Design and Implementation of Machine Learning Evaluation Metrics on HPCC Systems

A. Suryanarayanan
Dept. of Computer Science and Engineering
RV College of Engineering
Bengaluru, India
asuryanarayanan.cs17@rvce.edu.in

Arjuna Chala
HPCC Systems
LexisNexis Risk Solutions
Alpharetta, USA
arjuna.chala@lexisnexisrisk.com

Lili Xu
HPCC Systems
LexisNexis Risk Solutions
Alpharetta, USA
lili.xu@lexisnexisrisk.com

Shobha G
Dept. of Computer Science and Engineering
RV College of Engineering
Bengaluru, India
shobhag@rvce.edu.in

Jyoti Shetty
Dept. of Computer Science and Engineering
RV College of Engineering
Bengaluru, India
jyothis@rvce.edu.in

Roger Dev
HPCC Systems
LexisNexis Risk Solutions
Alpharetta, USA
roger.Dev@lexisnexisrisk.com

***Abstract*—The HPCC Systems Production Machine Learning bundles provide a diverse set of features that allow the parallelized creation and training of Machine learning models and a large set of evaluation metrics that can be used to test the trained model to ascertain its performance. To help monitor the models more closely however, a new set of evaluation methods that incorporate the analysis of clusters and the selection of features, as well as other commonly used tests, have been proposed, implemented, and tested. The implementations are written completely in Enterprise Control Language and support the various features provided by the Machine Learning bundles such as the Myriad Interface. This paper provides a comprehensive summary of the evaluation metrics currently available in the library, before presenting the details of the design and implementation of the new evaluation methods. It the goes on to present the results of testing these implementations against implementations present in the python scikit-learn library, and a few data visualisations demonstrating some uses of the implemented evaluation metrics.**

***Index Terms*—HPCC Systems, Machine Learning, Performance Evaluation Metrics**

I. Introduction

The HPCC Systems platform is an open source set of features that provides high-performance, parallel processing and delivery for big data applications. The platform provides predictive modeling functionalities that can be used to perform Machine Learning (ML) operations on Big data. currently the HPCC Systems Production ML Bundles are structured as shown in Table I.

TABLE I
HPCC Systems Production ML Bundles

Category	Bundles
Core bundles	ML_Core
	PBblas
Supervised Learning bundles	Linear Regression
	Logistic Regression
	GLM
	Support Vector Machines
	Learning Trees
Unsupervised Learning	KMeans
Natural Language Processing Bundles	Text Vectors

With regards to evaluation metrics, the supervised learning bundles are quite diverse and feature rich. The KMeans and NLP bundles however, have no evaluation methods to monitor trained models. An overview of the existing evaluation metrics present in different bundles is presented in the paper.

Keeping the existing metrics in mind, the main objective while choosing a new set of evaluation metrics to implement is to include methods for the analysis of unsupervised learning models (clustering), and to add diversity in the number of ways in which the models as well as the data can be monitored (by providing feature selection methods, for example). Some other popular evaluation metrics that are often cited in literature are also included. The proposed metrics that were implemented are detailed in the paper. This is followed by the results of testing of these implementations in various criteria such as features and accuracy, and then by a few data visualisations that demonstrate the use of the evaluation measures.

II. Summary of Existing Metrics

An in-depth summary of the various evaluation metrics present in the HPCC Systems Production ML Bundles, organized bundle-wise, is presented here. A brief overview of the same is provided in Table II.

A. *ML_Core bundle*

1) *Classification:*

a) Class stats: Returns the number and percentage of records that belong to each class in a classification, given any labelled dataset.

b) Confusion matrix: Returns the Confusion Matrix, counting the number of records present for each combination

TABLE II
SUMMARY OF EXISTING METRICS

Module	Type	Metrics
ML_Core	Classification	Class Stats
		Confusion Matrix
		Accuracy
		Accuracy by class
	Regression	Accuracy
Linear Regression	Coefficients	Standard Error
		T-statistic
		P-value
		Confidence Interval
	Model	ANOVA
		R2
		Adjusted R2
		AIC
		F Test
SVM	Classification	Confusion matrix
		N-Fold cross validation
	Regression	N-Fold cross validation
Learning Trees	Boosted regression forests	Accuracy
		Feature Importance
	Classification forests	Accuracy
		Accuracy by class
		Feature Importance
		Decision distance matrix
		Confusion matrix
		Uniqueness factor
	Learning forests	Feature Importance
		Uniqueness factor
		Decision distance matrix
	Regression forests	Feature Importance
		Uniqueness factor
		Decision distance matrix
		Accuracy
Logistic Regression	Classification	Binomial Confusion matrix
		Confusion matrix
		Deviance analysis
		Deviance detail
		Model deviance
		Null deviance
GLM	Regression	Binomial Confusion
		Deviance analysis
		Deviance detail
		Model deviance
		Null deviance
		Confusion matrix

of predicted Class and actual Class, given the actual values and the values predicted by the ML model.

c) Accuracy: Returns the accuracy of the prediction, given the actual values and the values predicted by the ML model. Evaluates the model based on the metrics presented in Table III.

d) Accuracy By Class: Provides class-wise accuracy measures such as Accuracy, Recall, and False Positive Rates, given the actual values and the values predicted by the ML model.

