MONASH University

Week 6

FIT5202 Big Data Processing

Classification Models

Updated by CM Ting – April 2025

Week 6 Agenda

- Week 5 Review
- Classification Algorithms
    - Decision Trees, Random Forest and Logistic Regression
- Model Evaluation
    - Confusion Matrix
    - ROC Curve
- Tutorial Use Case
    - Bank : Will customers subscribe?

Random Forest

- Use ensemble approach
  - The outcome of the model
    - Majority voting (mode) (for classification)
    - Mean of all outcomes (for regression)
- Generalise the model
  - Build multiple different (uncorrelated) trees
  - Avoid overfitting issue found in decision tree
- Use generalisation technique
  - Bagging (bootstrapping) – Randomise (with replacement) a different dataset (from the training dataset) used for training each tree.
  - Each tree uses a random subset of features for splitting nodes.

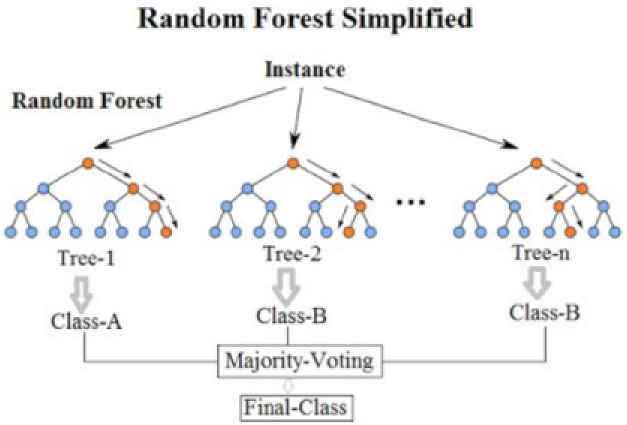
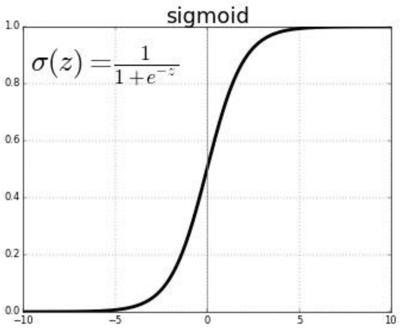


image: https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d

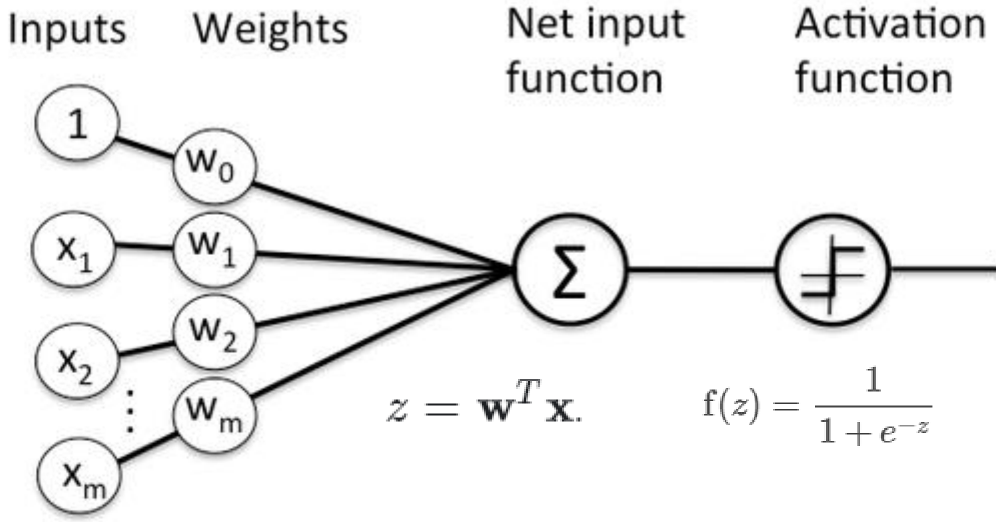
Binary Logistic regression

- Model for binary classification

  y = 0: (<50K)
  y = 1: (>50K)



