**Yasir Hassan**

**Machine Learning** **to Classify Single-Cell RNA Sequencing in Primary Glioblastoma**

**Contents:**

* Abstract
* Introduction
* Multiclass Classification
* Algorithm Performance Evaluation
* Conclusion
* References

**Abstraction:**

Latest advances in single-cell ribonucleic acid sequencing technologies enable deep insights into cellular development, measuring gene expression for thousands of cells in a single experiment. This requires the development of new types of computational approaches to extract the useful and valuable underlying biological information of individual cells in heterogeneous biological populations. As a result, a deep learning technique paper, introduced by UH professors, to classify single cell types data from five primary Glioblastomas. In the paper, they show that the deep learning method can correctly classify cell type not used during the training process of the algorithm (P. Guillen et al).

**Introduction:**

The cited paper is applying the deep learning concept to perform an automated single cells classification using a dataset from 430 cells from five primary glioblastomas (P. Guillen et al).

The rest of this report is a technical evaluation of the performance of six different Machine Learning algorithms to classify the five classes of Single Cell-Seq using the database Data\_Glioblastoma5Patients\_SC.csv. The various machine learning algorithms I am using for the classification, and evaluation, are Logistic Regression, KNN, Decision Tree, Random Forest, SVM and Multi-Layer Perceptron.

**Multiclass Classification:**

Multiclass classifiers (also called multinomial classifiers) can distinguish between more than two classes. In multiclass classification, each instance is assigned to just one class. Some algorithms (such as Random Forest classifiers or naive Bayes classifiers) are capable of handling multiple classes directly. Others (such as Support Vector Machine classifiers or Linear classifiers) are strictly binary classifiers. However, there are various strategies that can be used to perform multiclass classification using multiple binary classifiers such as a technique called one-versus-all (OvA) strategy, or, also called one-versus-the-rest (Müller, 2016).

**Machine Learning Algorithm Performance Evaluation for Classifying Single-Cell RNA Sequencing in Primary Glioblastoma:**

**Software used**: Jupyter notebook , Python, scikit learn, GridSearchCV, and other related libraries.

**Cross-Validation:** cross-validation is a statistical method of evaluating generalization performance thatis more stable and thorough than using a split into a training and a test set. In cross validation,the data is instead split repeatedly, and multiple models are trained.

\*I am using GridSearchCV for fine-tuning the hyperparameters.

\*For training all classifiers, I split the data set to 80% training and 20% testing.

\*For more understanding, I looked at the image representation of the confusion matrix.

\*Analyzing the confusion matrix gave me insights on ways to improve the classifier.

**Evaluating the Algorithm Performance:**

**1) Random Forest:**

I used the training data split to train the RandomForestClassifier by fitting it by the train data. I initially built the Random Forest by instantiating its object with 10 trees. This choice of 10 trees was arbitrary.

The trees that were built as part of the random forest which stored in the estimator\_ attribute.

I then, visualized the decision boundaries learned by each tree, together with their aggregate prediction as made by the forest.

from using the Random Forest Classifier, I obtained the confusion matrix and the accuracy, of classifying the five classes of Single Cell-Seq, on the test set, was **91**%, where the test set was not part of the training data.

The random forest gave an initial accuracy of **91%**, without tuning any parameters. For multi class classification, often the default parameters of the random forest already work quite well.

Nevertheless, to get a better accuracy, I used parameter optimization where I utilized scikit learn GridSearchCV classifier with Cross Validation equal to 5. After fitting the GridSearchCV with the training data, I found the best parameter that I could use to be 100 trees as estimators in the Random Forest Classifier. So, with using 100 trees, as one of the parameters in the Random Forest Classifier, the accuracy score improved from 91% to **96%** for classifying the five classes of Single Cell-Seq.

For Random Forest, Scikit-Learn did not have to run One v all (OvA) because Random Forest classifiers could directly classify instances into multi classes.

**2) Logistic Regression (for Multi Class Classification):**

I fitted the LogisticRegressionClassifier with the training data split, for training the model. Scikit learn immediately engages multinomial Logistic Regression once it detects the existence of more than two classes to be classified. The initial accuracy of the test set was **97%** for classifying the five classes of Single Cell-Seq. Apparently from the initial classifying score, Logistic Regression did very well with multi class classification… Multinomial logistic Regression is using the softmax function instead of the sigmoid function of the cross-entropy loss function. The softmax function squashes all values to the range [0,1] and the sum of the elements is 1.

