**ADS Midterm Report**

**Part 1: Data Wrangling**

**Data Download and Preprocessing:**

**Approach:**

* Enter the quarter, year, session username and session password in the config.json file in the below mentioned format.
  + Example: "Q42004","p\*\*\*@h\*\*\*u","O\*\*\*\*\*\*i"
* Get the entered year and next year using the first parameter in the config.json file.
* Login using the username and password provided as the second and third paramters in the config.json file.
* Take the response from the session and traverse through the URL to find the downloadable link for the entered year and next year.
* Download the data and return the file names.
* Read data from these files to perform the below analysis.
* The following observations have been modified:
  + Observations with blank CreditScore have been deleted
  + Observations with blank MortgageInsurancePercentage have been deleted
  + Blanks in FirstTimeBuyerFlag have been replaced with ‘U’

**Exploratory Data Analysis:**

**Analysis 1:** Find the state-wise frequency count of first time house buyers

**Approach:**

* Take the input year from the user
* Dynamically load/download the data by creating a file path/ generating a URL
* Calculate the state-wise frequency of people taking loan only if the ‘FirstTimeHomeBuyerFlag’ value is ‘Y’ in a dictionary with key as state name and value as the state-wise frequency.

**Conclusion:**

* Assuming that people opting for home loans are comparatively young and starting a family or planning to start a family, this customer information can help the loan lender firm have a target audience to advertise other loan packages like education/ car loan.

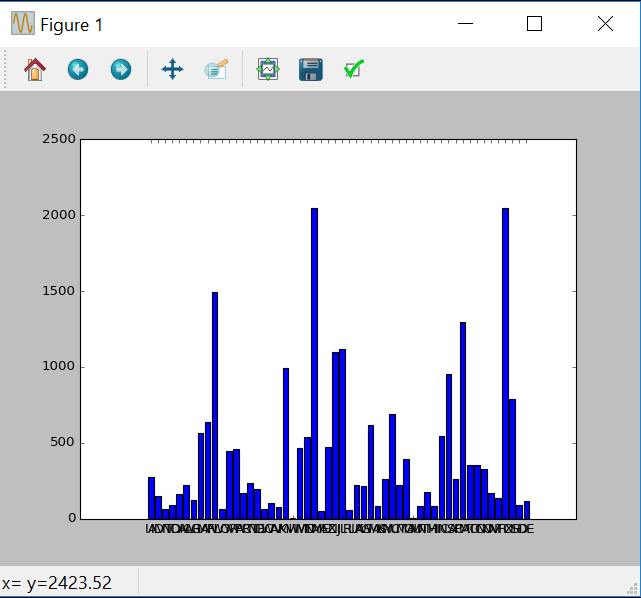
****

Figure 1:Analysis 1

**Analysis 2:** Find the frequency count of number of people taking loan in an area.

**Approach:**

* Take an input year from the user
* Dynamically load/download the data by creating a file path/ generating a URL
* Calculate the frequency of people taking loan state-wise to get an overall distribution flow.
* Calculate the frequency of people taking loan area-wise to get a deeper understanding about the trend and save it in a dictionary with key as state name and value as an inner dictionary with keys as postal code and values as area-wise frequency.

**Conclusion:**

* This frequency trend information can help the loan leader firm understand the states and areas their customers belong to. They can use this information to setup local branches to further cater to their customer’s needs.

**Analysis 3:** Find the frequency count of Occupancy Status state-wise

**Approach:**

* Take the input year from the user
* Dynamically load/download the data by creating a file path/ generating a URL
* Calculate the frequency count for every state based on the Occupancy Status

**Conclusion:** This state wise information will give loan lenders a clear picture about the loan usage. They could change their policies like have a smaller interest rate if the house is ‘OwnerOccupied’ and a higher interest of the house is going to be an ‘InvestmentProperty’.

