Assignment 3: Model Selection for Imbalanced Binary Classification Problem

Sourav Sarker (201892985) & Isaac Adeuwon (200925352)

Date: 8 March 2019

Introduction: For this assignment, we have used libraries `sys`, `pandas` and `numpy` for dataset handling. As an approach for feature selection and dealing with imbalanced dataset, we have employed scikit-learn libraries named: `SelectFromModel`, `RFECV` (Recursive feature elimination with cross-validation), `StratifiedKFold` and `GridSearchCV`. To get comparatively a good model, we have applied four different classification algorithms with sci-kit learn libraries. To execute this program from the command prompt:

`$ python A3.py A3_training_dataset.tsv A3_test_dataset.tsv`

Feature Selection: After uploading provided training and test dataset and extracting training dataset from the class, we initiated two approaches with `SelectFromModel` and `RFECV` for choosing the best features using `RandomForestClassifier` model. The Recursive feature elimination with cross-validation eliminated six features from the 71 features where we used three cross-validations. On the other hand, the SelectFromModel with 0.01 as the threshold picks eight features.

```
1: data←pd.read_csv('A3_training_dataset.tsv', where sep is '\t' and header is None)
2: real_test ← pd.read_csv('A3_test_dataset.tsv', where sep is '\t' and header is None)
3: X ← select all columns with rows from data except last
4: y ← select the last column from data
```

With Recursive CV:

```
5: selector ← RFECV(model_randomForest and cv=3)
6: selector ← fit X and y with the selector
7: X_new ← transform the X
```

Output: Shape of X_new is 3817, 65

With SelectFromModel:

```
8: sfm ← SelectFromModel(model_randomForest nd threshold=0.01)
9: sfm ← fit X and y with sfm
10: X_new ← transform the X
```

Output: Shape of X_new is 3817, 13

So, we decided to go with SelectFromModel to eliminate unnecessary features from the training dataset.

Imbalanced data handling and classification methods: Machine learning algorithms get affected when the data is highly imbalanced, that means where one class represents higher than the other class in size for a binary classification problem. To handle this problem, we applied the **Support Vector Classifier, KNN Classifier and Random Forest Classifier** with stratified 10-fold cross-validation under a grid search cross-validation. As we know, we measure accuracy from the number of correct predicted samples over the total number of samples.

$$\text{accuracy} = \frac{\text{nr. correct predictions}}{\text{nr. total predictions}} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}}$$

Fig: 1

For instance, if our classifier predicts 90% accuracy score, it means that out of 100 instances it predicts the appropriate class for 90 of them. So, for the imbalanced binary classification problem, this can be misleading. As a result, to get the actual performance of these models with an unbalanced dataset, we utilized average precision as a performance metric. Average precision gives us a mean score at all such possible thresholds, which is also related to the area under the precision-recall curve. It is a convenient metric to compare how well models are ordering the predictions for an imbalanced data, without considering any specific decision threshold.

```
1: strati_CV ← StratifiedKFold(where the fold is 10, the shuffle is True)
2: score_metric ← ['average_precision']
```

k-NN Classification Model: k-Nearest Neighbours (k-NN) classifies unclassified data point based on its distance from closest neighbours. In the below figure, we can see that a k-Nearest Neighbours model classifies the unknown data point by creating a circle with the point as the centre. A hyper-parameter, k decides the size of the circle. Adopting different value for k does not mean to the actual size of the circle; however, it indicates the closest data points inside the circle.

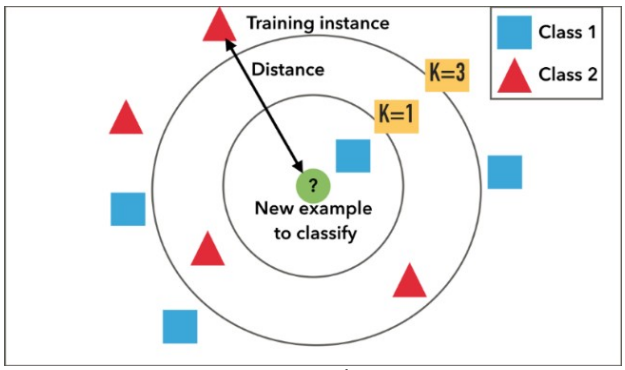

Fig: 2[1]

From this circle, by counting the number of neighbours from each with no other hyper-parameters set and the classification is done by a 'Vote', which means the class that is seen most frequently within the circle is chosen for the unknown point.

