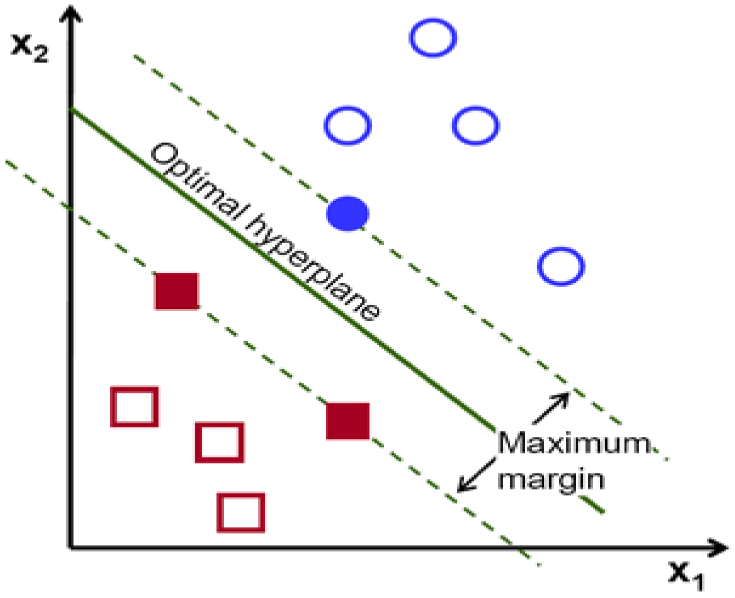
MACHINE LEARNING

Febin Zachariah – 800961027

SUPERVISED LEARNING-SUPPORT VECTOR MACHINES

INTRODUCTION

Support Vector machine can be defined as a discriminative classifier defined by a hyperplane. The main goal of support vector machine algorithm is to find the optimal separating hyperplane between the classes which maximizes the margin of the training data. It is primarily used for classification problems. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces. Following figure shows an Optimal Hyperplane which separates two classifiers.



Through this assignment, we have implemented Support vector machine algorithm in Amazon Baby Products Review and Handwritten Digits Dataset.

We are using R language for implementing this Support vector machine by using the e1071 library.

e1071 Library

It supports many functionalities like latent class analysis, fuzzy clustering, naïve Bayes classifier and Support Vector machine. We are here using e1071 library to create SVM model for our training datasets. By using the model obtained from the training data, we are predicting the classes for test data. An example of creating model is shown below:

model <- svm (V65~., data = train_data, kernel = "polynomial", type = "eps-regression", cross = 50)

Different parameters that we have used with **svm** function are described below:

- **formula**: Symbolic Representation of the model to be fit.
- **data**: training data frame on which the model is built.
- **type**: svm can be used as classification machine, regression machine or for novelty detection. Depending whether y is a factor or not, the default setting for type is C-classification or eps-regression, respectively, but may be overwritten by setting an explicit value.
  Other Possible values are:
    - C-classification
    - nu-classification
    - one-classification (for novelty detection)
    - eps-regression
    - nu-regression
- **Kernel**: Used for training and predicting. Possible values are:
    - Linear
    - Polynomial
    - Radial basis
    - Sigmoid
- **Degree**: parameter needed for kernel of type polynomial (default: 3)
- **Gamma**: This parameter is used with all kernels except linear. Gamma Parameter defines the influence of single training example with low values. Gamma parameters are the inverse of the radius of influence of samples selected by the model as vectors. When gamma is very small, the model is too constrained and cannot capture the complexity or "shape" of the data.
- **Cross**: if integer value k>0 is specified, a k-fold cross validation on the training data is performed to assess the quality of the model: the accuracy rate for classification and the Mean Squared Error for regression.
- **Cost**: cost of constraints violation (default: 1)—it is the 'C'-constant of the regularization term in the Lagrange formulation. The C parameter tells the SVM optimization how much you want to avoid misclassifying each training example. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points. For very tiny values of C, you should get misclassified examples, often even if your training data is linearly separable.

