CSCI 5408 ASSIGNMENT 6

By: Bhargav Dalal (B00785773) & Hardik Galiawala (B00777450)

DALHOUSIE UNIVERSITY

JULY 28, 2018

DALHOUSIE UNIVERSITY

Table of Contents

1. Objective:

In this assignment, we intend to learn to use scikit-machine learning took for data analysis and revisit the application of visualization to explore the data and use those insights in further analysis. We also plan to learn how to identify patterns through simulation of data.

2. Task Description:

We are trying to work on language identification problem given the sentence. But these sentences after performing vectorization becomes a very high dimensional problem. Thus, we are supposed to use a dimensionality reduction technique (PCA or TruncatedSVD). This will help us plot a few data points from each language type into less dimensional space giving us a brief idea about the given training data.

Based on this analysis, we must use Chi-Square statistics for selecting some number of features that may yield to the best accuracy.

We must use classifiers like Linear SVC, Logistic Regression, Decision Tree and Naïve Bayes to train our data model and then try to predict the data given in the test set [1]. We must use metrics like accuracy and confusion matrix for each model to report. Based on this report, we can evaluate which model works best for our problem. We are also supposed to perform hyper-parameter tuning. All these steps are needed to be combined in a pipeline which is given in "sklearn" package.

3. Feature Extraction and Selection:

Before extracting the features, we tried to subsample three hundred rows per language type (target label). Then we use CountVectorizer() method which returns a dense matrix of the count of occurrence of words in the given sub-sample. Due to this dense array, we use TruncatedSVD() to perform dimensionality reduction. Then these data points are plotted in a pair-plot as given below.

