pyspark

November 27, 2023

[1]:
```
import pyspark
```

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

[3]:
```
spark = SparkSession.builder.appName('Practise').getOrCreate()
```

[4]:
```
spark
```

[4]:
```
<pyspark.sql.session.SparkSession at 0x1da10b4b680>
```

0.1 Read dataset

[5]:
```
df_pyspark = spark.read.csv("/Muhibuddin/python_project/LearnPySpark/dataset/
↪data_user.csv",header=True,inferSchema=True)
```

[6]:
```
df_pyspark.show()
```

```
+----------+---+----------+
|      name|age|experience|
+----------+---+----------+
|Muhibuddin| 25|         2|
|    Sumbul| 29|         6|
| Syahdudin| 22|         3|
|     Kenny| 16|         1|
+----------+---+----------+
```

[7]:
```
type(df_pyspark)
```

[7]:
```
pyspark.sql.dataframe.DataFrame
```

0.2 Check Schema

[8]:
```
df_pyspark.printSchema()
```

```
root
 |-- name: string (nullable = true)
 |-- age: integer (nullable = true)
 |-- experience: integer (nullable = true)
```

[9]:
```
df_pyspark.count()
```

[9]:
```
4
```

0.3 Select data

[10]:
```
df_pyspark.select(['name','age']).show()
```

```
+----------+---+
|      name|age|
+----------+---+
|Muhibuddin| 25|
|    Sumbul| 29|
| Syahdudin| 22|
|     Kenny| 16|
+----------+---+
```

[11]:
```
df_pyspark.head(2)
```

[11]:
```
[Row(name='Muhibuddin', age=25, experience=2),
 Row(name='Sumbul', age=29, experience=6)]
```

[12]:
```
### Add column
df_pyspark = df_pyspark.withColumn("Experience After 5␣
 ↪year",df_pyspark['experience']+5).withColumn('Age after 5␣
 ↪year',df_pyspark['age']+5)
```

[13]:
```
df_pyspark.show()
```

```
+----------+---+----------+----------------------+----------------+
|      name|age|experience|Experience After 5 year|Age after 5 year|
+----------+---+----------+----------------------+----------------+
|Muhibuddin| 25|         2|                     7|              30|
|    Sumbul| 29|         6|                    11|              34|
| Syahdudin| 22|         3|                     8|              27|
|     Kenny| 16|         1|                     6|              21|
+----------+---+----------+----------------------+----------------+
```

[14]:
```
## Drop column
df_pyspark = df_pyspark.drop('Experience After 5 year')
```

[15]:
```
df_pyspark.show()
```

```
+----------+---+----------+----------------+
|      name|age|experience|Age after 5 year|
+----------+---+----------+----------------+
|Muhibuddin| 25|         2|              30|
|    Sumbul| 29|         6|              34|
| Syahdudin| 22|         3|              27|
|     Kenny| 16|         1|              21|
+----------+---+----------+----------------+
```

[16]:
```
### Rename the column
df_pyspark.withColumnRenamed("Age after 5 year","New Age after 5 year").show()
```

```
+----------+---+----------+--------------------+
|      name|age|experience|New Age after 5 year|
+----------+---+----------+--------------------+
|Muhibuddin| 25|         2|                  30|
|    Sumbul| 29|         6|                  34|
| Syahdudin| 22|         3|                  27|
|     Kenny| 16|         1|                  21|
+----------+---+----------+--------------------+
```

0.3.1 Pyspark handling missing valuesabs

1. Dropping column
2. Dropping rows
3. Various Parameter in dropping functionalities
4. Handling missing values by Mean, Median and Mode

[17]:
```
from pyspark.sql import SparkSession
from pyspark.sql.functions import mean
spark = SparkSession.builder.appName("BaruBelajar").getOrCreate()
```

[18]:
```
df_user = spark.read.csv("/Muhibuddin/python_project/LearnPySpark/dataset/
↪data_user2.csv",header=True,inferSchema=True)
```

[19]:
```
df_user.printSchema()
```

```
root
 |-- name: string (nullable = true)
 |-- age: integer (nullable = true)
```

```
 |-- experience: integer (nullable = true)
 |-- salary: string (nullable = true)
```

