Big Data Analytics

Dr Sirintra Vaiwsri | Email: sirintra.v@itm.kmutnb.ac.th

Classification

Classification (Chambers and Zaharia, 2018; Guller, 2015)

- Classification is a common type of supervised learning algorithm.
- It is a training algorithm for predicting a categorical label.
- Classification tasks can be grouped into:
    - Binary classification
    - Multiclass classification
    - Multilabel classification

Binary Classification

(Chambers and Zaharia, 2018; Guller, 2015)

- Binary classification is the most common.
- It classifies an observation into two labels which are positive and negative.
- Example:
  - Classify emails into a spam or not spam emails.

Multiclass Classification

(Chambers and Zaharia, 2018; Guller, 2015)

- Multiclass classification classifies an observation into one label where the label is chosen from more than two labels.
- Example:
  - Predicting the weather from rainy, sunny, cloudy, etc. labels

Multilabel Classification

(Chambers and Zaharia, 2018; Guller, 2015)

- Multilabel Classification predicts more than one label for the same observations.
- In other words, an input can produce multiple labels.
- Example:
  - A news article can be labelled as business and IT.

Classification Evaluation Example

(Guller, 2015; Medium, 2023)

- Accuracy refers to the number of correctly classified over the total number of data.
- It is calculated as:

$$Accuracy = \frac{TN + TP}{TN + FP + TP + FN}$$

where TN is true negative, TP is true positive, FN is false negative, and FP is false positive.

Classification Evaluation Example

(Guller, 2015; Medium, 2023)

- Precision refers to the number of actual positives over the total predicted positives.
- It is calculated as:

$$Precision = \frac{TP}{TP + FP}$$

Classification Evaluation Example

(Guller, 2015; Medium, 2023)

- Recall refers to the number of positives over the total actual positives.
- It is calculated as:

$$Recall = \frac{TP}{TP + FN}$$

Classification Evaluation Example

(Guller, 2015; Medium, 2023)

- F1 measure is the harmonic mean of precision and recall. It shows a balance between precision and recall.
- It is calculated as:

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Classification (Chambers and Zaharia, 2018; Guller, 2015)

- Classification tasks in supervised learning algorithms are such as:
    - Logistic Regression
    - Decision Trees
    - Support Vector Machine (SVM)
    - Ensembles

Logistic Regression (Chambers and Zaharia, 2018)

- Logistic Regression combines inputs (features) with the weights (coefficients) to get the probability of the class.
- The weights help to represent the importance of the feature.
    - Large weight means the variations of features significantly affect the outcome.
    - Smaller weight means the feature is less important.
- SparkML for Logistic Regression can use the same set of parameters as the Linear Regression.

Logistic Regression Implementation

```
# import library SparkSession
# import StringIndexer and VectorAssembler from pyspark.ml.feature
# import LogisticRegression from pyspark.ml.classification
# import MulticlassClassificationEvaluator from pyspark.ml.evaluation
# import Pipeline from pyspark.ml
```

Logistic Regression Implementation

```
# Create SparkSession
# Load data file into Dataframe (FBLiveTH)
# Use StringIndexer to prepare data where the
columns status_type and status_published are used
as an input to create indexes (status_type_ind and
status_published_ind), then fit and transform
# Use VectorAssembler to create vector of
status_type_ind and status_published_ind resulting
in the column features
```

Logistic Regression Implementation

```
# Create logistic regression where the status_type_ind is used as a label and setMaxIter (set maximum iteration), setRegParam (set regularisation parameter [0…1]), and setElasticNetParam (set ElasticNet parameter [0…1]) are also used

# Create a pipeline where the stages include output from the vector assembler and the created logistic regression
```

Logistic Regression Implementation

```
# Create train and test datasets using
randomSplit function where the output from
string indexer is used as an input
# Fit train data into the created pipeline to
create the pipeline model
# Use the created pipeline model to transform
test data resulting in the predictions Dataframe
# Show 5 rows of the predictions Dataframe
```

Logistic Regression Implementation

```
# Use MulticlassClassificationEvaluator to
create the evaluator where the status_type_ind
is used as the label column and prediction
column is used as the prediction column

# Show accuracy (evaluator.evaluate(predictions,
{evaluator.metricName: “accuracy”})), precision
(metricName: “weightedPrecision”), recall
(metricName: “weightedRecall”), and F1 measure
(metricName: “f1”)
```

Decision Trees (Chambers and Zaharia, 2018)

- Decision Trees can be used for both regression and classification.
- Decision trees for regression create output that is a single number per leaf node.
- Decision trees for classification create output that is a label per leaf node.
- It simply creates a big tree of decisions.
- Thus, it is easy for decision making.
- However, it has high cost consumption.

Decision Tree Classification Implementation

```
# import library SparkSession
# import StringIndexer, VectorAssembler, and OneHotEncoder from pyspark.ml.feature
# import DecisionTreeClassifier from pyspark.ml.classification
# import MulticlassClassificationEvaluator from pyspark.ml.evaluation
# import Pipeline from pyspark.ml
```

Decision Tree Classification Implementation

```
# Create SparkSession

# Load data file into Dataframe (FBLiveTH)

# Use StringIndexer to prepare data where the columns
status_type and status_published are used as inputs
to create indexes (status_type_ind and
status_published_ind)

# Use OneHotEncoder to create Boolean flag of
status_type_ind and status_published_ind as they will
be used as a component of vector in the next steps.
```

Decision Tree Classification Implementation

```
# Use VectorAssembler to create vector of encoded status_type_ind and status_published_ind resulting in the column features

# Create pipeline where the stages include output from string indexer, encoder, and vector assembler

# Fit Dataframe into the created pipeline to create the pipeline model

# Use the created pipeline model to transform the Dataframe data resulting in another Dataframe
```

Decision Tree Classification Implementation

```
# Create train and test datasets using randomSplit function where the output from the previous step is used as an input

# Create decision tree classification where the status_type_ind is used as a label and the output is features

# Fit train data into the created decision tree to create the model

# Use the created model to transform the test data resulting in a prediction Dataframe
```

Decision Tree Classification Implementation

```
# Use MulticlassClassificationEvaluator to create the evaluator where the status_type_ind is used as the label column and prediction column is used as the prediction column

# Show accuracy, precision, recall (metricName: “weightedRecall”), and F1 measure (metricName: “f1”)

# Also show the Test Error where it is calculated as 1.0 - accuracy
```

Assignment (2 points)

- Please implement the logistic regression and show the result to get 1 point.
- Please implement the decision tree classification and show the result to get 1 point.

References

- Chambers, B., & Zaharia, M. (2018). *Spark: The definitive guide: Big data processing made simple.* " O'Reilly Media, Inc.".
- Guller, M. (2015). Big data analytics with spark.
- Medium. https://medium.com. Accessed: 2023-09-12.