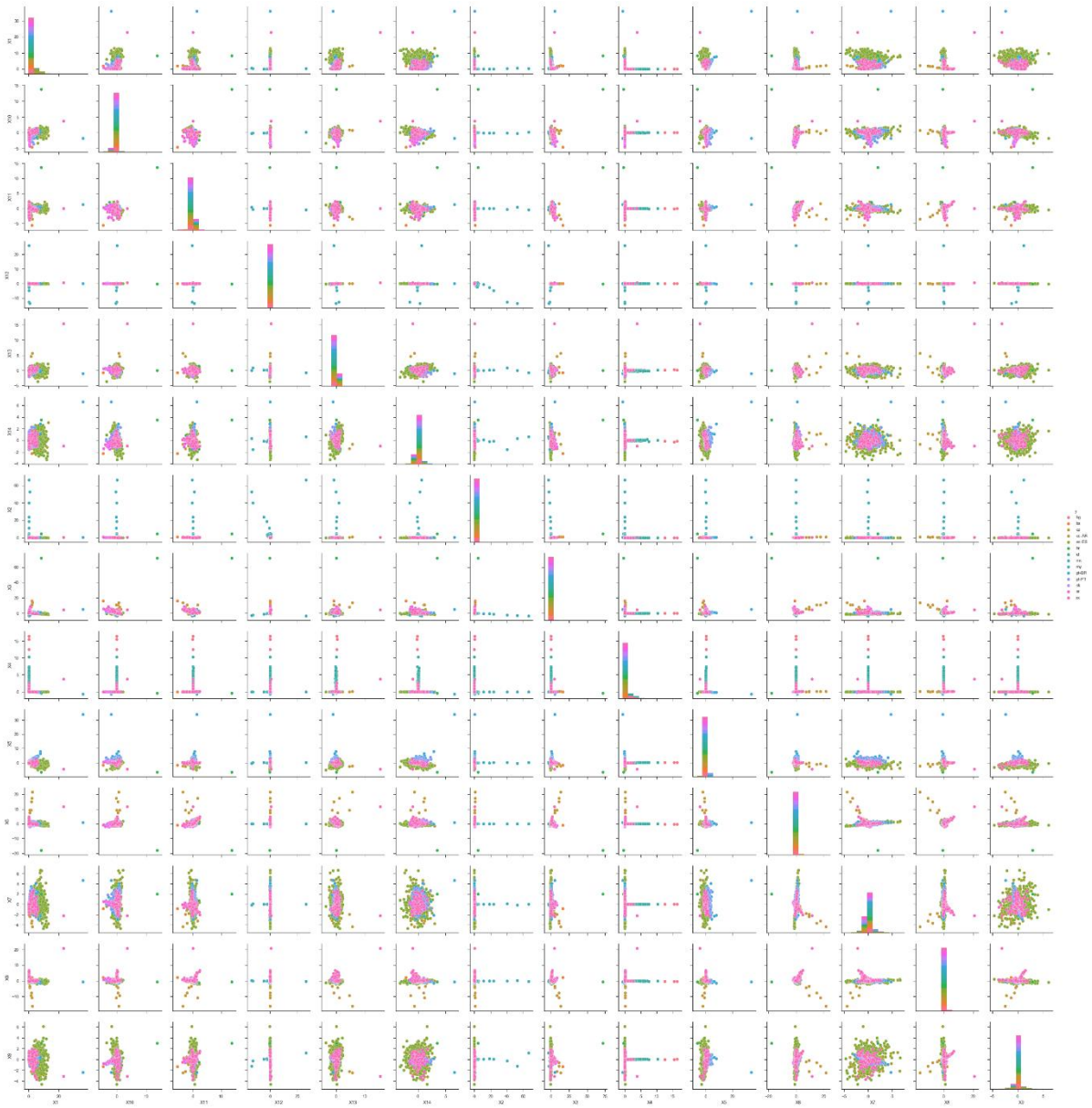

Fig: 3.1. Pairplot.

From the above figure, it can be clearly understood that the data can be explained in 14 components or dimensions. This gives us a promising idea that the data points give us a better understanding in a higher dimensional space. This helps us in dealing with the “Dimensionality Curse” which will directly allow us to make predictions with less number of features thereby reducing the time and computation power in training the dataset.

Furthermore, we have used SelectKBest() method provided in the scikit-learn package with the Chi2 statistical function to determine and select best features with the least correlation between one another. The reason behind doing this is Chi-Square test allows us to understand which categorical features hold significant association between one another [2].

4. Classification Algorithm:

As per the requirement of this assignment, we have used Decision Tree, Linear SVC, Naïve Bayes and Logistic Regression for classifying the dataset. All these algorithms are used at the end of the pipeline after performing CountVectorizer() and SelectKBest(chi2). We have also used "GridSearchCV()" to execute this pipeline. "GridSearchCV" selects the "k" hyperparameter which provides the best accuracy. It can be noted that "GridSearchCV" helps in hyperparameter tuning. We have only performed hyperparameter tuning for SelectKBest() method and not for any other algorithm for obvious reasons like high computational costs.

4.1. Decision Tree:

```
pipeDecisionTree = Pipeline([
        ('vectorizer', CountVectorizer()),
        ('selection', SelectKBest(chi2)),
        ('decision_tree', DecisionTreeClassifier())
        ])
```

Fig: 4.1 Decision Tree Pipeline.

This classifier is one of the predictive modeling approaches used in the field of machine learning [3]. It is also one of the methods associated with ensemble methods involving bagging. Decision Trees divides or classifies the data given the features in the form of nodes or a binary tree. The algorithm available in scikit-learn uses "Gini" purity as a default method of calculating purity of each leaf node. But if we specify it may use entropy and information gain as well. But for the sake of simplicity, we have not played with that hyperparameter in this assignment. It is important to note that DecisionTreeClassifier() method handles dense index well enough [4].

4.2. Linear SVC:

```
pipeLinearSVC = Pipeline([
        ('vectorizer', CountVectorizer()),
        ('selection', SelectKBest(chi2)),
        ('linear_svc', LinearSVC(multi_class = 'ovr'))
        ])
```

Fig: 4.2 Linear SVC Pipeline.

LinearSVC() is one of the more flexible methods of classification from "sklearn.svm" class [5]. This method supports both, the dense and sparse matrices and handles multi-class classification well. It uses "One v/s Rest" (ovr) while predicting.

4.3. Naïve Bayes:

```
# Reference for multinomial naive bayes
#http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html
pipeNaiveBayes = Pipeline([
        ('vectorizer', CountVectorizer()),
        ('selection', SelectKBest(chi2)),
        ('naive_bayes', MultinomialNB())
        ])
```

Fig: 4.3. Naïve Bayes Pipeline.

MultinomialNB() is one of the classifiers of class sklearn.naive_bayes. As the name suggest, it is a suitable classifier for classifying discrete features [6]. Given the prior knowledge of the features, we try to train and build a model which helps in predicting the posterior probability of the target variable.