2) Regression:

a) Accuracy: Returns the following metrics about the regression model, given the actual values and the values

TABLE III
ACCURACY MEASURES

Metric	**Explanation**
Error count	Number of misclassified samples
Raw Accuracy	The percentage of samples properly classified (0.0 - 1.0)
PoD	Power of Discrimination. Compares the performance of the classification model against random guessing.
PoDE	Power of Discrimination Extended. Compares the performance of the classification model against always choosing the most frequent class.

predicted by the ML model.

- Mean Squared Error
- Root Mean Squared Error
- R Squared Error

B. Linear Regression

1) Coefficient Evaluation:

a) Standard Error (SE): The standard error is an estimate of the standard deviation of the coefficient. Provides the standard error for each beta coefficient, and each dependent variable in the model. Only meaningful during the training phase.

b) T-statistic: The t-statistic is the coefficient divided by its standard error. A larger absolute value of the T-statistic indicates that the coefficient is more significant.

c) P-value: Tests the null hypothesis that a given coefficient is insignificant. A low P-value indicates that the coefficient is significant. A high P-value indicates that the coefficient is insignificant.

d) Confidence Interval: The Confidence Interval determines the interval within which each estimated coefficient must exist to satisfy a given confidence level that is required

2) Model Evaluation:

a) Analysis of Variance (ANOVA): The analysis of variance report provides an analysis of the various sources of variance produced in the predicted dependent data provided. It details the amount of variance explained by the model, and the unexplained variance due to error. The various metrics provided are shown in Table IV.

TABLE IV
METRICS PROVIDED IN ANOVA

Metric	Explanation
Total_SS	Total Sum of Squares (SS) variance of the dependent data.
Model_SS	The SS variance represented within the model.
Error_SS	The SS variance not reflected by the model.
Total_DF	The total degrees of freedom within the dependent data.
Model_DF	Degrees of freedom of the model.
Error_DF	Degrees of freedom of the error component.
Total_MS	The Mean Square (MS) variance of the dependent data.
Model_MS	MS variance represented within the model.
Error_MS	MS variance not reflected by the model.
Model_F	The F-Test statistic: Model_MS / Error_MS.

b) R squared: Statistical measure of how close the data points are to the fitted regression line.

c) Adjusted R squared: A normalized version of R Squared that does not arbitrarily increase with the number of features

d) F-Test: A measure of the likelihood that all coefficients of the model are insignificant. It indicates whether the regression model created provides a better fit to the data than a model that contains no independent variables.

e) Akaike Information Criterion (AIC): AIC is an Information Theory based criterion for assessing Goodness of Fit. It is used to create a ranking of various models, rather than to provide an absolute measure of quality. A smaller value indicates a better fit.

C. Learning Trees

All of the learning tree models may be evaluated with the relevant evaluation metrics present in the ML_Core bundle. In addition to these which are redefined in this bundle, some other evaluation metrics specific to learning trees are also defined.

1) Boosted Regression Forests:

a) Accuracy: Provides regression accuracy. Equivalent to the regression accuracy metric in ML Core.

b) Feature Importance: Determines the relative importance of features in the decision process of the model.

2) Classification Forests:

a) Accuracy: Equivalent to the classification accuracy metric in ML Core.

b) Accuracy by class: Equivalent to the accuracy by class metric in ML Core.

c) Feature Importance: Determines the relative importance of features in the decision process of the model.

d) Decision distance matrix: Calculates a matrix of distances between data points in Random Forest Decision Space.

e) Confusion matrix: Provides the confusion matrix for the result. Equivalent to the confusion matrix metric in ML Core.

f) Uniqueness Factor: Uniqueness Factor is an experimental metric that determines how far a given point is from a set of other points.

3) Learning Forests:

a) Feature importance
b) Uniqueness Factor
c) Decision distance matrix

4) Regression Forests:

a) Feature importance
b) Uniqueness Factor
c) Decision distance matrix
d) Accuracy

D. GLM (General Linear Model)

1) Binomial Confusion matrix: Computes the confusion matrix for a single response. Work items with multinomial responses are ignored by this function.

2) Confusion matrix: Computes the confusion matrix.

3) Deviance analysis: Analysis of deviance report. It compares the deviance information between two models, one base and the other proposed. It is analogous to the ANOVA report generated for the Least squares model. Metrics measured are:

- Deviance
- Residual deviance
- P-value
- Degrees of freedom
- Residual degrees of freedom

4) Deviance Detail: Deviance detail report. Provides deviance information for each observation.

