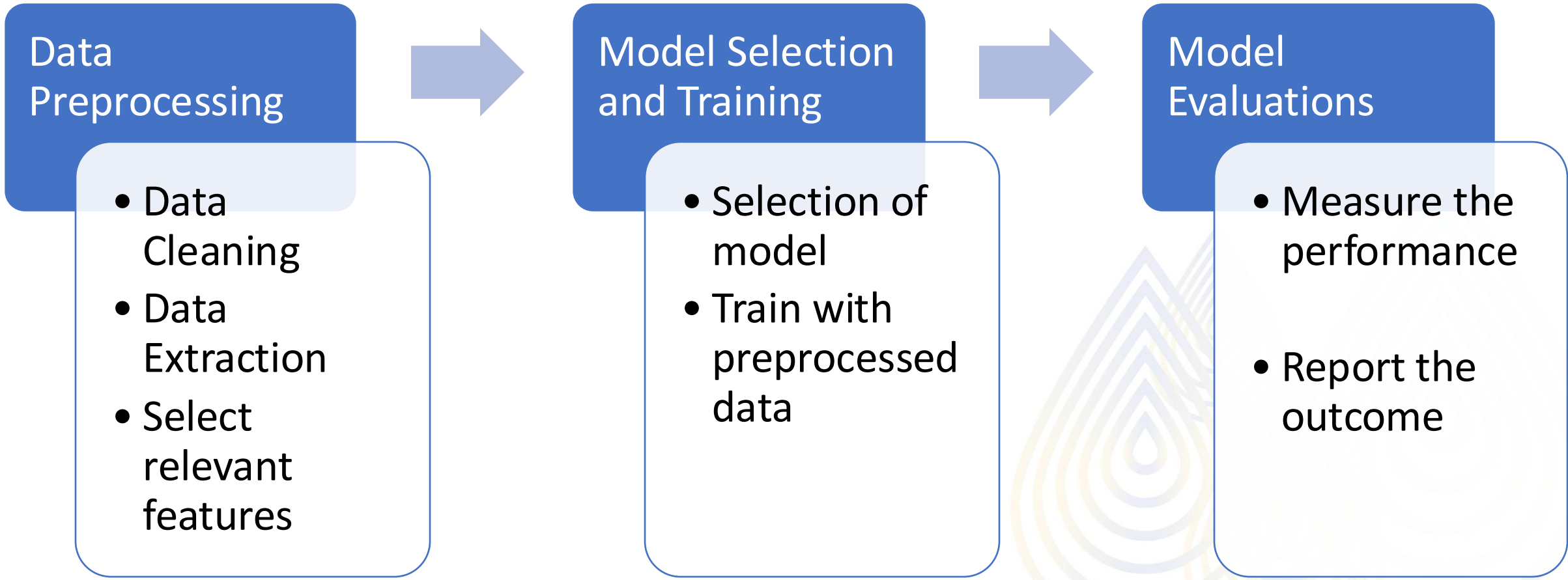
Mahidol University
Wisdom of the Land

BIG DATA PROCESSING

Machine Learning Pipeline with Spark

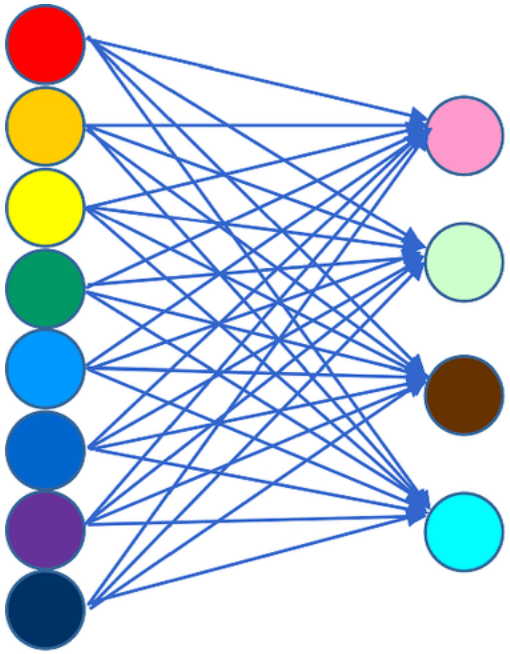
Mahidol University
Wisdom of the Land

Data Pipeline with Spark

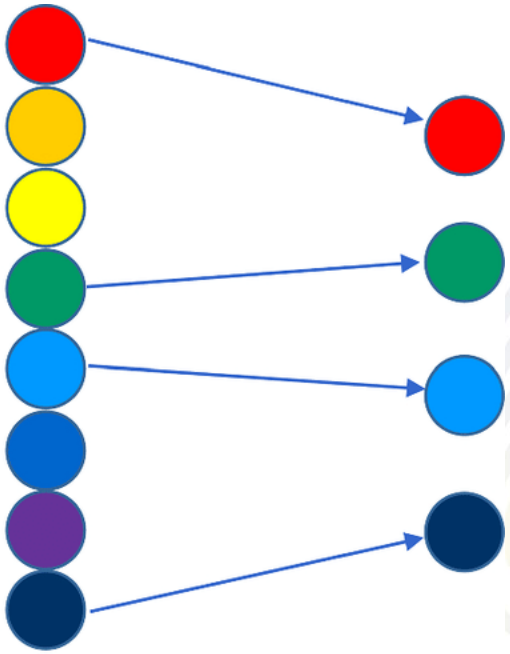


Data Preprocessing

- Data cleaning transforming and preparing raw data
- Use for each type of machine learning tasks



feature extraction

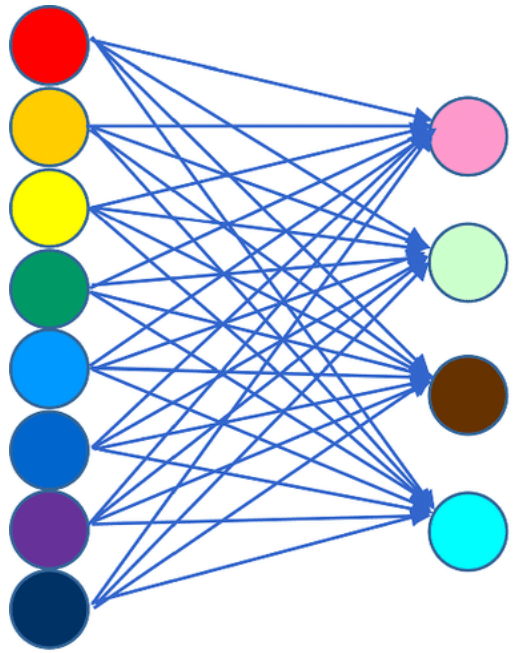


feature selection

Mahidol University
Wisdom of the Land

Feature Transformation

- Feature Transformers
  - Tokenizer
  - StopWordsRemover
  - n-gram
  - Binarizer
  - PCA
  - StringIndexer
  - IndexToString
  - StandardScaler



feature extraction

https://spark.apache.org/docs/latest/ml-features.html

Mahidol University
Wisdom of the Land

Tokenization

Tokenizer

Split text into induvial words of tokens

- **from pyspark.ml.feature import** Tokenizer
- tokenizer = Tokenizer(inputCol=sentence, outputCol=words)
- tokenized = tokenizer.transform(DataFrame)