**Output File:** **OccupancyStatus.csv**

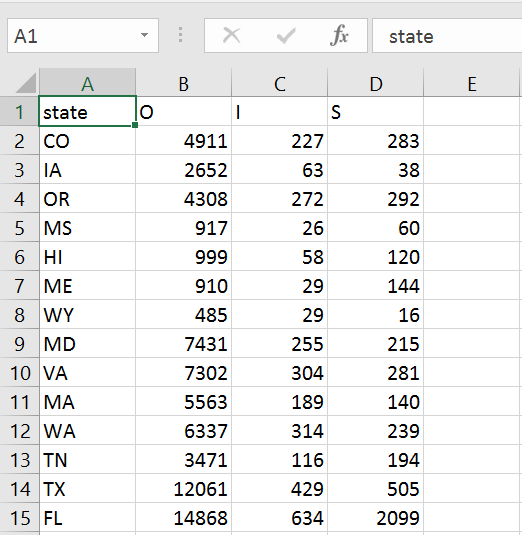
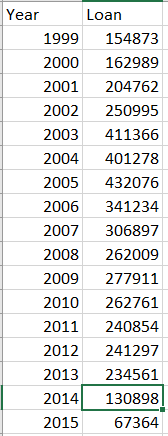
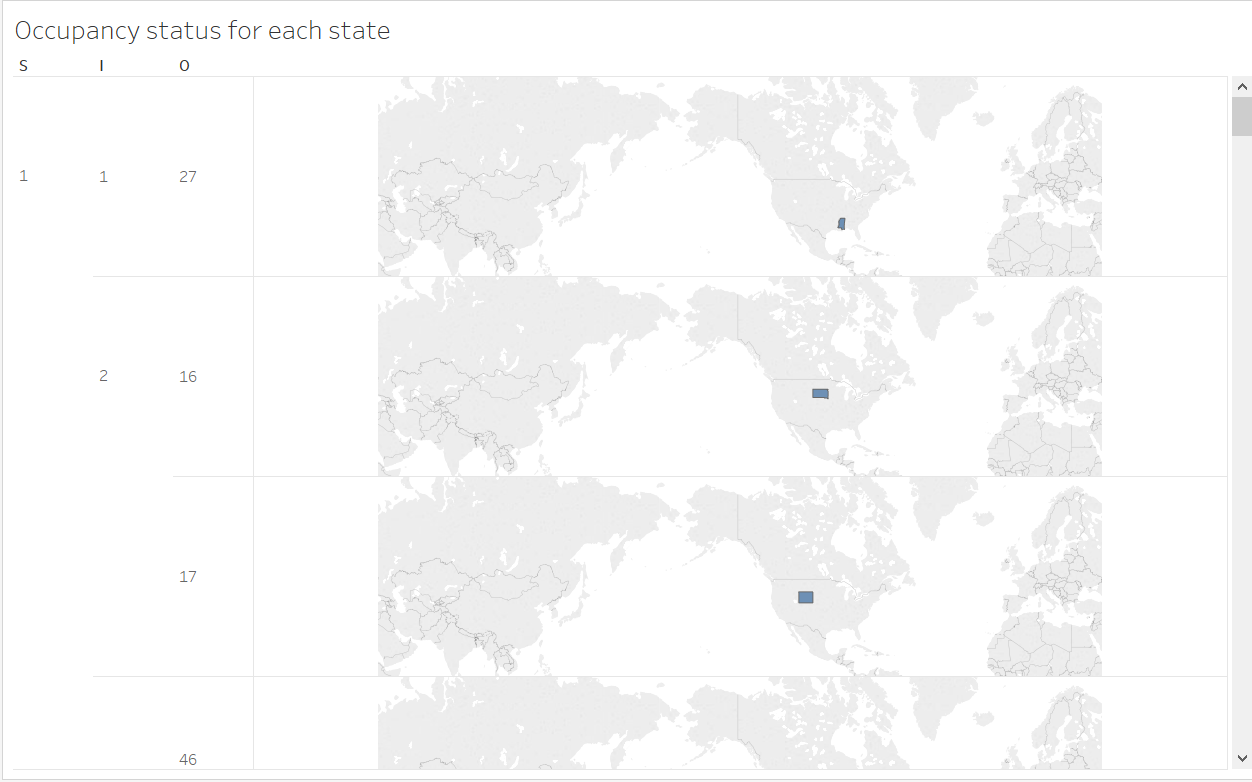
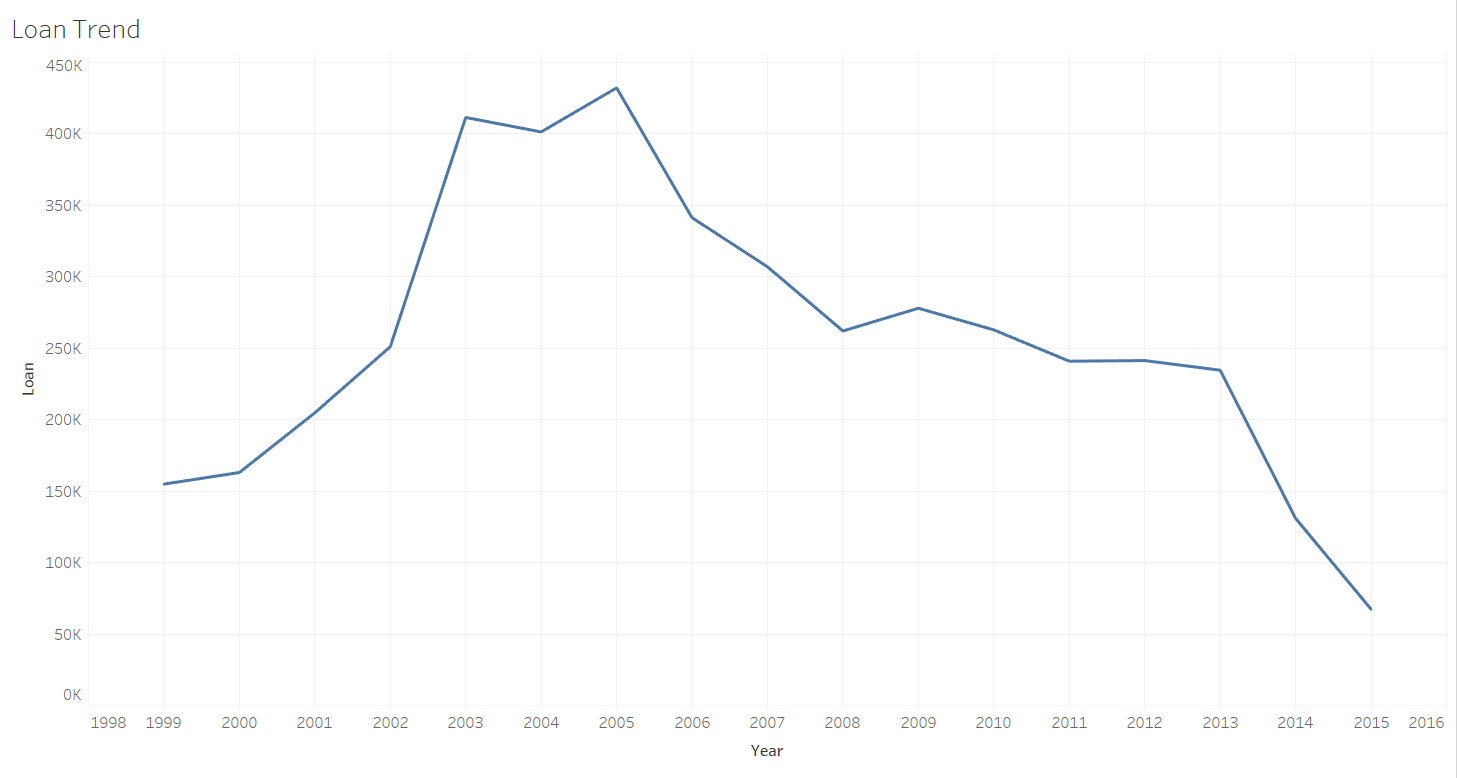
****

