Machine Learning Project Report

Praveen Mareedu
N16407897

What do we have ?

We have an input file '**cs6923Project.mat**' which internally consists of train_label.csv , train.csv , test.csv files .

train.csv– This file consists of 50000 (rows) x 77 (columns) matrix of data inside (X), which can be used to train the algorithm , In the below approach , I have used 90% of the data to preprocess and 10% of the data to cross validate the results . Although this ration can be varied , 90%:10% gave the best accuracy scores on the test data.

train_label.csv : This file consist of 50000 (rows) x 1 (column) matrix of data inside (Y), which gives the label information for each row in train file .

test.csv **:** This file consists of 50000 [rows] x 77 [columns] same as train file but for which we need to predict the label information .

test_label.csv : File generated with predicted label information for the data in test.csv file .

Algorithms to consider :

This is a supervised learning problem . We need to use classification Machine Learning algorithms to predict output labels. Some of the best classification algorithms that can be used are **Logistic Regression , SVM (Support Vector Machines), Discriminant Analysis , K-Nearest neighbors etc**.

Prerequisites :

This program is developed in Python using popular open source machine learning library 'sklearn' .
-Need **SKLEARN** packages installed on the machine .
-Need Python – **Scipy , Numpy** installed on the machine.
Please check out the end of the document to find the installation links.

Algorithms used to test :

In this project I tried to use two famous techniques Logistic Regression , SVM to train the classification algorithm and predict labels for the test data . Logistic regression being one of the easy approach and SVM because it produces high accuracy scores in the test runs. Algorithm speeds and memory consumptions are referred in the URL mentioned in the end of the document.

Parameters that can be tweaked and Results :

SVM :

Kernel : rbf (Gaussian) , linear , poly , sigmoid.

cache_size: To specify the kernel cache size.

C : Penalty Parameter for the error term.

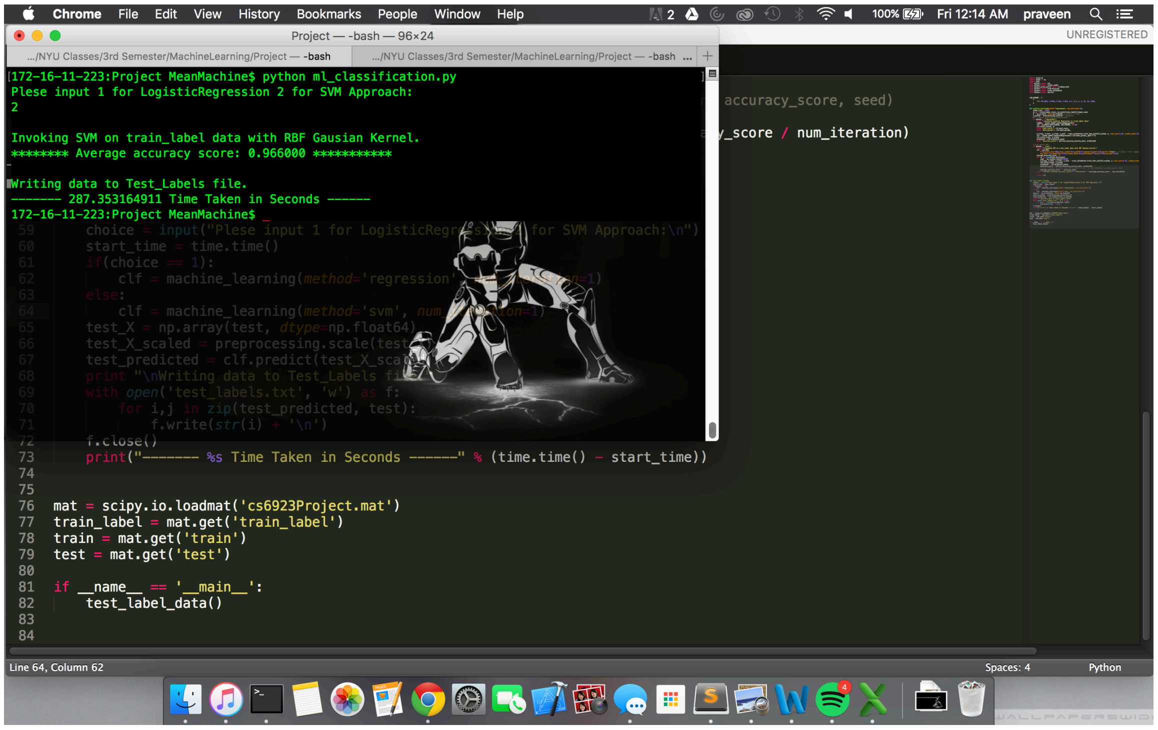
gamma : Kernel coefficient for 'rbf', 'poly' and 'sigmoid'