4.4. Logistic Regression

```
#Reference for multinomial logistic regression
# http://dataaspirant.com/2017/05/15/implement-multinomial-logistic-regression-python/
pipeLogisticReg = Pipeline([
        ('vectorizer', CountVectorizer()),
        ('selection', SelectKBest(chi2)),
        ('log_reg', LogisticRegression(multi_class = 'multinomial', solver = 'newton-cg'))
        ])
```

Fig: 4.4. Logistic Regression Pipeline.

As we already know that we are dealing with multi-class classification, we need to tweak LogisticRegression() to handle it. We cannot rely on sigmoid function and must rather use softmax to introduce non-linearities. We also need to specify multi_class = 'multinomial' and solver = 'newton-cg' because we are not using ovr prediction [7].

4.5. Performance and Feature Selection:

Please refer next part of the report where we have explained the analysis in detail.

5. Output:

5.1. Accuracy Comparison:

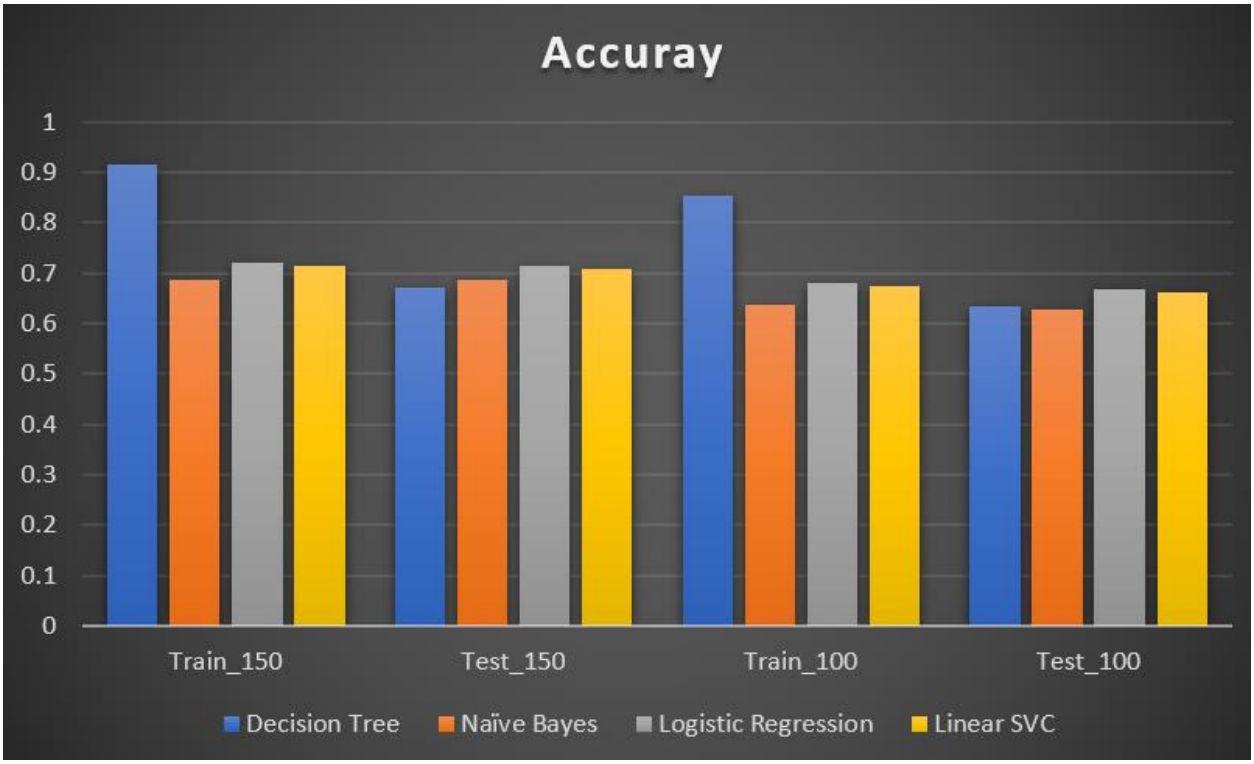

Fig: 5.1. Classifier v/s Feature selection v/s Accuracy Plot.

The above plot (Fig 5.1.) demonstrates a comparison of Training v/s Testing accuracy with feature selection per classifier. “Train_150” and “Test_150” denotes train and test accuracy when 150 features were selected using SelectKBest() method which was mentioned in the above section of this report. “Train_100” and “Test_100” denotes train and test accuracy when 100 features were selected.