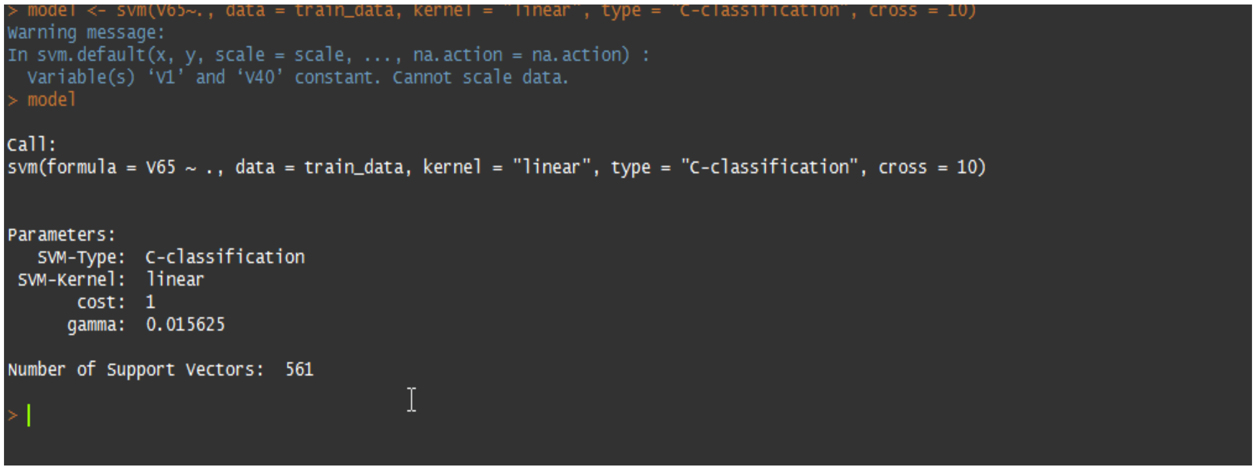
Optical Recognition of Handwritten Digits Dataset

IMPLEMENTATION

1) For the implementation of knn: **R language is used**
2) Packages used: **e1071.**
3) Load the packages by using **library** command in R if it is already installed, otherwise install the libraries by using **install. packages** command.
4) After that load the training data and testing data by using **read.csv** command.

5) The dataset for train and test data does not have a header. So, the heaters for the feature variables are names V1 to V64. The header for class variable is named V65.
6) Training data is stored in the **train_data** variable.
7) Test data is stored in the **test_data** variable.
8) By using the **svm function**, we are creating the model based on the training dataset. Following parameters are passed to the svm function
   1) Dataset 2) Kernel (linear, polynomial, sigmoid) 3) type (c classification, nu- classification)

   4)cross (number of folds for cross validation) 5) degree (used when kernel is polynomial)

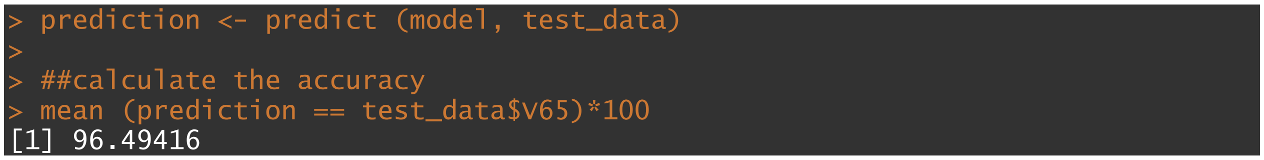


```
> model <- svm(V65~., data = train_data, kernel = "linear", type = "C-classification", cross = 10)
Warning message:
In svm.default(x, y, scale = scale, ..., na.action = na.action) :
  Variable(s) ‘V1’ and ‘V40’ constant. Cannot scale data.
> model

Call:
svm(formula = V65 ~ ., data = train_data, kernel = "linear", type = "C-classification", cross = 10)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1
      gamma:  0.015625

Number of Support Vectors:  561

> |
```

9) After the creation of the model, we are predicting the output class variable on the test data by using this model.
10) We have obtained a good accuracy with this SVM function by trying with different parameter values. Results and analysis are given in the next section.

