

# Application of Artificial Neural Networks to Identify Problematic Loans in the Financial Sector

Written Assignment for Business Data Analytics, Quantitative Methods  
and Visualization (LA F20 BDMAO1023U)

Fuad Salimzade (Student Number: 138465)

David Rivas Barbosa (Student Number: 138691)

Teachers: Daniel Hardt, Chris Zimmerman, Liana Razmerita

Date of Submission: 29 May 2020

Copenhagen Business School

Spring Semester 2020

Characters: 25,060

Pages: 15

# Table of Contents

<b>ABSTRACT .....</b>	<b>2</b>
<b>INTRODUCTION .....</b>	<b>3</b>
<b>METHODOLOGY .....</b>	<b>4</b>
<b>DATASET AND INITIAL PREPROCESSING:.....</b>	<b>4</b>
<b>FURTHER PREPROCESSING:.....</b>	<b>6</b>
<b>BUILDING THE MACHINE LEARNING MODEL:.....</b>	<b>8</b>
<i>Artificial Neural Networks.....</i>	<i>8</i>
<i>Logistic Regression.....</i>	<i>10</i>
<i>Random Forests .....</i>	<i>11</i>
<i>XGBoost .....</i>	<i>11</i>
<i>Gaussian Classifier .....</i>	<i>12</i>
<b>BUSINESS RELEVANCE AND REFLECTIONS.....</b>	<b>13</b>
<b>CONCLUSION .....</b>	<b>15</b>
<b>REFERENCES .....</b>	<b>16</b>
<b>APPENDICES .....</b>	<b>17</b>
<b>APPENDIX 1.....</b>	<b>17</b>
<b>APPENDIX 2.....</b>	<b>18</b>
<b>APPENDIX 3.....</b>	<b>19</b>
<b>APPENDIX 4.....</b>	<b>20</b>
<b>APPENDIX 5.....</b>	<b>21</b>
<b>APPENDIX 6.....</b>	<b>22</b>
<b>APPENDIX 7.....</b>	<b>23</b>
<b>APPENDIX 8.....</b>	<b>23</b>

## **Abstract**

This paper consists of the implementation of Machine Learning algorithms to build models that, after being trained and tested with finalized loans, could accurately predict the future status of current loans issued by LendingClub during the 2015-2018 period. The different Machine Learning algorithms used were: Artificial Neural Networks, Logistic Regression, Random Forests, XGBoost, and Gaussian Classifier. Ultimately, the model built using Artificial Neural Networks yielded the better testing results and was therefore used to obtain the loan status predictions for the totality of the current loans. Through the visualization of this results, patterns showed that a considerable increase in loan defaults was foreseeable, due to a noticeable cutback in interest rates in the period, which in turn incited a reckless increase in the total of borrowers, regardless of any certainty if whether they could confront this obligations. The model and its results serve as a tool that can be used to raise awareness about LendingClub's operations, in order to advise for a change which can reduce the company's risk, whilst adhering to its commitment of providing simple financial opportunities to individuals.

*Keywords: Artificial Neural Networks, Machine Learning, LendingClub, Loan Default.*

## Introduction

LendingClub is an American lending company founded in 2007. It is a peer-to-peer lending company which connects relatively small individual borrowers with groups of individual investors, minimizing the involvement of big financial institutions, and therefore avoiding big transaction costs, whilst charging a small fee to both the investors and the borrowers for its service (Investopedia, 2020).

Borrowers are given the option of applying for a personal loan for up to \$40,000 or a small business loan for up to \$500,000. Their applications are screened and analyzed by LendingClub, which in turn gives this processed information to the investors. Investors then decide their strategy by choosing their risk preference, diversification degree, and investment amount. The borrowers receive loans with specific payment terms and interest rates, which are owned by a group of different investors (LendingClub, 2020).

For this assignment, a dataset comprised of the information of all loans made through LendingClub during the 2015-2018 period was used to analyze borrower features and behavior. Ultimately, the objective was to use Machine Learning algorithms to answer our research question: Based on the prediction models built with finalized loan data, how many of the current loans from LendingClub are going to default in the future?

The dataset was first preprocessed in its comma-separated values (CSV) file form, removing useless information and transforming it into a more manageable format. Next, the dataset was split into smaller datasets depending of their loan status, 'Current' or 'Not-current'. The 'Not-current' loans dataset was used to train, and test Machine Learning models programmed in Python that could accurately predict loan status. Afterwards, the trained models were applied to the 'Current' loans dataset to obtain predicted results with a high accuracy.

The results found, as well as relevant information gathered throughout the process were visualized using Tableau software. All these results were analyzed from a business and ethical perspective, providing insights into any interesting patterns and correlations throughout the data.

## Methodology

### Dataset and Initial Preprocessing:

The initial dataset used in the assignment was obtained through Kaggle (Kaggle, 2020), a Google-owned online community where a wide variety of datasets are shared. The initial dataset includes all the loans from LendingClub issued from October 2015 to December 2018. This dataset has a total of 145 feature columns and 1,048,575 borrowers or rows. The large number of features correspond to a detailed description of every operation including the loan details, and financial and personal information from the borrower. A limited look into the original dataset is available in Appendix 1.

Some of the loan details are the *Loan Amount*, the *Term of the Loan*, the *Interest Rate*, the *Grade* given to the loan, the *Issue Date*, the dates of the *First* and (if applicable) the *Last Payments*, the *Purpose* for which it was requested, and most importantly for our analysis- the *Loan Status*.

The personal information includes the *State* of residence in the United States, the *Zip Code* of the borrower, the borrower's *Homeownership Status*, the borrower's *Employment Title*, and *Employment Length*.

The borrower's financial information is much more detailed. It includes the *Annual Income*, a *Debt-to-Income* ratio, *Public Records*, *Public Record Bankruptcies*, *Number of Delinquencies*, *Number of Credit Inquiries*, *Number of Open Credit Accounts*, *Number of Total Accounts*, *Number of Tax Liens*, *Revolving Balance*, *Joint Accounts*, any *Hardships Confronted*, and much more. The large number of features is, in great part, due to these financial details. However, many features are simple variations derived from the previously mentioned characteristics, e.g. *Number of Delinquencies in Last Two Years*, *Months Since Last Delinquency*, and so on.

Prior to feeding the dataset to the Machine Learning models and any preprocessing in Python, some basic preprocessing was done in its CSV form using Excel worksheets.

Two preliminary approaches were followed in the preprocessing part in its CSV form. A first approach resulted in the removal of most of the features, leaving only the 20 features that seemed the most important. Nonetheless, this procedure proved counterproductive given that the training and testing results of the Machine Learning models built with this dataset were mediocre, with

around 0.77 accuracy in both the training and testing sets, the model had been oversimplified, or in other terms underfitted. The set of features used was a great example of a cognitive bias: They had been picked depending of the intrinsic value we had given them, either by our own familiarity with the concepts or an unconscious bias, instead of being chosen after a careful feature importance analysis.