5) Model deviance: Creates a report of deviance for a model, in the deviance detail format, which includes the metrics:

- Akaike Information Criterion
- Degrees of freedom
- Deviance

6) Null deviance: Return Deviance information for the null model, which is a model with only an intercept.

E. Logistic Regression

Logistic regression has similar metrics as those defined in the GLM bundle, as GLM is a superset of the logistic regression model. These metrics are fine tuned to work better with Logistic regression.

III. Proposed Metrics

The metrics proposed for implementation that could help users monitor models more closely, are presented here. They include intrinsic and extrinsic measures of the performance of clustering models, tests that aid in feature selection, and a few popular benchmarks that are commonly found in literature. A list of these metrics is shown in Table V.

TABLE V
Overview of Proposed Metrics

Module	Metrics
ML_Core / Analysis / Classification	Area Under ROC Curve
ML_Core / Analysis / Classification / AccuracyByClass	F-Score
ML_Core / Analysis / Classification / Accuracy	Hamming Loss
ML_Core / Analysis / Clustering	Adjusted Rand Index
ML_Core / Analysis / Clustering	Silhouette Coefficient
ML_Core / Analysis / FeatureSelection	Chi squared feature test

A. Balanced F-Score

The F-Score, is a statistical measure of a tests accuracy. It is calculated as the weighted harmonic mean of the tests precision and recall. When both the precision and the recall are assigned equal weights, this score is called the balanced F-score. An F-Score of 1 indicates perfect precision and recall. An F-Score of 0 indicates the worst possible precision and recall. The F score can provide a more realistic measure of a tests performance by using both precision and recall.

1) Proposed Implementation: Since both the precision and recall per class are readily available by the use of the AccuracyByClass measure present in the ML_Core bundle, the Balanced F-Score may be directly implemented into this function within the transform that creates the result, by integrating the balanced harmonic mean of the precision and the recall into it. The process of obtaining the Balanced F-Score is detailed in Algorithm 1.

Algorithm 1 The addition of Balanced F-Score into the result structure of AccuracyByClass function.

```
Input: pred : values predicted by the model
Input: actual : actual values
Output: F-Score per class, per classifier, per work-item.
 1: $a \leftarrow AccuracyByClass(pred, actual)$
 2: $prcsn \leftarrow a.precision$
 3: $recll \leftarrow a.recall$
 4: $fscore \leftarrow 2 * prcsn * recll/(prcsn * recll)$
 5: return fscore
```

B. Hamming Loss

Hamming loss is a simple measure that is used to conveniently evaluate multi-label classification models. It is given by the fraction of labels that are incorrectly classified, among all the labels.

a) Proposed implementation: Since the error count is readily available in the Accuracy measure present in the ML_Core bundle, the Hamming Loss can be obtained as the ratio of the error count to the total number of samples. The process of obtaining the Hamming Loss is detailed in Algorithm 2.

Algorithm 2 The addition of Hamming loss into the result of the Accuracy function

```
Input: pred : values predicted by the model
Input: actual : actual values
Output: Hamming Loss per classifier, per work-item
 1: $acc \leftarrow Accuracy$
 2: $err \leftarrow acc.errcnt$ (Number of samples misclassified)
 3: $tot \leftarrow acc.totcnt$ (Total number of samples)
 4: $hamming\_loss \leftarrow err/tot$
 5: return $hamming\_loss$
```

C. Silhouette Coefficient

Silhouette analysis is a way to measure a points closeness to its own cluster as well as its separation from other clusters [1]. It provides an easy way to find the optimum value for k during k-means clustering.

Silhouette values lie in the range of (-1, 1). A value of +1 indicates that the sample point is far away from its neighboring cluster and very close to the cluster it is assigned. Similarly, a value of -1 indicates that the point is closer to its neighboring cluster than to the cluster it is assigned.

1) Computation: The Silhouette score for each individual sample is calculated as follows. If a is the average of the distances of the sample from all other samples in its assigned cluster, and b is the average of the distances of the sample from all points assigned to the closest cluster to the sample, not assigned to it (*Here, the closest cluster is the one that essentially produces the smallest b value*). Once these values are computed, the coefficient is given by (1)

$$S_i = \frac{b_i - a_i}{max(b_i, a_i)} \tag{1}$$

2) Proposed implementation: The proposed implementation takes as input the labels produced by the clustering algorithm and the positions (co-ordinates / features) of the samples. It returns the average silhouette coefficient of all the samples present per cluster, per work item. The implementation involves the creation of two variations. The first variation produces the silhouette score for each individual point, and the second variation gives the average of these scores to produce one score for all the points. The algorithm to produce the Silhouette score per point is detailed in Algorithm 3.