Classification threshold



Work_class

education

race

y = 1: (>50K)

>0.5

0.95

Probability of y=1 (>50K)

<0.5

y = 0: (<50K)

Linear regression with multiple input variables

MONASH University

Performance Metrics

Confusion Matrix

Actual Values

Predicted Values		Positive (1)	Negative (0)
	Positive (1)	TP	FP
	Negative (0)	FN	TN

Accuracy:

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

Recall:

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

Precision:

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

F_1 score:

$$F_1 = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}}$$

- **Recall (Sensitivity):** Out of all positive samples, how many we predicted correctly.
- **Precision:** Out of all positive samples we have predicted, how many are actually positive.
- **Accuracy:** Out of all samples, how many we predicted correctly (not appropriate for unbalanced classes)
- **F1-Score :** **W**eighted average of Precision and Recall.

Receiver Characteristic Operator (ROC) Curve

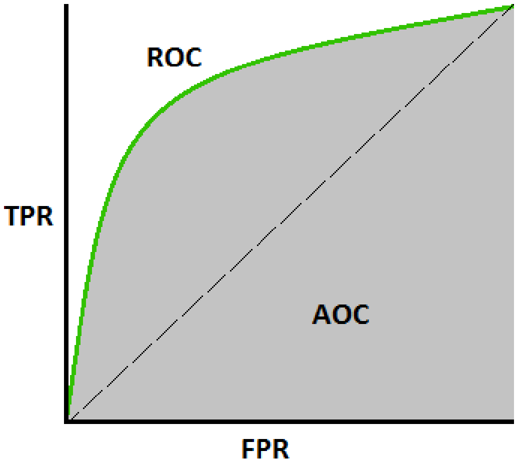
- Evaluate how effective classifiers are at separating classes.

True positive rate (Recall):

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

False positive rate:

$$FPR = \frac{FP}{FP + TN}$$



Shows the performance of a classification model at all classification thresholds.

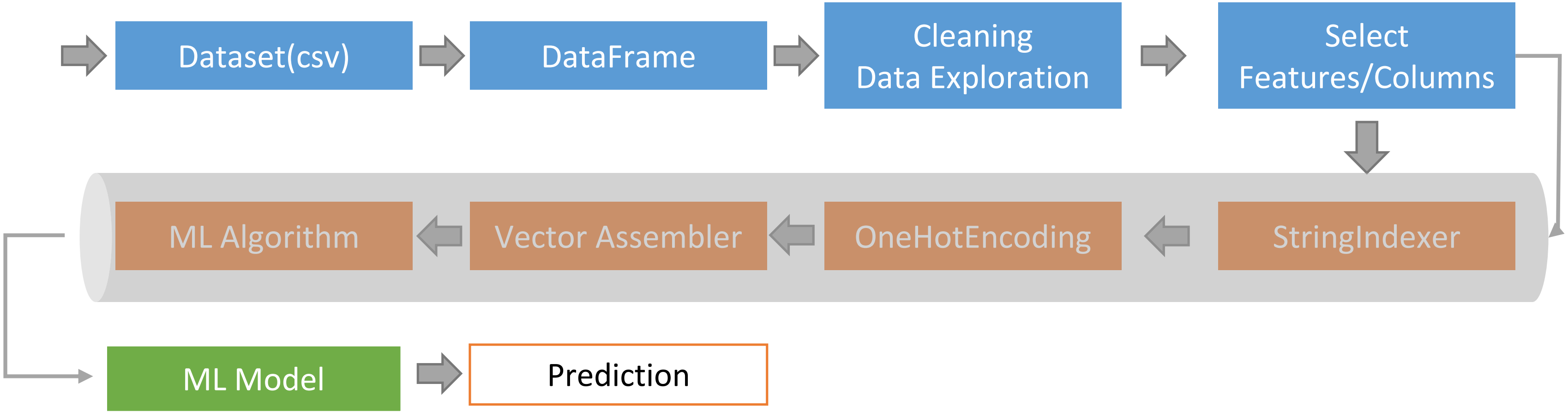
AUC stands for *Area under the ROC Curve.* It provides an aggregate measure of performance across all possible classification thresholds.

