-

Advance Data Science

Single-family loan data Mid Term

**Team 2**

Abhinav Tiwari

Dharit Shah

Nilesh Nerkar

SUMMARY

The report summarizes the analysis performed on Single-family loan data, provided by Freddie Mac(<http://www.freddiemac.com/news/finance/sf_loanlevel_dataset.html>). The data provided by Freddie Mac consists of the following details:

*Mortgages originated from January 1, 1999, through the “Origination Cutoff Date”, with monthly loan performance data through the “Performance Cutoff Date,” that were sold to Freddie Mac or back Freddie Mac Participation Certificates (PCs).*

* Fully amortizing 15-, 20-, and 30-year fixed-rate mortgages*

* Mortgages categorized as having verified or waived documentation.*

We then build predictive analytics models using the datasets. The problem presented is divided into 2 section:

**Section 1: Data wrangling**

* Data Download and pre-processing
* Exploratory Data analysis

**Section 2: Building and evaluating models**

* **Prediction** using Linear Regression, Random Forest, Neural Network KNN Algorithms
* **Classification** using Logistic Regression, Random Forest, Neural Network, SVN Algorithms

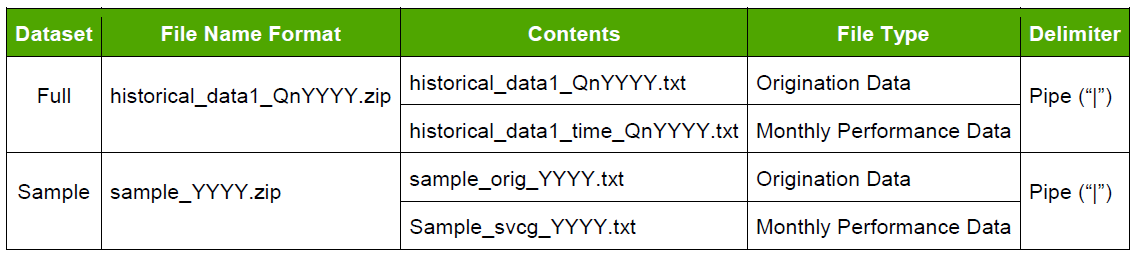
1 PART 1: DATA INGESTION AND WRANGLING

1.1 THE DATA

***Single Family Loan-Level Dataset***

For each calendar quarter, there is one file containing loan **origination data** and one file containing **monthly performance data** for each loan in the **origination data file**.

Freddie Mac has created a smaller dataset for those who may not require, or have the capability, to download the full Dataset. The sample dataset is a simple random sample6 of 50,000 loans selected from each full vintage year and a proportionate number of loans from each partial vintage year of the full Single Family Loan-Level Dataset. Each vintage year has one origination data file and one corresponding monthly performance data file, containing the same loan-level data fields as those included in the full Dataset. Due to the size of the dataset, the data has been broken up and compressed as detailed below. The files are organized chronologically by year and quarter.



**Data Download and pre-processing:**

The very first challenge was to programmatically download the data from Freddie Mac website (<https://freddiemac.embs.com/FLoan/Data/download.php>) and download and preprocess the “SAMPLE” file both for origination and performance data.

To download the file programmatically, first the user should register him/herself by creating username and password. Once logged in, the user can download all the file required for analysis. We have used the python webbot library for this purpose. To store the user credential, we need to store them in the request session so that user didn’t redirect back to the login page whenever he/she required to download a file from the Freddie Mac posted dataset.

Auto Navigate to the download page

Webbot - Web automation library for simple and easy web browser automation and offers extensive features but keeping it simple to use and master

from webbot import Browser  
from zipfile import ZipFile  
import sys  
import os  
import time  
import zipfile  
import datetime  
import glob  
import pandas as pd  
  
# Auto Navigate to the download page  
def navigate(user='tiwari.abhi@husky.neu.edu', password='c`PD{0f8'):  
 global web  
 web = Browser()  
 web.go\_to("https://freddiemac.embs.com/FLoan/secure/login.php?pagename=download3")  
 web.type(user, into='email')  
 web.type(password, into='password')  
 web.click('Submit Credentials')  
 web.click('Yes')

web.click('Continue')

Once, the user is logged in to the website, we will be using request session to drive the further functionality Now we will download from the Freddie Mac Website and download all the “**Sample**” files for our analysis purpose.

def download\_all\_files(styear=2005, edyear=2006):  
 years = list(range(styear, edyear + 1))  
  
 global file\_names  
 file\_names = []  
  
 for year in years:  
 web.click('sample\_{}.zip'.format(str(year)))  
 file\_names.append('sample\_{}.zip'.format(str(year)))  
 print('sample\_{}.zip has been downloaded'.format(str(year)))  
 time.sleep(60) # Some delay must be added since site blocks immediate requests in large numbers  
  
 # Path to where files have been downloaded  
 global download\_path  
 download\_path = ''  
 path\_list = os.getcwd().split('\\')[:3]  
  
 for item in path\_list:  
 download\_path = download\_path + item + '\\'  
  
 download\_path = download\_path + 'Downloads'  
  
 print(download\_path)  
  
  
def assure\_path\_exists(path):  
 if not os.path.exists(path):  
 os.makedirs(path)  
  
  
def extract\_zipped\_files\_to\_cwd(path\_to\_downloaded\_files):  
 global folder\_path\_all\_files  
 folder\_path\_all\_files = os.getcwd() + '\\' + 'Extracted Files'  
 zip\_ref = zipfile.ZipFile(path\_to\_downloaded\_files, 'r')  
 assure\_path\_exists(folder\_path\_all\_files)  
 zip\_ref.extractall(folder\_path\_all\_files)  
 zip\_ref.close()

1.2 Data Preprocessing and Cleaning

Since, these files are big in size and consist of huge amount of data, we need to preprocess these files before getting saved in our drive. These Zip file consist of two files:

**Origination File:**

For origination file, we first analyze the file size for all the year. Using python pandas, we create a data frame where we append all the data from the sample file for all the year. Origination file consist of 26 columns which consist of various details associated with the loan originated in each year.

We have many Null values and spaces (an invalid values) which we need to handle before using these files to compute a summary report. We must make sure that our data is in proper format with same datatype. We have created following functions:

def fillNAN\_orig(df):  
 df['fico'] = df['fico'].fillna(0)  
 df['flag\_fthb'] = df['flag\_fthb'].fillna('X')  
 df['cd\_msa'] = df['cd\_msa'].fillna(0)  
 df['mi\_pct'] = df['mi\_pct'].fillna(0)  
 df['cnt\_units'] = df['cnt\_units'].fillna(0)  
 df['occpy\_sts'] = df['occpy\_sts'].fillna('X')  
 df['cltv'] = df['cltv'].fillna(0)  
 df['dti'] = df['dti'].fillna(0)  
 df['ltv'] = df['ltv'].fillna(0)  
 df['channel'] = df['channel'].fillna('X')  
 df['ppmt\_pnlty'] = df['ppmt\_pnlty'].fillna('X')  
 df['prop\_type'] = df['prop\_type'].fillna('XX')  
 df['zipcode'] = df['zipcode'].fillna(0)  
 df['loan\_purpose'] = df['loan\_purpose'].fillna('X')  
 df['cnt\_borr'] = df['cnt\_borr'].fillna(0)  
 df['flag\_sc'] = df['flag\_sc'].fillna('N')  
 return df  
  
  
def changedatatype\_orig(df):  
 # Change the data types for all column  
 df[['fico', 'cd\_msa', 'mi\_pct', 'cnt\_borr', 'cnt\_units', 'cltv', 'dti', 'orig\_upb', 'ltv', 'zipcode',  
 'orig\_loan\_term']] = df[['fico', 'cd\_msa', 'mi\_pct', 'cnt\_borr', 'cnt\_units', 'cltv', 'dti', 'orig\_upb', 'ltv', 'zipcode','orig\_loan\_term']].astype('int64')  
 df[['flag\_sc', 'servicer\_name']] = df[['flag\_sc', 'servicer\_name']].astype('str')  
 return df

**Performance File:**

For performance file, we first analyze the file size for all the year. Using python pandas, we create a data frame where we append all the data for all the year. Performance file consist of 23 columns which consist of various information about the loan origination in a year. Since, the size of the file is very large, we decide to summarizes the input file for all the year during its preprocessing. Some of the major columns are defined below:

As we have many empty column values in our origination and preprocessing file, we need to clean those to ensure that we don’t have any NAN/NA value in our data. Also, we need to take care of the data type of column. These columns will be required while creating the summary matrices.

def fillNA\_performance(df):  
 df['current\_upb'] = df['current\_upb'].fillna(0)  
 df['delq\_sts'] = df['delq\_sts'].fillna('XX')  
 df['loan\_age'] = df['loan\_age'].fillna(-1)  
 df['mths\_remng'] = df['mths\_remng'].fillna(-1)  
 df['repch\_flag'] = df['repch\_flag'].fillna('X')  
 df['flag\_mod'] = df['flag\_mod'].fillna('N')  
 df['cd\_zero\_bal'] = df['cd\_zero\_bal'].fillna(-1)  
 df['dt\_zero\_bal'] = df['dt\_zero\_bal'].fillna('189901')  
 df['current\_int\_rt'] = df['current\_int\_rt'].fillna(0)  
 df['current\_dfr\_upb'] = df['current\_dfr\_upb'].fillna(0)  
 df['dt\_lst\_pi'] = df['dt\_lst\_pi'].fillna('189901')  
 df['mi\_recoveries'] = df['mi\_recoveries'].fillna(0)  
 df['net\_sale\_proceeds'] = df['net\_sale\_proceeds'].fillna('U')  
 df['non\_mi\_recoveries'] = df['non\_mi\_recoveries'].fillna(0)  
 df['expenses'] = df['expenses'].fillna(0)  
 df['legal\_costs'] = df['legal\_costs'].fillna(0)  
 df['maint\_pres\_costs'] = df['maint\_pres\_costs'].fillna(0)  
 df['taxes\_ins\_costs'] = df['taxes\_ins\_costs'].fillna(0)  
 df['misc\_costs'] = df['misc\_costs'].fillna(0)  
 df['actual\_loss'] = df['actual\_loss'].fillna(0)  
 df['modcost'] = df['modcost'].fillna(0)  
 df['step\_mod\_flag'] = df['step\_mod\_flag'].fillna('X')  
 df['def\_py\_mod'] = df['def\_py\_mod'].fillna('X')  
 df['eltv'] = df['eltv'].fillna(0)  
 return df

def changedatatype\_performance(df):  
 # Change the data types for all column  
 df[['loan\_seq', 'delq\_sts', 'repch\_flag', 'flag\_mod', 'net\_sale\_proceeds', 'step\_mod\_flag', 'def\_py\_mod']] = df[['loan\_seq', 'delq\_sts', 'repch\_flag', 'flag\_mod', 'net\_sale\_proceeds', 'step\_mod\_flag', 'def\_py\_mod']].astype('str')  
 df[['mth\_per', 'loan\_age', 'mths\_remng', 'dt\_zero\_bal']] = df[['mth\_per', 'loan\_age', 'mths\_remng', 'dt\_zero\_bal']].astype('int64')  
 df[['current\_upb', 'cd\_zero\_bal', 'current\_int\_rt', 'current\_dfr\_upb', 'dt\_lst\_pi', 'mi\_recoveries','non\_mi\_recoveries', 'expenses', 'legal\_costs', 'maint\_pres\_costs', 'taxes\_ins\_costs', 'misc\_costs','actual\_loss', 'modcost']] = df[['current\_upb', 'cd\_zero\_bal', 'current\_int\_rt', 'current\_dfr\_upb', 'dt\_lst\_pi', 'mi\_recoveries','non\_mi\_recoveries', 'expenses', 'legal\_costs', 'maint\_pres\_costs', 'taxes\_ins\_costs', 'misc\_costs','actual\_loss', 'modcost']].astype('float32')  
 return df

Once, we are done with the cleaning of the performance file, we will create a summarized version of the file based on certain column which are important for us. For this step, we created a function which will get the Max/Min/Average value of the columns in our summarized performance file.

def get\_current\_upb(group):  
 return {'min\_current\_upb': group.min(), 'max\_current\_upb': group.max()}  
  
  
def get\_delq\_sts(group):  
 return {'min\_delq\_sts': group.min(), 'max\_delq\_sts': group.max()}  
  
  
def get\_cd\_zero\_bal(group):  
 return {'min\_cd\_zero\_bal': group.min(), 'max\_cd\_zero\_bal': group.max()}  
  
  
def get\_mi\_recoveries(group):  
 return {'min\_mi\_recoveries': group.min(), 'max\_mi\_recoveries': group.max()}  
  
  
def get\_net\_sale\_proceeds(group):  
 return {'min\_net\_sale\_proceeds': group.min(), 'max\_net\_sale\_proceeds': group.max()}  
  
  
def get\_non\_mi\_recoveries(group):  
 return {'min\_non\_mi\_recoveries': group.min(), 'max\_non\_mi\_recoveries': group.max()}  
  
  
def get\_expenses(group):  
 return {'min\_expenses': group.min(), 'max\_expenses': group.max()}  
  
  
def get\_legal\_costs(group):  
 return {'min\_legal\_costs': group.min(), 'max\_legal\_costs': group.max()}  
  
  
def get\_maint\_pres\_costs(group):  
 return {'min\_maint\_pres\_costs': group.min(), 'max\_maint\_pres\_costs': group.max()}  
  
  
def get\_taxes\_ins\_costs(group):  
 return {'min\_taxes\_ins\_costs': group.min(), 'max\_taxes\_ins\_costs': group.max()}  
  
  
def get\_misc\_costs(group):  
 return {'min\_misc\_costs': group.min(), 'max\_misc\_costs': group.max()}  
  
  
def get\_actual\_loss(group):  
 return {'min\_actual\_loss': group.min(), 'max\_actual\_loss': group.max()}  
  
  
def get\_modcost(group):  
 return {'min\_modcost': group.min(), 'max\_modcost': group.max()}

1.3 Creating Summarized CSV file (Output)

Once the preprocessing and data cleaning steps are performed, we will have created our final output file, one for origination file named ‘OriginationCombinedCode’ and for performance file named ‘PerformanceCombinedSummary’. These final files will be used for our analysis performed in part 2.

