NORTHEASTERN UNIVERSITY

INFORMATION SYSTEMS SPRING ’20

Prediction Analysis of Interest Rate Using Machine Learning

DATA SCIENCE ENGINEERING METHODS AND TOOLS

INFO 6105 – 37555

Guided By:

Prof. Handan Liu

Created By:

Group 15

TEAM MEMBERS

Sr. No.	Name	Email ID	NUID
1	Akash Dubey	dubey.ak@northeastern.edu	001342165
2	Jayshil Jain	jain.ja@husky.neu.ed	001302139
3	Priyanka Malpekar	malpekar.p@husky.neu.edu	001302741
4	Siddhi Prabhu	prabhu.si@husky.neu.edu	001342165
5	Tanvi Gurav	gurav.t@husky.neu.edu	001306848

ACKNOWLEDGEMENT

It gives us great pleasure and immense satisfaction in presenting the project “Prediction Analysis of Interest Rate Using Machine Learning”. We would like to take this opportunity to acknowledge the innumerable guidance and support extended to us by our Professor Handan Liu in the execution and preparation of the project.

We would like to thank our TA's Mr. Vikas Kumar and Ms. Mingu Liu who has helped us with their valuable suggestions and guidance. Finally, we are intended to our very own Northeastern University for giving us the platform to express and exhibit our talent.

Any bouquets for the merit of this project should go to above mentioned persons.

TABLE OF CONTENTS

INTRODUCTION……………………………………………………………………………….………5

OVERVIEW……..…………………………………………………………………………….………8

OBJECTIVES….……………………………………………………………………….….……….9

MOTIVATION………………………………………………………………………..……..….10

RELATED WORK…………………………………………………………………..………….11

DATASET OVERVIEW……………………………………………………………….………12

ANALYZING THE DATA……………………………………………………….……………13

METHODOLOGY…………………………………………………………………….…….22

MODELS AND ALGORITHMS USED…………………………………………….………27

CALCULATIONS……………………………………………………………………….……..39

VISUALIZATIONS……………………………………………………………………….………42

FINAL RESULTS & CONCLUSION…………………………………………….……………46

FUTURE WORK………………………………………………………………………….………47

KEY TAKEAWAYS FOR USERS……………………………………………………………….48

DEFINITIONS AND ABBREVIATIONS………………………………………………………..49

REFERENCES…………………………………………………………………….……...…50

INTRODUCTION

Lending Club offers loans of various grades they assign that correspond to specific interest rates for investors. It is a peer-to-peer lending company, the largest of its kind in the world. Lending club aims to operate a platform at low cost to offer interest rates to our borrower members lower than the rates they could obtain through credit cards or traditional banks. The higher the interest rate, the riskier the grade. The risk comes in the form of defaults - whenever a loan defaults, investors end up losing a portion of their investment. On the basis of the borrower's credit score, credit history, desired loan amount and the borrower's debt-to-income ratio, Lending Club determines whether the borrower is credit worthy and assigns to its approved loans a credit grade that determines payable interest rate and fees. [1]

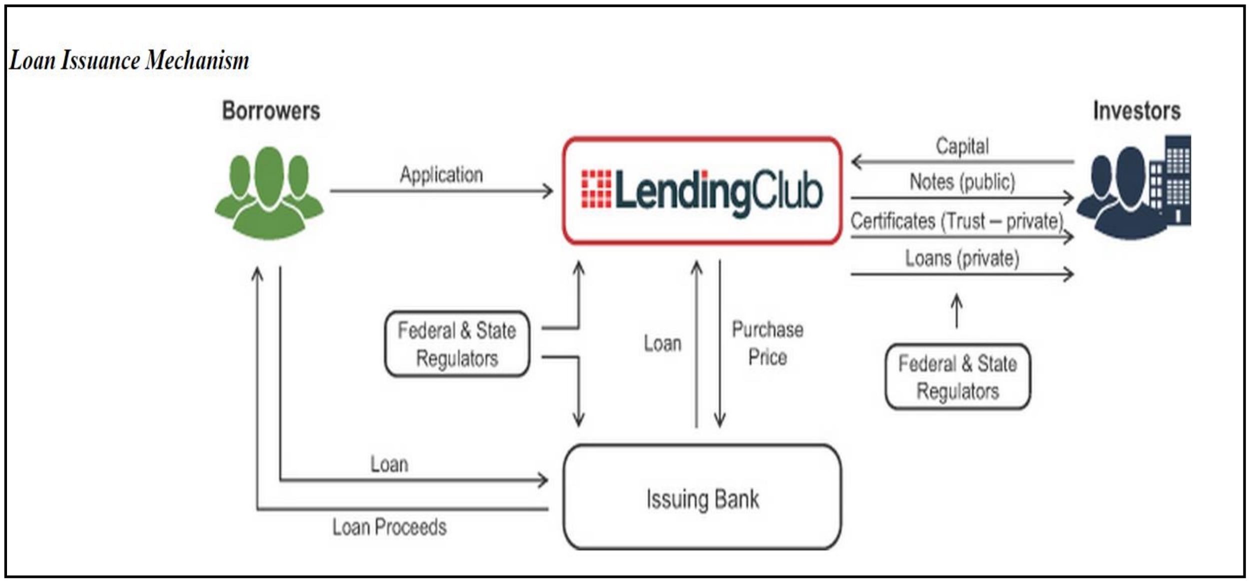


Figure 1: LendingClub Mechanism
(Source: http://blogs.ft.com/lex-live/liveblogs/2014-12-10-2/)

We believe that there is inherent variation between loans in a grade, and that we can use machine learning techniques to determine and avoid loans that are predicted to default. The Lending Club dataset contains a comprehensive list of features that we can employ to train our model for prediction. The dataset includes detailed information for every loan issued by Lending Club from 2007 to 2015. We will train and test a range of models to identify the best performing model. [2]

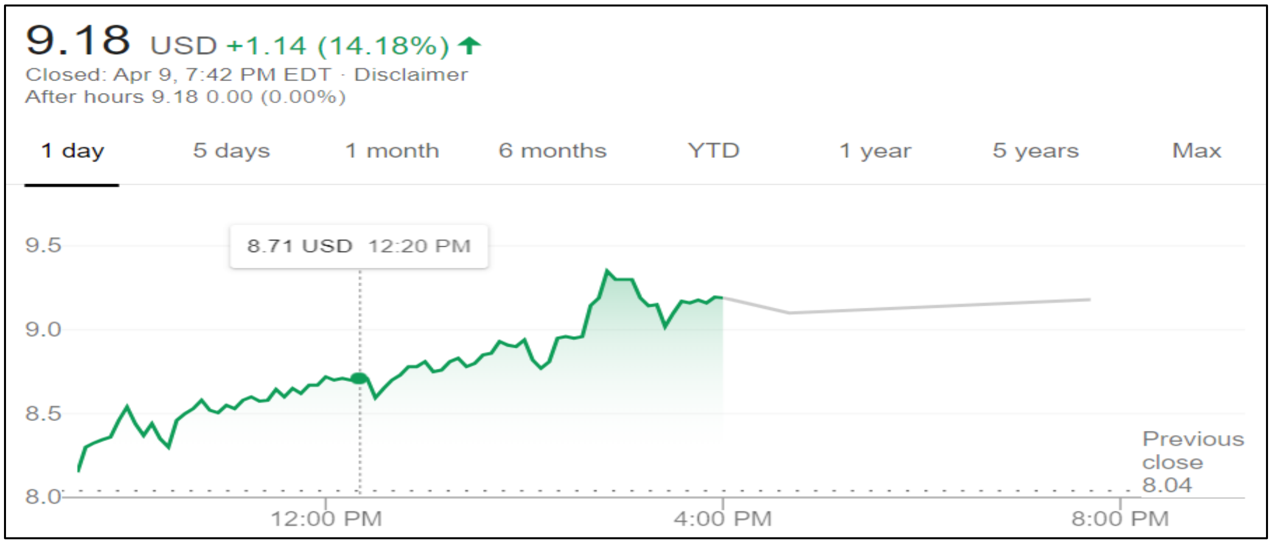


Figure 2: Lending Club Current Stock Data
(Source: https://finance.yahoo.com/quote/LC/)

LendingClub probably isn't the best option for borrowers with bad credit. That would bring a high interest rate and steep origination fee, meaning you could probably do better with a different type of loan. LendingClub requirements generally get high marks, but they might not be for everyone. [3] Here are some pros and cons that might help clarify the advantages and disadvantages:

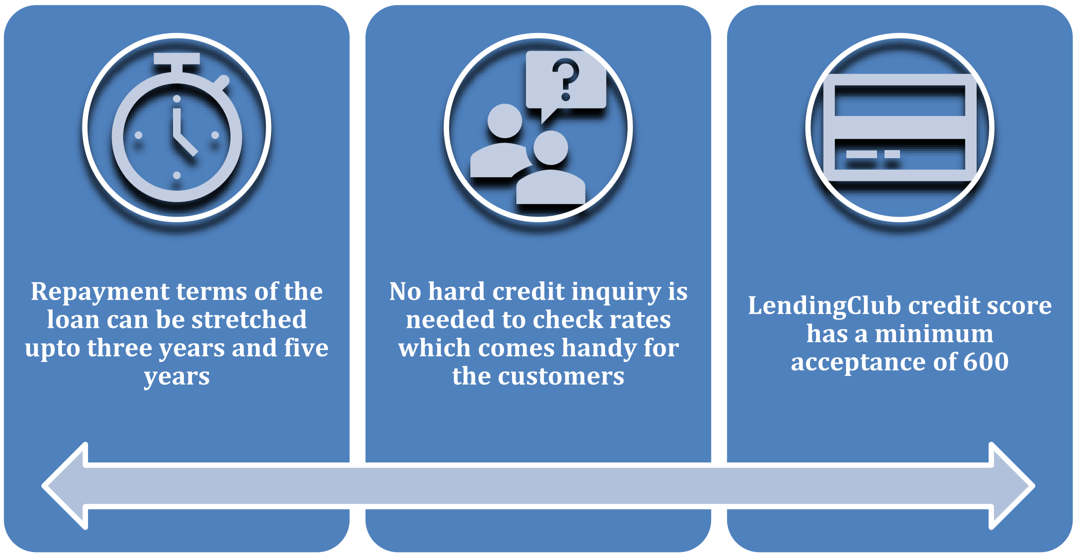

Repayment terms of the loan can be stretched upto three years and five years
No hard credit inquiry is needed to check rates which comes handy for the customers
LendingClub credit score has a minimum acceptance of 600

Figure 3: Pros of Lending Club
(Source: https://www.debt.org/credit/loans/personal/lending-club-review/)

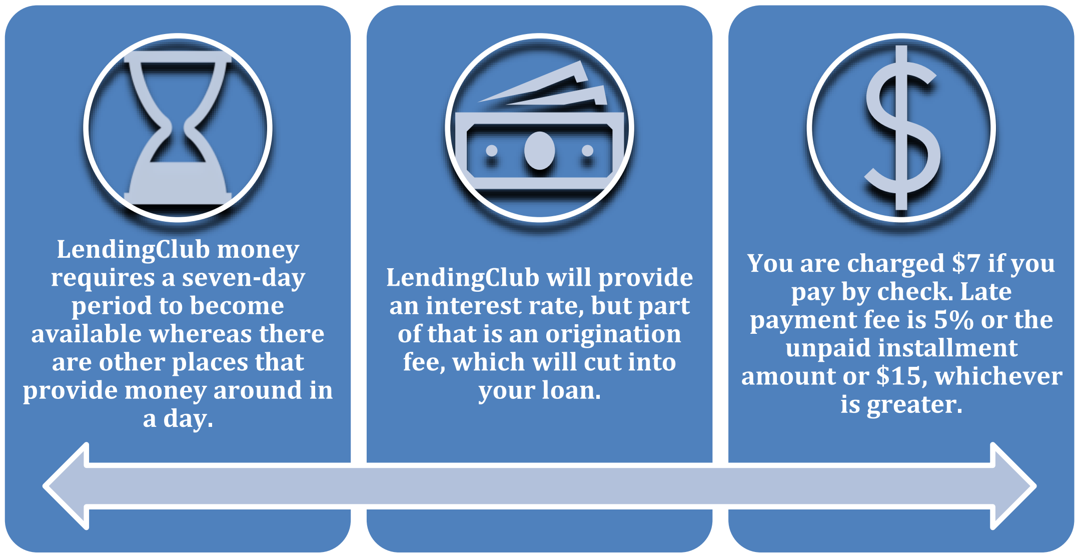

LendingClub money requires a seven-day period to become available whereas there are other places that provide money around in a day.
LendingClub will provide an interest rate, but part of that is an origination fee, which will cut into your loan.
You are charged $7 if you pay by check. Late payment fee is 5% or the unpaid installment amount or $15, whichever is greater.

Figure 4: Cons of Lending Club
(Source: https://www.debt.org/credit/loans/personal/lending-club-review/)

OVERVIEW

LendingClub is a US peer-to-peer lending company. The company asserted that $15.98 billion in loans had been arising through its platform up to December 31,2015. The Lending Club enables borrowers to possess unsecured personal loans between the range of $1,000 and $40,000. The standard loan period is of three years. Thereby, all the Investors can search and find the listing of loans on the website of Lending Club and therefore choose loans that they wish to invest in based on the information supplied about the borrower, the amount, grade, and the purpose of the loan. Investors make money from interest. Lending Club makes a profit by charging the borrowers a fee called the origination fee and for investors a service fee. With the release of Lending club data, a lot of researchers and practitioners are leveraging this Kaggle dataset to understand credit risk, borrowing patterns, investment opportunities and to understand consumer choice and behaviors. [4]

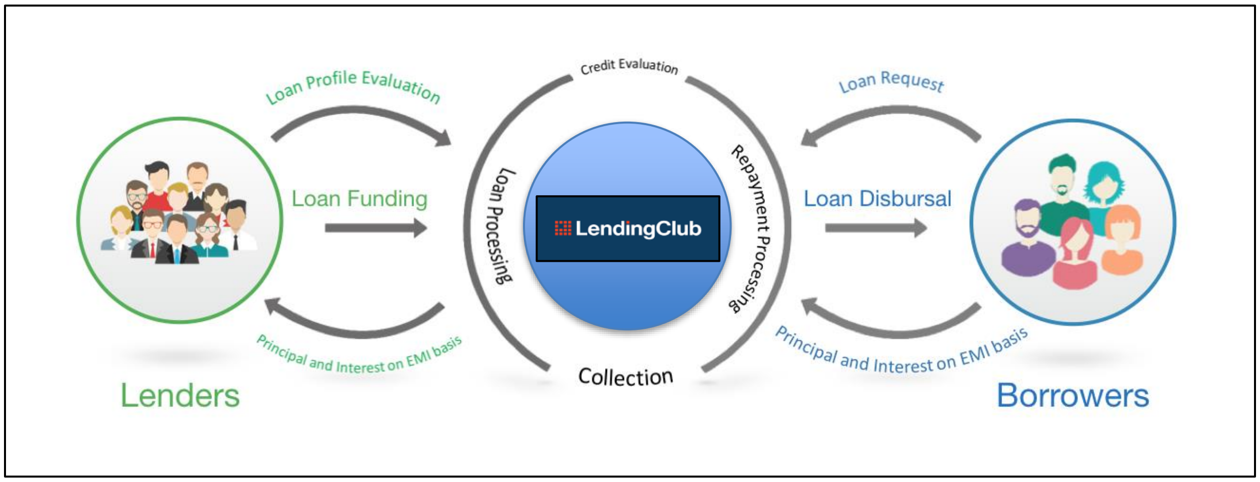


Figure 5: LendingClub Logo and Mechanism
(Source: https://www.lendingclub.com/)

OBJECTIVES

1. To work on loan predictions using machine learning models and provide a model that caters their needs.
2. To provide a full funded loan from lending club to clients
3. To predict the lowest possible interest rates
4. To avoid loans that are predicted to default
5. To get a desired loan duration
6. To provide an online lending platform where borrowers are able to obtain loans and investors can purchase notes backed by payments based on loans
7. To predict the expected returns for loans to a given borrower
8. To maximize our returns by predicting the probability of default of the borrower so as to help avoid investment in those high-risk notes [5]

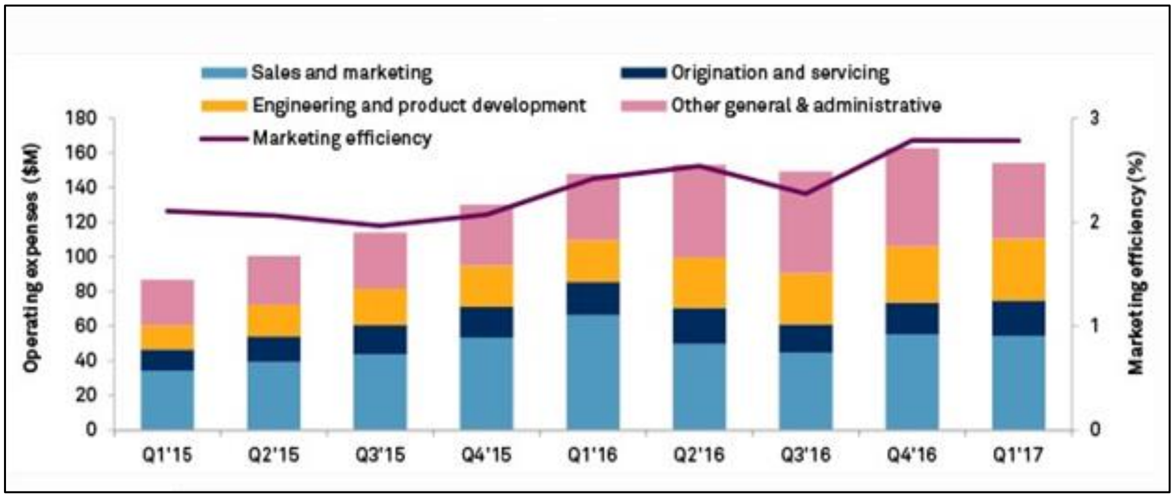


Figure 6: LendingClub Operating Expenses & Sales & Marketing Efficiency
Source: https://www.spglobal.com/en/research-insights/articles/breaking-down-digital-lenders-profitability-plans

MOTIVATION

With the rising popularity of peer-to-peer lending platforms in recent years, investors now have easy access to this alternative investment asset class by lending money to individual borrowers through platforms such as LendingClub, Prosper Marketplace, and Upstart, or to small businesses through Funding Circle. The process starts with borrowers submitting loan applications to the platform, which performs credit reviews and either approves or denies each application.