After a more cautious preprocessing, we ended up with 90 features, some of them modified from their original form. In addition, around 1,572 instances were removed because they had incomplete information in many of the features such as *Annual Income* and other ratios calculated with that information.

Some of the features were empty for most of the borrowers and where therefore deleted, including *Title*, *ID*, and *Member ID*. Other features such as *Zip Code* and *Description* were incomplete and didn't provide any insight for our analysis, therefore they were deleted.

The *Employment Title* feature seemed like a very interesting feature that could be of great use for analysis and visualization, nonetheless, it was also deleted. There were overwhelming discrepancies in how the borrowers had filled this feature. The instances for this feature were filled with typos, use of uncommon acronyms, overcomplications, and outliers which made it practically impossible to categorize the instances, especially in a dataset of this enormous size.

The reduction in the number of features was, in great part, the result of the removal of features related to hardships and their subsequent settlements. The description of the hardships was highly detailed through the use of several distinct features, but the proportion of borrowers that had suffered a hardship scenario was very small, only accounting for less than 0.55% of the dataset, therefore, the decision was made to delete these features instead of including NaN values for most of the instances.

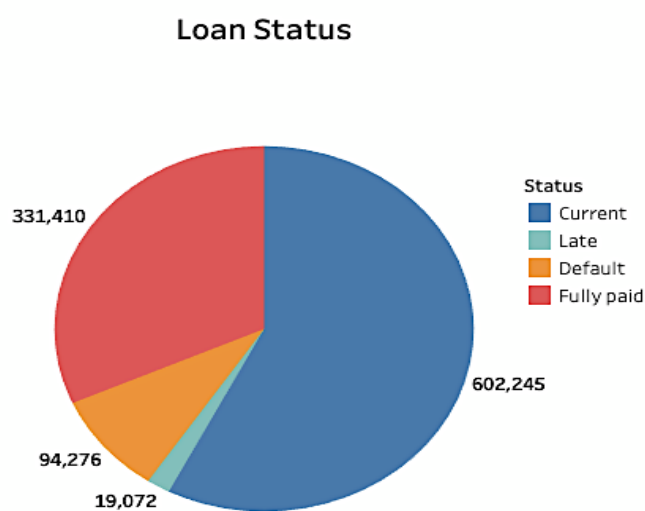
Some minor changes were made to categories such as *Employment Length*, in which some instances were modified to be integers. This modification in particular consisted of changing instances with “<1 year” or “+10 years” to 0.5 and 10 respectively.

Lastly, some of the most relevant features where reorganized to have only a small number of categories. *Loan Status* was slightly simplified into four categories: *Current*, *Late*, *Fully Paid*, and *Default*. Likewise, *Purpose* was changed to cut the number of categories from 13 to 6: *Business*

*loan, Car loans, Credit Card and Debt Consolidation, House-Major-Purchase and improvements, Medical loans, and Other.*

The shares of each of the different loan statuses in the preprocessed dataset can be seen in Image 1.

Once all the preprocessing in the CSV form was done, using filters, a final split into two datasets was made: One including only 'Current' loans and another showing the 'Not-current' loans (*Default, Fully Paid, Late*).



Visualized by Fuad Salimzade and David Rivas. Information retrieved from:<<https://www.kaggle.com/wendykan/lending-club-loan-data>>

*Image 1*

### Further Preprocessing:

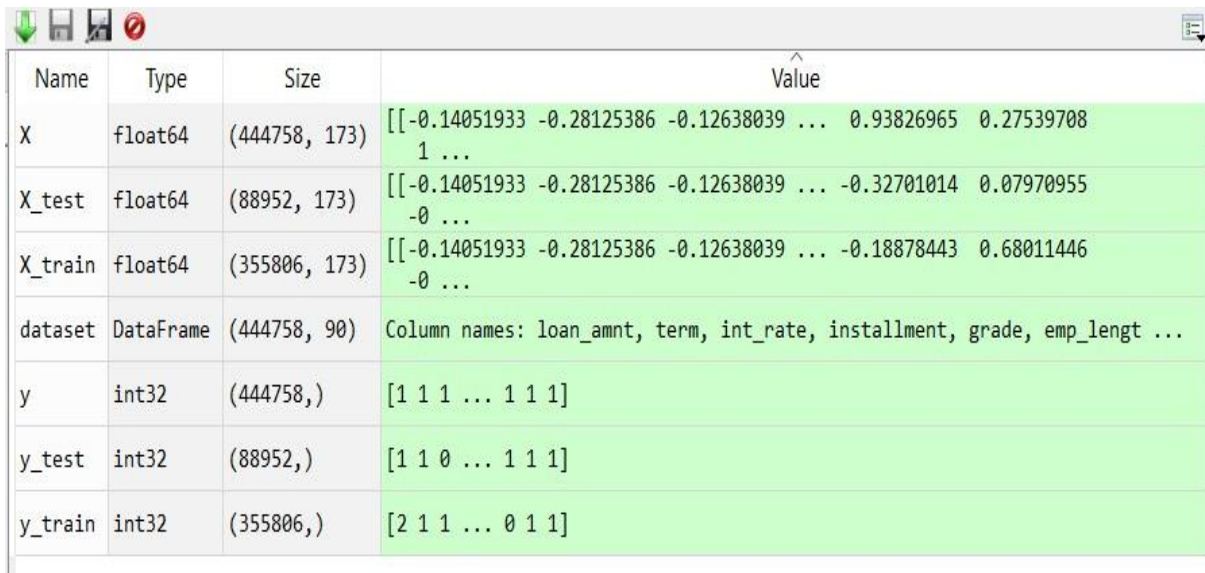
The objective of using Machine Learning in this assignment was to build, train, test, and optimize models using the 'Not-current' loan dataset so they could accurately learn and predict the status of a loan. Afterwards, the fitted models would be used for the 'Current' loan dataset to predict if these loans will be paid in full, will be late, or will ultimately go on default.

Before using any of the Machine Learning algorithms, a comprehensive preprocessing and data split were done using Spyder environment, which can be seen in detail in Appendix 2.

Firstly, the *numpy* and *pandas* libraries were imported. Using *pandas*, the ‘Not-current’ dataset, with a shape of (444758, 90) was imported. Then, the dataset was divided into data (X), consisting of the first 89 columns, and target value (y), consisting of a vector which states the status of each loan. Using *SimpleImputer* from *sklearn*, the NaN values remaining in the dataset were converted into 0.

Next, for (X) dataset, the *OneHotEncoder* and *ColumnTransformer* functions from the *sklearn* library were used to transform string values into dummy variables. A total of 5 column transformations were implemented for *Grade*, *Home Ownership*, *Purpose*, *Address State* and *Issue Date*. The first column at the beginning of each transformation was deleted to avoid the dummy trap. Column transformation resulted in 89 new columns, making our (X) dataset have a shape of (44758, 173). For the (y) categorical values, *LabelEncoder* was used to transform the target values into [0, 1, 2], *Default*, *Fully Paid* and *Late* respectively. Lastly, feature scaling through *StandardScaler* was applied to the (X) dataset.