We have also created some derived column like ‘Year’ and ‘Quarter’ which will help to create the summary metrics.

def createPerformanceCombined(sample\_perf\_files):  
 writeHeader2 = True  
 if "sample" in sample\_perf\_files:  
 filename = "Sample Performance Combined Summary.csv"  
  
 abc = glob.glob(sample\_perf\_files)  
  
 with open(filename, 'w', encoding='utf-8', newline="") as file:  
 for f in abc:  
 perf\_df = pd.read\_csv(f, sep='|',  
 names=['loan\_seq', 'mth\_per', 'current\_upb', 'delq\_sts', 'loan\_age', 'mths\_remng',  
 'repch\_flag', 'flag\_mod', 'cd\_zero\_bal', 'dt\_zero\_bal', 'current\_int\_rt',  
 'current\_dfr\_upb', 'dt\_lst\_pi', 'mi\_recoveries', 'net\_sale\_proceeds',  
 'non\_mi\_recoveries', 'expenses', 'legal\_costs', 'maint\_pres\_costs',  
 'taxes\_ins\_costs', 'misc\_costs', 'actual\_loss', 'modcost', 'step\_mod\_flag',  
 'def\_py\_mod', 'eltv'], skipinitialspace=False, low\_memory=False)  
  
 perf\_df = fillNA\_performance(perf\_df)  
 perf\_df = changedatatype\_performance(perf\_df)  
 summ\_df = pd.DataFrame()  
 summ\_df['loan\_seq'] = perf\_df['loan\_seq'].drop\_duplicates()  
 summ\_df = summ\_df.join(  
 (perf\_df['current\_upb'].groupby(perf\_df['loan\_seq']).apply(get\_current\_upb).unstack()), on='loan\_seq')  
 summ\_df = summ\_df.join((perf\_df['delq\_sts'].groupby(perf\_df['loan\_seq']).apply(get\_delq\_sts).unstack()),  
 on='loan\_seq')  
 summ\_df = summ\_df.join(  
 (perf\_df['cd\_zero\_bal'].groupby(perf\_df['loan\_seq']).apply(get\_cd\_zero\_bal).unstack()), on='loan\_seq')  
 summ\_df = summ\_df.join(  
 (perf\_df['non\_mi\_recoveries'].groupby(perf\_df['loan\_seq']).apply(get\_non\_mi\_recoveries).unstack()),  
 on='loan\_seq')  
 summ\_df = summ\_df.join((perf\_df['expenses'].groupby(perf\_df['loan\_seq']).apply(get\_expenses).unstack()),  
 on='loan\_seq')  
 summ\_df = summ\_df.join(  
 (perf\_df['legal\_costs'].groupby(perf\_df['loan\_seq']).apply(get\_legal\_costs).unstack()), on='loan\_seq')  
 summ\_df = summ\_df.join(  
 (perf\_df['maint\_pres\_costs'].groupby(perf\_df['loan\_seq']).apply(get\_maint\_pres\_costs).unstack()),  
 on='loan\_seq')  
 summ\_df = summ\_df.join(  
 (perf\_df['taxes\_ins\_costs'].groupby(perf\_df['loan\_seq']).apply(get\_taxes\_ins\_costs).unstack()),  
 on='loan\_seq')  
 summ\_df = summ\_df.join((perf\_df['misc\_costs'].groupby(perf\_df['loan\_seq']).apply(get\_misc\_costs).unstack()),  
 on='loan\_seq')  
 summ\_df = summ\_df.join(  
 (perf\_df['actual\_loss'].groupby(perf\_df['loan\_seq']).apply(get\_actual\_loss).unstack()), on='loan\_seq')  
 summ\_df = summ\_df.join((perf\_df['modcost'].groupby(perf\_df['loan\_seq']).apply(get\_modcost).unstack()),  
 on='loan\_seq')  
  
 if writeHeader2 is True:  
 summ\_df.to\_csv(file, mode='a', header=True, index=False)  
 writeHeader2 = False  
 else:

summ\_df.to\_csv(file, mode='a', header=False, index=False)

To validate year we have defined function called validate\_year(2005-2017)

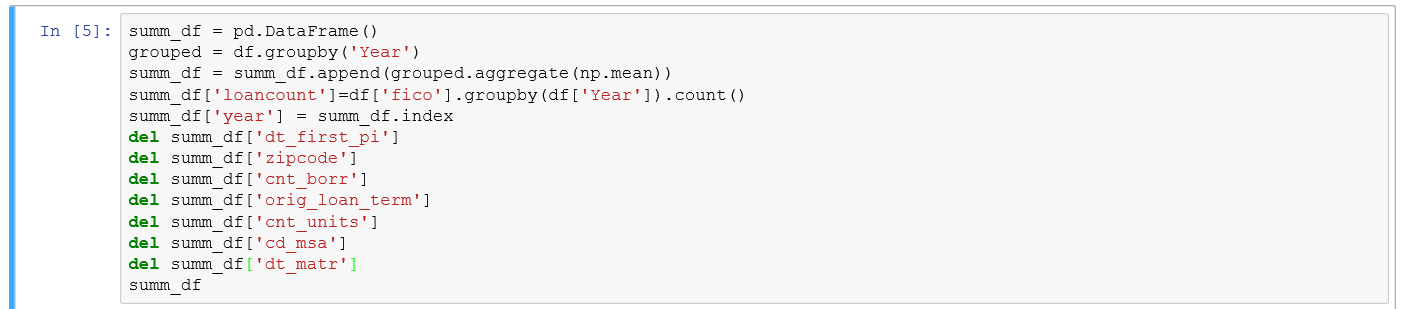
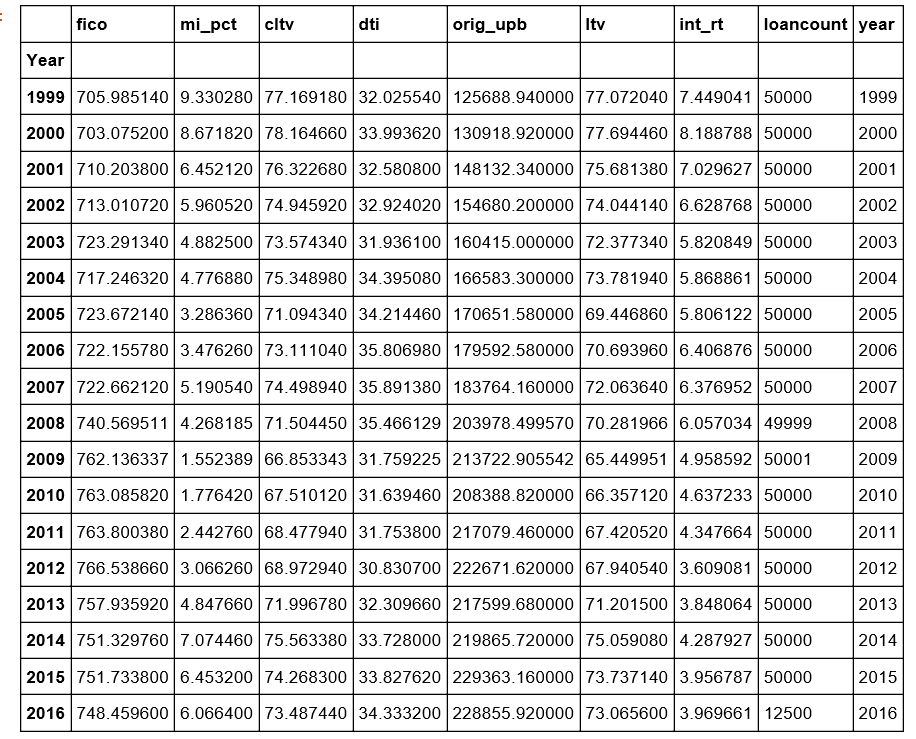
def validate\_year(year):  
 try:  
 year = int(year)  
 except:  
 print('Cannot Parse Year to Integer')  
 exit(0)  
  
 if not year in [2005,2006,2007,2008,2009,2010,2011,2012,2013,2014,2015,2016,2017]:  
 print('Data unavailabale for the requested year')  
 exit(0)  
 else:  
 return year

**NOTE:** As we are working on two different file, we need to know about the data and the relationship between the two files created. In origination file, we have idloan which is a unique loan sequence number with quarter and year of loan origination attached to it. In performance file, for a given year we have multiple rows associated with a loan number which depicts its performance. We don’t have any loan number duplicated in origination file with respect to year. It is always **Unique**.

PART 2: Exploratory Data Analysis

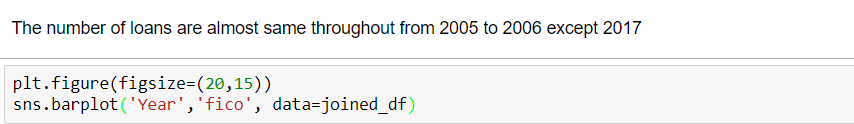
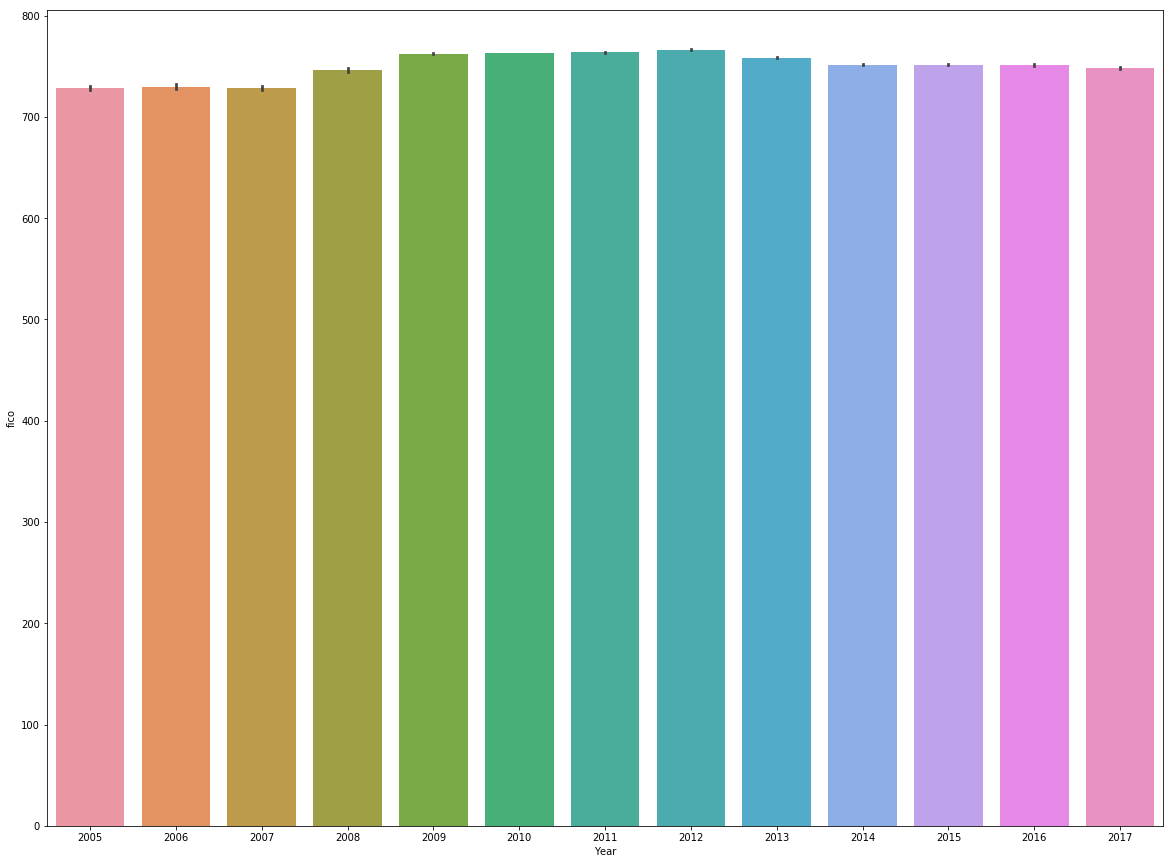
In Part 2, we were asked to write Jupyter notebook using Python to graphically represent different summaries of data and summarize our findings in this notebook.

We first create a pandas’ data frame for the origination file and group the data on the year.



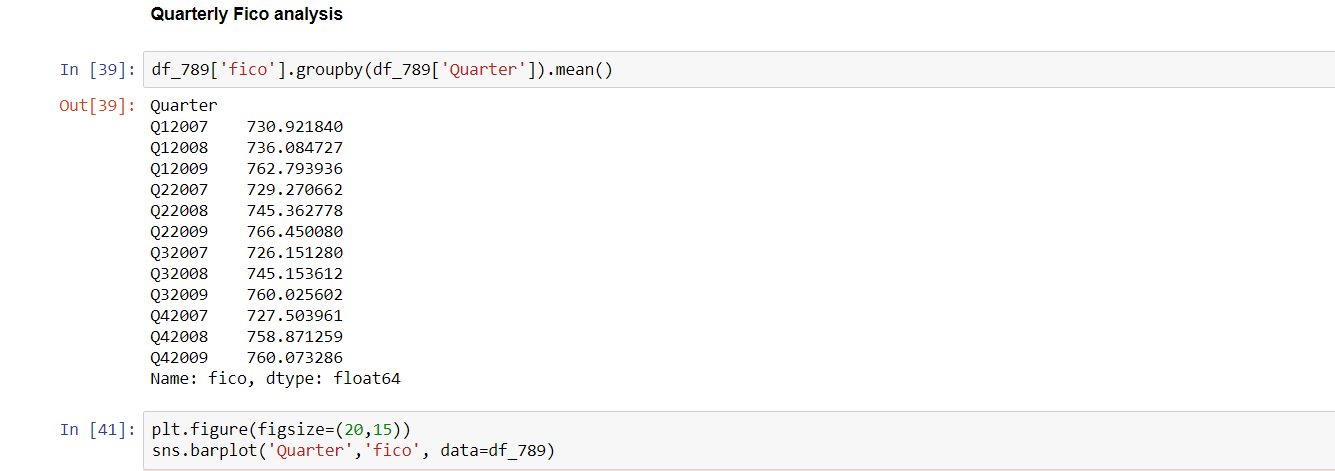
Here we show the various details associated with the origination file and the total loan count and mean of various important factor like fico score, interest rate based on year

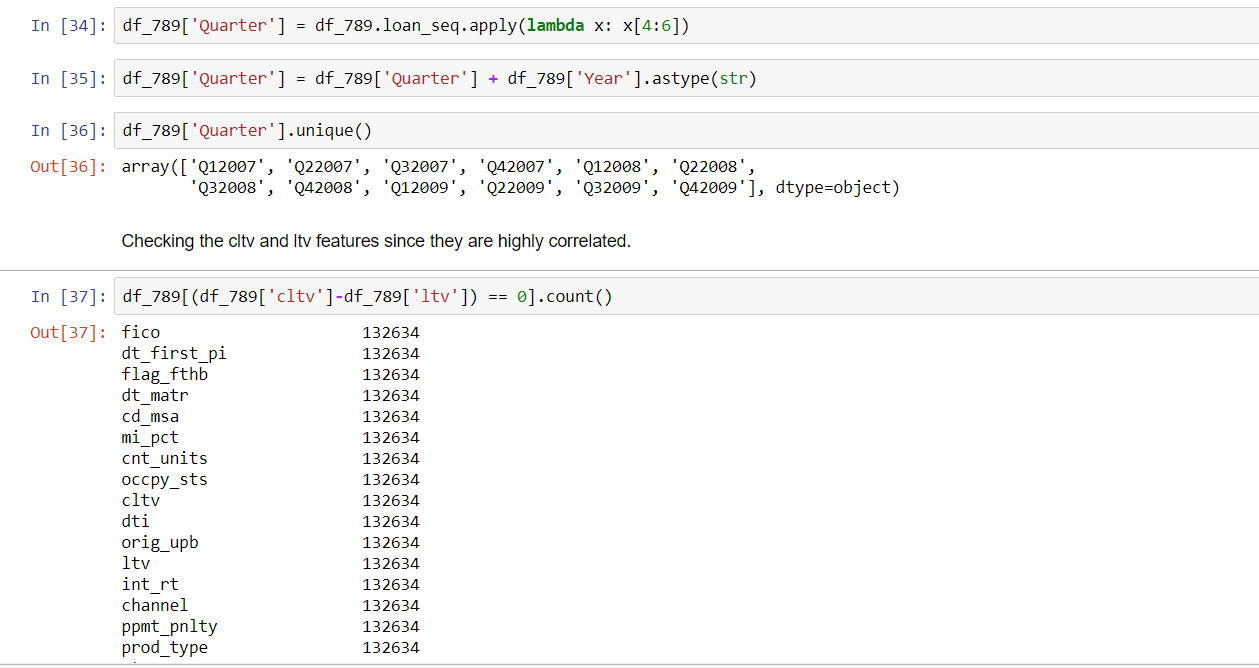
**FICO Based on Count**

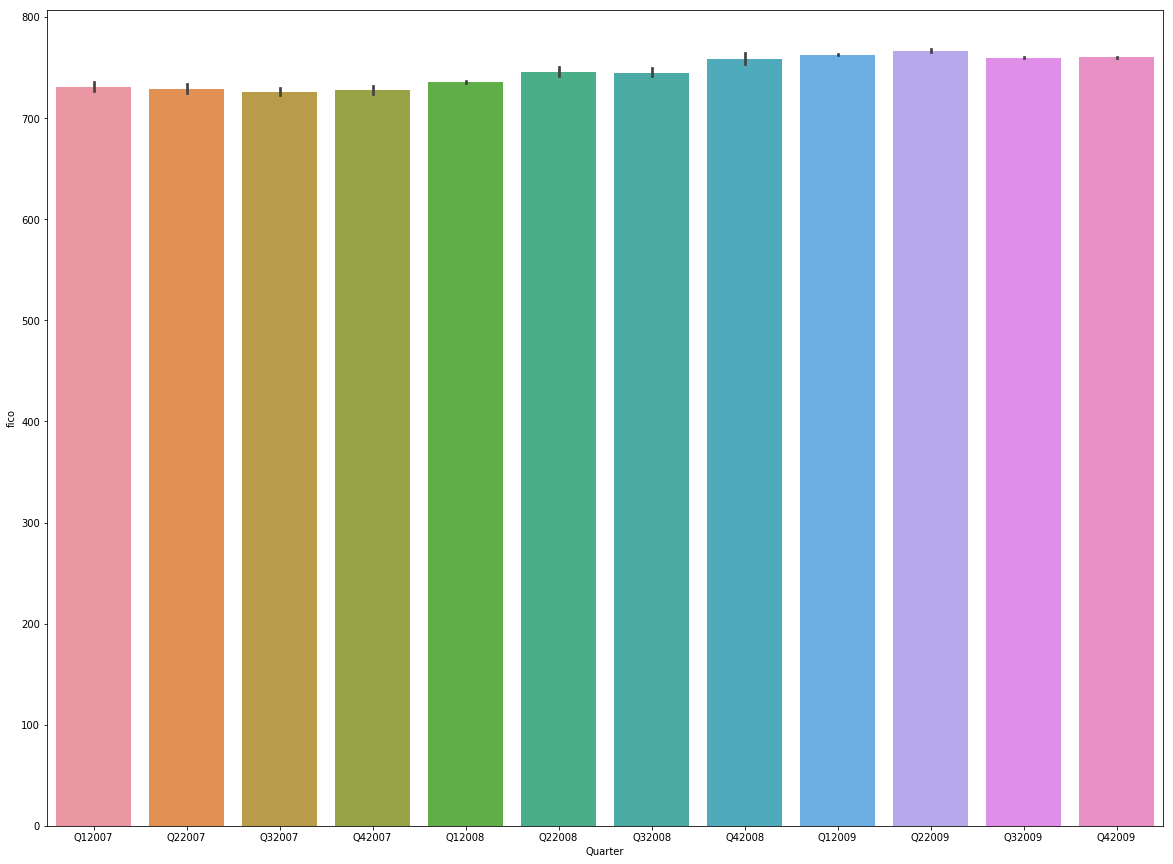
The number of loans are almost same throughout from 2005 to 2006 except 2017