Results and Observations

- For Cross value, we have selected 10 cross-fold validation. Accuracy of the model doesn’t change with different values of cross.
- By using the following model, we obtained an accuracy of 96.5%
  model <- svm (V65~., data = train_data, kernel = "linear", type = "C-classification", cross = 10)



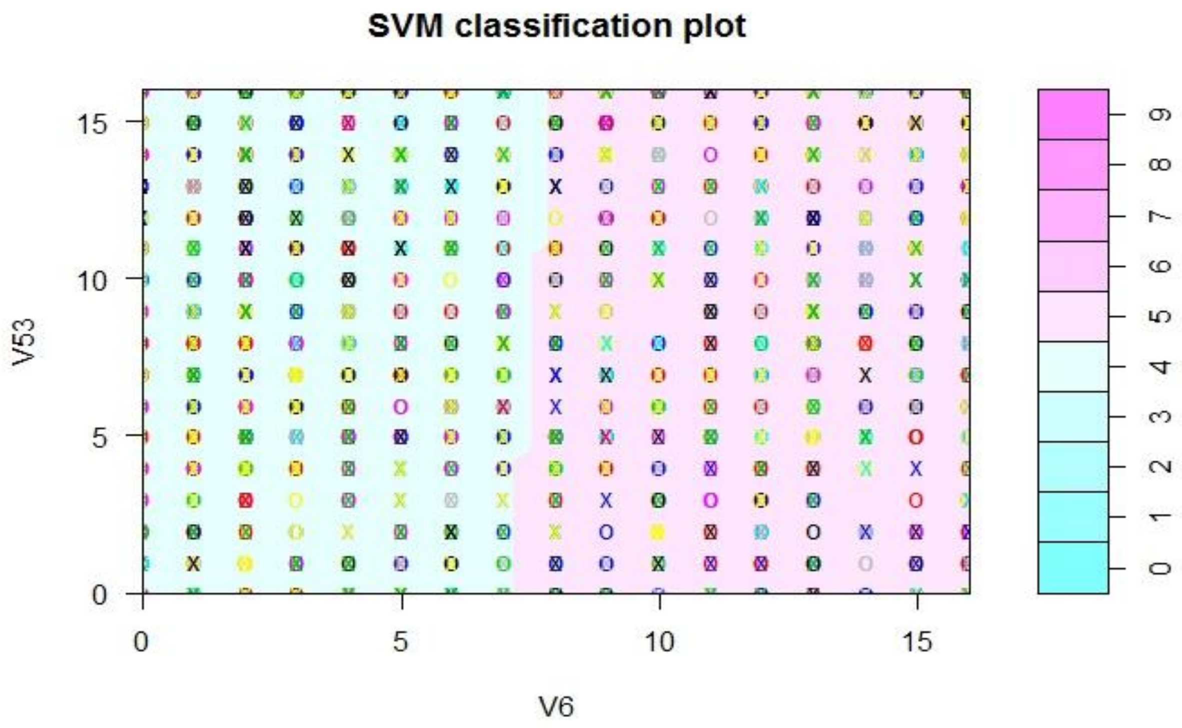
```
> prediction <- predict (model, test_data)
>
> ##calculate the accuracy
> mean (prediction == test_data$V65)*100
[1] 96.49416
```

- We have tried with different values for the parameters of SVM function to improve the accuracy of the model and we obtained the accuracy results as shown in the next table.