Furthermore, the (X) dataset was split into training and testing sets using the *train\_test\_split* function from the *sklearn* library with a *test\_size* equal to 0.2. Image 2 provides insights into the preprocessing results.



Name	Type	Size	Value
X	float64	(444758, 173)	[[-0.14051933 -0.28125386 -0.12638039 ... 0.93826965 0.27539708 1 ...
X_test	float64	(88952, 173)	[[-0.14051933 -0.28125386 -0.12638039 ... -0.32701014 0.07970955 -0 ...
X_train	float64	(355806, 173)	[[-0.14051933 -0.28125386 -0.12638039 ... -0.18878443 0.68011446 -0 ...
dataset	DataFrame	(444758, 90)	Column names: loan_amnt, term, int_rate, installment, grade, emp_lengt ...
y	int32	(444758,)	[1 1 1 ... 1 1 1]
y_test	int32	(88952,)	[1 1 0 ... 1 1 1]
y_train	int32	(355806,)	[2 1 1 ... 0 1 1]

Image 2



## Building the Machine Learning Model:

In order to build the most accurate prediction model possible, a wide variety of Machine Learning algorithms was used and tested. Image 3 shows the training and testing scores achieved with these different models. From the start, all the models were named identically as *classifier* for simplicity.

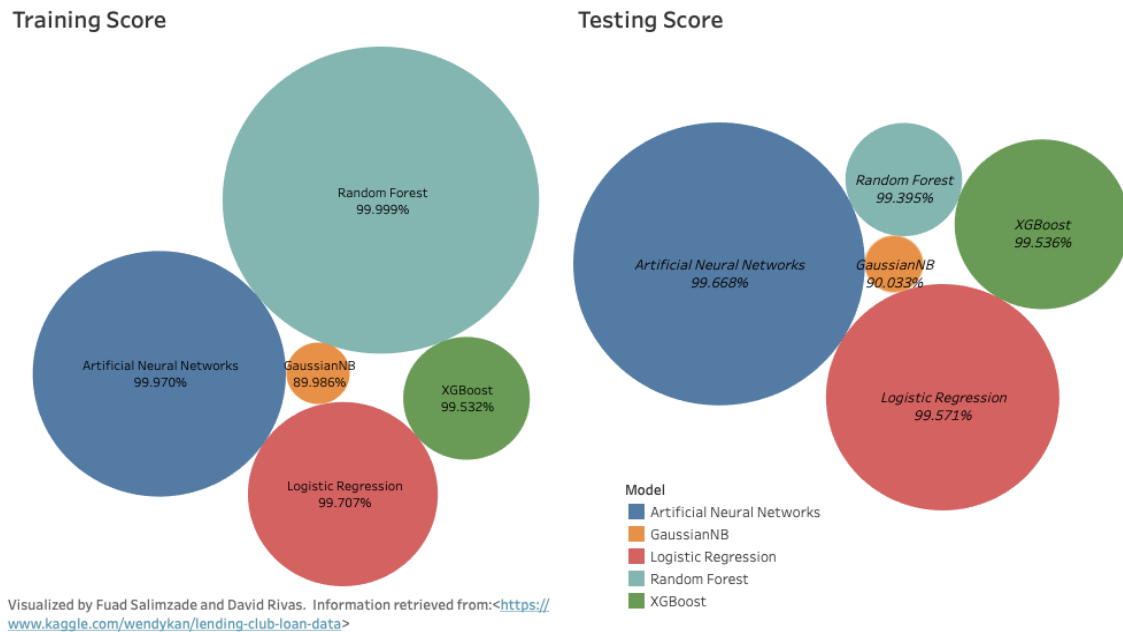


Image 3

## Artificial Neural Networks

Artificial Neural Networks is a deep learning model, a major subset of Machine Learning. “Artificial Neural Networks or ANN is an information processing paradigm that is inspired by the way the biological nervous system such as brain process information. It is composed of large number of highly interconnected processing elements (neurons) working in unison to solve a specific problem.” (TowardsDataScience, 2019)

Appendix 3 shows the full implementation of the code for the ANN classifier.

To build the ANN model, all the classes and functions were imported from the *keras* library using *tensorflow*. Unlike the other models, (y) target value was transformed using *ColumnTransformer* rather than *LabelEncoder* to be able to apply the ANN model.

To begin with, *Sequential* class was passed to initialize the *classifier* and *Dense* class was used to add input and hidden layers. After testing with 1 and 2 hidden layers, the decision was made to use a total of 3 hidden layers. In the first layer, input dimension corresponded to the number of features (*input\_dim = 173*) and the number of hidden neurons (*units*) used in each layer was 117, approximately 2/3 of the dataset size, in our case, the sum of 173 from (X) and 3 from (y). For the first and second hidden layer, the rectifier activation function '*relu*' was applied due to a relatively shorter waiting time and better delivery performance on large datasets. The output layer included only 3 output dimensions (*units = 3*) since the (y) target value had only 3 classes. Due to a multiclass output dimension, sigmoid activation function '*softmax*' was passed which, in turn, returned probabilities for each class.

As a last step in building the ANN *classifier*, the model was compiled using the *compile* method. The loss function was chosen to be '*categorical\_crossentropy*', and '*adam*' was passed for optimizer since it is best suited for neural nets and sparse data likewise in this model.

The model was trained using the *fit* function, and a *batch\_size* equal to 64 and *epochs* equal to 100 were passed, considering the data size, for a shorter waiting period. This process can be viewed in Appendix 4.

#### Analyzing the accuracy score

##### ANN results

Hidden layers	# of Type I errors	# of Type II errors	Accuracy	Profit/Loss (Average)	Profit/Loss (Median)
1	45	232	99.6796%	(\$2,738,470)	(\$2,244,000)
2	47	268	99.6335%	(\$3,236,374)	(\$2,652,000)
3	93	203	99.6627%	(\$1,610,865)	(\$1,320,000)

#### Image 4

Variable (y\_pred) was created using *predict* function in order to compare test results with predicted results. The *Confusion matrix* from sklearn.metrics library was used to calculate test accuracy, number of Type I, and Type II errors. *Profit/Loss* depicts how much the financial institution is to lose or profit if they have identified status of their future loans using the model built here. *Profit/Loss* was calculated multiplying the difference in false positives (Type I error) and false negatives (Type II error) by the average or median borrowed loan amount. Overall, considering

mean accuracy, number of errors and *Profit/Loss*, the classifier with 3 hidden layers was chosen to perform a further analysis to determine the status of current loans.

### *Predicting the “Current” Dataset*

After achieving an optimal accuracy score, the current loan dataset (variable name ‘Current’ with a shape of (602245, 89)) was imported and preprocessed in a similar way as the previous dataset (‘Not-current’). The preprocessed final ‘Current’ dataset had a shape of (602245, 173). To obtain the predicted values (*y\_new*), the *classifier.predict* function was used and then transformed into a CSV file (*‘predicted\_results.csv’*) using the *pandas* library.

