Final Project Report

Group 1

Tanay Dwivedi ME17B180

Dyava Naveen Reddy ME17B140

Patel Vaibhavkumar Kirtibhai ME17B176

Ishan Sunil Naryani ME17B183

Arsh MM17B001

Taher Murtaza Poonawala MM17B032

Indukuri Kumar Kaushik Varma ME17B143

Aug – Nov 2020

—

Introduction to Data Analytics

—

Prof. Nandan Sudarsanam

1. Introduction

The goal of this project is to build an end-to-end machine learning pipeline which can predict whether a loan taken by a customer will go default or not, and to derive key insights about the features that are important and helpful in the prediction. The dataset consists of following details on loans taken over a span of time:-

- ***ID*** – A unique identifier for every financial loan that is being considered.
- ***Loan type*** – Type of loan taken. This can be of two types, namely 'A' or 'B'.
- ***Occupation type*** – Occupation of the customer. This can be of three occupation types, namely 'X', 'Y', or 'Z'.
- ***Income*** – A continuous variable that indicates annual income of the customer. Note that, this is not the exact income value.
- ***Expense*** – A continuous variable that indicates the annual expense of the customer. Note that, this is not the exact expense value.
- ***Age*** – Age of customer. A value of '0' is considered as below 50 years, and value of '1' is considered as above 50 years.
- ***Score1 to Score5***: Represents five different metrics calculated by the organization, about the customer and the loan that is being considered.
- ***Labels*** – '0' means non-default, and '1' means default on that loan.

Section 2 covers the basic pre-processing steps, followed by Exploratory Data Analysis (EDA). Main focus here is on visualizing the distribution of data through graphs. This step played a key role in determining the relative importance of features, along with bringing down the subset of possible models to fit. In section 3, we describe data preparation and model planning steps. We have taken both supervised, as well as an unsupervised approach while developing the model. It also presents a brief overview of all the models we developed, along with their corresponding predictions results on cross-validation data. Our best model is explained in Section 3.7 in details, which we use to predict the final test data. Due to the fact that most of the loan taking customers are non-defaulters, the data set is highly unbalanced with a large number of observations for class '0' as compared to class '1'. Therefore, instead of using simple accuracy as metric, we are using balanced accuracy and F1-score for evaluating the performance of all the models. Section 4 concludes this report by summarizing key results obtained in the project. Our project Jupyter notebook is available at the following GitHub repository after the deadline date (i.e., from 06th January 2020, 23.59 IST onwards) :

https://github.com/tanaydw/projects

2. Discovery

The data set consists of two Comma Separated Value (csv) files. The first file *train_x.csv* contains 80,000 train instances with features listed in Section 1. A companion file *train_y.csv* gives the corresponding class labels for every unique ID in *train_x.csv*. The goal is to use this training data to learn a mapping which uses input features and predicts corresponding class labels. Large amount of missing values characteristic of real data is observed in the training data. The goal of pre-processing is to impute these missing values to the extent possible. However, for exploratory data analysis, all the missing values are dropped initially. This leaves us with **59,238 training instances**, which is large enough data set for visualizing the underlying distribution of data.

First, all the nominal features, i.e., Income, Expense, Score1, Score2, Score3, Score4, Score5 are standardized with column mean and column standard deviation of features. Following formula is used for reducing the features column to standard normal :-

$$x_{new,i,k} = \frac{x_{old,i,k} - \mu_{:,k}}{\sigma_{:,k}} \quad \text{... (1)}$$

Where $k \in \{Income,\ Expense,\ Score1,\ Score2,\ Score3,\ Score4,\ Score5\}, i \in \{row\ index\}$, and : represents summation over all row index. μ *and* σ have their usual meaning of mean and standard deviation, respectively.

Reducing feature columns to their standard normal in this way, we observe that the Expense and Score5 represent exactly the same feature. This helps us in two ways. First, this can be used to impute the missing values and second, one of the columns can be dropped after applying imputation technique to remove redundant data (in our case, we choose to drop Expense), thereby improving the stability of any model we fit.