Algorithm 3 Algorithm to produce Silhouette score for each point

```
Input: labels : labels assigned to the points
Input: points : co-ordinates / features of the points
Output: Silhouette score per point, per work-item.
 1: $points \leftarrow$ JOIN $samples, labels$
 2: Find 'a' values :
 3:     $pairs \leftarrow$ All pairs of points in same cluster
        {Pairs is the set of all ordered pairs of points
        that belong to the same cluster}
 4:     $dist \leftarrow$ Distances between these pairs
 5:     $a \leftarrow$ Average($dist$) per left point
 6: Find 'b' values :
 7:     $pairs \leftarrow$ All pairs of points from different clusters
        {Pairs is the set of all ordered pairs of points
        that belong to the different clusters}
 8:     $dist \leftarrow$ Distances between these pairs
 9:     $clust\_ave \leftarrow$ Average($dist$) per cluster, per left point
10:     $b \leftarrow$ min($clust\_ave$) among clusters, per left point
11: $silhouette \leftarrow (b - a)/max(a, b)$
12: return $silhouette$
```

D. Chi Squared Feature Selection Test

The Chi squared test is a statistical hypothesis test where the sampling distribution assumes the form of a χ^2 distribution when the null hypothesis assumed is correct [2]. In the context of feature selection, it may be used to establish the degree of dependence of two categorical variables by analysing the null hypothesis that they are independent of each other. Hence, it can be used to determine whether a classifier is dependent on a certain feature. This test can only be used when both variables are categorical.

1) Contingency Table: The contingency table represents the number of samples present in the data for each combination of sample category and feature category. To start with the chi2 test, a contingency table of the data is first constructed.

2) Chi square value and Null hypothesis testing: In the Chi squared test, we first assume the null hypothesis that the two variables are independent. We then proceed to construct a contingency table that supports this claim, based off of the sums of rows and columns of our existing contingency table [3]. We then calculate the chi square value from (2)

$$\chi^2 = \sum_i \frac{(e_i - o_i)^2}{e_i} \tag{2}$$

For a large population, the probability distribution of this value for a random sampling (small subset) of the population is known to approximately follow a χ^2 distribution, and hence the name of the coefficient. A large value of χ^2 indicates that it is less probable that the sample is not an outlier and is a good representation of the population.

Hence, when the null hypothesis that the sample is independent of a feature produces a large χ^2 value, it indicates that there is a low probability that the null hypothesis is true and hence null hypothesis is rejected and the dependence of the two variables is established. A lower χ^2 value indicates that it is more probable that the null hypothesis is true. It is not known for certain, however the null hypothesis cannot be rejected and the dependence of the variables cannot be established.

3) Proposed implementation: The implementation consists of two separate functionalities. The first part creates a contingency table of the data, and the second finds the chi2 values.

a) Contingency table: The contingency table function takes two different DiscreteField datasets and returns a contingency table for each combination of feature and classifier, per work item. Though a contingency table can be formed between any two categorical variables, the function names its parameters features and samples for convenience.

Algorithm 4 Algorithm to produce contingency table

```
Input: features : Independent data
Input: samples : Dependent data
Output: Contingency table per combination of classifier and
    feature, per work-item
1: $comb \leftarrow$ JOIN $samples, features$
2: $grouped \leftarrow$ GROUP $comb$ by samples, features
3: $contingency \leftarrow$ COUNT $grouped$ per group
4: return $contingency$
```

b) Chi Squared Test: The chi squared feature test implementation takes the independent (features) and dependent (samples) data as parameters, and returns the chi squared coefficient value and the number of degrees of freedom for each combination of classifier and feature.

Equation (2) is used to calculate the chi square coefficient value from the expected and actual values, in Line 16 of Algorithm 5.

Equation (3) is used to compute the entries of the expected contingency table mentioned at Line 9 of Algorithm 5.

$$E_{ij} = \frac{\sum_i O_{ij} \cdot \sum_j O_{ij}}{\sum_i \sum_j O_i j} \tag{3}$$

Equation (4) is used to calculate the number of degrees of freedom of the data, mentioned in Line 15 of Algorithm 5.

$$dof = (rows - 1) \cdot (cols - 1) \tag{4}$$

Algorithm 5 Algorithm to perform Chi squared feature selection test

```
Input: features : Independent data
Input: samples : Dependent data
Output: Chi Squared value and P - value per combination of
    classifier and feature, per work-item
1: $cont \leftarrow Contingency(samples, features)$
2: $f\_group \leftarrow$ GROUP $cont$ by feature
3: $f\_sums \leftarrow$ SUM($f\_group$) per group
4: $s\_group \leftarrow$ GROUP $cont$ by classifier
5: $s\_sums \leftarrow$ SUM($f\_group$) per group
6: $all\_group \leftarrow$ GROUP $cont$ by feature and classifier
7: $all\_sums \leftarrow$ SUM($all\_group$) per group
8: $et \leftarrow$ Expected contingency table using f_sums, s_sums,
    all_sum
9: $rows\_group \leftarrow$ GROUP $cont$ per feature
10: $rows \leftarrow$ COUNT $rows\_group$ per group
11: $cols\_group \leftarrow$ GROUP $cont$ per classifier
12: $cols \leftarrow$ COUNT $cols\_group$ per group
13: $dof \leftarrow (rows - 1) * (cols - 1)$
14: $\chi^2 \leftarrow (cont - et)^2 / et^2$
15: $p \leftarrow cdf(\chi^2, dof)$
16: return $(\chi^2, p)$
```

