Support Vector Classification

Ian R. Stewart

December 8th, 2018

Fall 2018 – COSC 528

Project 5

Abstract

This project performs a classification using a prebuilt support vector classifier (SVC) in Python to classify data for three problem sets: (1) radar signals of Earth's ionosphere to classify the structure, (2) speech data for vowel sounds from human participants, and (3) spectral data from satellite images to classify land type. The three problems contain unique number of instances and classes with no absent data located after each dataset was explored. The SVC was initially performed using a 40%-60% training-set split for the first two problems and the predefined test set for the final problem. This initial simulation of the SVC Python class employed the default values of penalty value (C) of 1.0, Radial Basis Function kernel, and a gamma value equal to the inverse of the product of the number of features and the data standard deviation. This initial SVC classification resulted in a test accuracy 88.15%, 84.51%, and 88.95% for the ionosphere, vowel sounds, and satellite imagery problems, respectively. A grid search cross validation was implemented to optimize the hyperparameters for the SVC using five cross validation splits. For each coarse grid search, the Radial Basis Function kernel was found to be optimal. A fine grid search was then performed on the penalty and gamma value to increase the accuracy of the model on the test data. Using the optimal hyperparameters obtained from the fine grid search, the test set accuracies from the SVC were 91.47%, 95.28%, and 91.5% for the three problems, respectively.

Table of Contents

List of Tables

List of Figures

1 Introduction

This project applies a support vector classifier (SVC) to three unique datasets using a prebuilt Python SVC library[1]. The three classification problems are as follows:

1. binary classification on ionosphere dataset using radar signals,
2. eleven-class classification on audible vowel sounds, and
3. seven-class classification on multispectral value from satellite images.

A brief overview of the data for the three problems is provided in the following subsections.

1.1. Ionosphere Dataset

The ionosphere dataset was collected in Goose Bay, Labrador, Canada in 1989 via an array of 16 high-frequency antennas. The antennas measure the free electrons in the Earth's ionosphere and classifies the data array as either '*g*' for good of '*b*' for bad, which correspond to the whether the data shows evidence for some type of structure (good) or no structure (bad). The dataset contains 34 features and 351 instances containing integer, real values. The last column (35th feature) contains the binary classification data as a string.

1.2. Vowel Sounds Dataset

The vowel sounds dataset was collected in the Cambridge, England and contains data for eleven steady state vowels sounds (classes) and 10 data features. Fifteen participants were used to create the dataset with each participant saying each vowel 6 times. Three features are provided along with the vowel sound data: *Train or Test*, *Speaker Number*, and *Sex*. This results in 14 total data features, including the classification, and 990 instances.

1.3 Satellite Imagery Dataset

The satellite imagery dataset was collected in Glasgow, Scotland consisting of multi-spectral values of pixels in (3 x 3) neighborhoods in an 82 x 100 sub-area of a satellite image. The classification associated with the data corresponds to the central pixel in each neighborhood. The classes in the dataset correspond to the one of eight classification of the images: (1) red soil, (2) cotton crop, (3) grey soil, (4) damp grey soil, (5) soil with vegetation stubble, (6) mixture class (none present), and (7) very damp grey soil. In total, 36 features comprise the dataset with 4435 and 2000 instances in the presplit training and test datasets, respectively.

1.4 Overview of Analysis Techniques

Support vector machines (SVM) are a powerful and widely implemented learning algorithm, which can be thought of as an extension of the perceptron. Using a perceptron for regression or classification purposes, the error (e.g. misclassification error, Gini coefficient, or entropy) is minimized, where an SVM is optimized by maximizing the

[1] The scikit-learn Python package contains support vector machine (SVM) library that is based on the popular, open source LIBSVM library. For further information regarding the class and the syntax, the reader can review the documentation found at: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html.

margin. The margin in this case is defined as the distance between the decision boundary and the training values closest to the boundary (i.e. these values are referred to as the support vectors). The SVM has the ability to be kernelized to solve nonlinear problems, which increases the algorithm's popularity among practitioners. The basic idea behind the kernel methods implemented in SVMs is to create nonlinear combinations of the original data features in order to obtain a decision boundary for linearly inseparable data (i.e. project the data into a higher dimensional space where the data is linearly separable).

A potential drawback to the projecting or mapping the data into a higher dimensional space with new features can increase the computation time required to solve the regression or classification. To solve this issue, a task referred to as the kernel-trick is performed. The kernel trick removes an expensive dot product computation of the original data with an inner product of the mapping function. To accomplish this, several kernel functions (i.e. can be thought of as a similarity function between a pair of samples) have been created with the most widely used kernel being the Radial Basis Function. This kernel is rarely referred to as a Gaussian kernel and minus sign inside the exponential inverts the distance measure into a score value ranging between 1 (similar) and 0 (dissimilar).

$$k(x^i, y^j) = \exp\left(-\frac{\left\|x^i - x^j\right\|^2}{2\sigma^2}\right) = \exp\left(-\gamma \left\|x^i - x^j\right\|^2\right)$$

Figure 1: Equation for the Radial Basis Function kernel.
Where, γ is a free parameter to be optimized equal to $(2\sigma^2)^{-1}$.

Other common kernels are the Linear kernel, Polynomial kernel, and the Sigmoid kernel. The kernels function for these kernels are as follows.

$$k(x, y) = x^T y + c$$

Figure 2: Equation for Linear kernel.
Where, c is a free parameter to be optimized.

$$k(x, y) = (\alpha x^T y + c)^d$$

Figure 3: Equation for Polynomial kernel.
Where, $\alpha, c,$ and d are a free parameter to be optimized.

$$k(x, y) = \tanh(\alpha x^T y + c)$$

Figure 4: Equation for Sigmoid kernel.
Where, α and c are a free parameter to be optimized.

In order to obtain optimal parameters for each kernel, a grid search is used exhaustively search over specified parameter values. The parameters of the estimator are further optimized by cross-validating the grid search over a parameter grid. For example, if the Radial Basis Function is picked as the kernel function for the SVC, the cross validation can optimize the error term and the γ parameters. Within Python, this is accomplished via the *sklearn.model_selection* library's *GridSearchCV*( ) function. This is the function that will be implemented in this project to first find optimal values via a coarse search and subsequently a fine search.