Logistic:

C : Regularization term

Results :

Cache Size 1000 and kernel RBF

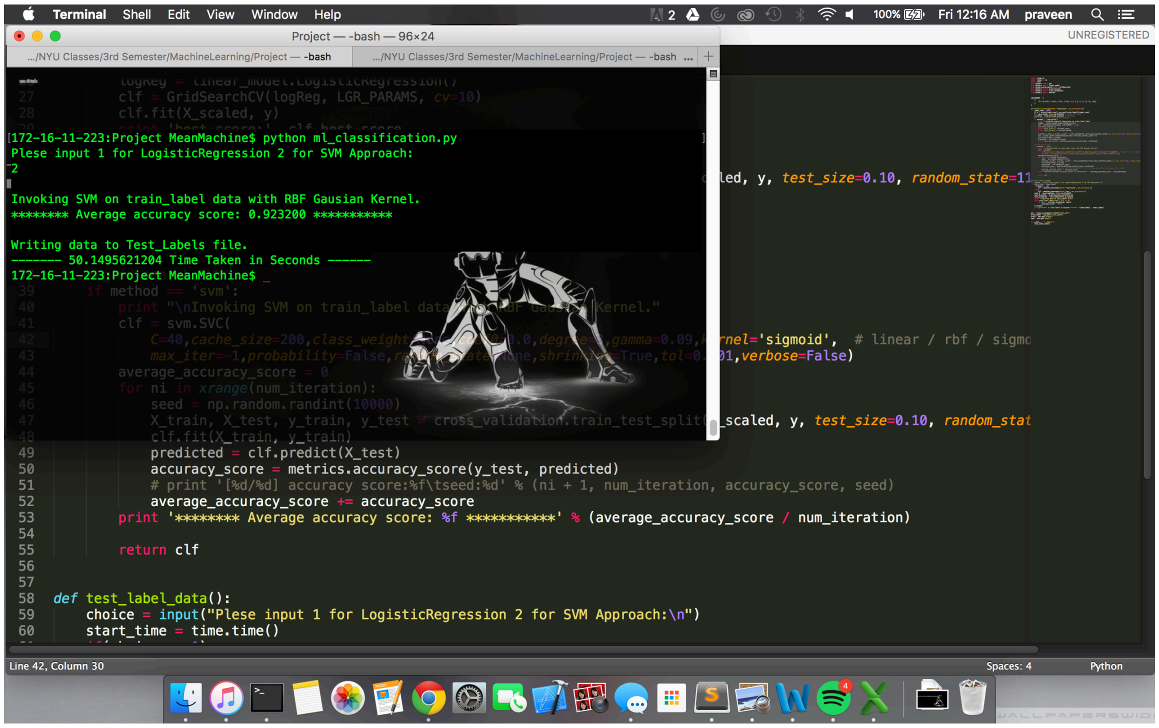


```
172-16-11-223:Project MeanMachine$ python ml_classification.py
Plese input 1 for LogisticRegression 2 for SVM Approach:
2

Invoking SVM on train_label data with RBF Gausian Kernel.
******** Average accuracy score: 0.966000 ***********

Writing data to Test_Labels file.
------- 287.353164911 Time Taken in Seconds ------
172-16-11-223:Project MeanMachine$ _
```

```python
    choice = input("Plese input 1 for LogisticRegr        for SVM Approach:\n")
    start_time = time.time()
    if(choice == 1):
        clf = machine_learning(method='regression'                 =1)
    else:
        clf = machine_learning(method='svm', num                   1)
    test_X = np.array(test, dtype=np.float64)
    test_X_scaled = preprocessing.scale(test
    test_predicted = clf.predict(test_X_sca
    print "\nWriting data to Test_Labels fi
    with open('test_labels.txt', 'w') as f:
        for i,j in zip(test_predicted, test):
            f.write(str(i) + '\n')
    f.close()
    print("------- %s Time Taken in Seconds ------" % (time.time() - start_time))


mat = scipy.io.loadmat('cs6923Project.mat')
train_label = mat.get('train_label')
train = mat.get('train')
test = mat.get('test')

if __name__ == '__main__':
    test_label_data()


```

```
accuracy_score, seed)
_score / num_iteration)
```

Cache Size 1000 and Kernel Function Sigmoid:

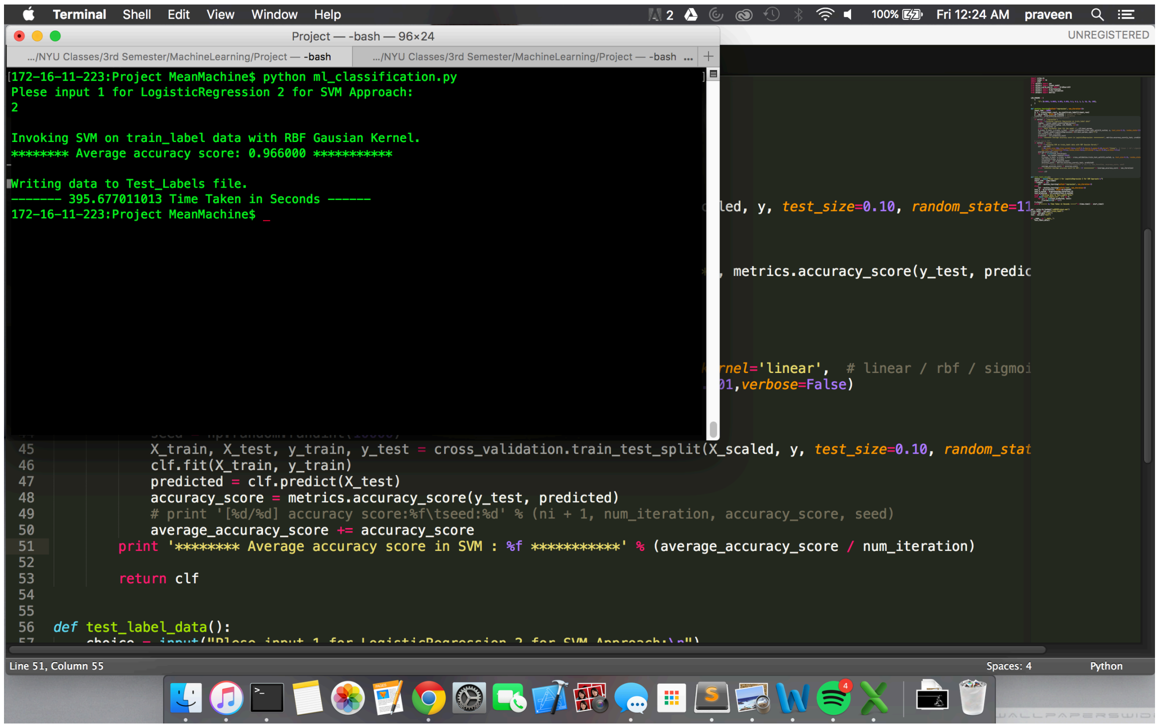


```
172-16-11-223:Project MeanMachine$ python ml_classification.py
Plese input 1 for LogisticRegression 2 for SVM Approach:
2

Invoking SVM on train_label data with RBF Gausian Kernel.
******** Average accuracy score: 0.923200 ***********

Writing data to Test_Labels file.
------- 50.1495621204 Time Taken in Seconds ------
172-16-11-223:Project MeanMachine$ _
```

```
rnel='sigmoid',  # linear / rbf / sigmo
01,verbose=False)
```

```
            predicted = clf.predict(X_test)
            accuracy_score = metrics.accuracy_score(y_test, predicted)
            # print '[%d/%d] accuracy score:%f\tseed:%d' % (ni + 1, num_iteration, accuracy_score, seed)
            average_accuracy_score += accuracy_score
        print '******** Average accuracy score: %f ***********' % (average_accuracy_score / num_iteration)

        return clf


def test_label_data():
    choice = input("Plese input 1 for LogisticRegression 2 for SVM Approach:\n")
    start_time = time.time()
```

Cache Size 200 and Linear Kernel :