Figure 3: OccupancyState.csv

**Analysis 4:** Total number of loans processed

* **Approach:** This frequency trend information can help the loan leader and mortgage firm understand the trends about the amount of loan consumed in the market and the intrest rest trends over the period of time. They can use this information to setup local branches to further cater to their customer’s needs.
* **Output File:**

****



**Part 2**

**Building and Evaluation Models**

**Approach:**

* Enter the quarter, year, session username and session password in the config.json file in the below mentioned format.
  + Example: "Q42004","p\*\*\*@h\*\*\*u","O\*\*\*\*\*\*i"
* Get the entered year and next year using the first parameter in the config.json file.
* Login using the username and password provided as the second and third paramters in the config.json file.
* Take the response from the session and traverse through the URL to find the downloadable link for the entered year and next year.
* Download the data and return the file names.
* Read data from these files to perform the below predictions and classification models.
* Add respective headers to every column in the csv file using add\_header() function
* Strip the data to remove all spaces from the csv file using df\_strip() function
* Find the percentage of missing values in every column in the dataset using missing\_values\_table() function
* Convert string type objects to numeric type for prediction and classification models using ConvertToNumeric1() and ConvertToNumeric2()
* Select input columns and place them in a dataframe X using cols\_to\_keep() function
* Select output column (OriginalInterestRate) and place it in dataframe Y.

**Prediction:**

**Linear Regression:**

**Approach:**

* Convert the 2D dataframe to 1D dataframe using np.ravel(2D) function
* Build a linear regression model using linear\_model.LinearRegression()
* Fit selected input and output data into this model using lm.fit(input,output)
* Find accuracy of the linear regression model using lm.score(inout,output)

**Train the model:**

* To predict the accuracy of the model, read the data for the next quarter and perform data cleaning using add\_header(), strip(), missing\_values\_table(), ConvertToNumeric1() and ConvertToNumeric2() functions.
* Select input values in X1 and actual output values in Y1.
* Again convert the actual output from 2D to 1D using np.ravel() function.
* Predict the linear model using lm.predict() for X1 that is input from next quarter.
* Calculate MAE using mean\_absolute\_error().
* Similary find MSE and Median Absolute Error using mean\_squared\_error() and median\_absolute\_error() respectively.

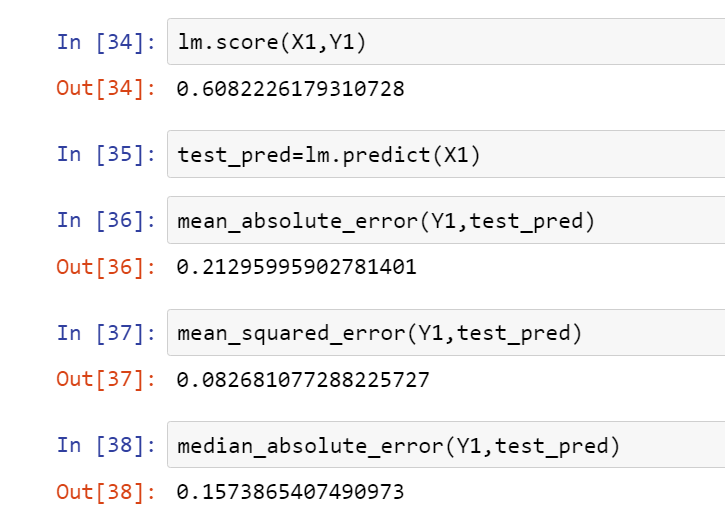
****

Figure 4: Linear Regression Output

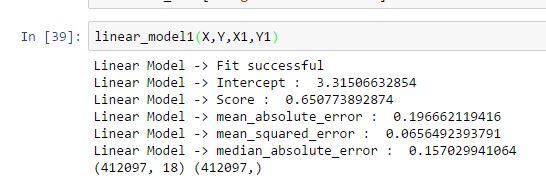


Figure 5: Linear Regression Output

**Random Forest:**

**Approach:**

* We are a generating Random Forest by creating multiple Decission Trees.
* DecissionTrees are created using DecissionTreeRegressor().
* Random Forest is created using RandomForestRegressor().
* fit(input, output) is used to fit the according to our selected input and output dataframes.