2 Data Exploration

The datasets for each of the three problems in this project are explored independently in the following subsections.

2.1 Ionosphere Dataset

The ionosphere dataset contained 35 total features, where the final feature contains the classification data for the data instances. The classification data only contained two classes (binary classification) corresponding to *good* or *bad* signal. The dataset features, minus the classification column, contained two columns with integer values. The first and second columns of the dataset contain two and one unique value, respectively. The first feature contains values of either 0 or 1 and the second feature contains all zeros. The number of unique values per data feature is provided in Figure 5.

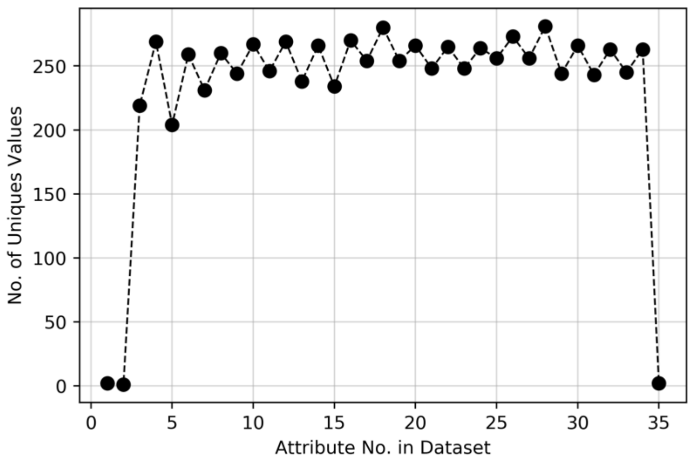


Figure 5: Number of unique values in ionosphere data.

The data was checked for missing or NaN (not a number) values resulting in all values accounted for in the dataset. The data was then standardized between -1 and 1 in order to assist the SVC process.

2.2. Vowel Sound Dataset

The vowel dataset contained a total of 14 features, including the classification column, and 990 instances. The number of unique values in each data feature was analyzed to check for any trends in the data with results provided in Figure 6 and Table 1.

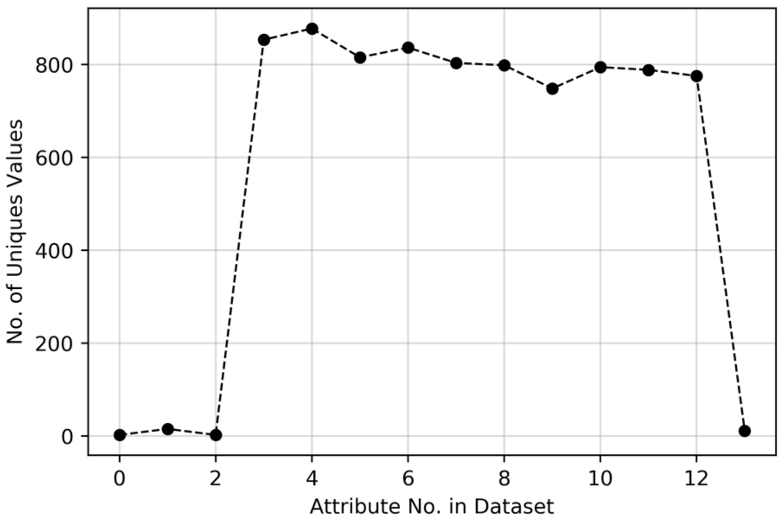


Figure 6: Number of unique values in vowel sound data.

Table 1: Number of unique values in vowel sound data.

Feature	No. of Unique
Train or Test	2
Speaker Number	15
Sex	2
Feature 0	853
Feature 1	877
Feature 2	815
Feature 3	836
Feature 4	803
Feature 5	798
Feature 6	748
Feature 7	794
Feature 8	788
Feature 9	775
Class	11

The first three data features (*Train or Test*, *Speaker Number*, *Sex*) were not used in the data analysis portion, as each deemed irrelevant, leaving 10 data features used for the data analysis. The remaining data features were standardized, resulting in all float data types for the data features, except for the classification column. The data was checked for missing or NaN (not a number) values resulting in all values accounted for in the dataset. The data was then standardized between -1 and 1 in order to assist the SVC process.

2.3 Satellite Imagery Dataset

Two datasets were provided for the satellite imagery dataset: one specifically for training with 4435 instances and the other for a test set containing 2000 instances. Both datasets provided contained 36 features, all of which containing integer values with no missing or NaN values. Seven classes are employed in the dataset with no values for one of the classes, resulting only six classes actually represented in the dataset. The number of unique values for each data feature was analyzed with the results provided in .

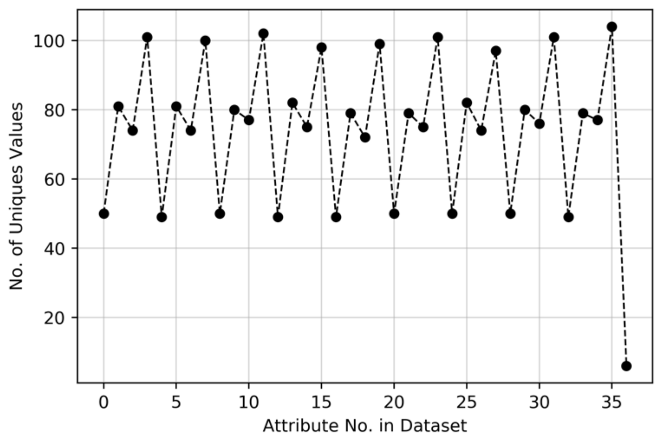


Figure 7: Number of unique values in satellite data.

Table 2: Number of unique values in vowel sound data.