E. Adjusted Rand Index

The rand index in data clustering is a statistical measure of similarity between two data clusterings [4]. The rand index however, does not necessarily produce a lower score for a random labelling of data. The Adjusted Rand Index is the corrected-for-chance version of the rand index, that uses the expected rand index value to correct the score. Adjusted Rand Index can assume values between -1 and 1. A value of 1 indicates perfectly matching clusterings, and a lower value indicates mismatched clusterings.

As an evaluation metric, the Adjusted Rand Index may be used to compare the values produced by a clustering algorithm to the actual values (ground truth), making ARI an extrinsic evaluation method.

1) Computation: The Adjusted Rand Index is calculated from the contingency table of the data as shown in (5), where n_{ij} is the element on the i^{th} row and j^{th} column, a_i is the sum of the i^{th} row, b_j is the sum of the j^{th} column, and n is the number of samples.

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}]/\binom{n}{2}}{\frac{1}{2}[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}] - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}]/\binom{n}{2}} \quad (5)$$

2) Proposed Implementation: The implementation takes as parameters the two different labellings / clusterings, and returns the Adjusted Rand Score per work-item. It involves finding the required sums from the contingency table of the data and combining them to form the ARI as per (5). The algorithm to find the ARI from the contingency table of the data is detailed in Algorithm 6.

Algorithm 6 Algorithm to obtain ARI from two different data clusterings

```
Input: labels1 : First set of labels
Input: labels2 : Second set of labels
Output: The Adjusted Rand Score per work-item
 1: $cont \leftarrow Contingency(labels1, labels2)$
 2: $row\_grouped \leftarrow$ GROUP $cont$ by row
 3: $row\_sums \leftarrow$ SUM($row\_grouped$) per group
 4: $row\_C2 \leftarrow \binom{row\_sums}{2}$
 5: $col\_grouped \leftarrow$ GROUP $cont$ by column
 6: $col\_sums \leftarrow$ SUM($col\_grouped$) per group
 7: $col\_C2 \leftarrow \binom{col\_sums}{2}$
 8: $all\_C2 \leftarrow \binom{cont}{2}$
 9: $n_{ij} \leftarrow$ SUM($all\_C2$)
10: $nC2 \leftarrow \binom{SUM(cont)}{2}$
11: $a \leftarrow$ SUM($row\_C2$)
12: $b \leftarrow$ SUM($col\_C2$)
13: $ari \leftarrow (n_{ij} - a*b/n)/(0.5(a+b) - a*b/n)$
14: return $ari$
```

F. Area Under Receiver Operating Characteristic Curve

The Receiver Operating Characteristic (ROC) Curve is a graphical plot that indicates the ability of a binary classifier to differentiate between two classes [5]. The curve is created by plotting the True Positive Rate (TPR) vs the False Positive Rate (FPR) of the classifier its threshold is varied from 0 to 1. (*When a classifier produces scores indicating the probability that a sample belongs to a particular class, the probability threshold is the probability above which a sample must be scored so that the sample is classified as positive.*)

The Area Under the ROC Curve (AUC) is a measure of the classifier's ability to distinguish between classes. When using normalized units, this area is also numerically equal to the probability that a random positive sample is scored higher than a random negative sample [5].

The AUC is an evaluation measure for binomial classification. It can, however, be extended to evaluate multi-class classifiers, by evaluating them in a one vs. all fashion or a one vs. one fashion.

1) Proposed Implementation: The implementation takes as parameters the actual labels of the dependent data and the predicted scores (per point, per class) produced by the classifier. The implementation supports multi-class classification by arranging the classes in a one vs. all fashion. The AUC is obtained as the probability that a random positive sample is scored higher than a random negative sample. The algorithm to obtain the AUC from the data provided is detailed in Algorithm 7.

Algorithm 7 Algorithm to obtain AUC from the actual labels and the predicted scores

```
Input: actual : Actual labels
Input: pred : Predicted scores per sample, per class
Output: AUC per class, per classifier, per work-item
 1: $comb \leftarrow$ JOIN $actual, pred$ with same id, classifier, class
      WHERE $isTrue \leftarrow true$ IF $class = actualclass$
 2: $pairs \leftarrow$ JOIN $comb, comb$ with same classifier, class
      WHERE $left.isTrue$ = true and $right.isTrue$ = false
      WHERE
        IF $left.score > right.score$:
          $pairs.score \leftarrow 1.0$
        ELSE IF $left.score = right.score$:
          $pairs.score \leftarrow 0.5$
        ELSE:
          $pairs.score \leftarrow 0.0$
 3: $auc \leftarrow$ AVERAGE($pairs.score$) per class, per classifier
 4: return auc
```

IV. Testing

To test the evaluation metrics implemented, the results of these functions were compared with the results produced by their corresponding implementations in the popular python machine learning library, Scikit-Learn [6]. The tests include various criteria such as multiple work-item, multiple classifiers, etc. To conduct the tests, randomised data was generated and fed to both implementations[1]. Testing indicates that the results produced by this implementation matches those produced by sklearn to a high degree of precision. The results of these tests are presented here.

