INDIAN INSTITUTE OF TECHNOLOGY MADRAS

INTRODUCTION TO DATA ANALYTICS
MS4610
Project Report

Shrisudhan (ME17B122)
Aziz Kanchwala (NA17B011)
Jeswant Krishna K (AE17B030)
Avinash Bagali (AE17B110)
Dhruv Shah (AE17B107)

January 6, 2021

1 Introduction

1.1 Problem Statement

Use the data provided and the information about the features to build a machine learning model to predict whether a loan will go default or not, and to understand which of the features are important and helpful in the prediction.

1.2 Description of the data

1.2.1 Training data-set X

The training data-set X is presented as an 80000 x 11 matrix containing 80000 data points each described by 11 features. The data-set contains a lot of *NaN* values for features which has to be dealt with using appropriate imputation techniques.

1.2.2 Training data-set Y

The training data-set Y contains labels '0' and '1' which correspond to non-default and default status respectively. This data-set too contains *NaN* values and hence also requires imputation.

1.2.3 Test data-set X

A data-set with no *NaN* values which is not expected to have been a random sample of the entire data-set.

1.3 Approach

As the given train data-set is highly imbalanced, accuracy is not the best measure of a classification algorithm's performance. The f1 score which is the harmonic mean of precision and recall should be considered for evaluation. We want our f1 score to be as high as possible while at the same time ensuring that our precision is high as the potential loss incurred by identifying a defaulter as a non-defaulter exceeds that of identifying a non defaulter as a defaulter. Thus we have to train our classification algorithms with the best class weights as obtained through techniques such as grid search.

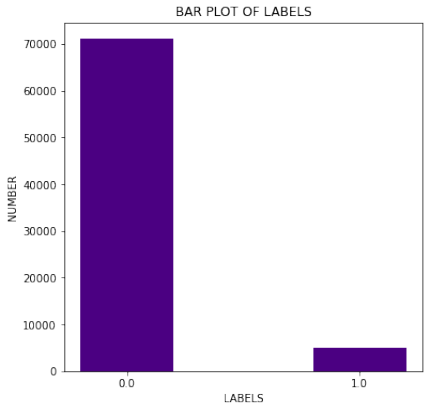


Figure 1: Plot of Labels showing the imbalance in data

2 Data-set

2.1 Data-set Analysis & Visualisation

2.1.1 Data-set Analysis

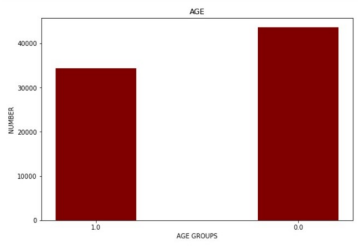
- Firstly, the variable **ID** is removed from training set x as well as training set y because it is unique to every entry and has no bearing on the result.
- From the three categorical predictors, **Age** is label encoded. This is done because age is assigned value 0 if the value is less than 50 and 1 if the age is more than 50 which suggests a sense of ordering i.e $1 > 0$ for the given data-set
- **Loan type** and **Occupation type** are encoded as one hot variables as there is no sort of natural ordering present in the values for these predictors ($A < B$ or $X > Y$ and less than Z) do not make sense.
- The mean and median are calculated for the **continuous variables** ignoring the NaN values and the results obtained are as follows:

Variable	Mean	Median
Expense	1733.993769	1736.276720
Income	15641.112448	15624.259290
Score 1	0.187617	0.189877
Score 2	192.065584	191.056193
Score 3	9.365450	8.883862
Score 4	600.397742	600.095436
Score 5	3417.740403	3418.793524

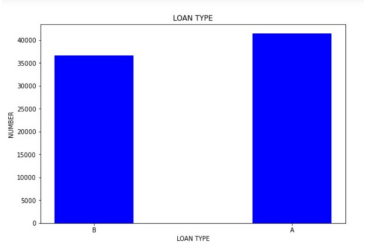
Table 1: Comparison on Mean and Median for continuous predictors

2.1.2 Visualising the raw data (Bar graphs and histograms)

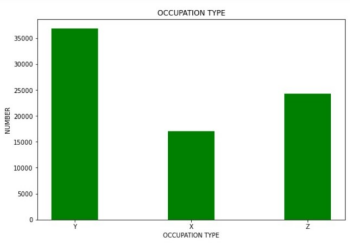
- The first three figures represent the bar graphs for the categorical predictors in the data-set. These give a general idea of the relative frequencies of those variables in our training set. (NaN values are ignored for convenience) .
- The next seven figures (all in blue) represent the histograms of all the numerical/continuous features in the training data-set.