Feature	No. of Unique
Feature 1	50
Feature 2	81
Feature 3	74
Feature 4	101
Feature 5	49
Feature 6	81
Feature 7	74
Feature 8	100
Feature 9	50
Feature 10	80
Feature 11	77
Feature 12	102
Feature 13	49
Feature 14	82
Feature 15	75
Feature 16	98
Feature 17	49
Feature 18	79
Feature 19	72
Feature 20	99
Feature 21	50
Feature 22	79
Feature 23	75
Feature 24	101
Feature 25	50
Feature 26	82
Feature 27	74
Feature 28	97
Feature 29	50
Feature 30	80
Feature 31	76
Feature 32	101
Feature 33	49
Feature 34	79
Feature 35	77
Feature 36	104
Class	6

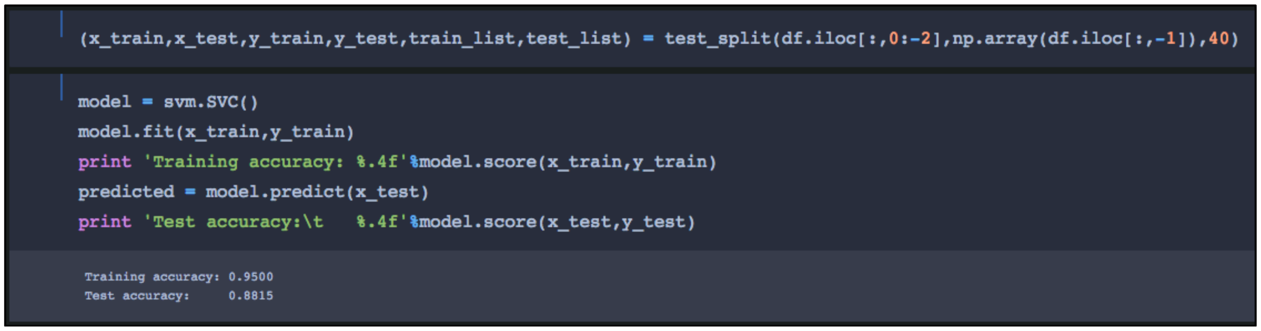
No data features can be dropped from the unique value analysis, as all are relevant and no single valued features. The data was checked for missing or NaN (not a number) values resulting in all values accounted for in the dataset. The data was then standardized between -1 and 1 in order to assist the SVC process.

3 Data Analysis

Similar to that of the previous section, each problem is analyzed independently in the following subsections providing the optimal parameters to provide the highest level of accuracy on the test (validation) dataset. The confusion matrix and other performance metrics are provided where applicable.

3.1. Ionosphere Dataset

The ionosphere dataset was randomly 40%-60% split into a training set and test set using a created *test_split*( ) function. Implementing the *SVC*( ) function with default inputs in Python's *sklearn.svm* library, the accuracy on the training set and test set resulted in 95.0% and 88.15%, respectively. The *SVC*( ) function defaults to a Radial Basis Function kernel, a penalty parameter (C) of 1.0, and gamma parameter calculated as the inverse of (no. of features × data standard deviation).



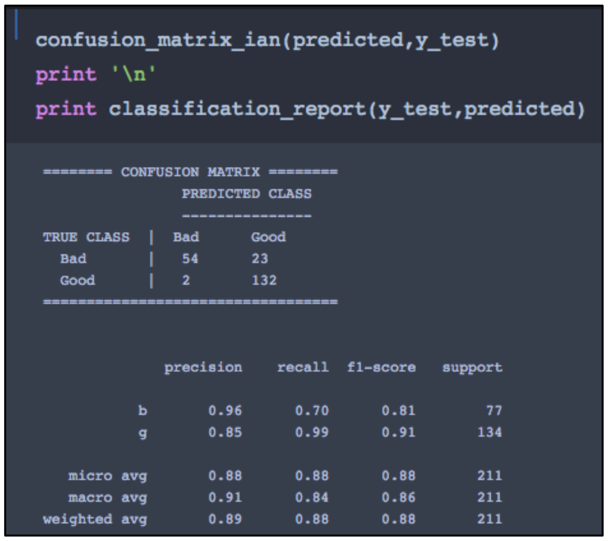
```
(x_train,x_test,y_train,y_test,train_list,test_list) = test_split(df.iloc[:,0:-2],np.array(df.iloc[:,-1]),40)
```

```
model = svm.SVC()
model.fit(x_train,y_train)
print 'Training accuracy: %.4f'%model.score(x_train,y_train)
predicted = model.predict(x_test)
print 'Test accuracy:\t   %.4f'%model.score(x_test,y_test)
```

```
Training accuracy: 0.9500
Test accuracy:     0.8815
```

Figure 8: Screenshot of data split and SVC accuracy for ionosphere dataset.

As this is a binary classification, a previously created function to create a confusion matrix was implemented on the predicted values along with *sklearn.metrics* library *classification_report*( ) function output. Using this training-test split, 25 misclassifications were calculated of the test set, resulting in the 88.15% accuracy (i.e. 25/211*100 results in 11.85% misclassification error). Important to note, since the dataset is randomly split into a training and test set, the accuracy will change depending on which instances are used for the two datasets.

