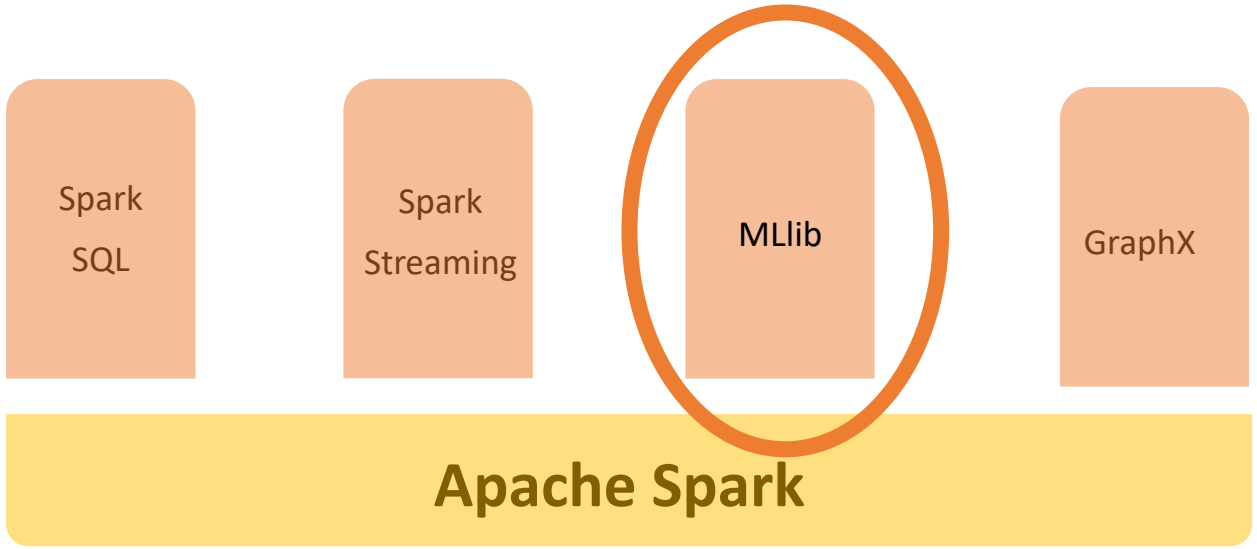
Apache Spark



Mahtab Jamali (Mahtab.Jamali@mau.se)
Ali Soleimani (ali.Soleimani@mau.se)

MALMÖ UNIVERSITY

Key Points

- In this LAB, with work with ApacheSpark, MLlip library.
- For implementation, you are expected to first install Spark on your Windows. However, if you encounter any issues regarding installation, you can use Google Colab instead.

- You are expected to do 2 projects: one involving a Machine Learning Project without Mllib Pipeline, and the other involving a Machine Learning Project with Mllib Pipeline. Each project consists of 15 points (totaling 30 points). Completion of both projects is mandatory to pass this lab.
- For Project Number 1, Checking Spark jobs (If you install Spark on your Windows) is optional, and if you skip this step, you will get 10 points. If you use Colab instead, you will get 15 points without this section because on colab you cannot check the jobs! . Regarding Project Number 2, there is no optional task and you are expected to do all steps. (Keep in mind that to pass all labs, you generally need 100 points.)

- There are several slides about Deep Learning with PySpark MLlib after the implementation slides. This section is intended solely for your knowledge, and if you wish to undertake a Deep Learning project such as CNN models, you can also utilize Spark. No implementation is expected for this part

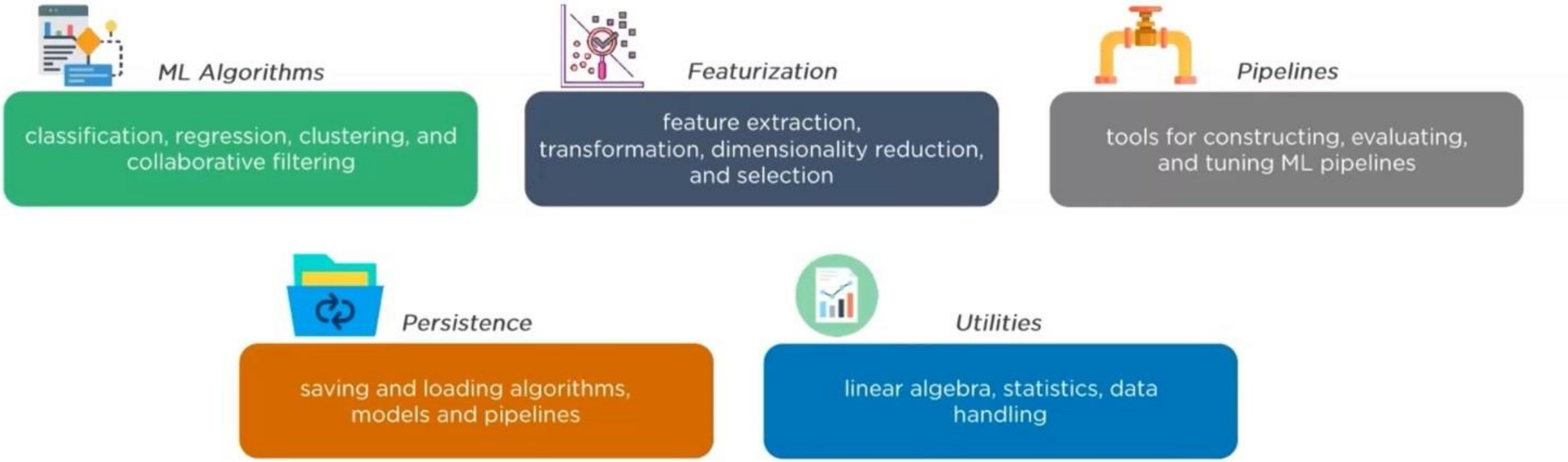
Spark MLlib

MALMÖ UNIVERSITY

Apache Spark MLlib

Apache Spark comes with a library named **MLlib** to perform Machine Learning tasks using the Spark framework.

Spark MLlib Tools



MLlib: Supported Algorithms

MALMÖ UNIVERSITY

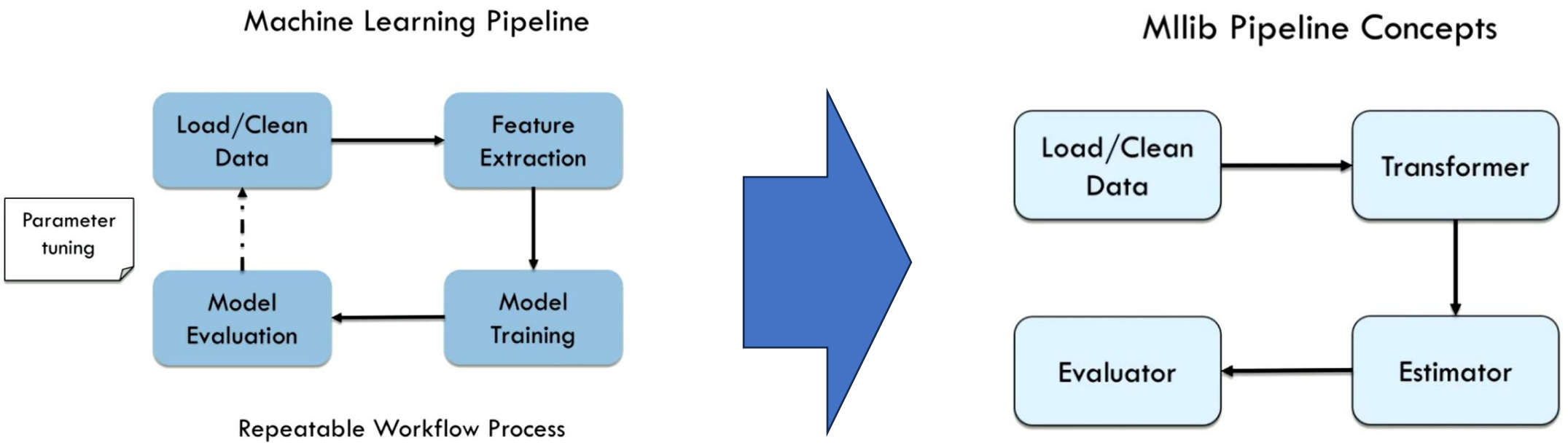
- **Data types**
- **Basic statistics**
  - Summary Statistics
  - Correlations
  - Stratified Sampling
  - Hypothesis Testing
  - Random Data Generation
- **Classification and regression**
  - **Linear Models (SVMs, logistic regression, linear regression)**
  - **Naive Bayes**
  - **Decision Trees**
  - **Ensembles of Trees** (Random Forests and Gradient - Boosted Trees)
- **Collaborative filtering**
  - Alternating Least Squares (ALS)
- **Clustering**
  - k–Means
  - Gaussian Mixture
  - Power Iteration
- **Dimensionality reduction**
  - Singular Value Decomposition (SVD)
  - Principal Component Analysis (PCA)
- **Feature extraction and transformation**
- **Optimization (developer)**
  - Stochastic Gradient Descent
  - Limited-Memory BFGS (L-BFGS)

Spark ML Concepts

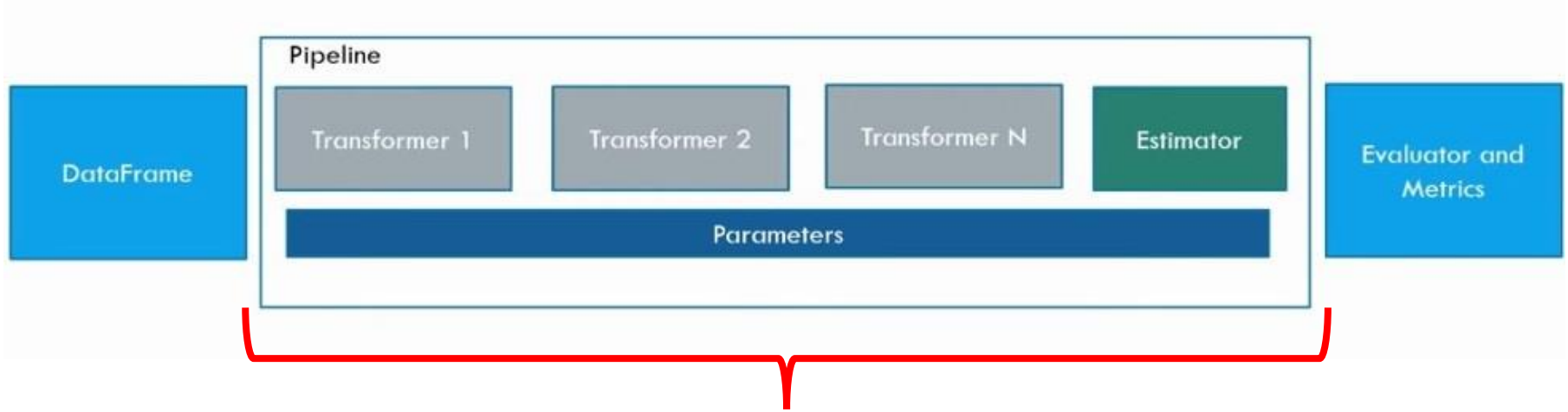
MALMÖ UNIVERSITY

Concept	Explanation
DataFrame	Spark ML uses DataFrame from Spark SQL as an ML dataset, which can hold a variety of data types. E.g., a DataFrame could have different columns storing text, feature vectors, true labels, and predictions
Transformer	A Transformer is an algorithm which can transform one DataFrame into another DataFrame. E.g., an ML model is a Transformer which transforms DataFrame with features into a DataFrame with predictions
Estimator	An Estimator is an algorithm which can be fit on a DataFrame to produce a Transformer. E.g., a learning algorithm is an Estimator which trains on a DataFrame and produces a model
Pipeline	A Pipeline chains multiple Transformers and Estimators together to specify an ML workflow
Parameter	All Transformers and Estimators now share a common API for specifying parameters

MLlib Pipeline Concepts



MLlib Pipeline



Pipeline is combination of Transformers and one Estimator

The input of a transformer is a dataframe and the output of the transformer is a dataframe.
The input of the estimator is a dataframe and the output of the dataframe is a model.