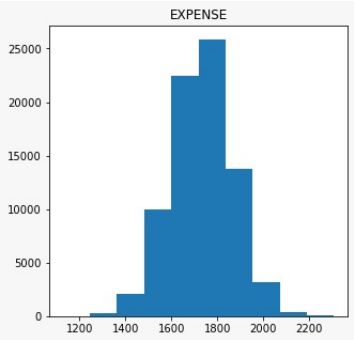
Age



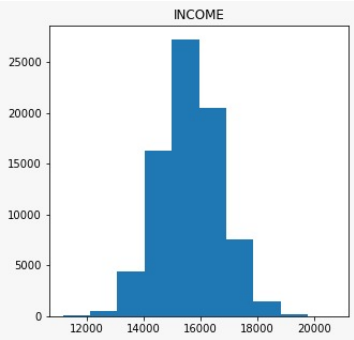
Loan Type



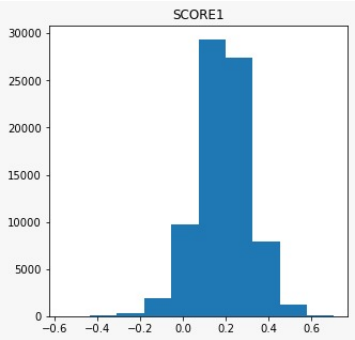
Occupation Type



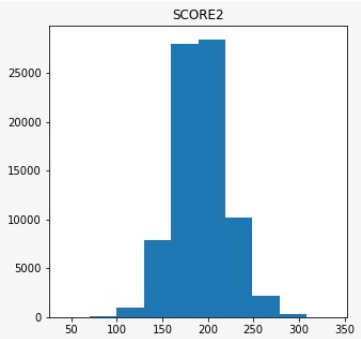
Expense



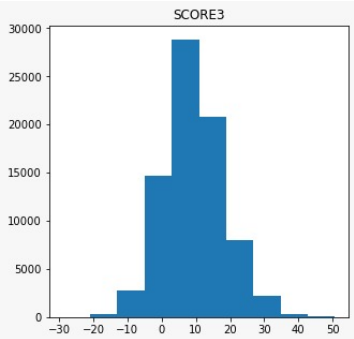
Income



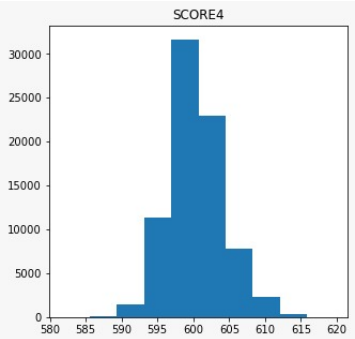
Score 1



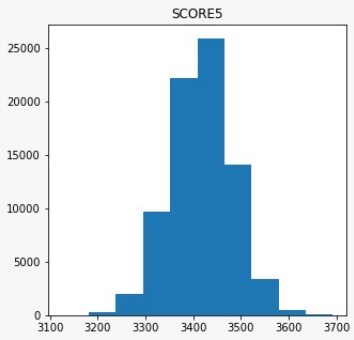
Score 2



Score 3



Score 4



Score 5

2.1.3 Box plots for numerical features

A boxplot is a standardized way of displaying the distribution of data based on a five number summary ("minimum", first quartile (Q1), median, third quartile (Q3), and "maximum"). It can tell you about your outliers and what their values are. It can also tell you if your data is symmetrical, how tightly your data is grouped, and if and how your data is skewed.

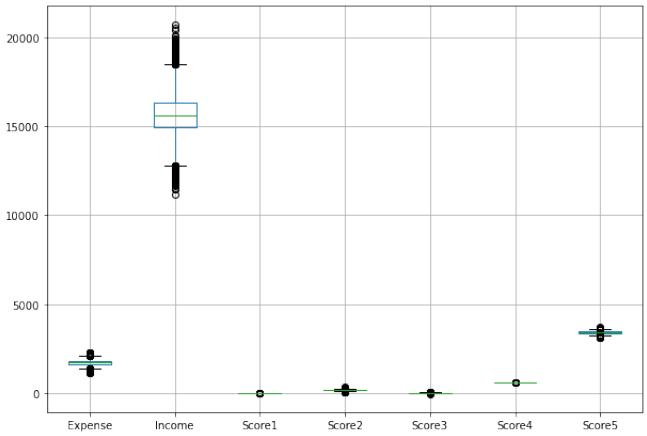
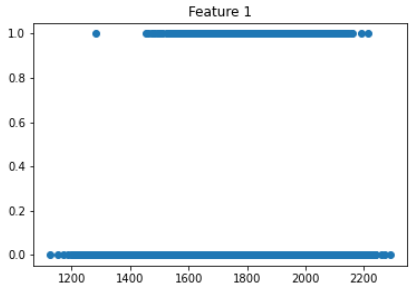
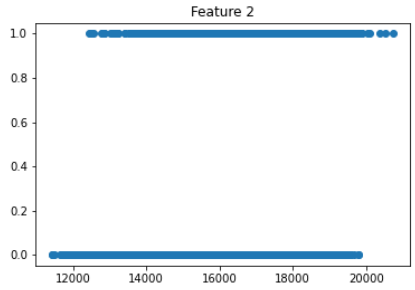