The following Machine Learning models were built in parallel to the ANN model. Although the results of the ANN model were ultimately the ones used for the prediction analysis, the rest of the models helped for comparison. Appendix 5 show a brief look into the coding of the next models.

## **Logistic Regression**

Logistic Regression models generally surpass other models in large datasets consisting of mainly sparse data (Müller & Guido, 2016). For this dataset, Logistic Regression performed above average if compared to other models and delivered the highest accuracy after ANN. The main advantage, discovered during the training stage, was that it required a smaller amount of time to train the model whilst showing consistent accuracy.

Initially, the classifier was trained with default parameters, and, hence, training and testing results were obtained using the *score* method. The training and testing accuracies obtained were 0.9970 and 0.9957 respectively, which showed that the model might be underfitting. In order to prevent underfitting and to increase training and test accuracy, Cross Validation (*cross\_val\_score*) and Grid Search (*GridSearch*) were implemented to tune the parameters.

Initially, cross validation of model was implemented using the default validation estimator, a 3-fold cross, and afterwards using a 5-fold cross (*‘cv = 5’*). The 5-fold cross validation delivered better accuracy equaling to 0.996810, with a standard deviation of 0.000262.

To further increase test accuracy, the regularization parameter  $C$  was tuned using the grid search method. After testing for different values of  $C$  ( $[0.01, 0.1, 1, 10]$ ), *grid.best\_params\_* showed  $C = 10$  to be the best parameter for regularization.

## Random Forests

Random forests is another powerful classification model which can embody in itself many decision trees, thus achieving higher accuracy through learning. However, the main disadvantage of this model is its tendency to overfit and the need of a deeper customization to avoid overfitting. While building Random Forests, as an exception to the other models built, *StandardScaling* wasn't applied, given that Random Forests don't need any scaling of the data (Müller & Guido, 2016, p. 85-89).

The parameter tuning in the model was fairly simple. The parameter of *max\_features* was kept to a default value of the square of the number of features. In the case of *n\_estimators*, several values were tested, the higher the number the better results were gained, nevertheless, it was ultimately set to 100, given that with larger numbers it took an excessive amount of time to train the model which, in turn, limited the ability to train the model for cases when *n\_estimators* was higher than 100.

The training score obtained through the Random Forest method was the highest amounting to 0.99999 raising concerns that model might be overfitting since no limit for *max\_depth* was set, and accordingly tuning this parameter seemed impossible due to very long waiting time. However, the generalization of new data, the test score equaled to 0.99395, although by no means deficient, was inferior to that of other models.

## XGBoost

Extreme Gradient Boosting (XGBoost) is a newly rising application of gradient boosted decision trees and can be highly customized while constructing models. (Developers, 2020). Building models using XGBoost allows the user to achieve great optimization with less waiting time in complex datasets and somewhat resembles the Random Forests model in its structure and in the same way, that does not require feature scaling to run the model.

In the model, the training and testing scores reached through the use of XGBoost were fairly consistent. Nevertheless, other models were imported from *sklearn*, *XGBClassifier* was imported from *xgboost* library. In comparison to Random Forest, XGBoost took smaller amount of time to fit and, at the same time, delivered better, but mediocre compared to the rest, test accuracy.

Regardless of the fact that a deeper customization and parameter tuning of the model could have potentially led to better scores, due to the relative simplicity and preference of building other models, especially ANN and Logistic classifiers, led to focusing more in tuning those models rather than XGBoost.

## **Gaussian Classifier**

Of all the models built, the Gaussian Classifier yielded the worst scores for both testing and training. The Gaussian Classifier was chosen amongst other common classifiers of the Naive Bayes family, namely Bernoulli and Multinomial classifiers, for its theoretical ability to handle high-dimensional datasets (Müller & Guido, 2016, p. 72).

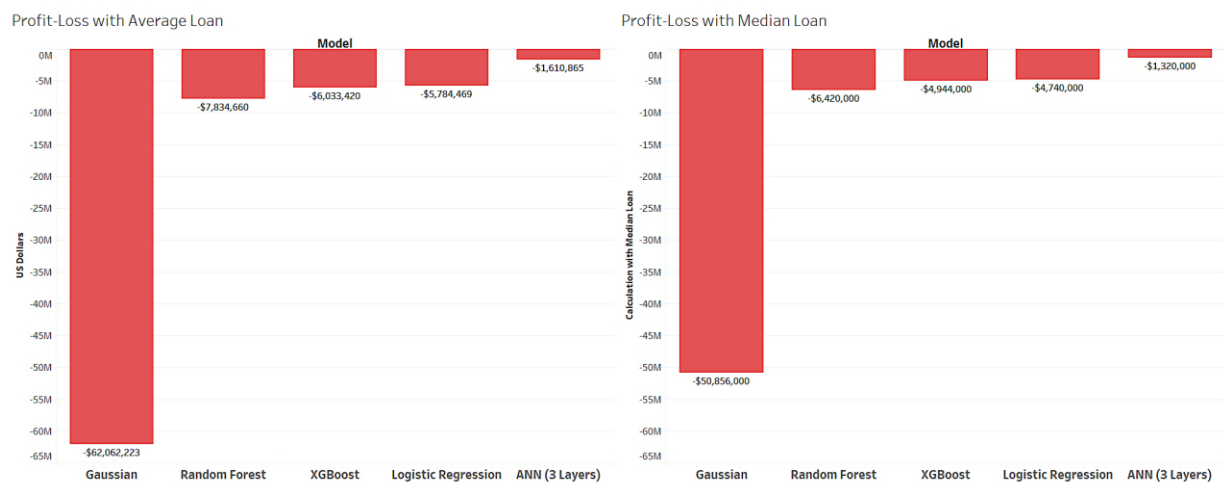
Compared to the other models built, the model using Gaussian Classifier took a remarkably lower time to train. Nevertheless, this timesaving advantage proved, somewhat, useless for the main objective it was built for, optimizing accuracy. It yielded training accuracy of 0.89986 and test accuracy of 0.90033.

The results of the classifier show a slight underperformance. In addition to this, the limited parameter tuning that could be performed in the classifier ultimately led to a preference for using other types of classifiers to build the model needed.

## Business Relevance and Reflections

The financial ramifications of the accuracy scores obtained throughout the different models built can be seen in the analysis shown in the Image 5. This analysis is similar to the one made in the Artificial Neural Networks section.

The confusion matrices of each of the models were used to get the specific number of misclassified instances. These instances were divided into false positives (Type I error), loans misclassified as Fully Paid, and false negatives (Type II error), loans misclassified as late or default when they were actually Fully Paid. As in the previous section, but for every model, the number of false positives was subtracted from the number of false negatives, and then multiplied by the mean and median loan amounts.



*Image 5*

The calculated results portray the amount of money LendingClub would hypothetically lose when using the models built to predict loan status, due to the slight inaccuracy of each model. In accordance with the testing score results, the Artificial Neural Networks model showed the smallest amount of losses due to misclassification. The models built with Logistic Regression, XGBoost, and Random Forests exhibited greater but similar losses. Last of all, as expected, the Gaussian Classifier model manifested the worst results by a considerable margin.

