Ensemble Learning

Rohit Chougule

Question 1.

We have the US-census dataset that has about 14 attributes and one class attribute.
To preprocess the dataset, the data needs to handle missing values, and normalization of the data would be required wherever necessary.
The below attributes have missing values:

Attribute	**Percentage of missing values**
workclass	6%
occupation	6%
native-country	2%

The missing values in the above attributes can be handled by replacing the missing values using *ReplaceMissingValue* function from Weka filters in unsupervised category, which replaces missing values with mean and mode from the data.
Once the missing values are handled, the some of the attributes(numeric attributes) need to be normalized to the scale of 0 to 1 using *Normalize* function from Weka filters in unsupervised category, which would bring the numeric attributes to the scale of 0 to 1. The features that would be normalized are- age, fnlwgt, education-num, capital-gain, capital-loss, hours-per-week.

After the data has been normalized, we can look in to remove the outlier values from the dataset, which may improve the accuracy for the classifier.
The data has about 30932 records, out of which approximately 11% (3508 rows) of the data seem to be outlier, which was identified using the *weka.filters.unsupervised.attribute.InterquartileRange -R first-last -O 3.0 -E 6.0.* The Interquartile function is useful in identifying the outlier from the data.
The outlier can then be removed using *weka.filters.unsupervised.instance.RemoveWithValues -S 0.0 -C 16 -L last*
Now, we are left with about 27434 rows in the dataset.

a. KNN (1-NN)

- When the *weka.attributeSelection.InfoGainAttributeEval* was applied on the data to select the attributes for classification(using Filter method), the attributes to be used further are:
  age, education, education-num, marital-status, relationship, capital-gain after selecting the top 6 attributes from the ranked attributes using Information gain.
- The k-NN(1NN Classifier) was applied using the classification function *weka.classifiers.lazy.IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""*
- The 1-NN Classifier resulted in the classification results as below:

```
=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.595    0.108    0.655      0.595    0.623      0.503    0.852     0.665     >50K
                 0.892    0.405    0.865      0.892    0.879      0.503    0.852     0.938     <=50K
Weighted Avg.    0.816    0.329    0.811      0.816    0.813      0.503    0.852     0.868

=== Confusion Matrix ===

    a     b   <-- classified as
 4168  2836 |     a = >50K
 2200 18220 |     b = <=50K
```

- The class of our training set data consists of skewed (imbalanced data) which can be seen from the summary below, the class for >50K constitutes to about 25% of the data, where as the class <=50K forms rest of the 75% of the data.

Ensemble Learning

Rohit Chougule

Name: class
Missing: 0 (0%)
Distinct: 2
Type: Nominal
Unique: 0 (0%)

No.	Label	Count	Weight
1	>50K	7004	7004.0
2	<=50K	20420	20420.0

- As the data is imbalanced, we need to use Balanced Accuracy Rate to evaluate the 1-NN Classifier model. We can use the F-measure which does the tradeoff for precision and recall for the classifier.
- We can use the $\beta = 1$ Harmonic mean of Precision and recall version of the F-measure.
- The F-measure can be evaluated as: $F1 = \frac{2 * Precision * Recall}{Precision + Recall}$ $F1 = \frac{2 * 0.811 * 0.816}{0.811 + 0.816} = 0.813$

Decision Trees- J48:

- The Decision tree classifier was implemented using the function *weka. classifiers.trees.J48 -C 0.25 -M 2* from the Classify tab in Weka, using the same attributes+

```
=== Detailed Accuracy By Class ===

                 TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
                 0.524     0.050     0.783      0.524     0.628      0.549     0.860     0.718     >50K
                 0.950     0.476     0.853      0.950     0.899      0.549     0.860     0.934     <=50K
Weighted Avg.    0.841     0.367     0.835      0.841     0.830      0.549     0.860     0.879

=== Confusion Matrix ===

     a     b   <-- classified as
  3670  3334 |     a = >50K
  1017 19403 |     b = <=50K
```

- For the above decision tree classifier, we again use the F-measure to evaluate the accuracy of the classifier.
- The F-measure can be evaluated as: $F1 = \frac{2 * Precision * Recall}{Precision + Recall}$ $F1 = \frac{2 * 0.835 * 0.841}{0.835 + 0.841} = 0.830$
- From the above F measure values, we can see that by F1 measure evaluation of both the classifiers, the decision tree classifier gives a higher F1-score.

b. Ensembles with bagging.