Figure 6: Box plot for numerical Features (Original Dataset)

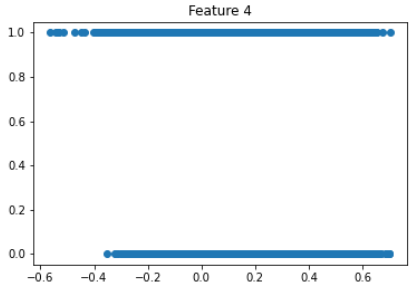
2.1.4 Univariate Visualisation of cleaned data



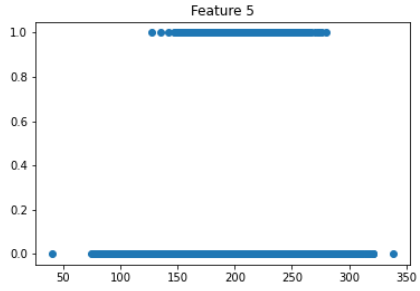
Expense



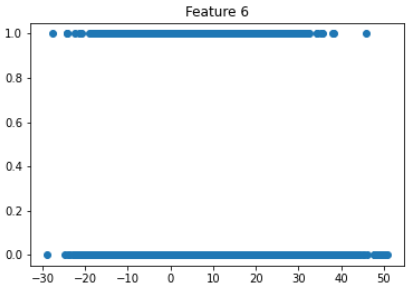
Income



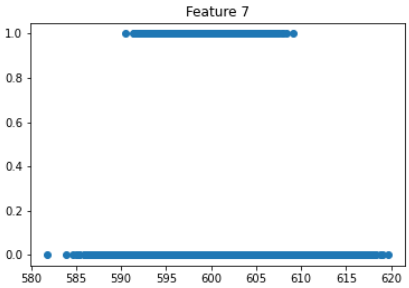
Score 1



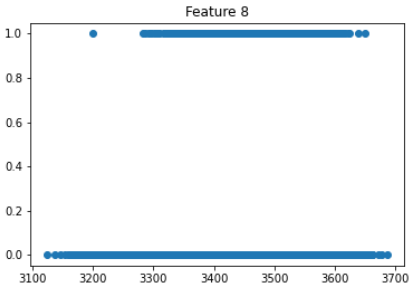
Score 2



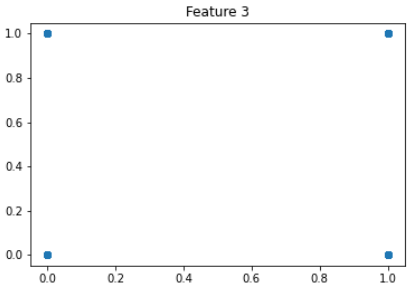
Score 3



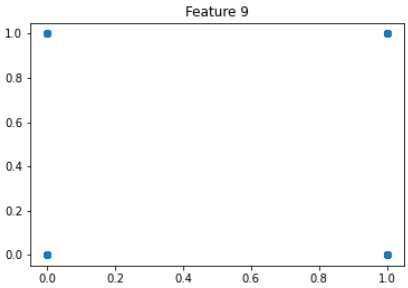
Score 4



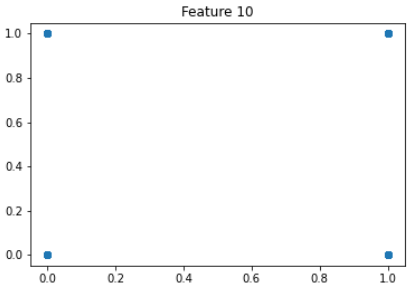
Score 5



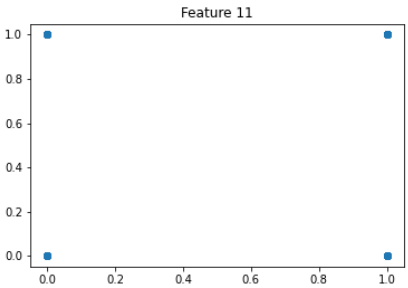
Age



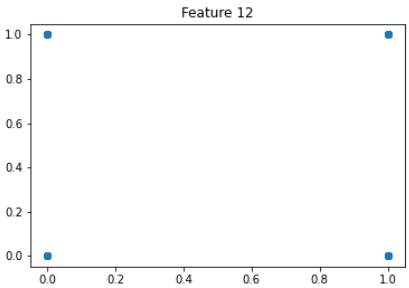
Loan Type A



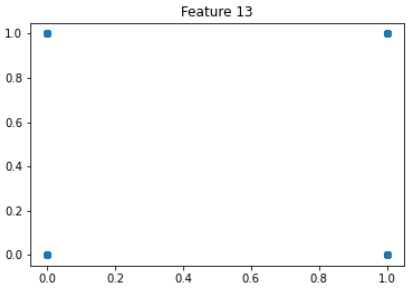
Loan Type B



Occupation Type X



Occupation Type Y



Occupation Type Z

In total, there are 10 features in the X_{train} and the label feature from y_{train}. We have also created dummy variables for *Loan type* and *Occupation type*. In previous figures, we can see how the individual features are spread over their range and the corresponding y values. For some features, we see there are a few entries which look isolated and could be interpreted as anomalies w.r.t to that feature.

2.2 Correlation between different Predictors

2.2.1 Correlation between two continuous predictors

Pearson's Correlation Coefficient is the test statistic that measures the statistical relationship or association between two continuous variables. Based on the method of co-variance, it measures not only the magnitude of the association but also the direction of the relationship.

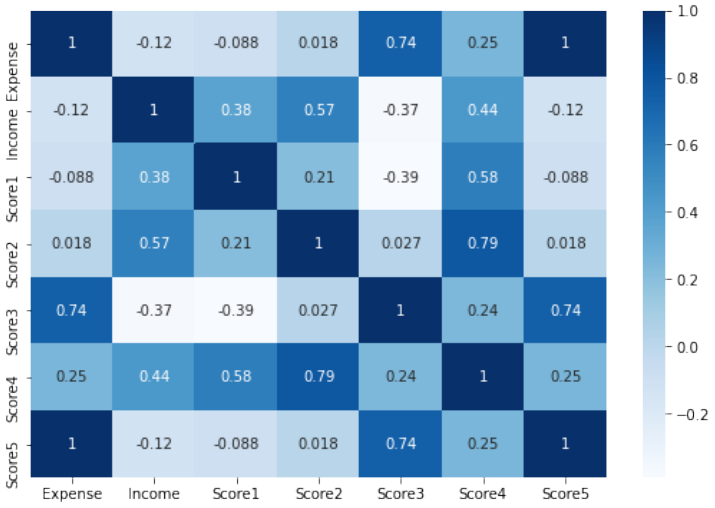


Figure 14: Correlation Matrix

2.2.2 Correlation between a continuous predictor and a categorical predictor

- **Independent t-test for two samples**
  The independent t-test, also called the two sample t-test, independent-samples t-test or student's t-test, is an inferential statistical test that determines whether there is a statistically significant difference between the means in two unrelated

