Capstone Project – In-Depth Analysis

Table of Contents

Mortgage Loans Analytics

Banks can now use mortgage loan analytics using Data Science techniques. The system can provide detail information of the mortgage loans and the mortgage loan markets. It is a powerful tool for mortgage brokers to seek counterparties and generate trading interests and is useful for the CFOs to conduct what-ifs scenarios on the balance sheets.

Fill in the follow fields in Loan file template:

- Loan ID: to identify the special loan
- Loan Type: to indicate the loan if fixed rate, or balloon loan , or ARM, or AMP (alternative mortgage product).
- Balance:
- Loan program type: to indicate conforming loan, FHA/VA loan, Jumbo loan or sub-prime loan
- Current coupon rate:
- Amortization type: the original amortization term
- Maturity: the maturity loan (the remaining term of the loan)
- FICO Score: the updated fico score
- LTV: the current loan to value ratio
- Loan Size: the loan amount of the loan
- Loan origination location (City & Zip)
- Unit Types (Types of property)

1. Predicting mortgage demand using machine learning techniques

It is difficult for the financial institutions to determine the amount of personnel needed to handle the mortgage applications coming in. There are multiple factors influencing the amount of mortgage applications, such as the mortgage interest rates, which cause the amount of mortgage applications to differ day by day. In this research we aim to provide more insight in the amount of personnel needed by developing a machine learning model that predicts the amount of mortgage applications coming in per day for the next week, using the CRISP-DM framework. After conducting a literature study and interviews, multiple features are generated using historical data from a Dutch financial institution and external data. A number of machine learning models are developed and validated using cross-validation. The predictions of our best model differ on average mortgage applications per day compared to the actual amount of mortgage applications. A dynamic dashboard solution is proposed to visualize the predictions, in which mortgage interest rate changes can be manually entered in the dashboard, and recommendations have been given for the deployment of the model at the financial institutions.
Methodology

Historical data and publicly available data were used as input for our predictive model, and five machine learning techniques (Decision Tree, Random Forest, Support Vector Machines, Support Vector Regression and KNN) were applied to create the predictions. The models are validated using repeated cross-validation, and evaluated using several evaluation criteria. We have also used ARIMA model, Linear Regression and Logistic Regression for predictive modeling.

Results:

The Random Forest model gave the best result on each of the four evaluation criteria used to evaluate the models. The Random Forest model is mortgage applications per day perform best, then Decision Tree model. The SVR model scored is the worse, SVM on a second place. The percent error of the Random Forest model is around of the actual amount of mortgage applications per day.

Linear Regression,

Logistic Regression

Random Forests (RF)

Support Vector Regression (SVR)

Support Vector Machine (SVM)

k-Nearest Neighbors

Decision Tree Classifier

Using Scikit-learn, optimization of decision tree classifier performed by only pre-pruning. Maximum depth of the tree can be used as a control variable for pre-pruning. In the following the example, we can plot a decision tree on the same data with max_depth=4.

Business Understanding

The background information is collected on the domain. A theoretical framework is developed by conducting a literature review which contains an overview of the related research in this research area, and an overview of the concepts in predictive analytics and the different models that are feasible for our Capstone project. For the domain analysis, background information on the mortgage application process and the mortgage domain is collected in order to get a better understanding of the different topics.

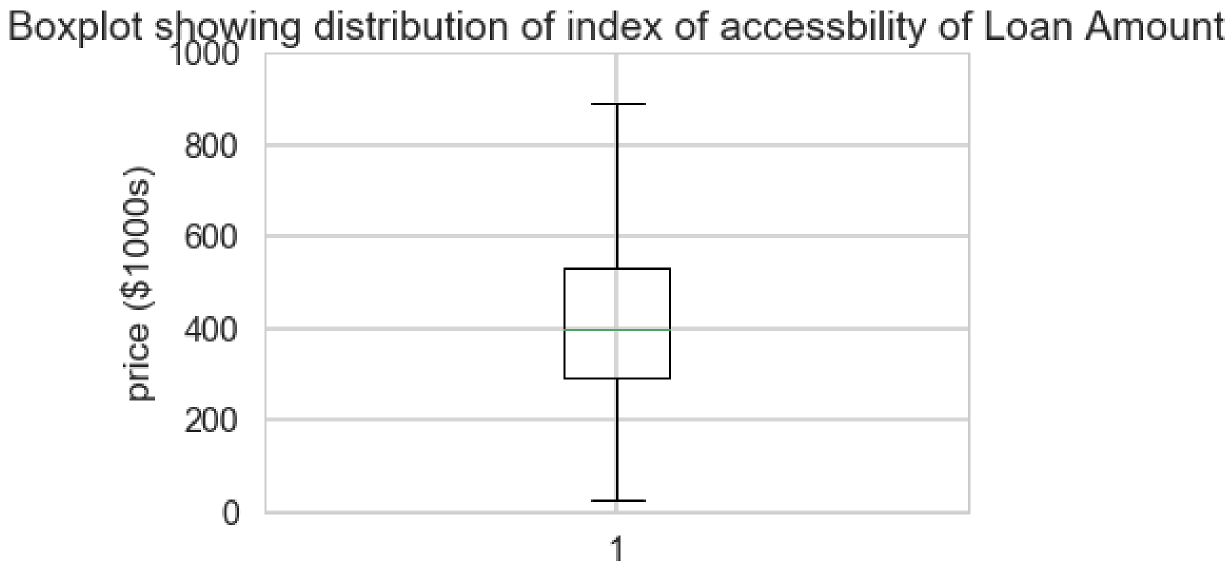
Data Understanding

In the Data Understanding stage the raw data is collected from the database using an SQL query and its characteristics and distributions are explored. Event logs are kept in the database and can be used for predictive modeling. Once the data is collected and saved as **csv** format, the data is explored and visualizations are made of the different variables to get an understanding of the data. With these visualizations we can already see some of the relationships in the data and identify possible features. The data exploration activity is important for becoming familiar with the data and identifying data quality problems.

Data Preparation

The goal of the Data Preparation stage is to transform and enrich the dataset so that it can be fed into the models. After the data is collected and explored, it can be pre-processed so that it can be used directly in our predictive model. With the pre-processed data one can perform feature engineering.

Using historical data and external data, different features can be generated. After the feature engineering activity, a subset of features will be selected that provide predictive value for our models.



Mortgage interest rates

Mortgage interest rates have a significant impact on the amount of mortgage applications. If the interest rates are low, the mortgages are relatively cheaper for the borrower as they have to pay less interest, which leads to an increased amount of mortgage applications. A high mortgage interest rate means the mortgage borrower pays a high amount of interest to the lender, which makes the mortgage less attractive for the borrower. Interest rate changes have a significant impact on mortgage applications, as was seen in November of last year, where a sudden increase in interest rates led to a large peak in mortgage applications. The main difference between the mortgages offered by these types of companies lies in the mortgage interest rates. Even a small difference in mortgage interest rates can often save or cost the borrower a vast amount of money, due to the large sum of a mortgage.

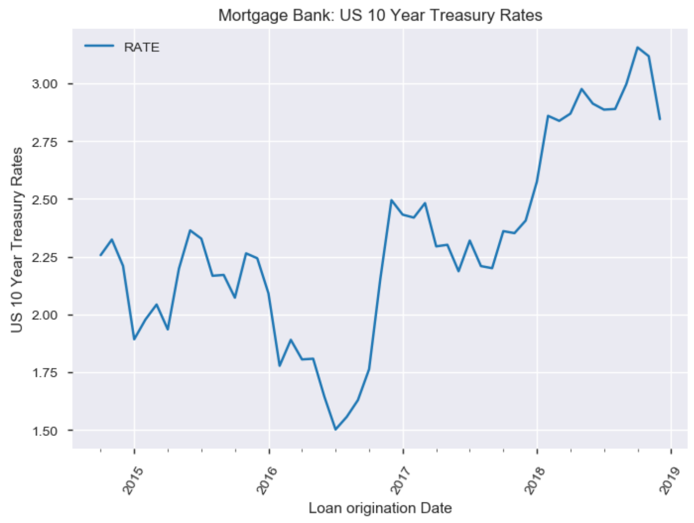
In general, there are two types of mortgage interest rate: variable rates (**ARM**) and fixed rates. Variable interest rates are generally lower than fixed interest rates, but can change every month. Fixed interest rates are slightly higher, but are fixed for a certain period of time. A fixed interest rate is generally preferred when the mortgage interest rates are expected to rise, or when the borrower wants to know its monthly expenses upfront. A variable interest rate (**ARM**) is preferred when interest rates are expected to decrease. If a financial institution has a significantly higher interest rate than its competitors, it will generally receive fewer mortgage applications as the independent mortgage advisors will forward its customers to a different mortgage lender.

For the financial institutions, there can be a number of reasons to change its mortgage interest rate. First of all, the mortgage interest rate is based on the cost of lending for the financial institutions itself.

If the cost of debt is higher, the financial institutions will compensate this by charging higher interest rates for its mortgages, in order to keep a profitable margin on their products. This cost of lending is mainly based on the capital market interest rate, for the long-term loans, and the short-term loans. If either of these changes significantly, one can expect the financial institutions to respond by changing their own mortgage interest rates. This usually happens after a few days.

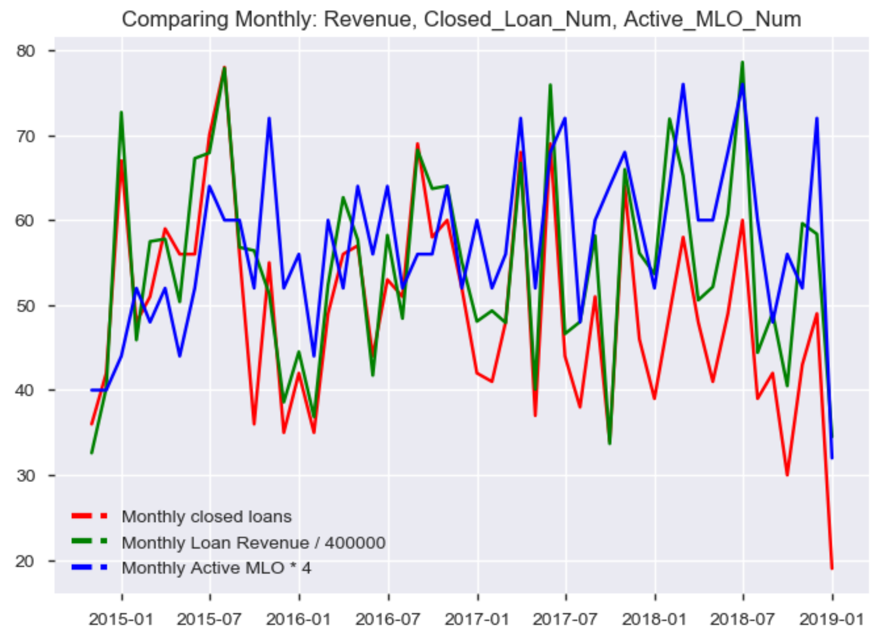
Second, financial institutions generally work with a budget for their mortgages. Based on the amount of funding they can get, and on the interest rates and the duration of the funding, they determine a budget for their mortgages for the upcoming period. Ideally, financial institutions want to match the duration of the fixed interest period of a mortgage with the duration of the lending of debt for that mortgage. Once a financial institution is almost out of budget for a specific fixed interest period, it may choose to increase the interest rate for mortgages with that fixed interest period. This way, borrowers will apply for mortgages with a different fixed interest period, or may choose to go to another mortgage lender.

Finally, financial institutions sometimes increase their interest rates during the summer months, and at the end of the year, as there is less personnel available to handle the requests due to vacations and holidays. With less personnel available they can handle less mortgage requests, so in order to keep the processing time the same they choose to reduce the input, by increasing the interest rates. Financial institutions may also specifically keep interest rates low for mortgages with a certain fixed interest period. Interest rate changes are not always directly influenced by changes in the cost of lending, but can have numerous reasons.
Generally, when US 10 Years Treasury Rate fluctuates, that leads lenders to adjust their internal bank rates accordingly. Also interest rates for consumers varies on several risk factors, such as DTI (Debt to Income Ratio), FICO Scores, Recent derogatory events on their credit history, stable job history, W2 or 1099, Stated Income, Profit or Loss Statements, Student Loans, Auto Payments, Credit utilization, Property types, number of households, rental history, etc.



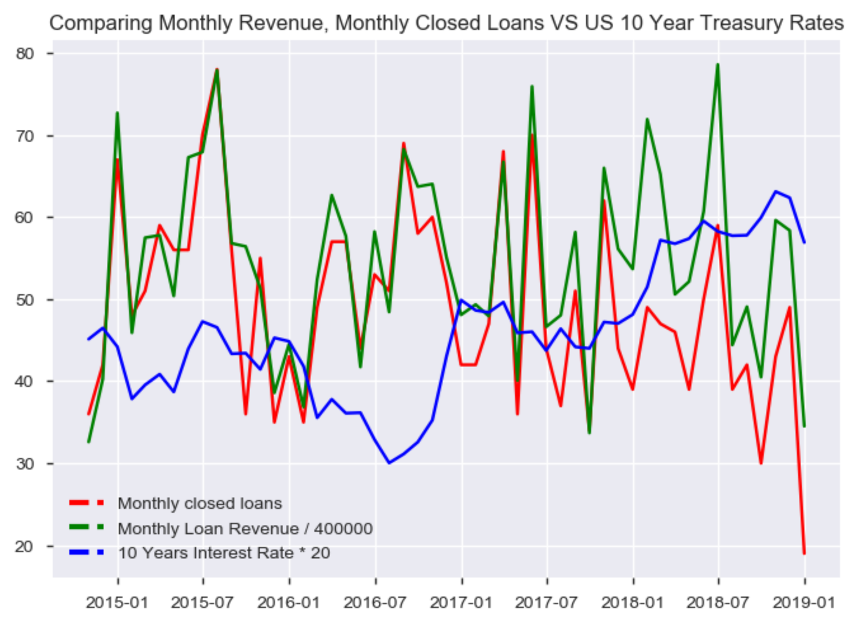
Interest Rate is currently historical low. In the short run rate may go ups and down but in the long run rate will go up. As housing price goes up, interest rate will go up to control the housing price.

Now we will compare Monthly Revenue, Monthly Closed Loan Number and Active Mortgage Loan Originators. We will count number of MLO actively closing loans on any given month.



As we know that number of producer is essential component in any given business. MLO (Mortgage loan Originator) is core component in Mortgage business. Many MLO works indedendently and interect directly to clients, involve in marketting and grow their business. There could be many MLO in Mortgage Bank, but active MLO generate more reverue for the bank. As number of active MLO goes up, which will directly and positively impact numbers of loan closed per month, eventually mortgage revenue will go up. On the other hand, once number of active MLO goes down, mortgage revenue and number of loan per month goes down as well. By visualizing the graphs, we can see that monthly data of closed loan numbers , monthly revenue and active MLO numbers ber months, all moving at the same direction.