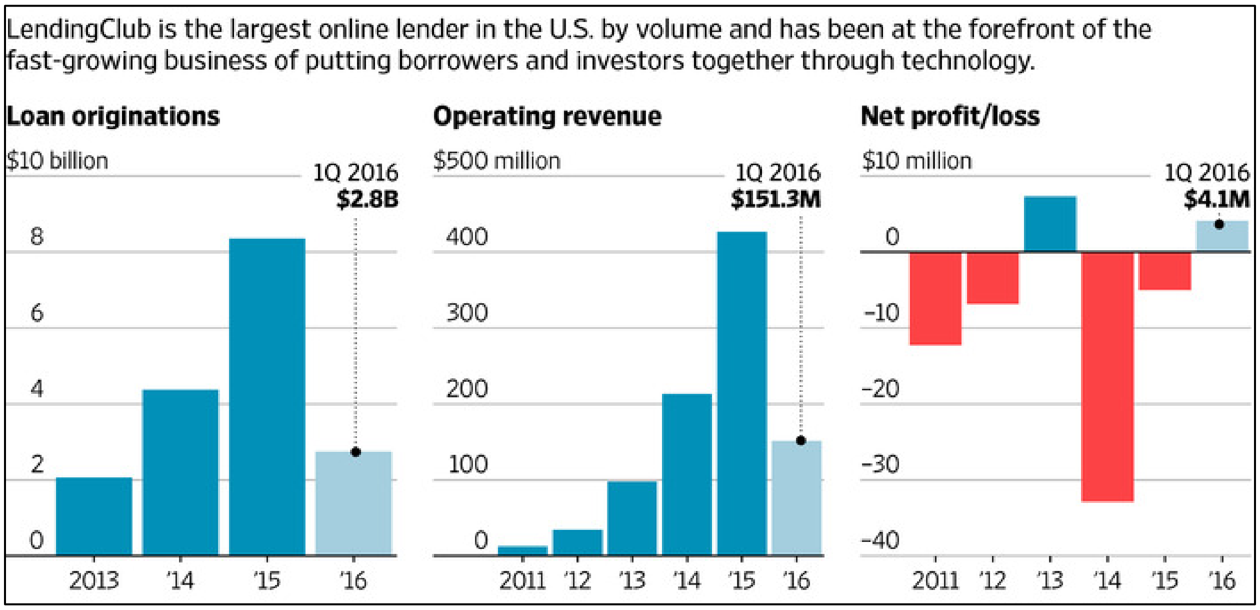


Figure 7: Loan Origination, Operating Revenue, Net Profit/Loss
(Source: https://www.wsj.com/articles/inside-the-final-days-of-lendingclub-ceo-renaud-laplanche-1463419379)

The platform also uses a proprietary model to determine the interest rate of approved loans based on the creditworthiness of borrowers. Approved loans are then listed on the platform for investor funding. Investors usually want to diversify their portfolio by only investing a small amount, e.g. $25, in each loan. Hence, it is desirable for investors to be able to independently evaluate the credit risk of many listed loans quickly and invest in those with lower perceived risks. This motivates us to build machine-learned classification and regression models that can quantify the credit risk with a LendingClub historical loan dataset. Specifically, we build and evaluate classifiers that predict whether a given loan will be fully paid by the borrower, as well as regressors that predict the annualized net return from investment in a given loan. Finally, we simulate and evaluate a simple loan selection strategy by investing in loans that pass a certain regression prediction threshold. [6]

RELATED WORK

Prior projects like "Predicting borrowers chance of defaulting on credit loans" have set great examples of applying machine learning to improve loan default prediction in a Kaggle competition, and authors for "Predicting Probability of Loan Default" have shown that Random Forest appeared to be the best performing model on the Kaggle data. However, despite of the early success using Random Forest for default prediction, real-world records often behaves differently from curated data, and a later study "Peer Lending Risk Predictor" presented that a modified Logistic Regression model could outperform SVM, Naive Bayes, and even Random Forest on Lending Club data. The fact that Logistic Regression performance could be immensely improved by simply adding penalty factor on misclassification gave rise to our interest in fine tuning other not-yet optimized models, in particular, SVM and Naive Bayes, to continue the search for a better predictive model in the realm of loan default. [7] Besides the difference in types of model that they focus on, the prior studies only used the out-of-the-box dataset from Kaggle or Lending Club, but research like "The sensitivity of the loss given default rate to systematic risk" has shown the linkage between default rate and macroeconomic factors, so we have decided to add in census data, with info like regional median income, to train our models on a more holistic set of features.

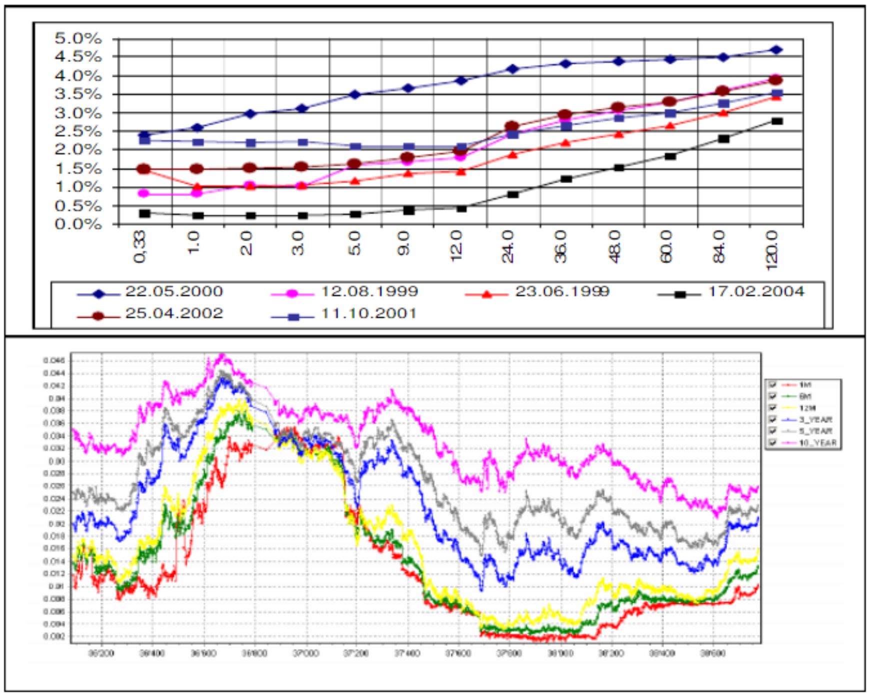


Figure 8: Examples of interest rate curves (top) & time series for some maturities (bottom) (Source:https://www.researchgate.net/publication/221165703_Machine_learning_analysis_and_modeling_of_interest_rate_curves)

DATASET OVERVIEW

(A) Specification of the Data Set

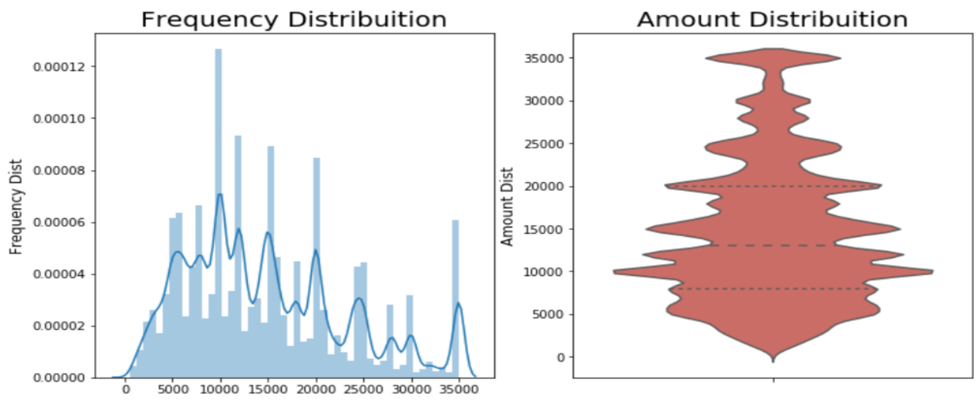
1. We have 74 columns of data out of which 20 are Categorical Variables and 54 are Numeric Variables
2. 38 of 74 columns allow NULL values
3. Most Categorical Variables are Descriptions while others are Date i.e. Days of Week and Months
4. Some Numeric Variables are derived based on the values of Other Numeric Variables

(B) Data Source

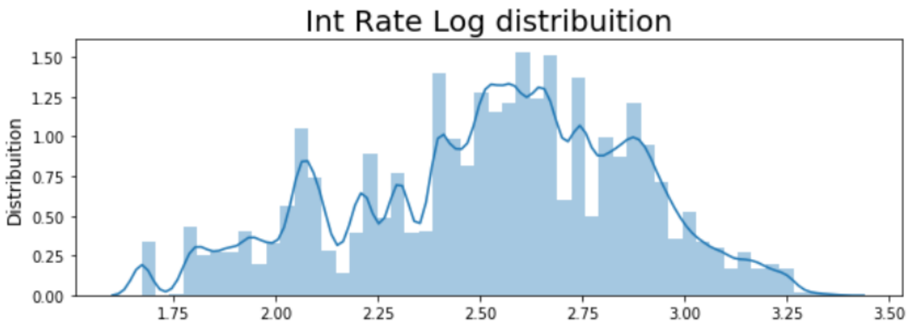
https://www.kaggle.com/wendykan/lending-club-loan-data

These files contain the latest payment information and all the necessary and appropriate loan data for all the loans issued through 2001-2015 which also includes current loan status (Current, Fully Paid, Late, etc.) The file containing loan data through the "present" contains complete loan data for all loans which are issued through the previous completed calendar quarter. There are certain additional features that include credit scores, address including zip codes and state, number of finance inquiries and various other collections. The file is a matrix that includes a total of 75 variables and about 890 thousand observations. Also, a data dictionary is placed in a different file.

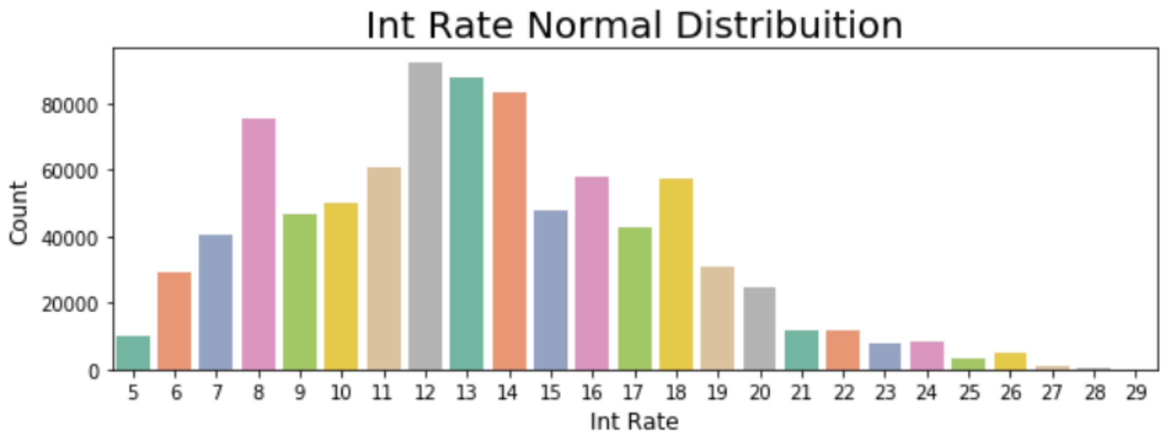
Visualization 1: Distribution of Loan Amount



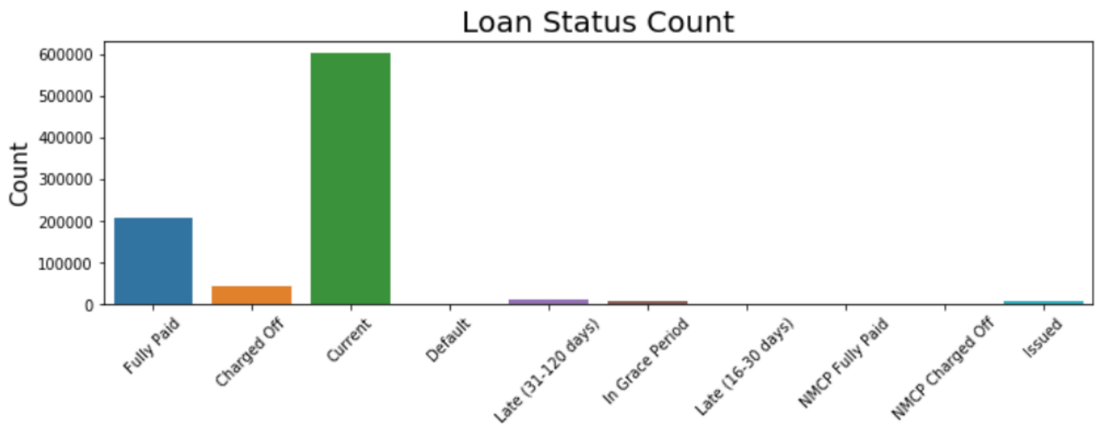
Visualization 2: Distribution of Interest Rate (Log Distribution)

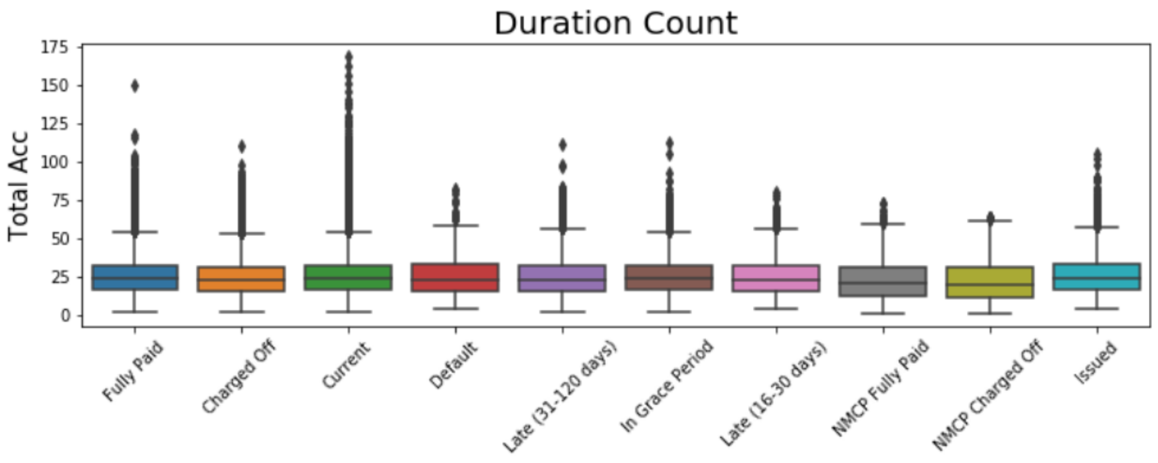


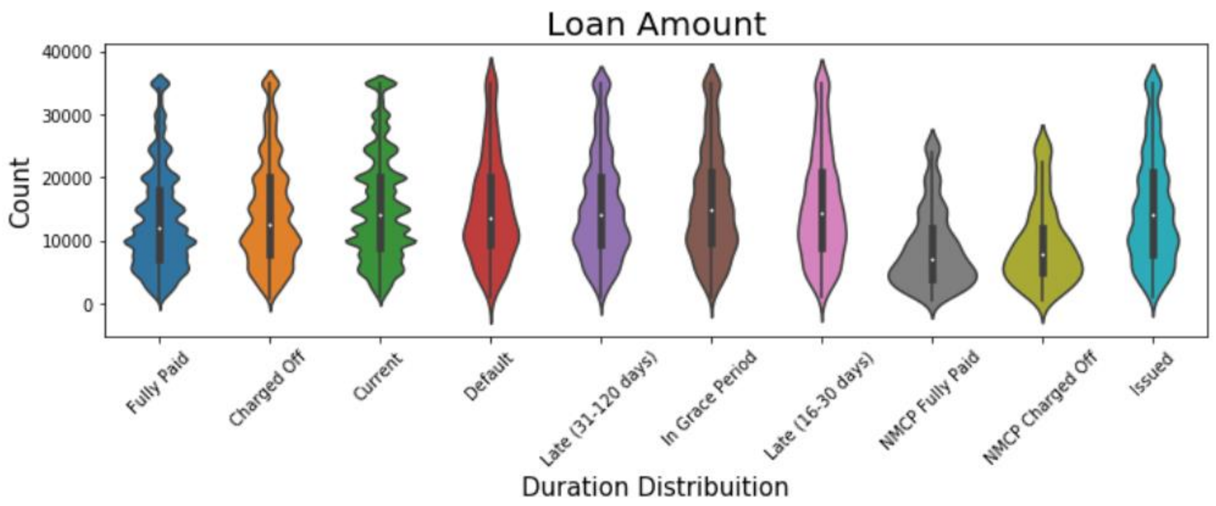
Visualization 3: Distribution of Interest Rate (Normal Distribution)