As we can see, there is very less difference between Logistic Regression and Linear SVC (test accuracy) and both are almost at par. But, Logistic Regression beats Linear SVC by a very small margin (1% approx.).

Thus, we can say that both have similar accuracy.

5.2. Confusion Matrix:

The confusion matrix is a metric which can be used to describe how well a model performs. It is like accuracy, but it gives more detailed information about the True-Positives, True-Negatives, False-Positives, and False-Negatives. It can also be used to calculate True-Positive Rate (TPR) and False-Positive Rate (FPR).

We have provided a few confusion matrix plots per classifier (test set) in the following sections of the report. Detailed analysis can be found in the jupyter notebook “ReportJupyter.ipynb”.

5.2.1. Decision Tree:

Test Confusion Matrix Plot:-

```
In [12]: plot_confusion_matrix(test_y, predict_test)
```

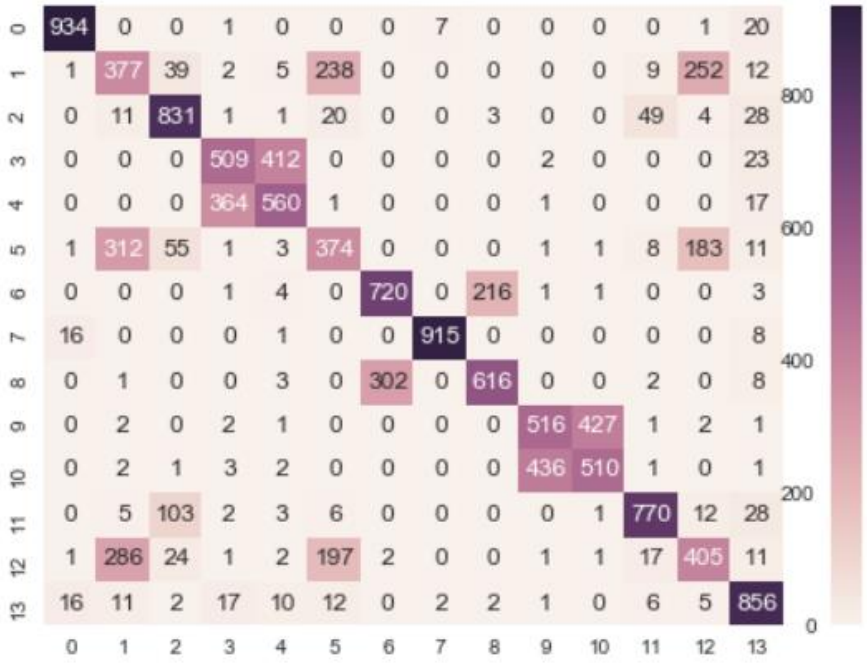

Fig: 5.2.1. Decision Tree (Test Confusion Matrix)