Then, for parameter optimization, and hyperparameters, to get a better score, I used GridSearchCV, with cross validation of 5. Then, I fitted the GridSearchCV classifier with the training data set. I found the best parameters to instantiate the LogisticRegressionClassifier as ‘C' = 100. As a result of parameter optimization, I obtained a new accuracy of **100 %** from the Logistic Regression Classifier for classifying the five classes of Single Cell-Seq. So, Logistic Regression is the best in multi class classification, for classifying the five classes of Single Cell-Seq.

**3)** **Multi-Layer Perceptron (MLP):**

I fitted the MLPClassifier with the training data split, for training the model. The hidden layer sizes I used were (500, 500, 500). The initial accuracy of the test set was **95%** for classifying the five classes of Single Cell-Seq. From the cross-validation accuracy, the score of the test set was great, for classifying the five classes of Single Cell-Seq.

The MLP model performed very well because MLPClassifier supports multi-class classification by applying softmax as the output function.

**4) Decision Tree:**

For training the model, I fitted the DecisionTreeClassifier with the training data split. The initial accuracy of the test set was **86%** for classifying the five classes of Single Cell-Seq. The score of the training set was 100% because the leaves were pure. The tree was grown deep enough that it could perfectly memorize all the labels on the training data.

Then, for parameter optimization, to get a better score, I used GridSearchCV, with cross validation of 5. Then, I fitted the GridSearchCV classifier with the training data set. I found the best parameters to instantiate the DecisionTreeClassifier as ‘criterion’ = ‘gini’ and ‘splitter’ = ‘best’. As a result of parameter optimization, I obtained n new accuracy of **89.5%** from the Decision Tree Classifier for classifying the five classes of Single Cell-Seq. So, the accuracy has improved.

I applied pre-pruning to the tree to stop developing the tree before fitting to the training data. I did this to stop building the tree after a certain depth has been reached. Here I set max\_depth =4, meaning only four consecutive levels for the tree. However, limiting the depth of the tree decreases overfitting. This might lead to a lower accuracy on the training set, in general.

Then, to get a better understanding of the Decision Tree and the classification of the 5 classes, I visualized the tree using the export\_graphviz function from the tree module.

**5)** **K Nearest Neighbors:**

For training the model, I fitted the KNeighborsClassifier with the training data split. I used 7 as the n\_neighbors. The initial accuracy of the test set was **62**% for classifying the five classes of Single Cell-Seq.

Then, for parameter optimization, to get a better score, I used GridSearchCV, with cross validation of 5. Then, I trained the GridSearchCV classifier with the training data set. I found the best parameters to instantiate the KNeighborsClassifier as ‘n\_neighbors’ = 3. As a result of parameter optimization, I obtained new accuracy of **68**% from the KNeighborsClassifier for classifying the five classes of Single Cell-Seq. So, the accuracy has slightly improved.

Apparently, this is one the worst accuracy score I have obtained from a model (KNeighbors) for classifying the five classes of Single Cell-Seq.

**6)** **Support Vector Machine:**

For training the model, I fitted the polynomial\_svm\_clf with the training data split. I used the parameters kernel = ‘poly’, polynomialFeatures (degree = 5), ‘C’ = 10, and loss = “hinge”.

After fitting the model, the initial accuracy of the test set was **53%** for classifying the five classes of Single Cell-Seq. After reviewing the model confusion matrix, it appeared that the model did not do any better than chance.

Then, for parameter optimization, and hyperparameters, to get a better score, I used GridSearchCV, with cross validation of 5. Then, I fitted the GridSearchCV classifier with the training data set. Then, I found the best parameters to instantiate the svm\_clf as 'C' = 1, and 'kernel': 'poly’.

As a result of parameter optimization, I obtained n new accuracy of **53 %** from the svm\_clf for classifying the five classes of Single Cell-Seq.

Apparently, in general, SVM is a binary classifier, defined to separate 2 classes.

**Conclusion:**

For multi class classification, classifying five classes of Single Cell-Seq, different machine learning algorithms perform differently (some algorithms perform very well and other algorithms perform very poorly).

The following is a list of the machine learning algorithm accuracy results that I have obtained from classifying the five classes of Single Cell-Seq:

* Logistic Regression: **accuracy = 100%**
* Random Forest: **accuracy = 96%**
* Multi-Layer Perceptron: **accuracy = 95%**
* Decision Tree:  **accuracy = 89%**
* K Nearest Neighbors **accuracy = 68%**
* Support Vector Machine **accuracy = 53%**

**References**

Guillen, P., Robinson, M., Ebalunode, J. (). Deep Learning to Classify Single-Cell RNA Sequencing in Primary Glioblastoma.

Müller, Andreas C.; Guido, Sarah (2016). Introduction to Machine Learning with Python (Kindle Location 38). O'Reilly Media. Kindle Edition.