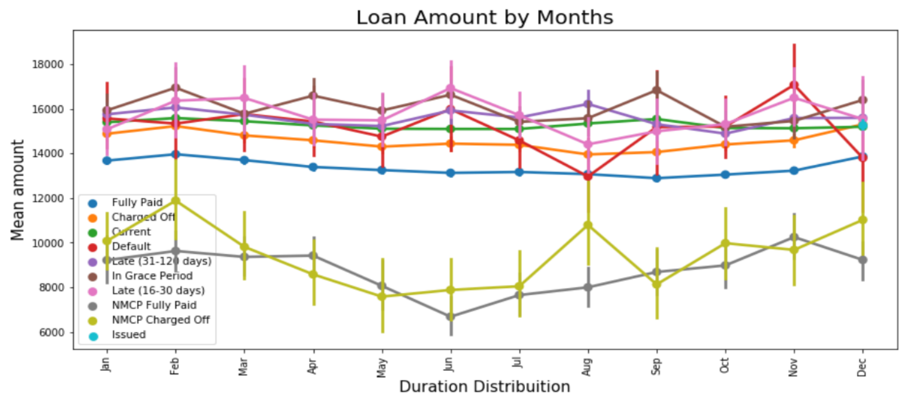
Visualization 4: Distribution of Loan Status Count, Duration Count, Loan Amount Count



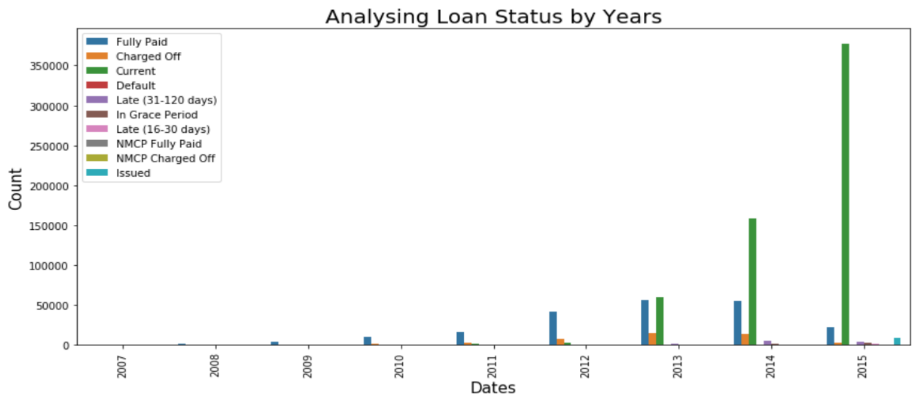




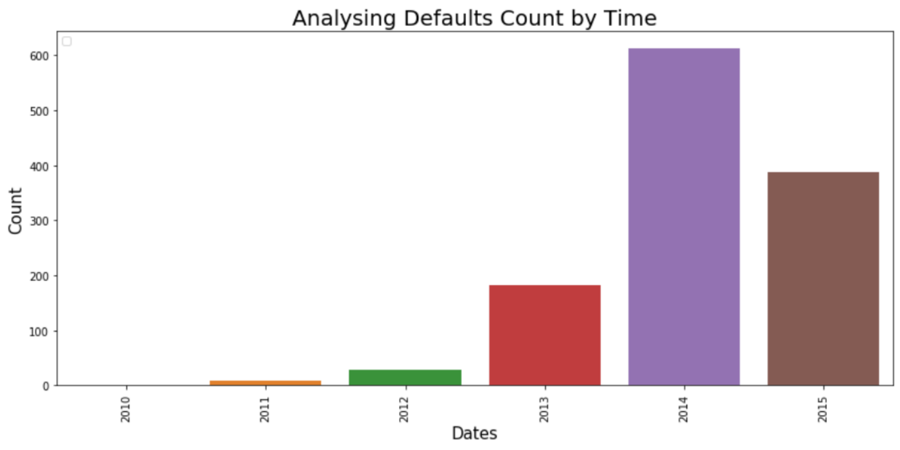
Visualization 5: Distribution of Issue Date



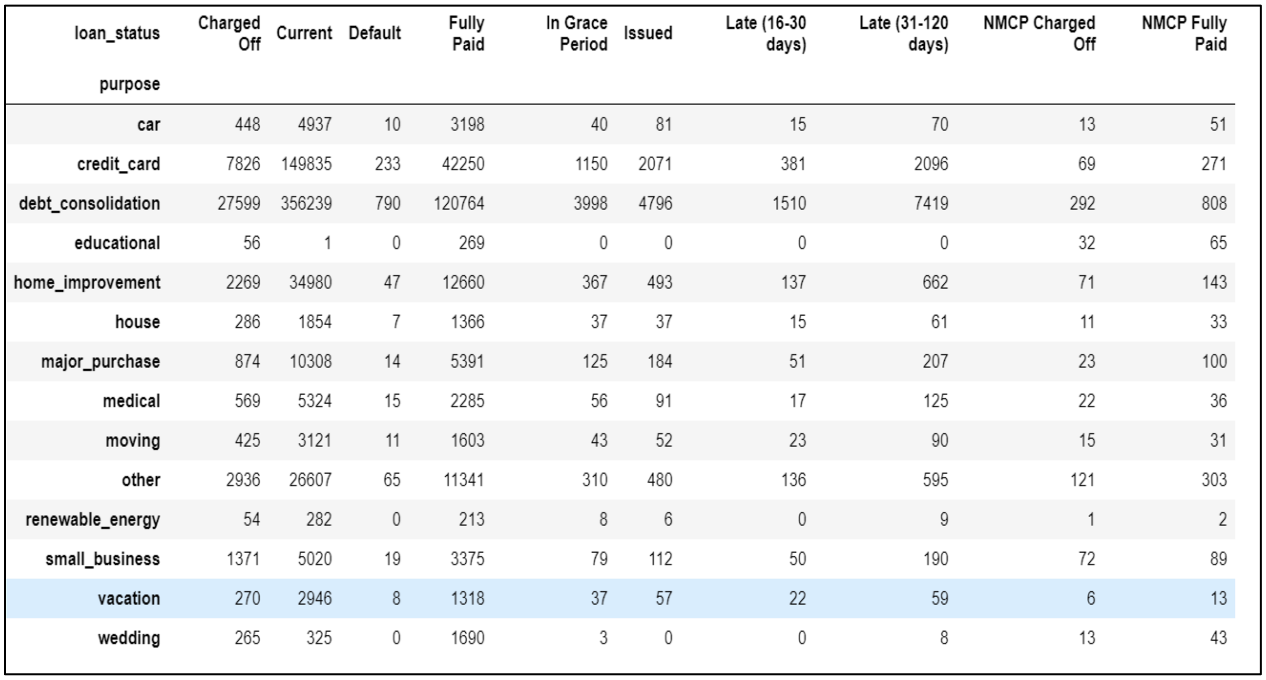
Visualization 6: Analyzing Loan Status by Years



Visualization 7: Analyzing Defaults Count by Time

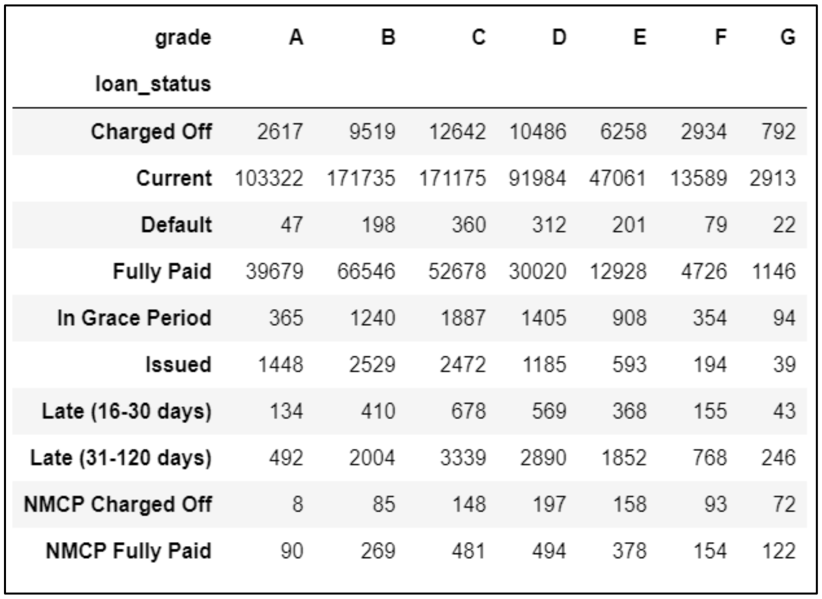


Visualization 8: Generating cross tab for Loan Status using heat map



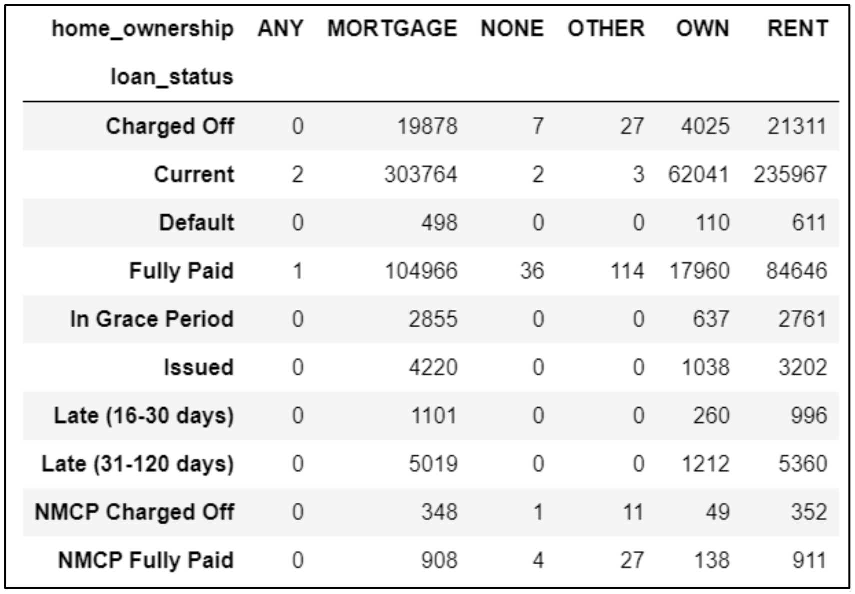
loan_status	Charged Off	Current	Default	Fully Paid	In Grace Period	Issued	Late (16-30 days)	Late (31-120 days)	NMCP Charged Off	NMCP Fully Paid
purpose										
car	448	4937	10	3198	40	81	15	70	13	51
credit_card	7826	149835	233	42250	1150	2071	381	2096	69	271
debt_consolidation	27599	356239	790	120764	3998	4796	1510	7419	292	808
educational	56	1	0	269	0	0	0	0	32	65
home_improvement	2269	34980	47	12660	367	493	137	662	71	143
house	286	1854	7	1366	37	37	15	61	11	33
major_purchase	874	10308	14	5391	125	184	51	207	23	100
medical	569	5324	15	2285	56	91	17	125	22	36
moving	425	3121	11	1603	43	52	23	90	15	31
other	2936	26607	65	11341	310	480	136	595	121	303
renewable_energy	54	282	0	213	8	6	0	9	1	2
small_business	1371	5020	19	3375	79	112	50	190	72	89
vacation	270	2946	8	1318	37	57	22	59	6	13
wedding	265	325	0	1690	3	0	0	8	13	43

Visualization 9: Generating cross tab for Grade using heat map



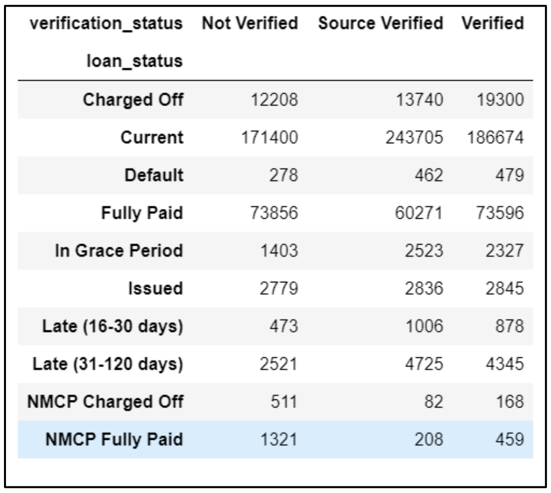
grade	A	B	C	D	E	F	G
loan_status							
Charged Off	2617	9519	12642	10486	6258	2934	792
Current	103322	171735	171175	91984	47061	13589	2913
Default	47	198	360	312	201	79	22
Fully Paid	39679	66546	52678	30020	12928	4726	1146
In Grace Period	365	1240	1887	1405	908	354	94
Issued	1448	2529	2472	1185	593	194	39
Late (16-30 days)	134	410	678	569	368	155	43
Late (31-120 days)	492	2004	3339	2890	1852	768	246
NMCP Charged Off	8	85	148	197	158	93	72
NMCP Fully Paid	90	269	481	494	378	154	122

Visualization 10: Generating cross tab for Grade using heat map



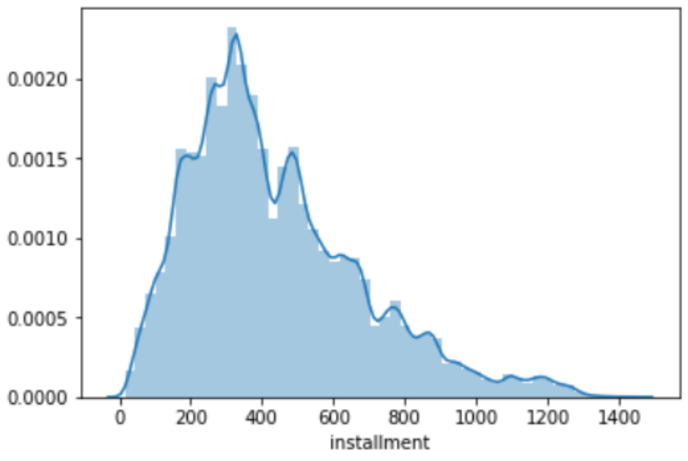
home_ownership loan_status	ANY	MORTGAGE	NONE	OTHER	OWN	RENT
Charged Off	0	19878	7	27	4025	21311
Current	2	303764	2	3	62041	235967
Default	0	498	0	0	110	611
Fully Paid	1	104966	36	114	17960	84646
In Grace Period	0	2855	0	0	637	2761
Issued	0	4220	0	0	1038	3202
Late (16-30 days)	0	1101	0	0	260	996
Late (31-120 days)	0	5019	0	0	1212	5360
NMCP Charged Off	0	348	1	11	49	352
NMCP Fully Paid	0	908	4	27	138	911

Visualization 11: Generating cross tab for Grade using heat map

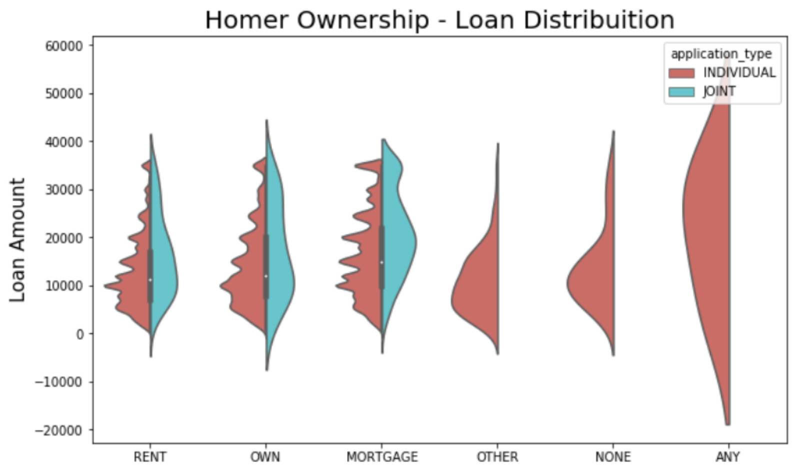


verification_status loan_status	Not Verified	Source Verified	Verified
Charged Off	12208	13740	19300
Current	171400	243705	186674
Default	278	462	479
Fully Paid	73856	60271	73596
In Grace Period	1403	2523	2327
Issued	2779	2836	2845
Late (16-30 days)	473	1006	878
Late (31-120 days)	2521	4725	4345
NMCP Charged Off	511	82	168
NMCP Fully Paid	1321	208	459

