Saptarshi Ghose
Machine Learning in Cancer

Lab 1 Write-Up

Part 1. Normal and Tumor Match Pair Analysis

a. Using SciKit Learn build a machine learning classifier that takes RNAseq profiles from matched normal tumor pairs and classifies the sample as Normal or Tumor. Compare the **nt.coding.csv** vs the **nt.all.csv**.

b. Using model selection methods of your choice determine which classical ML method performs best on the NT classification problem.

To tackle these two parts of the problem, I initially built a machine learning pipeline (in the ml_functions.py file) that instantiates several of the most common classical machine learning models (i.e. Random Forest, Logistic Regression, Naïve Bayes, and Decision Trees). The pipeline includes grids for different hyper-parameter combinations to test each model with (each tested in a loop function called run_simple_loop). From there, I built a function called clf_loop (also called by run_simple_loop) that builds out the 5-fold cross validation – along with detailed evaluation metrics (including auc-roc, precision at thresholds: [1, 2,5,10,20,30,50], recall at thresholds: [1, 2,5,10,20,30,50], F1 at thresholds: [1, 2,5,10,20,30,50], and confusion matrix) across each of the folds. Then, for each unique model/parameter combination, I took the mean of the evaluation metrics – with particular focus on auc-roc score – and compared the results.

For nt.coding, the best performing machine learning model that I tested was a Random Forest classifier with n_estimators: 200, max_depth: 5, max_features: sqrt, min_samples_split: 2, n_jobs = -1). This had an auc-roc score of .9906285.

For nt.all, the best performing machine learning model that I tested was also a Random Forest classifier with n_estimators: 200, max_depth: 10, max_features: sqrt, min_samples_split: 2, n_jobs = -1). This had an auc-roc score of .99037467.

c. Using feature selection methods of your choice determine a < 100 gene signature that can be used to classify Normal vs Tumor.

I used sklearn's SelectKBest method, using a chi-squared test evaluator, the find the 99 most relevant features used to classify normal vs. tumor. I ran this feature selection algorithm for each of the datasets – the corresponding outputs (including the list of 99 most relevant features for each dataset) are in the attached Python notebook.

I then ran the previous constructed machine learning pipeline on the 99 gene signature separately for nt.coding vs. nt.all and compared their corresponding evaluation metrics. The best performing model, according to average auc-roc score after 5-fold cross validation, achieved a score of .9834 in nt.coding while the best performing model for .98327 in nt.all. These results are quite both quite strong – and are only slightly less well-performing than the results achieved when running the whole datasets (with all features included) on the nt.coding and nt.all datasets.

d. Using **Keras**, build a deep learning classifier that performs the same classification task, and determine the learning curve (relationship of number of training samples to prediction accuracy) for your network, recommend using at least 10 training set sizes to estimate the learning curve.

With Keras, I built a deep learning classifier to perform the same classification task on the nt.coding and nt.all datasets. The learning curve plots across the epochs and folds are provided in the attached Python notebook. For the nt.coding dataset, the deep learning classifier returned an average accuracy score of .9786 while the nt.all dataset returned an average accuracy score of .9714. This was just slightly less accurate than the best performing model achieved through the classical machine learning algorithm methods implemented in my loop function for the above sections. I also ran the deep learning classifier on the <100 gene signature features for nt.coding and nt.all – achieving average accuracy scores of .9393 and .9286 respectively.

e. **Extra Credit**: Use the TPOT autoML system to search for a better solution to part a.

I used the TOPT autoML system on the provided datasets and exported the results to Python files I have included with this submission. For the nt.coding dataset, the best performing classifier was RandomForestClassifier(bootstrap=False, criterion="entropy", max_features=0.05, min_samples_leaf=1, min_samples_split=5, n_estimators=100) with an average cross-validation score of .96339. For the nt.all dataset, the best performing classifier was GradientBoostingClassifier(learning_rate=0.1, max_depth=2, max_features=0.8, min_samples_leaf=15, min_samples_split=2, n_estimators=100, subsample=0.9000000000000001) with an average cross-validation score of .954455. Both of these scores were respectively lower than what was achieved through the classification model loop from the above sections – though still quite high overall. In the future, I might tune the parameters from the TPOT classifier differently, such as by adding more generations for passes through the data, to test if relevant evaluation metrics improve.

.

Part 2. **Cancer Type Classifier for 18 Common Tumor Types.**

a. Using SciKit Learn build a machine learning classifier that classifies Cancer Type from the **type.coding.csv** and **type.all.csv** files. Compare the coding vs all genes cases.

b. Using model selection methods of your choice, determine which classical ML method performs best.

Using the same loop as above on Naïve Bayes and Logistic Regression models with several different test parameters in a grid that I looped through, I observed the following results:

For type.coding, the best performing machine learning model that I tested was a Logistic Regression model with the following parameters: LR{'C': 1, 'penalty': 'l1'}. This had an average accuracy score of .9668.

For type.all, the best performing machine learning model that I tested was also a Logistic Regression model with these parameters LR{'C': 1, 'penalty': 'l1'}. This had an average accuracy score of .9658.

c. Using feature selection methods of your choice, determine a < 100 gene signature that can be used to classify tumor type.

For this section, I again used sklearn's SelectKBest method, using a chi-squared test evaluator, the find the 99 most relevant features used to classify for 18 common tumor types. I ran this feature selection algorithm for each of the datasets – the corresponding outputs (including the list of 99 most relevant features for each dataset) are in the attached Python notebook.

d. Using **Keras**, build a deep learning classifier that performs the same classification task, and determine the learning curve (relationship of number of training samples to prediction accuracy) for your network.

With Keras, I built a deep learning classifier (with one-hot-encoding of the label) to perform the same classification task on the type.coding and type.all datasets. The learning curve plots across the epochs and folds are again provided in the attached Python notebook. For the type.coding dataset, the deep learning classifier returned an average accuracy score of .99421. while the type.all dataset returned an average accuracy score of .92407. This was again just slightly less accurate than the best performing model achieved through the classical machine learning algorithm methods implemented in my loop function for the above sections. I also ran the deep learning classifier on the <100 gene signature features for type.coding and type.all – achieving average accuracy scores of .94514 and .87130 respectively. These also performed reasonably well – but lower than with all features included.

<u>**Overall Comments**</u>

To improve upon these models and the analysis process in the future, I would consider distributing some of the workload and running it in parallel on an AWS cluster. This would allow me to test a larger grid of models and model parameters in a shorter amount of runtime – potentially leading to more useful models to choose from.

Note:

Referenced my code from previous machine learning classes, StackOverflow, and Sklearn/Keras documentation for this lab.