Why k-NN with Grid Search CV: A machine learning literature in 2003 identified that imbalanced class sizes with k-NN are both a practical and theoretical problem. A google scholar search shows several articles describing this issue with some strategies to mitigate this by fine-tuning the k-NN algorithm as:

- weighting neighbours by their distances
- weighting neighbours by the inverse of their class size convert neighbour counts into the fraction of each class that falls in k nearest neighbours.

For this assignment, our approach for k-NN with GridSearchCV was as follows:

```
1: k_range ← set the value from 3
   to 31
2: weight_options ← ['uniform' and
   'distance']
3: algorithms  ← algorithms  ←
   ['auto' 'ball_tree', 'kd_tree'
   and 'brute']
4: metric ← ['minkowski',
   'euclidean' and 'manhattan']
5: param_grid ← where n_neighbors =
   k_range, weights =
   weight_options, algorithms =
   algorithms and metrics=metric
6: knn ← KneighborsClassifier()
7: for the value of score in
   score_metric:
   1: grid_cv_knn ← GridSearchCV
      (knn,
   2: param_grid, cv=strati_CV,
   3: scoring=score)
   4: grid_cv_knn ← fit the
      grid_cv_knn with X_new and y)
   5: return the mean value ←
      grid_cv_knn.best_score_
8: return the best parameter ←
   grid_cv_knn.best_params_
9: return the best estimator ←
   grid_cv_knn.best_estimator_)
10:      return the standard
   deviation ←
   grid_cv_knn.cv_results_['std_tes
   t_score']
   [grid_cv_knn.best_index_]
```

Support Vector Machine: "Support Vector Machine" (SVM), a supervised machine learning model, most commonly used for classification problem. In general, this algorithm plots each data as data point into an n-dimensional space (where n = the total number of features) with the value of each feature being the value of a particular coordinate.

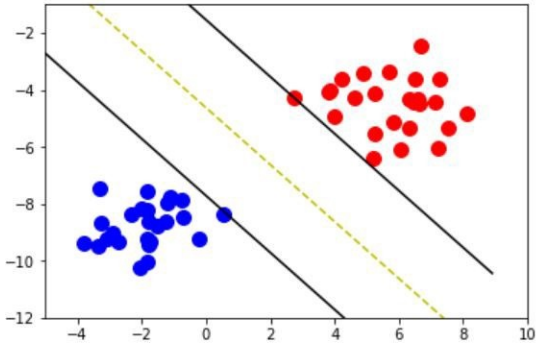

Fig: 3[2]

In addition to performing linear classification, the SVM can also perform non-linear classification by changing its kernel.

Why SVM with Grid Search CV: In handling an imbalanced binary classification task, the Support Vector Machine (SVM) is one of the methods reported to perform better compared to other algorithms such as discriminant analysis and logistic regression. Robustness of SVM's algorithm and ability to integrate with kernel-based learning are the prime strength of SVM which provides flexible analysis and optimized solution. Class-weighted SVM is designed to deal with imbalanced dataset by allocating higher misclassification penalties to training instances of the minority class.
For this assignment, our approach for SVM with GridSearchCV was as follows:

```
1: param_grid ←  {'C':
   [1,10,100,1000], 'gamma':
   [1,0.1,0.001,0.0001], 'kernel':
   ['linear','rbf'] and
   'class_weight':'balanced']}
2: sup_vector ← SVC()
3: for value of score in
   score_index:
   1. grid_cv_svc ←
   2. GridSearchCV(sup_vector,
      param_grid, cv=strati_CV and
      scoring=score)
```

```
    3. grid_cv_svc ← fit the
       grid_cv_svc with X_new and y
    4. return mean precision value ←
       grid_cv_svc.best_score_
4: return the best parameter ←
   grid_cv_svc.best_params_,'\n')
5: best the estimator ←
   grid_cv_svc.best_estimator_
6: return the standard deviation ←
   grid_cv_svc.cv_results_['std_tes
   t_score']
   [grid_cv_svc.best_index_]
```

Random Forest Classifier: Random forest (RF) builds multiple decision trees and consolidate them together to get a more accurate and stable prediction. For a prediction, the trained random forest algorithm uses the below pseudo-code:

1. Takes the test features and use the rules of each randomly created decision tree to predict the outcome and stores the predicted outcome (target)
2. Calculates the votes for each predicted target.
3. Considers the high voted predicted target as the final prediction.