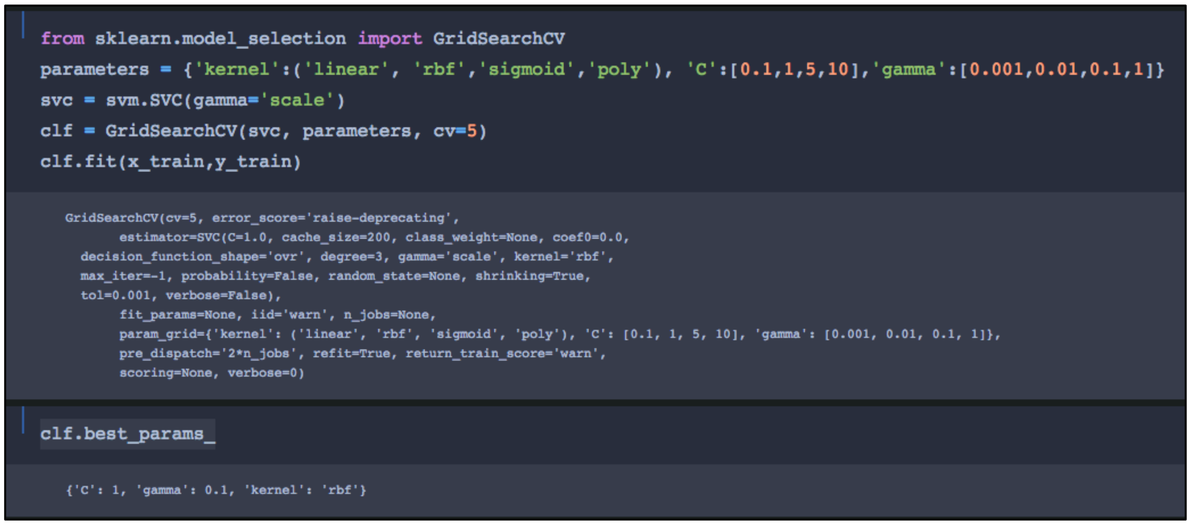


```
confusion_matrix_ian(predicted,y_test)
print '\n'
print classification_report(y_test,predicted)
```

```
======== CONFUSION MATRIX ========
                   PREDICTED CLASS
                   ---------------
TRUE CLASS  |  Bad       Good
  Bad       |   54       23
  Good      |   2        132
==================================


              precision    recall  f1-score   support

           b       0.96      0.70      0.81        77
           g       0.85      0.99      0.91       134

   micro avg       0.88      0.88      0.88       211
   macro avg       0.91      0.84      0.86       211
weighted avg       0.89      0.88      0.88       211
```

Figure 9: Confusion matrix and classification report using default SVC for ionosphere dataset.

A grid search was subsequently performed to obtain assist in parameter optimization of the SVC. Initially, coarse grid was implemented using a penalty values (0.1, 1, 5, 10), gamma values of (0.001, 0.01, 0.1, 1), kernel functions (linear, radial basis function, sigmoid, polynomial) five cross validations. This resulted in a penalty value of 1.0, gamma value of 0.1 and Radial Basis Function kernel.



```
from sklearn.model_selection import GridSearchCV
parameters = {'kernel':('linear', 'rbf','sigmoid','poly'), 'C':[0.1,1,5,10],'gamma':[0.001,0.01,0.1,1]}
svc = svm.SVC(gamma='scale')
clf = GridSearchCV(svc, parameters, cv=5)
clf.fit(x_train,y_train)
```

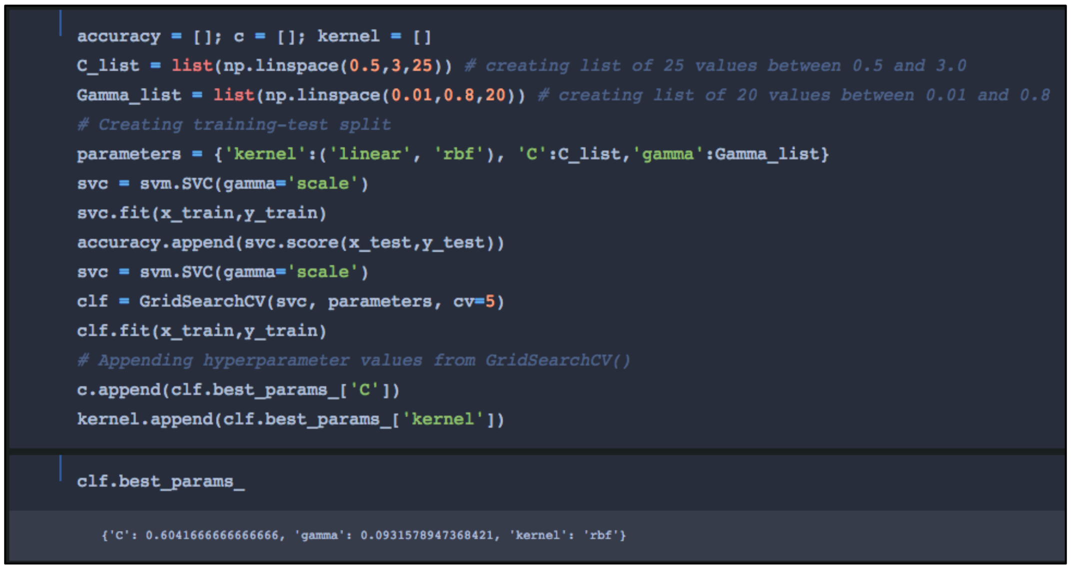
```
GridSearchCV(cv=5, error_score='raise-deprecating',
       estimator=SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False),
       fit_params=None, iid='warn', n_jobs=None,
       param_grid={'kernel': ('linear', 'rbf', 'sigmoid', 'poly'), 'C': [0.1, 1, 5, 10], 'gamma': [0.001, 0.01, 0.1, 1]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=0)
```

```
clf.best_params_
```

```
{'C': 1, 'gamma': 0.1, 'kernel': 'rbf'}
```

Figure 10: Screenshot of coarse grid search for ionosphere dataset.

A penalty value of 1.0 was chosen for the optimal kernel function Radial Basis Function using a gamma value of 0.1. A fine grid search was then performed using the following ranges of values for the Radial Basis Function: C-value [0.5 to 3 in 25 steps] and Gamma-value [0.01 to 0.8 in 20 steps].



```
accuracy = []; c = []; kernel = []
C_list = list(np.linspace(0.5,3,25)) # creating list of 25 values between 0.5 and 3.0
Gamma_list = list(np.linspace(0.01,0.8,20)) # creating list of 20 values between 0.01 and 0.8
# Creating training-test split
parameters = {'kernel':('linear', 'rbf'), 'C':C_list,'gamma':Gamma_list}
svc = svm.SVC(gamma='scale')
svc.fit(x_train,y_train)
accuracy.append(svc.score(x_test,y_test))
svc = svm.SVC(gamma='scale')
clf = GridSearchCV(svc, parameters, cv=5)
clf.fit(x_train,y_train)
# Appending hyperparameter values from GridSearchCV()
c.append(clf.best_params_['C'])
kernel.append(clf.best_params_['kernel'])
```

```
clf.best_params_
```

