Alonso Salcido, Chris Haub

Lab 5, SVM, SVC

**Introduction**

This project was designed to use Support Vector Machines (SVM) and Support Vector Classification (SVC) to make predictions using a large data set. The objective was to see which model was better at making correct predictions. SVM’s are a machine learning algorithm for classification, regression and outlier detection that are particularly useful for solving binary classification problems by separating classes of data using a decision boundary.

SVC’s are a variant of SVM’s that are used in binary and multi-class classification. The goal much like SVM’s is to find a decision boundary that maximally separates two or more classes in the feature space. The algorithm attempts to find a hyperplane that separates the classes while maximizing the margin between the hyperplane and the closest data points.

**Method and Result**

For the project, one CSV files was provided. It was a rather large dataset to work with on a standard computer. The Action variable is the Y data in our dataset and what we are predicting. In reviewing the data set it is highly imbalanced with more than 50% being Allow and less than 1% on Reset. A high amount of memory was being allocated to run the data set, so the values originally stored as float 64’s were converted into int 32’s. The data was reduced from 6,291,200 bites to 4,456,304 bites.

Chart, bar chart, histogram

Description automatically generated

Next Create Dummies was completed on the following features Source Port, Destination Port, NAT Source Port, NAT Destination Port. This was done to create binary variables from the categorical variables. This allows the algorithm to treat each category as a separate feature, rather than treating the categorical variable as a single feature with multiple categories. In theory this should improve the overall performance of the model. After create dummies the data set was extremely wide. Duplicates were then checked for and removed because they did not add value. It contained 57,649 columns and was reduced to 54,958, a reduction of 2,736 columns.

Due to the size of the data, a sparse data frame was used to save space. Instead of storing every value in the data frame it only stores the non-zero values and their locations. Logistic Regression was then performed as a point of reference for SVC and SDG models. The data was broken up in Train and Test Splits (80%, 20%). The overall macro average precision was 75% and the weighted average precision was 100%. It was able to predict accurately 3 of 4. Dual=False was used to speed up the training of the data set. The main issue with this model was predicting the smaller class. Model did not predict any observations into Reset-Both class.

**Logistic Regression**

Logistic Regression confusion matrix accurately predicted the Allow variables 7,520 and only 8 variables were incorrectly classified. For the Drop variable all variables were correctly predicted. The Deny variable 2,991 were correctly predicted and 7 incorrect.

Run time 17 seconds

A picture containing graphical user interface

Description automatically generated

Table

Description automatically generated

**Linear Support Vector Classification**

Before Linear Support Vector Classification was run Non-boolean features were scaled using StandardScaler. This method scales the data to have a mean of zero and standard deviation of one. Different parameters were tested on a smaller Train Test split. After the best parameters were found they were used in an 80/20 split. The class weights were then balanced. Macro average precision was 38% and weighted average precision was 70%.

Run time 12 minutes

Chart

Description automatically generated

Table

Description automatically generated

**Non-Linear Support Vector Classification**

After running a linear SVC a non-linear support vector classification was used. Radial Bases Function (RBF) was used as the kernel function. The RBF is a nonlinear kernel that can map the input data into a high-dimensional space, allowing SVM to find a nonlinear decision boundary in the transformed space. The RBF kernel measures the similarity between two data points based on their distance in the feature space. The macro average precision was 96%, weighted average precision was 100%

Run time 1 hour, 1 minute

Chart

Description automatically generated with low confidence

Table

Description automatically generated

**Stochastic Gradient Descent**

The last algorithm used was Stochastic Gradient Descent (SGD). The SGD classifier was run with the full scaled sparse data frame. Resulting in a macro average precision of 77%, weighted average precision of 99%. Then, the same classifier was used with multiple epochs and the data shuffled between each epoch. Shuffling the training data during each epoch can help to prevent the model from overfitting to the training data, as it ensures that the model is exposed to a diverse range of training examples. We used a total of 5 epochs. Macro average precision was 79%, weighted average precision was 100%.

Run time was less than one second.

A picture containing chart

Description automatically generated Table

Description automatically generated

**Conclusion**

The Linear Support Vector Classification was the weakest model we ran. This was due to it attempting to slice the data by lines. It was only able to predict two of the four classes. While being slower than other methods.

Both the non-linear Support Vector Classification and Stochastic Gradient Descent had similar results that were in the high nineties. Either model would be a good model to use. The highest advantage of SGD is run time, while it took over an hour for non-linear SVC to run. SGD was able to get the same predictions as non-linear SVC in less than one second.

**Appendix**

**Text

Description automatically generated**

**Text

Description automatically generated** **Text

Description automatically generated** Text

Description automatically generated Text

Description automatically generated Text

Description automatically generated Text

Description automatically generated Text

Description automatically generated Text

Description automatically generated Text

Description automatically generated Text

Description automatically generated A screenshot of a computer

Description automatically generated with medium confidence