[20]:
```
df_user.show()
```

```
+----------+----+----------+---------------+
|      name| age|experience|         salary|
+----------+----+----------+---------------+
|Muhibuddin|  25|         2| 30,000,000.00 |
|    Sumbul|  29|         6|  8,000,000.00 |
| Syahdudin|  22|         3|  4,000,000.00 |
|     Kenny|  16|         1|  3,500,000.00 |
|  Suryanto|  27|         5|  6,000,000.00 |
|     Yanti|  24|         3|  7,000,000.00 |
|   Kartono|NULL|         3|  3,000,000.00 |
|   Kartini|  43|        15| 40,000,000.00 |
|      NULL|NULL|        14|  4,050,000.00 |
|      NULL|  34|      NULL| 40,000,000.00 |
+----------+----+----------+---------------+
```

[21]:
```
### fill na age column using mean of age
average_age = df_user.select(mean('age')).collect()[0][0]

df_user.fillna(average_age,subset="age").show()
```

```
+----------+---+----------+---------------+
|      name|age|experience|         salary|
+----------+---+----------+---------------+
|Muhibuddin| 25|         2| 30,000,000.00 |
|    Sumbul| 29|         6|  8,000,000.00 |
| Syahdudin| 22|         3|  4,000,000.00 |
|     Kenny| 16|         1|  3,500,000.00 |
|  Suryanto| 27|         5|  6,000,000.00 |
|     Yanti| 24|         3|  7,000,000.00 |
|   Kartono| 27|         3|  3,000,000.00 |
|   Kartini| 43|        15| 40,000,000.00 |
|      NULL| 27|        14|  4,050,000.00 |
|      NULL| 34|      NULL| 40,000,000.00 |
+----------+---+----------+---------------+
```

[23]:
```
## Fill na using ml
from pyspark.ml.feature import Imputer

imputer = Imputer(
    inputCols=['age','experience'],
```

```
    outputCols=["{}_imputed".format(c) for c in ['age','experience']]
).setStrategy("mean")
```

[24]:
```
imputer.fit(df_user).transform(df_user).show()
```

```
+----------+----+----------+---------------+-----------+-----------------+
|      name| age|experience|         salary|age_imputed|experience_imputed|
+----------+----+----------+---------------+-----------+-----------------+
|Muhibuddin|  25|         2| 30,000,000.00 |         25|                2|
|    Sumbul|  29|         6|  8,000,000.00 |         29|                6|
| Syahdudin|  22|         3|  4,000,000.00 |         22|                3|
|     Kenny|  16|         1|  3,500,000.00 |         16|                1|
|  Suryanto|  27|         5|  6,000,000.00 |         27|                5|
|     Yanti|  24|         3|  7,000,000.00 |         24|                3|
|   Kartono|NULL|         3|  3,000,000.00 |         27|                3|
|   Kartini|  43|        15| 40,000,000.00 |         43|               15|
|      NULL|NULL|        14|  4,050,000.00 |         27|               14|
|      NULL|  34|      NULL| 40,000,000.00 |         34|                5|
+----------+----+----------+---------------+-----------+-----------------+
```

[25]:
```
df_user.select(mean("age")).show()
```

```
+--------+
|avg(age)|
+--------+
|    27.5|
+--------+
```

[28]:
```
### Drop column
## a row will be dropped only if all the values in that row are null
df_user.na.drop(how="all").show()
```

```
+----------+----+----------+---------------+
|      name| age|experience|         salary|
+----------+----+----------+---------------+
|Muhibuddin|  25|         2| 30,000,000.00 |
|    Sumbul|  29|         6|  8,000,000.00 |
| Syahdudin|  22|         3|  4,000,000.00 |
|     Kenny|  16|         1|  3,500,000.00 |
|  Suryanto|  27|         5|  6,000,000.00 |
|     Yanti|  24|         3|  7,000,000.00 |
|   Kartono|NULL|         3|  3,000,000.00 |
|   Kartini|  43|        15| 40,000,000.00 |
|      NULL|NULL|        14|  4,050,000.00 |
|      NULL|  34|      NULL| 40,000,000.00 |
+----------+----+----------+---------------+
```