**Train the Model:**

* Once the model is created we are training the model using next quarter data.

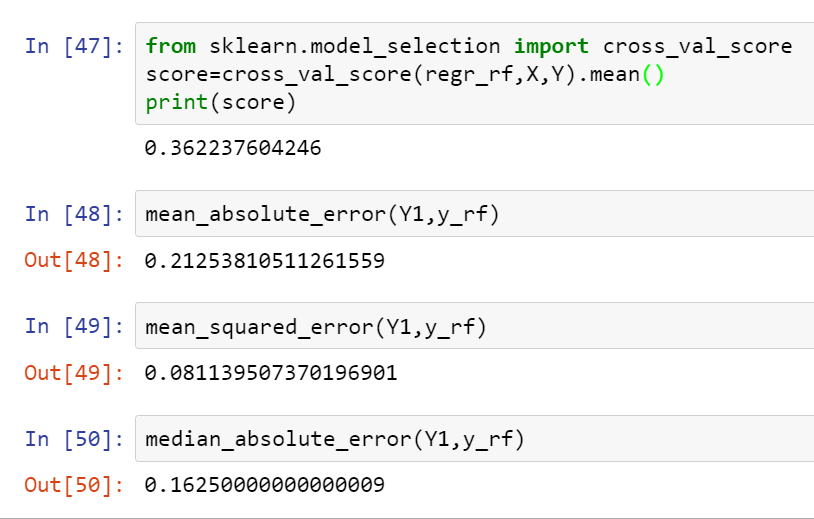
****

Figure 5: Random Forest Output

**NeuralNetworks:**

**Approach:**

* Build the neural net using buildNetwork() command.
* This command takes number of input parameters, number of hidden nodes for every hidden layer and number of output parameters as input.
* Add every row to the sample using addSample() function

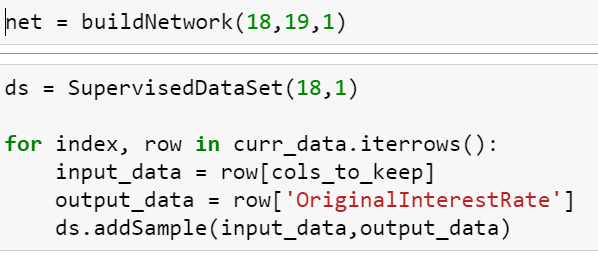
****

Figure 6: Neural Networks

**Train the Model:**

* Train the model using BackpropTrainer

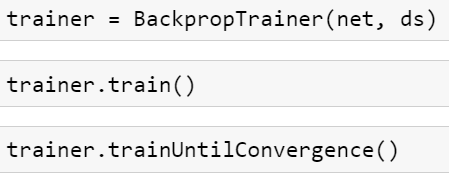
****

Figure 7: Neural Networks

**KNN Model:**

**Approach:**

* Take k as 3 to get 3 nearest neighbors.
* Nearest neighbors are found using the getNeighbors function which finds the distance using euclideanDistance functions for all the nodes and finds the finding the three least values.

**Train the model:**

* The model is trained using getResponse function.
* getAccuracy() function gives the accuracy of the model for a selected dataset.





**Algorithm selection**

After cross validation on the algorithms, based on RMS value generated by each model. It was concluded that the RMS value of Linear Regression has the least value. Now on this the trained algorithm different data set are executed.