Transformer and estimator

MALMÖ UNIVERSITY

TRANSFORMER

- Feature Transformers
  - Tokenizer
  - StopWordsRemover
  - n-gram
  - Binarizer
  - PCA
  - PolynomialExpansion
  - Discrete Cosine Transform (DCT)
  - StringIndexer
  - IndexToString
  - OneHotEncoder (Deprecated since 2.3.0)
  - OneHotEncoderEstimator
  - VectorIndexer
  - Interaction
  - Normalizer
  - StandardScaler
  - MinMaxScaler
  - MaxAbsScaler
  - Bucketizer
  - ElementwiseProduct
  - SQLTransformer
  - VectorAssembler
  - VectorSizeHint
  - QuantileDiscretizer
  - Imputer

ESTIMATOR

- Classification
  - Logistic regression
    - Binomial logistic regression
    - Multinomial logistic regression
  - Decision tree classifier
  - Random forest classifier
  - Gradient-boosted tree classifier
  - Multilayer perceptron classifier
  - Linear Support Vector Machine
  - One-vs-Rest classifier (a.k.a. One-vs-All)
  - Naive Bayes
- Regression
  - Linear regression
  - Generalized linear regression
    - Available families
  - Decision tree regression
  - Random forest regression
  - Gradient-boosted tree regression
  - Survival regression
  - Isotonic regression
- Linear methods

Project 1: Machine Learning Project without Mllib Pipline

Contents of implementation (without pipeline)

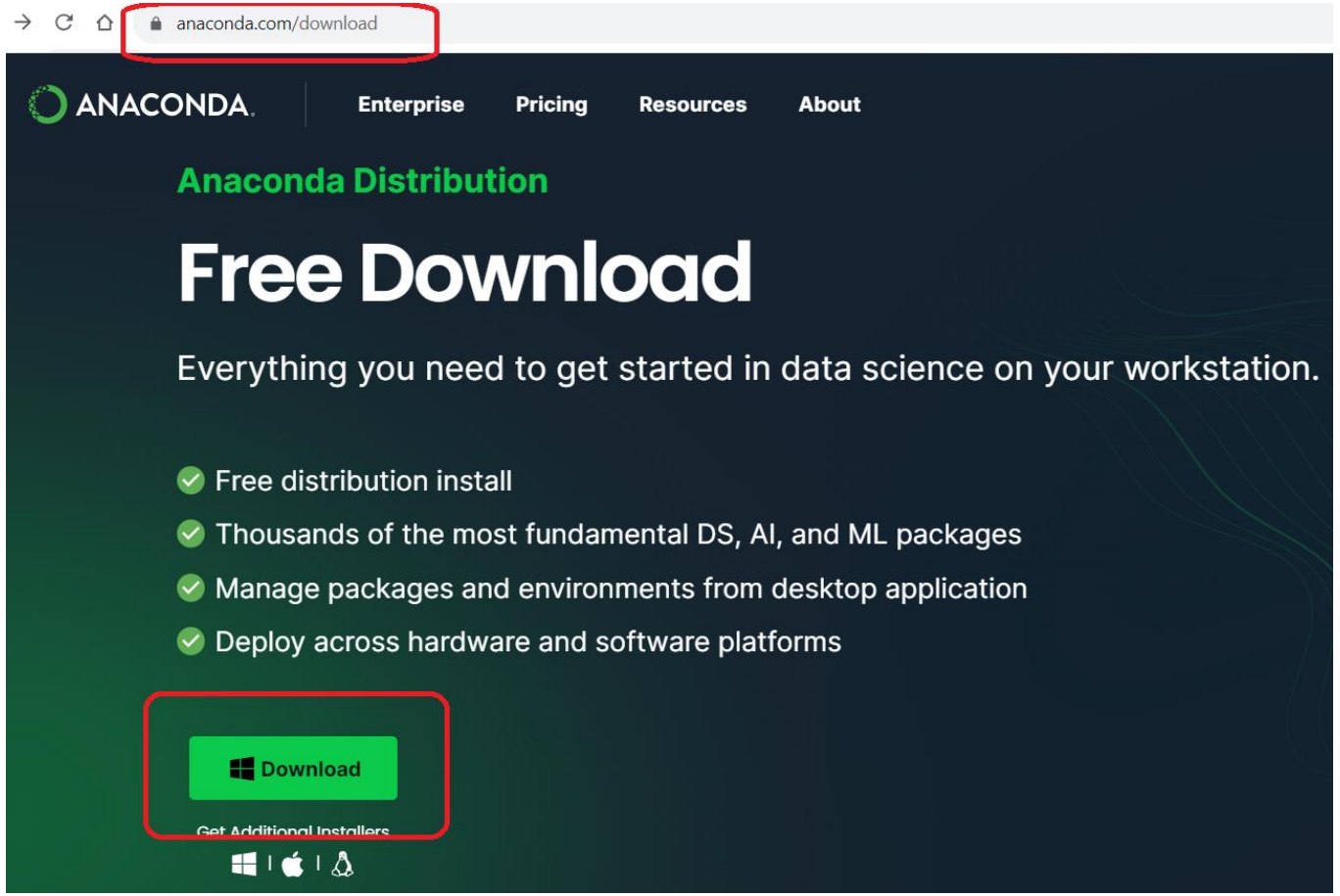
MALMÖ UNIVERSITY

- Setting up Spark Environment on Windows and Google Colab
- Data loading
- Data Preprocessing using SparkML
- Model Training and Testing using SparkML
- Prediction on Test data
- Evaluation of predictions