groups. For our training set, we group the continuous variables by the classes **Age** and **Loan type** and perform the t test. The test statistic and p-values are provided in the tables below:

Variable	*Expense*	*Income*	*Score 1*	*Score 2*	*Score 3*	*Score 4*	*Score 5*
t-Statistic	-2.720660738	140.715818	28.1758020	353.46787	8.939206	202.05778	-2.719937
p-value	0.00651656	0.0	8.231e-174	0.0	4.00068e-19	0.0	0.0065308

Table 2: Variables grouped based on **Age**

Variable	*Expense*	*Income*	*Score 1*	*Score 2*	*Score 3*	*Score 4*	*Score 5*
t-Statistic	5.80863147	-63.6526	-24.5556	-273.7177	19.74729	-141.0824	5.809240
p-value	6.32e-09	0.0	1.16e-132	0.0	1.36e-86	0.0	6.299e-09

Table 3: Variables grouped based on **Loan type**

- **One way ANOVA for three or more groups**
  The one-way analysis of variance **(ANOVA)** is used to determine whether there are any statistically significant differences between the means of three or more independent (unrelated) groups.
  Note : We should not perform multiple pairwise t-tests for a categorical variable consisting of more than 3 states as doing so increases the probability of making a **Type 1** error.
  For our training set, the continuous variables are grouped based on the predictor **Occupation type** and the One way ANOVA is performed. The results are summarized as follows:

Variable	*Expense*	*Income*	*Score 1*	*Score 2*	*Score 3*	*Score 4*	*Score 5*
F-Statistic	1778.5383	1385.3996	1230.222	21121.493	4379.995	38773.61	1778.518
p-value	0.0	0.0	1.16e-132	0.0	0.0	0.0	0.0

Table 4: Variables grouped based on **Occupation type** for ANOVA

Based on the results obtained by t-tests and one way ANOVA, we conclude that the continuous predictors and categorical predictors are **not independent**.