Since the data supplied by LendingClub encompasses large and comprehensive information about customers' personal and financial history, many various insights can be derived both in an individual and an aggregate context. In Appendix 6, the dashboard representing loans in a macro

dimension was analyzed through *Debt Ratio*, *Home Ownership*, and *State* to show patterns in loan statuses. It can be easily noticed that, people in densely populated urban areas tend to borrow more money and therefore, have more burdens, especially in the states of California, Texas, New York and Florida. In this regard, the very same states are prone to default more often largely due to a higher amount of loans borrowed.

By analyzing customers through *Home Ownership*, it can be inferred that people who rent have a relatively higher likelihood to default and be late on payment dates, in contrast to people living in mortgaged properties, who tend default the least. Moreover, at the bottom left of Appendix 6, the histogram shows the *Annual Income* distribution of all borrowers. People earning annually in a range of 30 to 75 thousand dollars borrow more frequently.

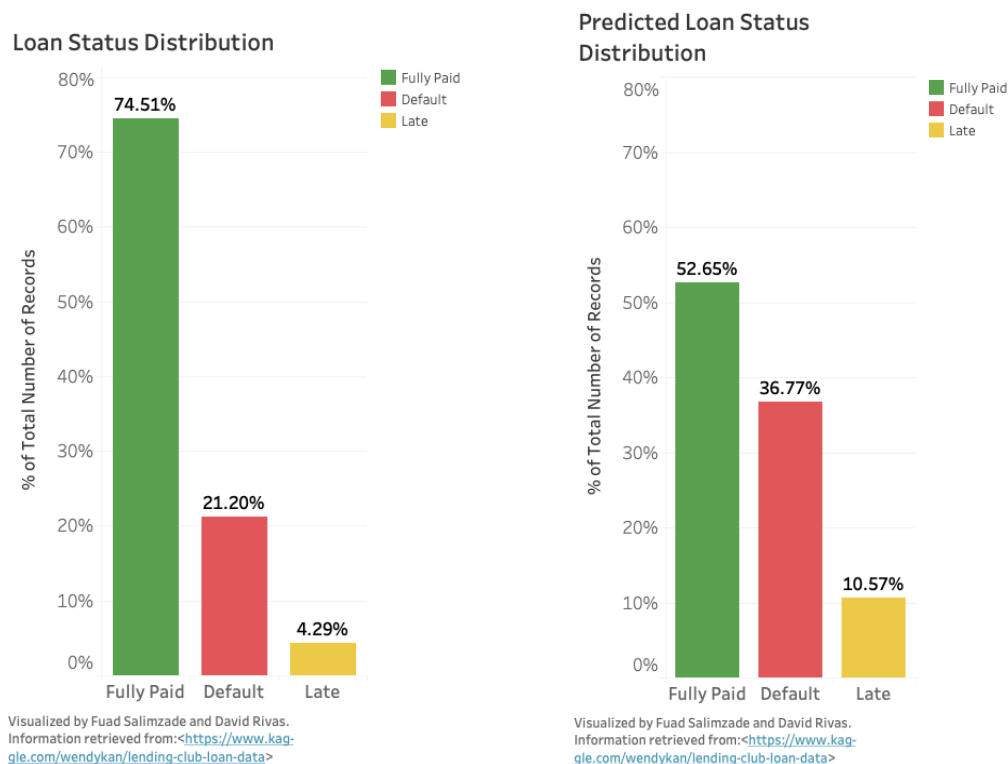


Image 6

The bar chart above depicts the differences in the of distribution of loan statuses between the results of the ‘Not-current’ loans and the predicted ‘Current’ loans. The predicted results were obtained through the ANN model.

In Image 6, the classifier predicts that the number of default and late loans will increase dramatically, around 15.57% in default and 6.28% in late, following a 21.86% decrease in fully paid loan rates. Several reasons can be proposed to reflect these changes in the loan status.

According to the information in Appendix 7, there is a shift towards lower interest rates in the current loans. The left skewness of the frequencies in the right-hand bar chart suggests that more loans were issued with lower interest rates which might explain the increase in the amount of total loans.

Appendix 8 shows the distribution of credit grades based on default loans. The previously mentioned pattern is also seen here, the percent of higher credit grades, especially A and B, that is predicted to default increases strikingly.

By looking into the visualized representation of both the ‘Not-current’ and predicted ‘Current’ loan statuses, it can be inferred that the decrease in the interest rates in the loans issued by LendingClub has led to an increase in the total number of loans, but more importantly, in the ratio of loans that will, predictably, default.

The prediction model built is a useful tool to gain insight into possible future losses for the company. Nevertheless, in accordance with LendingClub’s commitments of giving better opportunities to borrowers at low costs (About LendingClub, 2020), the model can be used to identify people whose financial livelihood is at risk due to a probable default and provide assistance to avoid this situation.

By using a similar model, LendingClub can identify its consumers’ risks and reform their operations to optimize profits and help their customers. From the results obtained, the suggestion would be to analyze the loan grading system more carefully, which would help the company by not misclassifying as many loans, and borrowers by not giving them an overoptimistic loan which would ultimately default.

## **Conclusion**

On the whole, the different prediction models built using Machine Learning algorithms show a great potential of applicability in today’s business context. By using the results given by the Artificial Neural Network model, a wide visualization of LendingClub’s loans was made which provided insight into its operations. The results ultimately raised flags in the current loans issued, which should help the company and clients to look for solutions before the predicted defaults become a reality.



## References

About LendingClub. (2020). LendingClub: About Us. Retrieved from:  
<https://www.lendingclub.com/company/about-us>

Investopedia. (2020). LendinClub Review. Retrieved from:  
<https://www.investopedia.com/lendingclub-review-4585233>

Kaggle. (2020). Lending Club Loan Data. Retrieved from:  
<https://www.kaggle.com/wendykan/lending-club-loan-data?select=loan.csv>

LendingClub. (2020). Alternative Investments: How it Works. Retrieved from:  
<https://www.lendingclub.com/investing/peer-to-peer>

Müller, A. C., & Guido, S. (2016). Introduction to machine learning with python: A guide for data scientists. Retrieved from <https://ebookcentral-proquest-com.esc-web.lib.cbs.dk:8443>

TowardsDataScience. (2019). Introduction to Artificial Neural Networks (ANN). Retrieved from:  
<https://towardsdatascience.com/introduction-to-artificial-neural-networks-ann-1aea15775ef9>

XGBoost Developers. (2020). XGBoost Tutorials: Introduction to Boosted Trees. Retrieved from: <https://xgboost.readthedocs.io/en/latest/tutorials/model.html>