[29]:
```
### Drop column
# a row will be dropped if it contains at least one null value.
df_user.na.drop(how="any").show()
```

```
+----------+---+----------+---------------+
|      name|age|experience|         salary|
+----------+---+----------+---------------+
|Muhibuddin| 25|         2| 30,000,000.00 |
|    Sumbul| 29|         6|  8,000,000.00 |
| Syahdudin| 22|         3|  4,000,000.00 |
|     Kenny| 16|         1|  3,500,000.00 |
|  Suryanto| 27|         5|  6,000,000.00 |
|     Yanti| 24|         3|  7,000,000.00 |
|   Kartini| 43|        15| 40,000,000.00 |
+----------+---+----------+---------------+
```

[30]:
```
### Drop column with threshold
## Rows are dropped only if they have less than or equal to 3 non-null values.
df_user.na.drop(how="all",thresh=3).show()
```

```
+----------+----+----------+---------------+
|      name| age|experience|         salary|
+----------+----+----------+---------------+
|Muhibuddin|  25|         2| 30,000,000.00 |
|    Sumbul|  29|         6|  8,000,000.00 |
| Syahdudin|  22|         3|  4,000,000.00 |
|     Kenny|  16|         1|  3,500,000.00 |
|  Suryanto|  27|         5|  6,000,000.00 |
|     Yanti|  24|         3|  7,000,000.00 |
|   Kartono|NULL|         3|  3,000,000.00 |
|   Kartini|  43|        15| 40,000,000.00 |
+----------+----+----------+---------------+
```

0.3.2 Pyspark Dataframes

1. FIlter Operation
2. &,|,==
3. ~

[54]:
```
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
```

[32]:
```
spark = SparkSession.builder.appName("Dataframe").getOrCreate()
```

[33]:
```
df_user = spark.read.csv("/Muhibuddin/python_project/LearnPySpark/dataset/
↪data_user2.csv",header=True,inferSchema=True)
```

[34]:
```
df_user.show()
```

```
+----------+----+----------+---------------+
|      name| age|experience|         salary|
+----------+----+----------+---------------+
|Muhibuddin|  25|         2| 30,000,000.00 |
|    Sumbul|  29|         6|  8,000,000.00 |
| Syahdudin|  22|         3|  4,000,000.00 |
|     Kenny|  16|         1|  3,500,000.00 |
|  Suryanto|  27|         5|  6,000,000.00 |
|     Yanti|  24|         3|  7,000,000.00 |
|   Kartono|NULL|         3|  3,000,000.00 |
|   Kartini|  43|        15| 40,000,000.00 |
|      NULL|NULL|        14|  4,050,000.00 |
|      NULL|  34|      NULL| 40,000,000.00 |
+----------+----+----------+---------------+
```

[35]:
```
average_age = df_user.select(mean('age')).collect()[0][0]

df_user = (df_user.dropna(subset="name")).fillna(subset='age',value=average_age)
```

[36]:
```
df_user.show()
```

```
+----------+---+----------+---------------+
|      name|age|experience|         salary|
+----------+---+----------+---------------+
|Muhibuddin| 25|         2| 30,000,000.00 |
|    Sumbul| 29|         6|  8,000,000.00 |
| Syahdudin| 22|         3|  4,000,000.00 |
|     Kenny| 16|         1|  3,500,000.00 |
|  Suryanto| 27|         5|  6,000,000.00 |
|     Yanti| 24|         3|  7,000,000.00 |
|   Kartono| 27|         3|  3,000,000.00 |
|   Kartini| 43|        15| 40,000,000.00 |
+----------+---+----------+---------------+
```

[39]:
```
## Clean data

df_user = df_user.withColumn('salary',regexp_replace(df_user['salary'], ",", "").
↪cast('int'))
```

[41]:
```
df_user.show()
```

```
+----------+---+----------+--------+
|      name|age|experience|  salary|
+----------+---+----------+--------+
|Muhibuddin| 25|         2|30000000|
|    Sumbul| 29|         6| 8000000|
| Syahdudin| 22|         3| 4000000|
|     Kenny| 16|         1| 3500000|
|  Suryanto| 27|         5| 6000000|
|     Yanti| 24|         3| 7000000|
|   Kartono| 27|         3| 3000000|
|   Kartini| 43|        15|40000000|
+----------+---+----------+--------+
```