•Setting up Spark Environment on Windows

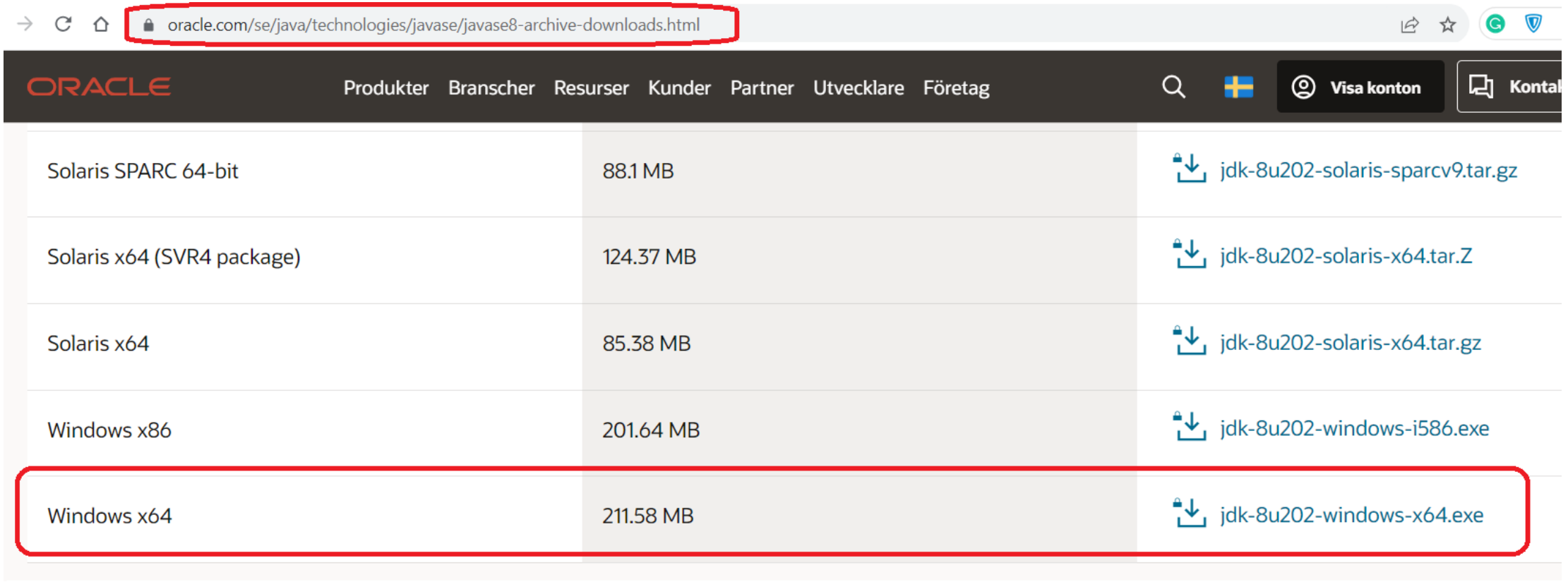
MALMÖ UNIVERSITY

1. Anaconda Installation



•Setting up Spark Environment on Windows

2. Downloading Java Version **8**
https://www.oracle.com/se/java/technologies/javase/javase8-archive-downloads.html



•Setting up Spark Environment on Windows

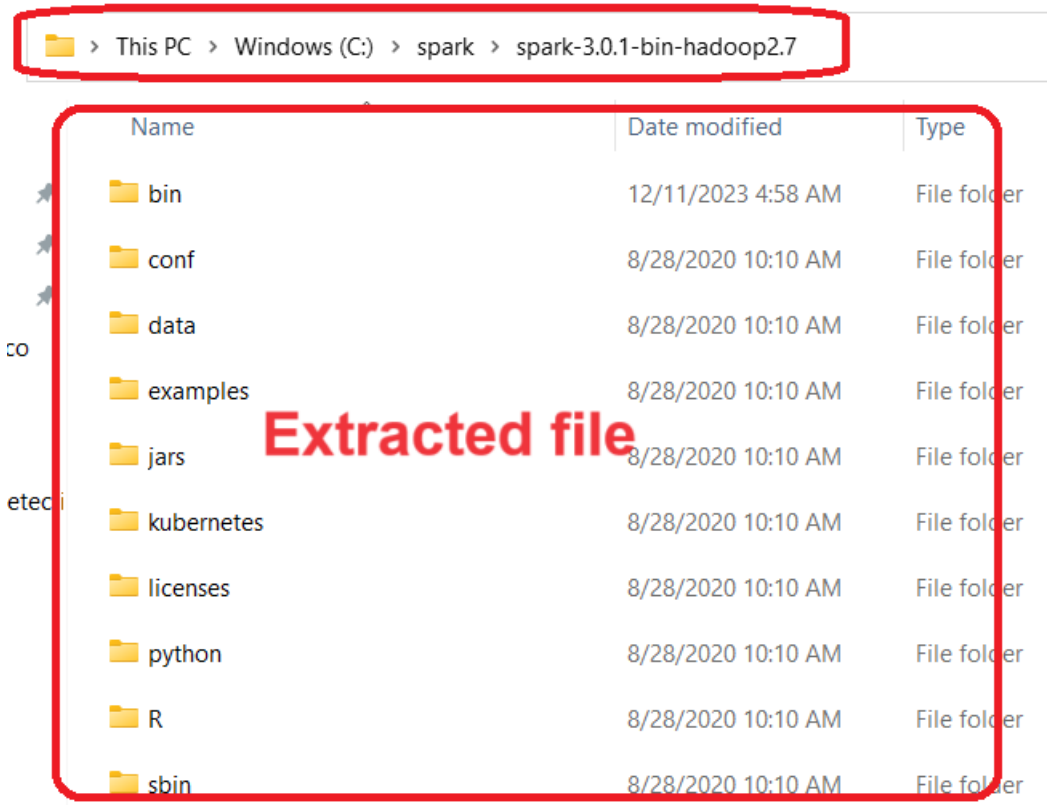
3. Spark Installation

- Go to the https://archive.apache.org/dist/spark/
- Open **spark-3.0.1/** folder
- Download **spark-3.0.1-bin-hadoop2.7.tgz**

•Setting up Spark Environment on Windows

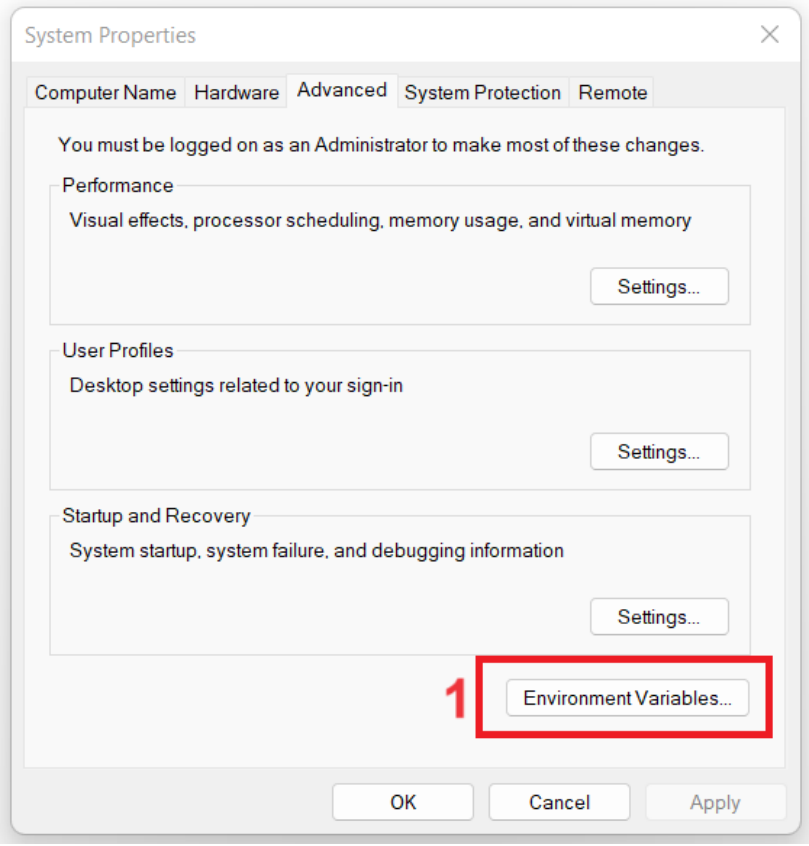
4. Spark Installation

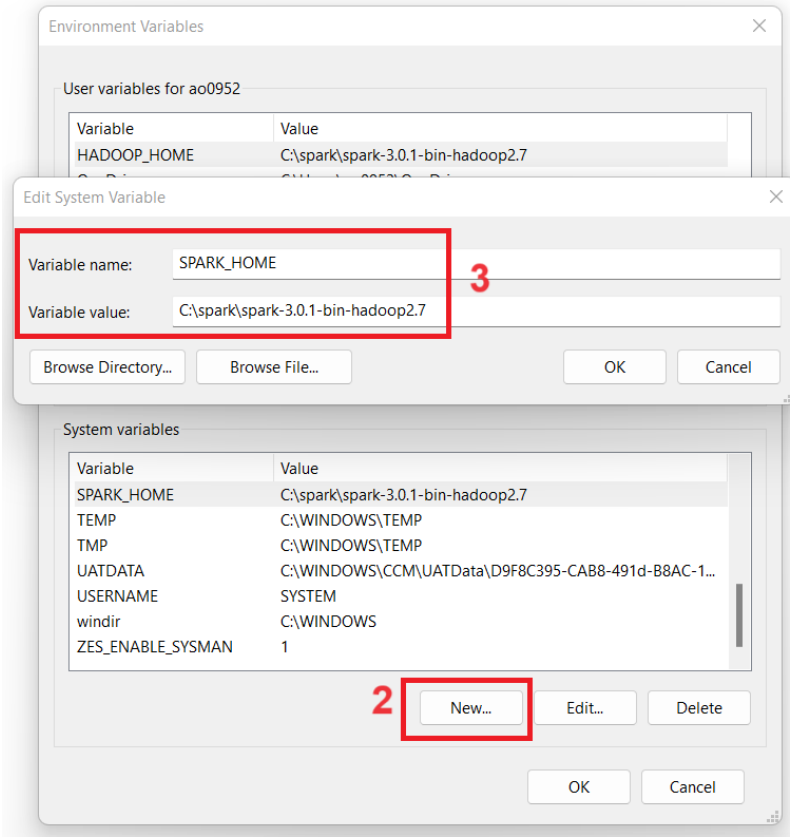
- Extract downloaded spark file
- Create a folder in Drive C titled ‘Spark’ and paste it there