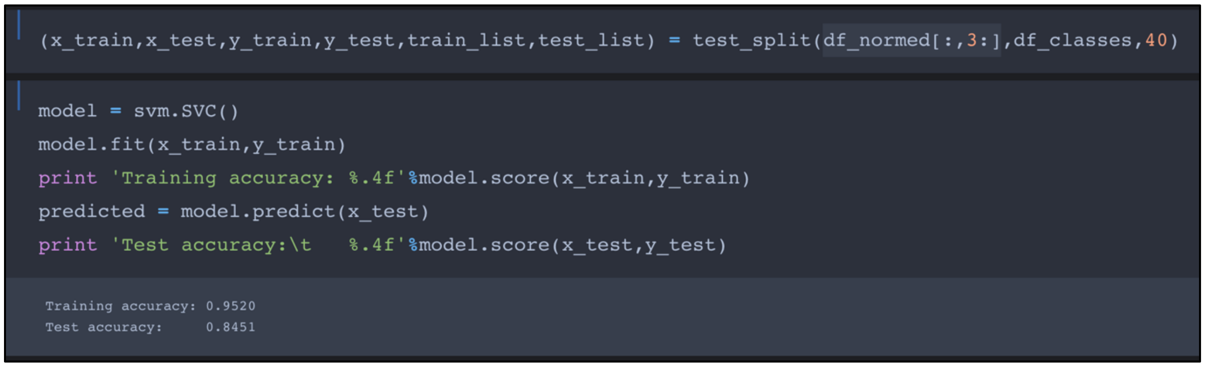
```
{'C': 0.6041666666666666, 'gamma': 0.0931578947368421, 'kernel': 'rbf'}
```

Figure 11: Screenshot of fine grid search for ionosphere data.

For this training-test split, the fine grid search resulted in the following values for the hyperparameters for five cross validations: C-value of 0.604 and Gamma-value of 0.093 for the Radial Basis Function. Using these values, *SVC*( ) was performed on the test set and resulted in 91.47% accuracy; a 3.32% accuracy increase from the default run.

3.2 Vowel Sound Dataset

Similar to the ionosphere dataset, the vowel sound dataset was randomly 40%-60% split into a training set and test set using a created *test_split*( ) function. Implementing the *SVC*( ) function with default inputs in Python's *sklearn.svm* library, the accuracy on the training set and test set resulted in 95.2% and 84.51%, respectively. The *SVC*( ) function defaults to a Radial Basis Function kernel, a penalty parameter (C) of 1.0, and gamma parameter calculated as the inverse of (no. of features × data standard deviation).



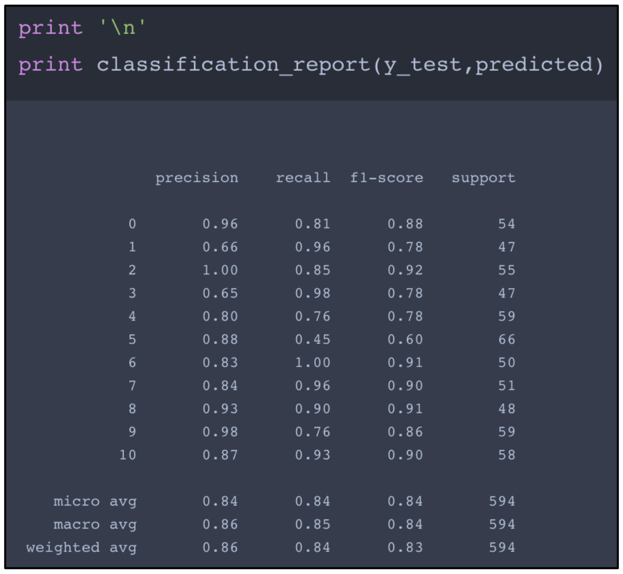
```
(x_train,x_test,y_train,y_test,train_list,test_list) = test_split(df_normed[:,3:],df_classes,40)
```

```
model = svm.SVC()
model.fit(x_train,y_train)
print 'Training accuracy: %.4f'%model.score(x_train,y_train)
predicted = model.predict(x_test)
print 'Test accuracy:\t   %.4f'%model.score(x_test,y_test)
```

```
Training accuracy: 0.9520
Test accuracy:     0.8451
```

Figure 12: Screenshot of data split and SVC accuracy for vowel sound dataset.

Using the *sklearn.metrics* library *classification_report*( ) function, the following Figure 13 shows the function output for the fit performance metrics.

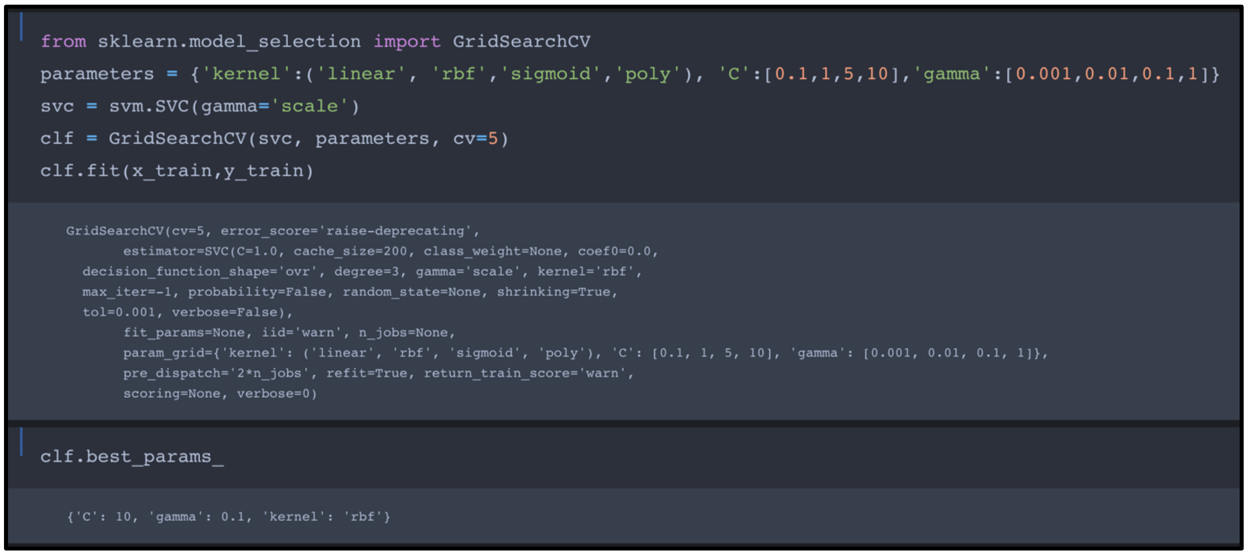


```
print '\n'
print classification_report(y_test,predicted)
```

```
             precision    recall  f1-score   support

          0       0.96      0.81      0.88        54
          1       0.66      0.96      0.78        47
          2       1.00      0.85      0.92        55
          3       0.65      0.98      0.78        47
          4       0.80      0.76      0.78        59
          5       0.88      0.45      0.60        66
          6       0.83      1.00      0.91        50
          7       0.84      0.96      0.90        51
          8       0.93      0.90      0.91        48
          9       0.98      0.76      0.86        59
         10       0.87      0.93      0.90        58

  micro avg       0.84      0.84      0.84       594
  macro avg       0.86      0.85      0.84       594
weighted avg       0.86      0.84      0.83       594
```