0.4 Filter Operations

[42]:
```
df_user.filter("experience<=3").select('name','salary').show()
```

```
+----------+--------+
|      name|  salary|
+----------+--------+
|Muhibuddin|30000000|
| Syahdudin| 4000000|
|     Kenny| 3500000|
|     Yanti| 7000000|
|   Kartono| 3000000|
+----------+--------+
```

[43]:
```
df_user.filter(df_user['salary']>=10000000).show()
```

```
+----------+---+----------+--------+
|      name|age|experience|  salary|
+----------+---+----------+--------+
|Muhibuddin| 25|         2|30000000|
|   Kartini| 43|        15|40000000|
+----------+---+----------+--------+
```

[44]:
```
## using &
df_user.filter((df_user['age']>=20) & (df_user['salary']<=10000000)).show()
```

```
+---------+---+----------+-------+
|     name|age|experience| salary|
+---------+---+----------+-------+
|   Sumbul| 29|         6|8000000|
|Syahdudin| 22|         3|4000000|
```

```
| Suryanto| 27|          5|6000000|
|    Yanti| 24|          3|7000000|
|  Kartono| 27|          3|3000000|
+---------+---+-----------+-------+
```

[45]:
```
## using | (or)

df_user.filter((df_user['age']<25) | (df_user.salary >= 10000000)).show()
```

```
+----------+---+----------+--------+
|      name|age|experience|  salary|
+----------+---+----------+--------+
|Muhibuddin| 25|         2|30000000|
| Syahdudin| 22|         3| 4000000|
|     Kenny| 16|         1| 3500000|
|     Yanti| 24|         3| 7000000|
|   Kartini| 43|        15|40000000|
+----------+---+----------+--------+
```

[46]:
```
## using ~ (not)

df_user.filter(~(df_user.age<25)).show()
```

```
+----------+---+----------+--------+
|      name|age|experience|  salary|
+----------+---+----------+--------+
|Muhibuddin| 25|         2|30000000|
|    Sumbul| 29|         6| 8000000|
|  Suryanto| 27|         5| 6000000|
|   Kartono| 27|         3| 3000000|
|   Kartini| 43|        15|40000000|
+----------+---+----------+--------+
```

0.4.1 Pyspark Group by and Aggreagate Functions

[47]:
```
from pyspark.sql import SparkSession
from pyspark.sql.functions import sum
spark = SparkSession.builder.appName("SparkLearn").getOrCreate()
```

[48]:
```
df_user = spark.read.csv("/Muhibuddin/python_project/LearnPySpark/dataset/test3.
↪csv",header=True,inferSchema=True)
```

[49]:
```
df_user.show()
```

```
+---------+-----------+------+
|     Name| Departments|salary|
+---------+-----------+------+
|    Krish|Data Science| 10000|
|    Krish|         IOT|  5000|
|   Mahesh|    Big Data|  4000|
|    Krish|    Big Data|  4000|
|   Mahesh|Data Science|  3000|
|Sudhanshu|Data Science| 20000|
|Sudhanshu|         IOT| 10000|
|Sudhanshu|    Big Data|  5000|
|    Sunny|Data Science| 10000|
|    Sunny|    Big Data|  2000|
+---------+-----------+------+
```

[50]:
```
## groupBy (name) column
df_user.groupBy("Name").sum().show()
```

```
+---------+-----------+
|     Name|sum(salary)|
+---------+-----------+
|Sudhanshu|      35000|
|    Sunny|      12000|
|    Krish|      19000|
|   Mahesh|       7000|
+---------+-----------+
```

[51]:
```
## Group by departments which gives maximum salary
df_user.groupBy("Departments").sum().show()
```

```
+-----------+-----------+
| Departments|sum(salary)|
+-----------+-----------+
|         IOT|      15000|
|    Big Data|      15000|
|Data Science|      43000|
+-----------+-----------+
```

[52]:
```
## mean Agg
df_user.groupBy("Departments").mean().show()
```

```
+-----------+-----------+
| Departments|avg(salary)|
+-----------+-----------+
|         IOT|     7500.0|
```

```
|    Big Data|     3750.0|
|Data Science|    10750.0|
+------------+-----------+
```