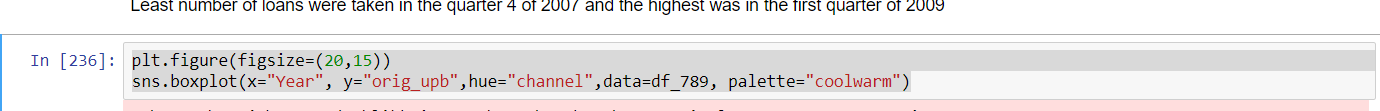
Loan Term and date of maturity are correlated, there are many correlation in the later part which are inter related(expenses, actual loss,legal costs, maintaince costs, taxes and misc costs)

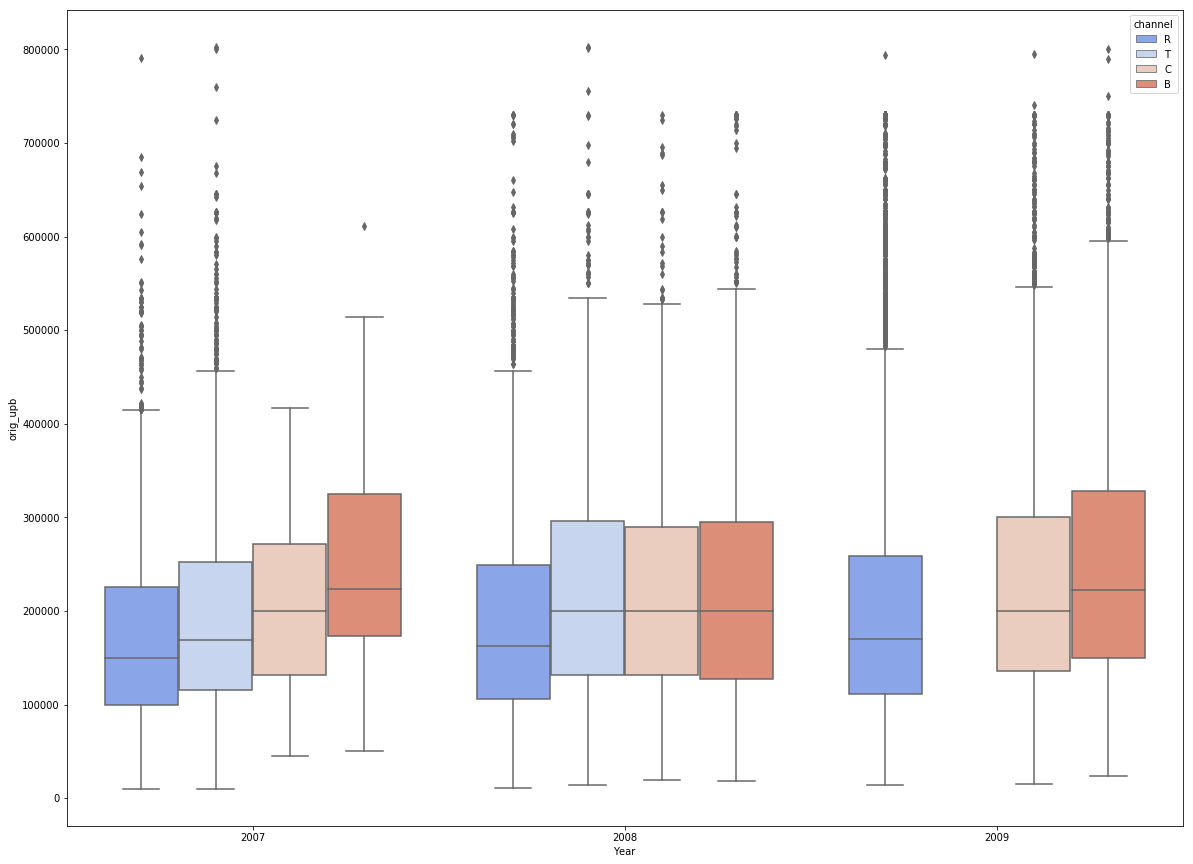
## Exploring the quarterly trends of features











Most of loan involved in the occupancy status investement property had less average UPB(unpaid balance) value



2 PART II: Building & Evaluating Models

**PREDICTION (Predicting Interest Rates)**

Here we are going to create a predictive model based on information from the origination data from the prior quarter using the various regression technique to calculate the following metrics:

**MAE (Mean Absolute Error) -** In [statistics](https://en.wikipedia.org/wiki/Statistics), the mean absolute error (MAE) is a quantity used to measure how close forecasts or predictions are to the eventual outcomes.

**RMSE (Root Mean Square Error) -** The root-mean-square deviation (RMSD) or root-mean-square error (RMSE) is a frequently used measure of the differences between values (sample and population values) predicted by a model or an estimator and the values observed. The RMSD represents the [sample standard deviation](https://en.wikipedia.org/wiki/Sample_standard_deviation) of the differences between predicted values and observed values. These individual differences are called [residuals](https://en.wikipedia.org/wiki/Errors_and_residuals_in_statistics) when the calculations are performed over the data sample that was used for estimation, and are called prediction errors when computed out-of-sample. The RMSD serves to aggregate the magnitudes of the errors in predictions for various times into a single measure of predictive power. RMSD is a good measure of [accuracy](https://en.wikipedia.org/wiki/Accuracy_and_precision), but only to compare forecasting errors of different models for a particular variable and not between variables, as it is scale-dependent

**Median Absolute Error-** The [median\_absolute\_error](http://scikit-learn.org/stable/modules/generated/sklearn.metrics.median_absolute_error.html#sklearn.metrics.median_absolute_error) is particularly interesting because it is robust to outliers. The loss is calculated by taking the median of all absolute differences between the target and the prediction.

2.1 APPROCH - PREDICTION



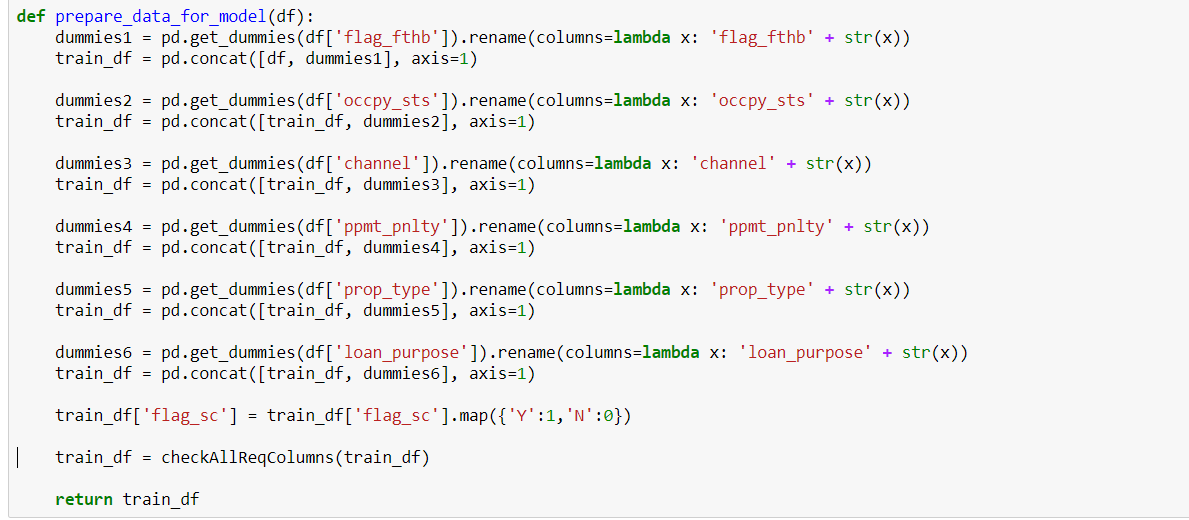
2.1.1 CREATING THE FUNCTION AND DOWNLOAD THE INPUT

We have created functions that would take hit the Freddie Mac website and download all the Historical file for our purpose. Based on the input Quarter, the file will be imported into python Data Frame. It went through following preprocessing before building model.

Handling the missing values:

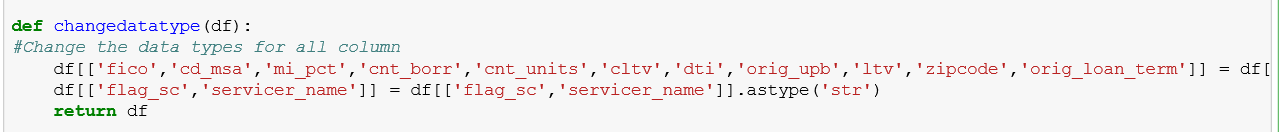


Created dummy columns (1…C):



2.1.2 CONVERSION OF DATA TYPE

Data types of the History Origination file is converted as below.



2.1.3 DIFFERENT MACHINE LEARNING ALGORITHMS AND OUTPUT

2.1.3.1 REGRESSION



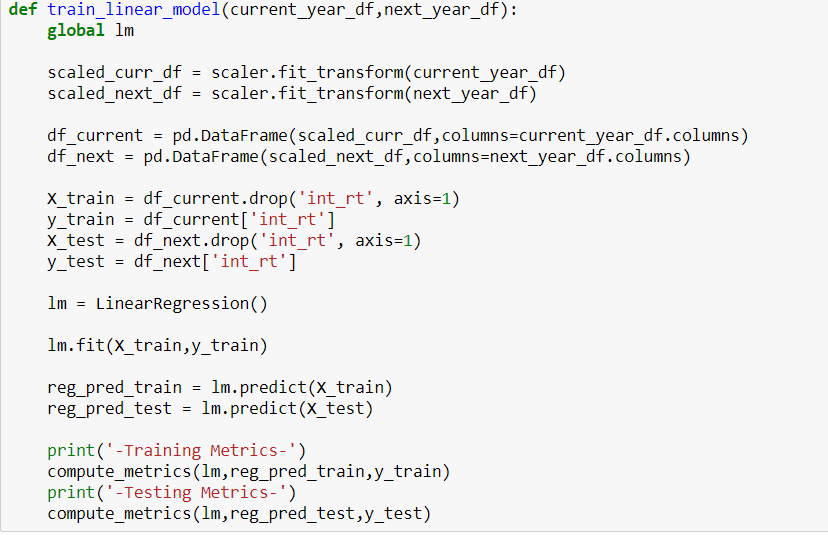
We have used the following regression techniques and compared the R2 and selected Liner Regression.

**Regression using Linear Regression:**

[**sklearn.linear\_model**](http://scikit-learn.org/stable/modules/classes.html#module-sklearn.linear_model)**.LinearRegression :** Ordinary least squares Linear Regression.

**Implementing Liner Regression**

Linear regression is an approach for modeling the relationship between a scalar dependent variable y and one or more explanatory variables.



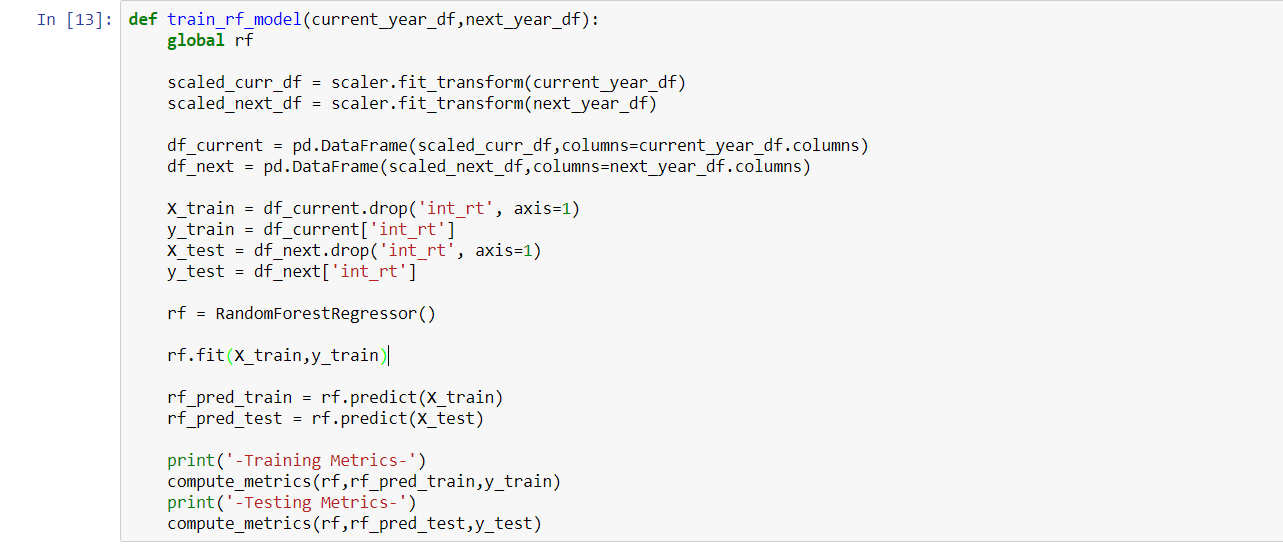
2.1.4.3 RANDOM FOREST



The random forest starts with a standard machine learning technique called a “decision tree”. This is a type of additive model that makes predictions by combining decisions from a sequence of base models.

**sklearn.ensemble.RandomForestRegressor**

A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if bootstrap=True (default).

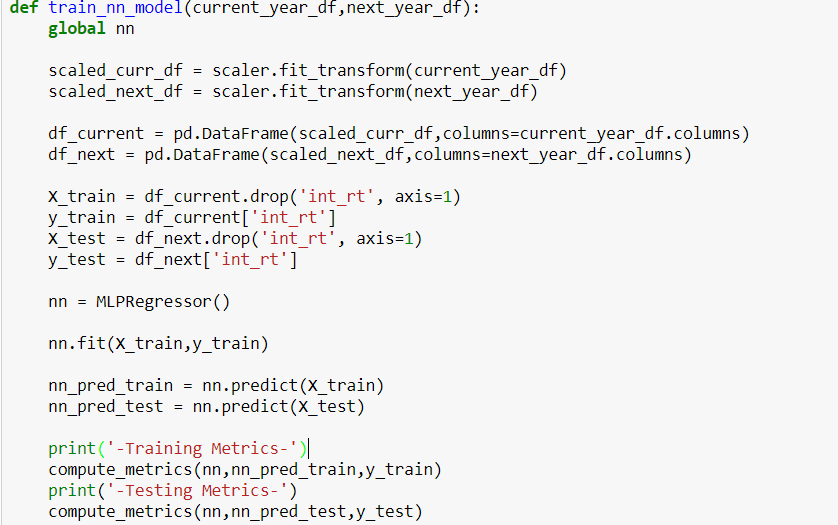


2.1.4.4 NEURAL NETWORK



Neural network terminology is inspired by the biological operations of specialized cells called neurons. A neuron is a cell that has several inputs that can be activated by some outside process.