Let's find out interest rate effect on Monthly Closed Loans and Monthly Revenue
Pearson correlation coefficient between Monthly Closed Loans & Monthly loan Rev: 0.8295841987344892
Pearson correlation coefficient between Monthly Interest & Monthly loans Closed Data: -0.3343475255276081



We can see the strong positive correlation between Monthly Closed Loans and Monthly Revenue. This graph also suggest that, as interest rates goes down, banks monthly revenue and numbers of loans increases, and when the Rates goes up, both Monthly Closed Loans and Monthly Revenue for the Mortgage bank decline. Pearson correlation coefficient between Monthly Interest & Monthly loans Closed Data is **-0.334**, which clearly proves that Monthly Interest Rates & Monthly loans Closed Data is **negatively correlated**.

Acquire 1000 **pairs bootstrap replicates of the Pearson correlation coefficient using the draw_bs_pairs()** function you wrote in the previous exercise for **CLTV data VS Qualification FICO** Data and **Monthly Loan_num_data VS. Monthly_loan_rev**. Compute the **95% confidence** interval for both using your bootstrap replicates. We have created a NumPy array of percentiles to compute. These are the 2.5th, and 97.5th. By creating a list and convert the list to a NumPy array using np.array(). For example, np.array([2.5, 97.5]) would create an array consisting of the **2.5th and 97.5th percentiles**.

CLTV data VS Qualification FICO Data : -0.037060429592168175 [-0.08 0.]
Monthly Loan_num_data VS. Monthly_loan_rev : 0.8295841987344892 [0.73 0.9]

''' It shows that there is **statistically significant relationship** between number of loans closed and loan revenue'''

Random Walk

Are Interest Rates or Monthly Loan Returns Prices a Random Walk?

Most returns prices follow a random walk (perhaps with a drift). We will look at a time series of Monthly Sales Revenue, and run the 'Augmented Dickey-Fuller Test' from the statsmodels library to show that it does indeed follow a random walk. With the ADF test, the "null hypothesis" (the hypothesis that we either reject or fail to reject) is that the series follows a random walk. Therefore, a low p-value (say less than 5%) means we can reject the null hypothesis that the series is a random walk.

Print out just the p-value of the test (adfuller_loan_rev_data[0] is the test statistic, and adfuller_loan_rev_data[1] is the p-value). Print out the entire output, which includes the test statistic, the p-values, and the critical values for tests with 1%, 10%, and 5% levels.

(-4.49967715626648, 0.00019690419763896495, 10, 40, {'1%': -3.6055648906249997, '5%': -2.937069375, '10%': -2.606985625}, 1307.285137861741)
The p-value of the test on loan_rev is: 0.00019690419763896495

'''According to this test, p-value is very low (lower than 0.05). We reject the hypothesis that monthly_loan_rev_data follow a random walk. '''

Let's try same for Monthly Loan Data:

(-5.7659481612690495, 5.532460937310067e-07, 0, 50, {'1%': -3.568485864, '5%': -2.92135992, '10%': -2.5986616}, 300.2408550906095)
The p-value of the test on loan_num is: 5.532460937310067e-07

According to this test, p-value is very low (lower than 0.05). We reject the hypothesis that monthly_loan_num_data follow a random walk.
Let's try same for Interest Rate Data:

(-1.396544767435691, 0.5839314748568241, 1, 49, {'1%': -3.5714715250448363, '5%': -2.922629480573571, '10%': -2.5993358475635153}, -34.97121939430423)
The p-value of the test on monthly_rate_data is: 0.5839314748568241

According to this test, **p-value is very is higher than 0.05**. We **cannot reject** the hypothesis that Monthly Interest Rate prices follow a random walk.

Are Interest Rates Auto correlated?

When we look at daily changes in interest rates, the autocorrelation is close to zero. However, if we resample the data and look at monthly or annual changes, the autocorrelation is negative. This implies that while short term changes in interest rates may be uncorrelated, long term changes in interest rates are negatively auto correlated. A daily move up or down in interest rates is unlikely to tell us anything about interest rates tomorrow, but a move in interest rates over a year can tell us something about where interest rates are going over the next year. And this makes some economic sense: over long horizons, when interest rates go up, the economy tends to slow down, which consequently causes interest rates to fall, and vice versa. If we want to look at the data by monthly and annually. We can easily resample and sum it up. I'm using 'M' as the period for resampling which means the data should be resampled on a month boundary and 'A' for annual data'. Finally find the autocorrelation of annual interest rate changes.

The autocorrelation of daily interest rate changes is -0.06

```
DATE
2015-12-31    0.10
2016-12-31    0.18
2017-12-31   -0.05
2018-12-31    0.37
Freq: A-DEC, Name: RATE, dtype: float64
The autocorrelation of annual interest rate changes is -0.97
```

'''Notice how the daily autocorrelation is small but the annual autocorrelation is large and negative'''

PREDICTIVE ANALYTICS

Predictive analytics is a field in data mining that encompasses different statistical and machine learning techniques that are aimed at making empirical predictions. These predictions are based on empirical data, rather than predictions that are based on theory only. In predictive analytics, several statistical and machine learning techniques can be used to create predictive models. These models are used to exploit patterns in historical data, and make use of these patterns in order to predict future events. These models can be validated using different methods to determine the quality of such a model, in order to see which model performs best.

There are generally two types of problems predictive analytics is used for: classification problems and regression problems. The main difference between these two problems is the dependent variable, the target variable that is being predicted. In classification problems, the dependent variable is categorical (e.g. credit status). In regression problems, the dependent variable is continuous (e.g. pricing).
The techniques that are used in predictive analytics to create a model depend heavily on the type of problem. For classification problems, classification techniques are used such **Random Forest and decision trees**. These techniques often consist out of one or multiple algorithms that can be used to construct a model. For decision trees, some of the algorithms are Classification and Regression Trees.

For regression problems, **regression techniques** such as multiple **linear regression, support vector machines or time series** are used. These techniques focus on providing a mathematical equation in order to represent the interdependencies between the independent variables and the dependent variable, and use these to make predictions. One of the most popular regression techniques is linear regression. When applied correctly, regression is a powerful technique to show the relationships between the independent and the dependent variables. However, linear regression requires some assumptions in the dataset. One of these assumptions is that there has to be a linear interdependency between the independent variables and the dependent variable. A pitfall of linear regression is that the regression line contains no information about the distribution of the data. It needs to be combined with a visualization of the regression line in order to draw conclusions.

DATA UNDERSTANDING

The Data Understanding stage has been split up in two parts: data collection and data exploration. In the data collection we will discuss characteristics of the event log data, and how the data has been extracted from the database. In the data exploration, the data and its characteristics are explored to extract useful information for our models.

DATA COLLECTION

Since we are only interested in the event log data we will only be using one of the tables. This table contains data about every mortgage application. Every action performed by the system or by a user on a mortgage application is logged, and the status before and after that specific action is logged. For our analysis we are mainly interested in the date and time at which each of the mortgage applications have entered the system.

Besides Mortgage Application DataSet, we have joined two separate (**10 Years US Treasury Rate, Home Supply Index)** with our existing Mortgage Application DataSet to enhance predictive power of our model.

DATA EXPLORATION

Since our dataset can be grouped per day to create meaningful visualizations. The dataset contains data from October 2014 until December 2018. As can be seen from the graphs, there seems to be a seasonal pattern on a monthly level, but from these graphs it is not very clear. It also seems like there are some outliers, so these data points will have to be investigated to see if they will have to be included in our model, as there can be multiple underlying reasons for outliers in our dataset. It also seems there is an increase in mortgage applications during the last few months of each year. The amount of applications per day during these months is higher compared to the other months. This can have multiple explanations so this will have to be accounted for in the model.

The density plot shows the distribution of the amount of mortgage applications. It seems the distribution of the amount of mortgage applications is normally distributed, slightly skewed to the right with a long tail. This is due to the outliers mentioned before. The median seems to be at around mortgage applications per day.

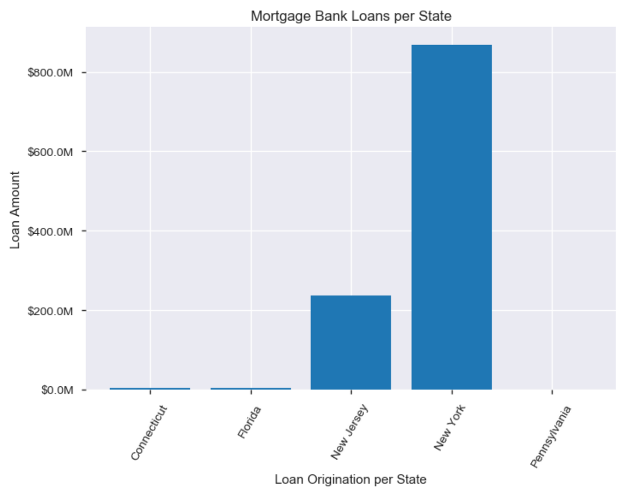
We can analyze mortgage loan origination for the bank per state; we also can find the total counts of different types of loan for the bank. Based on this information bank can allocate resources:

=========   Total Sales by State   ==============

Total Sales in Connecticut	***: $ 3604641.0***
Total Sales in Florida	***: $ 3371495.0***
Total Sales in New York	***: $ 868836552.28***
Total Sales in New Jersey	***: $ 236456261.06***

```
Total Sales in Pennsylvania      : $ 600920.0

=====================================================

Unique Loan Types      :  ['Residential' 'FHA' 'Commercial' 'Conventional' 'Other' 'VA']

=====================================================
Number of loan per Types :  Loan Type
Commercial                  100
Conventional                1736
FHA                         418
Other                       31
Residential                 215
VA                          1
```

Let's visualize the loan data per state;



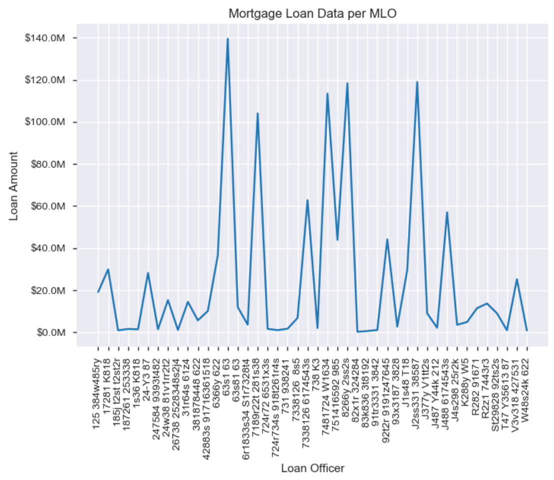
We can see that New York and New Jersey is the major Loan Origination market for the Bank. Let's explore Loan Origination Data with Unit Types:

```
        total_unit_type = data.groupby(data['Unit Type']).size()
        print('Loan originated in all States per unit types : \n', total_unit_type)
```

```
==============================
 Loan originated in all States per unit types :
 Unit Type
Condo          216
Coop           76
FourFamily     36
Industrial     1
Land           1
MixedUse       25
MultiFamily    20
OneFamily      1276
PUD            6
```

ThreeFamily	**129**
TwoFamily	**707**
Warehouse	**8**

We can see that One Family and Two Family property dominates the Loan Origination shares for the mortgage bank. As a result company should focus on suitable loan products particularly for 1-4 family loan products. This Mortgage Bank also originates good number of Condo and Coop.



Some MLO generates major portions of the sales revenue for the Bank, many of the MLO is performing way below company standards. Additional research required for the root causes behind the performance. Let's explore the Loan Statistics data for the Mortgage Bank:

************************* Loan Statistics for Mortgage Bank ***************************

Average Loan Amount is	***: $ 445099.91***
Median Loan Amount is	***: $ 400000.00***
Standard deviation of is	***: $ 288803.96***
Minimum Loan Amount is	***: $ 23600.00***
Maximum Loan Amount is	***: $ 6125000.00***
Total of Loan Amount is	***: $ 1113194869.34***
10% of Loan Amount is below	***: $ 200000.00***
25% of Loan Amount is below	***: $ 292500.00***
50% of Loan Amount is below	***: $ 417000.00***
75% of Loan Amount is below	***: $ 533000.00***
90% of Loan Amount is below	***: $ 696500.00***

FICO Score is one of the critical information for processing mortgage loan application.