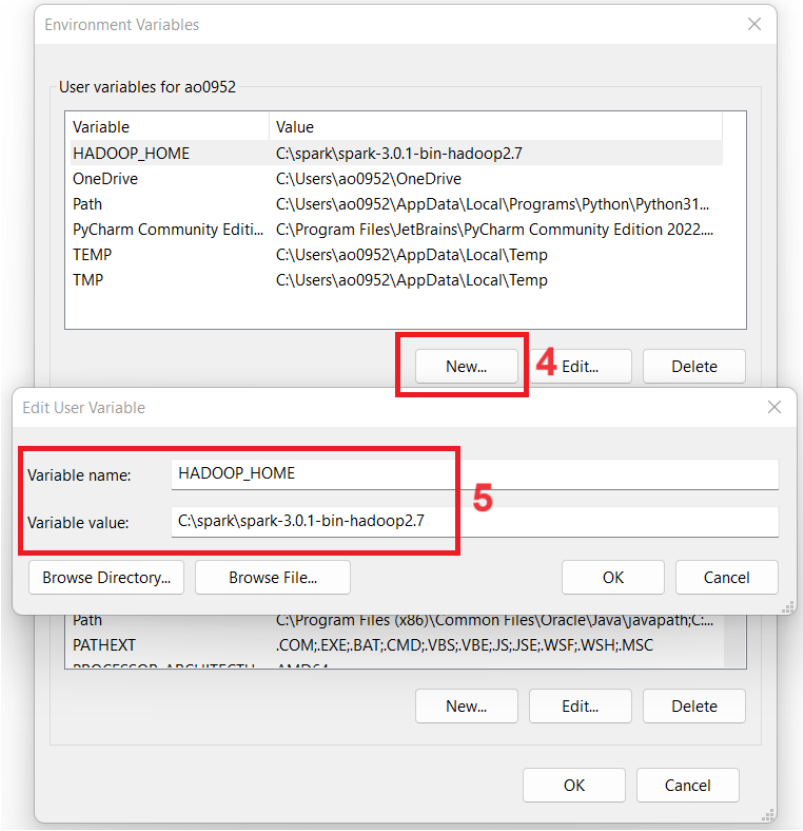


•Setting up Spark Environment on Windows

5. Go to the **advanced system settings** of your PC, and add 2 variables.

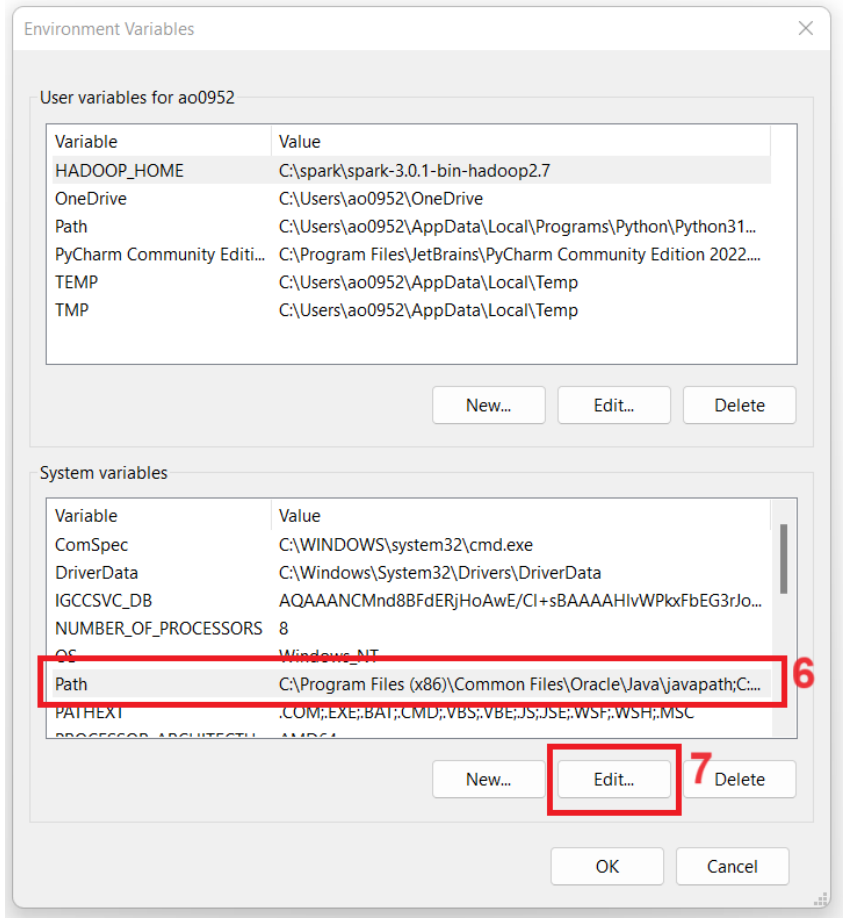


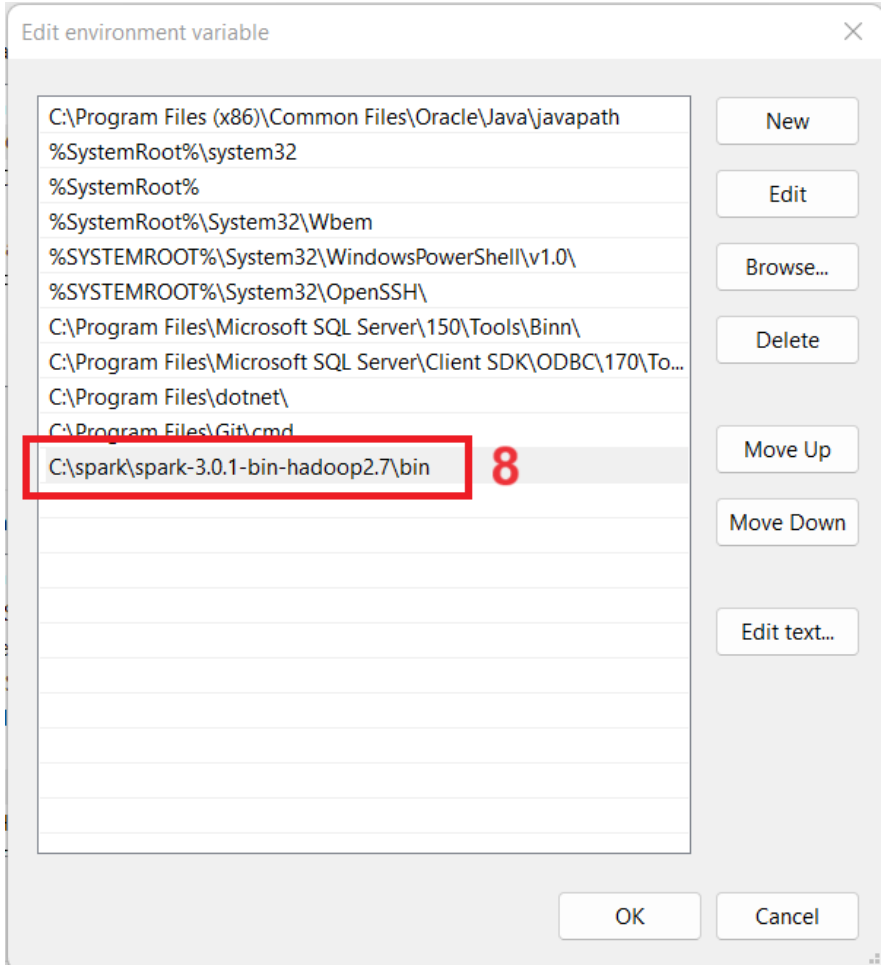




•Setting up Spark Environment on Windows

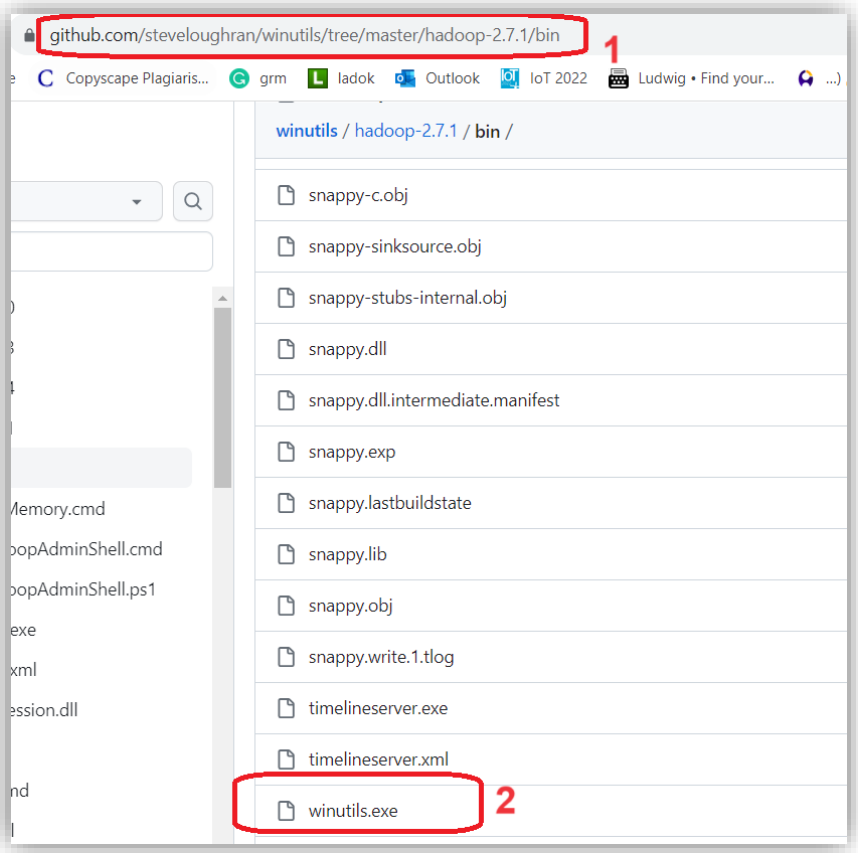
6. Follow the instruction and add **bin folder** to the **Path** section

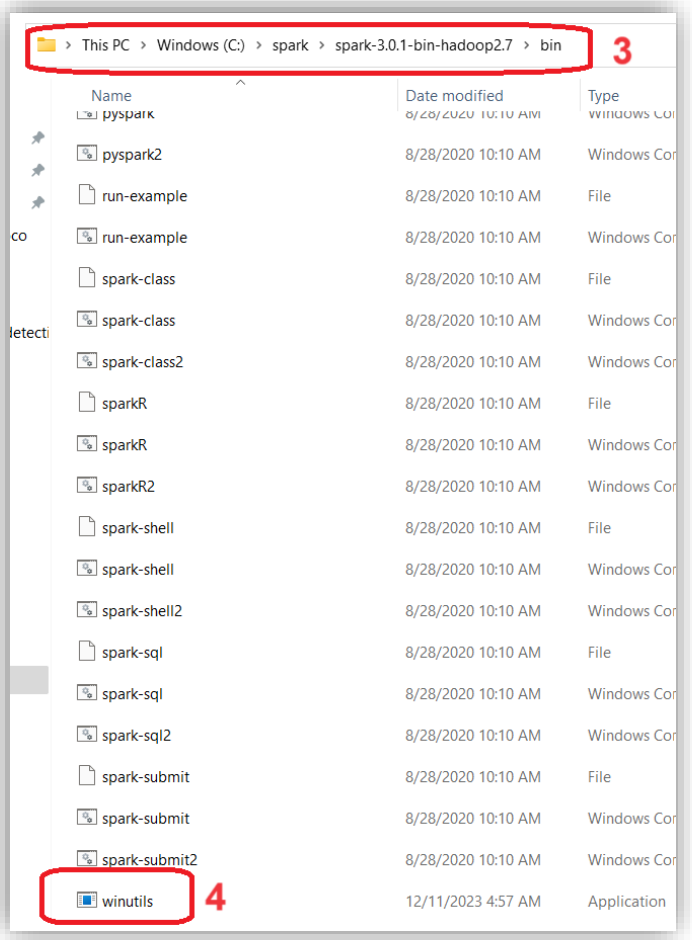




•Setting up Spark Environment on Windows

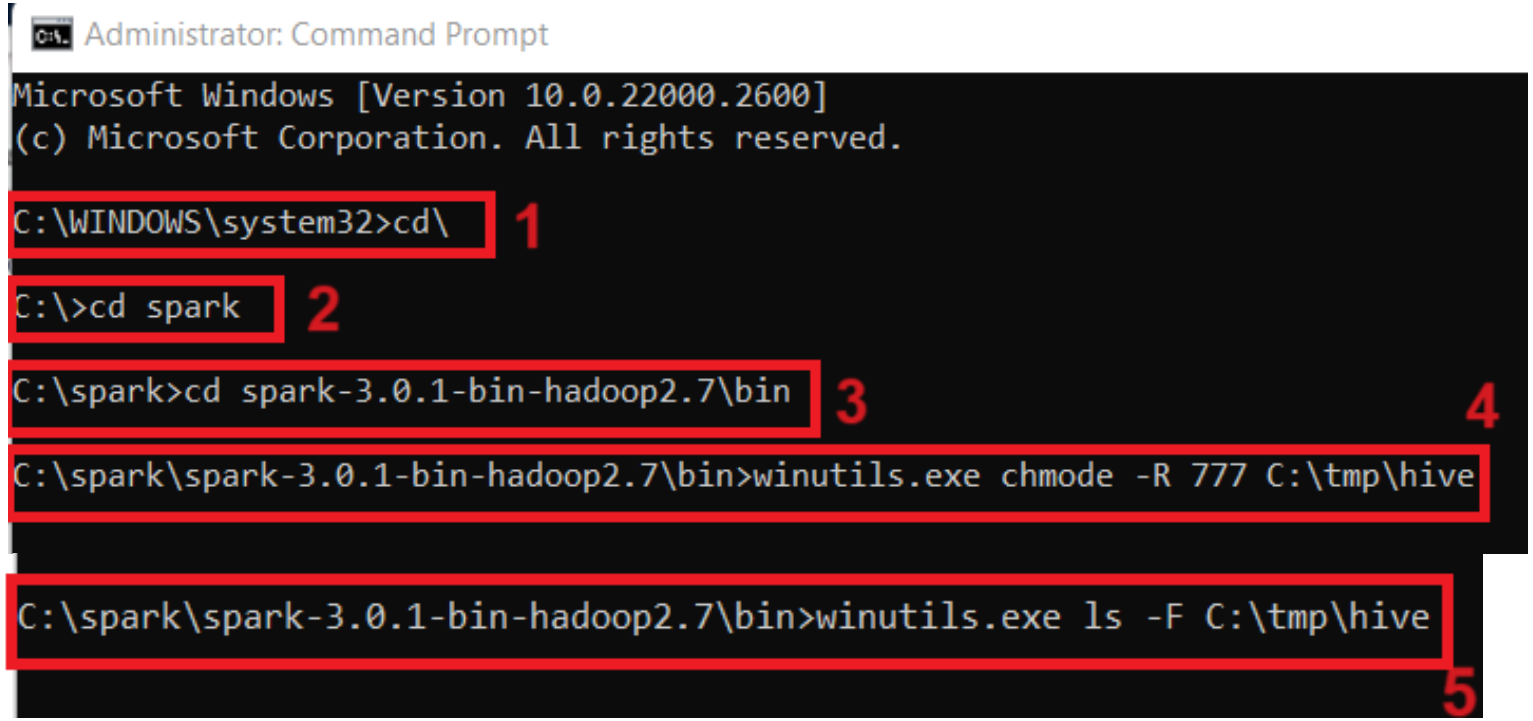
7. Download Winutils.exe from the following link:
https://github.com/steveloughran/winutils/tree/master/hadoop-2.7.1/bin
8. Paste it in the bin folder of spark folder