“Hi I love you → “Hi “I “love “you

Mahidol University
Wisdom of the Land

Stop words remover

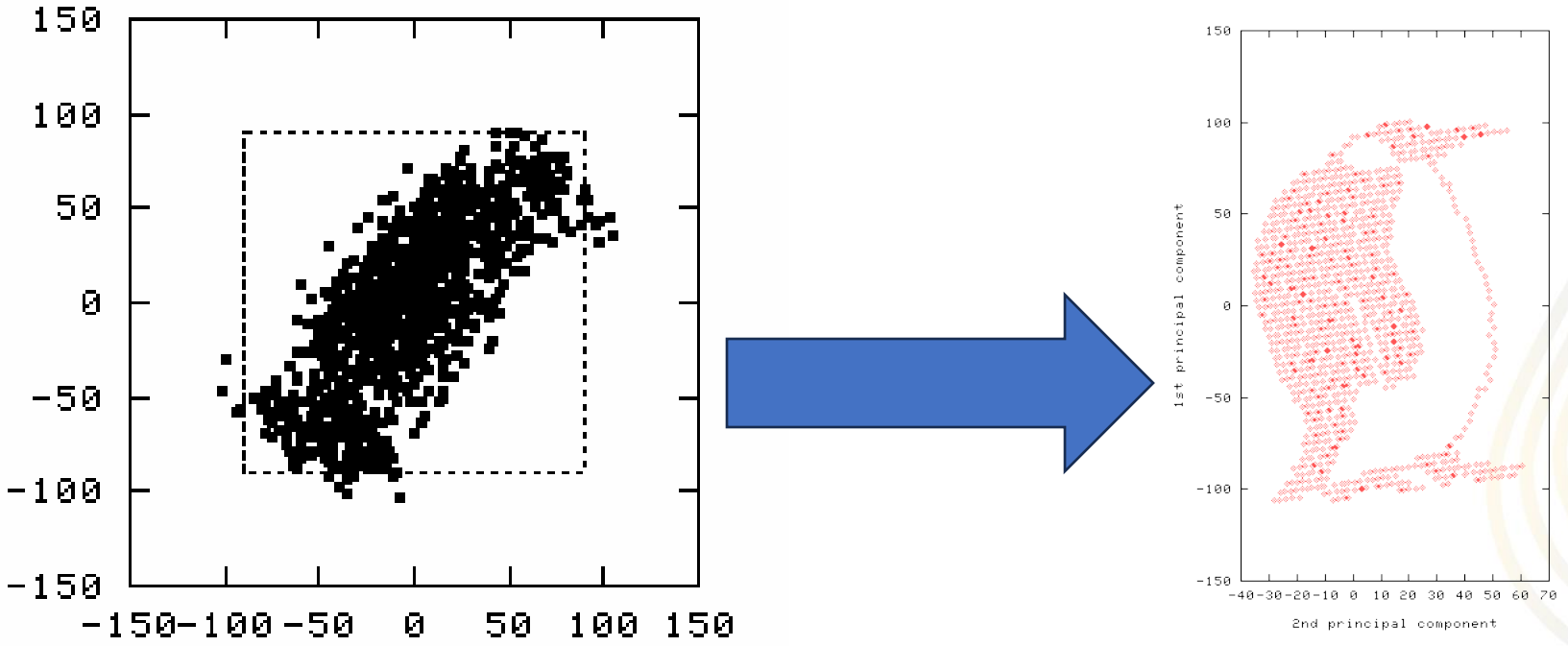
- StopWordsRemover
- Remove a few words (a,an, the, I, or your specified stop word)

```
from pyspark.ml.feature import StopWordsRemover
remover = StopWordsRemover(inputCol=raw, outputCol=filtered)
remover.transform(sentenceData).show(truncate=False)
```

Principal Component Analysis

PCA

- Identifies small set of features to explain variance
- Transforms high-dimensional data to lower dimensional data using new components