2.2.3 Correlation between two categorical predictors

Chi-Square Test of Independence
The Chi-Square test of independence is used to determine if there is a significant relationship between two nominal (categorical) variables. The frequency of each category for one nominal variable is compared across the categories of the second nominal variable. The data is then displayed in a **contingency table** where each row represents a category for one variable and each column represents a category for the other variable.
The test statistic is given by,

$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i} \tag{1}$$

- **Age and Loan type**

States	A	B
0	37662	7015
1	4836	30487

Table 5: Observed Values for Age and Loan type

States	A	B
0	23733.539325	20943.460675
1	18764.460675	16558.539325

Table 6: Expected Values for Age and Loan type

- **Loan type and Occupation type**

States	X	Y	Z
A	3583	21679	17236
B	13880	16025	7597

Table 7: Observed Values for Loan type and Occupation type

States	X	Y	Z
A	9276.782	20029.307	13191.910
B	8186.218	17674.693	11641.090

Table 8: Expected Values for Loan type and Occupation type

- **Age and Occupation type**

States	X	Y	Z
0	1582	21569	21526
1	15881	16135	3307

Table 9: Observed Values for Age and Occupation type

States	X	Y	Z
0	9752.430	21056.270	13868.299
1	7710.569	16647.730	10964.701

Table 10: Expected Values for Age and Occupation type

Based on the values of the test-statistic and p-values for different combinations as shown in Table 11, we conclude that the categorical features are **not independent**.

Combinations	Test statistic	P-value
Age/Loan	39489.398	0.0
Age/Occupation	25107.53	0.0
Loan/Occupation	10389.375	0.0

Table 11: Test statistic and p-value for different combinations

2.2.4 Variance Inflation Factor

Variance Inflation Factor (VIF) is a measure of multi-collinearity in a dataset comprising of multiple variables. VIF for a variable is calculated by regressing a variable against all the other predictors in the dataset and calculating the R-squared metric. The VIF for that variable is then given by,

$$VIF_1 = \frac{1}{1 - R^2_{1.2...k}} \tag{2}$$

A general rule of thumb is to have VIF values less than 10 for a variable. If VIF values exceed 10, multi-collinearity in the dataset is accounted for by techniques such as naive feature elimination or more sophisticated techniques such as PCA.

Feature	Expense	Income	Score1	Score2	Score3	Score4	Score5
VIF	2.087912e+06	10.05946	935.1907	759.0623	836.687	2101.057	2.087971e+06

Table 12: VIF with all features present in the final imputed dataset

Feature	Expense	Income	Score1	Score2	Score3	Score4
VIF	57.641515	10.059404	935.163122	759.037294	836.661936	2100.996304

Table 13: VIF without **Score 5** in the final imputed dataset

Feature	Expense	Income	Score1	Score2	Score3
VIF	3.716035	3.293397	2.299127	5.283922	9.931846

Table 14: VIF without **Score 4** and **Score 5**

From the above tables, we conclude that feature **Expense** and **Score 5** are nearly identical in the final imputed data-set and hence removing one of them significantly reduces the multi-collinearity in the data. If we further carry on the process of removing predictors iteratively, we find that removing **Score 4** causes all the VIF values to drop below 10 for the continuous variables and hence is a good place to stop.

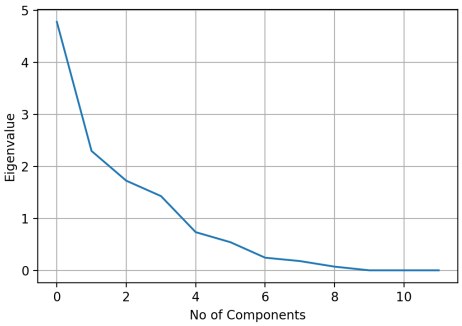
2.2.5 Principal Component Analysis

We use PCA to perform Dimensionality Reduction. Dimensionality reduction helps us to compress the data in an efficient way so that we capture almost all the variance in the data but with with fewer features, allowing for shorter training run times.
Following are the steps involved in choosing the number of Principal Components:

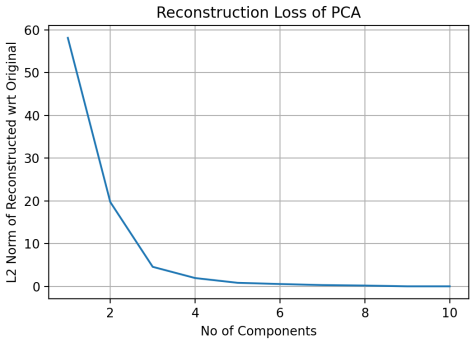
- The mean representation error when we represent a point in N dimensions with M principal components, is the sum of the (N-M) smallest eigenvalues of the Covariance matrix (**S**) associated with the data.
- We plot the values of the 13 Eigenvalues of S and we see that after the 5^{th} eigenvalue (marked as 4 on the plot), the eigenvalues are very small in comparison and we can infer that the first 5 principal components capture most of the variance in the data.
- We also plot the representation error vs the number of principal components taken, examining the 2 graphs, we choose 5 Principal components for our analysis.