Type	Kernel	Cost (C)	Gamma	Accuracy
C-classification	polynomial	100	10	97.88536
C-classification	polynomial	100	100	97.88536
C-classification	polynomial	0.0001	100	97.88536
C-classification	polynomial	0.0001	1000	97.88536
C-classification	polynomial	1000	1000	97.88536
C-classification	linear	1	0.01563	96.49416
C-classification	linear	100	10	96.49416
C-classification	linear	0.1	100	96.54981
C-classification	linear	0.0001	1000	95.21425
C-classification	linear	1000	1000	96.49416
nu-classification	polynomial	1	0.01563	93.4335
nu-classification	polynomial	100	100	93.4335
nu-classification	polynomial	0.0001	1000	93.4335
nu-classification	polynomial	1000	1000	93.4335
nu-classification	linear	1	0.01563	92.65442
nu-classification	linear	100	10	92.65442
nu-classification	linear	1000	1000	92.65442
one-classification	linear	100	0.01563	10.68447
one-classification	linear	0.0001	1000	10.68447
C-classification	sigmoid	1	0.01563	10.18364
C-classification	sigmoid	100	10	10.18364
C-classification	sigmoid	0.0001	100	10.12799

- We can see that highest accuracy is obtained when Kernel is polynomial and type is C-classification. (97.88%)
- Lowest accuracy is obtained kernel is sigmoid and type is C-classification. We got accuracy of only 10.12%
- With nu-classification, we are getting an accuracy above 92% with different parameter values.
- With Type one-classification also, we are getting a very less accuracy.

- It is hard to plot a graph with the 64 features and output classification in a scatter plot.
- Therefore, we have performed correlation analysis on the dataset and found out two features which are highly correlated to the class variable.
- Highly correlated variables are: V6 and V53.
- Following graph describes the SVM classification based on the highly-correlated variables.
- Following plot is a scatter plot of the classification model by highlighting the classes and support vectors. The 'o' symbols in the plot represent the data and the 'x' symbol represent the support vectors.



Conclusions

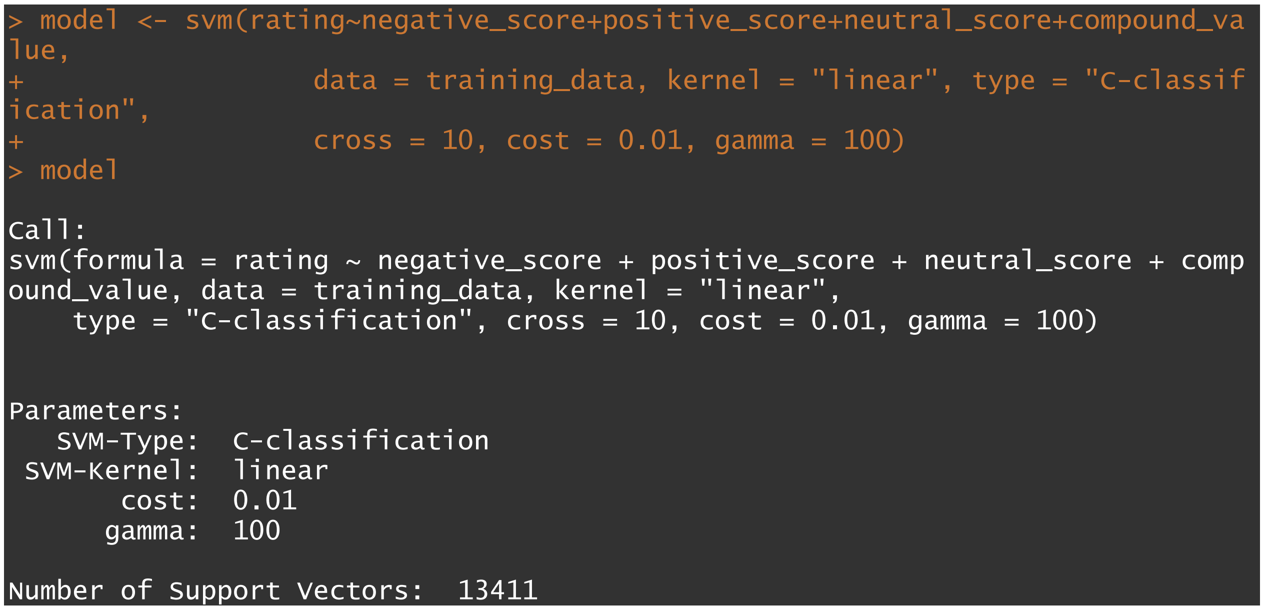
- We have obtained a maximum accuracy of 97.88% on the test data.
- We have constantly obtained an accuracy above 95% for c-classification
- We are getting more accuracy for polynomial kernel
- Successfully implemented Support Vector Machine algorithm by using e1071 library.