Resampling the dataset without loss of accuracy:
weka.filters.unsupervised.instance.Resample -S 1 -Z 10.0 -no-replacement

Applying ensembles with Bagging for decision tree classifier & KNN:
Ensemble Size: 20(numIterations)

- *weka.classifiers.meta.Bagging -P 100 -S 1 -num-slots 1 -I 20 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2*
- *weka.classifiers.meta.Bagging -P 100 -S 1 -num-slots 1 -I 20 -W weka.classifiers.lazy.IBk -- -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""*

Ensemble Size: 40(numIterations)

- *weka.classifiers.meta.Bagging -P 100 -S 1 -num-slots 1 -I 40 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2*
- *weka.classifiers.meta.Bagging -P 100 -S 1 -num-slots 1 -I 40 -W weka.classifiers.lazy.IBk -- -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""*

Ensemble Size: 60(numIterations)

- *weka.classifiers.meta.Bagging -P 100 -S 1 -num-slots 1 -I 60 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2*

Ensemble Learning

Rohit Chougule

- *weka.classifiers.meta.Bagging -P 100 -S 1 -num-slots 1 -I 60 -W weka.classifiers.lazy.IBk -- -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""*

Ensemble Size: 80(numIterations)

- *weka.classifiers.meta.Bagging -P 100 -S 1 -num-slots 1 -I 80 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2*
- *weka.classifiers.meta.Bagging -P 100 -S 1 -num-slots 1 -I 80 -W weka.classifiers.lazy.IBk -- -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""*

Ensemble Size 100(numIterations)

- *weka.classifiers.meta.Bagging -P 100 -S 1 -num-slots 1 -I 100 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2*
- *weka.classifiers.meta.Bagging -P 100 -S 1 -num-slots 1 -I 100 -W weka.classifiers.lazy.IBk -- -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""*

Performance of Decision tree and KNN (1NN Classifier with different Bag Sizes)

Ensemble Size	Decision Tree Classification Accuracy	Decision Tree F-Measure	KNN Classification Accuracy	KNN F-Measure
20	83.33%	0.824	80.99%	0.806
40	83.26%	0.824	81.18%	0.808
60	83.44%	0.826	81.47%	0.810
80	83.36%	0.825	81.36%	0.810
100	83.47%	0.826	81.32%	0.809

From the above tabulated data for the ensemble for Decision tree and 1NN Classifier accuracy for various ensemble sizes, it can be seen that the classification accuracy is impacted in a very small scale when going varying ensemble sizes. To be specific about Decision tree classifier, it can be seen that the accuracy of the classifier increases as the number of ensembles(iterations) are increased. However, for the 1NN classifier the classification accuracy increases up to certain ensemble size and then reduces again.

We can select the Decision tree classifier with ensemble size 100 to check the accuracy of the classifier on different bagsize:

Bagsize size percent	Decision Tree Classifier Accuracy (Ensemble Size: 100)
10	83.00%
20	83.51%
30	83.55%
40	83.66%
50	83.58%
60	83.69%
70	83.80%
80	83.44%
90	83.69%
100	83.47%

From the above classifier accuracy table for varying bag size percentage, it can be inferred that the accuracy of the decision tree classifier is the highest with bag size 70%

c. Ensembles with Random Sub-spacing:

Applying ensembles with Random sub-spacing for Decision tree classifier & KNN:

Ensemble Size:20

- *weka.classifiers.meta.RandomSubSpace -P 0.5 -S 1 -num-slots 1 -I 20 -W weka.classifiers.lazy.IBk -- -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""*
- *weka.classifiers.meta.RandomSubSpace -P 0.5 -S 1 -num-slots 1 -I 20 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2*

Ensemble Learning

Rohit Chougule

Ensemble Size: 40

- *weka.classifiers.meta.RandomSubSpace -P 0.5 -S 1 -num-slots 1 -I 40 -W weka.classifiers.lazy.IBk -- -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""*
- *weka.classifiers.meta.RandomSubSpace -P 0.5 -S 1 -num-slots 1 -I 40 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2*