**Classification - Part 1 (Algorithm selection)**

This part deals with classification of data that are required to make prediction on that class:

**Approach:**

* Enter the quarter, year, session username and session password in the config.json file in the below mentioned format.
  + Example: "Q42004","p\*\*\*@h\*\*\*u","O\*\*\*\*\*\*i"
* Get the entered year and next year using the first parameter in the config.json file.
* Login using the username and password provided as the second and third paramters in the config.json file.
* Take the response from the session and traverse through the URL to find the downloadable link for the entered quarter and next quarter.
* Download the data and return the file names.
* Read data from these files to perform the below predictions and classification models.
* Add respective headers to every column in the csv file using add\_header() function
* Strip the data to remove all spaces from the csv file using df\_strip() function
* Find the percentage of missing values in every column in the dataset using missing\_values\_table() function
* Convert string type objects to numeric type for prediction and classification models using ConvertToNumeric1() and ConvertToNumeric2()
* Select input columns and place them in a dataframe X using cols\_to\_keep() function

Select output column (OriginalInterestRate) and place it in dataframe Y.

**Algorithm 1: Logistic Regression**

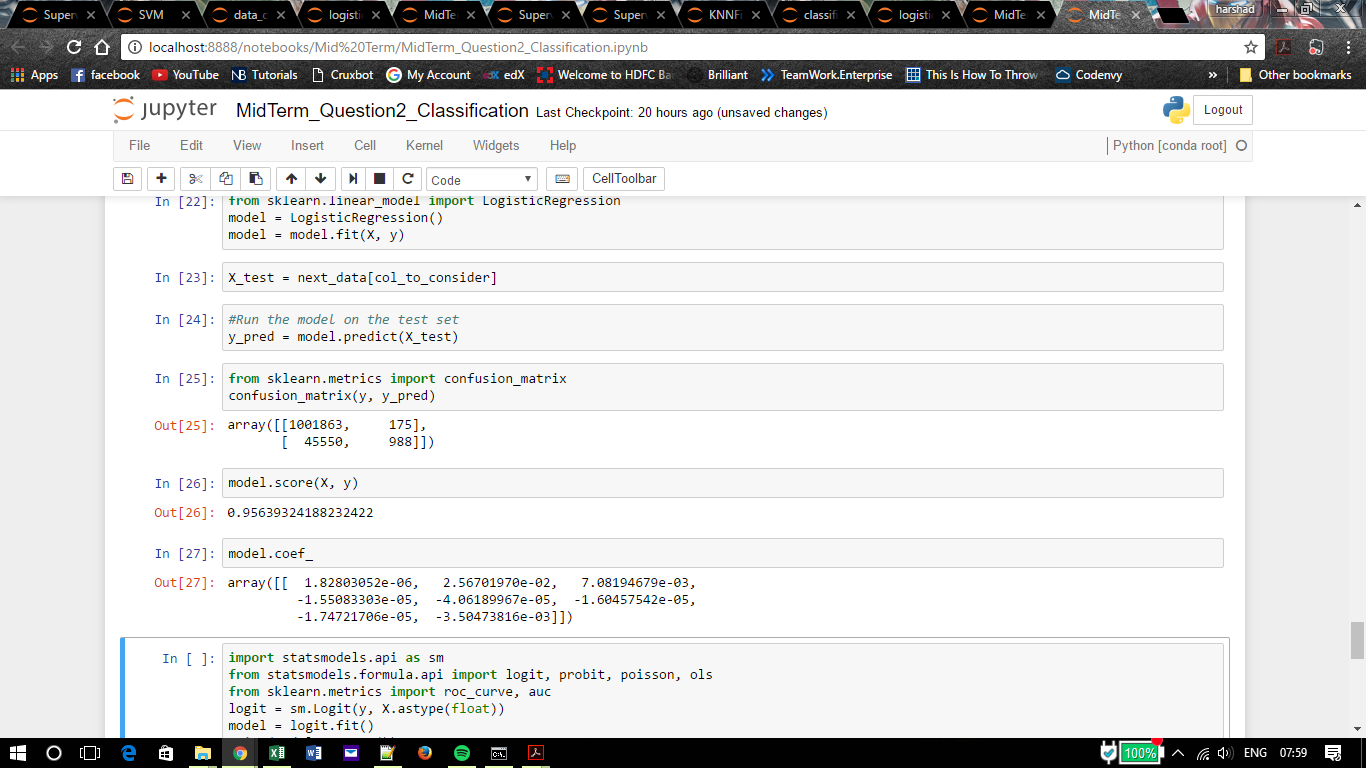
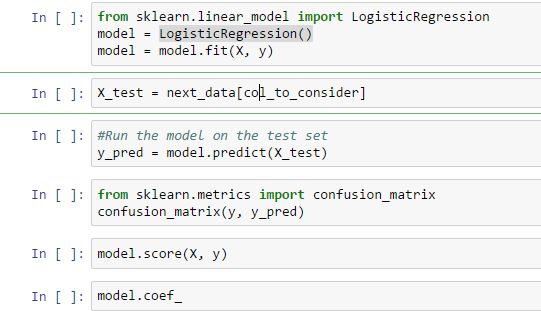
**Approach:**