Amazon Reviews Sentiment Analysis Dataset

IMPLEMENTATION

1) For the implementation of knn: **R language is used**
2) Packages used: **e1071.**
3) Load the packages by using **library** command in R if it is already installed, otherwise install the libraries by using **install. packages** command.
4) Headers of the datasets: name, review, rating.
5) To improve the sentiment score accuracy, we implemented sentiment calculation by using Vader Sentiment library perform and it was giving better performance compared to RSentiment library from R.
6) The SentimentIntensityAnalyzer method takes the text review as input and output four values, namely, “pos” (positive score of the review), “neg” (negative score of the review), “neu” (neutral score of the review), “compound” (normalized value of the sum of all sentiment scores).
7) After calculating the sentiment scores of both training data and test data, sentiment scores of both the files are stored in separated files namely sentiment_train.csv and sentiment_test.csv respectively.
8) Now we are loading this files in our R script by using read.csv command.
9) By using the **svm function**, we are creating the model based on the training dataset. Following parameters are passed to the svm function
   2) Dataset 2) Kernel (linear, polynomial, sigmoid) 3) type (c classification, nu- classification)

   4)cross (number of folds for cross validation) 5) degree (used when kernel is polynomial)



```
> model <- svm(rating~negative_score+positive_score+neutral_score+compound_va
lue,
+                   data = training_data, kernel = "linear", type = "C-classif
ication",
+                   cross = 10, cost = 0.01, gamma = 100)
> model

Call:
svm(formula = rating ~ negative_score + positive_score + neutral_score + comp
ound_value, data = training_data, kernel = "linear",
    type = "C-classification", cross = 10, cost = 0.01, gamma = 100)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.01
      gamma:  100

Number of Support Vectors:  13411
```

10) After the creation of the model, we are predicting the output class variable on the test data by using this model.
11) We have obtained a good accuracy with this SVM function by trying with different parameter values. Results and analysis are given in the next section.

Results and Observations

- For Cross value, we have selected 10 cross-fold validation. Accuracy of the model doesn't change with different values of cross.
- By using the following model, we obtained an accuracy of 60.09%
  model <- svm (rating~negative_score+positive_score+neutral_score+compound_value,
  data = training_data, kernel = "linear", type = "C-classification",
  cross = 10, cost = 0.01, gamma = 100)
- We have tried with different values for the parameters of SVM function to improve the accuracy of the model and we obtained the accuracy results as shown in the following table.