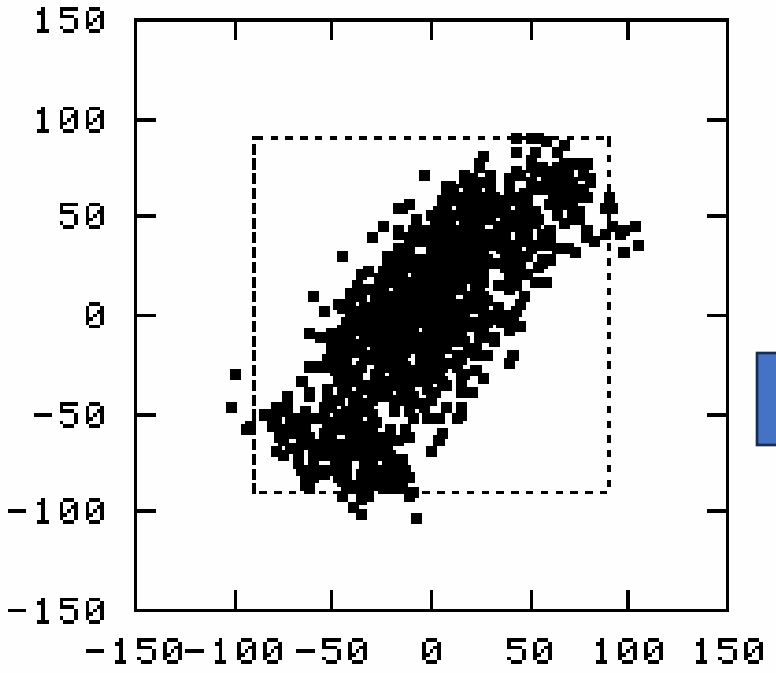
Mahidol University
Wisdom of the Land

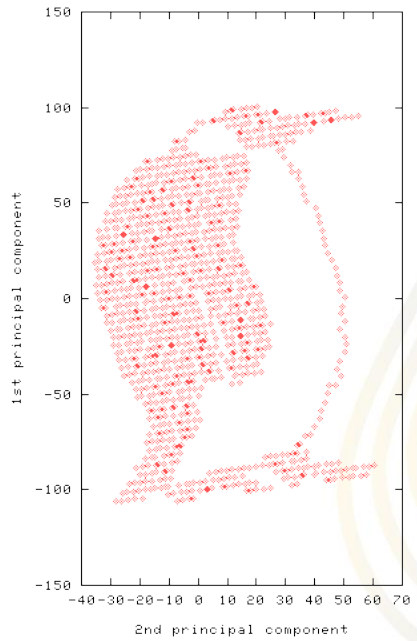
Binarization

- Binarize based on treshole
- **from pyspark.ml.feature import** Binarizer
- binarizer = Binarizer(threshold=0.5, inputCol=input, outputCol=bin)
- binarizedDF = binarizer.transform(DataFrame)

Principal Component Analysis

```
from pyspark.ml.feature import PCA
pca = PCA(k=3, inputCol=features, outputCol=pcaFeatures)
model = pca.fit(df)
result = model.transform(df).select(pcaFeatures)
```





Mahidol University
Wisdom of the Land

Feature Transformation (Scaling)

- Normalizer
- StandardScaler
- RobustScaler
- MinMaxScaler
- MaxAbsScaler

Mahidol University
Wisdom of the Land

Data Scaling

- StandardScaler
  - Mean =0
  - Std =1

- MinMaxScaler
  - Transform data to range 0-1
  - X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))

- MaxAbsScaler
  - Transform data to range [-1 , 1]
  - Divide by maximum absolute value

Mahidol University
Wisdom of the Land

Step for scaling

- Import
  - **from pyspark.ml.feature import** StandardScaler

- Create Scaler
  - scaler = StandardScaler(inputCol=features, outputCol=scaledFeatures)
- *# Compute summary statistics by*
  - scalerModel = scaler.fit(dataFrame)
- *# Scaling each feature.*
  - scaledData = scalerModel.transform(dataFrame)

Mahidol University
Wisdom of the Land

Feature Extraction

- TF-IDF
  - Identify the important words using Term frequency-inverse document frequency
  - Reflect the importance of a term to a document
  - Represent each word by number of occurrence

- Word2Vec
  - Represent words in text as a Vector

Mahidol University
Wisdom of the Land

Convert Categories to numeric

- StringIndexer
  - Categories to number
- One-hot encoding

Label Encoding

Food Name	String Indexer	Calories
Apple	1	95
Chicken	2	231
Broccoli	3	50

→

One Hot Encoding

Apple	Chicken	Broccoli	Calories
1	0	0	95
0	1	0	231
0	0	1	50

Mahidol University
Wisdom of the Land

Example

Try by yourself Example → Feature_Extraction_and_Transformation_using_Spark.ipynb

Data set for model selection

- For not so small data set, obtaining the resulting performance is usually by spitting to 3 data sets.