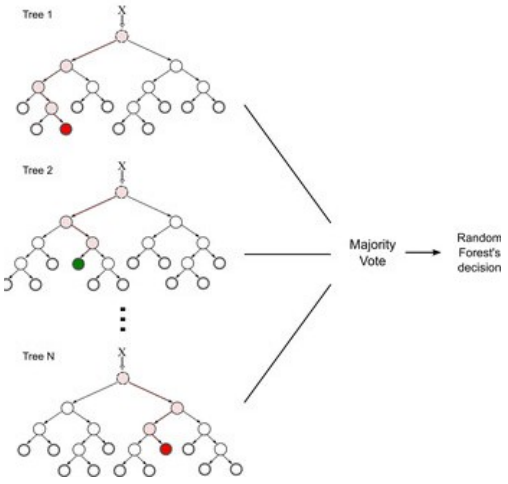

Fig: 4[3]

Why Random Forest Classifier with Grid Search CV: Random forest classifier reduces over-fitting problem when we have more trees in a forest. Another excellent quality of the random forest algorithm is that it is straightforward in deciding the relative importance of each feature. Sci-kit learn provides an excellent tool for this that measures features importance by inspecting at how much the tree nodes, which use that feature, reduce impurity across all trees in the forest. It calculates the score automatically for each feature after training and scales the results so that the sum of all importance is equal to 1.

And our approach for this assignment:

```
1: model←RandomForestClassifier(n_e
   stimators=100,
   criterion='entropy',
   max_depth=5,
   max_features='auto',
   min_impurity_split=None,
   n_jobs=-1, random_state=0 and
   class_weight='balanced')
2: param_grid ← {'n_estimators':
   [100, 200, 700],'max_features':
   ['auto', 'sqrt', 'log2']}
3: for value of score in
   score_metric:
   1: random_forest ←
      GridSearchCV(model,
      param_grid, cv=strati_CV and
      scoring=score)
   2: random_forest ← fit
      random_forest with X_new and
      y
   3: return mean precision value ←
      random_forest.best_score_
4: return the best parameter ←
   random_forest.best_params_,'\n')
5: best the estimator ←
   random_forest.best_estimator_
6: return the standard deviation ←
   random_forest.cv_results_['std_t
   est_score']
   [random_forest.best_index_])
```

Performance Metric Comparison:

Model	Average Precision	Standard Deviation	Best Parameter
K-NN	0.57	0.0718	n_neighbours = 28, weights = distance, metric = manhattan
SVM	0.48	0.053	C = 1, class_weight = balanced, gamma = 0.0001, kernel = rbf
RFC	0.7	0.0602	n_estimators = 700, class_weight = balanced

Table: 01

Conclusion: We presented three different models to handle the provided imbalanced data by tuning up different parameters to get better performance. We used average precision to compare how well out models are ordering the predictions, without considering any specific decision threshold. Based on the highest value of average precision, we can conclude that the Random Forest Classifier is a good model with the provided dataset.

References:

1. Adi Bronshtein (2017) A Quick Introduction to K-Nearest Neighbours Algorithm. [https://medium.com/@adi.bronshtein/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7]. Accessed 7th March, 2019

2. Madhu Sanjeevi (Mady) (2017) Chapter 3: Support Vector machine with Math. [https://medium.com/deep-math-machine-learning-ai/chapter-3-support-vector-machine-with-math-47d6193c82be]. Accessed 7th March, 2019

3. David Carrasco (2017) Random Forest – Modelling The Titanic Voyage with R [https://blog.datatons.com/2017/05/16/random-forest-titanic-voyage/]. Accessed 7th March, 2019