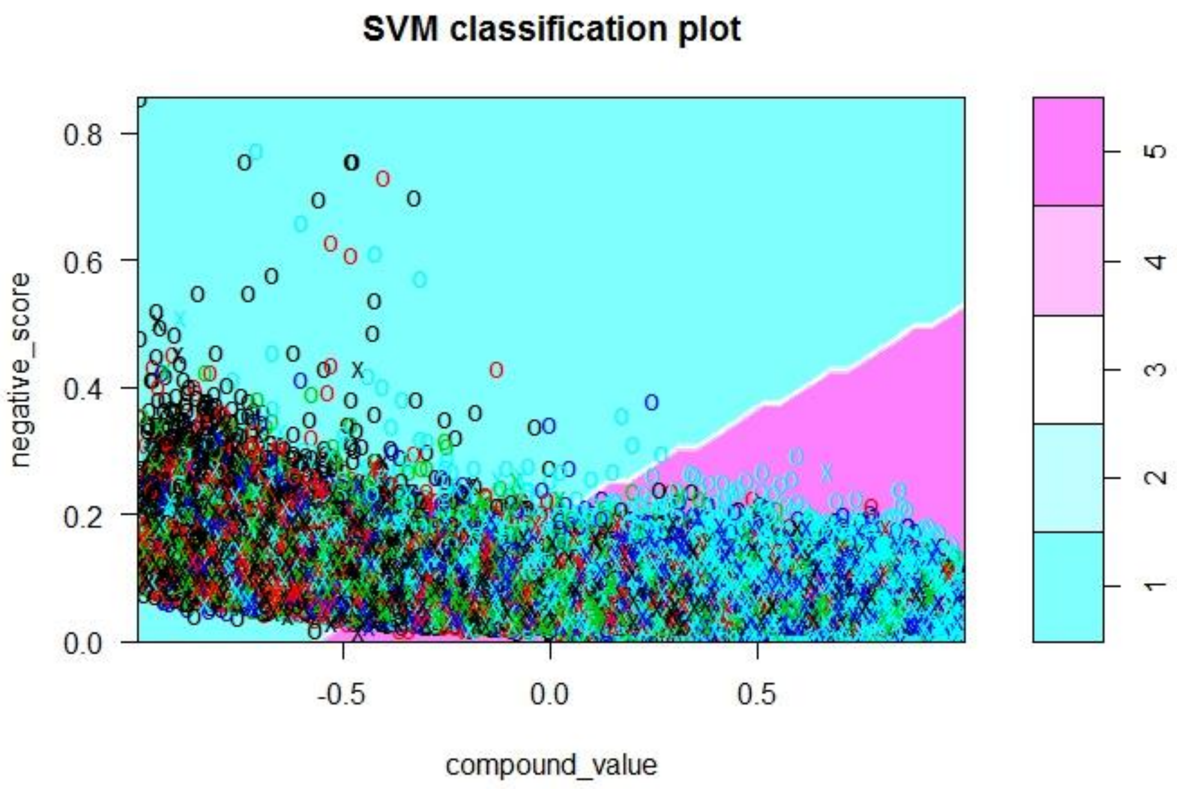
Type	Kernel	Cost (C)	Gamma	Accuracy
C-classification	linear	0.1	10	60.09208
C-classification	linear	0.01	100	60.09208
C-classification	linear	0.001	100	60.07846
C-classification	linear	1	0.25	60.07573
C-classification	polynomial	1	0.25	60.0049
C-classification	linear	0.0001	1000	57.99711
C-classification	sigmoid	1	0.25	52.25434

