

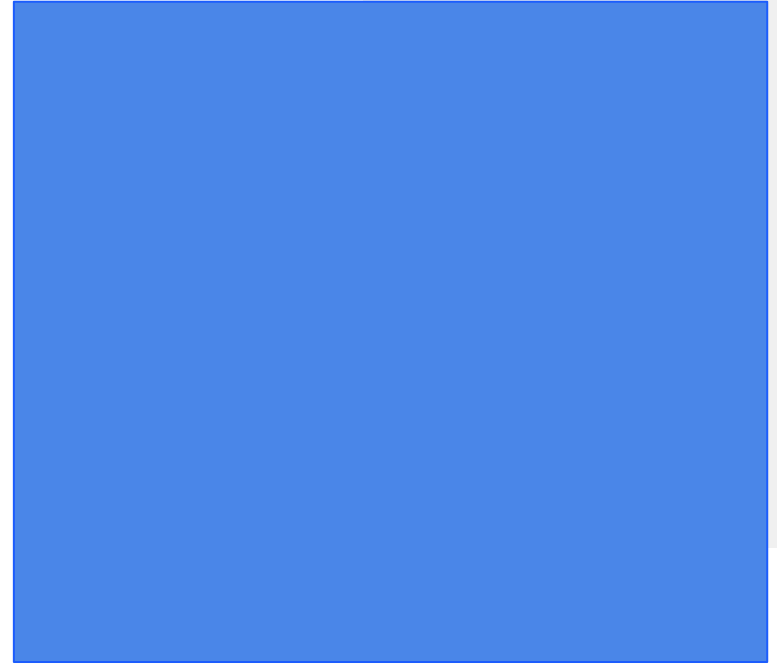


Big Data Analytics

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



Regression




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

- Regression is one of the supervised learning algorithms.
- Regression is used to predict a continuous variable (real number) from a set of features.
- Thus, it is a training algorithm for predicting a numerical label.
- For example, from a number of reactions, you might want to predict how many people are likely to love the FB live.
- A model trained by a regression algorithm:
 - Simple regression - it involves one label and one feature.
 - Multiple regression - it involves one label and multiple features.
 - Multivariate regression - it involves multiple labels and multiple features.



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

- Regression tasks in supervised learning algorithms are such as:
 - Linear Regression
 - Decision Tree
 - Ensembles
- 

Linear Regression (Chambers and Zaharia, 2018; Guller, 2015)

- Linear regression uses training data to fit the linear model with a coefficient.
- It is used for predicting the relationship between independent variable (feature) and dependent variable (label)
- For example,
 - A linear model is $y = a + bx$, where y is the label, x is a feature, a is the intercept of the line, and b is the linear regression coefficient.
 - As we use a training dataset, y , a , b , and x are known, thus, we can train the model to predict the label y .

Linear Regression

(Chambers and Zaharia, 2018; Guller, 2015; Geeksforgeeks, 2023)

- Model hyperparameters:
 - *family* - it can be binary (binary classes) or multinomial (multi-classes)
 - *elasticNetParam* - it is a floating point in the range of 0 to 1 which is used to specify the mix of L1 and L2 regularisation. L1 regularisation mostly creates the value of 0. L2 regularisation creates a value reaching to 0 but not completely 0. *elasticNetParam* should be tuned by testing with different values.
 - *fitIntercept* - it determines whether or not the model should fit an intercept. It can be True or False.

Linear Regression

(Chambers and Zaharia, 2018; Guller, 2015; Geeksforgeeks, 2023)

- *regParam* - it determines how much weight or coefficient to assign for the regularisation where the value must be greater than or equal to 0.
- *standardization* - it can be True or False depending upon whether or not the inputs were standardised before being used in the model.
- Note: Regularisation is a technique for minimising the loss function (quantifies the difference between predicted and actual values) and preventing overfitting (model learns from noise as trained with too much data) or underfitting (model is unable to learn the training data resulting in low performance).

Linear Regression (Chambers and Zaharia, 2018; Guller, 2015)

- Training parameters:
 - *maxIter* - it is the total number of iterations, where the default is 100.
 - *tol* - it specifies a threshold and stops the iterations before the number of iterations reaches the *maxIter*.
 - *weightCol* - it is the name of the weight or coefficient column. You can weigh the labels you know are correct more than the labels you do not know.