Higher the **area under the ROC curve** (AUC), the better the classifier

- Increasing classification threshold **decreases** both **TPR** and **FPR**
- ROC curve closer to top left corner, the better

https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

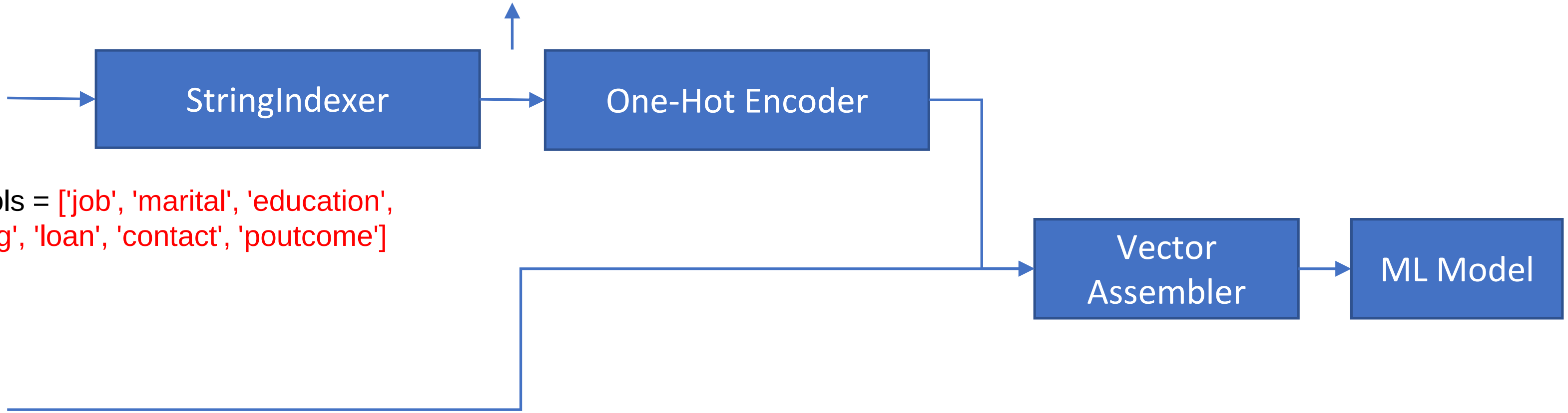
Bank Use Case: Will the customers subscribe?



Feature Transformation

Except 'label' column

Categorical
features
(string-type)



categoryInputCols = ['job', 'marital', 'education', 'default', 'housing', 'loan', 'contact', 'poutcome']

Numerical
features

numericInputCols = ['age', 'balance', 'duration', 'campaign', 'pdays', 'previous']

Pipelines for Feature Transformation

```
categoryInputCols = ['job', 'marital', 'education', 'default', 'housing', 'loan', 'contact', 'poutcome']
numericInputCols = ['age', 'balance', 'duration', 'campaign', 'pdays', 'previous']
categoryOutputCol = 'deposit'
categoryCols = categoryInputCols+[categoryOutputCol]
```

```
# Define the output columns
outputCols=[f'{x}_index' for x in categoryInputCols]
outputCols.append('label')

# Create the index values for categorical values
inputIndexer = StringIndexer(inputCols=categoryCols, outputCols=outputCols)
```

I/O for StringIndexer

```
inputCols_OHE = [x for x in outputCols if x!='label']
outputCols_OHE = [f'{x}_vec' for x in categoryInputCols]

#Define OneHotEncoder with the appropriate columns
encoder = OneHotEncoder(inputCols=inputCols_OHE, outputCols=outputCols_OHE)
```