Figure 13: Classification report using default SVC for vowel sound dataset.

Even though accuracy values of 95.2% and 84.51% for the training and test sets are relatively high, the accuracy can potentially be improved by optimizing the hyperparameters via a grid search using cross validation. Identical parameters as the ionosphere were used for the hyperparameters: penalty values (0.1, 1, 5, 10), gamma values of (0.001, 0.01, 0.1, 1), kernel functions (linear, radial basis function, sigmoid, polynomial) and five cross validations. This resulted in a penalty value of 10, gamma value of 0.1 and Radial Basis Function kernel.



```
from sklearn.model_selection import GridSearchCV
parameters = {'kernel':('linear', 'rbf','sigmoid','poly'), 'C':[0.1,1,5,10],'gamma':[0.001,0.01,0.1,1]}
svc = svm.SVC(gamma='scale')
clf = GridSearchCV(svc, parameters, cv=5)
clf.fit(x_train,y_train)
```

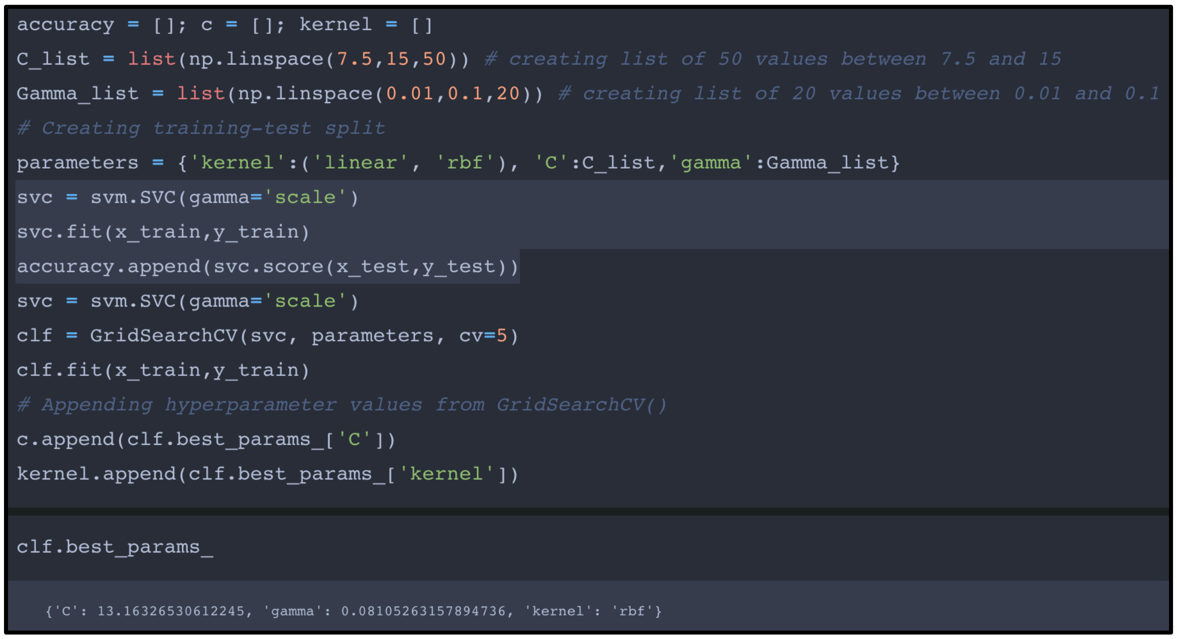
```
GridSearchCV(cv=5, error_score='raise-deprecating',
       estimator=SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False),
       fit_params=None, iid='warn', n_jobs=None,
       param_grid={'kernel': ('linear', 'rbf', 'sigmoid', 'poly'), 'C': [0.1, 1, 5, 10], 'gamma': [0.001, 0.01, 0.1, 1]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=0)
```

```
clf.best_params_
```

```
{'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}
```

Figure 14: Screenshot of coarse grid search for vowel sound dataset.

A fine grid search was then performed using the following ranges of values for the Radial Basis Function: C value [7.5 to 15 in 50 steps] and Gamma value [0.01 to 0.1 in 20 steps]. The results from the fine grid are provided in .