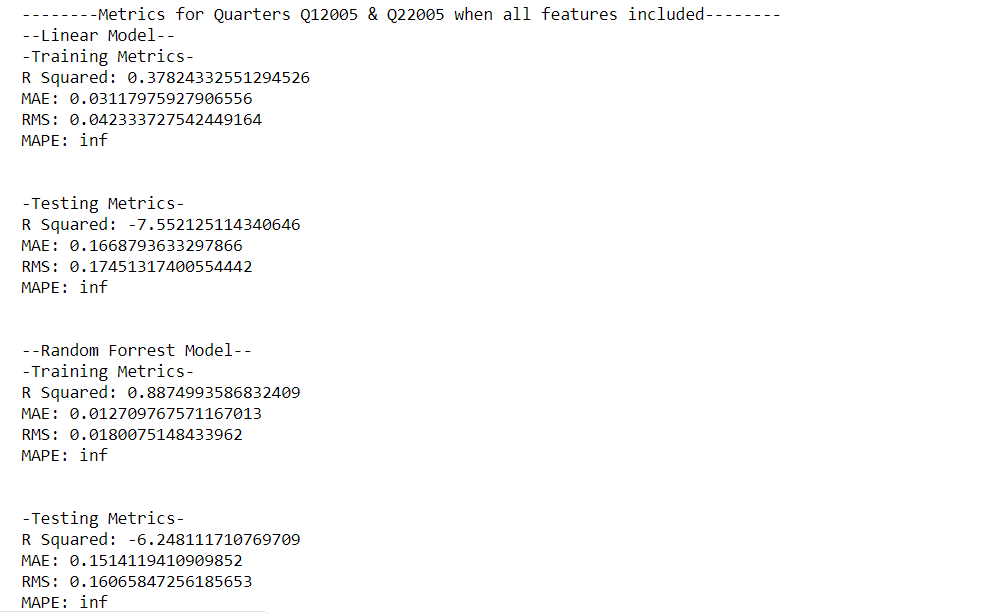
The artificial equivalent of a neuron is a node (also sometimes called neurons, but I will refer to them as nodes to avoid ambiguity) that receives a set of weighted inputs, processes their sum with its activation function, and passes the result of the activation function to nodes further down the graph.

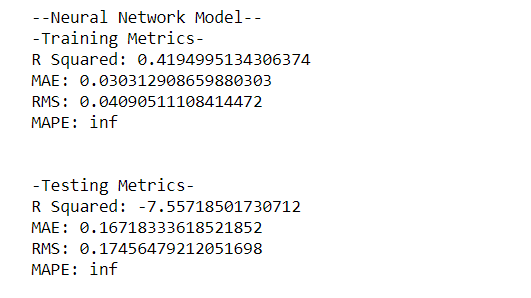


2.1.4.5 STORING AND RETURNING THE RESULTS

We are calculating the results from the different machine learning algorithms, where we are capturing following details:

ModelName, RMSE.baseline, MAE.baseline and Median Absolute error.

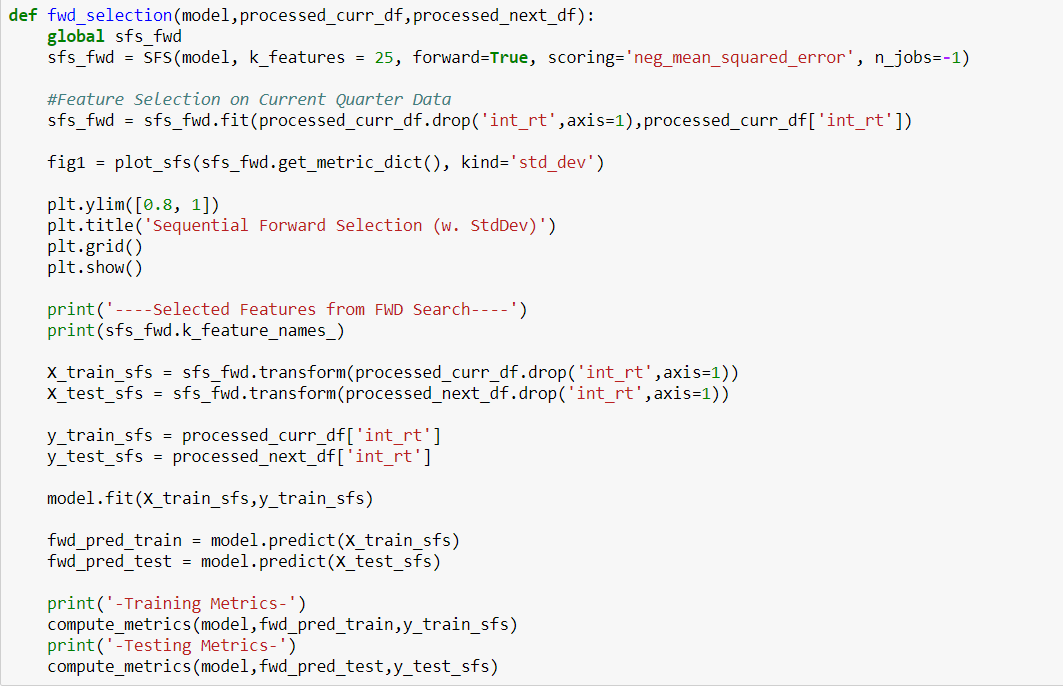


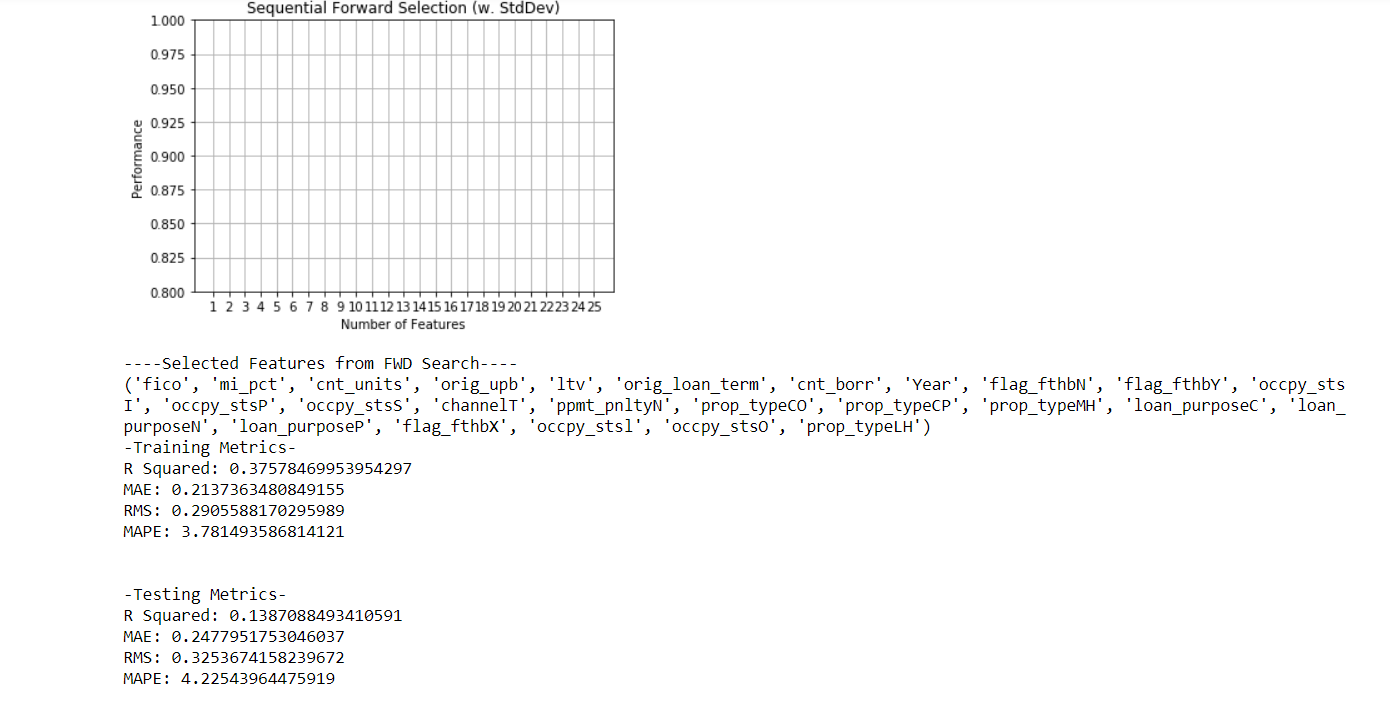


Sequvential Feature Selection

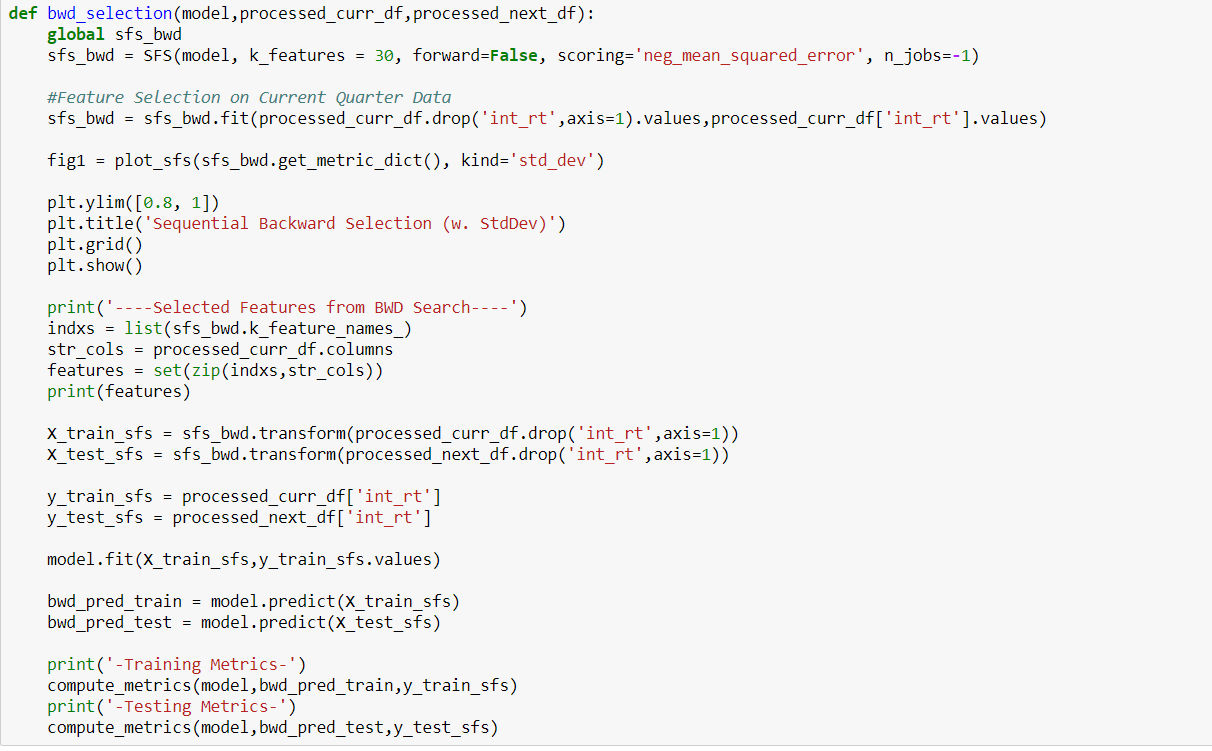
\Sequential feature selection algorithms are a family of greedy search algorithms that are used to reduce an initial d-dimensional feature space to a k-dimensional feature subspace where k < d. The motivation behind feature selection algorithms is to automatically select a subset of features that is most relevant to the problem. The goal of feature selection is two-fold: We want to improve the computational efficiency and reduce the generalization error of the model by removing irrelevant features or noise. A wrapper approach such as sequential feature selection is especially useful if embedded feature selection -- for example, a regularization penalty like LASSO -- is not applicable.