A. Silhouette Coefficient

The test produces 100 random samples and labels per work-item for 10 work-items. These are then prepared and fed to the ECL and sklearn implementations. Test results are displayed in Table VI

B. Adjusted Rand Index

The test produces two sets of 100 random labelling per work-item, for 10 work-items. These two sets of labels are used as input for ARI. Test results are presented in Table VII. It is seen that the ARI values are close to 0, as is characteristic of ARI for random labelling.

[1]The sklearn functions were called using a python embedding in ECL

TABLE VI
SILHOUETTE COEFFICIENT TEST RESULTS

wi	ml_core	sklearn
1	-0.03499985277281574	-0.03499985277281574
2	-0.04373078961366114	-0.04373078961366117
3	-0.05427326202883921	-0.05427326202883918
4	-0.04114268392204368	-0.0411426839220437
5	-0.03865472959848591	-0.03865472959848589
6	-0.04478419957147606	-0.04478419957147603
7	-0.04834040668062919	-0.04834040668062916
8	-0.04504062512850892	-0.04504062512850892
9	-0.03757730044761803	-0.03757730044761803
10	-0.03556636932743301	-0.03556636932743307

TABLE VII
ARI TEST RESULTS

wi	ml_core_ari	sklearn_ari
1	-0.004141934076345619	-0.004141934076345619
2	0.000425714333552275	0.000425714333552275
3	0.003105967798419173	0.003105967798419173
4	-0.003734508312867772	-0.003734508312867772
5	0.002285490924763328	0.002285490924763328
6	0.009869712322321729	0.009869712322321729
7	0.003147931816092201	0.003147931816092201
8	0.01253929199009587	0.01253929199009587
9	-0.001166232167691474	-0.001166232167691474
10	0.002469761987977875	0.002469761987977875

C. Area Under ROC Curve

The test produces 100 samples with 3 different classifiers and 4 different classes, per work-item for 2 work-items. Test results are displayed in Table VIII. It is seen that the AUC values are close to 0.5, as expected of a random classification.

D. Chi squared feature test

The test produces a dependent (classifier) variable set of 100 samples with 3 classifiers and 4 classes per work-item, for 2 work-items, and a similar independent (feature) variable set with 3 features (instead of classifiers). These two sets are used as input to the chi squared feature test. Test results are presented in Table IX.

V. VISUALISATIONS

To demonstrate the use of these evaluation metrics, a few data visualisations performed on the results of analysis of real world datasets are presented.

A. Silhouette score

The Silhouette score can be used to find the inherent quality of the clustering results without the use of ground truth values. Hence, the score can be used to find the optimal value of K in K-Means clustering by selecting the model which has the highest average silhouette score. Fig. 1 shows the variation of the average silhouette score against variation in K value. Fig. 2 shows the silhouette scores per sample when K = 2.

TABLE VIII
AUC TEST RESULTS

wi	classifier	class	ml_core	sklearn
1	1	0	0.5448284823284824	0.5448284823284824
1	1	1	0.4767732596394262	0.4767732596394262
1	1	2	0.4615491586705604	0.4615491586705604
1	1	3	0.5232666666666667	0.5232666666666667
1	2	0	0.4768685955394816	0.4768685955394817
1	2	1	0.5083425346583241	0.5083425346583241
1	2	2	0.4790710610485329	0.4790710610485329
1	2	3	0.4982242516489092	0.4982242516489092
1	3	0	0.5137202525497815	0.5137202525497815
1	3	1	0.5066728025770824	0.5066728025770824
1	3	2	0.4696147962731192	0.4696147962731192
1	3	3	0.5645611364789447	0.5645611364789447
2	1	0	0.4770519669742593	0.4770519669742594
2	1	1	0.4994556341861731	0.4994556341861731
2	1	2	0.5496323529411765	0.5496323529411764
2	1	3	0.5640860215053763	0.5640860215053762
2	2	0	0.5763720252549781	0.5763720252549781
2	2	1	0.4282871828884099	0.4282871828884099
2	2	2	0.4531510717494545	0.4531510717494545
2	2	3	0.5320945945945946	0.5320945945945946
2	3	0	0.5729780564263323	0.5729780564263323
2	3	1	0.4291958041958042	0.4291958041958042
2	3	2	0.5132688492063492	0.5132688492063492
2	3	3	0.4643010752688172	0.4643010752688173