# Appendices

## Appendix 1

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	
	id	member_id	loan_amnt	funded_amnt	funded_amnt_term	int_rate	installment	grade		sub_grade	emp_title		emp_length	home_ownership	annual_inc
1			2500	2500	2500 36 months	13.56	84.92 C			C1	Chef		10+ years	RENT	55000
2			30000	30000	30000 60 months	18.94	777.23 D			D2	Postmaster		10+ years	MORTGAGE	90000
3			5000	5000	5000 36 months	17.97	180.69 D			D1	Administrative		6 years	MORTGAGE	59280
4			4000	4000	4000 36 months	18.94	146.51 D			D2	IT Supervisor		10+ years	MORTGAGE	92000
5			30000	30000	30000 60 months	16.14	731.78 C			C4	Mechanic		10+ years	MORTGAGE	57250
6			5550	5550	5550 36 months	15.02	192.45 C			C3	Director COE		10+ years	MORTGAGE	152500
7			2000	2000	2000 36 months	17.97	72.28 D			D1	Account Manager		4 years	RENT	51000
8			6000	6000	6000 36 months	13.56	203.79 C			C1	Assistant Director		10+ years	RENT	65000
9			5000	5000	5000 36 months	17.97	180.69 D			D1	Legal Assistant III		10+ years	MORTGAGE	53580
10			6000	6000	6000 36 months	14.47	206.44 C			C2			< 1 year	OWN	300000
11			5500	5500	5500 36 months	22.35	211.05 D			D5			< 1 year	MORTGAGE	50000
12			28000	28000	28000 60 months	11.31	613.13 B			B3	Consultant		2 years	MORTGAGE	70000
13			11200	11200	11200 36 months	8.19	351.95 A			A4	Job Coach Supervisor		10+ years	MORTGAGE	65000
14			6500	6500	6500 36 months	17.97	234.9 D			D1	Quality Field Engineer		4 years	MORTGAGE	154000
15			22000	22000	22000 60 months	12.98	500.35 B			B5	Teller		10+ years	MORTGAGE	65000
16			3500	3500	3500 36 months	16.14	123.3 C			C4	respiratory therapist		10+ years	MORTGAGE	80000
17			7000	7000	7000 36 months	12.98	235.8 B			B5	Worship Director		4 years	MORTGAGE	102500
18			25000	25000	25000 60 months	16.91	620.11 C			C5	Processor		10+ years	MORTGAGE	23878
19			16000	16000	16000 60 months	20.89	431.87 D			D4	Neonatal Nurse Practitioner		4 years	MORTGAGE	120000
20			13000	13000	13000 60 months	14.47	305.67 C			C2	Stationary Engineer		10+ years	MORTGAGE	75000
21			10000	10000	10000 36 months	13.56	339.65 C			C1			< 1 year	MORTGAGE	65000
22			13000	13000	13000 36 months	14.47	447.29 C			C2	Exhibits director		10+ years	MORTGAGE	55000
23			9600	9600	9600 36 months	23.4	373.62 E			E1	driver coordinator		9 years	RENT	65000
24			3500	3500	3500 36 months	20.89	131.67 D			D4	gas attendant		10+ years	MORTGAGE	40000
25			16000	16000	16000 60 months	26.31	481.99 E			E4	Financial Relationship Associate		< 1 year	RENT	33000
26															

## Appendix 2

```
9 #Importing libraries
10 import pandas as pd
11 import numpy as np
12 import matplotlib.pyplot as plt
13
14 dataset = pd.read_csv("loan-2-notcurrent.csv")
15
16 X = dataset.iloc[:,0:89].values
17 y = dataset.iloc[:, -1].values
18
19 from sklearn.impute import SimpleImputer
20 imputer=SimpleImputer(missing_values= np.nan, strategy='constant',
21                       fill_value = 0)
22 X=imputer.fit_transform(X)
23
24 from sklearn.preprocessing import OneHotEncoder
25 from sklearn.compose import ColumnTransformer
26
27 #Credit grade transformation
28 ct = ColumnTransformer(transformers = [('encoder', OneHotEncoder(),[4])],
29                          remainder = 'passthrough',
30                          n_jobs = -1)
31 X = np.array(ct.fit_transform(X))
32
33 X = X[:,1:]
34
35 #House ownership transformation
36 ct = ColumnTransformer(transformers = [('encoder', OneHotEncoder(),[11])],
37                          remainder = 'passthrough',
38                          n_jobs = -1)
39 X = np.array(ct.fit_transform(X))
40
41 X = X[:,1:]
42
43 #Purpose transformation
44 ct = ColumnTransformer(transformers = [('encoder', OneHotEncoder(),[16])],
45                          remainder = 'passthrough',
46                          n_jobs = -1)
47 X = np.array(ct.fit_transform(X))
48
49 X = X[:,1:]
50
51 #State transformation
52 ct = ColumnTransformer(transformers = [('encoder', OneHotEncoder(),[21])],
53                          remainder = 'passthrough',
54                          n_jobs = -1)
55 X = np.array(ct.fit_transform(X))
56
57 X = X[:,1:]
58
59 #Date transformation
60 ct = ColumnTransformer(transformers = [('encoder', OneHotEncoder(),[69])],
61                          remainder = 'passthrough',
62                          n_jobs = -1)
63 X = np.array(ct.fit_transform(X))
64
65 X = X[:,1:]
66
67 #transforming y to categorical value
68 from sklearn.preprocessing import LabelEncoder
69 le=LabelEncoder()
70 y=le.fit_transform(y)
71
72 print(np.bincount(y))
73
74 from sklearn.preprocessing import OneHotEncoder
75 ohe = OneHotEncoder()
76 y = ohe.fit_transform(y.reshape(-1, 1)).toarray()
77
78
79 #Scalling the features
80 from sklearn.preprocessing import StandardScaler
81 sc = StandardScaler()
82 X = sc.fit_transform(X)
83
84 #Splitting data
85 from sklearn.model_selection import train_test_split
86 X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 0,
87                                                    test_size = 0.20)
88
```

## Appendix 3

```
103 """*****Implementing Neural Networks*****"""
104
105 import tensorflow
106 from tensorflow.keras.models import Sequential
107 from tensorflow.keras.layers import Dense
108
109
110 #implementing the model
111
112 classifier = Sequential()
113
114 #Adding input and first hidden layer
115
116 classifier.add(Dense(units = 117, activation = 'relu', input_dim = 173,
117                      kernel_initializer = 'glorot_uniform'))
118
119 #Adding hidden layer
120 classifier.add(Dense(units = 117, activation = 'relu',
121                      kernel_initializer = 'glorot_uniform'))
122
123 classifier.add(Dense(units = 117, activation = 'relu',
124                      kernel_initializer = 'glorot_uniform'))
125
126 #Adding other hidden layer
127 classifier.add(Dense(units=3, activation='softmax'))
128
129
130 #Compiler
131 classifier.compile(optimizer='adam', loss='categorical_crossentropy',
132                  metrics=['accuracy'])
133
134 #fitting the model
135
136 classifier.fit(X_train, y_train, batch_size=64, epochs=100)
137
138 classifier.summary()
139
140
141 #predicting y on X_test
142 y_pred=classifier.predict(X_test)
143
144
145 #Creating confusion matrix
146 from sklearn.metrics import confusion_matrix
147 cm = confusion_matrix(y_test.argmax(axis = 1), y_pred.argmax(axis = 1))
148
149
150 """*****Predicting new dataset on the model*****"""
151
152
153 current_data = pd.read_csv("loan-2-onlycurrent.csv")
154
155 X_new = current_data.values
156
157
158 #Predicting y_new
159 y_new = classifier.predict(X_new)
160
161 y_new = pd.DataFrame(y_new)
162
163 y_transformed = pd.DataFrame()
164
165 y_transformed = y_transformed.append(y_new)
166
167 y_transformed.columns = ["Fully Paid", "Default", "Late"]
168
169 y_transformed.to_csv("Predicted_results")
170
```