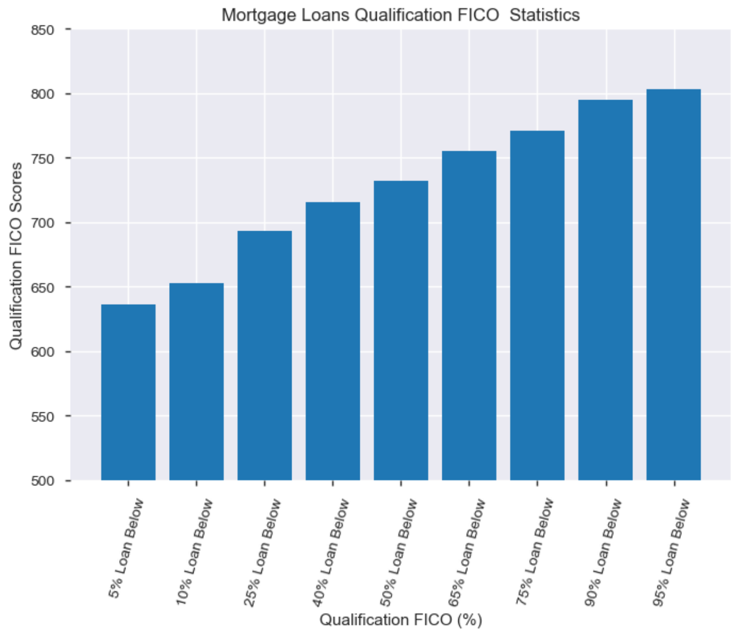
****************** Qualification FICO Statistics for Mortgage Bank ******************

Average FICO is	***: $ 727.81***
Median FICO is	***: $ 732.00***
Standard deviation is	***: $ 56.61***
Minimum FICO is	***: $ 0.00***
Maximum FICO is	***: $ 825.00***
5% of FICO is below	***: $ 636.00***
10% of FICO is below	***: $ 653.00***
25% of FICO is below	***: $ 693.00***

40% of FICO is below	**:**	***$ 716.00***
50% of FICO is below	**:**	***$ 732.00***
65% of FICO is below	**:**	***$ 755.00***
75% of FICO is below	**:**	***$ 771.00***
90% of FICO is below	**:**	***$ 795.00***
95% of FICO is below	**:**	***$ 803.00***

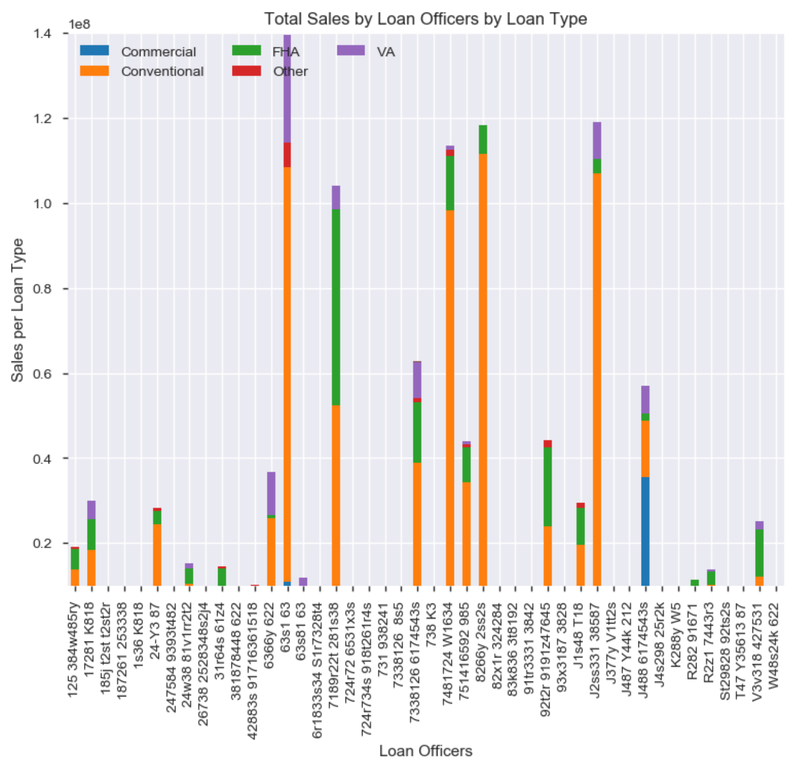
===

We can see that 50% of the consumers FICO score is between 693 and 771. Let's visualize FICO statistics
fico_score = [fico_q7, fico_q0, fico_q1, fico_q5, fico_q2, fico_q6, fico_q3, fico_q4, fico_q8]
fico_pct=['5% Loan Below', '10% Loan Below', '25% Loan Below', '40% Loan Below', '50% Loan Below', '65% Loan Below', '75% Loan Below','90% Loan Below', '95% Loan Below']

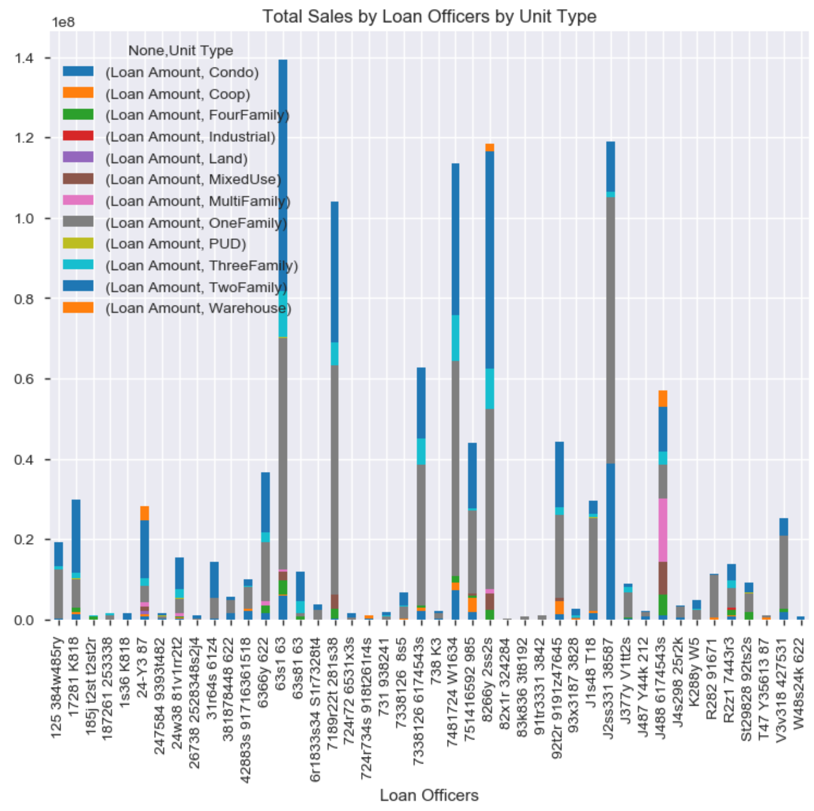


We can see that 90% of the consumers FICO score is over 650. Most of cases applicant with lower FICO score will not qualify for Conventional mortgages. In many cases, FHA Loan type could be only option remaining for the applicants with FICO score; many bank uses cut-off points for FICO Score (640-680) for conventional mortgages. FHA accepts FICO score below 600.

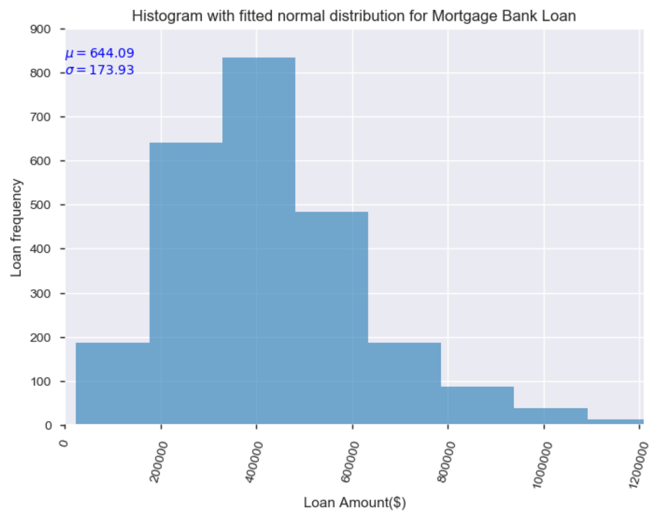
Let's Analyze Mortgage Loan Officer's Sales Volume per Loan Type. Processing different loan types need different expertise. Some loan officer do not have experience for VA or commercial loan at all. Few Loan officers are expert in commercial side of loan origination process. In order to Loan Origination for 1-4 units of residential properties, MLO must be licensed by the State. On the other hand other properties such as multi-family, mixed use or commercial loans do not require MLO to be licensed by the State.



Similarly, we can explore Loan Officer's Sales Volume per Unit Type.



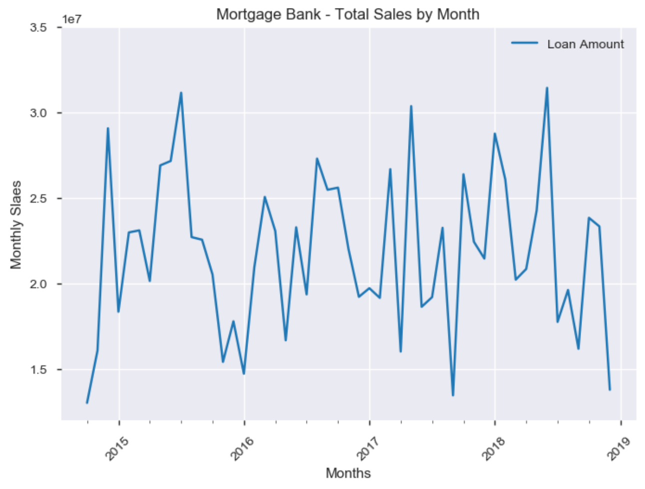
Now that we know who the biggest customers are and how they purchase products, we might want to look at purchase patterns in more detail. Let's take another look at the data and try to see how large the individual purchases are. A histogram allows us to group purchases together so we can see how big the customer transactions are.



excess kurtosis of normal distribution (should be 0): 10.20423896299732
skewness of normal distribution (should be 0): 1.6231675369974472
mean : 644.086535155362
var : 30252.443004451427
skew : 1.6231675369974472
kurt : 10.20423896299732

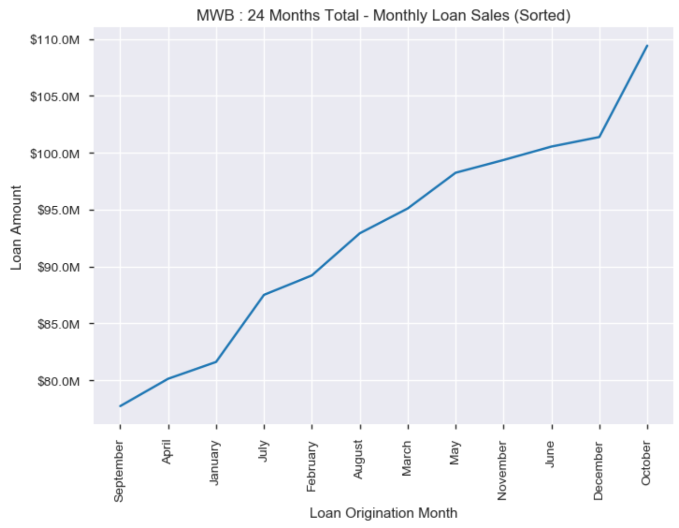
The fancy text to show us what the parameters of the distribution are (mean and standard deviation). We can create a histogram with 20 bins to show the distribution of purchasing patterns. Fit a normal distribution to the data then plot the histogram.

One of the really cool things that **Pandas** allows us to do is **resample** the data. We want to look at the data by month, we can easily resample and sum it all up. I'm using 'M' as the period for resampling which means the data should be resampled on a month boundary.



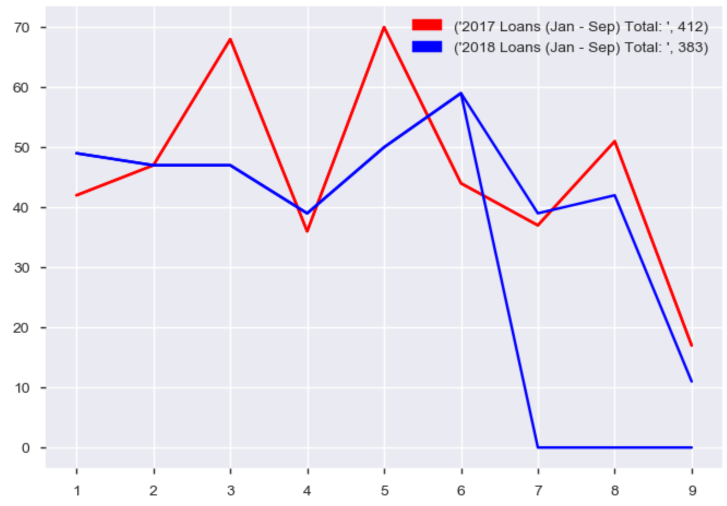
We can see that monthly mortgage loan sales volume varies between $15M and $32M. Another interesting find is Loan sales are at the peak during summer seasons. Winter sales are normally slow but 2018 was an exception.
data = pd.read_csv('c:/scripts/mwb2014.csv', index_col='Created Date', parse_dates=True, encoding='cp1252')

Best month for Sales

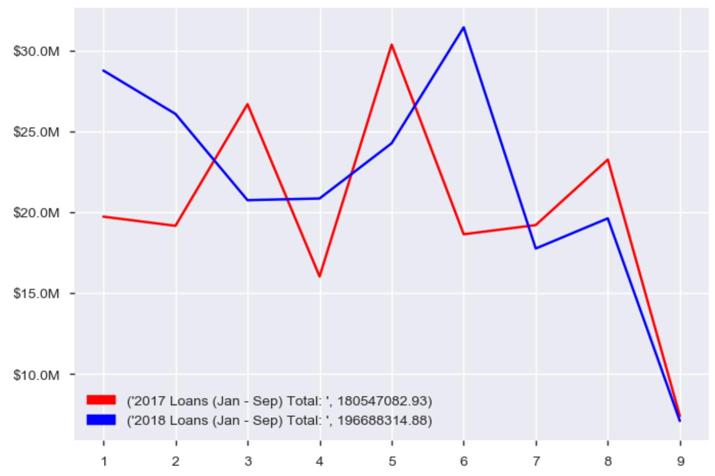


Monthly Worst Sales: January, April and September and Best is June, December and October.

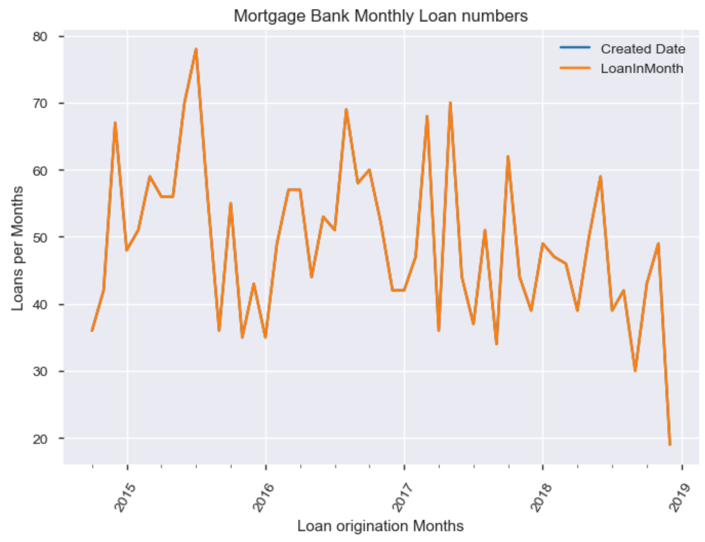
Comparing year-over-year performance:



In 2017, Mortgage Bank has closed more loan compare to 2018



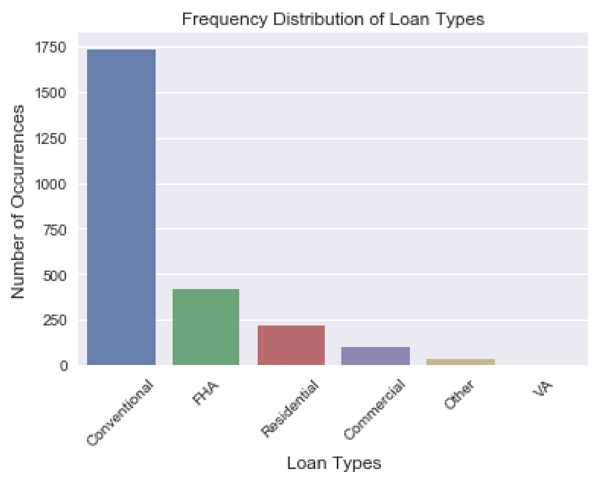
In 2017, Mortgage Bank sales revenue was (M_amount_17.sum()) = $180M
For 2018, (M_amount_18.sum())= $197M

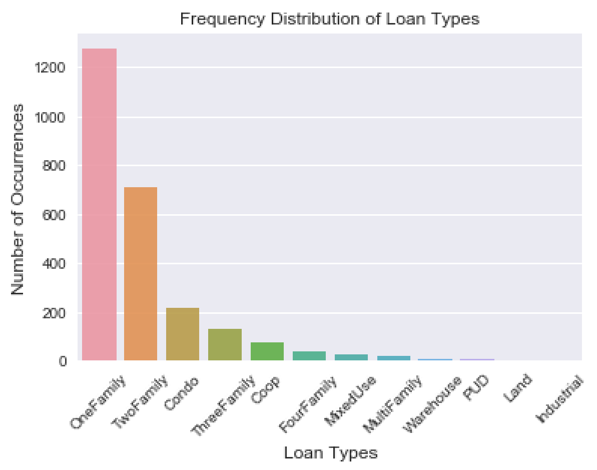


Pearson correlation coefficient between Monthly Closed Loans & Monthly loan Rev: 0.8295841987344892
Monthly Loan numbers and Monthly Sales volumes are strongly positively correlated. Once we analyze number of the loans closed per month, we find the similarity between sales volume and loan numbers. As average loan numbers goes up, total monthly sales volume goes up. Any given months, if the number of loans closed are higher; we find that average loan amounts are low for that particular months. In other words, this may suggest that loan processing requirements and guidelines loans with higher loan amount take longer time to close the loans with lower loan amounts.