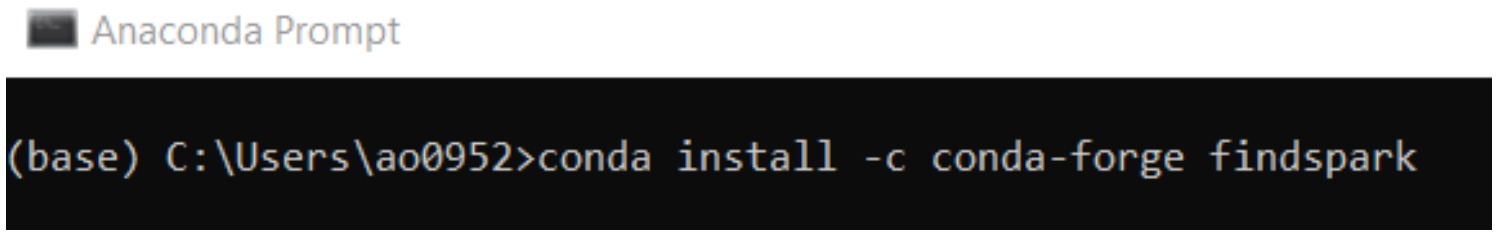
•Setting up Spark Environment on Windows

9. Create a folder in Drive C (C:\tmp\hive)
10. Open cmd of your PC (Run as administrator mode)
11. Follow the instructions below

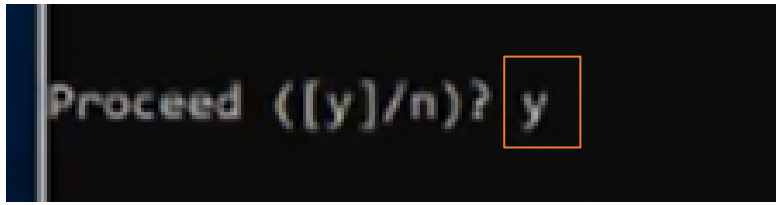


•Setting up Spark Environment on Windows

12. Open Anaconda Prompt on your personal PC
13. Follow the instructions to install **findspark**



1



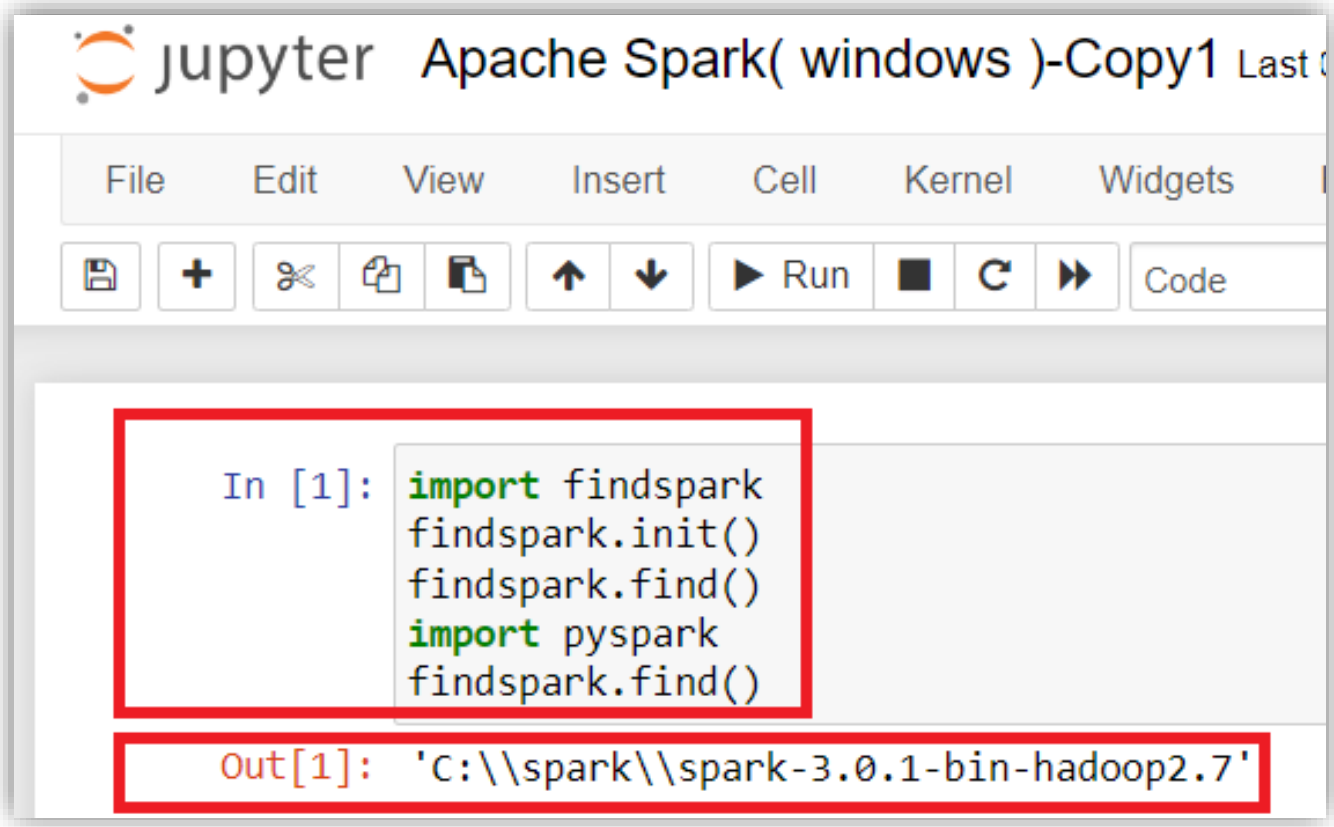
2

•Setting up Spark Environment on Windows

MALMÖ UNIVERSITY

If you have followed all the steps correctly, after entering the following code in Jupyter Notebook, you will get : 'C:\\spark\\spark-3.0.1-bin-hadoop2.7'

You did it



•Setting up Spark Environment on Colab

MALMÖ UNIVERSITY

If you encounter issues installing Spark on your Windows system, you can utilize Google Colab instead.

```
# Setting up the PySpark environment

# Download Java Virtual Machine (JVM)
!apt-get install openjdk-8-jdk-headless -qq > /dev/null

# Replace 'path_to_spark_archive' with the actual path to your uploaded Spark archive
path_to_spark_archive = '/content/drive/MyDrive/big data course/spark-3.4.1-bin-hadoop3.tgz'

# Unzip the spark file
!tar xf "{path_to_spark_archive}"

## Add environmental variables
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = '/content/spark-3.2.1-bin-hadoop3.2'

# Install library for finding Spark
!pip install -q findspark

# Import the library
import findspark

# Replace 'path_to_spark' with the actual path to your Spark installation directory
path_to_spark = '/content/spark-3.4.1-bin-hadoop3'

# Initiate findspark with the correct path
findspark.init(path_to_spark)

# Check the location for Spark
findspark.find()
```

```
# Import SparkSession
from pyspark.sql import SparkSession

# Create a Spark Session
spark = SparkSession.builder \
        .master("local") \
        .appName("Titanic data") \
        .getOrCreate()

# Check Spark Session Information
spark
```

SparkSession - in-memory

SparkContext

Spark UI

Version
v3.4.1
Master
local
AppName
Titanic data

•After Spark installation on Windows

MALMÖ UNIVERSITY

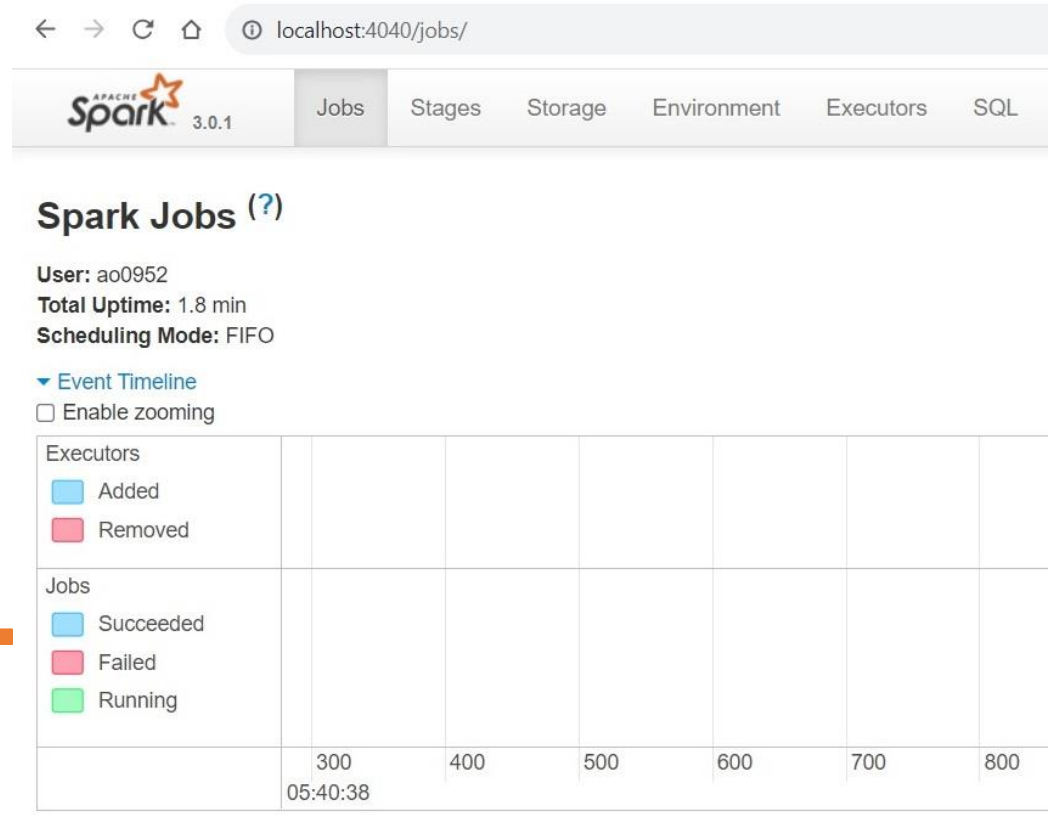
After installing Spark on your personal PC, it's time to dive into coding. In the next slides, you will learn how to import the necessary libraries, initialize Spark, and, finally, train a model for your dataset. So, let’s go!