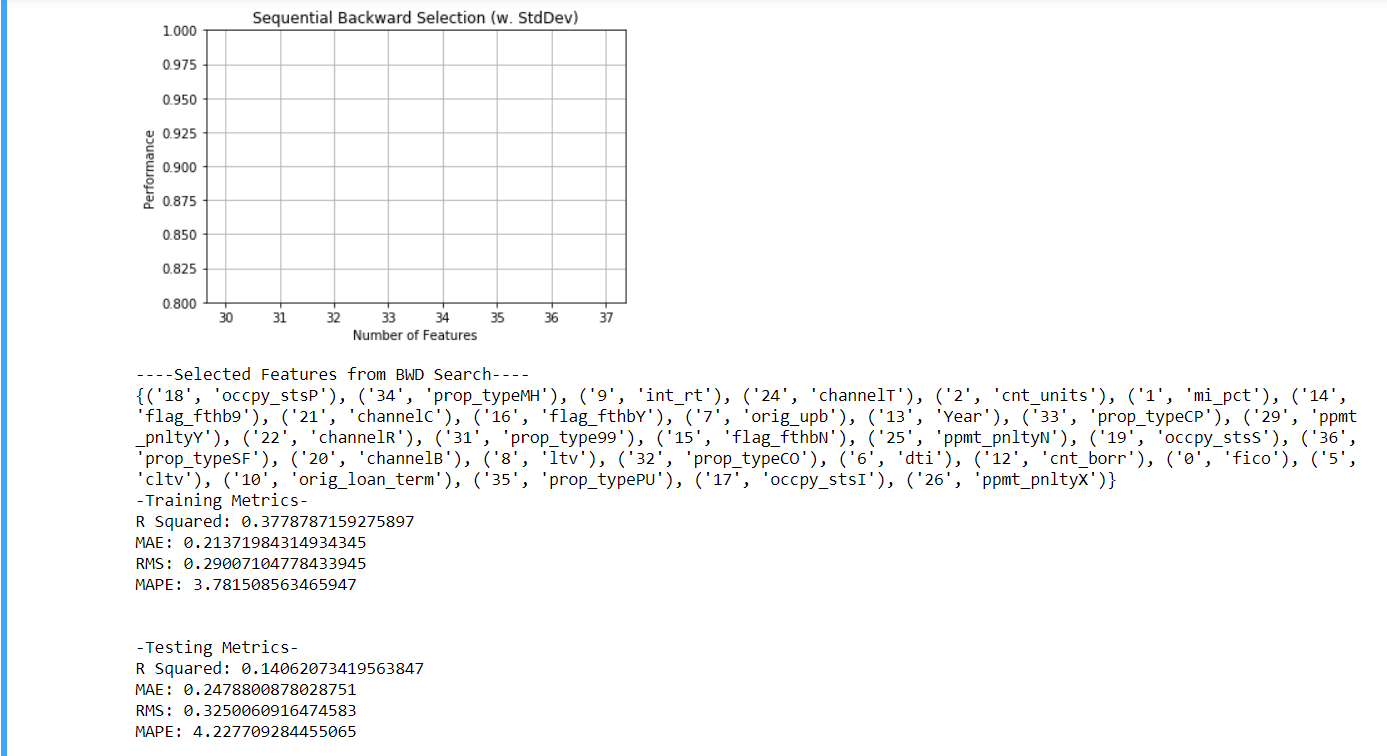
Forward Selection



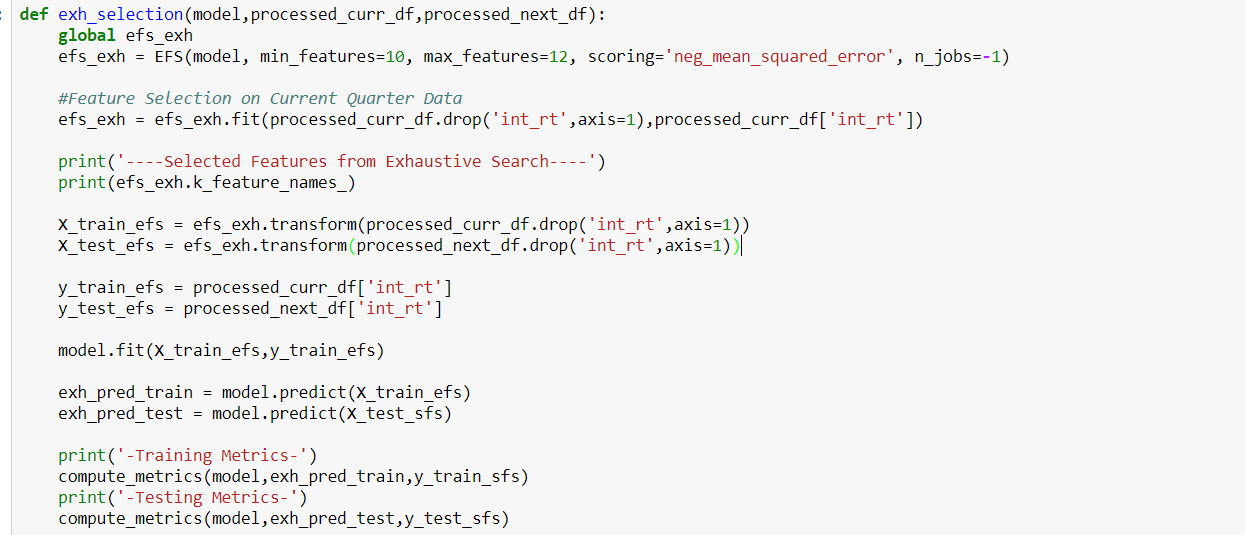


Backward Selection





Exhaustive Feature Selection

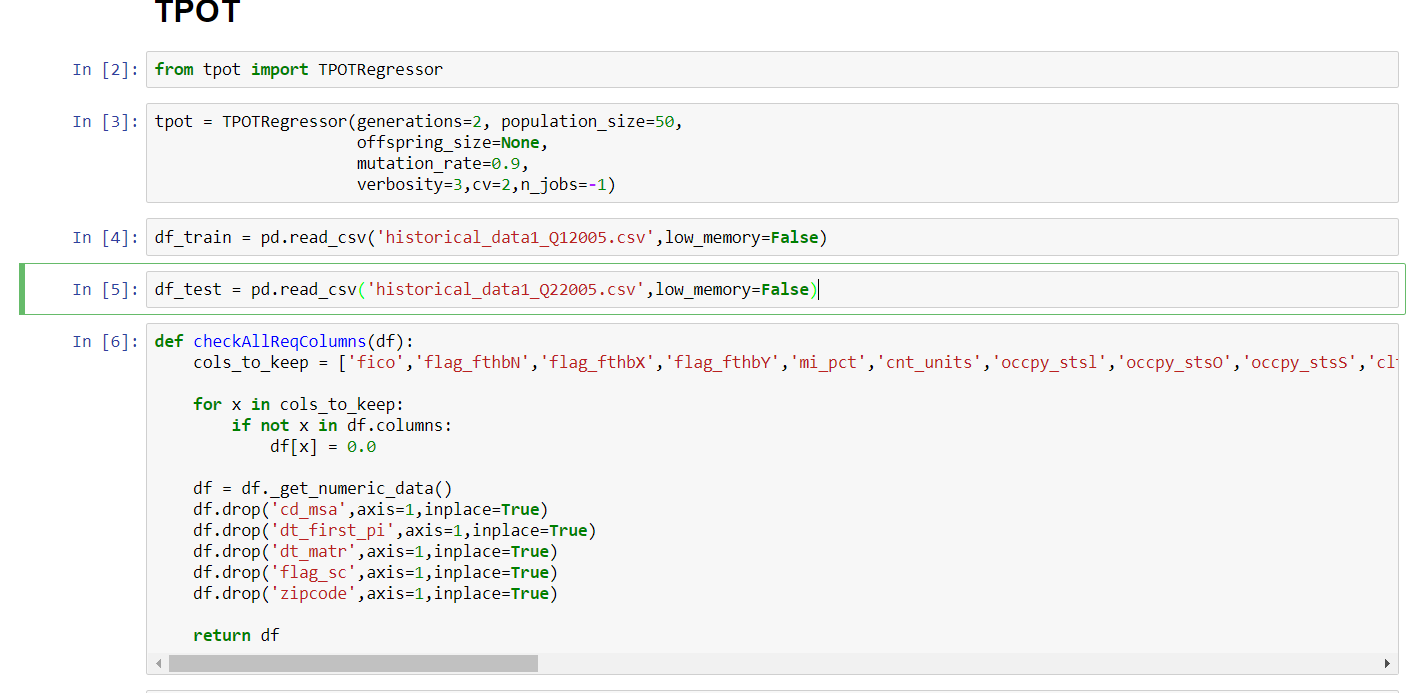
This exhaustive feature selection algorithm is a wrapper approach for brute-force evaluation of feature subsets; the best subset is selected by optimizing a specified performance metric given an arbitrary regressor or classifier. For instance, if the classifier is a logistic regression and the dataset consists of 4 features, the alogorithm will evaluate all 15 feature combinations (if min\_features=1 and max\_features=4)

Try TPOT, H20.Ai and AutoSKLearn Automl algorithms

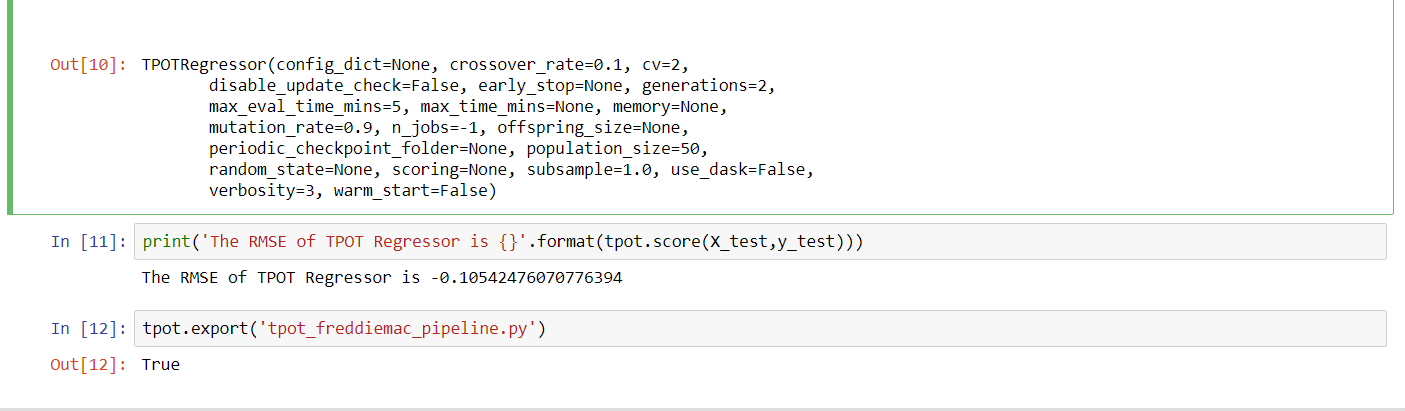
TPOT

The 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. TPOT was developed by [Dr. Randal Olson](http://www.randalolson.com/) while a postdoctoral student with [Dr. Jason H. Moore](http://epistasis.org/jason-h-moore-phd/) at the [Computational Genetics Laboratory](http://epistasis.org/) of the University of Pennsylvania and is still being extended and supported by this team.

The goal of TPOT is to automate the building of ML pipelines by combining a flexible [expression tree](https://en.wikipedia.org/wiki/Binary_expression_tree) representation of pipelines with stochastic search algorithms such as [genetic programming](https://en.wikipedia.org/wiki/Genetic_programming). TPOT makes use of the Python-based [scikit-learn](http://scikit-learn.org/) library as its ML menu.



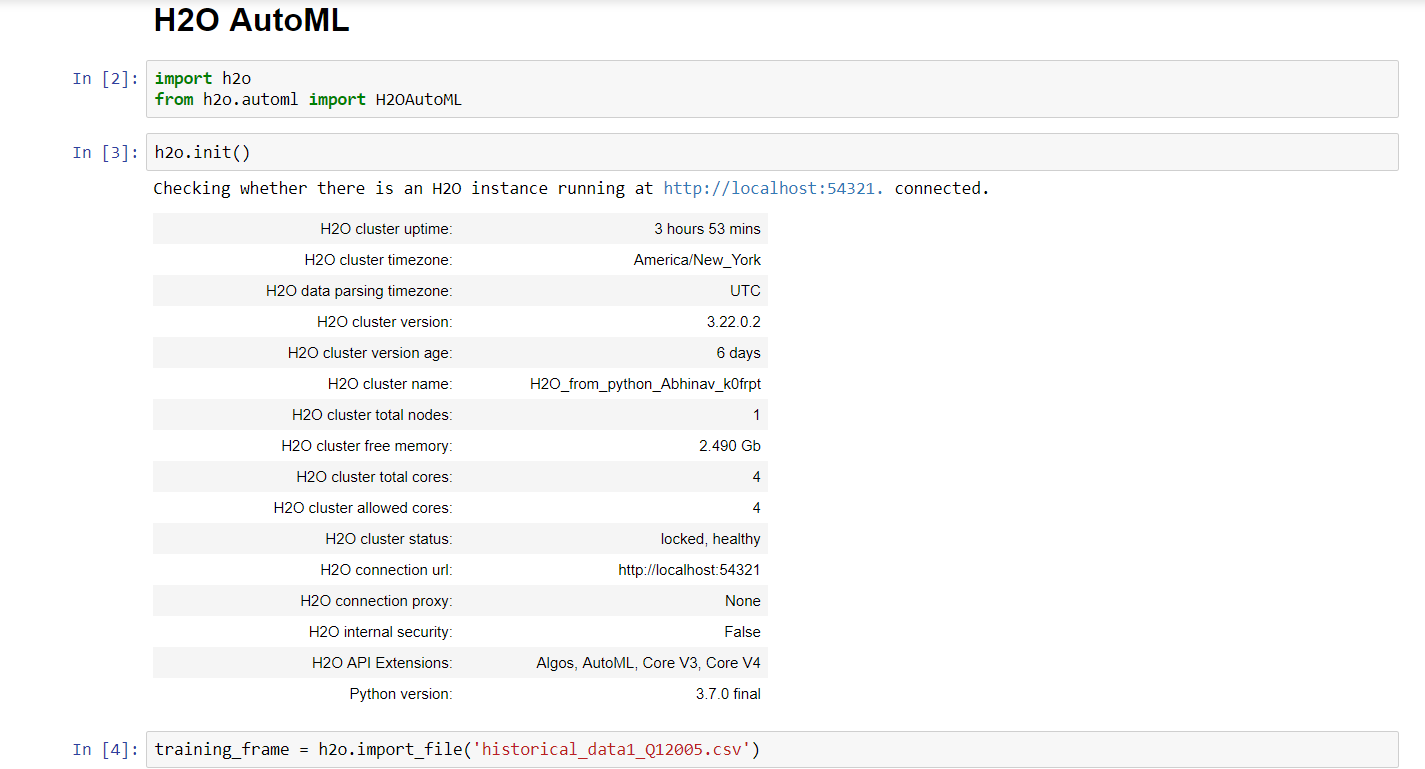


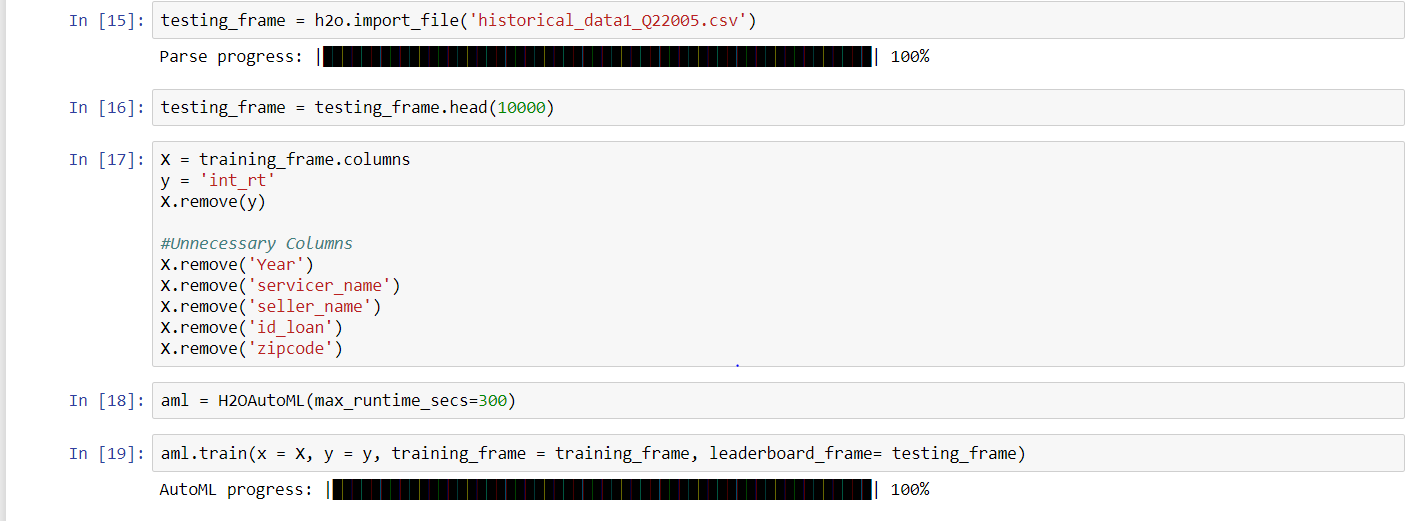


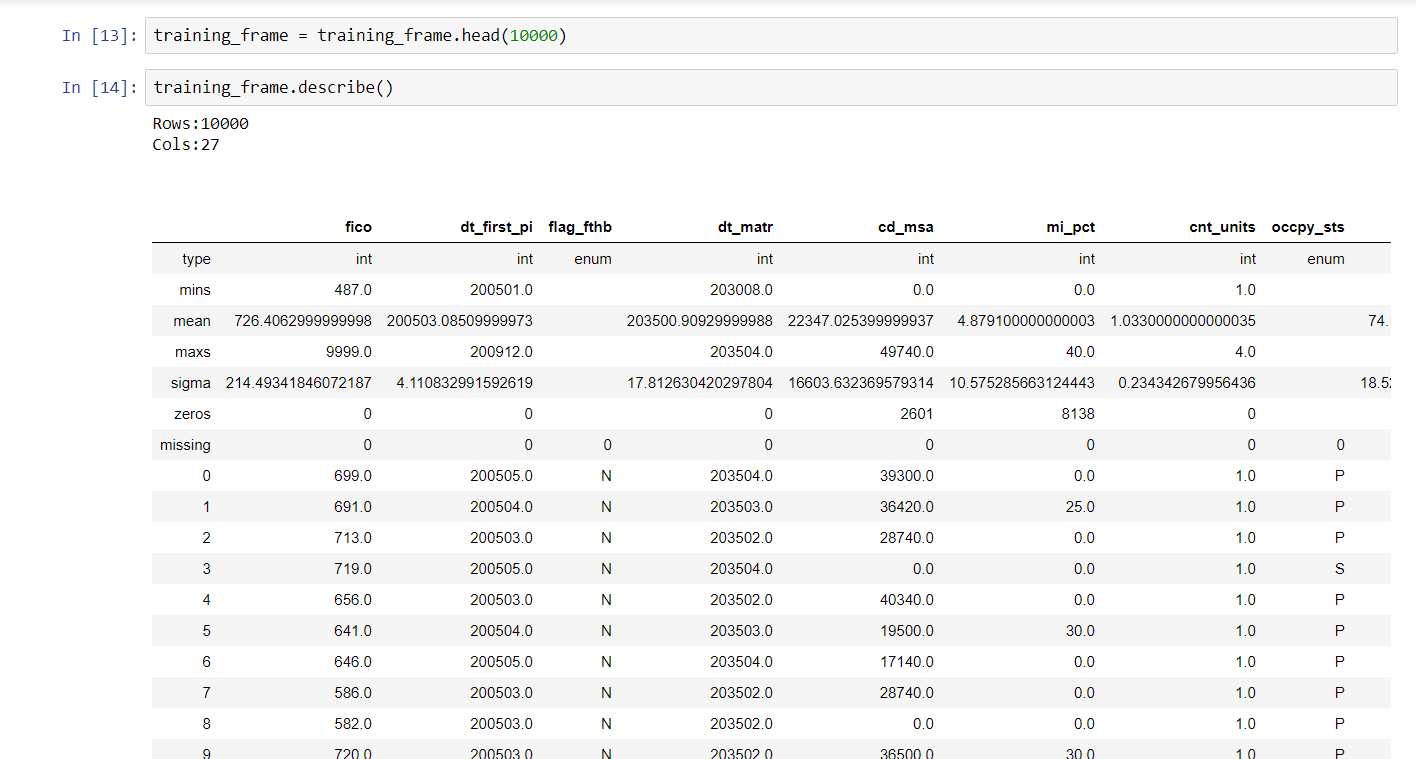
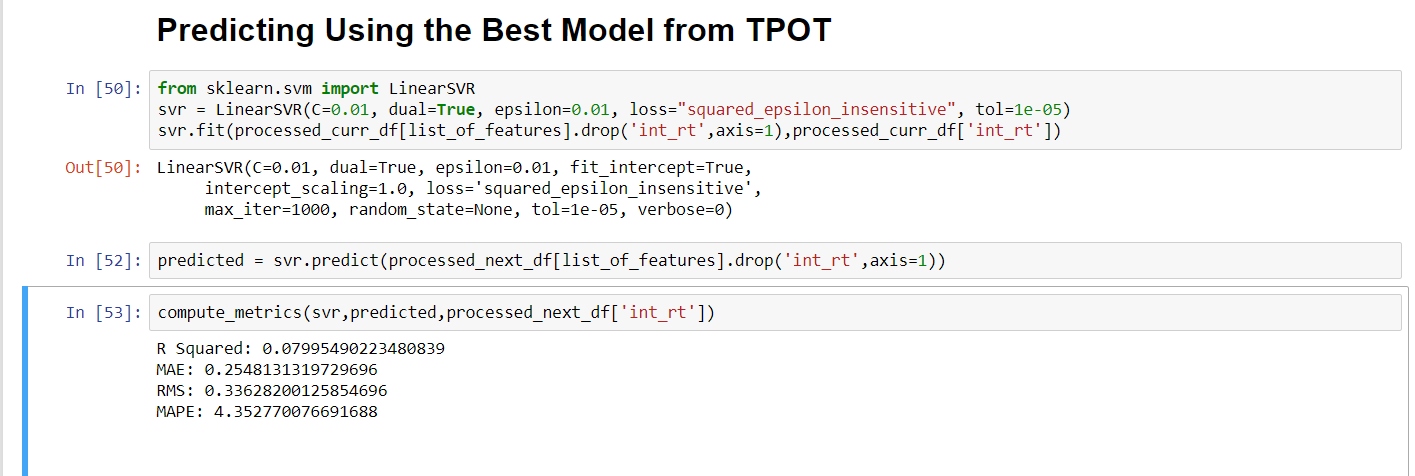
# H2O AutoML

AutoML is a function in H2O that automates the process of building a large number of models, with the goal of finding the "best" model without any prior knowledge or effort by the Data Scientist.