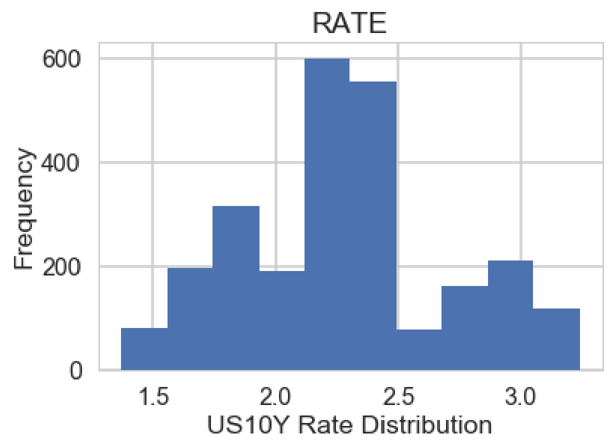
'''Visual exploration is the most effective way to extract information between variables.

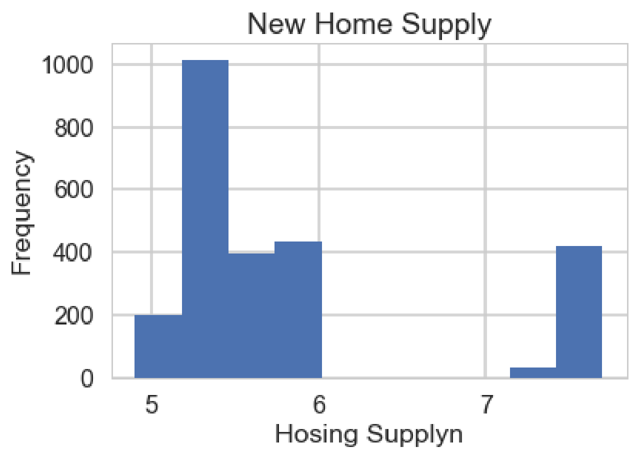
We can plot a barplot of the frequency distribution of a categorical feature using the seaborn package, which shows the frequency distribution of the mortgage dataset column.

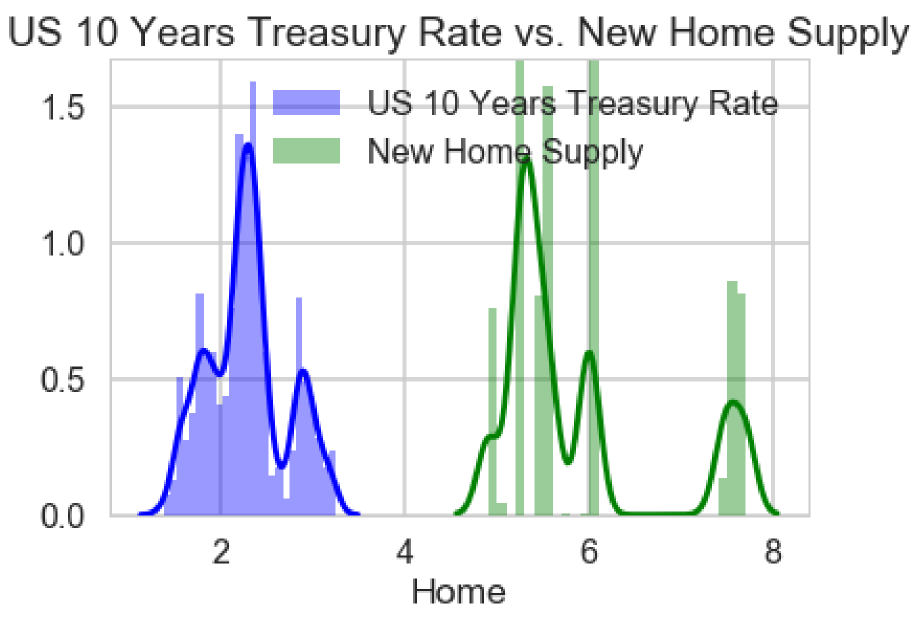




#The distribution is somewhat left skewed with a mean to the left







''' Home Supply goes up, RATE goes up ''' '''
The distribution plot comparing US 10 Years Treasury Rate & New Home Supply shows that US 10Y RATE is normally distributed and New Home Supply is skewed to the right.

DATA PREPARATION

The Data Preparation stage contains three elements: data pre-processing, feature engineering and feature selection. In the data pre-processing we pre-process the data so that it contains the right format for our models. In the feature engineering we create a list of features, using the predictors. In the feature selection we select a subset of features that are useful for our model.

Encoding Categorical Data

There are different techniques to encode the categorical features to numeric quantities.
The techniques are as following:

- Replacing values
- Encoding labels
- One-Hot encoding
- Replace Values

Let's start with the most basic method, which is just replacing the categories with the desired numbers. This can be achieved with the help of the replace() function in pandas. The idea is that you have the liberty to choose whatever numbers we want to assign to the categories according to the business use case.

validate the faster operation of the category dtype by timing the execution time of the same operation done on a DataFrame with columns as category dtype and object dtype by using the time library.

Label Encoding

Another approach is to encode categorical values with a technique called "label encoding", which allows you to convert each value in a column to a number. Numerical labels are always between 0 and n_categories-1.

We can do label encoding via attributes .cat.codes on your DataFrame's column.'''

```
data_lc['city_code'] = data_lc['City'].cat.codes
data_lc['zip_code'] = data_lc['Zip'].cat.codes
```

One-Hot encoding

The basic strategy is to convert each category value into a new column and assign a 1 or 0 (True/False) value to the column. This has the benefit of not weighting a value improperly.

```
data_lc = pd.get_dummies(data_lc, columns=['Fix'], prefix = ['Fix'])
data_lc['Fix_True']
data_lc['Fix_False']
data_lc['Fix_True'].head()
data_lc['Fix_True'].value_counts()
```

DATA PRE-PROCESSING

In the data pre-processing phase we pre-process our data to a suitable format for our predictive model. This phase consists out of the following steps. First, the data is grouped by day. Since we want to get insight in the amount of personnel needed on a daily level, we want to group the amount of mortgage applications per working day.

Second, as there are minimal applications coming in during the weekends, we choose to transfer these applications to Monday. Third, mortgage applications that entered the system before the 14th of October 2014 have been removed from our dataset. As this data would not be useful for our model, it is excluded. Finally, missing dates have been added to the model, as these need to be predicted as well.

Handling Categorical Data in Python

Learn the common tricks to handle categorical data and preprocess it to build machine learning models! The categorical features in many datasets generally include different categories or levels associated with the observation, which are non-numerical and thus need to be converted so the computer can process them. The difference between categorical and continuous data in your dataset and identifying the type of data to do basic exploration of such data to extract information from it.

One of the most common data pre-processing steps is to check for null values in the dataset. You can get the total number of missing values in the DataFrame by the following one liner code. We can do a mode imputation for those null values. The function fillna() is handy for such operations.

```
print(data['Zip'].isnull().values.sum())
#Let's also check the column-wise distribution of null values:
print(data['Zip'].isnull().sum())
```

Find Unique Values

The chaining of method .value_counts() in the code below. This returns the frequency distribution of each category in the feature, and then selecting the top category, which is the mode, with the .index attribute.

Another Exploratory Data Analysis (EDA) step to do on categorical features is the frequency distribution of categories within the feature, which can be done with the .value_counts() method.'''

```
data['City'].value_counts()
```

To know the count of distinct categories within the feature you can chain the previous code with the .count() method:

```
data['City'].value_counts().count()
```

Preprocessing: scaling

Here below I (i) scale the data, (ii) use k-Nearest Neighbors and (iii) check the model performance. I'll use scikit-learn's scale function, which standardizes all features (columns) in the array passed to it.'''

```
k-NN score for test set: 0.678643
k-NN score for training set: 0.770000 or 77%
       precision   recall  f1-score   support

    0     0.46     0.38     0.41      16
    1     0.73     0.88     0.80     347
    2     0.41     0.21     0.27      87
    3     0.00     0.00     0.00       2
    4     0.48     0.21     0.29      48
    5     0.00     0.00     0.00       1

avg / total    0.64     0.68     0.64     501
```

All these measures improved by 0.1325, which is a **13.25% improvement and significant**! As hinted at above, before scaling there were a number of predictor variables with ranges of different order of magnitudes, meaning that one or two of them could dominate in the context of an algorithm such as k-NN. The two main reasons for scaling our data are

Our predictor variables may have significantly different ranges and, in certain situations, such as when implementing k-NN, this needs to be mitigated so that certain features do not dominate the algorithm; We want our features to be unit-independent, that is, not reliant on the scale of the measurement involved. If we both scale our respective data, this feature will be the same for each of us.

FEATURE ENGINEERING

Feature engineering is the process of encoding the predictors in a way that they can be useful for prediction. We generate features for our model that may have predictive power based on our domain knowledge of the mortgage application domain. There are a number of predictors that influence the amount of mortgage applications.

FEATURE SELECTION

Feature Selection is one of the core concepts in machine learning which hugely impacts the performance of your model. The data features that you use to train your machine learning models have a huge influence on the performance you can achieve.

How to select features and what are Benefits of performing feature selection before modeling your data?
Reduces Overfitting: Less redundant data means less opportunity to make decisions based on noise.
Improves Accuracy: Less misleading data means modeling accuracy improves.Reduces
Training Time: fewer data points reduce algorithm complexity and algorithms train faster.

Feature Selection Methods:

I will share 3 Feature selection techniques that are easy to use and also gives good results.

- **Univariate Selection**
- **Feature Importance**
- **Correlation Matrix with Heatmap**

1. Univariate Selection
Statistical tests can be used to select those features that have the strongest relationship with the output variable. The scikit-learn library provides the SelectKBest class that can be used with a suite of different statistical tests to select a specific number of features.
The example below uses the chi-squared (chi²) statistical test for non-negative features to select 10 of the best features from the Mobile Price Range Prediction Dataset.

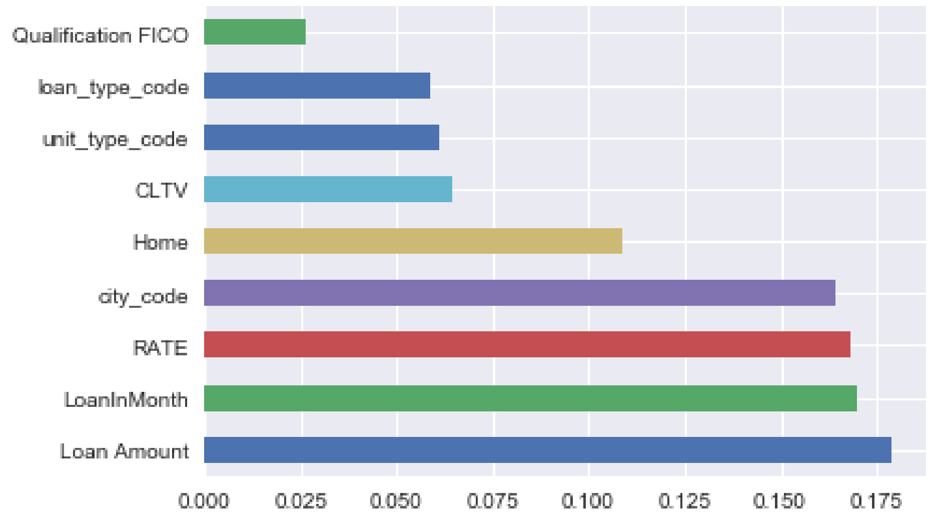
		Score
6	**CLTV**	**80.536598**
0	**Loan Amount**	**16.250345**

4	**unit_type_code**	**14.523317**
3	**Qualification FICO**	**14.370298**
2	**loan_purpose_code**	**12.874750**
1	city_code	6.169496
5	loan_type_code	1.500709
8	RATE	1.213212
7	LoanInMonth	1.186288

2. Feature Importance

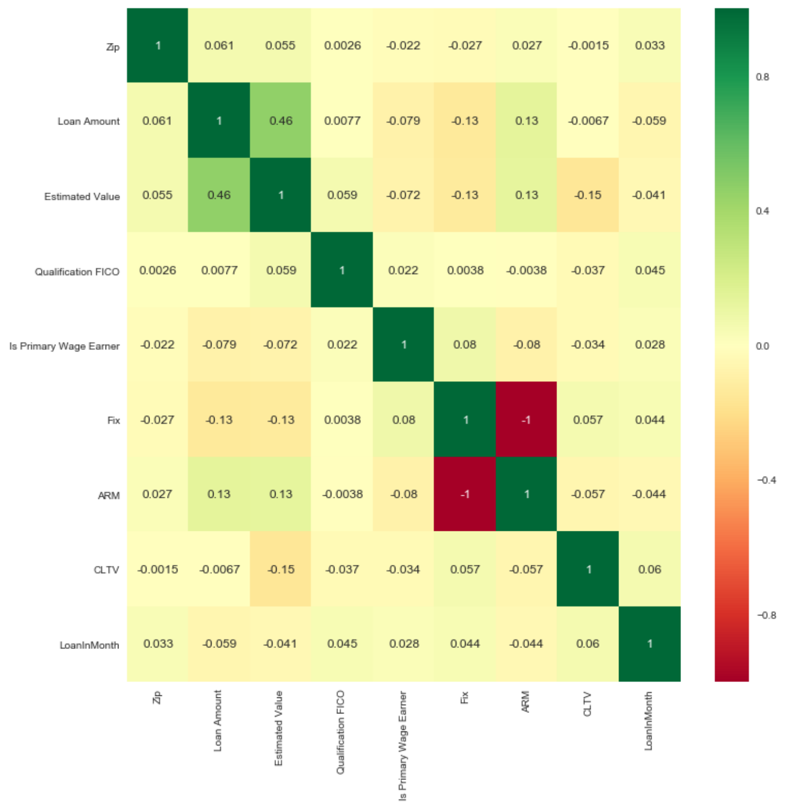
The feature importance of each feature of our dataset by using the feature importance property of the model. Feature importance gives us a score for each feature of our data, the higher the score more important or relevant is the feature towards your output variable. Feature importance is an inbuilt class that comes with Tree Based Classifiers, we will be using Extra Tree Classifier for extracting the top 10 features for the dataset.'''