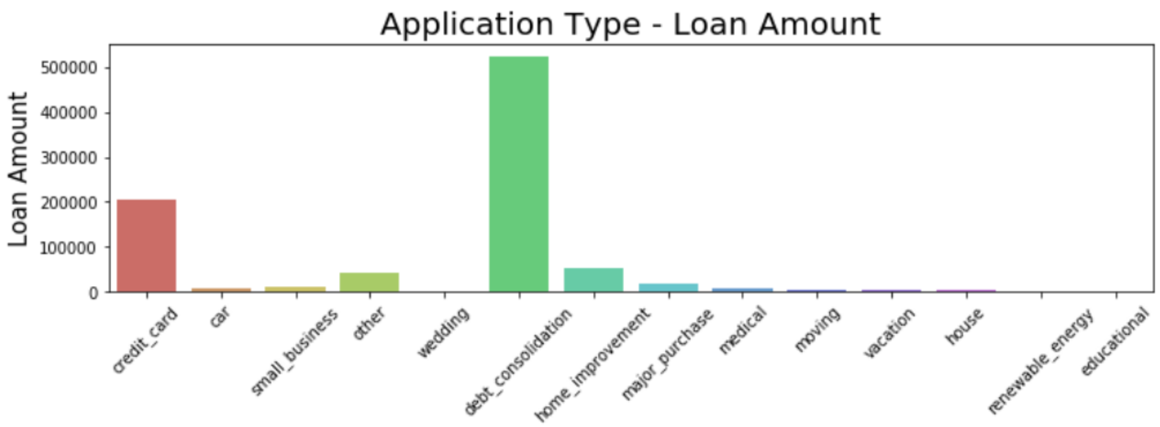
Visualization 12: Understanding the installment column



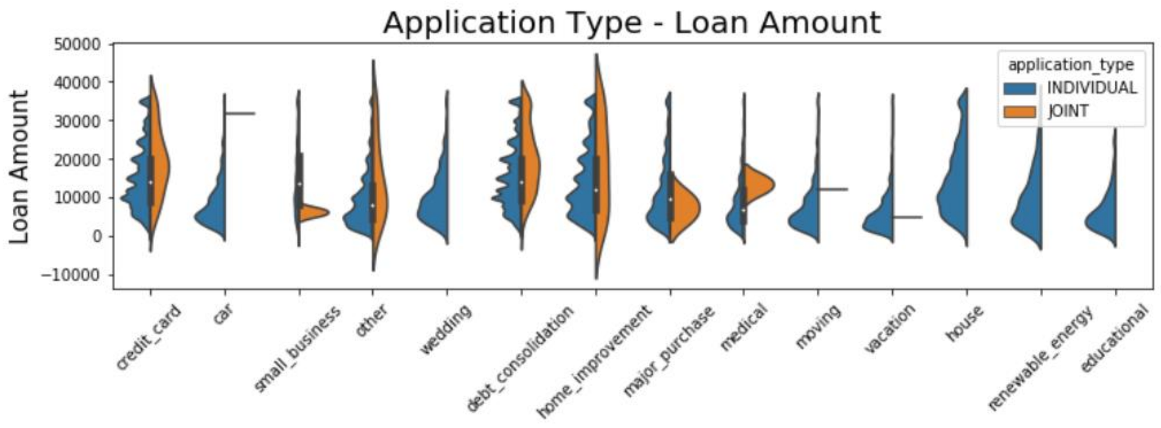
Visualization 13: Looking up Home Ownership (Loan Distribution) by Loan Amount



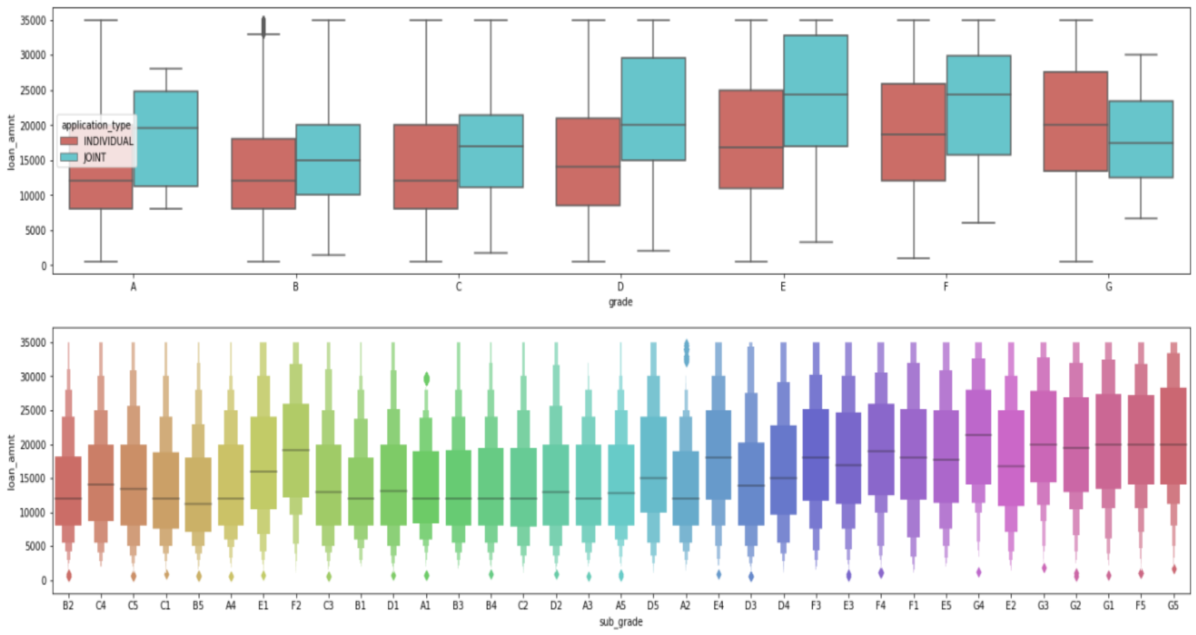
Visualization 14: Distribution of Purpose



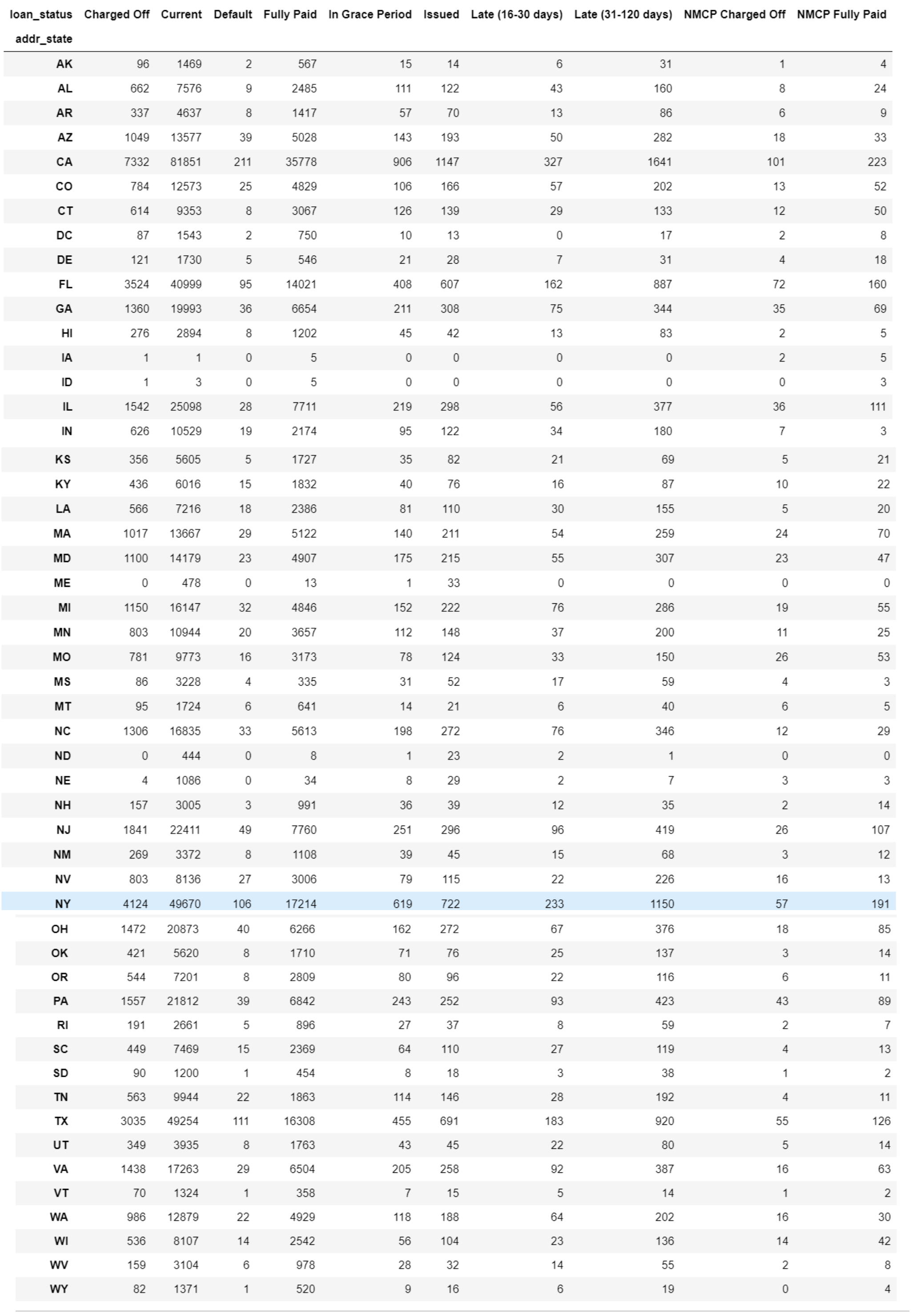
Visualization 15: Analyzing Loan Amount by Application Type



Visualization 16: Analyzing Grades

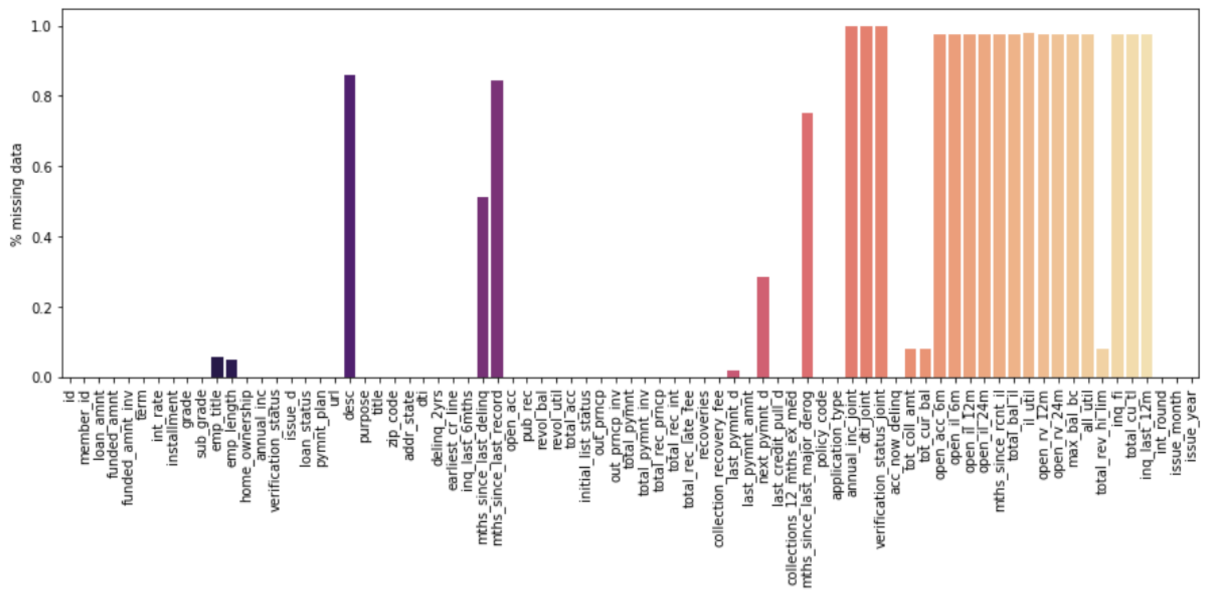


Visualization 17: Heatmap of Address State vs Loan Status

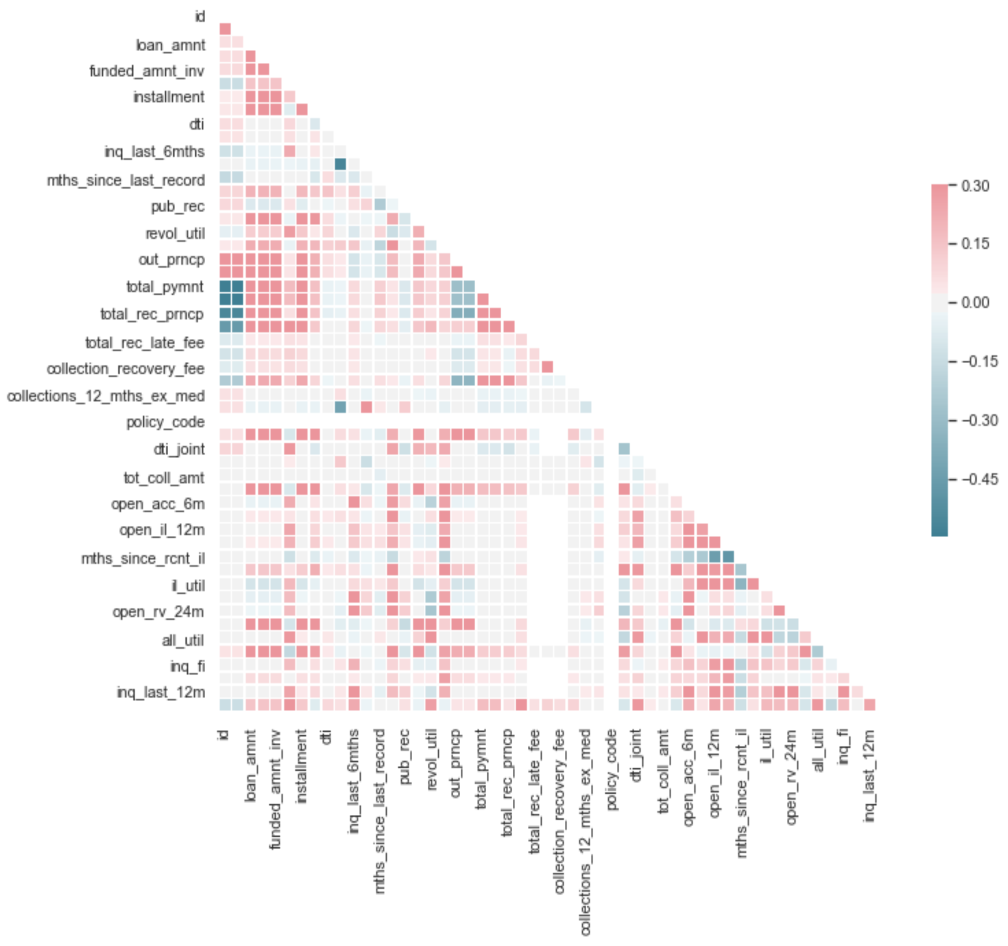


loan_status addr_state	Charged Off	Current	Default	Fully Paid	In Grace Period	Issued	Late (16-30 days)	Late (31-120 days)	NMCP Charged Off	NMCP Fully Paid
AK	96	1469	2	567	15	14	6	31	1	4
AL	662	7576	9	2485	111	122	43	160	8	24
AR	337	4637	8	1417	57	70	13	86	6	9
AZ	1049	13577	39	5028	143	193	50	282	18	33
CA	7332	81851	211	35778	906	1147	327	1641	101	223
CO	784	12573	25	4829	106	166	57	202	13	52
CT	614	9353	8	3067	126	139	29	133	12	50
DC	87	1543	2	750	10	13	0	17	2	8
DE	121	1730	5	546	21	28	7	31	4	18
FL	3524	40999	95	14021	408	607	162	887	72	160
GA	1360	19993	36	6654	211	308	75	344	35	69
HI	276	2894	8	1202	45	42	13	83	2	5
IA	1	1	0	5	0	0	0	0	2	5
ID	1	3	0	5	0	0	0	0	0	3
IL	1542	25098	28	7711	219	298	56	377	36	111
IN	626	10529	19	2174	95	122	34	180	7	3
KS	356	5605	5	1727	35	82	21	69	5	21
KY	436	6016	15	1832	40	76	16	87	10	22
LA	566	7216	18	2386	81	110	30	155	5	20
MA	1017	13667	29	5122	140	211	54	259	24	70
MD	1100	14179	23	4907	175	215	55	307	23	47
ME	0	478	0	13	1	33	0	0	0	0
MI	1150	16147	32	4846	152	222	76	286	19	55
MN	803	10944	20	3657	112	148	37	200	11	25
MO	781	9773	16	3173	78	124	33	150	26	53
MS	86	3228	4	335	31	52	17	59	4	3
MT	95	1724	6	641	14	21	6	40	6	5
NC	1306	16835	33	5613	198	272	76	346	12	29
ND	0	444	0	8	1	23	2	1	0	0
NE	4	1086	0	34	8	29	2	7	3	3
NH	157	3005	3	991	36	39	12	35	2	14
NJ	1841	22411	49	7760	251	296	96	419	26	107
NM	269	3372	8	1108	39	45	15	68	3	12
NV	803	8136	27	3006	79	115	22	226	16	13
NY	4124	49670	106	17214	619	722	233	1150	57	191
OH	1472	20873	40	6266	162	272	67	376	18	85
OK	421	5620	8	1710	71	76	25	137	3	14
OR	544	7201	8	2809	80	96	22	116	6	11
PA	1557	21812	39	6842	243	252	93	423	43	89
RI	191	2661	5	896	27	37	8	59	2	7
SC	449	7469	15	2369	64	110	27	119	4	13
SD	90	1200	1	454	8	18	3	38	1	2
TN	563	9944	22	1863	114	146	28	192	4	11
TX	3035	49254	111	16308	455	691	183	920	55	126
UT	349	3935	8	1763	43	45	22	80	5	14
VA	1438	17263	29	6504	205	258	92	387	16	63
VT	70	1324	1	358	7	15	5	14	1	2
WA	986	12879	22	4929	118	188	64	202	16	30
WI	536	8107	14	2542	56	104	23	136	14	42
WV	159	3104	6	978	28	32	14	55	2	8
WY	82	1371	1	520	9	16	6	19	0	4

Visualization 18: Analyzing % of Missing value from the Dataset



Visualization 19: Correlation Matrix



(I) CLEANSING AND PREPROCESSING

(A) Importing the Data and removing unnecessary columns

Columns with empty values for most of the rows as well as columns with the same values across all rows are dropped in order to have a cleaner dataset. Free form text columns are also dropped because we posited that these fields would have more noise and are better tackled at a later stage when we have better understanding of the problem.