I/O for OneHoeEncoder

```
# inputCols are all the encoded columns from OHE plus numerical columns
inputCols=outputCols_OHE
assemblerInputs = outputCols_OHE + numericInputCols

# Define the assembler with appropriate input and output columns
assembler = VectorAssembler(inputCols = assemblerInputs, outputCol="features")
```

I/O for VectorAssembler

```
stage_1 = inputIndexer
stage_2 = encoder
stage_3 = assembler

stages = [stage_1,stage_2,stage_3]

pipeline = Pipeline(stages = stages)
pipelineModel = pipeline.fit(df)
df_pipeline = pipelineModel.transform(df)
df_pipeline.printSchema()
```

DecisionTreeClassifier

class pyspark.ml.classification.**DecisionTreeClassifier**(**, featuresCol: str = 'features', labelCol: str = 'label', predictionCol: str = 'prediction', probabilityCol: str = 'probability', rawPredictionCol: str = 'rawPrediction', maxDepth: int = 5, maxBins: int = 32, minInstancesPerNode: int = 1, minInfoGain: float = 0.0, maxMemoryInMB: int = 256, cacheNodeIds: bool = False, checkpointInterval: int = 10, impurity: str = 'gini', seed: Optional[int] = None, weightCol: Optional[str] = None, leafCol: str = '', minWeightFractionPerNode: float = 0.0*) [source]

Decision tree learning algorithm for classification. It supports both binary and multiclass labels, as well as both continuous and categorical features.

https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.classification.DecisionTreeClassifier.html#pyspark.ml.classification.DecisionTreeClassifier

Decision Tree Classifier in Pyspark

Inputs and Outputs

We list the input and output (prediction) column types here. All output columns are optional; to exclude an output column, set its corresponding Param to an empty string.

Input Columns

Param name	Type(s)	Default	Description
labelCol	Double	"label"	Label to predict
featuresCol	Vector	"features"	Feature vector

Output Columns

Param name	Type(s)	Default	Description	Notes
predictionCol	Double	"prediction"	Predicted label	
rawPredictionCol	Vector	"rawPrediction"	Vector of length # classes, with the counts of training instance labels at the tree node which makes the prediction	Classification only
probabilityCol	Vector	"probability"	Vector of length # classes equal to rawPrediction normalized to a multinomial distribution	Classification only
varianceCol	Double		The biased sample variance of prediction	Regression only

https://spark.apache.org/docs/latest/ml-classification-regression.html#decision-trees

BinaryClassificationEvaluator

class `pyspark.ml.evaluation.BinaryClassificationEvaluator`(**, rawPredictionCol='rawPrediction', labelCol='label', metricName='areaUnderROC', weightCol=None, numBins=1000*) [source]

Evaluator for binary classification, which expects input columns rawPrediction, label and an optional weight column. The rawPrediction column can be of type double (binary 0/1 prediction, or probability of label 1) or of type vector (length-2 vector of raw predictions, scores, or label probabilities).

New in version 1.4.0.

`metricName` = *Param(parent='undefined', name='metricName', doc='metric name in evaluation (areaUnderROC|areaUnderPR)')*

https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.evaluation.BinaryClassificationEvaluator.html

MulticlassClassificationEvaluator

class `pyspark.ml.evaluation.MulticlassClassificationEvaluator`(**, predictionCol='prediction', labelCol='label', metricName='f1', weightCol=None, metricLabel=0.0, beta=1.0, probabilityCol='probability', eps=1e-15*) [source]

Evaluator for Multiclass Classification, which expects input columns: prediction, label, weight (optional) and probabilityCol (only for logLoss).

`metricName` = *Param(parent='undefined', name='metricName', doc='metric name in evaluation (f1|accuracy|weightedPrecision|weightedRecall|weightedTruePositiveRate|weightedFalsePositiveRate|weightedFMeasure|truePositiveRateByLabel|falsePositiveRateByLabel|precisionByLabel|recallByLabel|fMeasureByLabel|logLoss|hammingLoss)')*

Thank You!

See you next week.