5.2.2. LinearSVC:

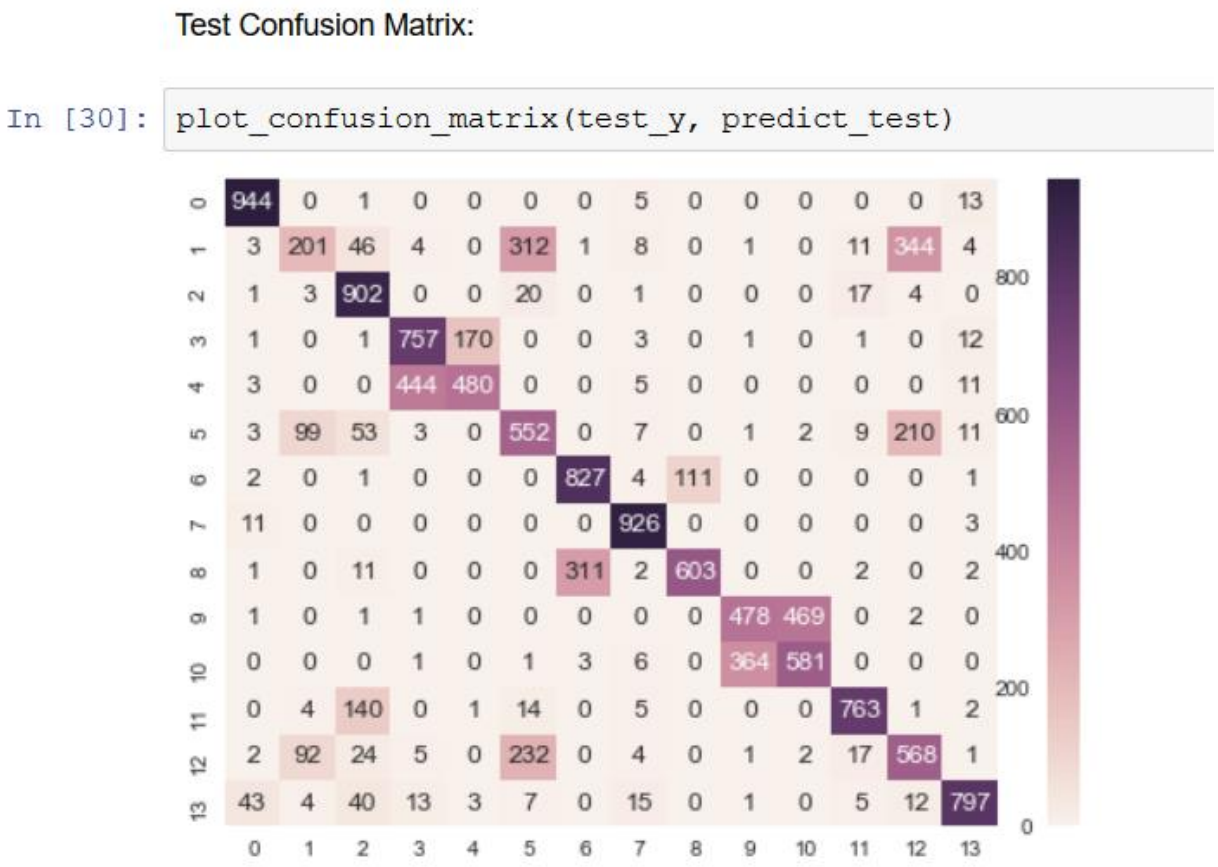

Fig: 5.2.2. Linear SVC (Test Confusion Matrix)