Evaluating Data	Count
Total Row Count	**66553425**
Total Missing Values in Dataset	**17998490**
total_percentage = (missing_values/total_values)*100 **Hence, the total percentage of missing values in dataset is 27.043672057448582**	

Further, we have dropped unnecessary columns which are not required in order to make a cleaner dataset. We have taken the sum of the missing values in the dataset and divided it with the total values. After which, we have calculated the total percentage of missing values in the data as per the total values.

We then created more cleaner data by dropping all the columns which have missing values more than 50%. Also, we eliminated all the target values in the dataset which has no relation with the target variable.

- **top_mising_values = df_miss[df_miss["Percentage %"]>0]**
- **top_mising_values.reset_index(inplace=True)**
- **df.drop(top_mis[top_mising_values[‘Percentage%’]>50][‘index’],axis=1,inplace=True)**

Thus, in this stage we have succeeded in creating a cleaner dataset by eliminating the missing values and by dropping all the redundant and unnecessary columns.

- **df.drop(['title','earliest_cr_line','last_pymnt_d','last_credit_pull_d','zip_code','addr_state', 'next_pymnt_d',"policy_code", "id", "member_id","emp_title","url","purpose"], axis=1, inplace=True)**

(B) Implementing Mice to fill missing values

MICE is "multiple imputation by chained equations". Basically, missing data is predicted by observed data, using a sequential algorithm that is allowed to proceed to convergence.

(1) We have started filling in the missing data with plausible guesses at what the values might be.
(2) for each variable, we have predicted the missing values by modeling the observed values as a function of the other variables.

- ⇨ **imputer = mice.MICEData(df)**
- ⇨ **imputer.set_imputer('x1', formula='x2 + np.square(x2) + x3')**
- ⇨ **for j in range(20):**
  imputer.update_all()
- ⇨ **imputer.data.to_csv("data_after_mice.csv")**

At each step, we have updated the predictions of the missing values. Further, we have calculated stats model imputation formulas and updated the values available in the specified range. Thus, missing values have been successfully filled by implementing Mice.

(C) Normalize Data with Sklearn.ipynb

We then have approached to perform Normalization on the dataset to bring the data set on the same scale. We have split the dataset into 2 data frames out of which 1 needs to be normalized. Later, we performed normalization then summarized the transformed data. We have then calculated the correlation with interest rate after normalization for performing analysis and visualizations.

- ⇨ **Normalizing the data:**
  scaler_data = Normalizer().fit(X)
  normalizedData = scaler_data.transform(X)

- ⇨ **Summarizing the transformed data**
  np.set_printoptions(precision=3)

- ⇨ **Displaying transformed data**
  print(normalizedData[0:5,:])

(II) FEATURE SELECTION

Feature engineering is the process of using domain knowledge of the data to create features that make machine learning algorithms work. If feature engineering is done correctly, it increases the predictive power of machine learning algorithms by creating features from raw data that help facilitate the machine learning process. [8]

(A) Implementing Lasso CV for feature Selection

Lasso (“Least Absolute shrinkage and selection operator”) is a regression analysis method that performs both variable selection and regularization in order to enhance the prediction accuracy and interpretability of the statistical model it produces. The data is then categorized into three sets and treated differently: train, test and cv. We then split the data into dependent and independent variables to perform lasso cv for variable selection. Thus, we have successfully implemented feature selection by performing dimension reduction using LassoCV. [9]

- **Split the data in Train, Test, CV**

```
size = df.shape[0]
rs = 1
Train, Test = train_test_split(df, test_size= 0.2, random_state= rs)
CV, Test = train_test_split(Test, test_size=0.5, random_state = rs)
```

- **print(Train.shape, CV.shape, Test.shape)**

```
Output:(709820, 92) (88727, 92) (88728, 92)
```

- **Split the data in dependent and independent variable**

```
Train_y = np.array(Train["int_rate"])
CV_y = np.array(CV["int_rate"])
Test_y = np.array(Test["int_rate"])
Train_x = Train.drop(["int_rate"], axis = 1)
CV_x = CV.drop(["int_rate"], axis = 1)
Test_x = Test.drop(["int_rate"], axis = 1)
```

- **Perform LassoCV for variable selection**

```
modellassocv = LassoCV(alphas = [1, 0.1, 0.001, 0.0001, 10, 1000]).fit(Train_x,
Train_y)
lassopred = modellassocv.predict(CV_x)
print("RMSE of LassoCV: ", np.sqrt(mean_squared_error(lassopred, CV_y)))
coeff = modellassocv.coef_
x = list(Train_x)
x_position = [i for i, _ in enumerate(x)]
```

```
plt.figure(figsize = (10,40))
plt.barh(x_position, coeff, color='green')
plt.ylabel("Features")
plt.xlabel("Coefficients")
plt.title("Coefficients LassoCV")
plt.yticks(x_position, x)
plt.show()
```

⇨ `Output: RMSE of LassoCV:  0.6796729352761176`

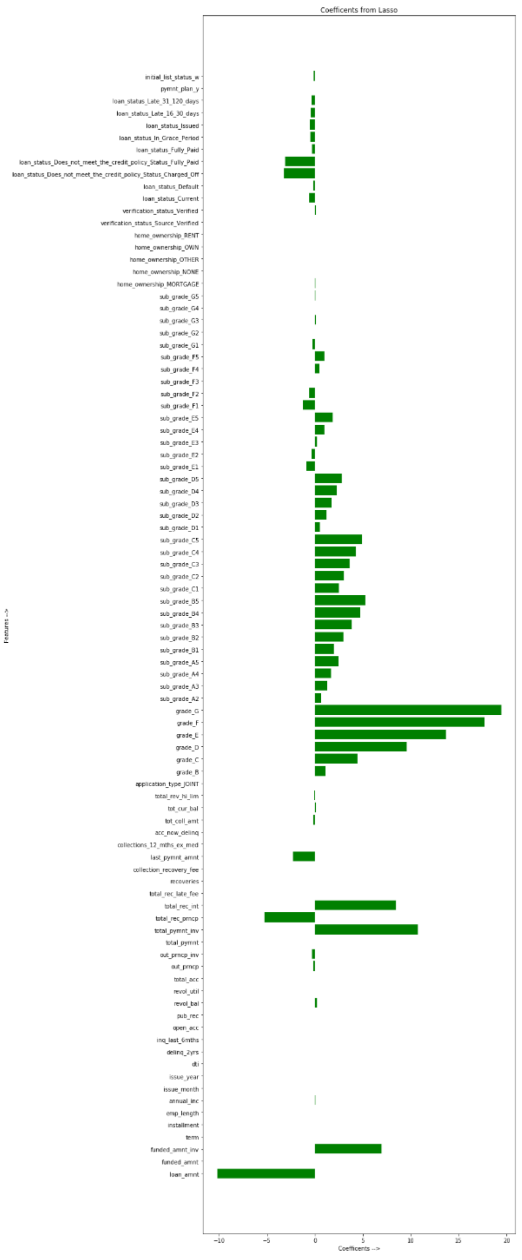


Figure 9: Coefficients from Lasso

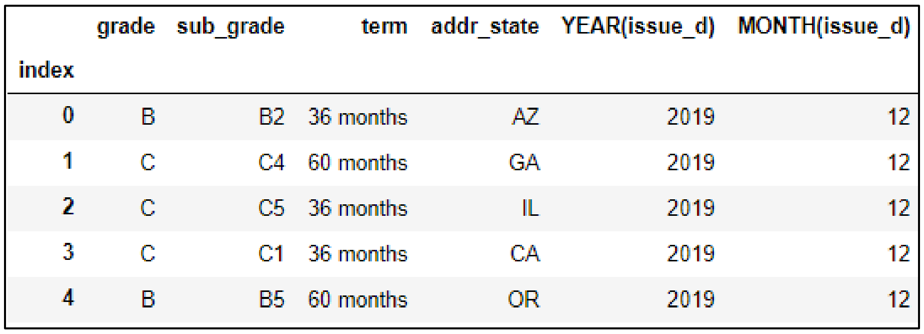
(B) Implementing Feature Tools

Feature tool helps the user to find some features that are necessary for mathematical calculation like avg, min, max, count etc. Feature tools have a lot of limitations like you need to pass proper entities before gaining features out of it. So, if a user needs to find some function related to mathematics then these feature tools are useful. Feature tool works well only with the numeric data. A user needs to decide manually first what features he needs and then pass the primitives accordingly. In short it is manually feature selection.

⇨ Make an entityset and add the entity on feature.csv
es = ft.EntitySet(id = 'data')
es.entity_from_dataframe(entity_id = 'data', dataframe = df, make_index = True, index = 'index')

⇨ List all the avaliable primitives
ft.list_primitives()

⇨ Run deep feature synthesis with transformation primitives
feature_matrix, feature_defs = ft.dfs(entityset = es, target_entity = 'data', trans_primitives = ['year', 'month'])



```
      grade  sub_grade        term  addr_state  YEAR(issue_d)  MONTH(issue_d)
index
    0     B         B2   36 months          AZ           2019              12
    1     C         C4   60 months          GA           2019              12
    2     C         C5   36 months          IL           2019              12
    3     C         C1   36 months          CA           2019              12
    4     B         B5   60 months          OR           2019              12
```

Figure 10: feature_matrix.head()

(III) MODELS & ALGORITHMS USED

(A) Auto SKLearn

Auto-SKLearn automatically searches for the right machine learning algorithm for a new machine learning dataset and optimizes its parameters. It is open source, implemented in python and built around the scikit-learn library. It contains a machine learning pipeline which takes care of missing values, categorical features, sparse and dense data, and rescaling the data. Next, the pipeline applies a preprocessing algorithm and an ML algorithm. [10] Auto-SKLearn includes 15 ML algorithms, 14 preprocessing methods, and all their respective hyperparameters, yielding a total of 110 hyperparameters; the resulting space is illustrated in Figure 1, where only the hyperparameters in grey boxes are active. Auto-SKLearn is an automated machine learning toolkit and a drop-in replacement for a scikit-learn estimator:

- **import autosklearn.classification**
- **class = autosklearn.classification.AutoSklearnClassifier()**
- **class.fit(X_train, y_train)**
- **prediction = class.predict(X_test)**

Auto-SKLearn frees a machine learning user from algorithm selection and hyperparameter tuning. It leverages recent advantages in Bayesian optimization, meta-learning and ensemble construction.

- **Creating training and test datasets**
  X_train, X_test, y_train, y_test =
  sklearn.model_selection.train_test_split(fullData[features],
  fullData[target_col],test_size=0.20, random_state=42)

- **Performing regression on the training dataset**
  autoskml = autosklearn.regression.AutoSklearnRegressor()
  autoskml.fit(X_train, y_train)

- **The predict function calls the predict function of the underlying scikit-learn model and returns an array with the predictions.**
  prediction = autoskml.predict(X_test)
  autoskml.fit(X_train, y_train)
  y_hat_test = autoskml.predict(X_test)
  x_hat_train=autoskml.predict(X_train)

(B) H2O AI (Hyper Parameter Optimization)

The H2O Python Module is a tool for rapidly turning over models, doing data munging, and building applications in a fast, scalable environment without any of the mental anguish about parallelism and distribution of work. In random forests, a random subset of candidate features is used to determine the most discriminative thresholds that are picked as the splitting rule. In extremely randomized trees (XRT), randomness goes one step further in the way that splits are computed. [11] As in random forests, a random subset of candidate features is used, but instead of looking for the most discriminative thresholds, thresholds are drawn at random for each candidate feature, and the best of these randomly generated thresholds is picked as the splitting rule. This usually allows to reduce the variance of the model a bit more, at the expense of a slightly greater increase in bias.

H2O supports extremely randomized trees (XRT) via histogram_type="Random". When this is specified, the algorithm will sample N-1 points from min…max and use the sorted list of those to find the best split. The cut points are random rather than uniform.

Hypertuning: A model hyperparameter is a configuration that is external to the model and whose value cannot be estimated from data. They are often used in processes to help estimate model parameters. They are often tuned for a given predictive modeling problem.

- **Libraries and Packages to be imported**
  ```
  import sys
  import h2o
  from h2o.estimators.random_forest import H2ORandomForestEstimator
  from h2o.estimators.glm import H2OGeneralizedLinearEstimator
  from h2o.grid.grid_search import H2OGridSearch
  from h2o.estimators.deeplearning import H2ODeepLearningEstimator
  ```

- **Import and start H2O on same machine as running python process**
  ```
  import h2o
  h2o.init()
  ```

- **Calculate MAPE (Mean absolute percentage error) value**
  The mean absolute percentage error (MAPE) is a statistical measure of how accurate a forecast system is. [12] It measures this accuracy as a percentage and can be calculated as the average absolute percent error for each time period minus actual values divided by actual values. Where At is the actual value and Ft is the forecast value, this is given by:

$$\mathrm{M} = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|$$

The mean absolute percentage error (MAPE) is the most common measure used to forecast error and works best if there are no extremes to the data (and no zeros).

⇨ **def MAPE(test, predict):**
mape_value = 0
for i, j in zip(test['int_rate'].as_data_frame().values,predict.as_data_frame().values):
    mape_value += np.abs((i-j)/i)
    mape_value = (mape_value/test.shape[0])*100
return mape_value

⇨ **Importing Data after mice file in two different data frames**
df_linear_frame = h2o.import_file('data_after_mice.csv')
df = h2o.import_file('C:/Users/data_after_mice.csv')

⇨ **Generate training and testing datasets**
train, valid, test = df.split_frame(
ratios=[0.6,0.2],
seed=1234,
destination_fram=['train.hex','valid.hex','test.hex'])

train_linear, valid_linear, test_linear = df_linear_frame.split_frame(
ratios=[0.6,0.2],
seed=1234,
destination_fram=['train.hex','valid.hex','test.hex'])

⇨ **Defining Response and Predictors**
The outcome variable is also called the response or dependent variable, and the risk factors and confounders are called the predictors, or explanatory or independent variables. [13] In regression analysis, the dependent variable is denoted "Y" and the independent variables are denoted by "X".

predictor_values = ['grade_C','grade_D','grade_E',
'grade_F','grade_G','total_rec_int',
'total_pymnt_inv','funded_amnt_inv','sub_grade_B5',
'sub_grade_C5','sub_grade_C4','sub_grade_C3','sub_grade_B4','sub_grade_D5']
Response_values = 'int_rate'

Predict the test and generate a MAPE for the model using algorithms like Random Forest, Neural networks and Linear Regression.