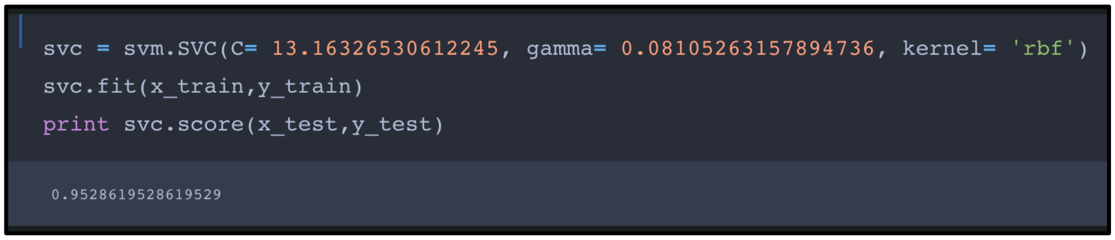


```
accuracy = []; c = []; kernel = []
C_list = list(np.linspace(7.5,15,50)) # creating list of 50 values between 7.5 and 15
Gamma_list = list(np.linspace(0.01,0.1,20)) # creating list of 20 values between 0.01 and 0.1
# Creating training-test split
parameters = {'kernel':('linear', 'rbf'), 'C':C_list,'gamma':Gamma_list}
svc = svm.SVC(gamma='scale')
svc.fit(x_train,y_train)
accuracy.append(svc.score(x_test,y_test))
svc = svm.SVC(gamma='scale')
clf = GridSearchCV(svc, parameters, cv=5)
clf.fit(x_train,y_train)
# Appending hyperparameter values from GridSearchCV()
c.append(clf.best_params_['C'])
kernel.append(clf.best_params_['kernel'])
```

```
clf.best_params_
```

```
{'C': 13.16326530612245, 'gamma': 0.08105263157894736, 'kernel': 'rbf'}
```

Figure 15: Screenshot of fine grid search for vowel sound data.



```
svc = svm.SVC(C= 13.16326530612245, gamma= 0.08105263157894736, kernel= 'rbf')
svc.fit(x_train,y_train)
print svc.score(x_test,y_test)
```

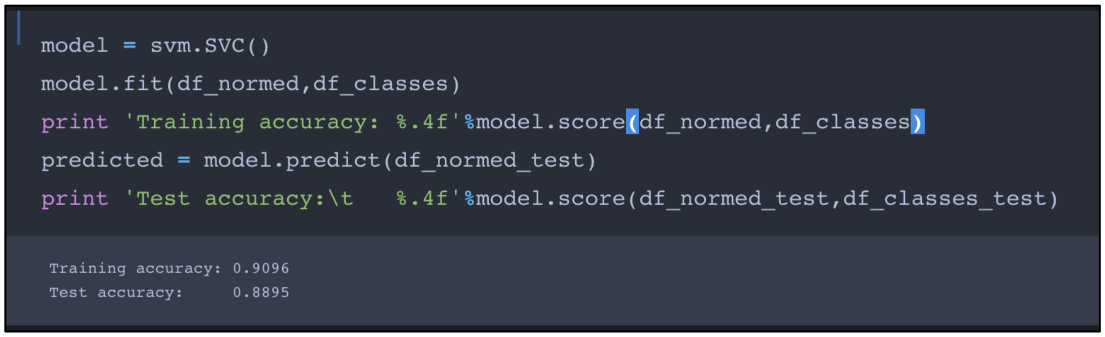
```
0.9528619528619529
```

Figure 16: Accuracy of SVC using optimal values for vowel sound data.

The fine grid search resulted in the following values for the hyperparameters for five cross validations: C-value of 13.16 and Gamma-value of 0.081 for the Radial Basis Function. Using these values, *SVC*( ) was performed on the test set and resulted in 95.28% accuracy; a 10.78% accuracy increase from the default run.

3.3 Satellite Imagery Dataset

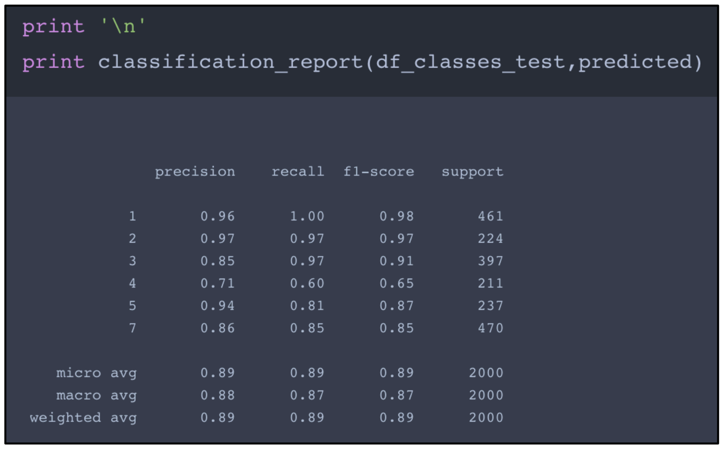
The satellite imagery dataset was not randomly split, as two unique datasets were already provided to train and test the algorithm. Implementing the *SVC*( ) function with default inputs in Python's *sklearn.svm* library, the accuracy on the training set and test set resulted in 90.96% and 89.95%, respectively. The *SVC*( ) function defaults to a Radial Basis Function kernel, a penalty parameter (C) of 1.0, and gamma parameter calculated as the inverse of (no. of features × data standard deviation).



```
model = svm.SVC()
model.fit(df_normed,df_classes)
print 'Training accuracy: %.4f'%model.score(df_normed,df_classes)
predicted = model.predict(df_normed_test)
print 'Test accuracy:\t   %.4f'%model.score(df_normed_test,df_classes_test)
```

```
Training accuracy: 0.9096
Test accuracy:     0.8895
```

Figure 17: Initial SVC accuracy for satellite imagery dataset.