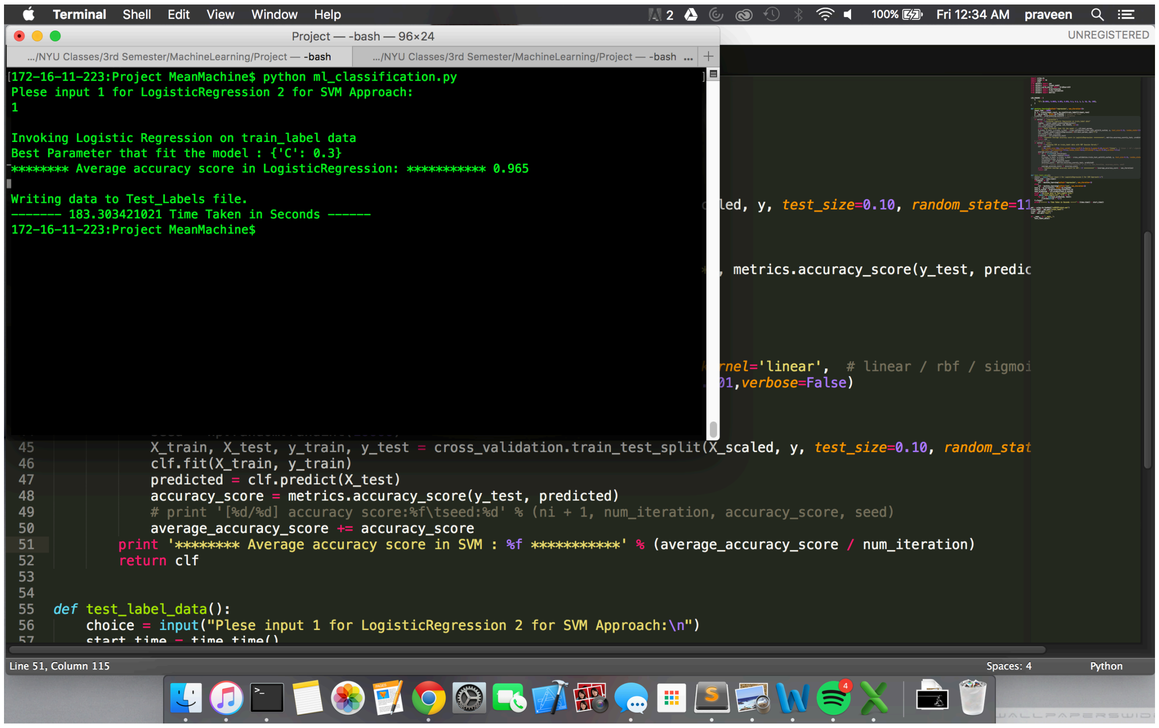


```
172-16-11-223:Project MeanMachine$ python ml_classification.py
Plese input 1 for LogisticRegression 2 for SVM Approach:
2

Invoking SVM on train_label data with RBF Gausian Kernel.
******** Average accuracy score: 0.966000 ***********

Writing data to Test_Labels file.
------- 395.677011013 Time Taken in Seconds ------
172-16-11-223:Project MeanMachine$ _
```

```
rnel='linear',  # linear / rbf / sigmoi
01,verbose=False)
```

```
            X_train, X_test, y_train, y_test = cross_validation.train_test_split(X_scaled, y, test_size=0.10, random_stat
            clf.fit(X_train, y_train)
            predicted = clf.predict(X_test)
            accuracy_score = metrics.accuracy_score(y_test, predicted)
            # print '[%d/%d] accuracy score:%f\tseed:%d' % (ni + 1, num_iteration, accuracy_score, seed)
            average_accuracy_score += accuracy_score
        print '******** Average accuracy score in SVM : %f ***********' % (average_accuracy_score / num_iteration)

        return clf


def test_label_data():
```

Logistic Regression :


Terminal Shell Edit View Window Help
Fri 12:34 AM praveen
Project — -bash — 96×24
.../NYU Classes/3rd Semester/MachineLearning/Project — -bash
UNREGISTERED
Line 51, Column 115
Spaces: 4
Python

```
172-16-11-223:Project MeanMachine$ python ml_classification.py
Plese input 1 for LogisticRegression 2 for SVM Approach:
1

Invoking Logistic Regression on train_label data
Best Parameter that fit the model : {'C': 0.3}
******** Average accuracy score in LogisticRegression: *********** 0.965

Writing data to Test_Labels file.
------- 183.303421021 Time Taken in Seconds ------
172-16-11-223:Project MeanMachine$
```

```
led, y, test_size=0.10, random_state=11
, metrics.accuracy_score(y_test, predic
rnel='linear',  # linear / rbf / sigmoi
01,verbose=False)
```

```python
            X_train, X_test, y_train, y_test = cross_validation.train_test_split(X_scaled, y, test_size=0.10, random_stat
            clf.fit(X_train, y_train)
            predicted = clf.predict(X_test)
            accuracy_score = metrics.accuracy_score(y_test, predicted)
            # print '[%d/%d] accuracy score:%f\tseed:%d' % (ni + 1, num_iteration, accuracy_score, seed)
            average_accuracy_score += accuracy_score
        print '******** Average accuracy score in SVM : %f ***********' % (average_accuracy_score / num_iteration)
        return clf


def test_label_data():
    choice = input("Plese input 1 for LogisticRegression 2 for SVM Approach:\n")
    start_time = time.time()
```

We can notice the time taken to execute the program varies in the each run with cache size . As the cache size increases the time taken to computer the program decreases and also with kernel RBF the accuracy levels are high with multiple runs .

References :

Classification Algorithms and their speed , memory consumption :

http://www.mathworks.com/help/stats/supervised-learning-machine-learning-workflow-and-algorithms.html?refresh=true

Scipy , Numpy :

http://www.scipy.org/install.html

Scikit Learn Kit :

http://scikit-learn.org/stable/install.html