- **Builds a Distributed Random Forest (DRF) on a parsed dataset, for regression or classification**

  rf = H2ORandomForestEstimator(ntrees=20, max_depth = 60, stopping_rounds = 2, score_each_iteration = True, seed = 1000000)
  rf.train(x = predictors_values, y = response_values, training_frame = train, validation_frame = valid)

- **Output**

```
drf Model Build progress: |███████████████████████████████████████████████████████| 100%
Model Details
=============
H2ORandomForestEstimator :  Distributed Random Forest
Model Key:  DRF_model_python_1553282803100_25


ModelMetricsRegression: drf
** Reported on train data. **

MSE: 0.7280196567362235
RMSE: 0.8532406792554041
MAE: 0.4694499951492592
RMSLE: 0.07120868905774304
Mean Residual Deviance: 0.7280196567362235

ModelMetricsRegression: drf
** Reported on validation data. **

MSE: 0.6750968130417894
RMSE: 0.8216427526862203
MAE: 0.4529064636679627
RMSLE: 0.06868108253567803
Mean Residual Deviance: 0.6750968130417894
```

(C) Linear Regression

Linear regression is a statistical approach for modelling the relationship between a dependent variable with a given set of independent variables. Thus, linear regression technique finds out a linear relationship between x (input) and y (output). It is used to predict a quantitative response Y from the predictor variable X. [14] It is made with an assumption that there's a linear relationship between X and Y. The equation of the above line is:

$$\mathbf{Y = mx + b}$$

Where b is the intercept and m is the slope of the line. So basically, the linear regression algorithm gives us the most optimal value for the intercept and the slope (in two dimensions). The y and x variables remain the same, since they are the data features and cannot be changed. The values that we can control are the intercept(b) and slope(m). There can be multiple straight lines depending upon the values of intercept and slope. Basically, what the linear regression algorithm does is it fits multiple lines on the data points and returns the line that results in the least error.

This same concept can be extended to cases where there are more than two variables. This is called multiple linear regression. For instance, consider a scenario where you have to predict the price of the house based upon its area, number of bedrooms, the average income of the people in the area, the age of the house, and so on. In this case, the dependent variable (target variable) is dependent upon several independent variables. A regression model involving multiple variables can be represented as:

$$\mathbf{y = b0 + m1b1 + m2b2 + m3b3 + \ldots \ldots mnbn}$$

Parameters Used

⇨ Y = int_rate

⇨ X = grade_C,grade_D,grade_E, grade_F,grade_G,total_rec_int, total_pymnt_inv,funded_amnt_inv, sub_grade_B5, sub_grade_B4,sub_grade_C5, sub_grade_C4,sub_grade_C3,sub_grade_B4,sub_grade_D5

⇨ Alpha = 1

Significance of Parameters

⇨ **Y (Target Variable) = Y is a required variable.** We specify the column to use as the dependent variable. For a regression model, this column must be numeric (Real or Int). We have used int_rate as a target variable because we have based all our evaluation and modelling with respect to interest rate.

⇨ **X (Predictor Variable) = X is also a required variable.** We specify the names of the columns on which the target variable depends. These are called independent variables. We have used grade_C,grade_D,grade_E, grade_F,grade_G,total_rec_int, total_pymnt_inv,funded_amnt_inv,sub_grade_B5, sub_grade_B4,sub_grade_C5, sub_grade_C4,sub_grade_C3,sub_grade_B4,sub_grade_D5 as our predictor variable as after using feature selection tools like lasso cross validation and feature hasher we came to a conclusion that these are the best predictors.

⇨ **Alpha = 1**
The alpha parameter controls the distribution between the ℓ1ℓ1 (LASSO) and ℓ2ℓ2 (ridge regression) penalties. The penalty is defined as P(α,β)=(1−α)/2||β||22+α||β||1=∑j[(1−α)/2β2j+α|βj|] a value of 1.0 represents LASSO, and a value of 0.0 produces ridge regression.

(D) Neural Networks

Neural Networks are a class of models within the general machine learning literature. Neural networks are a specific set of algorithms that have revolutionized machine learning. They are inspired by biological neural networks and the current so-called deep neural networks have proven to work quite well. [15]

- **Significant Parameters used**
- Y = int_rate
- X = grade_C,grade_D,grade_E,grade_F,grade_G, total_rec_int, total_pymnt_inv, funded_amnt_inv, sub_grade_B5, sub_grade_B4,sub_grade_C5, sub_grade_C4,sub_grade_C3,sub_grade_B4,sub_grade_D5
- **Optimizers:** Gradient descent is an iterative machine learning optimization algorithm to reduce the cost function. Here we have used GCD
- **Epochs:** Specify the number of times to iterate (stream) the dataset. The value can be a fraction. In our case it is 15
- **Hidden:** Specify the hidden layer sizes (e.g., 100,100). The value must be positive.
- **Mini_batch_size:** Specify a value for the mini-batch size. (Smaller values lead to a better fit; larger values can speed up and generalize better.) which is 10
- **Activation:** Specify the activation function (Tanh, Tanh with dropout, Rectifier, Rectifier with dropout, Maxout, Maxout with dropout). Here, we used RELU (Rectified Linear Unit.)

- **Impute numerical missing values with mean**
  fullDataset[num_cols] = fullDataset[num_cols].fillna(fullDataset[num_cols].mean())

- **Impute categorical missing values with -9999**
  fullDataset[cat_cols] = fullDataset[cat_cols].fillna(value = -9999)

- **Splitting Training and Testing Data**
  features=list(set(list(fullDataset.columns))-set(target_col))
  X = fullDataset[features].values
  y = fullDataset[target_col].values
  X_train, X_valid, y_train,y_valid = train_test_split(X, y, test_size=0.20, random_state=42)

⇨ **Calculating MAPE for Training and Testing data, Plotting**

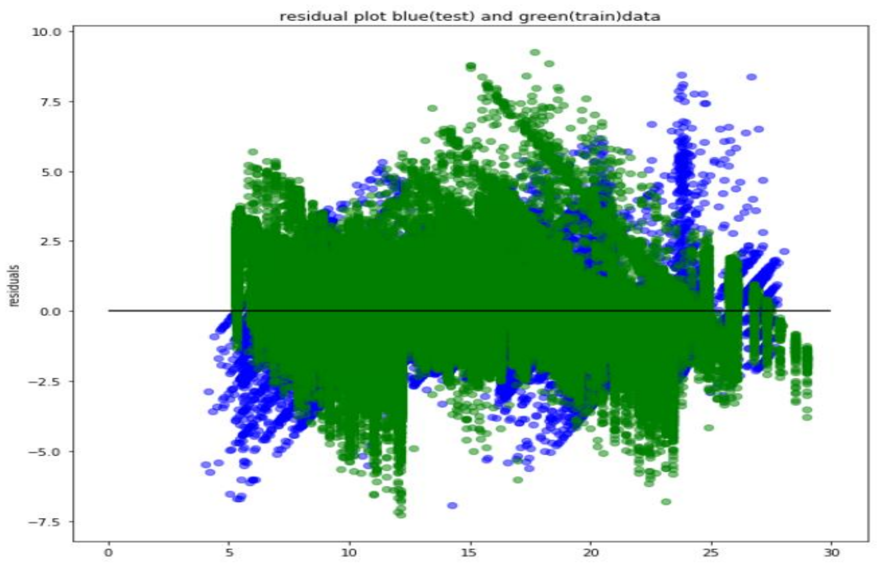


Figure 11: Residual Plot

⇨ **Hyperparameter Tuning with GRIDSEARCHCV**

```
cvscoresTest =[]
cvscoresTrain=[]
def baseline_model(optimizer='adam'):
    model = Sequential()
    model.add(Dense(50, input_dim=14, activation= "relu"))
    model.add(Dense(25, activation= "relu"))
    model.add(Dense(1, activation= "relu"))
    model.compile(loss= "mse" , optimizer=optimizer, metrics=["accuracy"])
    return model
estimatorF = KerasRegressor(build_fn=baseline_model, nb_epoch=100, batch_size=5,
verbose=0)
optimizers = ['rmsprop', 'adam','SGD']
batch_size = [10, 20, 50]
epochs = [5, 10, 15]
param_grid = dict(optimizer=optimizers,batch_size=batch_size, epochs=epochs)
grid = GridSearchCV(estimator=estimatorF,param_grid=param_grid)
grid_result = grid.fit(X_train.values, y_train.values)
```

(E) Random Forests

A Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees. The random forest algorithm combines multiple algorithms of the same type i.e. multiple decision trees, resulting in a forest of trees, hence the name "Random Forest". The random forest algorithm can be used for both regression and classification tasks. [16]

In case of a regression problem, for a new record, each tree in the forest predicts a value for Y (output). The final value can be calculated by taking the average of all the values predicted by all the trees in forest. Or, in case of a classification problem, each tree in the forest predicts the category to which the new record belongs. Finally, the new record is assigned to the category that wins the majority vote.

The random forest algorithm is not biased, since there are multiple trees and each tree is trained on a subset of data. Basically, the random forest algorithm relies on the power of "the crowd"; therefore, the overall biasedness of the algorithm is reduced. Even if a new data point is introduced in the dataset the overall algorithm is not affected much since new data may impact one tree, but it is very hard for it to impact all the trees. The algorithm works well when you have both categorical and numerical features. It works well when data has missing values, or it has not been scaled well.

Parameters Used

- ⇨ Y = int_rate
- ⇨ X = grade_C,grade_D,grade_E,grade_F,grade_G, total_rec_int,total_pymnt_inv, funded_amnt_inv, sub_grade_B5, sub_grade_B4, sub_grade_C5, sub_grade_C4,sub_grade_C3, sub_grade_B4, sub_grade_D5n_estimators = 10
- ⇨ random_state = 42

Significance of Parameters

- ⇨ n_estimators: (number of trees)
- ⇨ random_state: seed: Specify the random number generator (RNG) seed for algorithm components dependent on randomization. The seed is consistent for each H2O instance so that you can create models with the same starting conditions in alternative configurations.

- ⇨ **Select features that were given as Significant from Lasso-CV Feature Selection Algorithm**
  sel_features = ['grade_C','grade_D','grade_E', 'grade_F','grade_G', 'total_rec_int', 'total_pymnt_inv', 'funded_amnt_inv', 'sub_grade_B5', 'sub_grade_B4','sub_grade_C5', 'sub_grade_C4','sub_grade_C3','sub_grade_D5','int_rate']

⇨ **Converting Data frames to Individual Arrays of Features and Labels to Fit to a Model**

```
labels = np.array(features['int_rate'])
features= features.drop('int_rate', axis = 1)
feature_list = list(features.columns)
features = np.array(features)
```

⇨ **Using Skicit-learn to split data into training and testing sets**

```
train_features, test_features, train_labels, test_labels = train_test_split(features, labels,
test_size = 0.25, random_state = 42)
```

⇨ **Generate an Evaluation Function to Run the Model**

```
def eval(model,X_train,X_test,Y_train,Y_test):
    model.fit(X_train,Y_train)
    predictions_train = model.predict(X_train)
    predictions_test = model.predict(X_test)
    errors_train = abs(predictions_train - Y_train)
    errors_test = abs(predictions_test - Y_test)
    mape_train = 100 * np.mean(errors_train / Y_train)
    mape_test = 100 * np.mean(errors_test / Y_test)
    accuracy_train = 100 - mape_train
    accuracy_test = 100 - mape_test
return accuracy_train,accuracy_test
```

⇨ **Importing the Model and Providing the Necessary Parameters**

```
Model Performance
Average Error(Train Data): 0.1535 of int rate.
Average Error(Test Data): 0.3821 of int rate.
Accuracy(Train Data) = 98.68%.
Accuracy(Test Data) = 96.76%.
Mape(Train Data): 1.3202 of int rate
Mape(Test Data): 3.2373 of int rate
```

⇨ **Listing the Variable Importance**

```
importances = list(rf.feature_importances_)
feature_importances = [(feature, round(importance, 2)) for feature, importance in
zip(feature_list, importances)]
```

⇨ **Implementing Model with the Most Important Features**

```
important_indices = [feature_list.index('grade_D'), feature_list.index('grade_E'),
feature_list.index('grade_F'), feature_list.index('grade_C'), feature_list.index('grade_G'),
feature_list.index('total_rec_int'), feature_list.index('total_pymnt_inv'),
feature_list.index('funded_amnt_inv'),  feature_list.index('sub_grade_B4'),
feature_list.index('sub_grade_B5'),feature_list.index('sub_grade_C5')]
train_important = train_features[:, important_indices]
test_important = test_features[:, important_indices]
```

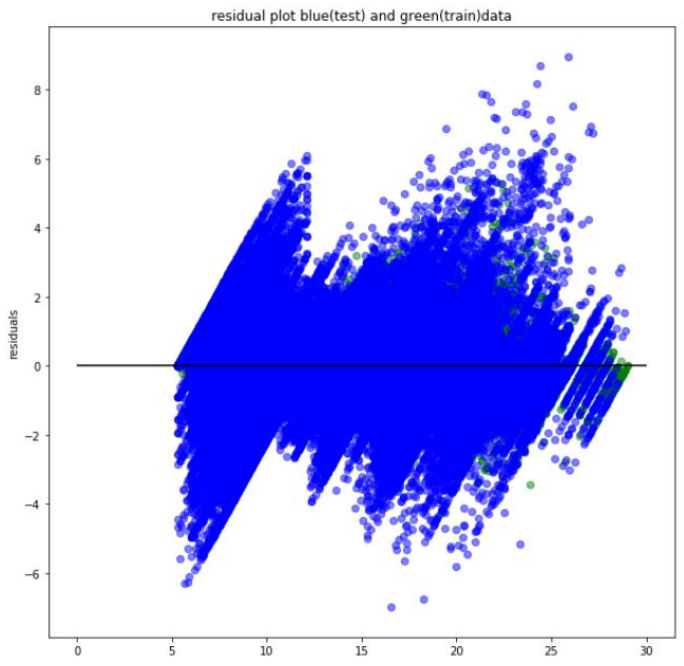


Figure 12: Residual Plot for Evaluating Best Features

⇨ **Examine the Default Random Forest to Determine Parameters**

⇨ **Random Search [Grid] With Cross Validation**

```
{'bootstrap': [True, False],
 'max_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, None],
 'max_features': ['auto', 'sqrt'],
 'min_samples_leaf': [1, 2, 4],
 'min_samples_split': [2, 5, 10],
 'n_estimators': [10, 20, 30, 50, 100]}
```

⇨ **Evaluate the Random Forest Model with the Best Parameters**

```
Model Performance
Average Error(Train Data): 0.2239 of int rate.
Average Error(Test Data): 0.4055 of int rate.
Accuracy(Train Data) = 98.10%.
Accuracy(Test Data) = 96.60%.
Mape(Train Data): 1.9012 of int rate
Mape(Test Data): 3.3966 of int rate
```

(F) TPOT

TPOT is meant to be an assistant that gives us ideas on how to solve a particular machine learning problem by exploring pipeline configurations that you might have never considered, then leaves the fine-tuning to more cons trained parameter tuning techniques such as grid search.

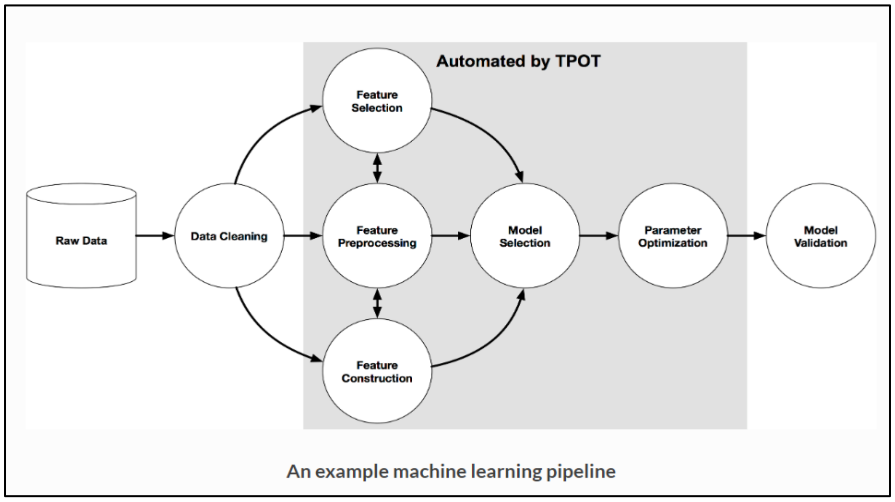


Figure 12: TPOT Mechanism
(Source: http://epistasislab.github.io/tpot/)

TPOT will take a while to run on larger datasets, but it's important to realize why. With the default TPOT settings (100 generations with 100 population size), TPOT will evaluate 10,000 pipeline configurations before finishing. To put this number into context, think about a grid search of 10,000 hyperparameter combinations for a machine learning algorithm and how long that grid search will take. That is 10,000 model configurations to evaluate with 10-fold cross-validation, which means that roughly 100,000 models are fit and evaluated on the training data in one grid search. That's a time-consuming procedure, even for simpler models like decision trees. [17]

TPOT has what its developers call a genetic search algorithm to find the best parameters and model ensembles. It could also be thought of as a natural selection or evolutionary algorithm. TPOT tries a pipeline, evaluates its performance, and randomly changes parts of the pipeline in search of better performing algorithms. Typical TPOT runs will take hours to days to finish (unless it's a small dataset), but you can always interrupt the run partway through and see the best results so far. TPOT also provides a warm_start parameter that lets you restart a TPOT run from where it left off.