#Components	1	2	3	4	5	6	7	8	9	10	11	12	13
Variance *(in %)*	0.353	0.691	0.811	0.886	0.937	0.976	0.988	0.995	0.999	0.999	1.000	1.000	1.000

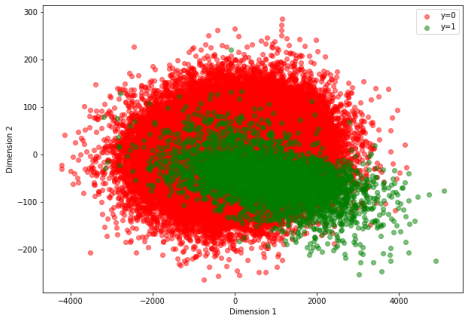
Table 15: No. of principal components v/s percentage of total variance



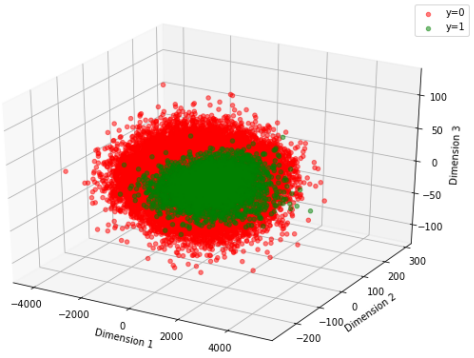
Eigenvalue magnitudes in descending order



PCA Decomposition Reconstruction Loss



n=2



n=3

Figure 16: Eigenvalue Decomposition (Red label shows y=0, Green label shows y=1)

2.3 Dealing with NaN values

For our dataset, there are many NaN values spread throughout all the input data, with many entries missing multiple feature values.

Feature	Expense	Income	Loan type	Occupation type	Age	Score1	Score2	Score3	Score4	Score5
#NaN values	2044	1955	2011	1859	2014	1940	2036	1955	1972	1998

Table 16: Number of invalid entries for different features

We can see in Table 16, each feature has about 2000 entries missing. Also, a total of 17719 entries had NaN values present in them.The easiest way to deal with this issue is to simply remove all the data entries that contain NaN values. But, removing those will create a substantial decrease in the amount of data (22%) we have and subsequently, a worse model overall.
There are multiple methods to impute data which would fill in for the missing values. We use the following methods and compared the results with the unimputed dataset. All of these are available from *sklearn.impute* module in Python.

- **KNN Imputation** : This method uses k-nearest neighbours algorithm to fill in the missing data. As it is a regression method, we need to convert all data entries to numerical values. The default distance measure is Euclidean which ignores entries with NaN values in them and does the regression accordingly, selecting the values according to the hyper-parameter k.

- **MICE imputation**: Multivariate Imputation with Chained Equations is a sequential regression algorithm and so, it also accepts only numerical values. It assumes that the missing data is missing at random and does not have dependence on non-observed quantities.
  Initially, a simple imputation is done on all variables and then one value is removed and then a regression model is built. The removed values are replaced with predictions and then this is repeated multiple times.

- **Miss Forest Imputation** : This method is similar to MICE for initialisation but instead of a regression fit, it uses random forests to predict missing values. This method is advantageous as it can handle both numeric as well as categorical data. It also assumes that the missing data is random, but it is comparatively immune to noisy data unlike KNN imputation. This method does take more time and is not interpretative because of random forests, but they are not major concerns for our cause.

MODEL	Unimputed	KNN Imputation	MICE Imputation	Miss Forest Imputation
Bayes Classifier	0.757	0.766	**0.771**	0.759
Logistic Regression	0.589	**0.613**	0.611	0.596
Decision Tree (entropy)	0.837	0.843	**0.849**	0.848
Decision Tree (Gini)	0.839	0.847	**0.849**	0.845
KNN (uniform weights)	0.875	0.888	**0.895**	0.893
KNN (distance weights)	0.877	0.887	**0.893**	0.892
SVC (rbf kernel)	0.903	0.910	**0.914**	0.909
Random Forest	0.879	0.886	**0.891**	**0.891**
XGboost Classifier	0.890	0.895	**0.901**	0.897