Next, pair plots are plotted using seaborn to visualize the correlation between any two features in training data. Pair plots are preferred overo heatmaps because it gives an idea of underlying distribution of the data. *Figure 1* shows some of the plots. For the visualization of all the pair plots, please refer to project Jupyter Notebook. Following insights about can be drawn from pair plots :-

1. Expense and Score5 represent exactly the same features, see *Figure 1 (a)*.
2. Age is a feature made by thresholding Score2, see *Figure 1 (b)*. Thus, if Score2 > 0.0979, Age is '1', else Age is '0'.
3. Gaussian distributions can be observed in plots, see *Figure 1 (c)* and *Figure 1 (d).*
4. ID does not play any role in prediction. Therefore, it can be dropped safely.
5. No correlation is observed in ordinal features (categorical), namely Occupation type, Loan type and Age. Therefore, these can be dropped safely as well.

3. Model Planning & Building

Point 1 and 2 from observations in Section 2 above are used for data imputation as mentioned previously. Imputing the data with this knowledge increased the data set size to **67,097 training instances**. Since these data were sufficient for learning the models, further data imputation was not carried out. Observations from Point 4 and point 5, led us to use Income, Score1, Score2, Score3, Score4, Score5 as important nominal features for building all our models. However, for modelling Decision Trees and Random Forest, we include ordinal features, namely Occupation type and Loan type since these models can handle ordinal features well. Performance of all the models are listed in *Table 1.*

3.1 Decision Trees

We first took a supervised learning approach where training data represents our feature vector and class labels their corresponding targets. Income, Score1, Score2, Score3, Score4, Score5, Occupation type and Loan type were chosen as features for training the model. Ordinal Features of Occupation type and

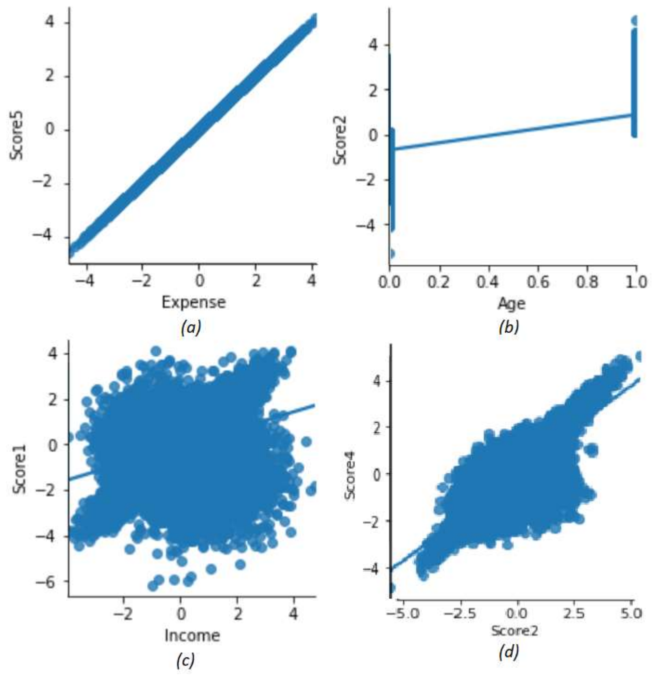

Figure 1. Pair plots. (a) Expense vs Score 5, (b) Age vs Score2, (c) Income vs Score1, (d) Score2 vs Score4.

Loan type are converted using one-hot encoding. For deciding best hyperparameters choice for decision tree, namely max tree depth, min leaf size, split criteria, we performed exhaustive grid search. Selecting the most appropriate hyperparameters this way, we got a balanced accuracy of 0.893 and an F1-score of 0.798 on cross-validation data.

3.2 Random Forest

Considering the fact that ensemble methods usually have higher accuracy than non-ensemble methods, the next model which we built was Random Forest. Similar feature selection as Decision Trees, i.e., Income, Score1, Score2, Score3, Score4, Score5, Occupation type and Loan type with one-hot encoding for ordinal features was developed. Exhaustive grid search was again our approach for hyperparameter tuning. A balanced accuracy of 0.886 and an F1-score of 0.853 was observed for Random Forest, see *Table 1.*

3.3 Artificial Neural Networks

Neural Networks being highly non-linear models can fit complicated data distributions very well. Therefore, a simple feed-forward multi-layer perceptron was implemented using Tensorflow with input layer having same nominal features as Decision Tree and Random Forest, five hidden layers and a binary class output layer. Sigmoid activation function was used for final layer on which final prediction was thresholded at 0.5.