- ⇨ **Library Used**
  tpot import TPOTRegressor

⇨ **Splitting the data into predictors and features**

```
predictors = ['grade_C','grade_D','grade_E',
'grade_F','grade_G','total_rec_int',
'total_pymnt_inv','funded_amnt_inv','sub_grade_B5',
'sub_grade_C5','sub_grade_C4','sub_grade_C3','sub_grade_B4','sub_grade_D5']
response = ['int_rate']
```

⇨ **Instantiating, fitting, & scoring the TPOT classifier is like any other sklearn classifier**

```
tpot = TPOTRegressor(generations=5, population_size=50, verbosity=2, config_dict =
tpot_config)
tpot.fit(X_train, y_train)
print(tpot.score(X_test, y_test))
```

⇨ **TPOT writes the optimized pipeline to an external python file**

```
tpot.export(tpot_lendingClub_pipeline_file.py)
```

⇨ **Output:**

```
Generation 1 - Current best internal CV score: -1.5233035049539687
Generation 2 - Current best internal CV score: -1.5233035049539687
Generation 3 - Current best internal CV score: -1.5233035049539687
Generation 4 - Current best internal CV score: -1.5233027505419723
Generation 5 - Current best internal CV score: -1.5233027505419723

Best pipeline: LinearRegression( CombineDFs (CombineDFs (input_matrix,
input_matrix), LinearRegression (input_matrix)))
-1.5258951979649398
```

⇨ **The code from exported tpot_lendingClub_pipeline_file.py file which is as below:**

```
mape = 0
for i,j in zip(y_test, results):
    mape += np.abs((i-j)/i)
 mape = (mape*100)/y_test.shape[0]
mape
```

⇨ **Output:** 8.551144342590337

CALCULATIONS

(A) Linear Regression

Implementing Lasso (L1), Ridge (L2) and Elastic Net Regularization

- **Calculating R2 Value**

```
np.round(lm_lasso.score(test_x,test_y)*100,2)
np.round(lm_ridge.score(test_x,test_y)*100,2)
np.round(lm_elastic.score(test_x,test_y)*100,2)
```

- **Creating an Evaluation Function to Run Models**

```
def evaluate(model,X_train,X_test,Y_train,Y_test):
    model.fit(X_train,Y_train)
    predictions_train = model.predict(X_train)
    predictions_test = model.predict(X_test)
    errors_train = abs(predictions_train - Y_train)
    errors_test = abs(predictions_test - Y_test)
    mape_train = 100 * np.mean(errors_train / Y_train)
    mape_test = 100 * np.mean(errors_test / Y_test)
    accuracy_train = 100 - mape_train
    accuracy_test = 100 - mape_test
    print('Model Performance')
    print('Average Error(Train Data): {:0.4f} of int rate.'.format(np.mean(errors_train)))
    print('Average Error(Test Data): {:0.4f} of int rate.'.format(np.mean(errors_test)))
    print('Accuracy(Train Data) = {:0.2f}%.'.format(accuracy_train))
    print('Accuracy(Test Data) = {:0.2f}%.'.format(accuracy_test))
    print('Mape(Train Data): {:0.4f} of int rate'.format(mape_train))
    print('Mape(Test Data): {:0.4f} of int rate'.format(mape_test))
 return accuracy_train,accuracy_test
```

Implementing Linear Regression

- **Calculating MAPE and display accuracy**

```
mape = 100 * (errors / y_test)
accuracy = 100 - np.mean(mape)
print('Accuracy:', round(accuracy, 2), '%.')
```

- **Calculating MAPE after 5 Fold CV**

```
kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
cvscores =[]
```

```
kf = KFold(n_splits=5, random_state=42, shuffle=True)
i=1
model = LinearRegression()
for train_index, test_index in kf.split(X):
    X_trainD, X_testD = X[train_index], X[test_index]
    y_trainD, y_testD = y[train_index], y[test_index]
    i=i+(int(len(X_train)/5))
    model = LinearRegression()
    model.fit(X_trainD, y_trainD)
    pred= model.predict(X_testD)
    mape=np.mean(np.abs((y_testD-pred) / y_testD)) * 100
    cvscores.append(mape)
mape=np.mean(cvscores)
```

(B) Random Forest

⇨ **Generate an Evaluation Function to Run the Model**

```
def evaluate(model,X_train,X_test,Y_train,Y_test):
    model.fit(X_train,Y_train)
    predictions_train = model.predict(X_train)
    predictions_test = model.predict(X_test)
    errors_train = abs(predictions_train - Y_train)
    errors_test = abs(predictions_test - Y_test)
    mape_train = 100 * np.mean(errors_train / Y_train)
    mape_test = 100 * np.mean(errors_test / Y_test)
    accuracy_train = 100 - mape_train
    accuracy_test = 100 - mape_test
    print('Model Performance')
    print('Average Error(Train Data): {:0.4f} of int rate.'.format(np.mean(errors_train)))
    print('Average Error(Test Data): {:0.4f} of int rate.'.format(np.mean(errors_test)))
    print('Accuracy(Train Data) = {:0.2f}%.'.format(accuracy_train))
    print('Accuracy(Test Data) = {:0.2f}%.'.format(accuracy_test))
    print('Mape(Train Data): {:0.4f} of int rate'.format(mape_train))
    print('Mape(Test Data): {:0.4f} of int rate'.format(mape_test))
    plt.figure(figsize = (10,10))
    plt.scatter(predictions_train,(predictions_train - Y_train),c='g',s=40,alpha=0.5)
    plt.scatter(predictions_test,(predictions_test - Y_test),c='b',s=40,alpha=0.5)
    plt.hlines(y=0,xmin=0,xmax=30)
    plt.title('residual plot blue(test) and green(train)data')
    plt.ylabel('residuals')
return accuracy_train,accuracy_test
```

(C) TPOT

⇨ **Calculating MAPE**

```
mape = 0
for i,j in zip(y_test, results):
   mape += np.abs((i-j)/i)
mape = (mape*100)/y_test.shape[0]
mape
```

(D) H2O

⇨ **Creating an evaluation function to calculate MAPE**

```
def MAPE(test, predict):
mape = 0
for i, j in zip(test['int_rate'].as_data_frame().values, predict.as_data_frame().values):
         mape += np.abs((i-j)/i)
         mape = (mape/test.shape[0])*100
return mape
```

(E) Neural Networks

⇨ **Implementing Five Cross Validation**

```
for train_index, test_index in kf.split(X):
   X_trainD, X_testD = X[train_index], X[test_index]
   y_trainD, y_testD = y[train_index], y[test_index]
   model = Sequential()
   model.add(Dense(50, input_dim=14, activation= "relu"))
   model.add(Dense(25, activation= "relu"))
   model.add(Dense(1, activation= "relu"))
   model.compile(loss= "mean_squared_error" , optimizer="adam",
metrics=["mean_squared_error"])
   model.fit(X_trainD, y_trainD,batch_size=50, epochs=10)
   pred= model.predict(X_valid)
   mape=np.mean(np.abs((y_valid-pred) / y_valid)) * 100
   cvscores.append(mape)
```

⇨ **Calculating MAPE for Training and Testing data, Plotting**

```
predtest= model.predict(X_valid)
mapeTest=np.mean(np.abs((y_valid-predtest) / y_valid)) * 100
predtrain= model.predict(X_train)
mapeTrain=np.mean(np.abs((y_train-predtrain) / y_train)) * 100
```

VISUALIZATIONS

In order to understand the dataset and extract valuable insights from it we have analyzed the dataset using **Tableau and PowerBI** which we are currently learning **in INFO7370 Designing Advanced Data Architectures for Business Intelligence course under Professor Rick Sherman** where in the visualization tools helps in combining the ease of dragging-and-dropping visual analytics with the flexibility to dynamically integrate data from analytical systems. [18] Below are visualizations which helped us understanding the client data:

PowerBI: https://app.powerbi.com/groups/me/dashboards/3c780c29-47a3-4f41-a4c8-28c0c2e550c0?route=groups%2Fme%2Freports%2Fbbd57c83-9a3f-4736-9bb4-4eaede8a48f2%2FReportSectionada63672042896b8705e&noSignUpCheck=1

Tableau Visualization 1: Average Interest Rate by States in US

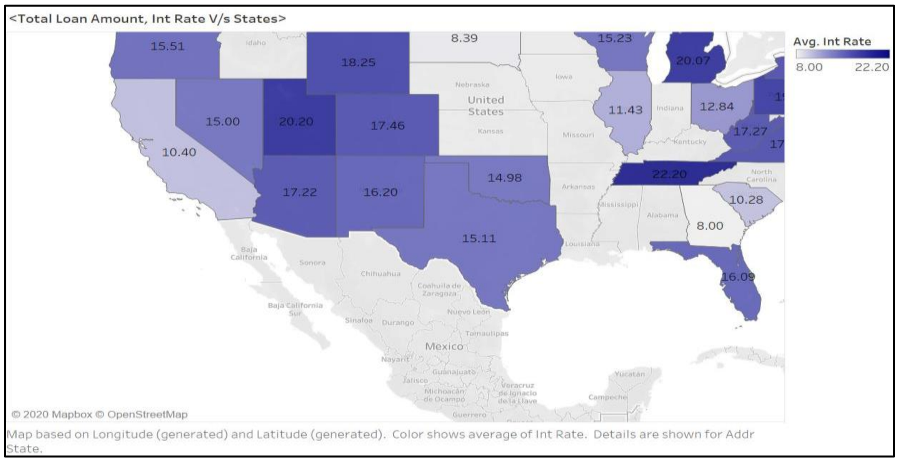
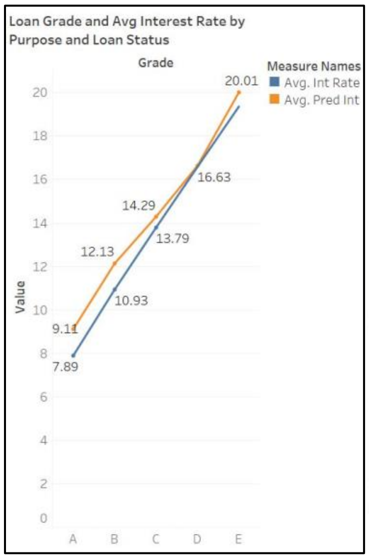
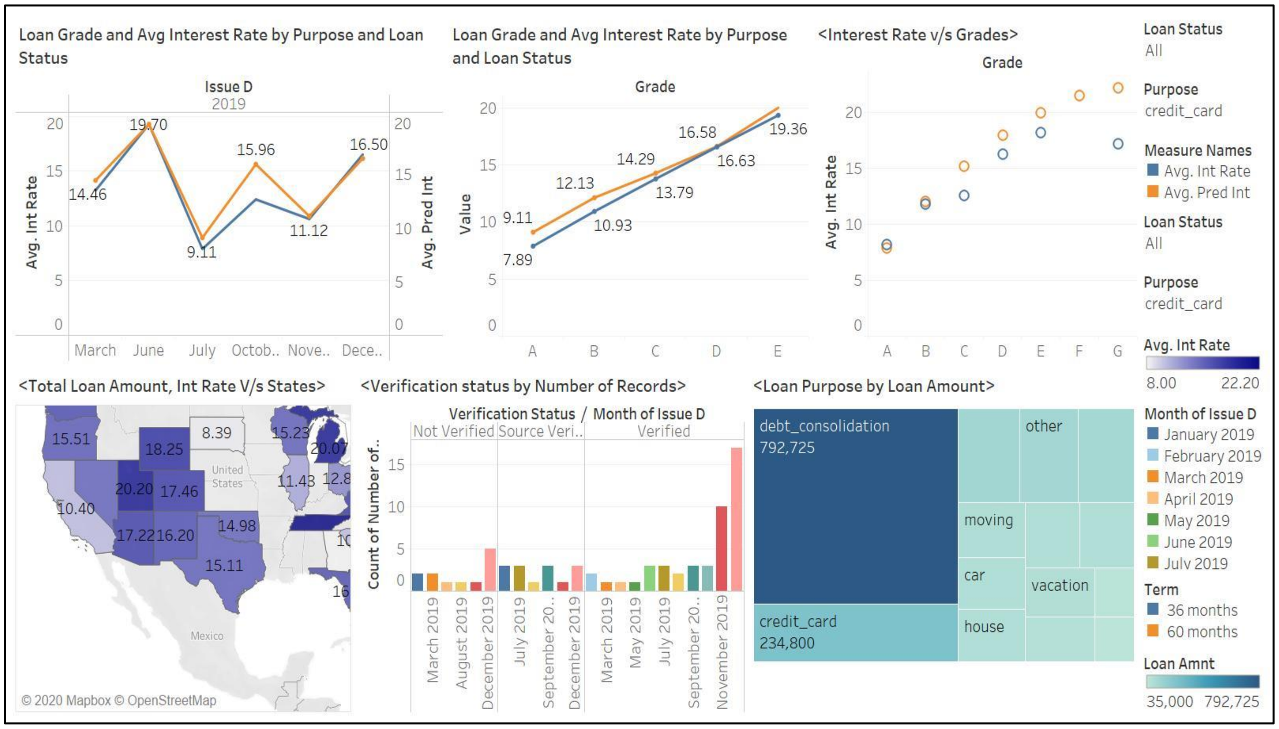

<Total Loan Amount, Int Rate V/s States>
Avg. Int Rate
8.00
22.20
15.51
18.25
8.39
15.23
20.07
15.00
20.20
17.46
11.43
12.84
17.27
10.40
17.22
16.20
14.98
22.20
10.28
8.00
15.11
16.09
United States
Mexico
© 2020 Mapbox © OpenStreetMap
Map based on Longitude (generated) and Latitude (generated). Color shows average of Int Rate. Details are shown for Addr State.

Tableau Visualization 2: Loan Grade and Avg Interest Rate by Purpose and Loan Status

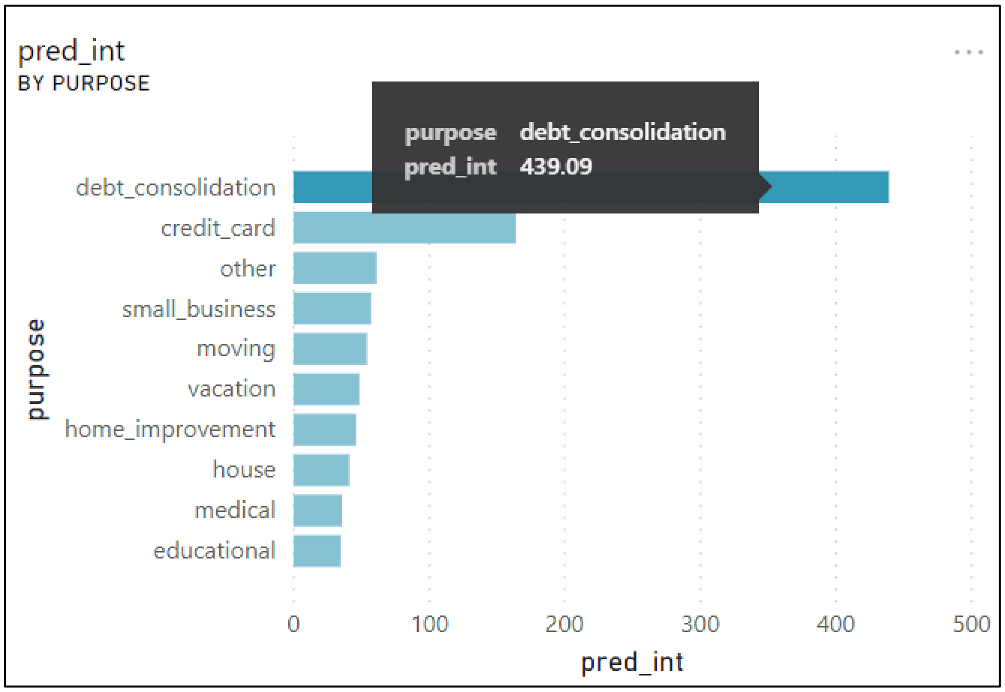


The trends of Avg. Int Rate and Avg. Pred Int for Grade. Color shows details about Avg. Int Rate and Avg. Pred Int. The data is filtered on Purpose and Loan Status. The Purpose filter keeps credit_card. The Loan Status filter keeps Charged Off, Current, Does not meet the credit policy. Status:Charged Off, Does not meet the credit policy. Status:Fully Paid and Fully Paid.