Table 17: Cross Validation F-1 Scores for different Imputation Methods

3 Models

3.1 Different Models Chosen

- Bayes classifier, Logistic Regression, Decision trees, KNN, Support Vector Machines, Random Forest, XGBoost Classifier were used on the training datasets.
- **K-fold cross-validation** was used to decide the final classifier to be used to make final predictions. K-fold cross validation is a trade-off between testing on only one validation set which suffers from the problem of **high bias** and leave one out cross validation (LOOCV) which suffers from the problem of **high variance**.
  Cross validation does not have a statistical background but is a computationally amenable technique to estimate an algorithm's performance.
- Firstly, the **f1 score** was calculated for all classification algorithms without accounting for the optimal hyper-parameters. Once the choice of the classifier was made (based on highest f1 score), the **optimal** model hyper-parameters were determined using the **grid search** technique.
- Table 17 has the cross validation f1 scores for all the models and all the training datasets. The details about those models are noted in the section 3.2.

Support Vector Machines (SVM) with **radial basis function** kernel was chosen to make final predictions as it gave the highest F1 score of all.

- **Rationale** : If we use squared exponential kernel, then the method is non-parametric, whereas if we use polynomial kernels, the model is parametric.
  In a way a non-parametric model means that the complexity of the model is potentially infinite, it's complexity can grow with the data. If you give it more and more data, it will be able to represent more and more complex relationships.
  In contrast a parametric model's size is fixed, so after a certain point your model will be saturated, and giving it more and more data won't help . Hence the **rbf** kernel defines a function space that is a lot larger.
- Class weights, C and gamma were decided based on grid search.

3.2 Solving Missing Value Problem

In this section, we tabulate the results of all models on the imputed datasets.

3.2.1 Unimputed Dataset

Model	Hyperparameters	Cross-val F-1 Score
Bayes Classifier	-	0.757
Logistic Regression	l2 regression C = 0.01	0.589
Decision Tree (entropy)	min_samples_leaf = 11	0.837
Decision Tree (gini)	min_samples_leaf = 11	0.839
KNN (uniform weights)	k = 9	0.875
KNN (distance weighed)	K = 9	0.877
SVC (rbf kernel)	C = 10 ; gamma = 1	0.903
Random Forest	min_samples_leaf = 1; max_features = 0.8 max_samples = 0.8	0.879
XGboost Classifier	learning rate = 0.3 max_depth = 20 n_estiamtors = 1r90	0.890

Table 18: Models trained with Unimputed dataset

3.2.2 KNN Imputation Dataset

Model	Hyperparameters	Cross-val F-1 Score
Bayes Classifier	-	0.766
Logistic Regression	l2 regression C = 0.01	0.613
Decision Tree (entropy)	min_samples_leaf = 9	0.843
Decision Tree (gini)	min_samples_leaf = 9	0.847
KNN (uniform weights)	K = 3	0.888
KNN (distance weighed)	K = 3	0.887
SVC (rbf kernel)	C = 10 ; gamma = 1	0.910
Random Forest	min_samples_leaf = 1; max_features = 0.8 max_samples = 0.8	0.886
XGboost Classifier	learning rate = 0.3 max_depth = 20 n_estimators = 190	0.895

Table 19: Models trained with KNN imputed dataset

3.2.3 Miss forest Imputation Dataset

Model	Hyperparameters	Cross-val F-1 Score
Bayes Classifier	-	0.771
Logistic Regression	l2 regression C = 0.01	0.611
Decision Tree (entropy)	min_samples_leaf = 11	0.849
Decision Tree (gini)	min_samples_leaf = 11	0.849
KNN (uniform weights)	K = 3	0.895
KNN (distance weighed)	K = 3	0.893
SVC (rbf kernel)	C = 10 ; gamma = 1	0.914
Random Forest	min_samples_leaf = 1; max_features = 0.8 max_samples = 0.8	0.891
XGboost Classifier	learning rate = 0.3 max_depth = 20 n_estimators = 190	0.901

Table 20: Models trained with Miss Forest imputed dataset

3.2.4 MICE Imputation Dataset

Model	Hyperparameters	Cross-val F-1 Score
Bayes Classifier	-	0.759
Logistic Regression	l2 regression C = 0.01	0.596
Decision Tree (entropy)	min_samples_leaf = 11	0.848
Decision Tree (gini)	min_samples_leaf = 11	0.845
KNN (uniform weights)	K = 3	0.893
KNN (distance weighed)	K = 3	0.892
SVC (rbf kernel)	C = 10 ; gamma = 1	0.909
Random Forest	min_samples_leaf = 1; max_features = 0.8 max_samples = 0.8	0.891
XGboost Classifier	learning rate = 0.3 max_depth = 20 n_estimators = 190	0.897

Table 21: Models trained with MICE Imputed dataset

All the models above is trained using Duplication resampling method as SMOTE resampling method was found to be not performing better (shown in section 3.3). In duplication resampling the entire Minority class is repeated in the final dataset as multiple copies. And, the number of copies is determined by the weight value which was also found specifically for each of the individual models along with hyperparameter search.