Initialize SparkSession

MALMÖ UNIVERSITY

```
In [2]: from pyspark.sql import SparkSession

In [3]: spark = SparkSession \
            .builder \
            .appName ('Titanic Data') \
            .getOrCreate()

In [4]: spark

Out[4]: SparkSession - in-memory
        SparkContext

        Spark UI
        Version
        v3.0.1
        Master
        local[*]
        AppName
        Titanic Data
```

If you click on Spark UI on Windows, the Spark environment will open in your browser, allowing you to monitor the processes of Spark masters and workers and manage them. Initially, the environment is empty.



Reading Data

MALMÖ UNIVERSITY

- Dataset (Titanic): https://www.kaggle.com/c/titanic/data

```
df = (spark.read
            .format("csv")
            .option("header","true")
            .load("/content/drive/MyDrive/big data course/titanic dataset/train.csv"))
```

```
df.show(5)
```

```
+-----------+--------+------+--------------------+------+---+-----+-----+----------------+-------+-----+--------+
|PassengerId|Survived|Pclass|                Name|   Sex|Age|SibSp|Parch|          Ticket|   Fare|Cabin|Embarked|
+-----------+--------+------+--------------------+------+---+-----+-----+----------------+-------+-----+--------+
|          1|       0|     3|Braund, Mr. Owen ...|  male| 22|    1|    0|       A/5 21171|   7.25| null|       S|
|          2|       1|     1|Cumings, Mrs. Joh...|female| 38|    1|    0|        PC 17599|71.2833|  C85|       C|
|          3|       1|     3|Heikkinen, Miss. ...|female| 26|    0|    0|STON/O2. 3101282|  7.925| null|       S|
|          4|       1|     1|Futrelle, Mrs. Ja...|female| 35|    1|    0|          113803|   53.1| C123|       S|
|          5|       0|     3|Allen, Mr. Willia...|  male| 35|    0|    0|          373450|   8.05| null|       S|
+-----------+--------+------+--------------------+------+---+-----+-----+----------------+-------+-----+--------+
only showing top 5 rows
```

Selecting some columns (if needed)

MALMÖ UNIVERSITY

- From pyspark.sql.functions import col and then select columns

```
from pyspark.sql.functions import col
```

```
dataset = df.select(col('Survived').cast('float'),
                    col('Pclass').cast('float'),
                    col('Sex'),
                    col('Age').cast('float'),
                    col('Fare').cast('float'),
                    col('Embarked')
                    )
```

```
dataset.show(4)
```

```
+--------+------+------+----+-------+--------+
|Survived|Pclass|   Sex| Age|   Fare|Embarked|
+--------+------+------+----+-------+--------+
|     0.0|   3.0|  male|22.0|   7.25|       S|
|     1.0|   1.0|female|38.0|71.2833|       C|
|     1.0|   3.0|female|26.0|  7.925|       S|
|     1.0|   1.0|female|35.0|   53.1|       S|
+--------+------+------+----+-------+--------+
only showing top 4 rows
```

Removing null values (if needed)

MALMÖ UNIVERSITY

```
from pyspark.sql.functions import isnull, when, count, col
```

```
dataset.select([count(when(isnull(c), c)).alias(c) for c in dataset.columns]).show()
```

```
+--------+------+---+---+----+--------+
|Survived|Pclass|Sex|Age|Fare|Embarked|
+--------+------+---+---+----+--------+
|       0|     0|  0|177|   0|       2|
+--------+------+---+---+----+--------+
```

```
dataset = dataset.replace('?', None)\
         .dropna(how='any')
```

```
dataset.select([count(when(isnull(c), c)).alias(c) for c in dataset.columns]).show()
```

```
+--------+------+---+---+----+--------+
|Survived|Pclass|Sex|Age|Fare|Embarked|
+--------+------+---+---+----+--------+
|       0|     0|  0|  0|   0|       0|
+--------+------+---+---+----+--------+
```

Converting categorical variables to numeric values

- Spark only supports numeric values and is incapable of handling categorical variables. For modeling, all categorical variables must be converted to numeric values. To achieve this, StringIndexer is employed.

1

```
dataset.show(3)
```

Should be converted

```
+--------+------+------+----+-------+--------+
|Survived|Pclass|   Sex| Age|   Fare|Embarked|
+--------+------+------+----+-------+--------+
|     0.0|   3.0|  male|22.0|   7.25|       S|
|     1.0|   1.0|female|38.0|71.2833|       C|
|     1.0|   3.0|female|26.0|  7.925|       S|
+--------+------+------+----+-------+--------+
only showing top 3 rows
```

2

```
from pyspark.ml.feature import StringIndexer
```

3

```
dataset = StringIndexer(
    inputCol='Sex',
    outputCol='Gender',
    handleInvalid='keep').fit(dataset).transform(dataset)
```

```
dataset = StringIndexer(
    inputCol='Embarked',
    outputCol='Boarded',
    handleInvalid='keep').fit(dataset).transform(dataset)
```

4

```
dataset.show(2)
```

```
+--------+------+------+----+-------+--------+------+-------+
|Survived|Pclass|   Sex| Age|   Fare|Embarked|Gender|Boarded|
+--------+------+------+----+-------+--------+------+-------+
|     0.0|   3.0|  male|22.0|   7.25|       S|   0.0|    0.0|
|     1.0|   1.0|female|38.0|71.2833|       C|   1.0|    1.0|
+--------+------+------+----+-------+--------+------+-------+
only showing top 2 rows
```

Finally, we can drop ‘sex’ and ‘ embarked’ columns

5

```
#Drop unnecessary columns
dataset = dataset.drop('Sex')
dataset = dataset.drop('Embarked')
```

Feature engineering

- Spark learns through two columns, label and feature. Therefore, all columns except the target column must be combined into a single column. This is accomplished via VesctorAssembler.

```
#Assemble all the feautures with VectorAssembler
from pyspark.ml.feature import VectorAssembler
```

```
require_featured = ['Pclass', 'Age', 'Fare', 'Gender', 'Boarded']
assembler = VectorAssembler(inputCols=require_featured, outputCol='features')
transformed_data = assembler.transform(dataset)
```

```
transformed_data.show(5)
```

```
+--------+------+----+-------+------+-------+--------------------+
|Survived|Pclass| Age|   Fare|Gender|Boarded|            features|
+--------+------+----+-------+------+-------+--------------------+
|     0.0|   3.0|22.0|   7.25|   0.0|    0.0|[3.0,22.0,7.25,0....|
|     1.0|   1.0|38.0|71.2833|   1.0|    1.0|[1.0,38.0,71.2833...|
|     1.0|   3.0|26.0|  7.925|   1.0|    0.0|[3.0,26.0,7.92500...|
|     1.0|   1.0|35.0|   53.1|   1.0|    0.0|[1.0,35.0,53.0999...|
|     0.0|   3.0|35.0|   8.05|   0.0|    0.0|[3.0,35.0,8.05000...|
+--------+------+----+-------+------+-------+--------------------+
only showing top 5 rows
```

Label
(target variables)

feature

Modeling

MALMÖ UNIVERSITY

```
#spliting dataset into train and test
(training_data, test_data) = transformed_data.randomSplit([0.8,0.2])
print("Number of train samples: " + str(training_data.count()))
print("Number of train samples: " + str(test_data.count()))
```

```
Number of train samples: 572
Number of train samples: 140
```

```
from pyspark.ml.classification import RandomForestClassifier
```

```
rf = RandomForestClassifier(labelCol='Survived',
                            featuresCol='features',
                            maxDepth=5)
```

```
model = rf.fit(training_data)
```

```
predictions = model.transform(test_data)
```

Evaluation

MALMÖ UNIVERSITY

```
#Evaluation

from pyspark.ml.evaluation import MulticlassClassificationEvaluator
```

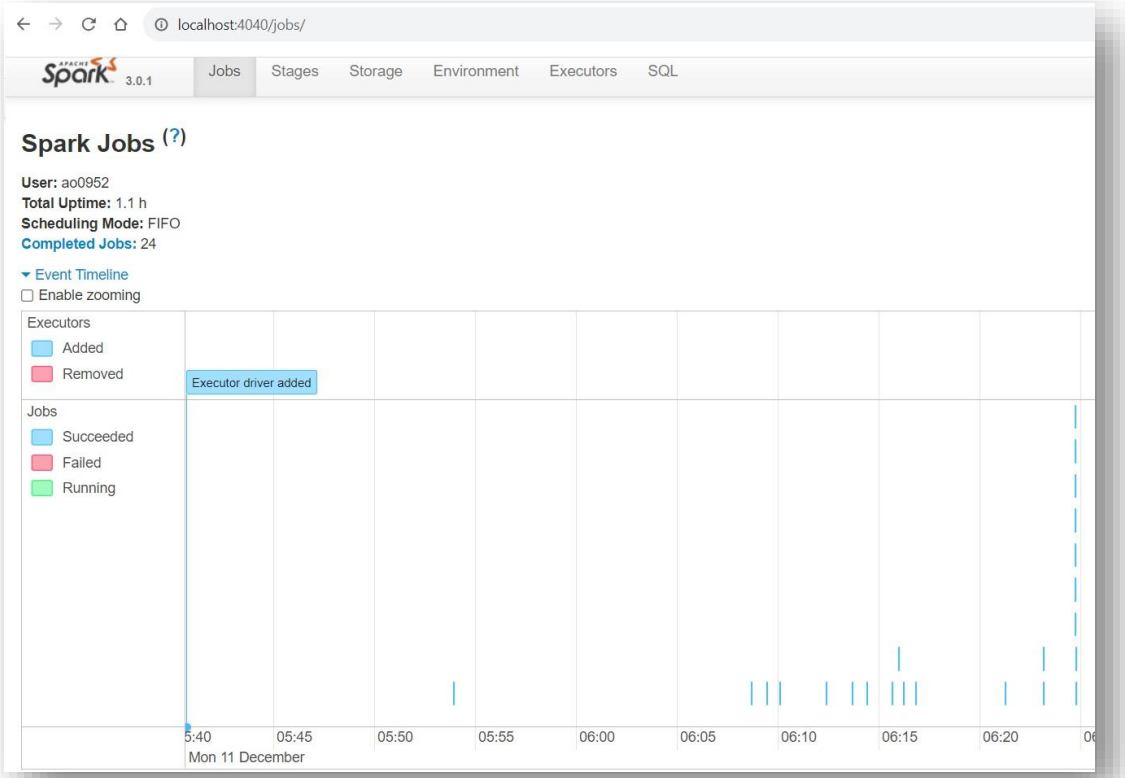
```
evaluator = MulticlassClassificationEvaluator(
    labelCol='Survived',
    predictionCol='prediction',
    metricName = 'accuracy')
```