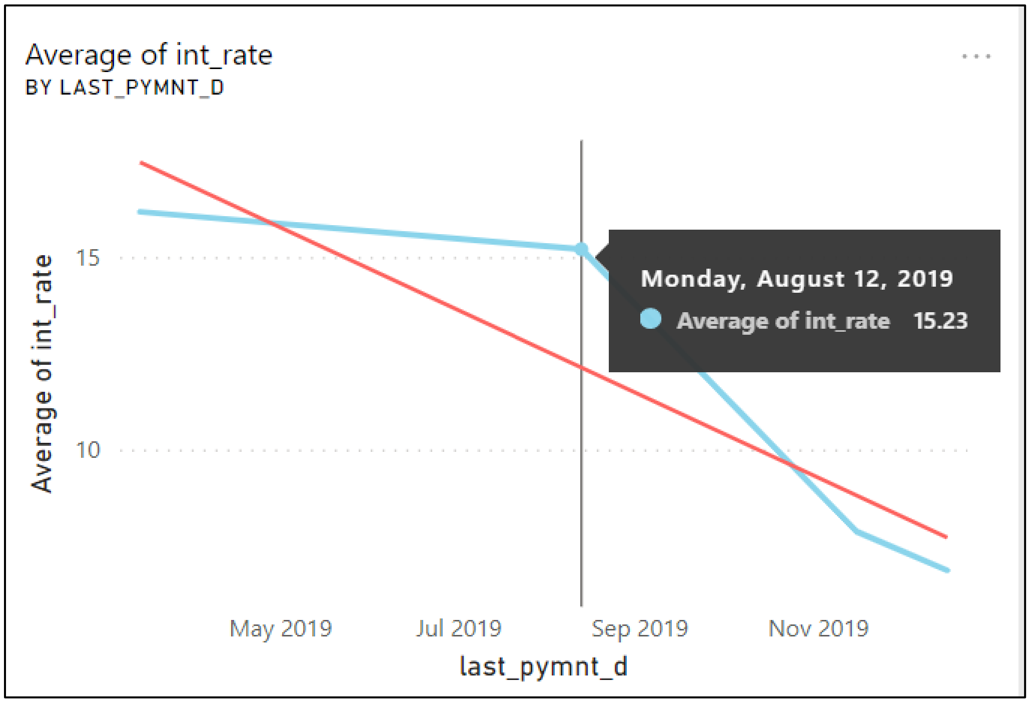
Tableau Visualization 3: Consolidated Interactive Dashboard



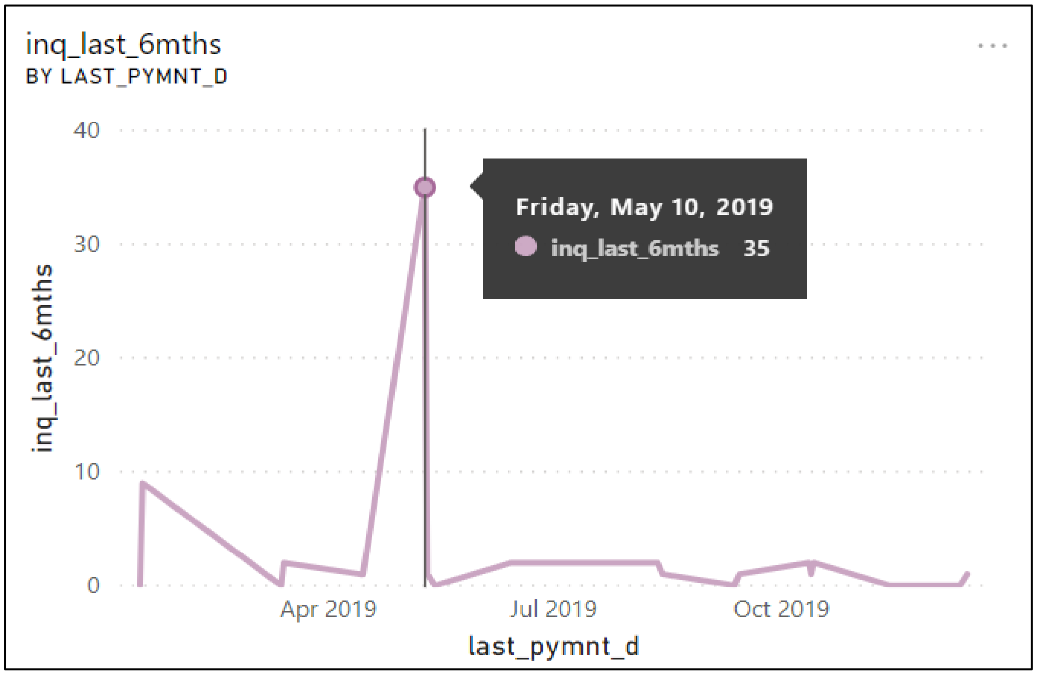
PowerBI Visualization 1: Predicted Interest Rate by Loan Status



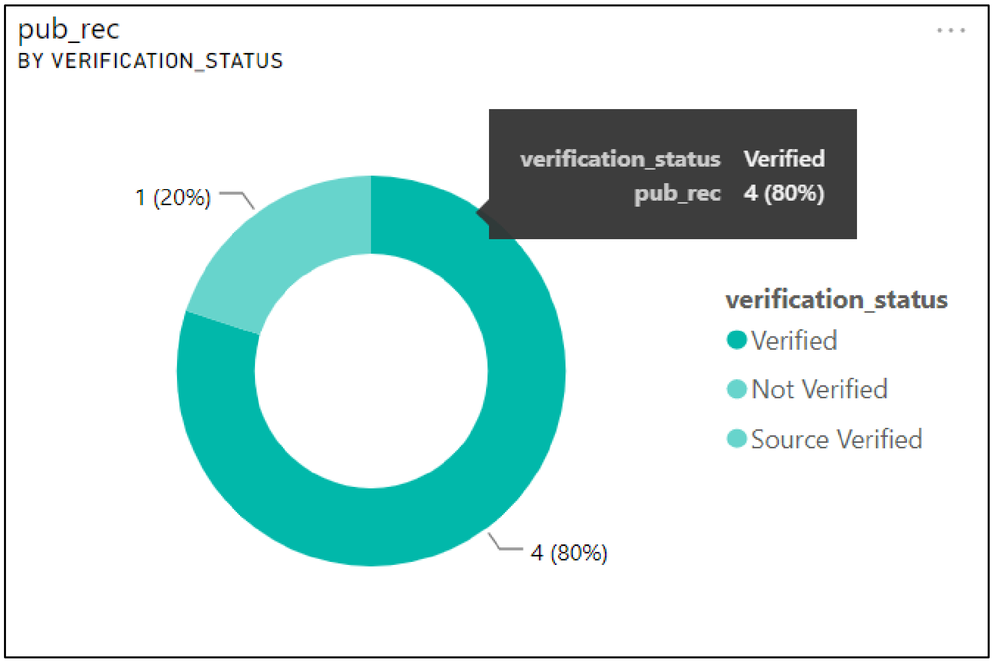
PowerBI Visualization 2: Average Interest Rate by Last Payment Date



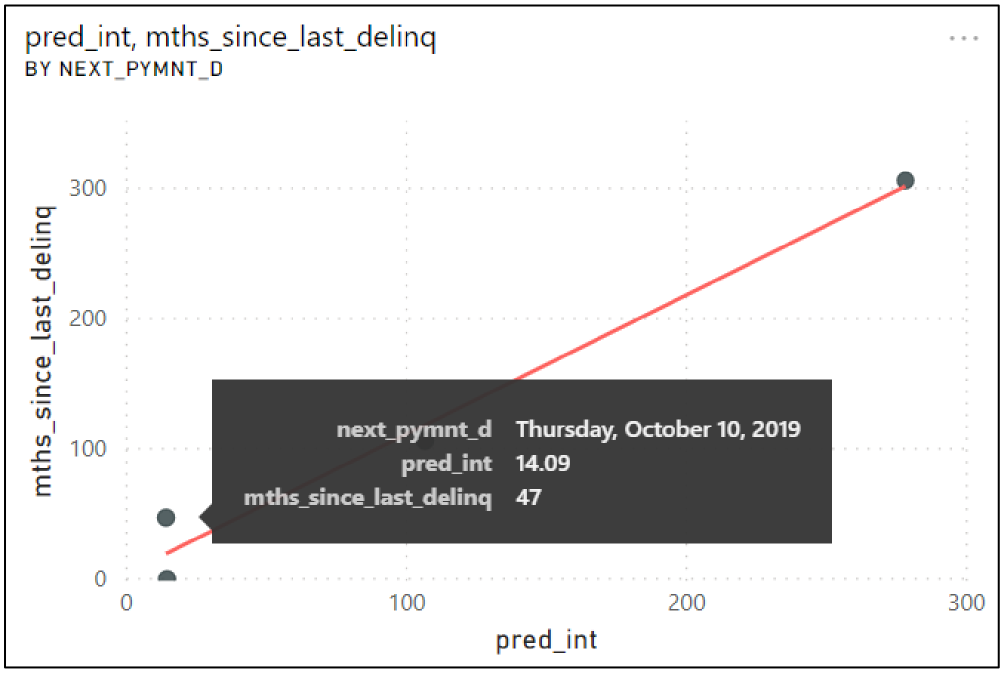
PowerBI Visualization 3: Average Interest Rate by Last Payment Date



PowerBI Visualization 3: ‘pub_rec’ by Loan Verification Status



PowerBI Visualization 3: Predict Interest, ‘mths_since_last_delinq’ Next Payment Date



FINAL RESULTS & CONCLUSION

The selected test cases based on 6 criteria:

- ⇨ **Annual Income:** 2 test cases for high annual income, low income, mid-range annual income each
- ⇨ **Term:** Selected 5 case each for both given term i.e. 36, 60 months
- ⇨ **Loan amount:** 5 cases for high loan amount, low amount, mid-range loan amount each
- ⇨ **Purpose:** Took a different case for each of the 10 purposes
- ⇨ **State:** Selected a case for each of the 50 states
- ⇨ **Enquiries in Last 6 months:** Selected 2 test cases for lower number enquiries, 2 for higher enquiries, and 2 for mid number of enquiries

MAPE for each Models:

- ⇨ Linear Regression: 8.54
- ⇨ Neural Network: 5.75
- ⇨ Random Forest: 3.4
- ⇨ T-POT: 11.72825015878589
- ⇨ AutoSKLearn: 12.38768 (No models are better than Random Model)
- ⇨ H2O: 6.55582879

Amongst all the three models we can see that **Random Forest Regressor** gives us the best predictions. Given, any test case including the ones carried out by our team, the predicted values can be seen satisfactory as compared to the actual data of interest rates provided by the lending club. So, if we test, our model, against any scenario, be risk averse, risk taking or any other scenario, our model will give a satisfactory predicted estimate of the interest rate. Also, it tells us about the **most optimized ML Model that is Random Forest with a MAPE of 3.80**

GITHUB Link:
https://github.com/akashmdubey/Lending-Club-Interest-Rate-Prediction-Using-Supervised-ML

Codelabs Link:
https://codelabs-preview.appspot.com/?file_id=1rYZCIi7hOFTiRhG1IEAnrO7JBcLSmFqHaxtiy4rvMDk#0

FUTURE WORK

We obtained a regression prediction threshold based on the training set and simulated the strategy on the test set. Both sets comprise loans initiated within the same periods (2012-2015). We can check to see if the strategy generalizes to future loans by testing it on 2016-2018 loans that have finalized. Practically speaking, this would be a much more useful metric for investors. We worked with a 70% training and 30% test split for simplicity in this project. [6] The absence of a development set didn't afford us much opportunity to tune the hyperparameters of our models, such as the number of decision trees to use in random forest models, and the number of hidden layers and neurons of each layer in neural network models. Having a small development set would enable us to tune some hyper-parameters quickly to help improve model performance metrics.

We can also make better use of existing features in the LendingClub dataset. One example is loan description which the borrower enters at the time of loan application. Instead of dropping such freeform features, we can try applying some statistical natural language processing techniques. Finally, we notice that LendingClub also publishes declined loan datasets. We can add these declined loans as negative examples to our dataset, which helps further alleviate the class imbalance problem.

KEY TAKEAWAYS FOR USERS

Reference 1: https://www.lendingclub.com/info/statistics.action

- From the total loan issuance graph, the amount of loans disbursed year over year increased as we go from 2011 to 2018, and within the years, the growth is exponential in every quarter.
- The maximum reported loan purpose radial pie shows that of Refinancing contributed a total of 45.6% followed by Credit card payoffs which held a total of 22.58% and then the remaining Other 31.82%
- Loan issuance by state graph tells us that over 19 states in the US issued total loan amounts of over $50 million and more, whereas 14 states held a total of $25-50 million and the remaining 18 states issued total loan amount between $0-25 million.

Reference 2: https://www.lendingclub.com/info/statistics-performance.action

- Investor account returns by average age of portfolio shows that of all investor accounts on the LendingClub platform that have invested in at least 100 Notes and that have not purchased or sold Notes on the Folio Investing Note Trading Platform.
- We have adjusted NAR on the Y axis against average age of portfolios. It is the weighted average age of notes.
- If I invest in multiple loans then my adjusted NAR would be high in the beginning and over the period of time as we traverse, the adjusted NAR decreases due to multiple reasons, one of which is the loan can end up being charged off.
- If a loan is concentrated and that loan ends up being charged off, then the NAR would be 0, therefore it is advisable to invest into multiple loans with at least the minimum amount of Notes rather than just investing into one loan with all your funds.

Reference 3: https://www.lendingclub.com/info/demand-and-credit-profile.action

- Average interest rate multi-line chart shows the comparison of Loan Grades and their Terms over the years v/s the interest rates.
- As we move from grades A to G, there is an increase in the interest rate of those loans. It also means that a loan with a grade of A will yield a lower interest rate as compared to that of a G grade loan.
- Over the years, the loan grades A to D have a constant interest rate band whereas there is a significant increase in the interest rates with loan grades E to G with a substantial increase of 10%
- Grade mix over time shows that lower the grade of the loan, lower issuance for that loan have been made over the years. Loan grades B, C followed by A were the maximum issued loan grades.

DEFINITIONS AND ABBREVIATIONS

ABBREVIATIONS	FULL FORM	DEFINITION
MICE	Multiple Imputation by Chained Equations	Multivariate imputation by chained equations (MICE), sometimes called “fully conditional specification” or “sequential regression multiple imputation” has emerged in the statistical literature as one principled method of addressing missing data. [19]
MAPE	Mean Absolute Percentage Error	The mean absolute percentage error (MAPE), also known as mean absolute percentage deviation (MAPD), is a measure of prediction accuracy of a forecasting method in statistics, for example in trend estimation, also used as a loss function for regression problems in machine learning. [20]
LASSO	Least Absolute Shrinkage and Selection Operator	In machine learning, LASSO is a regression analysis method that performs both variable selection and regularization in order to enhance the prediction accuracy and interpretability of the statistical model it produces. [21]
DRF	Distributed Random Forest	DRF is a powerful classification and regression tool. When given a set of data, DRF generates a forest of classification or regression trees, rather than a single classification or regression tree. [22]
RELU	Rectified Linear Unit	The Rectified Linear Unit is the most used activation function in deep learning models. The function returns 0 if it receives any negative input, but for any positive value xx it returns that value back. So it can be written as f(x)=max(0,x)f(x)=max(0,x) [23]
TPOT	Tree-Based Pipeline Optimization Tool	TPOT was one of the very first AutoML methods and open-source software packages developed for the data science community. [24]

REFERENCES

[1] https://www.lendingclub.com/public/how-peer-lending-works.action
[2] http://cs229.stanford.edu/proj2015/199_report.pdf
[3] https://www.debt.org/credit/loans/personal/lending-club-review/
[4] https://en.wikipedia.org/wiki/LendingClub
[5] https://rstudio-pubs-static.s3.amazonaws.com/203258_d20c1a34bc094151a0a1e4f4180c5f6f.html
[6] http://cs229.stanford.edu/proj2018/report/69.pdf
[7] https://docplayer.net/39484619-3-algorithms-and-results-where-s.html
[8] https://elitedatascience.com/feature-engineering
[9] https://www.researchgate.net/publication/330380054_High-Dimensional_LASSO-Based_Computational_Regression_Models_Regularization_Shrinkage_and_Selection
[10] https://www.kdnuggets.com/2016/08/winning-automl-challenge-auto-sklearn.html
[11] http://h2o-release.s3.amazonaws.com/h2o/rel-slater/3/docs-website/h2o-py/docs/h2o.html
[12] https://www.statisticshowto.com/mean-absolute-percentage-error-mape/
[13] http://sphweb.bumc.bu.edu/otlt/MPH-Modules/BS/BS704_Correlation-Regression/BS704_Correlation-Regression_print.html
[14] https://www.statisticssolutions.com/what-is-linear-regression/
[15] https://www.kdnuggets.com/2018/02/8-neural-network-architectures-machine-learning-researchers-need-learn.html
[16] https://www.geeksforgeeks.org/random-forest-regression-in-python/
[17] https://towardsdatascience.com/tpot-automated-machine-learning-in-python-4c063b3e5de9
[18] https://www.tableau.com/about/blog/2019/7/enrich-data-tableau-machine-learning-using-algorithmia
[19] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3074241/
[20] https://en.wikipedia.org/wiki/Mean_absolute_percentage_error
[21] https://en.wikipedia.org/wiki/Lasso_(statistics)
[22] http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/drf.html
[23] https://www.kaggle.com/dansbecker/rectified-linear-units-relu-in-deep-learning
[24] http://automl.info/