5.2.3. Naïve Bayes:

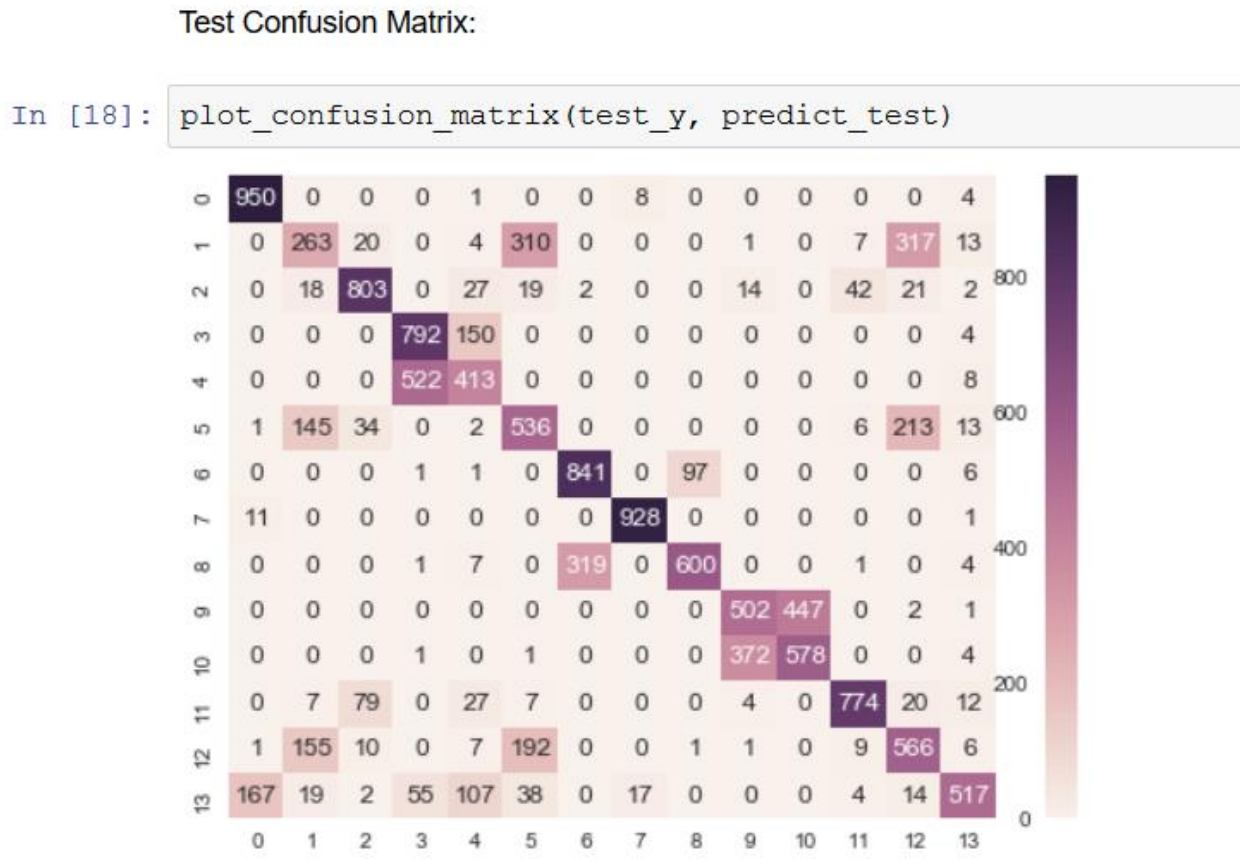

Fig: 5.2.1. Naïve Bayes (Test Confusion Matrix)

5.2.4. Logistic Regression:

Test Confusion Matrix:

```
In [24]: plot_confusion_matrix(test_y, predict_test)
```

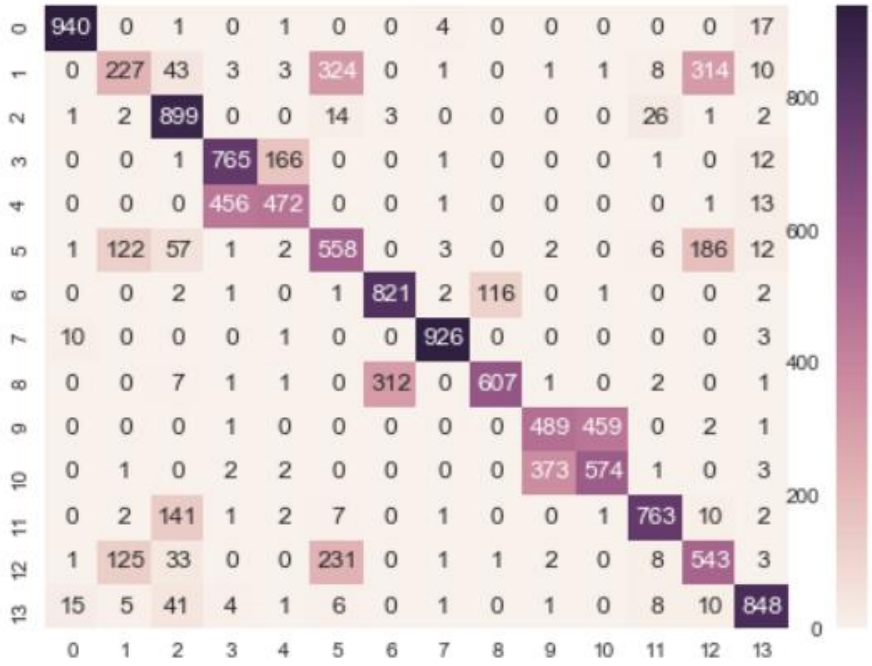

Fig: 5.2.4. Logistic Regression (Test Confusion Matrix)

6. Code Submission:

GitHub Repo link: [Assignment6Repo]

Bibliography

[1] "github," [Online]. Available: https://github.com/Simdiva/DSL-Task. [Accessed 29 07 2018].

[2] "stattrek," [Online]. Available: https://www.stattrek.com/chi-square-test/independence.aspx?Tutorial=AP. [Accessed 29 07 2018].

[3] "wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Decision_tree_learning. [Accessed 29 07 2018].

[4] scikit-learn, "scikit-learn," [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier. [Accessed 29 07 2018].

[5] "svc," [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html. [Accessed 29 07 2018].

[6] "naivebayes," [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html. [Accessed 29 07 2018].

[7] "logreg," [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html. [Accessed 29 07 18].