[0.18 0.16 0.03 0.06 0.06 0.06 0.17 0.17 0.11]



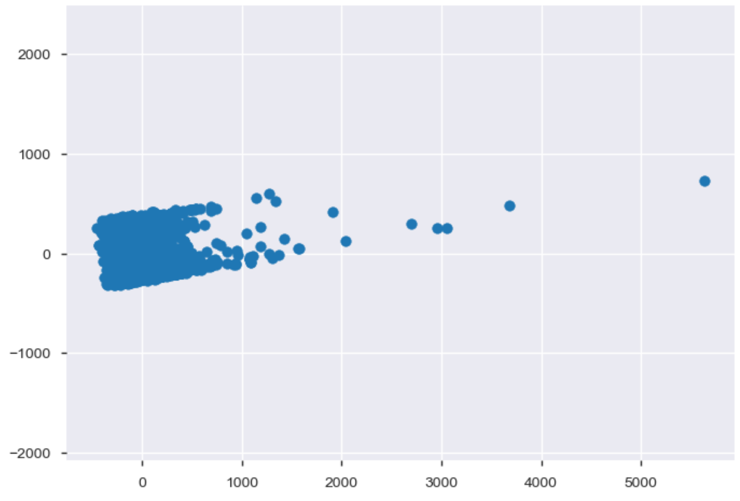
3. Correlation Matrix with Heatmap

Correlation states how the features are related to each other or the target variable.
Correlation can be positive (increase in one value of feature increases the value of the target variable) or negative (increase in one value of feature decreases the value of the target variable)
Heatmap makes it easy to identify which features are most related to the target variable, we will plot heatmap of correlated features using the seaborn library.



PCA (Principal Component Analysis)

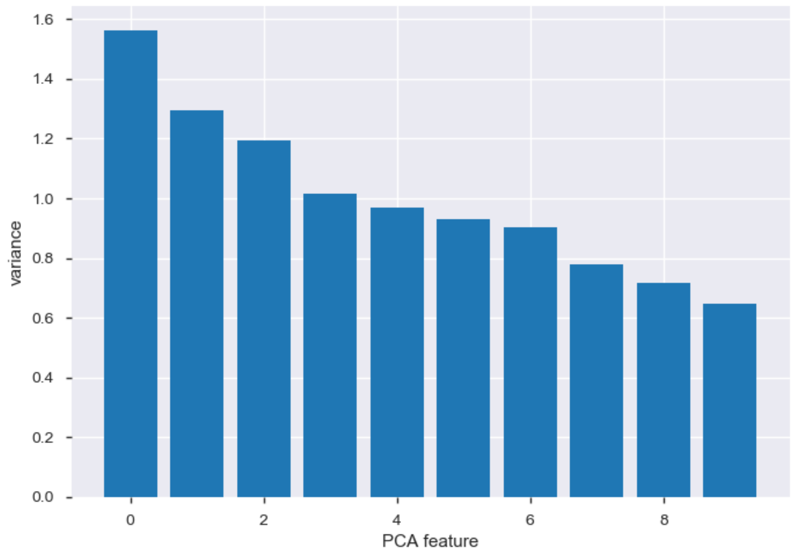
PCA use PCA to de-correlate these measurements, then plot the de-correlated points and measure their Pearson correlation. Compute Pearson correlation coefficient between two arrays.



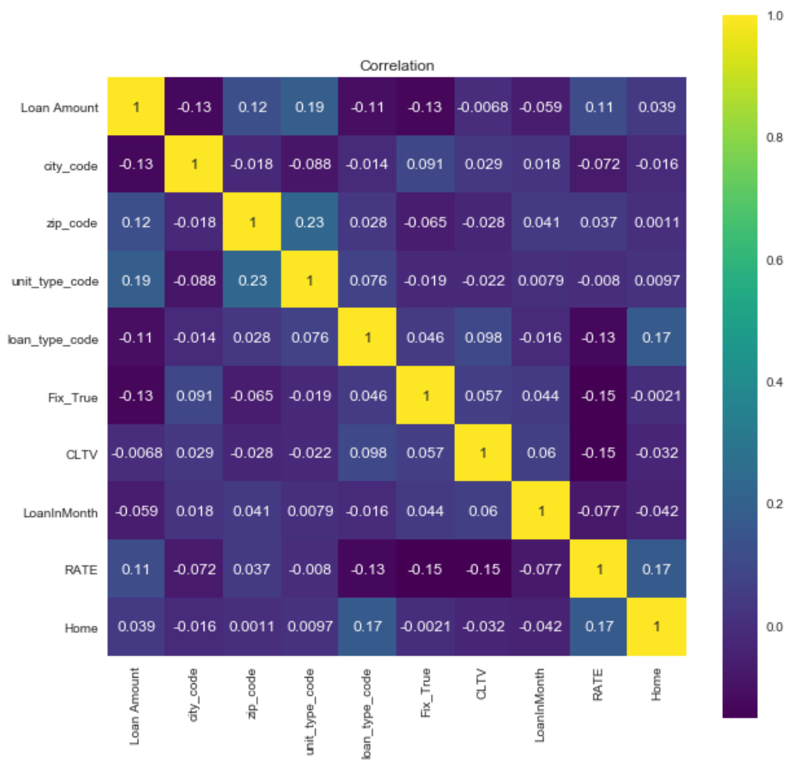
Variance of the PCA features

The dataset is 10-dimensional. But what is its intrinsic dimension? Make a plot of the variances of the PCA features to find out. As before, samples is a 2D array, where each row represents a fish. You'll need

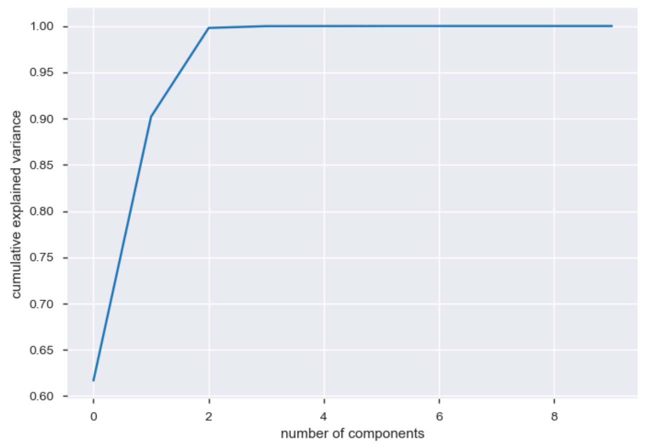
to standardize the features first. Tthe use of principal component analysis for dimensionality reduction, for visualization of high-dimensional data, for noise filtering, and for feature selection within high-dimensional data. Because of the versatility and interpretability of PCA, it has been shown to be effective in a wide variety of contexts and disciplines. Given any high-dimensional dataset, I tend to start with PCA in order to visualize the relationship between points), to understand the main variance in the data and to understand the intrinsic dimensionality (by plotting the explained variance ratio).
Now we want to know how many principal components we can choose for our new feature subspace. A useful measure is the "explained variance ratio". The explained variance ratio tells us how much information (variance) can be attributed to each of the principal components. We can plot bar graph



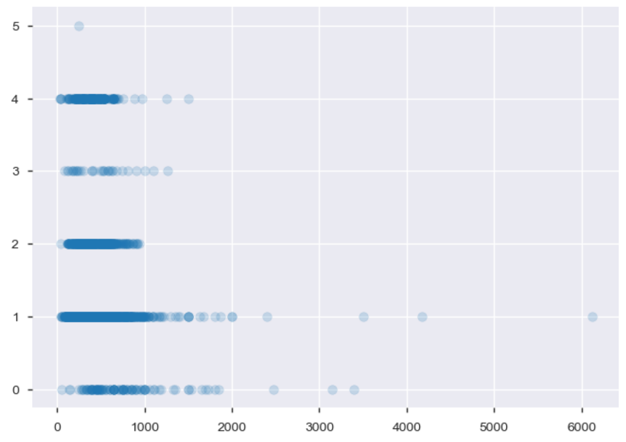
'''Finding Correlation between Features and Target Variable in mortgage Dataset using Heatmap'''



#Let us load the basic packages needed for the PCA analysis



To see what these numbers mean, let's visualize them as vectors over the input data, using the "components" to define the direction of the vector, and the "explained variance" to define the squared-length of the vector:

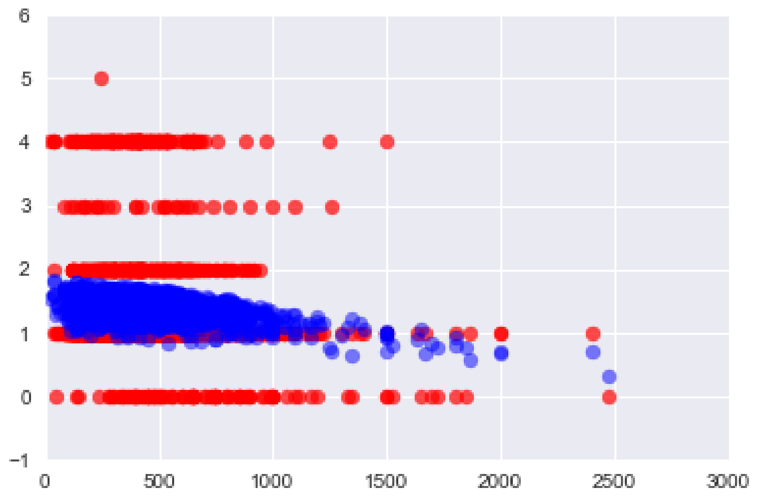


PCA as dimensionality reduction

Using PCA for dimensionality reduction involves zeroing out one or more of the smallest principal components, resulting in a lower-dimensional projection of the data that preserves the maximal data variance.

Here is an example of using PCA as a dimensionality reduction transform:

original shape: (2501, 9)
transformed shape: (2501, 6)
The transformed data has been reduced to a 6 dimension. To understand the effect of this dimensionality reduction, we can perform the inverse transform of this reduced data and plot it along with the original data:'''



The light points are the original data, while the dark points are the projected version. This makes clear what a PCA dimensionality reduction means: the information along the least important principal axis or axes is removed, leaving only the component(s) of the data with the highest variance. The fraction of variance that is cut out (proportional to the spread of points about the line formed in this figure) is roughly a measure of how much "information" is discarded in this reduction of dimensionality.

This reduced-dimension dataset is in some senses "good enough" to encode the most important relationships between the points: despite reducing the dimension of the data by 50%, the overall relationship between the data points are mostly preserved.

MODELING

In the Modeling stage we discuss the activities related to the model building part of our project. A selection of five modeling techniques is made that are applicable to our capstone project. From each of these five modeling techniques, a model is built with the feature set provided earlier, and the models are validated using repeated cross-validation.

SELECTION OF MODELING TECHNIQUES

For our modeling we use a combination of predictive techniques. Multiple techniques are selected and applied on the data. For the non-linear regression techniques, we use Support Vector Regression (SVR) and Neural Networks (NN). SVR has shown to obtain excellent performances in regression and time series applications. Neural Networks are a widely used method for time series data that generally gives mixed results.

MODEL BUILDING

Using these five techniques (Linear Regression, Logistic Regression, SVM, SVR, Decision Tree, RF, and KNN) we can create five models. We use the list of features mentioned in section 6.3 as input for our models. A total of 12 features are included, the remaining features were excluded after performing feature selection. For each of the five models hyperparameters were tuned, using grid search. Hyperparameters are the model-specific parameters that are used for optimizing the model. They generally have to be tuned in order to optimize the model's performance, and reduce the variance and bias of the model. By training the model with different values of the size and the decay, and evaluating its performance, we can select the hyperparameters that result in the best performing model in terms of predictive power.

ARMA Model

Estimating an AR Model

We will estimate the AR(1) parameter, φ, of one of the Rate, Revenue, Loan_num, series that generated in the earlier . Since the parameters are known for a series, it is a good way to understand the stimation routines before applying it to real data. For monthly_rate_data with a true φ of 0.9, we will print out the estimate of φ. In addition, we will also print out the entire output that is produced when you fit a time series, so we can get an idea of what other tests and summary statistics are available in statsmodels.

```
ARMA Model Results
==============================================================================
Dep. Variable:               y          No. Observations:              51
Model:                  ARMA(1, 0)      Log Likelihood                 23.792
Method:                 css-mle         S.D. of innovations            0.149
```

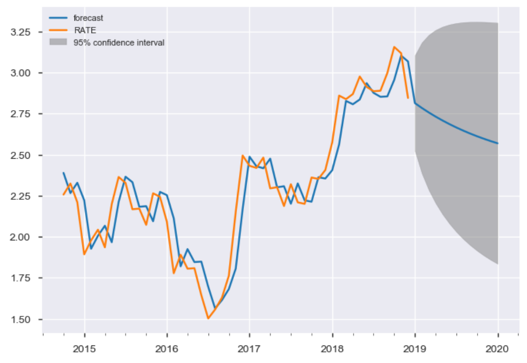
```
Date:                    Thu, 31 Jan 2019        AIC                         -41.584
Time:                    11:49:00                BIC                         -35.788
Sample:                  0                       HQIC                        -39.369
==============================================================================
            coef    std err        z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const       2.3886      0.247      9.653      0.000       1.904       2.874
ar.L1.y     0.9310      0.045     20.707      0.000       0.843       1.019
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1            1.0741           +0.0000j            1.0741            0.0000
-----------------------------------------------------------------------------
When the true phi=0.9, the estimate of phi (and the constant) are:[2.39 0.93]
```

Forecasting with an AR Model

In addition to estimating the parameters of a model, we can also do forecasting using statsmodels. The in-sample is a forecast of the next data point using the data up to that point, and the out-of-sample forecasts any number of data points in the future. These forecasts can be made using either the predict() method if we want the forecasts in the form of a series of data, or using the plot_predict() method we you want a plot of the forecasted data. We will supply the starting point for forecasting and the ending point, which can be any number of data points after the data set ends.