Table 1. *Train and Validation Balanced Accuracy & F1-score for various Models*

S. No.	Model	Train Balanced Accuracy	Train F1-Score	Validation Balanced Accuracy	Validation F1-Score	Features
1	Decision Trees	0.964	0.942	0.893	0.798	Nominal + Ordinal
2	Random Forest	0.999	0.987	0.886	0.853	Nominal + Ordinal
3	Artificial Neural Networks	0.992	0.971	0.972	0.915	Nominal
4	Support Vector Machines	0.891	0.861	0.883	0.853	Nominal
5	K-Nearest Neighbours	0.906	0.884	0.902	0.880	Nominal
6	Gaussian Naïve Bayes	0.706	0.568	0.711	0.572	Nominal
7	Gaussian Mixture Models	0.994	0.957	**0.993**	**0.954**	Nominal

Rectified linear units (ReLU) were used for rest of the hidden layers. Area Under Curve (AUC) was chosen as the corresponding loss function on which the neural network was optimized. Adam being practically the best optimizer was our natural choice for optimization. A balanced accuracy of 0.972 and an F1-score of 0.915 was achieved using Artificial Neural Networks, see *Table 1*.

3.4 Support Vector Machines (SVM)

The main idea behind support vector machines is to construct decision boundaries which minimizes the misclassification rate. For training an SVM, we used all the nominal features, i.e., Income, Score1, Score2, Score3, Score4, and Score5. After standardizing the nominal features, Radial Basis Function (RBF) kernels were applied due to visible underlying Gaussian Distribution. However, SVM did not gave a performance better than Neural Networks indicating that they could not construct a decision boundary properly. This can be attributed to the fact that data instances overlap in higher dimension, thereby failing to give a defined separation boundary. We observe a validation balanced accuracy of 0.883 and an F1-score of 0.853, see *Table 1*.

3.5 K-Nearest Neighbours (KNN)

For applying K-Nearest Neighbours, we used nominal features and standardized them just like in case of Support Vector Machines. This endured that the distance calculations for KNN can be implemented on a scale which is independent of any variables. Selecting K = 5, we achieve validation balanced accuracy of 0.902 and F1-score of 0.880, see *Table 1*.

3.6 Gaussian Naïve Bayes

Due to the continuous variables in nominal features and using the fact that visualization suggests that the data is distributed normally, Gaussian Naïve Bayes was considered as a possible approach. All the nominal features are used as input features and corresponding class labels are predicted. We observe a balanced accuracy of 0.711 and an F1-score of 0.572, see *Table 1*. These are strong indications of the fact that one class is composition of more than one Gaussian distribution. Relative high performance of KNN also assists in coming to this conclusion.

3.7 Gaussian Mixture Models (GMMs)

Point 3 in observations of Section 2, along with observations in Section 3.5 and Section 3.6 indicates that underlying Gaussian nature of the data can be exploited during modelling. This motivated us to approach the problem in rather **unsupervised** setting and use Gaussian Mixture Models (GMMs) for predicting the class labels.

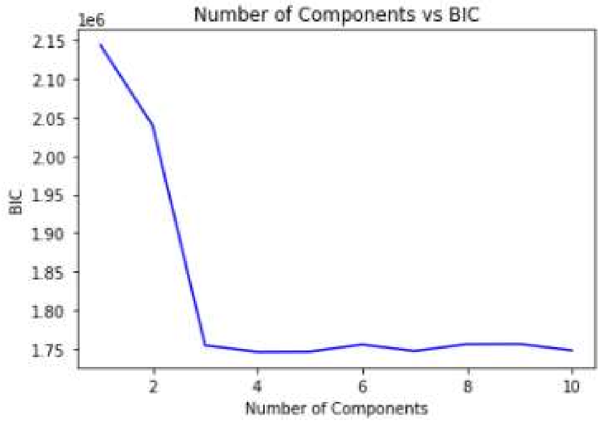

Figure 2. Number of Clusters vs Bayesian Information Criteria (BIC) for selecting optimal clusters, in this case 4.