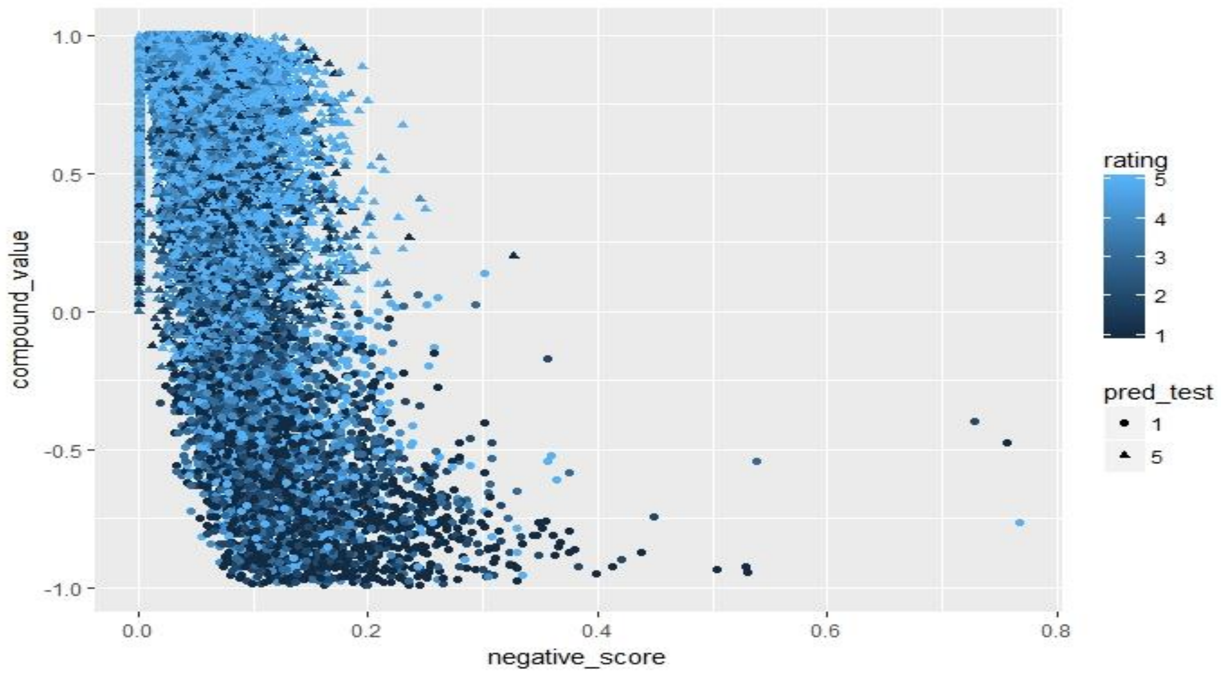
- We got highest accuracy of 60.09% with C-Classification and linear kernel
- Lowest accuracy is obtained kernel is sigmoid and type is C-classification. We got accuracy of only 52.25%
- With nu-classification, we are getting the following error (http://stackoverflow.com/questions/26987248/nu-is-infeasible)

```
Error in svm.default (x, y, scale = scale, ..., na.action = na.action) :
 specified nu is infeasible!
```

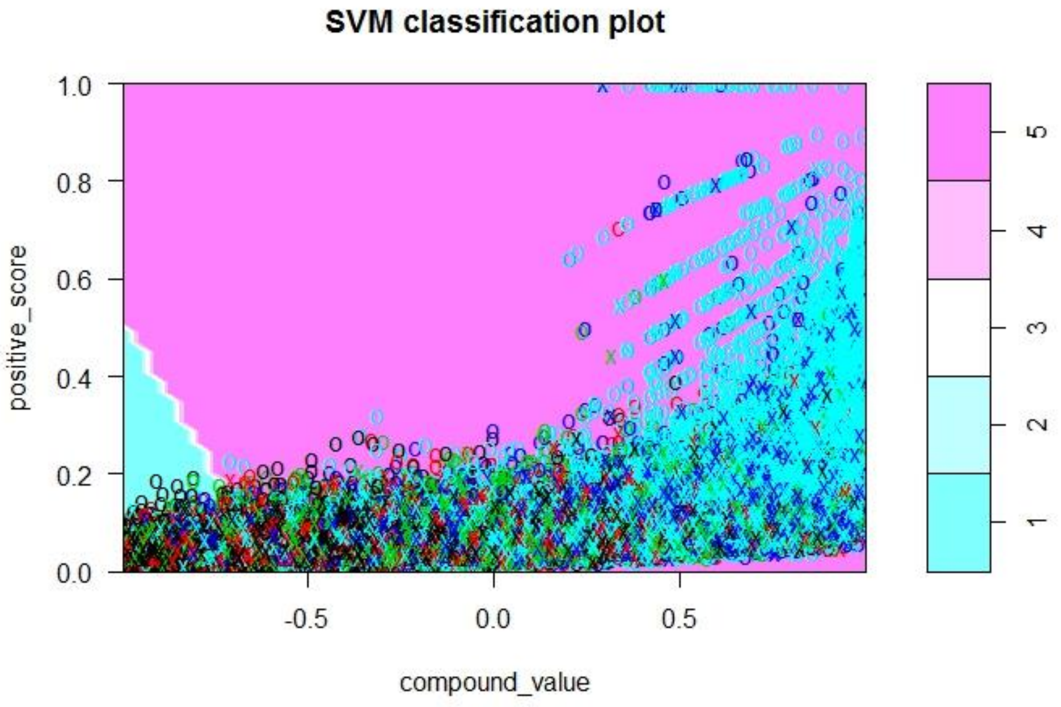
Nu-SVM is a constrained formulation of SVM (equivalent with the original up to reparameterization) which poses a hard bound on the allowed misclassification. If this bound cannot by satisfied, then the associated convex optimization problem becomes infeasible.