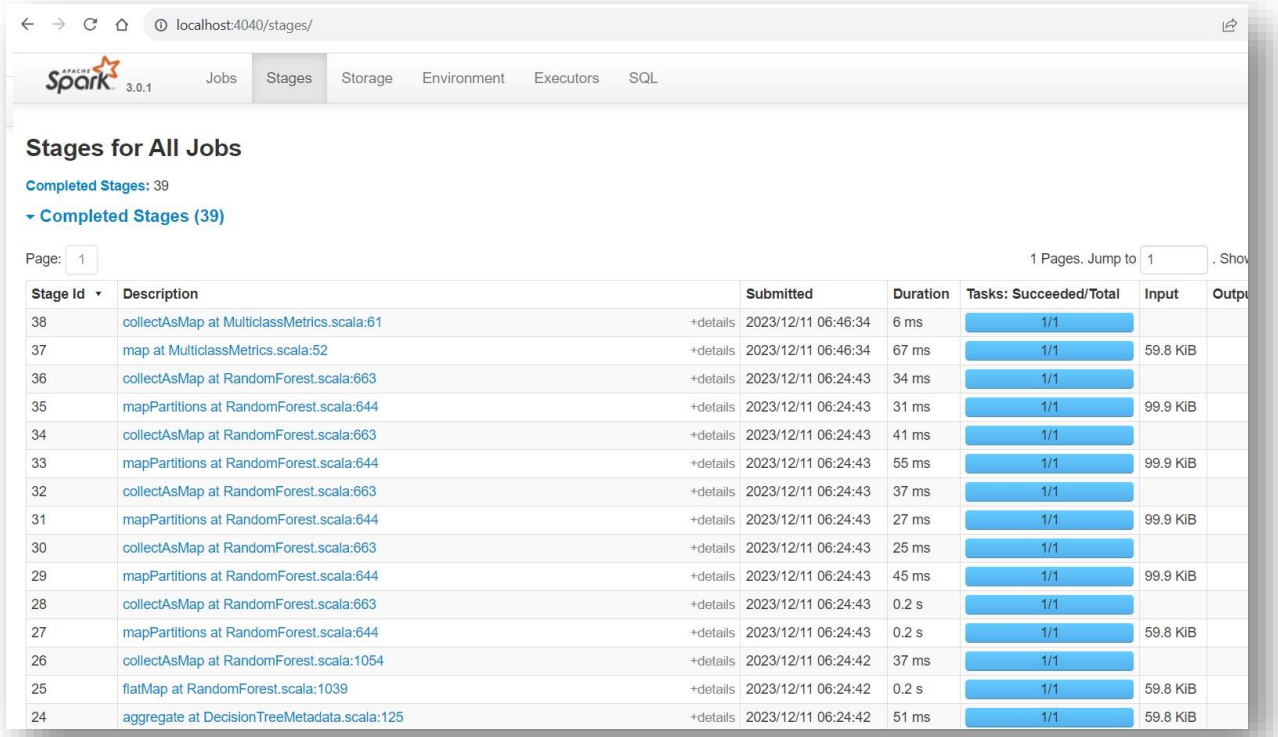
```
accuracy = evaluator.evaluate(predictions_test)
print('Training Accuracy = ', accuracy)
```

```
Training Accuracy =  0.8091603053435115
```

Checking Spark jobs

After all you can check **Spark Jobs** on your local machine and manage them.





Project 2: Machine Learning Project with Mllib Pipline

Contents of implementation (with pipeline)

MALMÖ UNIVERSITY

Contents:

1. Setting up the environment
2. Read data
3. Leverage Spark ML pipeline

Setting up the environment

MALMÖ UNIVERSITY

```
# Setting up the PySpark environment

# Download Java Virtual Machine (JVM)
!apt-get install openjdk-8-jdk-headless -qq > /dev/null

# Replace 'path_to_spark_archive' with the actual path to your uploaded Spark archive
path_to_spark_archive = '/content/drive/MyDrive/big data course/spark-3.4.1-bin-hadoop3.tgz'

# Unzip the spark file
!tar xf "{path_to_spark_archive}"

## Add environmental variables
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = '/content/spark-3.2.1-bin-hadoop3.2'

# Install library for finding Spark
!pip install -q findspark

# Import the library
import findspark

# Replace 'path_to_spark' with the actual path to your Spark installation directory
path_to_spark = '/content/spark-3.4.1-bin-hadoop3'

# Initiate findspark with the correct path
findspark.init(path_to_spark)

# Check the location for Spark
findspark.find()
```

Initialize SparkSession

MALMÖ UNIVERSITY

```
# Import SparkSession
from pyspark.sql import SparkSession

# Create a Spark Session
spark = SparkSession.builder \
        .master("local") \
        .appName("Titanic data") \
        .getOrCreate()

# Check Spark Session Information
spark
```

SparkSession - in-memory

SparkContext

Spark UI

Version
v3.4.1
Master
local
AppName
Titanic data

Reading data

MALMÖ UNIVERSITY

```
df = (spark.read
            .format("csv")
            .option("header","true")
            .load("/content/drive/MyDrive/big data course/titanic dataset/train.csv"))
```

```
df.show(3)
```

```
+-----------+--------+------+--------------------+------+---+-----+-----+----------------+-------+-----+--------+
|PassengerId|Survived|Pclass|                Name|   Sex|Age|SibSp|Parch|          Ticket|   Fare|Cabin|Embarked|
+-----------+--------+------+--------------------+------+---+-----+-----+----------------+-------+-----+--------+
|          1|       0|     3|Braund, Mr. Owen ...|  male| 22|    1|    0|       A/5 21171|   7.25| null|       S|
|          2|       1|     1|Cumings, Mrs. Joh...|female| 38|    1|    0|        PC 17599|71.2833|  C85|       C|
|          3|       1|     3|Heikkinen, Miss. ...|female| 26|    0|    0|STON/O2. 3101282|  7.925| null|       S|
+-----------+--------+------+--------------------+------+---+-----+-----+----------------+-------+-----+--------+
only showing top 3 rows
```

Importing functions

MALMÖ UNIVERSITY

```
from pyspark.sql import functions as F
from pyspark.sql import types as T

# StringIndexer is similar to labelencoder which gives a label to each category
# OneHotEncoder created onehot encoding vector
from pyspark.ml.feature import StringIndexer, OneHotEncoder

# VectorAssembler is used to create vector from the features. MOdeling takes vector as an input
from pyspark.ml.feature import VectorAssembler

# DecisionTreeClassifier is used for classiication problems
from pyspark.ml.classification import RandomForestClassifier
```

Using pipeline

MALMÖ UNIVERSITY

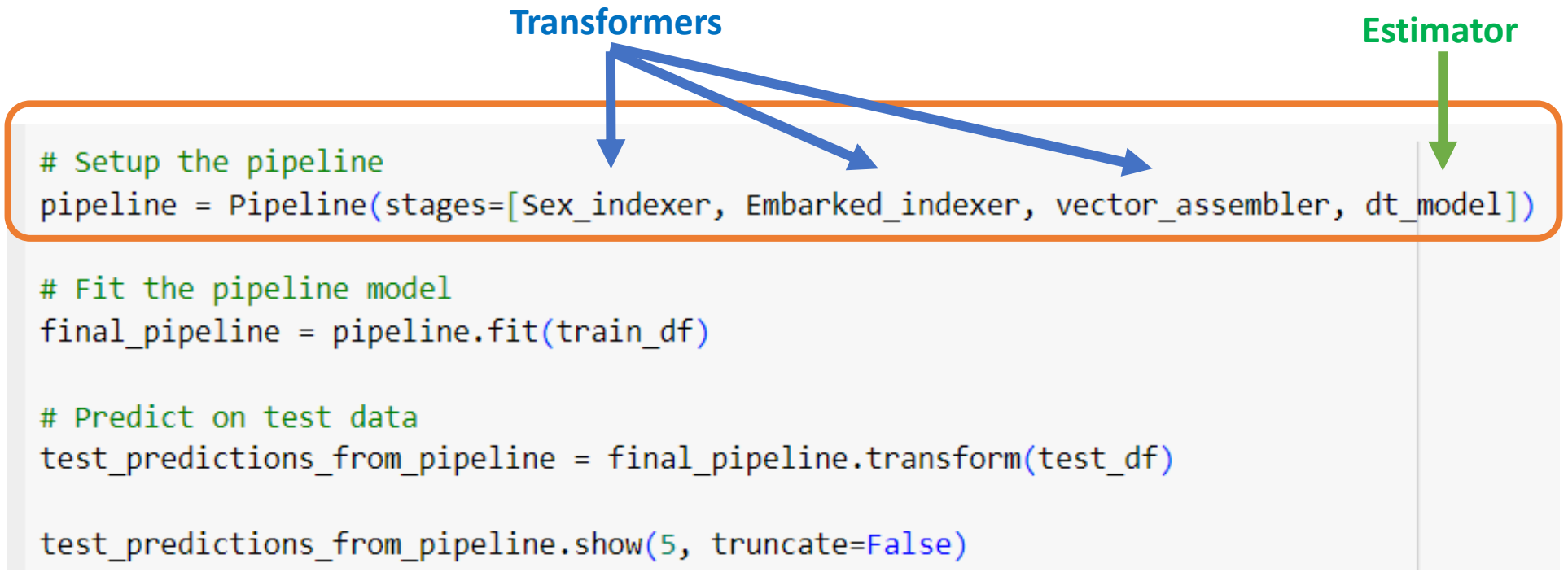
```
# Import pipeline from PySpark ML
from pyspark.ml import Pipeline
```

```
(train_df, test_df) = dataset.randomSplit([0.8, 0.2], 11)
print("Number of train samples: " + str(train_df.count()))
print("Number of test samples: " + str(test_df.count()))
```

```
Number of train samples: 562
Number of test samples: 150
```

```
# Label Encoding of categorical variables  without any .fit or .transform
Sex_indexer = StringIndexer(inputCol="Sex", outputCol="Gender")
Embarked_indexer = StringIndexer(inputCol="Embarked", outputCol="Boarded")


#Assemble all the feautures with VectorAssembler
inputCols = ['Pclass', 'Age', 'Fare', 'Gender', 'Boarded']
outputCol = "features"
vector_assembler = VectorAssembler(inputCols = inputCols, outputCol = outputCol)


# Modeling using DecisionTreeClassifier
dt_model = RandomForestClassifier(labelCol="Survived", featuresCol="features")
```