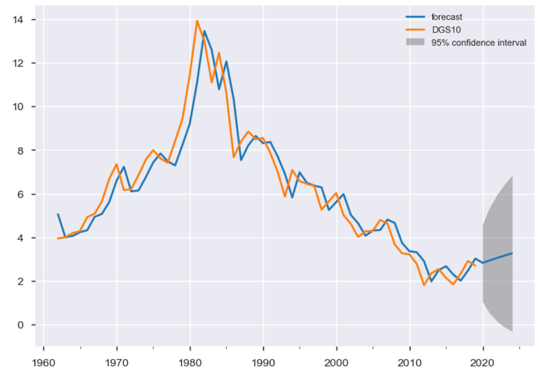
For the simulated series Monthly Interest Rate with **φ=0.9**, we will plot in-sample and out-of-sample forecasts. Being able to forecast interest rates is of enormous importance, not only for bond investors but also for individuals like new **homeowners** who must decide **between fixed and floating rate** mortgages.

There is some **mean reversion** in interest rates over long horizons. In other words, when interest rates are high, they tend to drop and when they are low, they tend to rise over time. Currently they are below long-term rates, so they are expected to rise, but an **AR model** attempts to quantify how much they are expected to rise.



Since we have only used only 4 years of Monthly Interest Rate Data, we can see the short term downward momentum on the rate. A daily move up or down in interest rates is unlikely to tell us anything about interest rates tomorrow, but a move in interest rates over a year can tell us something about where interest rates are going over the next year. The DataFrame daily_data contains daily data of **10-year interest rates from 1962 to 2017**.

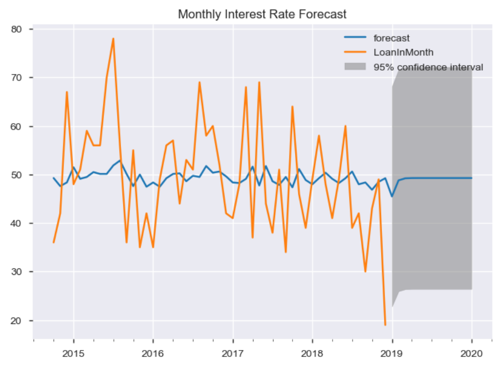
The autocorrelation of daily interest rate changes is 0.07
The autocorrelation of annual interest rate changes is -0.22



Over long horizons, when interest rates go up, the economy tends to slow down, which consequently causes interest rates to fall, and vice versa. According to an AR(1) model, 10-year interest rates are forecasted to rise from 2.16%, towards the end of 2017 to 3.35% in five years.

Forecast expected monthly closed loans

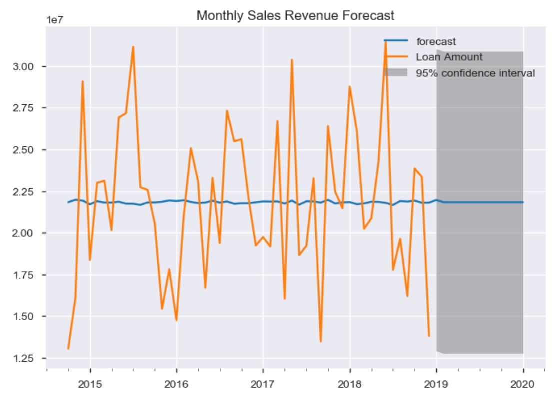
Our Mortgage DataSet contains data from Oct 2014 to December 2018. Let's forecast Monthly Closed Loans and Monthly Revenue. We will **forecast monthly_loan_num_data** using an AR(1) model



'''Above we have plotted the original series and the forecasted series. With 95% confidence interval, Expected Loans per month will be around 50. Low end is 28 & High End is 70'''

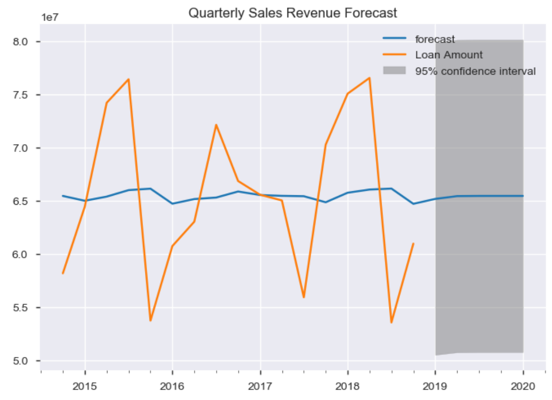
Forecast expected monthly revenue

Similarly, we may forecast expected monthly revenue and plot original series and the forecasted series



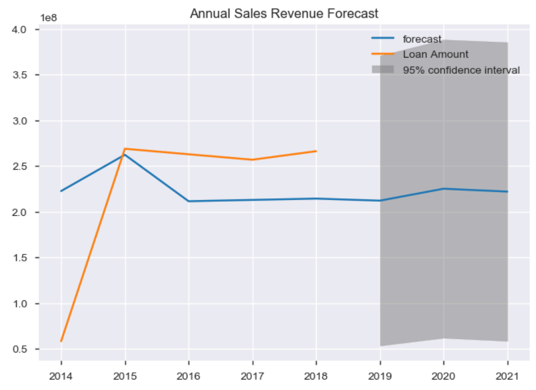
With 95% confidence interval, Expected Revenue per month will be around 22M. Low end is 13M & High End is 32M

Forecasting Quarterly Sales Revenue



We have plotted the original series and the forecasted series. With **95% confidence** interval, Expected **Revenue per Quarters will be around 67M**. Low end is 52M & High End is 80M

Forecasting Annual Sales Revenue



In the above graph, we have plotted the original series and the forecasted series. We are expecting little drop in annual mortgage revene. Our current annual revenue is $270M. By end of 2021, with 95% confidence interval, Expected Revenue per Years will be around 220M. Low end is 70M & High End is 360M. With more annual data, we should be able to do better annual revenue prediction.

Linear regression

Purpose of linear regression
Given a dataset containing predictor variables X and outcome/response variable Y, linear regression can be used to:

Build a predictive model to predict future values, using new data X where Y is unknown.
Model the strength of the relationship between each independent variable X_i and Y
Many times, only a subset of independent variables X_i will have a linear relationship with Y
Need to figure out which X_i contributes most information to predict Y
It is in many cases, the first pass prediction algorithm for continuous outcomes.

Linear Regression is a method to model the relationship between a set of independent variables X (also knowns as explanatory variables, features, predictors) and a dependent variable Y. This method assumes the relationship between each predictor X is linearly related to the dependent variable Y.

Independence means that the residuals are not correlated -- the residual from one prediction has no effect on the residual from another prediction. Correlated errors are common in time series analysis and spatial analyses.

size of the training feature set is (2000, 9)
size of the test feature set is (501, 9)
size of the training Target set is (2000,)
size of the test Target set is (501,)
The train root mean squarred error is : 236.63863445055156
The test root mean squarred error is : 399.2661053605523

RMSE of the test data is closer to the training RMSE (and lower) if you have a well trained model. It will be higher if we have an over fitted model.

The R squared metric is : 0.10545

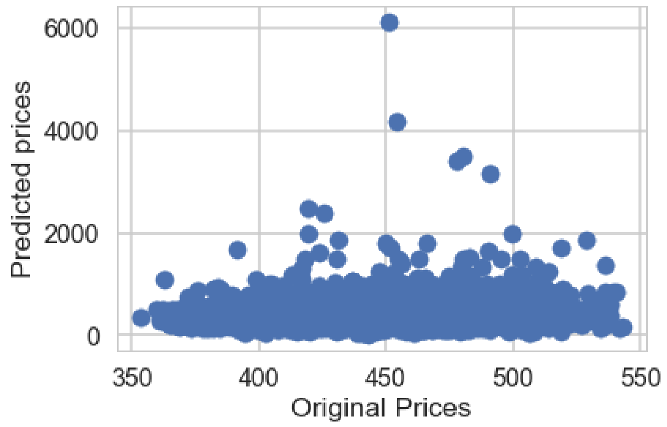
'''The R^2 in scikit learn is the coefficient of determination. It is 1 - residual sum of square / total sum of squares. Since R^2 = 1 - RSS/TSS, the only case where RSS/TSS > 1 happens when our model is even worse than the worst model assumed (which is the absolute mean model).
cv_score is : [-106960.71 -29878.56 -67051.6   -70738.7   -48806.29 -39731.71 -118698.09 -72105.09 -34297.42 -182435.55]
cv_score is : -77070.37197952245
The **cross validation** root mean squarred error is : **277.6155110571498**

Fitting Linear Regression using statsmodels

Statsmodels is a great Python library for a lot of basic and **inferential statistics**. It also provides basic regression functions using an R-like syntax, so it's commonly used by statisticians. The version of least-squares we will use in statsmodels is called ordinary least-squares (OLS). There are many   other versions of least-squares such as partial least squares (PLS) and weighted least squares (WLS).