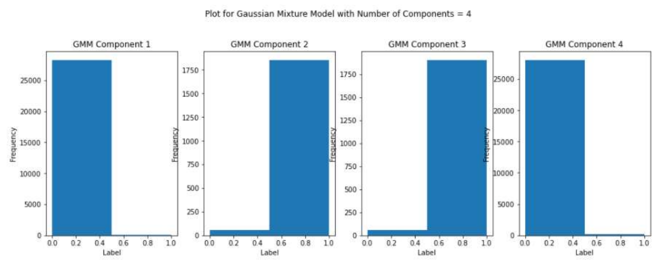

Figure 3. Histograms indicating clear separation of classes among cluster.

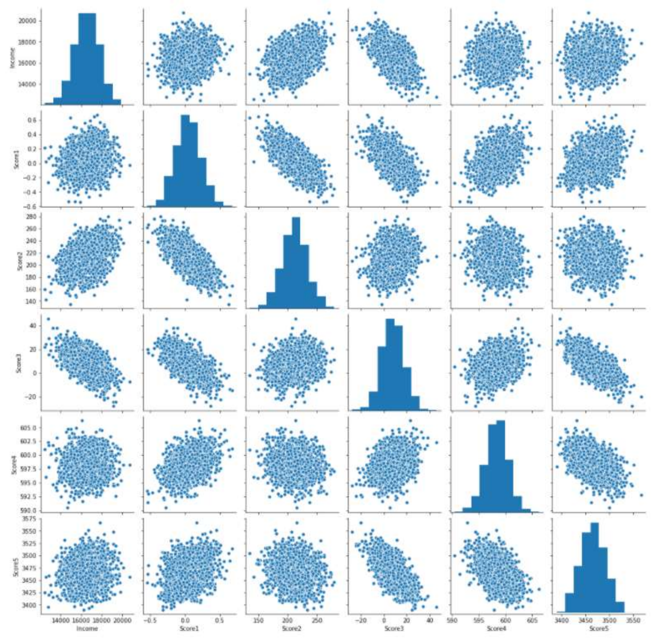

Figure 4. Visualizing Pair plot of Cluster 1.

Income, Score1, Score2, Score3, Score4, and Score5 were selected as input features. Now the goal was to form clusters in training data such that one single cluster corresponds to a given class. Fitting Gaussian Mixture Models in higher dimension, we observe that four clusters form the most optimal groups. Number of clusters were selected based on finding the minimum of Bayesian Information Criteria (BIC) for various cluster settings, see *Figure 2.* Out of these four clusters, almost all the instances in cluster 1 and cluster 4 belongs to class label '0' and all the instances in cluster 2 and cluster 3 belongs to class label '1'. This is indicated in *Figure 3*, where clear class separation in histograms of all the four clusters can be observed. Fitting GMMs this way, we achieved a balanced accuracy of 0.993 and an F1-score of 0.954 on cross-validation data. This is clear indication of the fact that GMMs have generalized well and explain the data distribution with a remarkably high accuracy. Following the good predictions performance from the model, we have analysed the variants of GMM by dropping various feature (best subset selection). However, dropping more than one features leads to decrease in performance, therefore indicating to use all the six nominal features forming clusters in higher dimensions are accurate. *Figure 4,* which is similar to *Figure 1,* shows the pair plots after segregating data into clusters. Unlike *Figure 1* clearly separated gaussian normal distributions can be observed in *Figure 4*.

4. Conclusion

We began by visualizing the distribution of data using pair plots. We implemented seven machine learning models, presenting their results and key conclusions about each of these models. The underlying distribution of the data set and class label is essentially a Mixture of Gaussian Distributions. 4 such powerful clusters have been separated from each other using Gaussian Mixture Models. Doing this we achieved an accuracy of 0.993 and an F1-score of 0.954 on validation set. This indicates that our model has generalized well and can be used to predict unknown test instances. *test_x.csv* file is predicted based on this model and finally saved as *pred.csv* file.

5. References

1. Prof. Nandan Sudarsanam's Course on Introduction to Data Analytics, IIT Madras, Aug-Nov 2020.

2. Gareth, Daniela Witten, Trevor Hastie, Robert Tibshirani, An Introduction to Statistical Learning: with Applications in R.

3. Scikit-learn API, https://scikit-learn.org/stable/