The current version of AutoML (in H2O 3.16.\*) trains and cross-validates a default Random Forest, an Extremely-Randomized Forest, a random grid of Gradient Boosting Machines (GBMs), a random grid of Deep Neural Nets, a fixed grid of GLMs, and then trains two Stacked Ensemble models at the end. One ensemble contains all the models (optimized for model performance), and the second ensemble contains just the best performing model from each algorithm class/family (optimized for production use).







2.1.4.6 PREDICTION MODEL EVALUATION: WHICH MODEL TO CHOOSE

We had compared results of all the models decided **Random Forest** gives us much better result.

1. Higher average predictive accuracy
2. Moderate prediction speed
3. Performs well with large number of observations
4. Handles lots of irrelevant features well

**WHAT -IF Analysis**

**Financial Crisis Analysis**

When the [housing bubble](http://www.investopedia.com/terms/h/housing_bubble.asp) of 2001-2007 burst, it caused a [mortgage](http://www.investopedia.com/terms/m/mortgage.asp) security meltdown. This contributed to a general [credit crisis](http://www.investopedia.com/terms/c/credit-crisis.asp), which evolved into a worldwide [financial crisis](http://www.investopedia.com/terms/f/financial-crisis.asp). Many critics have held the United States Congress - and its unwillingness to rein in [Fannie Mae](http://www.investopedia.com/terms/f/fanniemae.asp) and [Freddie Mac](http://www.investopedia.com/terms/f/freddiemac.asp) - responsible for the credit crisis.

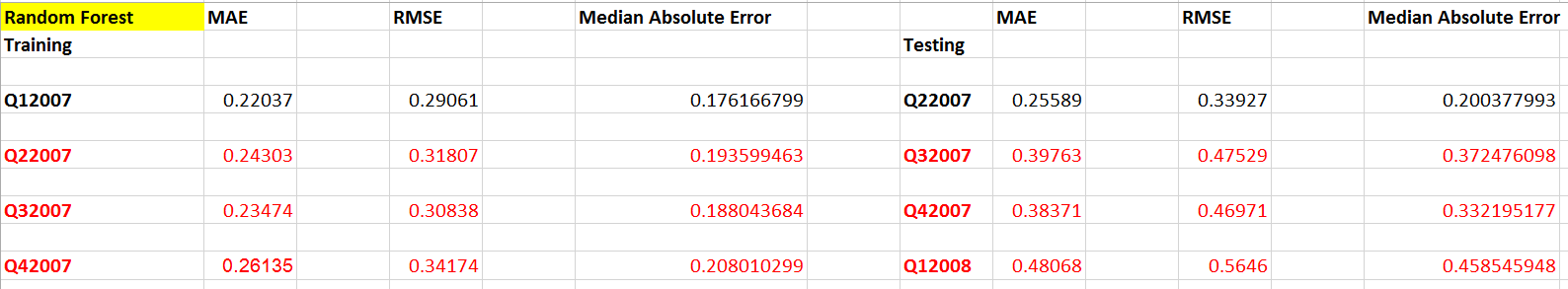
In the fall of 2007, Freddie Mac shocked the market by announcing large credit-related loses, fueling the fire for the argument that the two companies pose a tremendous risk to the entire financial system.

(<http://www.investopedia.com/articles/economics/08/fannie-mae-freddie-mac-credit-crisis.asp>)

The Federal Home Loan Mortgage Corporation (Freddie Mac) announced that it will no longer buy the most risky subprime mortgages and mortgage-related securities.

In July 24, 2007 Countrywide Financial Corporation warned of “difficult conditions.” This is evident from the Q32007 Testing measures as the difference between Training and Testing RMSE increased substantially by around 16%.

In November 1, 2007 financial market pressures intensified, reflected in diminished liquidity in interbank funding markets. This is evident in Q42007 Testing measures as the difference between Training and Testing RMSE increased substantially by around 22%.



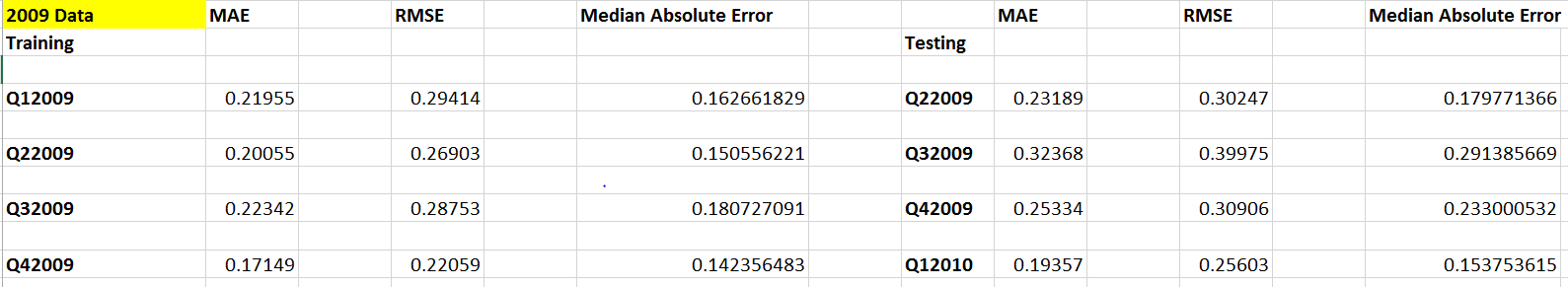
**Two Years Later (2009):**

Financial markets recovered substantially since March 2009 when the financial stress began to ease and market conditions started to improve. ( <http://www.oecd.org/daf/fin/financial-markets/44563803.pdf>)

In 2009, Freddie Mac played a critical role in supporting the nation’s housing market by:

* Providing $548.4 billion of liquidity to the mortgage market, helping finance approximately 2.2 million conforming single-family loans and approximately 253,000 units of multifamily rental housing.
* Helping more than 272,000 borrowers stay in their homes or sell their properties through the company’s long-standing foreclosure avoidance programs and the Home Affordable Modification program (HAMP), including 129,380 loans that remained in HAMP trial periods as of December 31, 2009 according to information provided by the Making Home Affordable (MHA) program administrator.
* Refinancing approximately $379 billion of single-family loans, creating an estimated $4.5 billion in annual interest savings for borrowers nationwide – this includes approximately 169,000 borrowers whose payments were reduced by an average of $2,000 annually under the Freddie Mac Relief Refinance MortgageSM

( <http://www.freddiemac.com/news/archives/investors/2010/2009er-4q09.html>)



As clearly evident from the analysis, Measures are pretty much stable for 2009.

**Economic Boom (1999,2013):**

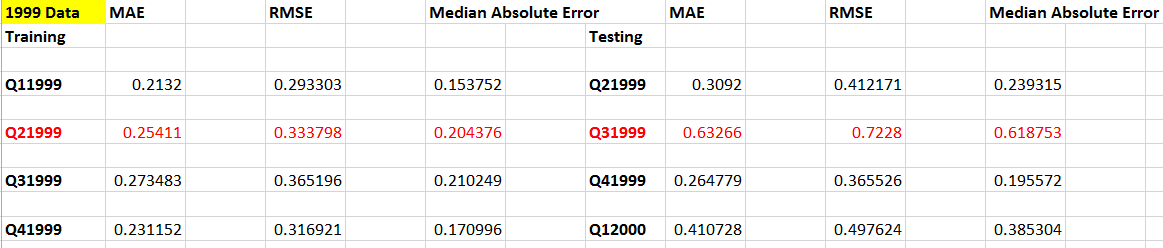
**1999:**

The easing of credit also coincided with spectacular stock market run-ups from 1999 to 2000

Freddie Mac financed homes for more than 2 million families and achieved record earnings per share of $2.96, an increase of 28 percent over 1998.

(<http://www.freddiemac.com/investors/pdffiles/annual99.pdf>)

(<https://en.wikipedia.org/wiki/1990s_United_States_boom>)



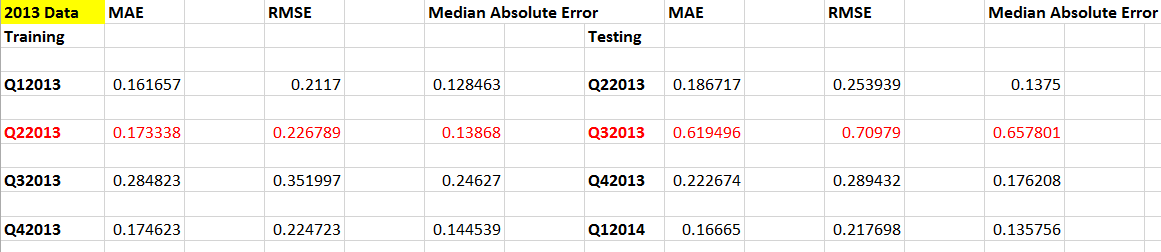
**2013:**

In 2013, Mortgage rates peaked at 4.6% in August and have held steady since September and several accounting events had significant impacts on the Enterprises’ reported financial results. Fannie Mae and Freddie Mac reported levels of 2013 net income are greater than at any prior time in their respective histories. Their historically high net income was driven by reversals of previously accrued losses associated with deferred tax assets (DTA) and their allowance for loan and lease losses (ALLL)—plus revenue from legal settlements of representation and warranties claims and lawsuits regarding private-label securities that the Enterprises purchased as investments. FHFA does not expect benefits of this nature to be repeated in future years and does not expect the 2013 levels of net income to be approached anytime in the foreseeable future.

(<https://www.fhfa.gov/AboutUs/Reports/ReportDocuments/FHFA_2013_Report_to_Congress.pdf#page=18>)

(<http://www.foxbusiness.com/features/2013/12/23/housing-market-in-2013-prices-rise-as-lending-remains-tight.html>)

Drastic change in Training and Testing measures for the highlighted rows clearly shows the transition in economic trends during Q2 and Q3 around 48%.



Would you recommend using this model for the next quarter? Justify

The proposed model will perform well for the next quarter, if there are not major changes in the data patterns such as financial crisis or economic boom.

**CLASSIFICATION (LOAN DELINQUENCY STATUS)**

In Loan Performance file, we have a column name delq\_sts on which we should predict the Loan Delinquency Status by training the data on the quarter provided and predict the result for the next quarter.

3.3 GENERIC APPROACH: CLASSIFCATION



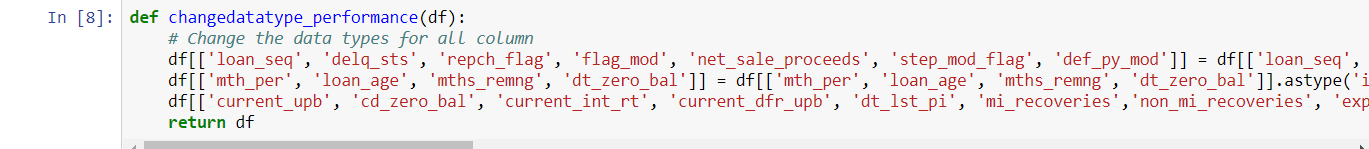
2.1.1 CREATING THE FUNCTION AND DOWNLOAD THE INPUT

We have created functions that would take hit the Freddie Mac website and download all the Historical file for our purpose. Based on the input Quarter, the file will be imported into python Data Frame. It went through following preprocessing before building model.

Handling the missing values:



Created dummy columns (1…C):



2.1.2 CONVERSION OF DATA TYPE

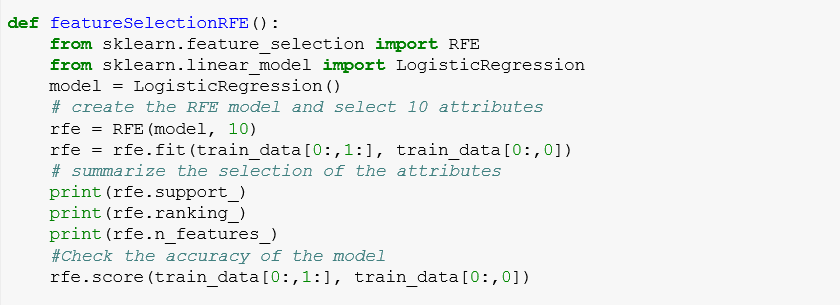
Data types of the History Performance file is converted as below.



2.1.3 FEATURE SELECTION

Before proceeding with our models, we have done best feature selection using three algorithms.

The best features that add to the predictive power of the model and irrelevant features removed from the model. We implemented following feature selection techniques in Python:



We selected variables as per the **RFE ranking** and used those for further analysis while making sure that all the datasets contain same number of columns. Performing all the feature selection methods we shortlisted below features to best predict or model

We will discuss the algorithm used in the below section for all the Machine Learning algorithm used for classifications

3.3.1 DIFFERENT MACHINE LEARNING ALGORITHMS AND OUTPUT

1. 3.1.1 LOGISTIC REGRESSION



Binary Logistic Regression is a special type of regression where binary response variable is related to a set of explanatory variables, which can be discrete and/or continuous. We are using the logistic regression model for training the model for the quarter supplied and predicting the delinquency status based on the trained model.