[55]:
```
df_user.groupBy("Departments").agg(
    sum("salary").alias("Salary"),
    avg("Salary").alias("Avg Salary"),
    count("Departments").alias("Dept Count")
).show()
```

```
+------------+------+----------+----------+
| Departments|Salary|Avg Salary|Dept Count|
+------------+------+----------+----------+
|         IOT| 15000|    7500.0|         2|
|    Big Data| 15000|    3750.0|         4|
|Data Science| 43000|   10750.0|         4|
+------------+------+----------+----------+
```

0.5 Tutorial 6

0.5.1 Example of pyspark ML

[56]:
```
from pyspark.sql import SparkSession
spark.stop()
spark = SparkSession.builder.appName("SparkML").getOrCreate()
spark
```

[56]:
```
<pyspark.sql.session.SparkSession at 0x1da31be3f80>
```

[57]:
```
## Read data training

training = spark.read.csv("/Muhibuddin/python_project/LearnPySpark/dataset/test1.
 ↪csv", header=True, inferSchema=True)
training.printSchema()
```

```
root
 |-- Name: string (nullable = true)
 |-- age: integer (nullable = true)
 |-- Experience: integer (nullable = true)
 |-- Salary: integer (nullable = true)
```

[58]:
```
training.show()
```

```
+---------+---+----------+------+
|     Name|age|Experience|Salary|
+---------+---+----------+------+
|    Krish| 31|        10| 30000|
|Sudhanshu| 30|         8| 25000|
|    Sunny| 29|         4| 20000|
|     Paul| 24|         3| 20000|
|   Harsha| 21|         1| 15000|
|  Shubham| 23|         2| 18000|
+---------+---+----------+------+
```

[59]:
```
## show columns
training.columns
```

[59]:
```
['Name', 'age', 'Experience', 'Salary']
```

[60]:
```
from pyspark.ml.feature import VectorAssembler
featureassembler =␣
 ↪VectorAssembler(inputCols=["age","Experience"],outputCol="Independent Feture")
```

[61]:
```
output = featureassembler.transform(training)
```

[62]:
```
output.show()
```

```
+---------+---+----------+------+------------------+
|     Name|age|Experience|Salary|Independent Feture|
+---------+---+----------+------+------------------+
|    Krish| 31|        10| 30000|       [31.0,10.0]|
|Sudhanshu| 30|         8| 25000|        [30.0,8.0]|
|    Sunny| 29|         4| 20000|        [29.0,4.0]|
|     Paul| 24|         3| 20000|        [24.0,3.0]|
|   Harsha| 21|         1| 15000|        [21.0,1.0]|
|  Shubham| 23|         2| 18000|        [23.0,2.0]|
+---------+---+----------+------+------------------+
```

[63]:
```
output.columns
```

[63]:
```
['Name', 'age', 'Experience', 'Salary', 'Independent Feture']
```

[64]:
```
finalized_data = output.select("Independent Feture","Salary")
```

[65]:
```
finalized_data.show()
```

```
+------------------+------+
|Independent Feture|Salary|
+------------------+------+
```

```
|       [31.0,10.0]| 30000|
|        [30.0,8.0]| 25000|
|        [29.0,4.0]| 20000|
|        [24.0,3.0]| 20000|
|        [21.0,1.0]| 15000|
|        [23.0,2.0]| 18000|
+------------------+------+
```

[66]:
```
## train data
from pyspark.ml.regression import LinearRegression

train_data, test_data = finalized_data.randomSplit([0.75,0.25])
regressor = LinearRegression(featuresCol='Independent Feture', labelCol='Salary')
regressor = regressor.fit(train_data)
```

[67]:
```
### Coefficients

regressor.coefficients
```

[67]:
```
DenseVector([5000.0, -5000.0])
```

[68]:
```
## Intercept
regressor.intercept
```

[68]:
```
-84999.9999995608
```

[69]:
```
## prediction
pred_results = regressor.evaluate(test_data)
```

[70]:
```
pred_results.predictions.show()
```

```
+------------------+------+------------------+
|Independent Feture|Salary|        prediction|
+------------------+------+------------------+
|        [23.0,2.0]| 18000|19999.999999988155|
|        [29.0,4.0]| 20000|39999.999999911844|
|       [31.0,10.0]| 30000|20000.000000035623|
+------------------+------+------------------+
```

[71]:
```
pred_results.meanAbsoluteError,pred_results.meanSquaredError
```

[71]:
```
(10666.666666621459, 167999999.99857134)
```