Linear Regression

(Chambers and Zaharia, 2018; Guller, 2015)




- Prediction parameters:
 - *threshold* - it is a probability threshold that identifies when to predict a class. It is a Double type within the range of 0 and 1.
 - *thresholds* - it is an array of threshold values used for multiclass.



Linear Regression Evaluation Example

(Simplilearn, 2023; Towardsdatascience, 2023)

- Mean Squared Error (MSE) is the average of the squared difference between the predicted and actual values.
 - The smaller the MSE, the better because it means the model produces fewer errors, where the errors are measured by the dispersion of the data points from its mean).
- 

Linear Regression Evaluation Example

(Simplilearn, 2023; Towardsdatascience, 2023)

- R-Squared (R^2) or Coefficient of Determination is the measure of the strength of the relationship between the model and dependent variable (label).
 - A high R^2 means the model is good.
 - Low R^2 means the model is not better than the mean value.
 - Negative R^2 means the model is worse than the mean value.

Linear Regression Implementation

```
# import library SparkSession
# import StringIndexer and VectorAssembler from
pyspark.ml.feature
# import LinearRegression from
pyspark.ml.regression
# import RegressionEvaluator from
pyspark.ml.evaluation
# import Pipeline from pyspark.ml
```

Linear Regression Implementation

```
# Create SparkSession
# Load data file into Dataframe (FBLiveTH)
# Use StringIndexer to prepare data where the
# columns num_reactions and num_loves are used as
# input to create indexes (num_reactions_ind and
# num_loves_ind), then fit and transform
# Use VectorAssembler to create vector of
# num_reactions_ind and num_loves_ind resulting in
# the features
```

Linear Regression Implementation


```
# Create linear regression where the  
num_loves_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  
linear regression
```



Linear 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
```



Linear Regression Implementation

```
# Use RegressionEvaluator to create the  
evaluator where the num_loves_ind is used as the  
label column and the prediction column is used  
as the prediction column
```

```
# Show MSE
```

```
(evaluator.setMetricName("mse").evaluate(predictions) and R2
```

```
(evaluator.setMetricName("r2").evaluate(predictions)
```


Linear Regression Implementation

To show plot:

```
# Import seaborn
# Import IntegerType from pyspark.sql.types
# Import matplotlib.pyplot
# Import col and desc from pyspark.sql.functions
```



Linear Regression Implementation

```
# Select num_loves and prediction column where  
the values in these columns are converted to  
IntegerType and order by the column prediction  
in descending order
```

```
# Use selected data as data and convert it  
toPandas, then use lmplof function of seaborn  
where x is num_loves and y is prediction column
```

```
# Use matplotlib to show the plot
```



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 Regression Implementation

```
# import library SparkSession
# import StringIndexer, VectorAssembler, and
OneHotEncoder from pyspark.ml.feature
# import DecisionTreeRegressor from
pyspark.ml.regression
# import RegressionEvaluator from
pyspark.ml.evaluation
# import Pipeline from pyspark.ml
```

Decision Tree Regression Implementation

```
# Create SparkSession
# Load data file into Dataframe (FBLiveTH)
# Use StringIndexer to prepare data where the columns
num_reactions and num_loves are used as inputs to
create indexes (num_reactions_ind and num_loves_ind)
# Use OneHotEncoder to create Boolean flag of
num_reactions_ind and num_loves_ind as they will be
used as a component of a vector in the next steps.
```

Decision Tree Regression Implementation

```
# Use VectorAssembler to create vector of encoded  
num_reactions_ind and num_loves_ind resulting in the  
column features  
  
# Create a 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 Regression Implementation

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

```
# Create decision tree regression where the  
num_loves_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 RegressionEvaluator to create the  
evaluator where the num_loves_ind is used as the  
label column and the prediction column is used  
as the prediction column
```

```
# Show R2  
(evaluator.setMetricName("r2").evaluate(predictions))
```


Assignment (2 points)

- Please implement the linear regression and show the result to get 1 point.
- Please implement the decision tree regression 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.
 - Geeksforgeeks. <https://www.geeksforgeeks.org>. Accessed: 2023-09-07.
 - Simplilearn. <https://www.simplilearn.com>. Accessed: 2023-09-07.
 - Towardsdatascience. <https://towardsdatascience.com>. Accessed: 2023-09-07.
- 