Import the following libraries to calculate the logit summary for the logistic regression:

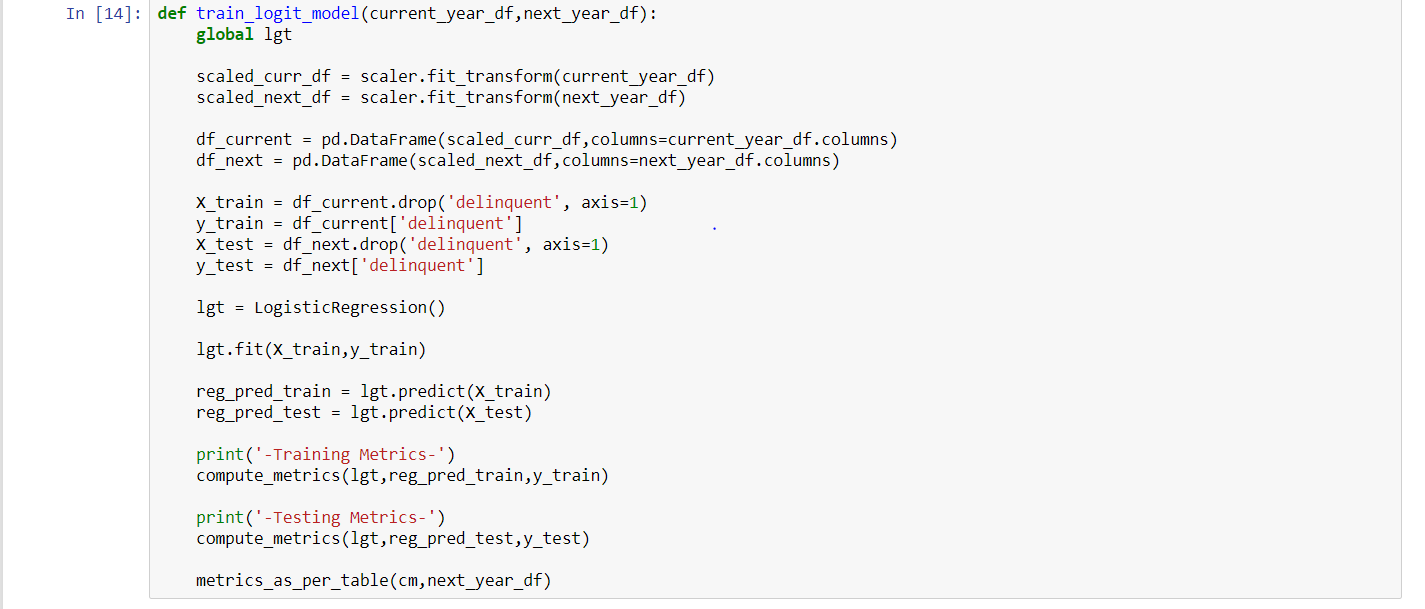
**from sklearn.linear\_model import LogisticRegression**

**model = LogisticRegression()**

**import statsmodels.api as sm**

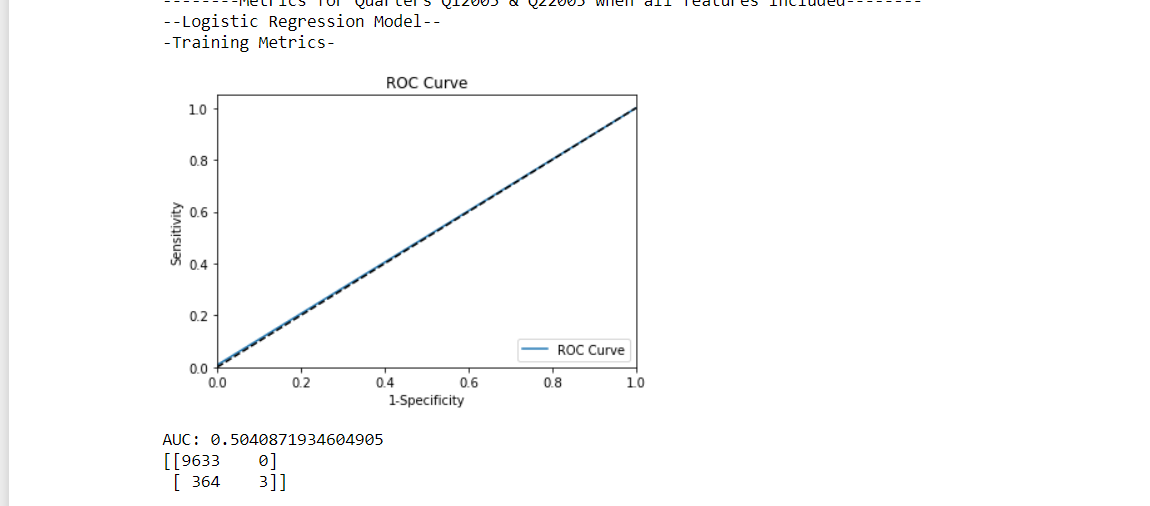
**from statsmodels.formula.api import logit, probit, poisson, ols**

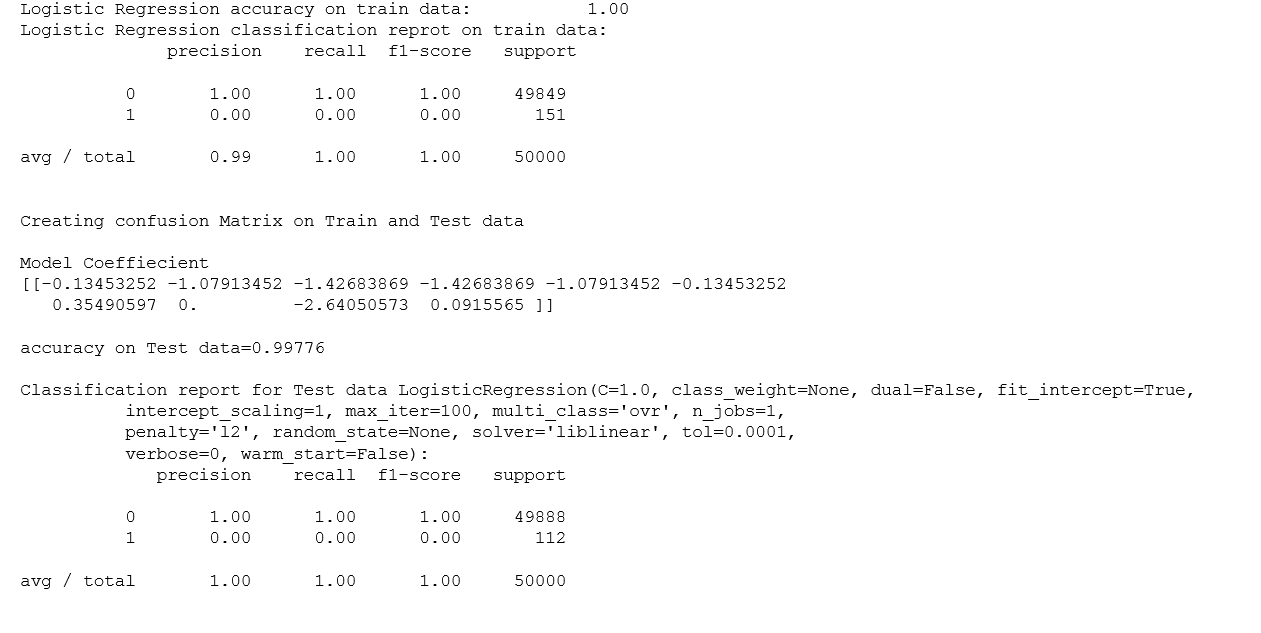
**Apply Logit Function:**

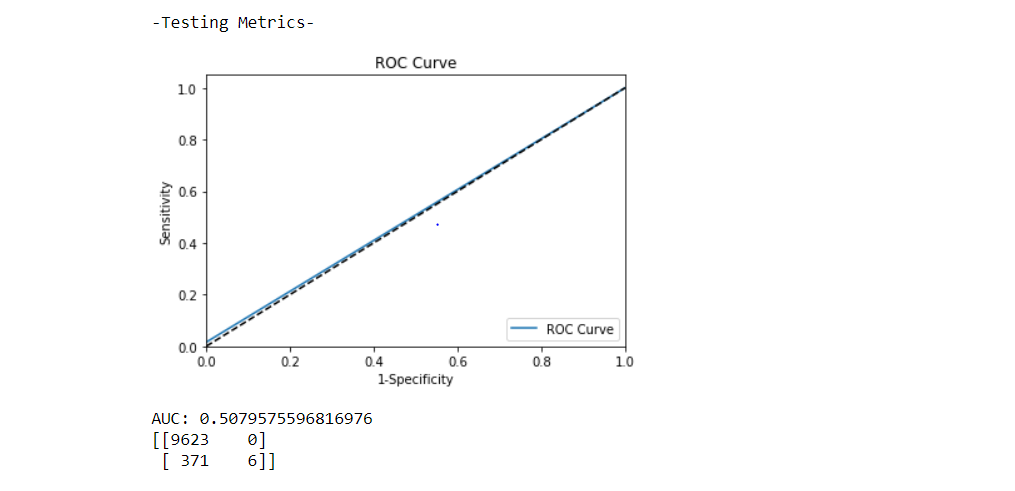


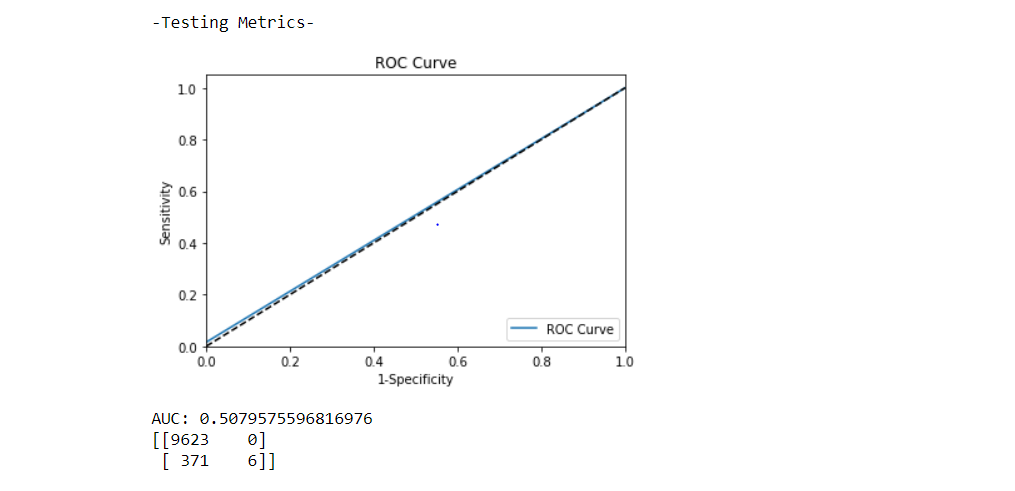
Now predict the delinquency status based on the Test data and generate the accuracy and Classification report.

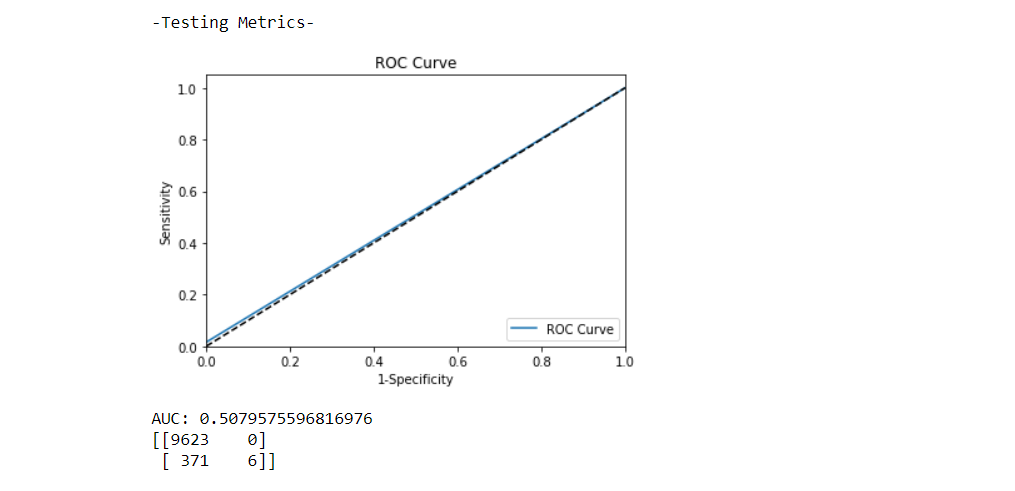
In this case, our function will result the Confusion matrix as shown below and the ROC curve:

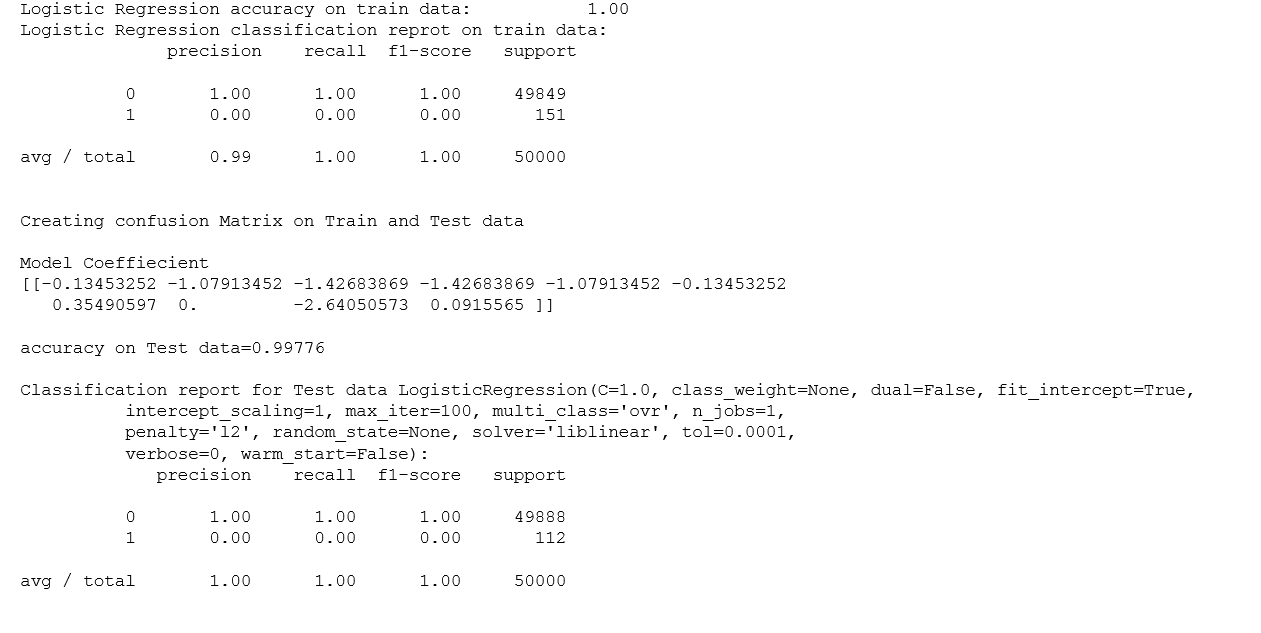








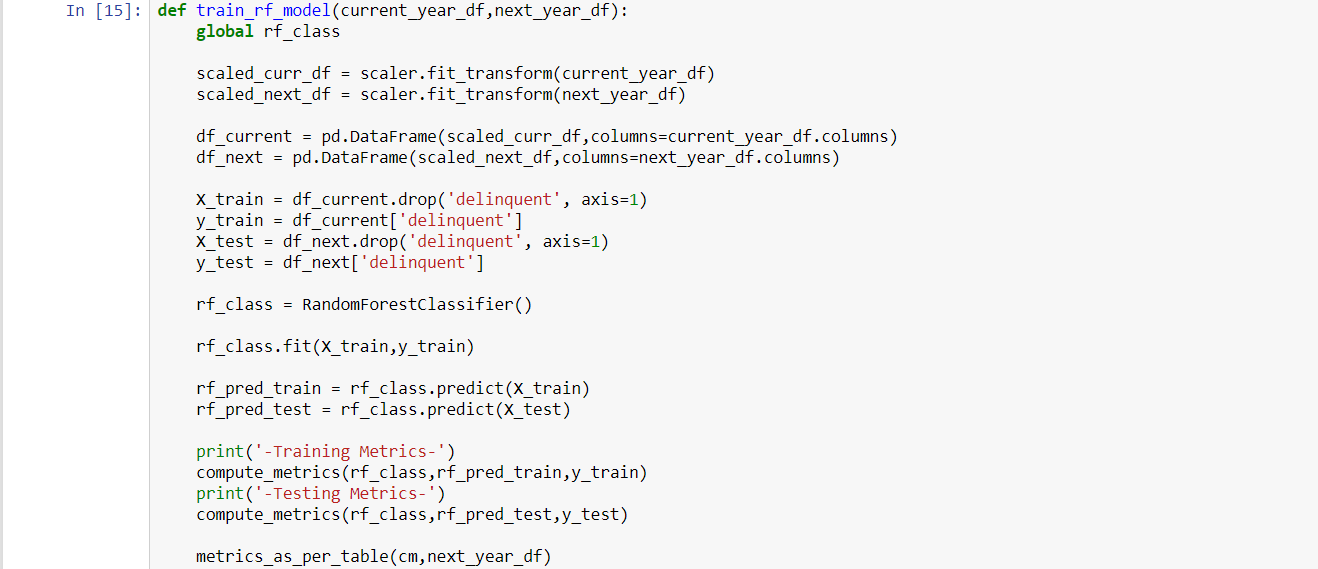




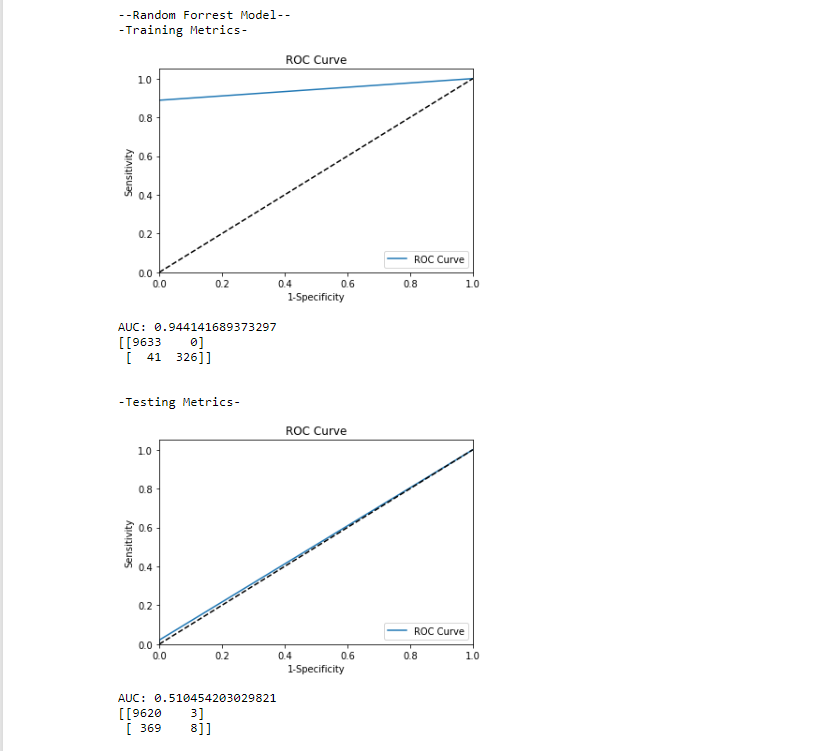


3.3.1.3 RANDOM FOREST

Random Forests grows many classification trees. To classify a new object from an input vector, put the input vector down each of the trees in the forest. Each tree gives a classification, and we say the tree "votes" for that class. The forest chooses the classification having the most votes (over all the trees in the forest).