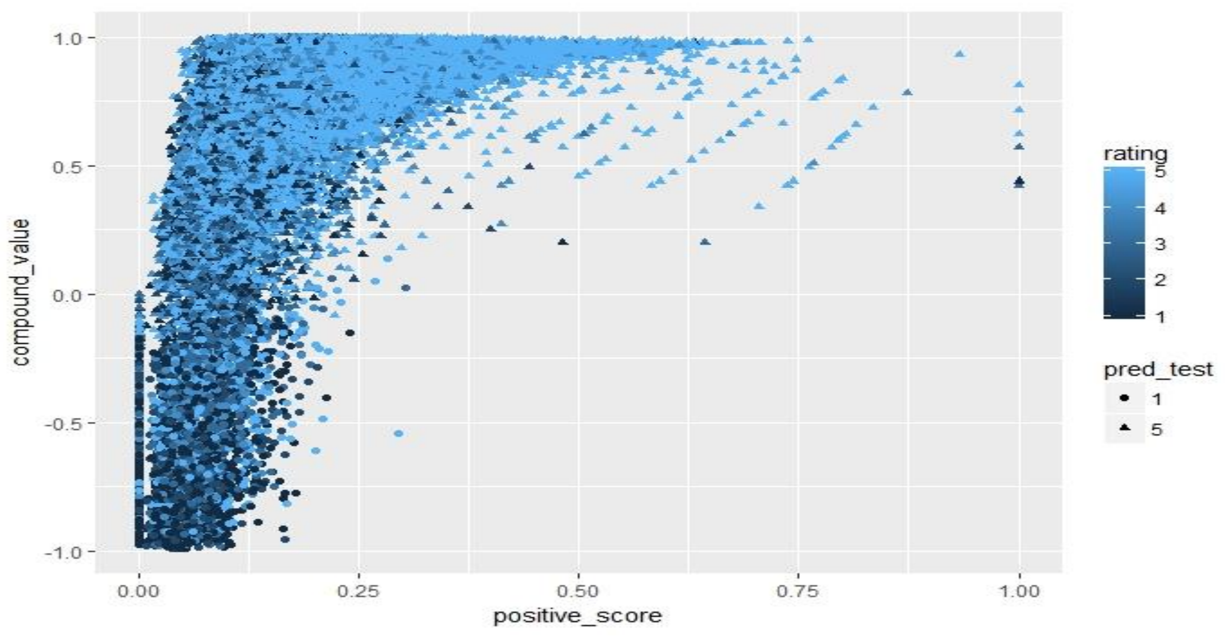
- To visualize the SVM classification with respect to features, we have uses correlation analysis to find the highly-correlated features. Highly correlated features are: negative_score, positive_score and compound_value
- Following graph describes the SVM classification based on negative_ score and compound_value.





- Following graph describes the SVM classification based on positive_ score and compound_value.





- We can see from the above figures that misclassification is happening in the case of ratings 2,3 and 4.
- Ratings 2,3 and 4 are getting classified as 5 and 1 during the prediction.

Conclusions

- We have obtained a maximum accuracy of 60.09% on the test data.
- We are getting more accuracy for linear kernel.
- Successfully implemented Support Vector Machine algorithm by using e1071 library.

Collaborated with: Ashwin Venkatesh Prabhu

References:

https://discuss.analyticsvidhya.com/t/how-to-manipulate-the-cost-function-in-svm-in-r/6029

https://cran.r-project.org/web/packages/e1071/e1071.pdf

https://en.wikipedia.org/wiki/Support_vector_machine

https://escience.rpi.edu/data/DA/svmbasic_notes.pdf

http://rischanlab.github.io/SVM.html