In Logistic Regression we used glm() function in R to train and hence determine the output variables. In this the model is trained and then tested with any given testing data set. We performed the backward selection approach in glm() to find the best suited variables. After selecting the variable from R we did the following to train the data set

**Train the model:**

* The model is trained using LogisticRegression().fit().
* To predict the accuracy of the model, read the data for the next quarter and perform data cleaning using add\_header(), strip(), missing\_values\_table(), ConvertToNumeric1() and ConvertToNumeric2() functions.
* Select input values in X1 and actual output values in Y1.
* Again convert the actual output from 2D to 1D using np.ravel() function.
* Predict the linear model using lm.predict() for X1 that is input from next quarter

The code is given below:



**SVM:**

**Approach:**

* Generate a Support Vector Machine model using svm.SVC command.
* Add input and output to this model using the fit command.
* As this model only takes Integer values, use input.astype(X) to convert input values to integer.

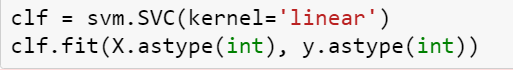
****

Figure 8:SVM

**Train the Model:**

* Train the created model using predict function.
* Calculate the performance using crosstab function and give columns from next quarter as input.

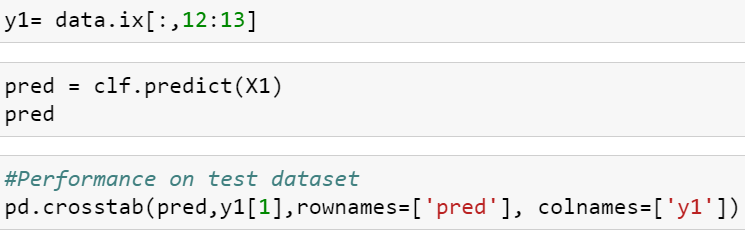
****

Figure 9:SVM

**Regression Tree**

**Approach:**

* We are a generating Random Forest by creating multiple Decission Trees.
* DecissionTrees are created using DecissionTreeRegressor().
* Random Forest is created using RandomForestRegressor().
* fit(input, output) is used to fit the according to our selected input and output dataframes.

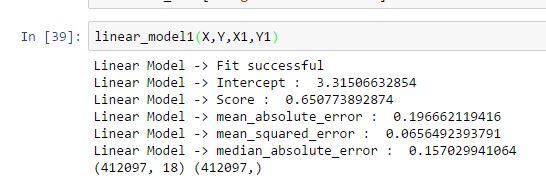
**Train the Model:**

* Once the model is created we are training the model using next quarter data.

**Neural Network**

**Approach:**

* Build the neural net using buildNetwork() command.
* This command takes number of input parameters, number of hidden nodes for every hidden layer and number of output parameters as input.
* Add every row to the sample using addSample() function



**Tableau Link:**

<https://public.tableau.com/profile/publish/Ass2P2A2/LoanTrend#!/publish-confirm>

[https://public.tableau.com/profile/publish/Ass2P2A1/Dashboard1#!/publish-confirm](https://www.google.com/url?q=https%3A%2F%2Fpublic.tableau.com%2Fprofile%2Fpublish%2FAss2P2A1%2FDashboard1%23!%2Fpublish-confirm&sa=D&sntz=1&usg=AFQjCNG6dNWDwTj5_1l93EhvN7eCX29_dg)