OLS Regression Results

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.026
Model:                            OLS   Adj. R-squared:                  0.025
Method:                 Least Squares   F-statistic:                     16.97
Date:                Fri, 01 Feb 2019   Prob (F-statistic):           9.82e-14
Time:                        03:57:19   Log-Likelihood:                -542.85
No. Observations:                2501   AIC:                             1096.
Df Residuals:                    2496   BIC:                             1125.
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      1.0463      0.054     19.536      0.000       0.941       1.151
LoanInMonth    0.0006      0.000      1.602      0.109      -0.000       0.001
RATE          -0.1074      0.015     -7.335      0.000      -0.136      -0.079
Home           0.0088      0.007      1.260      0.208      -0.005       0.023
CLTV           0.0045      0.002      1.913      0.056      -0.000       0.009
==============================================================================
Omnibus:                     1214.830   Durbin-Watson:                   1.884
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             4810.993
Skew:                          -2.509   Prob(JB):                         0.00
Kurtosis:                       7.581   Cond. No.                         291.
==============================================================================
```

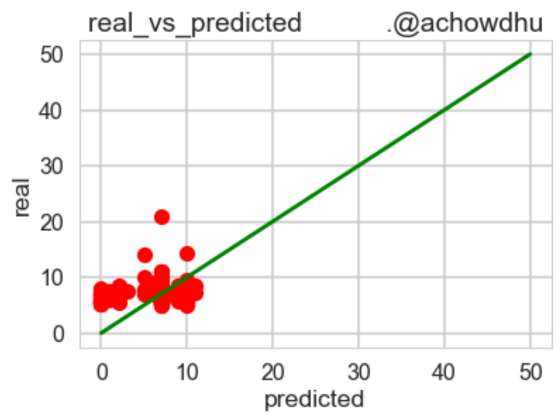


Fitting Linear Regression using sklearn

np.mean((lm.predict(X) - y) ** 2))
Mean squared error (Fix_True Rate): 0.09
Mean squared error (loan_type_code): 0.80
Mean squared error (Loan Amount): 75543.58
Mean squared error (unit_type_code): 7.76

''' Let's try Linear Regression: with Scale'''

Linear Regression - Root Mean Square Error: 2.8077294021457067



Lasso Regression

plot_real_vs_predicted(y_test,y_lasso_predict)

Lasso model - Root Mean Square Error: 2.739546

''' Let's try **Ridge Regression**: '''

''' Now let's try to do **regression via Elastic Net**. '''

''' Now let's try to do regression via **Stochastic Gradient Descent**. '''

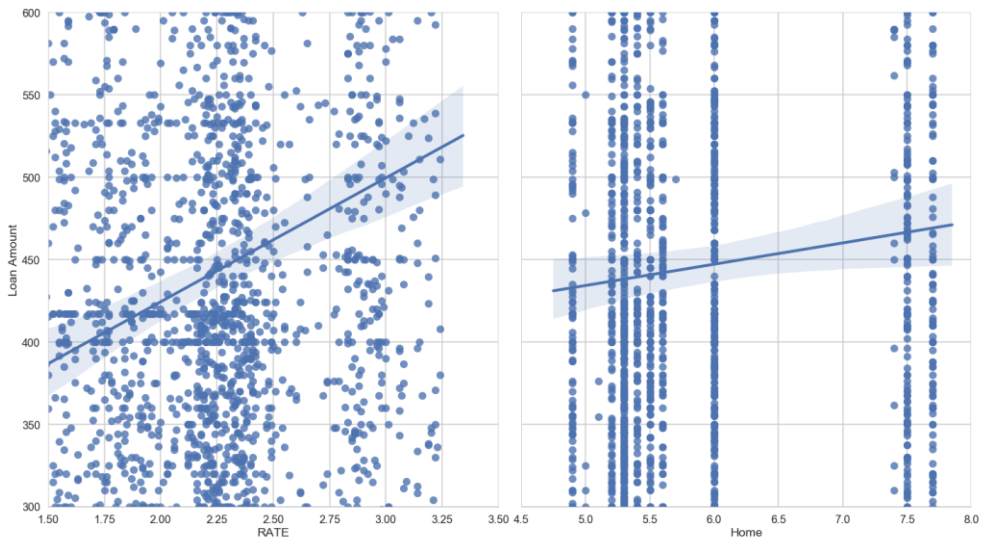
Ridge Regression - Root Mean Square Error: 2.80
Elastic Net model: 0.446 * X2 + 0.414 * X0 + 0.105 * X3 + -0.051 * X1 + -0.0 * X4 + -0.0 * X5 + 0.0 * X6 + -0.0 * X7 + 0.0 * X8
Elastic Net - Root Mean Square Error: 2.74

Stochastic Gradient Descent model: 0.565 * X0 + 0.541 * X2 + 0.25 * X3 + -0.144 * X1 + -0.044 * X7 + -0.034 * X5 + 0.012 * X8 + 0.012 * X6 + 0.003 * X4
Stochastic Gradient Descent - Root Mean Square Error: 2.77

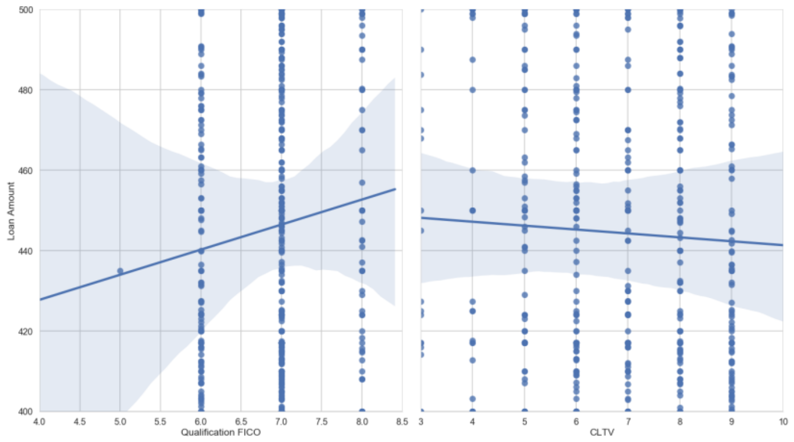
Lasso and Elastic Net Regression seems to be better performing.

Now we have a pandas DataFrame called model_data containing all the data we want to use to predict Mortgage Loan prices (“Loan Amount”).

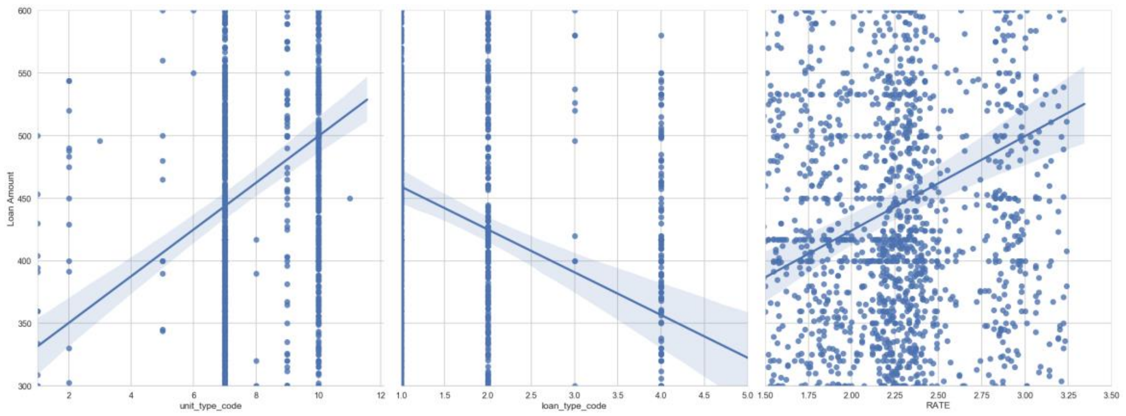
Let's create a variable called ‘Loan Amount’ which will contain the prices. This information is contained



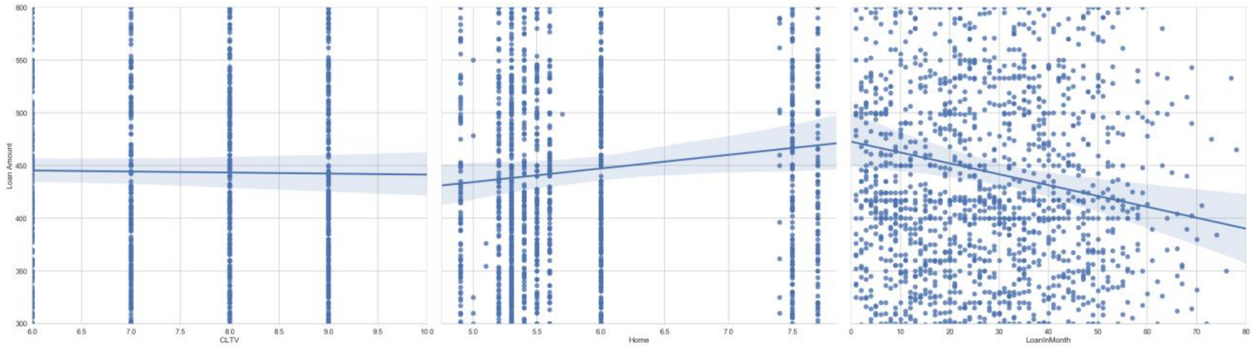
US10Y goes up loan amount slightly increases (Loan Amounts in 1000s)
Home Supply increases, loan amount also increases (Loan Amounts in 1000s)
Interest Rate Chage has bigger impact on Loan Amount compare to Home Supply



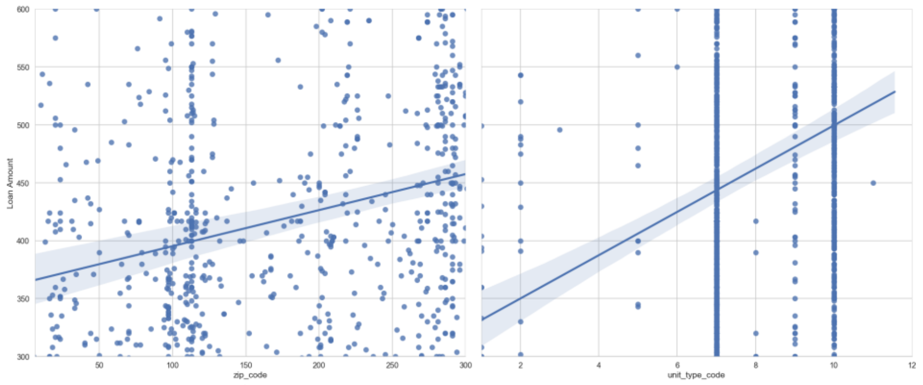
Majority of the FICO scores between 600 and 820 (Loan Amounts in 1000s)
Majority of the CLTV scores between 30 and 100 (Loan Amounts in 1000s)
FICO goes up, Loan Amount goes up
CLTV goes up, Loan Amount Goes down



As num of unit decreases ave Loan Amount also decreases
loan_type_code: Conventional is high volume but Slightly low average loan amount
Two Family (Code = 10) has higher Loan Amount that three Family (Code = 9)
Conventional Mortgage has Higher Loan Amount FHA
Loan Amount increases with 10 Years Treasury Rate.



CLTV does not have much impact on Loan Amount
As number of loans per month increases, loan amount decreases
As Home Supply increases, loan amount also increases



NYC NJ (zip_code 1st digit starting with 7 & 11) seems to be closing more loans and generation more revenues for the Mortgage Bank. City shows similar results. Population density plays bigger role on Loan Amount. As number of loans per month increases, loan amount decreases

Logistic Regression

Logistic Regression Accuracy: 0.6906187624750499

Centering, Scaling and Logistic Regression

Logistic Regression score for training set: 0.707000 or 70%

```
          precision    recall  f1-score   support

        0      0.33      0.10      0.15        20
        1      0.68      0.99      0.80       339
        2      0.50      0.01      0.02        96
        3      0.00      0.00      0.00         5
        4      0.00      0.00      0.00        41

avg / total      0.57      0.67      0.55       501
```

Out of the box, this logistic regression performs better than K-NN (with or without scaling). Let's now scale our data and perform logistic regression:

Scaled Logistic Regression score for test set: 0.682635 or 68%

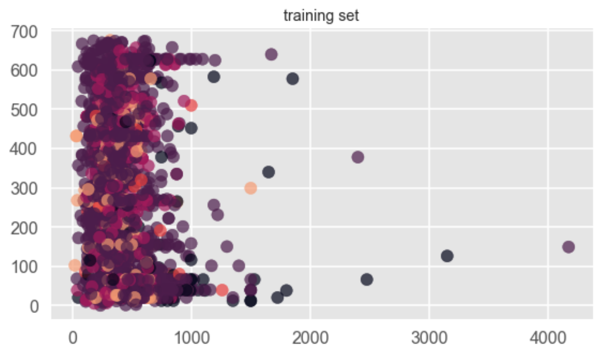
```
          precision    recall  f1-score   support

        0      0.00      0.00      0.00        30
        1      0.68      1.00      0.81       338
        2      0.80      0.05      0.09        82
        3      0.00      0.00      0.00         7
        4      0.00      0.00      0.00        44

avg / total      0.59      0.68      0.56       501
```

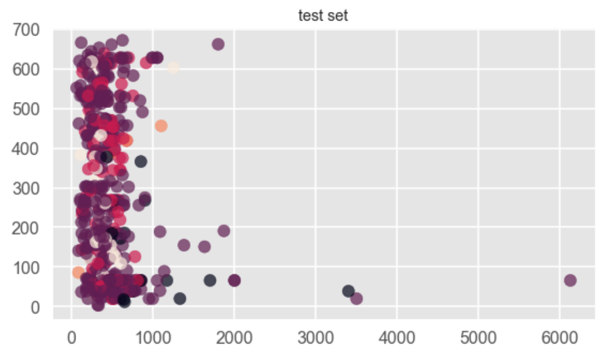
This is very interesting! The performance of **logistic regression did not improve** with data scaling. Why not, particularly when we saw that **k-Nearest Neigbours** performance **improved** substantially with scaling? The reason is that, if there predictor variables with large ranges that do not effect the target variable, a regression algorithm will make the corresponding coefficients ai small so that they do not effect predictions so much. K-nearest neighbours does not have such an inbuilt strategy and so we very much needed to scale the data.'''

Scaling Synthesized Data

Scaling numerical data (that is, multiplying all instances of a variable by a constant in order to change that variable's range) has two related purposes: i) if your measurements are in different currencies and, then, if we both scale our data, they end up being the same & ii) if two variables have vastly different ranges, the one with the **larger range may dominate your predictive model**, even though it may be less important to our target variable than the variable with the smaller range. What we saw is that this problem identified in ii) occurs with **k-NN**, which explicitly looks at **how close data are to one another** but **not in logistic regression** which, when being trained, will shrink the relevant coefficient to account for the lack of scaling. We can see was how the models performed before and after scaling.

Let's now split into testing & training sets & plot both sets:



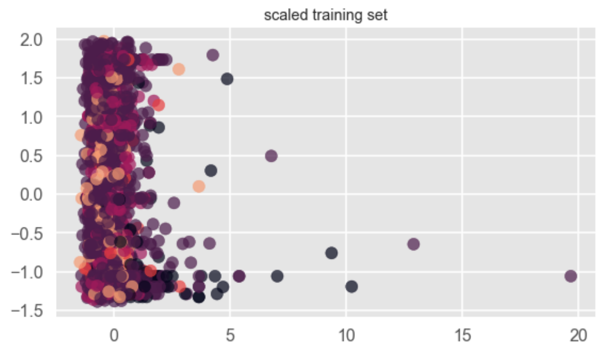


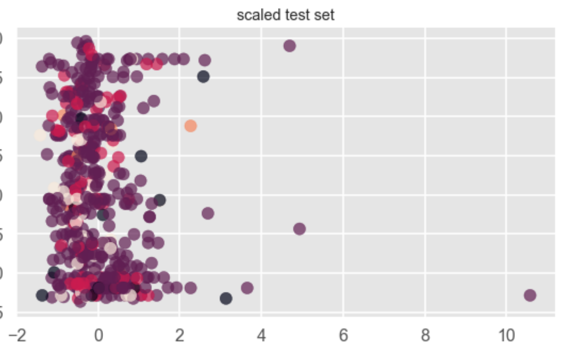
Looking good! Now let's instantiate a k-Nearest Neighbors voting classifier & train it on our training set

k-NN score for test set: 0.646707 or 65%
k-NN score for training set: 0.734000 or 73%

```
       precision   recall  f1-score   support

  0      0.27      0.20      0.23       20
  1      0.69      0.90      0.78      339
  2      0.42      0.16      0.23       96
  3      0.00      0.00      0.00        5
  4      0.00      0.00      0.00       41

avg / total     0.56      0.65      0.58      501
```
We can notice the improvement for KNN compare to Logistic Regression

Now with scaling KNN: I'll now scale the predictor variables and then use k-NN again:





k-NN score for test set: 0.698603 or 70%

It doesn't perform any better with scaling! This is most likely because both features were already around the same range. It really makes sense to scale when variables have widely varying ranges. To see this in action, we're going to add another feature. Moreover, this feature will bear no relevance to the target variable: it will be mere noise.

Adding Gaussian noise to the signal:

We add a third variable of Gaussian noise with mean 0 and variable standard deviation σ. We'll call σ the strength of the noise and we'll see that the stronger the noise, the worse the performance of k-Nearest Neighbours

k-NN score for test set: 0.275000 or 28%

This is a **horrible model**! How about we **scale** and check out performance?

knn_accuracy for test set: 0.93 or 93%
KNN : - Output of Real Data : Conv=5, FHA=4, Res=3, Comm=2: [3 3 1 2 3 1 1 2 0 3 2 3 3 1 3 2 2 1 0 0]
KNN : - Output of prediction: Conv=5, FHA=4, Res=3, Comm=2: [3 3 1 2 3 1 1 2 0 3 2 3 3 1 3 2 2 1 0 2]

With Scale and Synthesize the data we can see huge improvement on the model accuracy.. ***28% to 93%***
Let's do same for Logistic Regression and check out the performance.

logistic regression score for test set: 0.925000 or 92.5%
We can see big improvement. We have seen the essential place in the data scientific pipeline by preprocessing, in its scaling and centering incarnation, and we have done so to promote a holistic approach to minimize the challenges of machine learning.

Random Forests (RF)

Random forests is a supervised ***learning algorithm***. It can be used both for classification and regression. It is also the most flexible and easy to use algorithm. A forest is comprised of trees. It is said that the more trees it has, the more robust a forest is. Random forests creates decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting. It also provides a pretty good indicator of the feature importance. Random forests has a variety of applications, such as recommendation engines, image classification and feature selection. It can be used to classify loyal loan applicants, identify fraudulent activity and predict diseases.