Ensemble Size: 60

- *weka.classifiers.meta.RandomSubSpace -P 0.5 -S 1 -num-slots 1 -I 60 -W weka.classifiers.lazy.IBk -- -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""*
- *weka.classifiers.meta.RandomSubSpace -P 0.5 -S 1 -num-slots 1 -I 60 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2*

Ensemble Size: 80

- *weka.classifiers.meta.RandomSubSpace -P 0.5 -S 1 -num-slots 1 -I 80 -W weka.classifiers.lazy.IBk -- -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""*
- *weka.classifiers.meta.RandomSubSpace -P 0.5 -S 1 -num-slots 1 -I 80 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2*

Ensemble Size: 100

- *weka.classifiers.meta.RandomSubSpace -P 0.5 -S 1 -num-slots 1 -I 100 -W weka.classifiers.lazy.IBk -- -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""*
- *weka.classifiers.meta.RandomSubSpace -P 0.5 -S 1 -num-slots 1 -I 100 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2*

Performance of Decision tree classifier and KNN on different ensemble sizes:

Bag Size	Decision Tree Classification Accuracy	Decision Tree F-Measure	KNN Classification Accuracy	KNN F-Measure
20	82.04%	0.815	82.93%	0.818
40	82.22%	0.818	82.53%	0.814
60	82.15%	0.816	82.82%	0.816
80	82.15%	0.816	82.45%	0.814
100	82.11%	0.816	82.45%	0.814

From the above tabulated data for accuracy for the ensembles for Decision tree and 1NN Classifier using Random sub-spacing, we can see that the accuracy of the ensemble for decision tree classifier is high when the ensemble size is 40, however, on substantially decreasing the ensemble size the accuracy gradually reduces. However, the behavior with KNN (1NN classifier) is a bit different, the accuracy is the highest when the ensemble size is about 20.

The accuracy for the KNN is high, we may hence, select the best performing subspace size from KNN to check the behavior of the classifier.

Performance of the KNN Classification accuracy for different subspace sizes:

Subspace size	KNN Classification Accuracy (Ensemble Size: 20)
0.1	77.16%
0.2	77.16%
0.3	82.27%
0.4	82.27%
0.5	82.93%
0.6	80.78%
0.7	80.78%
0.8	79.39%
0.9	79.39%
1.0	77.16%

On changing the number of features (i.e sub-space size), the classification increases gradually until the subspace size is 0.5, and eventually decreases further.
From this we may infer, that using about 50% percent of the actual features we get the highest accuracy for the KNN classifier, when the ensemble size is 20.
Hence it may be concluded that the accuracy for the Decision tree classifier is better when using Bagging technique, where as KNN is better with random sub-spacing.

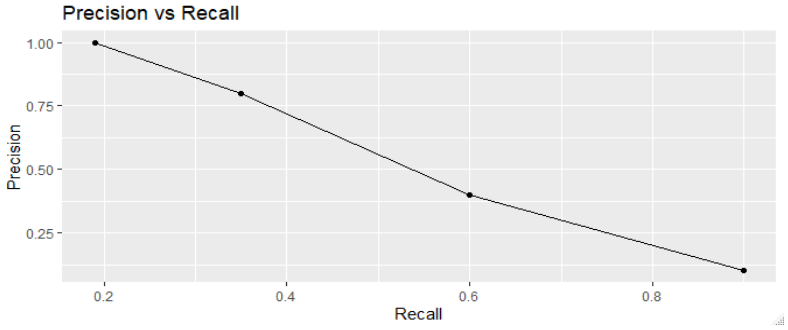
d. Which set of classifiers is expected to benefit from bagging techniques more and which set of classifiers is expected to benefit from random sub-spacing techniques more? For your dataset, determine the best ensemble strategy for each of the two classifiers.

- Bagging may be ineffective when using a classifier like KNN(1NN), in this dataset. For the dataset in this scenario, we can infer from the accuracy of both the classifiers that the Decision tree classifier seems to give more accuracy in the Bagging as well as the Random sub-spacing technique. However, the accuracy for KNN is increased when random sub-spacing as compared to Bagging as Random sub-spacing adds some diversity into the ensemble by utilizing different features when calculating distance in KNN.
- The best ensemble strategy for each of the classifiers:
  Decision Tree Classifier with ensemble size 100, bag size=70%, with an accuracy of 83.80%
  1NN Classification with ensemble size=20, subspace size=0.5, with an accuracy of 82.93%

Question 2.

a. Explain the precision-recall tradeoff.