TABLE IX
CHI SQUARED TEST RESULTS

wi	fnumber	snumber	ML_Core	sklearn
			x2	*x2*
1	1	1	12.43326734488751	12.43326734488751
1	1	2	8.77679914973071	8.776799149730714
1	1	3	3.793700963978115	3.793700963978115
1	2	1	8.911347003011189	8.911347003011187
1	2	2	16.08060104987661	16.08060104987662
1	2	3	12.4372971386786	12.4372971386786
1	3	1	6.767299970525701	6.767299970525702
1	3	2	4.619610607791617	4.619610607791617
1	3	3	3.062009813936293	3.062009813936293
2	1	1	7.969163931684418	7.969163931684418
2	1	2	7.608984158778655	7.608984158778657
2	1	3	15.05414651311763	15.05414651311763
2	2	1	13.22877370090364	13.22877370090364
2	2	2	9.759613685378492	9.759613685378492
2	2	3	7.630526461140101	7.630526461140102
2	3	1	9.186606685605684	9.186606685605682
2	3	2	15.42328296912561	15.42328296912561
2	3	3	18.2292779233544	18.2292779233544

B. Area Under Curve

The area under the curve can be used to study the performance of a classification model. The chart in Fig. 3 shows the variation in AUC as the number of training samples increases.

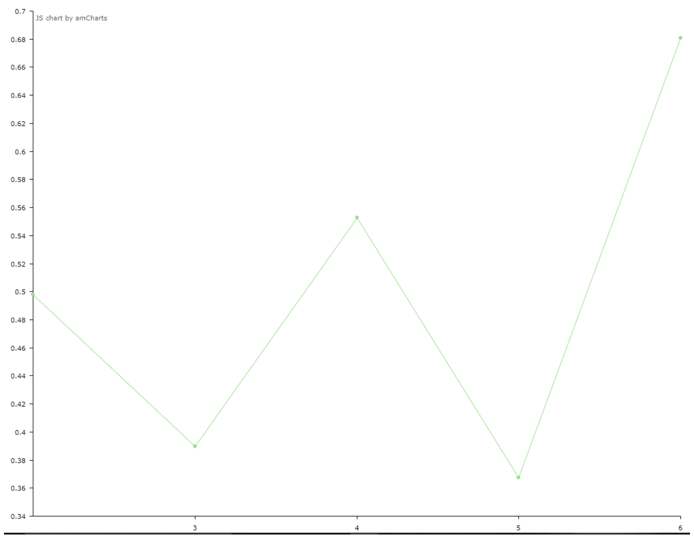

Fig. 1. Average Silhouette score for different values of K in K-Means clustering

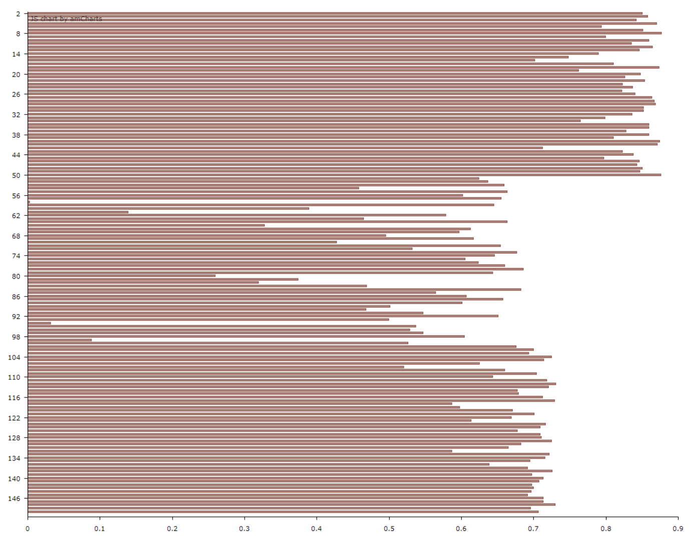

Fig. 2. Silhouette scores per sample for K = 2 in K-Means clustering

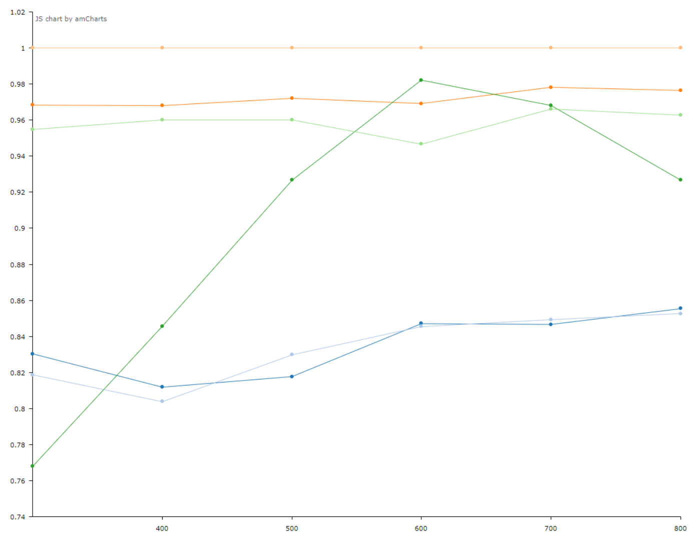

Fig. 3. Change in AUC for different number of training samples used for training the model