As we can see from the above imputations the **Miss Forest** is found to be performing better on most of the classification algorithms and also has the highest Cross Validation F1 score achieved for SVC (rbf kernel).

3.3 Solving Class Imbalance Problem

- One of the main reasons for bad performance is the imbalance in the dataset i.e. no of datapoints in Class 0 and Class 1 is not equal and has very high ratio of $\approx$ 10. So one of the ways this can be solved is by using

  - Sampling based approaches like Synthetic Minority Over-Sampling Technique (SMOTE) etc.
  - Cost sensitive classification

  The SMOTE method was tested out but if wasn't able to improve much and it also lead to degradation of results for some of the algorithms. We can see the SMOTE method used and the F1 correspondingly for individual models shown below.

Model	SMOTE Cross Val F1	Duplication Cross Val F1
KNN (Uniform weighted)	0.873	**0.895**
KNN (Distance weighted)	0.871	**0.893**
SVC (rbf kernel)	0.909	**0.914**
Random Forest	0.892	**0.891**
XGBoost Classifier	0.899	**0.901**

Table 22: SMOTE Vs Duplication Resampling

We can see that Non-SMOTE duplication weighted method is better than SMOTE method.

3.4 Best Classification Algorithm

3.4.1 Ensembling of different models

The performance on the dataset was also tested by ensembling all the best individual models using Voting classifier. All the individual models were given weights for each of their predictions appropriately and the final output was thresholded relative to a constant fixed value which was chosen appropriately. As we have seen **Miss Forest** imputation to be performing better the Voting classifier is trained on it.

- **Threshold of Voting Classifier** = 1.5 i.e.

$$y_{Pred} = \begin{cases} 1, & \text{if } E_{Pred} > 1.5 \\ 0, & \text{otherwise} \end{cases}$$

Where $E_{Pred} = \sum y_{Pred_i} * ClassWeight_i$

- The Validation set composition is 20% of total dataset and has Class 0 = 14996 and Class 1 = 1004.
- Below, we tabulate the Voting ensemble and the weights for each model in it.

Model	Precision	Recall	Accuracy(%)	Cross-Val F1 Score
KNN (Uniform weighted)	0.9554	0.8493	98.77	0.8992
KNN (Distance weighted)	0.9492	0.8483	98.72	0.8959
SVC (rbf kernel)	0.9439	0.8783	98.88	0.9099
Random Forest	0.9286	0.8667	98.71	0.8966
XGBoost Classifier	0.9442	0.8667	98.81	0.9038
Voting Classifier	0.9317	0.8995	98.98	0.9153

Table 23: Ensembling of Individual models for creating Voting Classifier

Model	Model Weight
KNN (Uniform weighted)	0.85
KNN (Distance weighted)	0.85
SVC (rbf kernel)	0.89
Random Forest	0.88
XGBoost Classifier	0.88

Table 24: Class weights of Individual models in creating Voting Classifier

We can see from the above table that using the Voting Classifier ensembling technique lead to a marginal increase in the F1 Score as compared to the individual models. The final Cross Validation F1 Score achieved was **0.9153**

4 Results & Conclusions

- The K-Nearest Neighbours model does not give the best results because as the number of dimensions increases, the notion of distance or proximity becomes more and more unclear. This is typically referred to as the **Curse of dimensionality** in machine learning and we were hit by it like all college students do. Furthermore, since the training time too is much larger as compared to other classification algorithms, there was no real benefit of using KNN model for the given problem.In other words, KNN model was not able to offset variance by bias.
- The Linear Discriminant analysis(LDA) and the Quadratic Discriminant Analysis(QDA) models was discarded because the f1-score and accuracy score obtained on the validation set were quite low. Also, in cases where the training data-set is large we learnt that a **discriminative model** may perform better than a **generative model** and hence switched our attention to logistic regression and support vector machines
- Discriminative models tend to overfit when the data is high dimensional. As for logistic regression, the model is incapable of learning complex relationships as it is

a parametric approach that learns the best linear decision boundary possible. Here the bias is not offset by variance.

- Decision trees tend to overfit the training data and perform poorly on the validation set especially when the number of features is large.
- The support vector machines have the kernel trick that solve the problem of high dimensional data well, they were preferred over logistic regression and decision trees.
- By applying principal component analysis, we observe that much of the variability in our dataset could be explained using only 5 features out of 13 and since the features so obtained are orthogonal, this solves the problem of multi-collinearity in our data as well.

Predictions on Test Dataset

The prediction of the model on test dataset having 20000 datapoints is found to have 6791 0's and 13209 1's.