Using pipeline



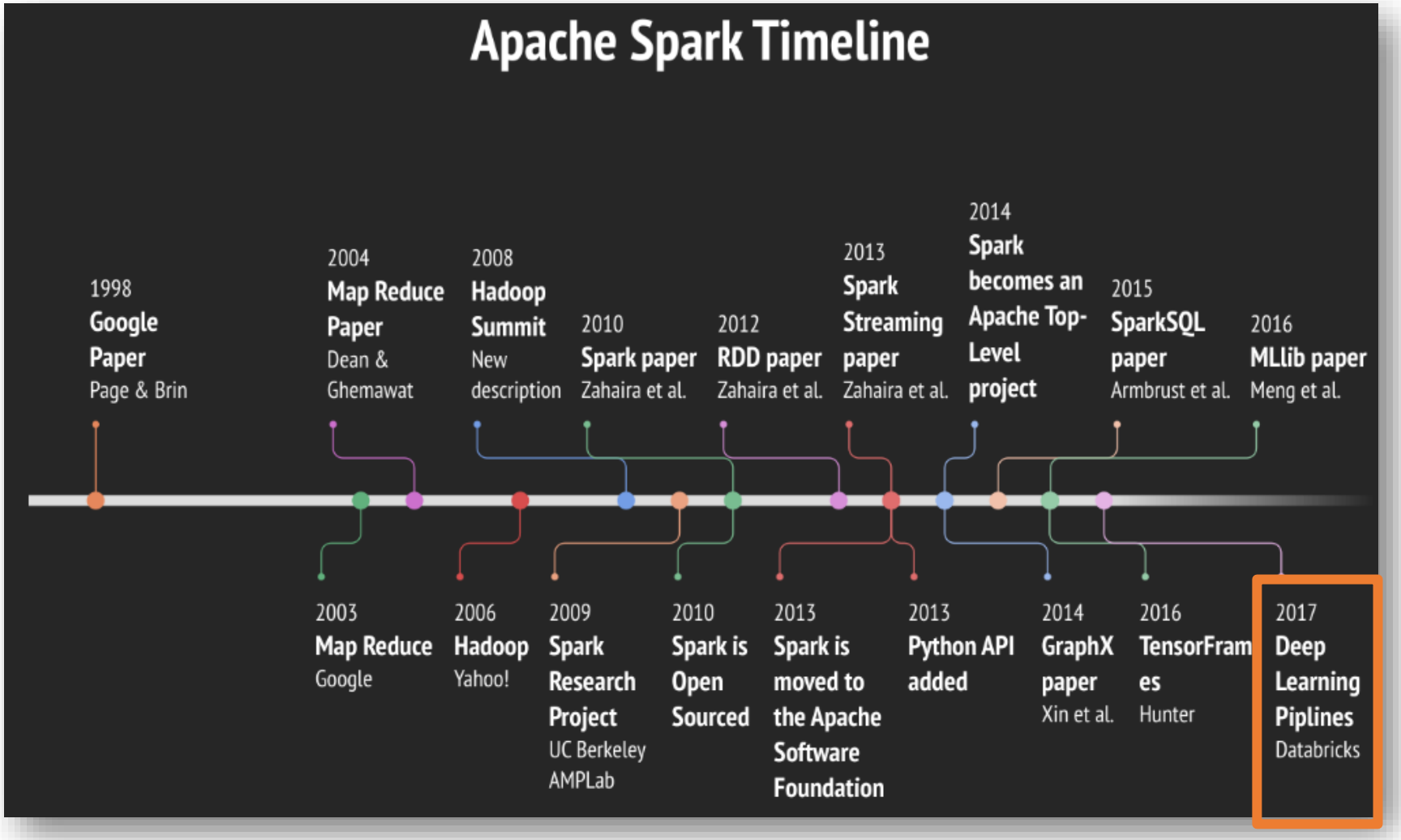
Deep learning with PySpark Mllib

Deep Learning Pipelines for Apache Spark

Deep Learning Pipelines is a new library published by Databricks to provide deep learning models and transfer learning via integration of popular deep learning libraries with MLlib Pipelines and Spark SQL.

Deep Learning Pipelines provides a suite of tools around working with and processing images using deep learning. The tools can be categorized as:

- **Working with images**
- **Transfer learning**
- **Applying deep learning models at scale**



Using Spark to load images as a DataFrame

MALMÖ UNIVERSITY

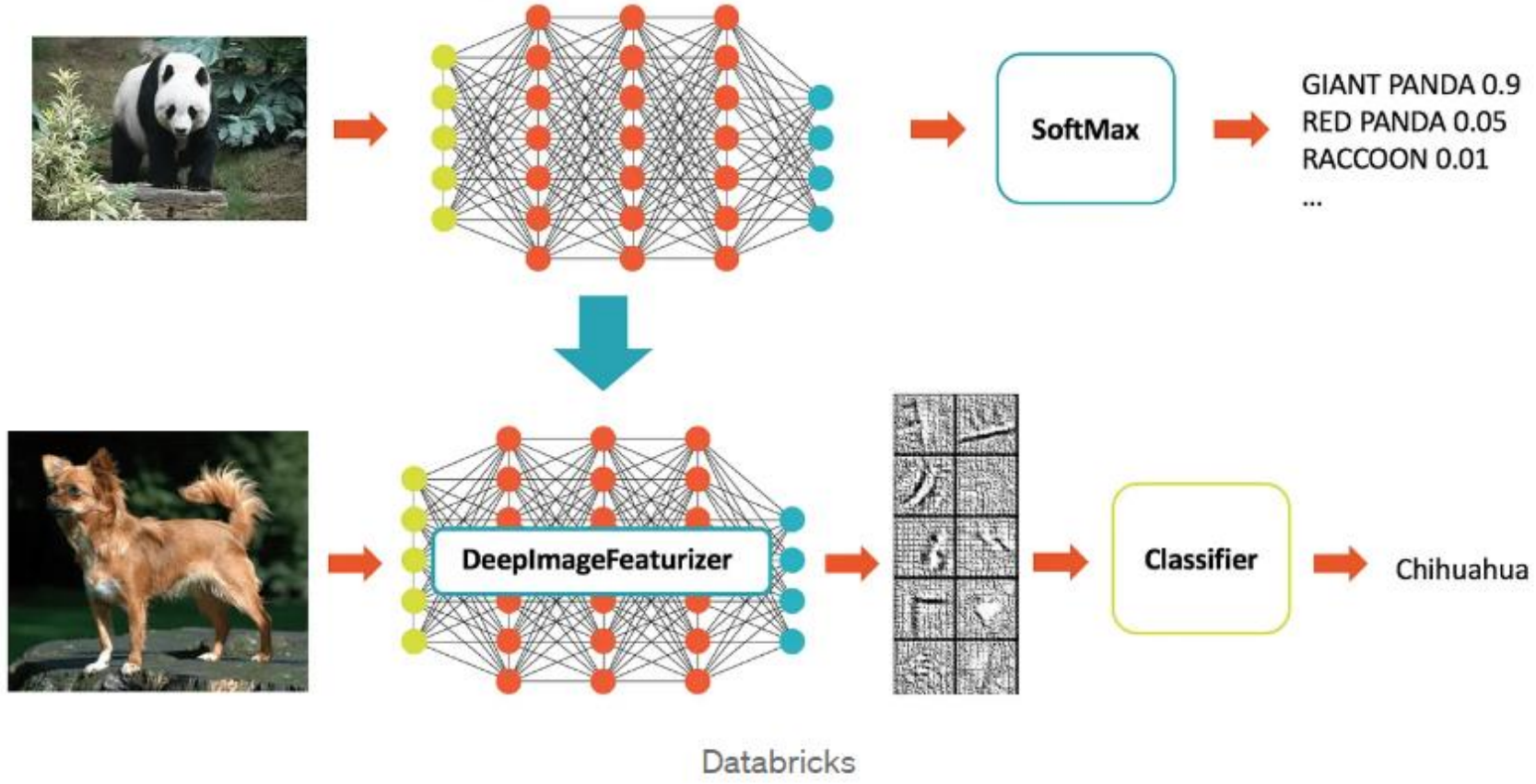
pip install sparkdl
from sparkdl import readImages

```
[21] image_df.show()
```

```
+--------------------+--------------------+
|            filePath|               image|
+--------------------+--------------------+
|file:/content/flo...|{RGB, 263, 320, 3...|
|file:/content/flo...|{RGB, 313, 500, 3...|
|file:/content/flo...|{RGB, 209, 320, 3...|
+--------------------+--------------------+
```

Transfer learning

Deep Learning Pipelines enables fast transfer learning with the concept of a *Featurizer*.



Transfer learning

MALMÖ UNIVERSITY

```
from pyspark.ml.classification import LogisticRegression
from pyspark.ml import Pipeline
from sparkdl import DeepImageFeaturizer

featurizer = DeepImageFeaturizer(inputCol="image", outputCol="features", modelName="InceptionV3")
lr = LogisticRegression(maxIter=10, regParam=0.05, elasticNetParam=0.3, labelCol="label")
p = Pipeline(stages=[featurizer, lr])

p_model = p.fit(train_df)
```

References

Machine Learning

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

- https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/5669198905533692/3647723071348946/3983381308530741/latest.html
- https://github.com/FavioVazquez/deep-learning-pyspark
- https://towardsdatascience.com/deep-learning-with-apache-spark-part-2-2a2938a36d35

Deep Learning