- **Training data**
  - For training the data, from the **selected** parameters/model

- **Validating data**
  - For finding the **RIGHT parameters/model**.
  - E.g. If it's linear /non linear mode

- **Testing data**
  - For performance evaluation from the unseen data
  - The model will **no longer changed**
  - This step will only try to obtain the **ERROR**

Training set

- **Training data**
  - For training the data, from the **selected** parameters/model
  - Train the model to create the best sets

- **Validating date**
  - For finding the **RIGHT parameters/model**.
    - Normally split from training data

Mahidol University
Wisdom of the Land

Data validation

- Evaluating Training data
- Validating your data
- Testing your performance of the chosen model

The holdout Technique

- **Split dataset into two groups**
  - **Train**
  - **Validation**

Advantage:
- Simple and easy
- Good for big enough data
- Fast

Disadvantage:
- It can be biased

Mahidol University
Wisdom of the Land

ML Pipeline with Spark

- Combine all steps together
- Handles multiple steps
- Organize and automate routine tasks for deployment proces

Mahidol University
Wisdom of the Land

Example of using pipelines

- Define Stages (steps)
  - vectorAssembler=VectorAssembler(inputcol= input, outputCol="features)
  - scaler= StandardScaler(Inputcols= "features, outputCol="scaled_features)
  - lr= LineraRegression(featureCol= "scaled _features)

- Create Pipeline

  pyspark,ml. import Pipeline

  Pipeline=Pipeline(stages[vectorAssembler, scaler, lr])

Mahidol University
Wisdom of the Land

Parameter Tuning

Create parameter

```
paramGrid = ParamGridBuilder()\
              .addGrid(lr.regParam, [0.1, 0.01]) \
             .addGrid(lr.fitIntercept, [False, True])\
            .build()
```

Mahidol University
Wisdom of the Land

Model Selection and Data Tuning

Create a paramter

```
val = TrainValidationSplit(estimator=pipeline,
    estimatorParamMaps=paramGrid, # Define parameter Grid
    evaluator=RegressionEvaluator(), # Define evaluator
     trainRation=.8) # use 80% for training
```

Mahidol University
Wisdom of the Land

Model Selection and Data Tuning

```
crossval = CrossValidator(estimator=pipeline,
 estimatorParamMaps=paramGrid, # Define parameter Grid

evaluator=BinaryClassificationEvaluator(), # Define evaluator

numFolds=3) # use 3+ folds in practice
```

Mahidol University
Wisdom of the Land

Parameter grid

```
paramGrid = ParamGridBuilder() \
.addGrid(lr.fitIntercept, [False, True]) \
.addGrid(lr.maxIter, [5, 10,20]) \
.build()
```

Add grid for each stage in the pipeline

Mahidol University
Wisdom of the Land

Evaluator

- RegressionEvaluator
- rmse (default): root mean squared error
- mse: mean squared error
- r2: R^2 metric
- mae: mean absolute error
- var: explained variance
- BinaryClassificationEvaluator
  - Area under Curve ROC(default), Area under Curve (PR)
- MultilabelClassificationEvaluator
- f1 (default), accuracy
- weightedPrecision, weightedRecall, weightedTruePositiveRate,weightedFalsePositiveRate, weightedFMeasure
- truePositiveRateByLabel, falsePositiveRateByLabel, precisionByLabel, recallByLabel, fMeasureByLabel

Mahidol University
Wisdom of the Land

Model Persistence

- To save the model for future use (deployment)

Save the whole pipeline

- `cvModel.bestModel.save("./model_store")`

Save the prediction pipeline

- `cvModel.bestModel.stages[2].save("./model_store")`

Load the model

- `loaded_model = PipelineModel.load("./model_store/")`
-