```
Unscalled: Random Forest Classifier Accuracy :  0.7844311377245509
Scalled: Random Forest Classifier Accuracy   :  0.8023952095808383
RFC Confussion Matrix:
[[  7  19   0   0   0]
 [  0 348  12   0   2]
 [  0  37  33   0   1]
 [  1   5   0   0   0]
 [  0  19   3   0  14]]
```

'''With Random Forest Classifier we have height Accuracy of 78%, with scaling (Loan Types : Conv=5, FHA=4, Res=3, Comm=2:) prediction accuracy has ***improved to 80%*** '''

Let's analyze Interest Rate types using Random Forest Classifier. We have achieved greater accuracy compare to Loan Types. We have Fixed Rate mortgage where Fix_True = 1 & Fix_True = 0 for ARM (Adjustable Rate Mortgage)

RandomForestClassifier : Output of X[975:995]: Fix_True =1 & ARM =0: [1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0]

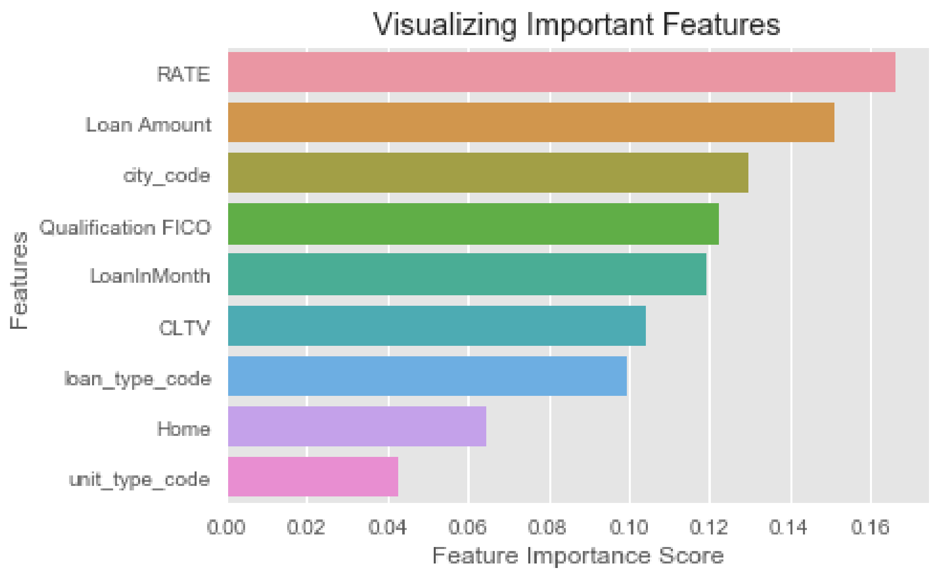
RandomForestClassifier : Output of prediction: Fix_True =1 & ARM =0: [1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0]
Random Forest Classifier Accuracy : 0.8862275449101796 or 89%
RFC Confussion Matrix:
[[13 44]
[13 431]]

Finding Important Features in Scikit-learn

Here, we are finding important features or selecting features in the Mortgage Loan dataset. In scikit-learn, we can perform this task in the following steps:

- First, we need to create a random forests model.
- Second, use the feature importance variable to see feature importance scores.
- Third, visualize these scores using the seaborn library.'''

'''We can also visualize the feature importance. Visualizations are easy to understand and interpretable. For visualization, we have used a combination of matplotlib and seaborn. Higher the value, greater the feature importance'''



Support Vector Regression (SVR)

SUPPORT VECTOR REGRESSION. Those who are in Machine Learning or Data Science are quite familiar with the term SVM or Support Vector Machine. But SVR is a bit different from SVM. As the name suggest the SVR is an regression algorithm, so we can use SVR for working with continuous Values instead of Classification which is SVM.

SVM-SVR accuracy: -0.0026638040886672876 which is unacceptable for our Mortgage Loan Data Sets

Support Vector Machine (SVM)

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. In two dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side.

Support Vector Classifier Accuracy : 0.6447105788423154 or 64%
SVC Confussion Matrix:

```
[[  2  17   1   0   1]
 [  9 290  32   2  22]
 [  2  53  24   0   3]
 [  0   1   0   1   0]
 [  1  31   3   0   6]]
```

Tunning Parameter for SVM

```
from sklearn.grid_search import GridSearchCV
svc = svm.SVC()
 ….
param_grid = {
        "kernel" : ['linear', 'rbf', 'sigmoid'],
        "gamma" : [.1, 1, 10],
        "C" : [1, 5, 10]}

print('Support Vector Classifier Accuracy : ', svm_accuracy)
```

Support Vector Classifier Accuracy : 0.6866267465069861 or 69%

Because of tuning the parameter SVM accuracy has ***improved 8%***

k-Nearest Neighbors (KNN)

k-Nearest Neighbors: FIT

Having explored the Congressional mortgage records dataset, we have build our classifier. Here, we will fit a k-Nearest Neighbors classifier to the mortgage dataset. The features need to be in an array where each column is a feature and each row a different observation or data point. The target needs to be a single column with the same number of observations as the feature data. Notice we named the feature array X and response variable y: This is in accordance with the common scikit-learn practice.
We need create an instance of a k-NN classifier with 6 neighbors (by specifying the n_neighbors parameter) and then fit it to the data.

k-Nearest Neighbors: Predict

Having fit a k-NN classifier, we can use it to predict the label of a new data point. However, there is no unlabeled data available since all of it was used to fit the model! We will use your classifier to predict the label for this new data point, as well as on the training data X that the model has already seen.

size of the training feature set is (2000, 9)
size of the test feature set is (501, 9)
size of the training Target set is (2000,)
size of the test Target set is (751,)

Mean of Unscaled Features: 114.7400015416056
Standard Deviation of Unscaled Features: 200.86870126441687
Mean of Scaled Features: -4.6087893474154617e-17
Standard Deviation of Scaled Features: 1.0

KNeighborsClassifier Accuracy for Loan Types: 0.6826347305389222 or 68%
KNN : - Output of Real Data : Conv=1, FHA=2, Res=4, Comm=0: [1 0 0 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 0 2]
KNN : - Output of prediction: Conv=1, FHA=2, Res=4, Comm=0: [1 1 0 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 2]

Preprocessing: scaling

Here below I (i) scale the data, (ii) use k-Nearest Neighbors and (iii) check the model performance. I'll use scikit-learn's scale function, which standardizes all features (columns) in the array passed to it.'''

k-NN score for test set: 0.678643
k-NN score for training set: 0.770000 or 77%

All these measures improved by 0.1325, which is a **13.25% improvement and significant**!

Decision Tree Classifier

Using Scikit-learn, optimization of decision tree classifier performed by only pre-pruning. Maximum depth of the tree can be used as a control variable for pre-pruning. In the following the example, we can plot a decision tree on the same data with max_depth=4. Other than pre-pruning parameters, We have also tried other attribute selection measure such as entropy

This pruned model is less complex, explainable, and easy to understand than the previous decision tree model plot.

Decision Tree Algorithm
A decision tree is a flowchart-like tree structure where an internal node represents feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome. The topmost node in a decision tree is known as the root node. It learns to partition on the basis of the attribute value. It partitions the tree in recursively manner call recursive partitioning. This flowchart-like structure helps you in decision making. It's visualization like a flowchart diagram which easily mimics the human level thinking. That is why decision trees are easy to understand and interpret.

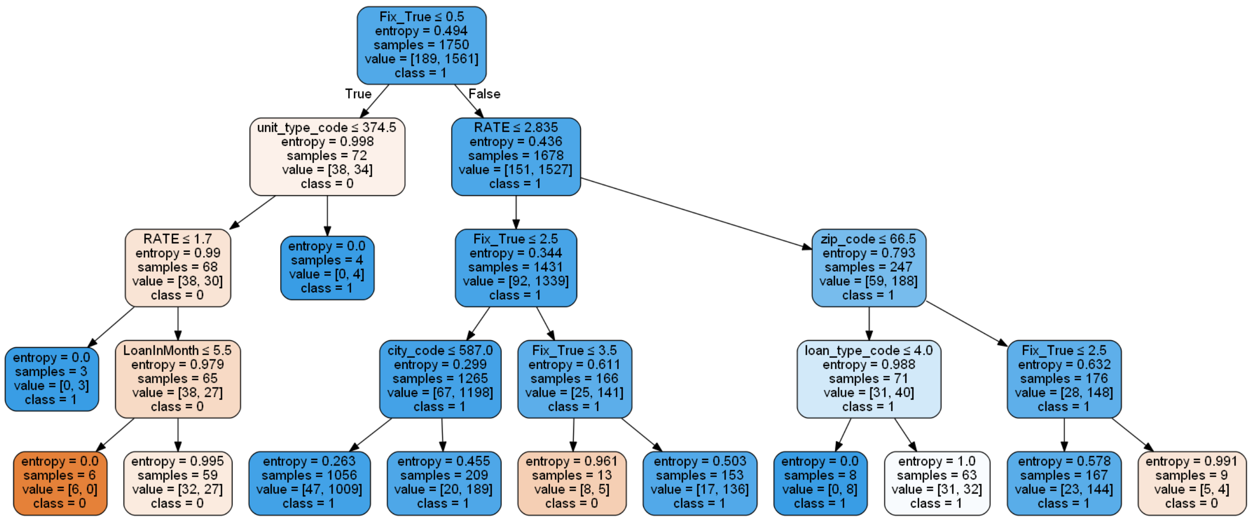
print('Decision Tree Classifier : - Input of Real Data : Fix_True =1 & ARM =0: ', X[975:995,])
print('Decision Tree Classifier : - Output of X[975:995]: Fix_True =1 & ARM =0:' , y[975:995])

print('Decision Tree Classifier : - Output of prediction: Fix_True =1 & ARM =0:', y_pred_dt_out)

Decision Tree Classifier Accuracy : 0.833555259653795 or 83.4%
Decision Tree Classifier: - Output of X[975:995]: Fix_True =1 & ARM =0: [1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0]
Decision Tree Classifier : - Output of prediction: Fix_True =1 & ARM =0: [1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1]

Visualizing Decision Trees

We have used Scikit-learn's export_graphviz function for display the tree within a Jupyter notebook. For plotting tree, you also need to install graphviz and pydotplus. export_graphviz function converts decision tree classifier into dot file and pydotplus convert this dot file to png or displayable form on Jupyter Notebook.



The most important features of our model are as follows:

- **The financial institution's interest rate**
- **Changes in the financial institution's interest rates**
- **The amount of mortgage applications on the previous day**

By analyzing the results of our model, we can see that in particular the 'outliers' (i.e. the days with an extremely high amount of mortgage applications) are consistently under-predicted. These outliers are often influenced by changes in mortgage interest rates, which implies that there is still room for Improvement in our model. As the changes in interest rate is one of the most important features of our model and are influenced by many factors, and hence hard to predict, a dynamic dashboard is proposed. In this dashboard, interest rate changes can be entered manually, so that their impact on the amount of mortgage applications is shown real-time.

A predictive model was created using the SVM, KNN, Random Forest, and Deaccession Tree technique, which predicts the amount of mortgage applications per day(LoanInMonth) with a mean absolute error of mortgage applications per day. This can directly be converted to the amount of personnel needed at the mortgage application department of the financial institutions, by dividing it by the amount of mortgage applications handled per person per day. Based on mortgage per months, company can manage core human resources, such as Loan Processors, Mortgage Loan Originators, Underwriters, secondary market analyst, lock desk personal, compliance personals, & others.

The mortgage interest rates have the biggest impact on our model, but are difficult to predict.
Several features (**10 Years US Treasury Rate, Home Supply Index**) can be added to the model in order to improve its predictive power. Furthermore, there is still improvement in the feature regarding mortgage interest rate changes, as a significant part of the error of our model is caused by under-prediction of the outliers.

7.3 MODEL VALIDATION

Since our models are trained on historical data, we are not sure how these models will perform during the forecasting. In general, the explanatory power of the model on historical data is almost always higher than then predictive power of the model on new data. If this difference is too big, we talk about over fitting. This happens when the model is too complex and starts capturing noise as well. In order to prevent over fitting and accurately estimate the predictive power of our model, we use repeated 10-fold cross-validation. 10-fold cross-validation is one of the most widely used methods for estimating the prediction error. Using 10-fold cross-validation, we are able to split the dataset in out-of-sample data and in-sample data. The dataset is randomly split in 10 equally sized subsets, and the model is run 10 times where each of the 10 subsets is used as the validation set once. The other 9 sets are used as training sets. By changing the validation set at every run, every data point has been used for validation exactly once. After 10 observations, the results of the observations are averaged. This entire process is repeated five times, in order to account for outliers affecting the cross-validation outcome. Afterwards, the performance of the five repeats is averaged to calculate the final performance of the models. Using this method, we can get a feeling of how the model will perform on ‘unknown’ data. This is still not a guarantee that the model will perform the same on live data, but it will give a decent approximation.

8 EVALUATION

In the Evaluation stage, the models are evaluated using several evaluation criteria, and the results of the models are discussed. A final selection of the best model is made, using an independent t-test to check if the best model performs significantly better than the other models. Also the most important features for our models are selected.

8.1 MODEL EVALUATION

In order to measure the model’s performance we use four different metrics: Root Mean Square Error (RMSE), Mean Absolute Error (MAE), MAE/Mean and R2. These are some of the most common metrics used for the evaluation of regression models. Each of these metrics use the residuals (i.e. the differences between the observed values and the predicted values) in order to measure the performance of the model.

RMSE uses a squared value of the absolute error, hence giving more emphasis on higher residuals. This metric is particularly useful because of this last characteristic. One of the key elements of the model is that it should be able to predict 'outliers', i.e. the non-standard days, with an unusually high amount of mortgage applications. RMSE takes this into account by giving more

8.2 DISCUSSION OF RESULTS

Using these metrics to evaluate the performance of the models, we can already see differences between the models. We can see that the Random Forest model performed best, not only in terms of RMSE, but also in MAE, MAE/Mean and R2. SVM slightly worse model. We can also see that the Decisions Tree model performs better, which was expected. This is a basic decision tree, whereas RF ensemble methods of multiple decision trees. In order to test whether the difference in RMSE between RF are statistically significant, an independent t-test can be conducted ($p < 0.05$). For the t-test, the out-of-fold performances (i.e. the predictions on the 10 different folds that were used as validation set in our cross-validation) of the five repeats of the RF and KNN models are compared using the . Using a 95% significance level, there appears to be a significant difference between the average RMSE of the RF and Decision Tree models.

We have also compared Logistic Regression and KNN. We can see big improvement on model accuracy rate with centering scaling and data synthesizing. We have seen the essential place in the data scientific pipeline by preprocessing, in its scaling and centering incarnation, and we have done so to promote a holistic approach to minimize the challenges of machine learning.

We have also compared five different types of Linear Regression model. LASSO Regression and Elastic Net Regression seems to be top performers.

NYC NJ (zip_code 1st digit between 7 & 11) seems to be closing more loans and generation more revenues for the Mortgage Bank. City shows similar results. Population density plays bigger role on Loan Amount.

Loan Amount increases with 10 Years Treasury Rate and Home Supply. As number of loans per month increases, loan amount decreases. As interest Rate goes up, monthly number of loans goes down but average Loan Amount goes up.

Majority of Loans Amount ranges between $400000 to $600000. Majority of the CLTV scores between 30 and 100 (Loan Amounts in 1000s). CLTV goes up, Loan Amount slightly goes down, but CLTV does not have much impact on the Loan Amount.

Majority of the FICO scores between 600 and 820 (Loan Amounts in 1000s). FICO goes up, Loan Amount goes up, which make sense; people with good credit score have more borrowing power. People with the lower credit score prefer FHA Loan Types over Conventional Loan Types. Conventional Loan is the leader among the loan types. Conventional Mortgage has Loan volume compare to FHA; Conventional is high volume but slightly low average loan amount

As number of unit decreases average Loan Amount also decreases. One and Two family houses seem to be more popular among the borrowers. Two Family (Code = 10) has higher Loan Amount that three Family (Code = 9).