C. Adjusted Rand Index

The Adjusted Rand Index (ARI) may be used to compare different clustering models by comparing their results. The visualisation in Fig. 4 shows the comparison between different models, which differ in their K-values in K-Means clustering. The ARI is also used to compare results with the ground truth to check the accuracy of the result. The visualisation in Fig. 5 shows the comparison of the ground truth with models of differing K values in K-Means clustering.

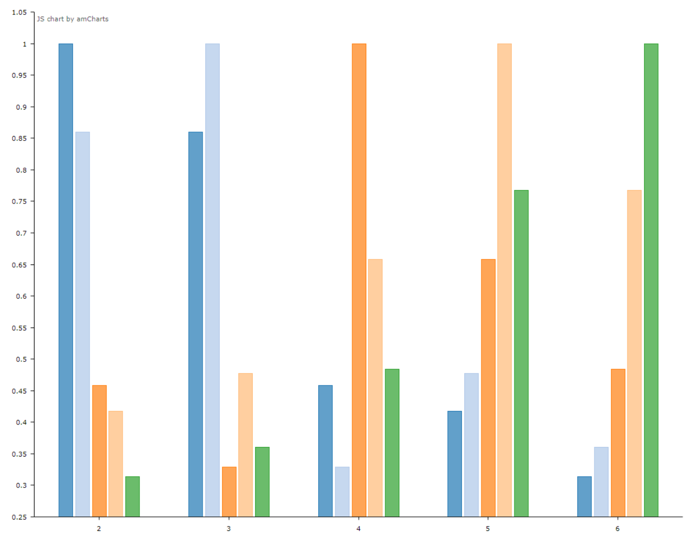

Fig. 4. ARI showing comparison between different models which differ in their K-values

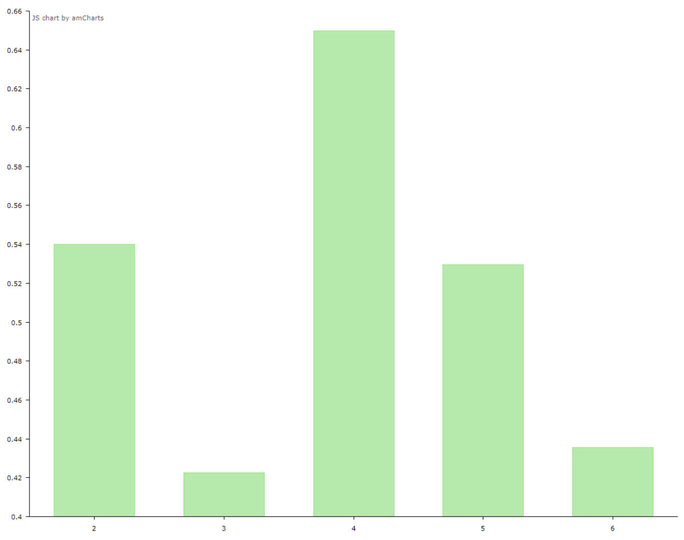

Fig. 5. ARI showing comparison of models with different K-values, with the ground truth labels

VI. Conclusion

The six evaluation metrics for machine learning models chosen for implementation for the HPCC Systems Production Machine Learning Bundles were successfully implemented and tested. These metrics can help users monitor their Machine learning models better and can help tweak their hyperparameters. The implementations also support the various features of the production ML bundles such as the Myriad interface.

References

[1] P. J. Rousseeuw, “Silhouettes: A graphical aid to the interpretation and validation of cluster analysis,” *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53 – 65, 1987. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0377042787901257

[2] K. P. F.R.S., “X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling,” *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 50, no. 302, pp. 157–175, 1900. [Online]. Available: https://doi.org/10.1080/14786440009463897

[3] R. A. Fisher, “On the Interpretation of 2 from Contingency Tables, and the Calculation of P,” Jan. 1922. [Online]. Available: https://doi.org/10.2307/2340521

[4] W. M. Rand, “Objective criteria for the evaluation of clustering methods,” *Journal of the American Statistical Association*, vol. 66, no. 336, pp. 846–850, 1971. [Online]. Available: http://www.jstor.org/stable/2284239

[5] T. Fawcett, “An introduction to roc analysis,” *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861 – 874, 2006, rOC Analysis in Pattern Recognition. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S016786550500303X

[6] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, “API design for machine learning software: experiences from the scikit-learn project,” in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.