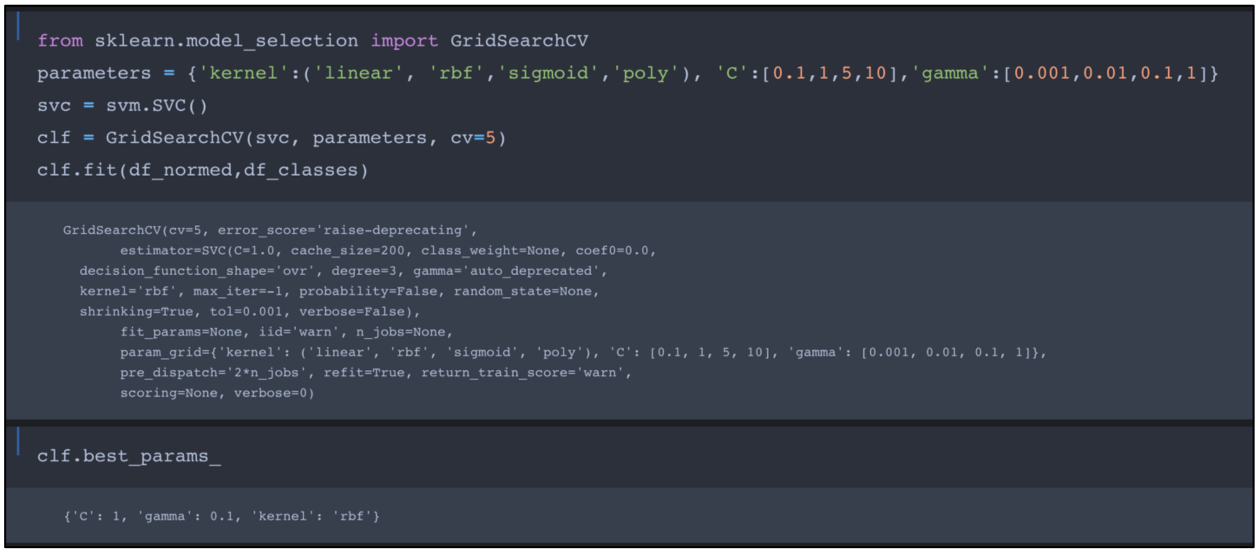


```
print '\n'
print classification_report(df_classes_test,predicted)
```

```
              precision    recall  f1-score   support

           1       0.96      1.00      0.98       461
           2       0.97      0.97      0.97       224
           3       0.85      0.97      0.91       397
           4       0.71      0.60      0.65       211
           5       0.94      0.81      0.87       237
           7       0.86      0.85      0.85       470

   micro avg       0.89      0.89      0.89      2000
   macro avg       0.88      0.87      0.87      2000
weighted avg       0.89      0.89      0.89      2000
```

Figure 18: Classification report using default SVC for satellite imagery dataset.

A coarse grid search was then performed penalty values (0.1, 1, 5, 10), gamma values of (0.001, 0.01, 0.1, 1), kernel functions (linear, radial basis function, sigmoid, polynomial) and five cross validations. This resulted in a penalty value of 10, gamma value of 0.1 and Radial Basis Function kernel.



```
from sklearn.model_selection import GridSearchCV
parameters = {'kernel':('linear', 'rbf','sigmoid','poly'), 'C':[0.1,1,5,10],'gamma':[0.001,0.01,0.1,1]}
svc = svm.SVC()
clf = GridSearchCV(svc, parameters, cv=5)
clf.fit(df_normed,df_classes)
```

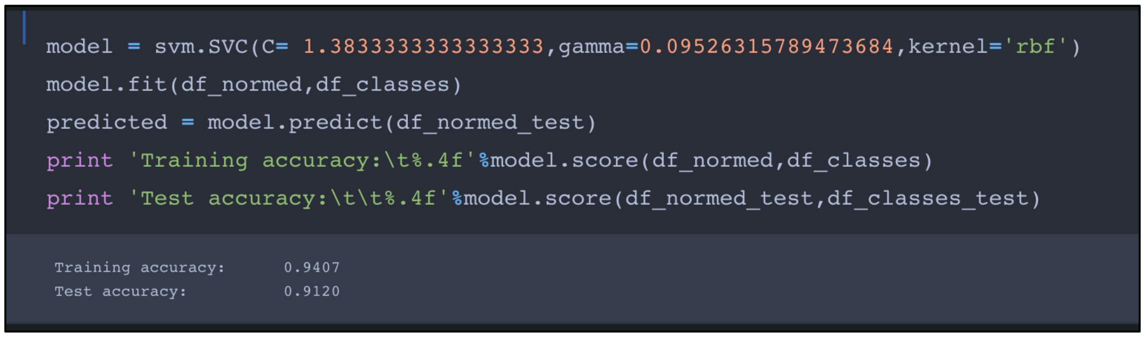
```
GridSearchCV(cv=5, error_score='raise-deprecating',
       estimator=SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='rbf', max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False),
       fit_params=None, iid='warn', n_jobs=None,
       param_grid={'kernel': ('linear', 'rbf', 'sigmoid', 'poly'), 'C': [0.1, 1, 5, 10], 'gamma': [0.001, 0.01, 0.1, 1]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=0)
```

```
clf.best_params_
```

```
{'C': 1, 'gamma': 0.1, 'kernel': 'rbf'}
```

Figure 19: Screenshot of coarse grid search for satellite imagery dataset.

Using these values from the coarse grid search, a range of values for hyperparameters are used to optimize the learning algorithm. The range of values utilized for the fine coarse search are as follows: C-values from 0.1 to 1.5 in 25 steps and Gamma-values from 0.01 to 0.8 in 20 steps. The fine grid search resulted in a C-value of 0.5 and a Gamma-value of 0.135 for the Radial Basis Function kernel.



```
model = svm.SVC(C= 1.3833333333333333,gamma=0.09526315789473684,kernel='rbf')
model.fit(df_normed,df_classes)
predicted = model.predict(df_normed_test)
print 'Training accuracy:\t%.4f'%model.score(df_normed,df_classes)
print 'Test accuracy:\t\t%.4f'%model.score(df_normed_test,df_classes_test)
```

```
Training accuracy:      0.9407
Test accuracy:          0.9120
```

Figure 20: Accuracy of SVC using optimal values for satellite imagery data.

Using these hyperparameter values, the SVC resulted in an accuracy of 94.07% and 91.2% on the training and test data resulting in an increase 3.11% and 2.25%, respectively.

References

Alpaydin, Ethem. *Introduction to Machine Learning*. Third Edition. The MIT Press. Cambridge, Massachusetts. (2014). ISBN: 978-0-262-02818-9.

Raschka, Sebastian. *Python Machine Learning*. Packt Publishing Ltd. Birmingham, UK. (2015). ISBN: 978-1-78355-513-0.