## Appendix 4

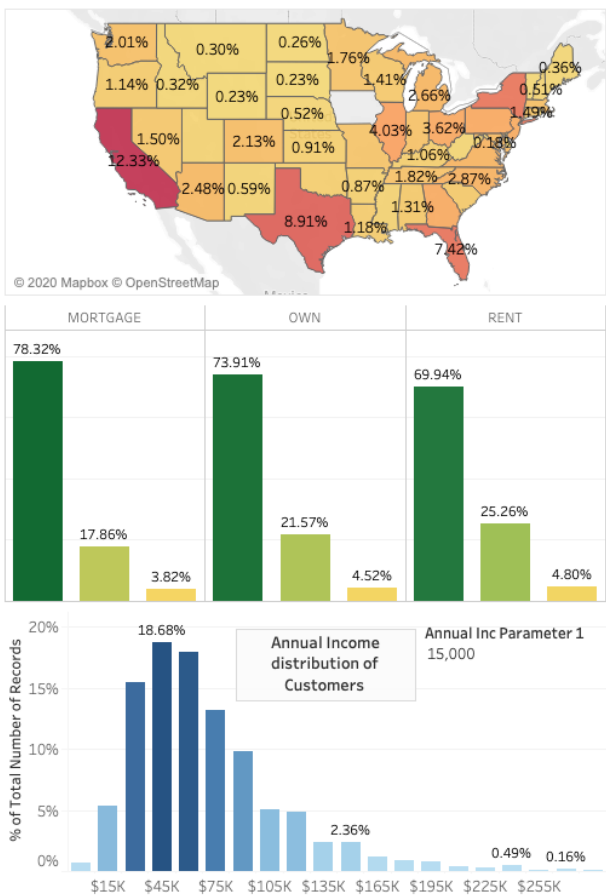
```
Epoch 85/100
355806/355806 [=====] - 16s 44us/sample - loss: 0.0017 -
accuracy: 0.9996
Epoch 86/100
355806/355806 [=====] - 14s 39us/sample - loss: 0.0022 -
accuracy: 0.9995
Epoch 87/100
355806/355806 [=====] - 15s 41us/sample - loss: 0.0019 -
accuracy: 0.9995
Epoch 88/100
355806/355806 [=====] - 11s 32us/sample - loss: 0.0017 -
accuracy: 0.9995
Epoch 89/100
355806/355806 [=====] - 11s 32us/sample - loss: 0.0020 -
accuracy: 0.9996
Epoch 90/100
355806/355806 [=====] - 11s 31us/sample - loss: 0.0018 -
accuracy: 0.9996
Epoch 91/100
355806/355806 [=====] - 11s 31us/sample - loss: 0.0016 -
accuracy: 0.9996
Epoch 92/100
355806/355806 [=====] - 11s 31us/sample - loss: 0.0016 -
accuracy: 0.9995
Epoch 93/100
355806/355806 [=====] - 11s 31us/sample - loss: 0.0014 -
accuracy: 0.9996
Epoch 94/100
355806/355806 [=====] - 11s 31us/sample - loss: 0.0022 -
accuracy: 0.9995
Epoch 95/100
355806/355806 [=====] - 14s 38us/sample - loss: 0.0015 -
accuracy: 0.9997
Epoch 96/100
355806/355806 [=====] - 12s 34us/sample - loss: 0.0016 -
accuracy: 0.9996
Epoch 97/100
355806/355806 [=====] - 15s 41us/sample - loss: 0.0024 -
accuracy: 0.9996
Epoch 98/100
355806/355806 [=====] - 15s 41us/sample - loss: 0.0021 -
accuracy: 0.9995
Epoch 99/100
355806/355806 [=====] - 16s 44us/sample - loss: 0.0013 -
accuracy: 0.9996
Epoch 100/100
355806/355806 [=====] - 13s 36us/sample - loss: 0.0016 -
accuracy: 0.9996
Out[7]: <tensorflow.python.keras.callbacks.History at 0x1b1788290>
```

## Appendix 5

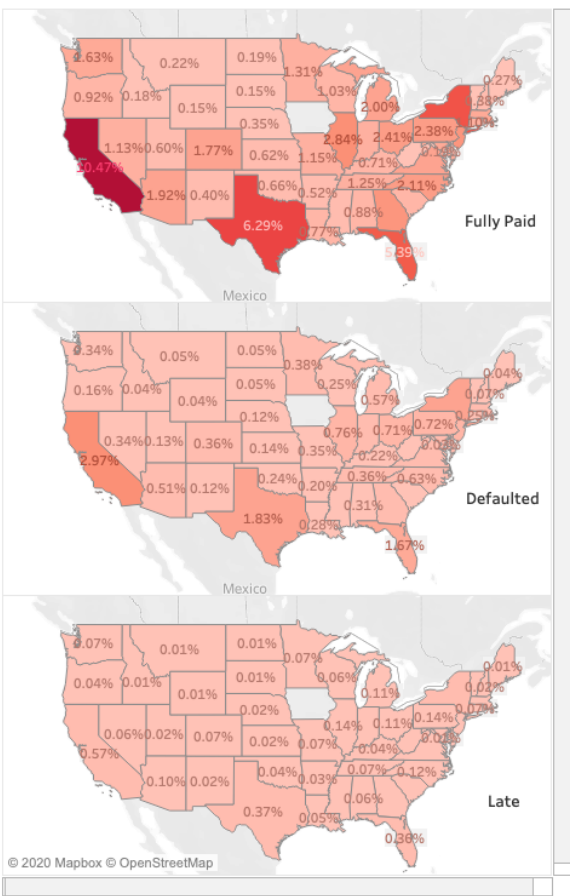
```
89 #Gaussian Classification
90 from sklearn.naive_bayes import GaussianNB
91 classifier = GaussianNB()
92 classifier.fit(X_train, y_train)
93 classifier.score(X_train, y_train)
94 classifier.score(X_test, y_test)
95
96 #XGBoost
97 import xgboost
98 classifier = xgboost.XGBClassifier()
99 classifier.fit(X_train, y_train)
100 classifier.score(X_train, y_train)
101 classifier.score(X_test, y_test)
102
103 #Random Forest
104 from sklearn.ensemble import RandomForestClassifier
105 classifier = RandomForestClassifier(n_estimators = 100, max_features = 'sqrt',
106                                   criterion = 'entropy', random_state = 0)
107 classifier.fit(X_train, y_train)
108 classifier.score(X_train, y_train)
109 classifier.score(X_test, y_test)
110
111 #LogisticRegression
112 from sklearn.linear_model import LogisticRegression
113 classifier = LogisticRegression(random_state = 0, n_jobs = -1)
114 classifier.fit(X_train, y_train)
115 classifier.score(X_train, y_train)
116 classifier.score(X_test, y_test)
117
118 from sklearn.model_selection import cross_val_score
119 cvs = cross_val_score(estimator = LogiR, X= X_train, y = y_train ,cv = 5)
120 print(cvs.mean())
121 print(cvs.std())
122
123 from sklearn.model_selection import GridSearchCV
124 c_param = {'C': [0.01, 0.1, 1, 10]}
125 grid = GridSearchCV(LogiR, param_grid= c_param)
126 grid.fit(X_train, y_train)
127 grid.score(X_train, y_train)
128 grid.score(X_test, y_test)
129
```

