**SVM**

Required Python libraries: Pandas, Numpy, Scikit-learn, Matplotlib

You will also need to have software installed to run and execute the Jupyter notebook.

Template code is provided in the 4A-SVM.ipynb notebook file.

**DATA LOADING AND PREPROCESSING**

First, training data that is stored as .csv is loaded using pandas. Next, labels are stored as a column vector of shape (100, 1). 100 being the number of training samples and 1 is the label for each sample that is either 0 or 1.

0 for Simple Substitution and 1 for Vigenere.

After that, training features are created using CountVectorizer function of Scikit-learn that converts the training feature into a feature set of 1882 features. We also transform our test data using CountVectorizer.

Shape of training features: (100, 1882)

Shape of training labels: (100,)

Shape of validation features: (20, 1882)

Shape of validation labels: (20,)

**CREATING MODEL**

Deciding which kernel to use:

1. First we make multiple classifers using SVC from sklearn.svm, we make a classifier for each kernel. (“rbf”, “linear”, “poly”, “sigmoid”).
2. We plotted the Training score and Cross validation scores of all the classifiers on our training data.
3. While cross validating, we use 3 re-shuffling and splitting iterations and test size of 0.2
4. plot\_learning\_curve(estimator, title, X, y, ylim=None, cv=None, n\_jobs=1, train\_sizes=np.linspace(.1, 1.0, 5)) function is used to plot the graphs.

Parameters of plot\_learning\_curve()

* estimator : object type that implements the "fit" and "predict" methods.

An object of that type which is cloned for each validation.

* title : string

Title for the plot.

* X : array-like, shape (n\_samples, n\_features)

Training vector, where n\_samples is the number of samples and n\_features is the number of features.

* y : array-like, shape (n\_samples) or (n\_samples, n\_features), optional

Target relative to X for classification or regression;

* ylim : tuple, shape (ymin, ymax), optional

Defines minimum and maximum yvalues plotted.

* cv : int, cross-validation generator or an iterable, optional

We input the cross-validation object we want to use

* n\_jobs : integer, optional

Number of jobs to run in parallel (default 1).

**MAKING PREDICTIONS AND EVALUATION**

After plotting the graphs for different kernels we can easily observe that Linear kernel performs the best. So we fit our data in the SVC linear kernel classifer and make our predictions on the test\_data.

1. We fit our training data using clf\_linear classifer, which is essentially a SVC classifier having linear kernel.

fit(train, target)

* train : It is the training data or feature set we are using
* target : It is the label to the corresponding feature set or training data

1. We make predictions on the test\_data by calling the predict function on our linear classifier.

predict(test\_data)

* test\_data : Feature set for which we are making the predictions

We evaluate our predictions using classification report and confusion matrix. We just pass our predictions and the expected value of the test\_data and we get the confusion matrix and classification report.