- Precision & Recall are the evaluation metrics for classification problems which have imbalanced data in Machine Learning. A general description of the Precision and Recall measures in Information retrieval is given as:
  Precision: Proportion of retrieved results that are relevant.
  Recall: Proportion of relevant results that are retrieved.
- For some of the applications, we need to tradeoff between these metrics for evaluation Precision & Recall defined as below, and the TP, FP, FN, TN values can be evaluated using the confusion matrix.



$Precision = \frac{TP}{TP+FP}$ $Recall = \frac{TP}{TP+FN}$,

- We can plot the precision, recall measures using a Precision-Recall PR curve to compare what precision and recall at fixed intervals.
- At times with skewed data, using the Accuracy measure ($Accuracy = \frac{TP+TN}{TP+FP+TN+FN}$) may give misleading results, as they might result in predicting outcome as the majority class. Hence, we need to choose the decision threshold very carefully. In some classification problems, such as medical applications like cancer prediction, the cost of false positive or false negative is relatively very high, due to which we need to be more confident about choosing a decision threshold.
- To select a decision threshold accurately, we need to tradeoff between the precision-recall measures for which we can use F-measure.
- $F = \frac{1+\beta^2 * Precision * Recall}{\beta^2 * Precision + Recall}$ The β parameter here controls the trade-off:
  $\beta > 1$ Focus more on Recall, $\beta < 1$ Focus more on Precision, $\beta = 1$ Harmonic mean of Precision and recall.
- The $\beta = 1$ is the most used variant which is called F1-measure: $F1 = \frac{2 * Precision * Recall}{Precision + Recall}$

Ensemble Learning

Rohit Chougule

- We can move the decision threshold value to higher or lower values to modify the sensitivity of a model.
- The decision threshold value is the key when differentiating between a positive or negative outcome of a classifier.

b. Explain the bias-variance tradeoff.

- The error of a classifier for a given data can consist of three parts:
  1. Bias shows how close the classifiers predicted values are from the actual values.
  2. Variance is the error from sensitivity to small changes in the training set.
  3. Minimum classification error which is the noise or the outlier in the data.

In Machine Learning, we often need to tradeoff between Bias and variance source of the error. Underfitting occurs in a Machine Learning model when a model is unable to understand and capture the trend of the data. The machine learning model does not fit the data well enough to accurately predict the results. Generally, high bias value and low variance value tends to underfit the model. Overfitting is the phenomenon when the model includes the noise or the outlier into the trend of the data which may lead to prediction to be biased towards the majority class. Hence, we need a tradeoff to balance and minimize the bias and variance source of the error, so that the model works well on the testing data and does not predict accurately only on the training dataset.

c. Why do we need activation function in a perceptron?

- A perceptron is a basic component in any neural network, perceptron is a neural network having only one layer, whereas, a network with multiple layers or perceptron's is called a neural network.
- A perceptron works like binary classifier, which helps in classifying the input data. A perceptron consists of several components like Input value, weight and bias, activation functions.
- An activation function is necessary to determine the output for a perceptron as a perceptron results in values between either 0 or 1.
- A single perceptron may be able to handle only linearly separable problems
- However, we can have a continuous output value when have a non-linear activation function.
- An issue with a perceptron is that a very small change in the inputs for the perceptron can have large changes in the output.
- There are different non-linear activation functions to generate continuous output.
- The sigmoid, hyperbolic tangent, and rectifier(ramp) function are examples of non-linear activation functions.

d. What does R^2 measure represent in the case of linear regression?

- Coefficient of determination (R^2) is used to measure the strength of linear relationship.
- it is the fraction of variation that is explained by linear regression model.
- The coefficient of determination (R^2) suggests how well the regression line matches the actual data. The closer the value of R2 to 1, the better is the fit.
- If a fitted linear regression model is defined by:

$$\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_i$$

- The sources of error in a regression model are:
- SST: The total sum of squared deviations in Y from its mean.
- SSR: Total sum of squared deviation of the best fit from the mean. (Variation in Y explained by regression line)
- SSE: The total sum of squared residuals (difference in fit and observed data)

$$SST = SSR + SSE$$

$$R^2 = \frac{SSR}{SST} \; OR \; R^2 = 1 - \frac{SSE}{SST}$$