# Appendix 6

Total monthly debt to income ratio

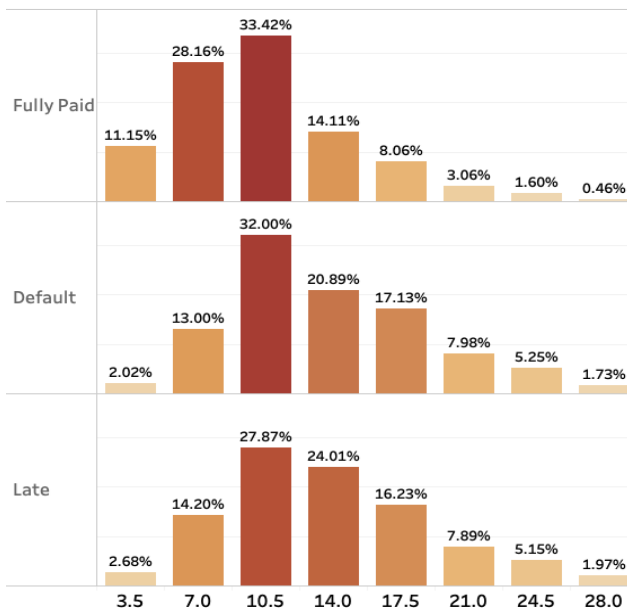


Loan Status by state



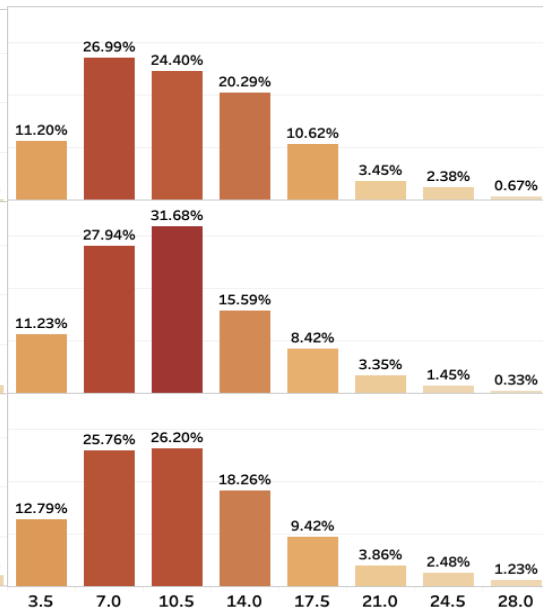
## Appendix 7

Interest Rate Distribution



Visualized by Fuad Salimzade and David Rivas. Information retrieved from: <https://www.kaggle.com/wendykan/lending-club-loan-data>

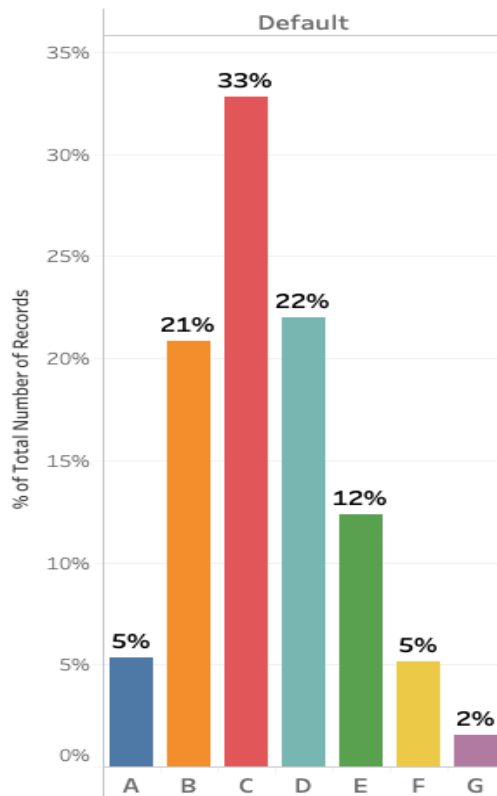
Interest Rate Distribution (Predicted)



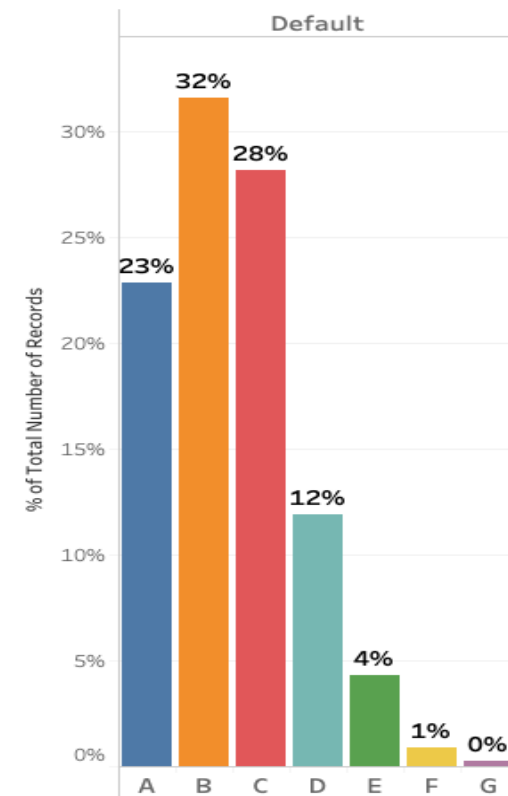
Visualized by Fuad Salimzade and David Rivas. Information retrieved from: <https://www.kaggle.com/wendykan/lending-club-loan-data>

## Appendix 8

Predicted



Visualized by Fuad Salimzade and David Rivas. Information retrieved from: <https://www.kaggle.com/wendykan/lending-club-loan-data>



Visualized by Fuad Salimzade and David Rivas. Information retrieved from: <https://www.kaggle.com/wendykan/lending-club-loan-data>