CTGAN for oversampling

A Generative Adversarial Network (GAN) is trained on the original dataset to yield synthetic data in order to reduce the class imbalance.

CTGAN is a collection of Deep Learning based Synthetic Data Generators for single table data, which are able to learn from real data and generate synthetic clones with high fidelity.

A total of ~8 lakh rows of synthetic data is generated from the CTGAN. Now, we split the data into train and test splits in an 80:20 ratio. Since class 1 is in the minority, we randomly sample entries from the GAN generated synthetic data to bring the imbalance down to 60.00%:40.00%. A 1:1 ratio is avoided to keep some original bias intact. Along with GAN, we also tried upsampling the data using the SMOTE technique, which was done in non deep learning models as well.

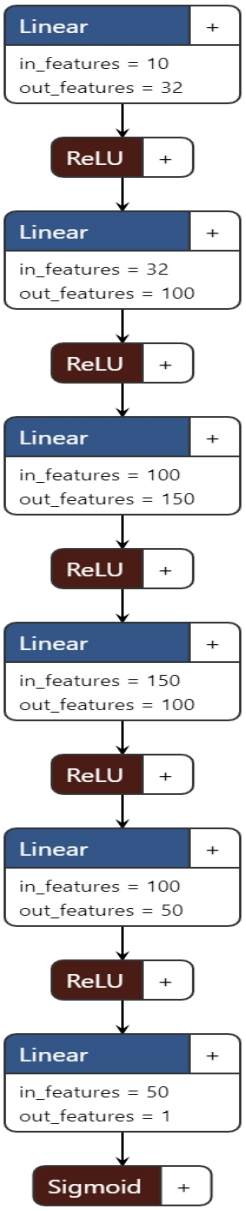
The categorical variables, namely "Loan type", "Occupation type" and "Age" are kept intact as they are pre-encoded. The numerical variables are scaled down using standard scaling i.e. $\mathbf{(X-\mu)/\sigma}$.

We train a deep learning model with Binary Cross-Entropy loss (binary classification) and using Adam optimizer using the PyTorch library. The model parameters and results are almost the same for both types of upsampling methods and are shown below.

Batch size	No. of epochs	Learning rate	No. of layers
32	23	0.001	6

Validation accuracy	0.991
Weighted F1 score	0.99

Model Architecture:



Results

Just looking at the F1 scores of all models, we could conclude that the Artificial Neural Network gives us the best F1 score among all models. But, there is a flaw. The CTGAN (to create synthetic data) was trained on the entire train + validation dataset. This might have caused the GAN to use information from the validation rows to generate synthetic data which was later used to upsample the training dataset to reduce its class imbalance. Due to this

suspected leakage of validation data into the train, we deem the validation results unreliable. **The deep learning model with SMOTE resampling gives the best results.**

For our final prediction result we **compare the probabilities of each of the models** (ML and DL) for Class 1, and choose the prediction with the higher confidence, i.e. if ML result gave a 40% confidence to Class 1 and DL result gave a 90% confidence to Class 1, we chose the label as Class 1 based on high confidence of the DL result.