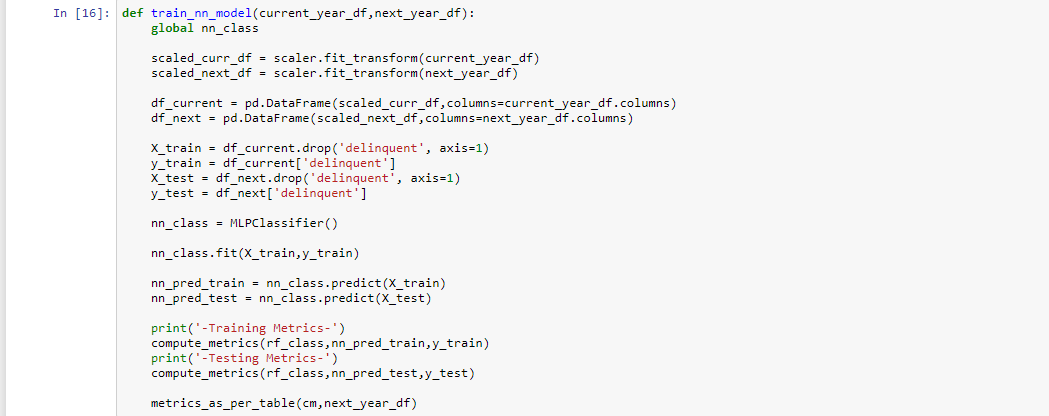
**Accuracy Result & Classification Report**

In this case, our function will result the Confusion matrix as shown below and the ROC curve:

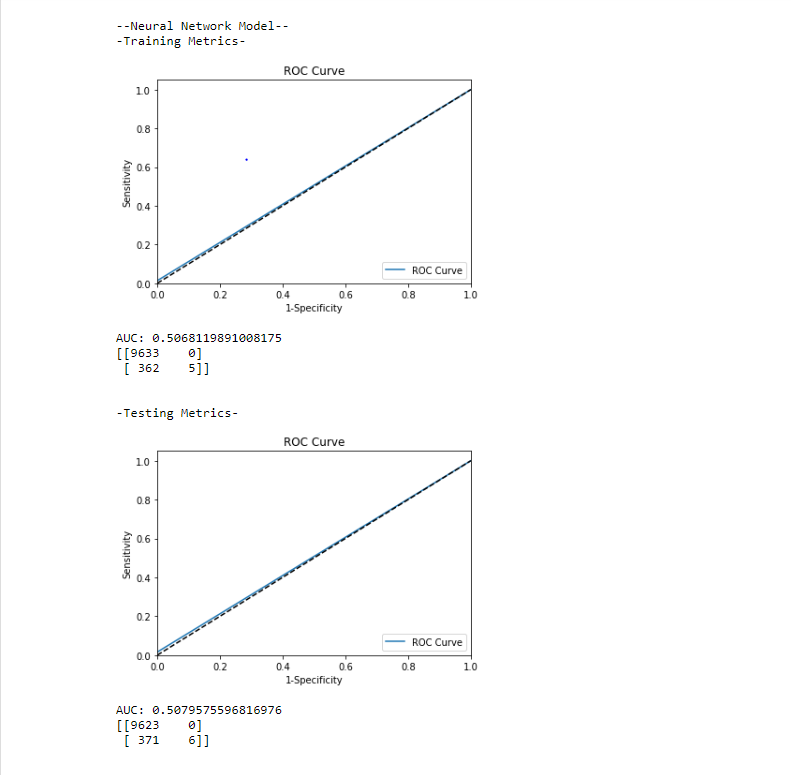
3.3.1.4 NEURAL NETWORK



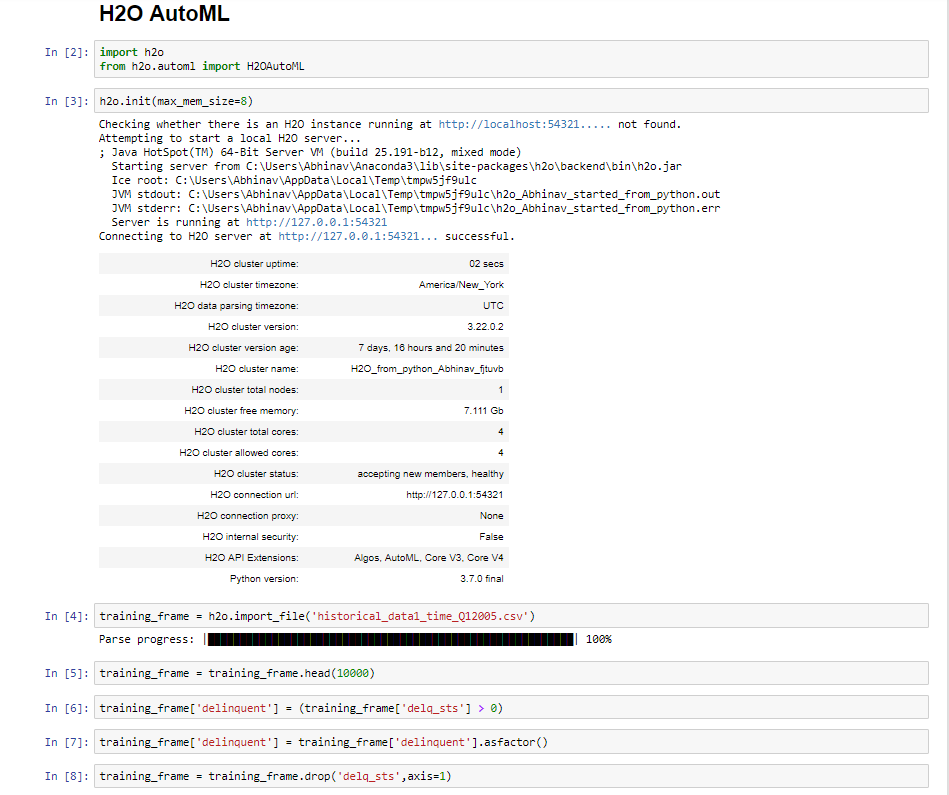
Artificial neural networks are relatively crude electronic networks of neurons based on the neural structure of the brain. They process records one at a time, and learn by comparing their classification of the record (i.e., largely arbitrary) with the known actual classification of the record.

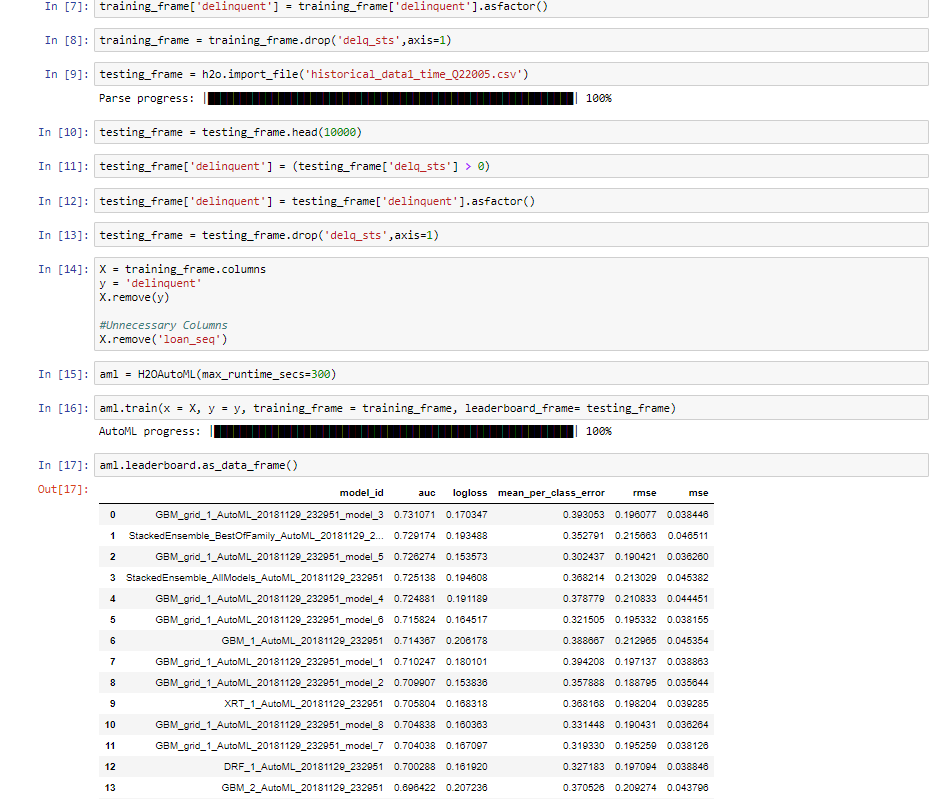


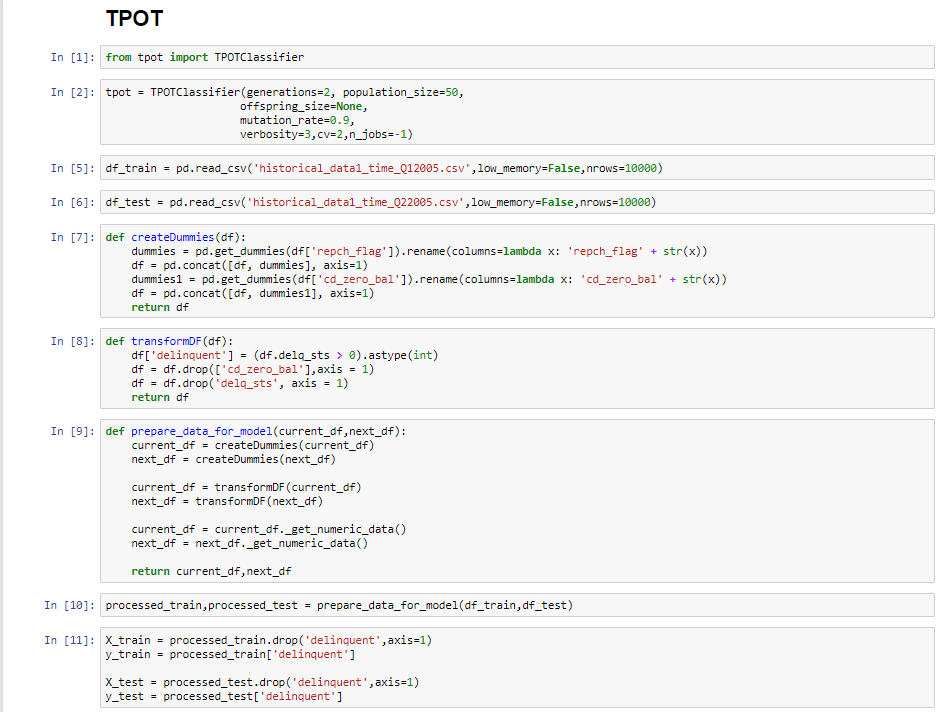
**Accuracy Result & Classification Report**

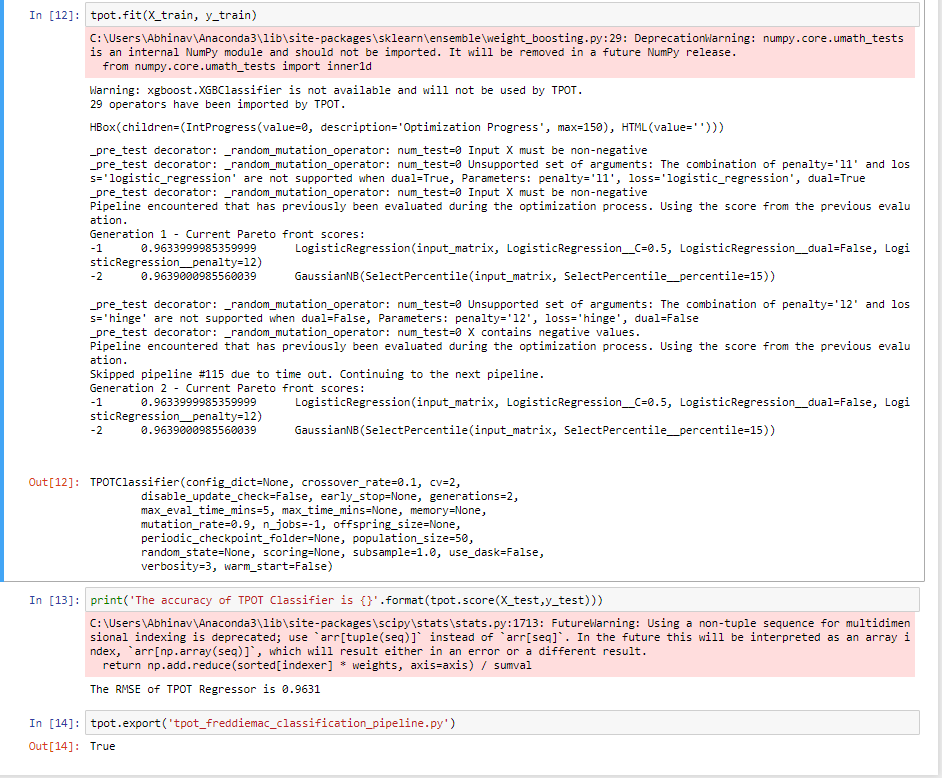
In this case, our function will result the Confusion matrix as shown below and the ROC curve:

Try TPOT, H20.Ai and AutoSKLearn Automl algorithms









**Comment on the quality of the model and it’s outputs. What can you do to do better? Would you recommend using this model to predict delinquents in the next quarter? Justify your answers**

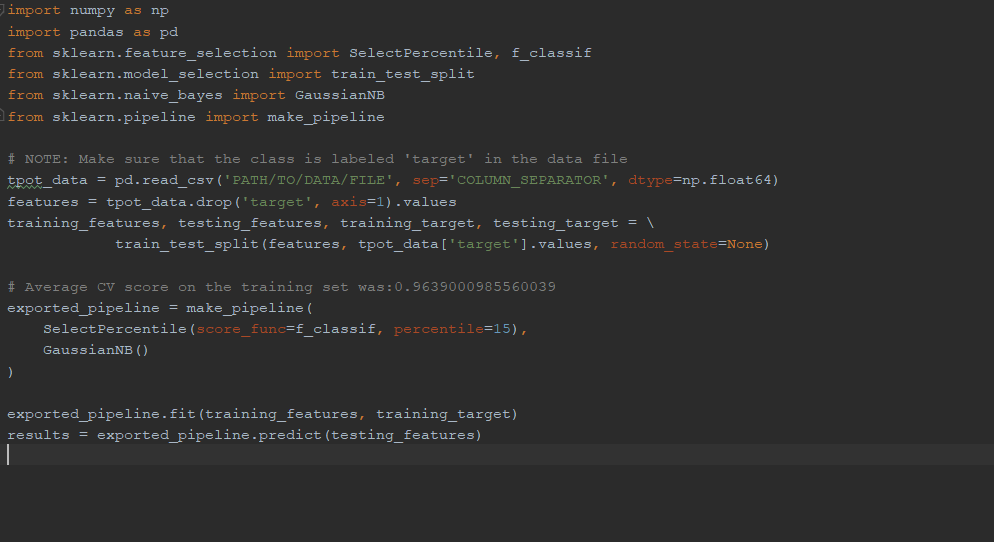
**Random forest (or random forests) is an ensemble classifier that consists of many decision trees and outputs the class that is the mode of the class's output by individual trees**

Why – Random Forest?

* Rand Forest is fast to build. Even faster to predict!
* Practically speaking, not requiring cross-validation alone for model selection significantly speeds training by 10x-100x or more.
* Automatic predictor selection from large number of candidates
* Resistance to over training
* Ability to handle data without preprocessing
* data does not need to be rescaled, transformed, or modified resistant to outliers
* automatic handling of missing values
* Cluster identification can be used to generate tree-based clusters through sample proximity

PIPELINE

Classification



Prediction

