PySpark - Intro to Mlib, Dataframe API, Example of Pyspark ML ¶

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

In [2]:
```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("Practice").getOrCreate()
```

In [3]:
```
spark
```

Out[3]:

SparkSession - in-memory

SparkContext

Spark UI (http://EXLAPLPNyCdxfzp.corp.exlservice.com:4040)

Version

v3.3.0

Master

local[*]

AppName

Practice

In [13]:
```
#reading dataset using spark session
df_pyspark=spark.read.csv("sparktest_3.csv", header=True, inferSchema=True)
df_pyspark.show()
```

```
+---------+--------+---+----------+------+
|     name|    dept|age|experience|salary|
+---------+--------+---+----------+------+
|    Krish|   Sales| 31|         6| 30000|
|Sudhanshu| Finance| 30|         4| 25000|
|    Sunny|      IT| 29|         4| 20000|
|     Paul|Products| 24|         3| 20000|
|   Harsha|   Sales| 21|         1| 15000|
|  Shubham|      IT| 23|         2| 18000|
|   Mahesh|Products| 35|        10| 40000|
|     Ravi| Finance| 34|         8| 38000|
|   Ankita| Finance| 36|        12| 50000|
+---------+--------+---+----------+------+
```

In [14]:
```
df_pyspark.columns
```

Out[14]:
```
['name', 'dept', 'age', 'experience', 'salary']
```

In [15]:
```
#Features grouping from list of two columns to one feature using vectorassembl
er
from pyspark.ml.feature import VectorAssembler
featureassembler=VectorAssembler(inputCols=["age","experience"],outputCol="ind
ependent_feature")
```

In [16]:
```
output=featureassembler.transform(df_pyspark)
```

In [17]:
```
output.show()
```

```
+---------+--------+---+----------+------+------------------+
|     name|    dept|age|experience|salary|independent_feature|
+---------+--------+---+----------+------+------------------+
|    Krish|   Sales| 31|         6| 30000|         [31.0,6.0]|
|Sudhanshu| Finance| 30|         4| 25000|         [30.0,4.0]|
|    Sunny|      IT| 29|         4| 20000|         [29.0,4.0]|
|     Paul|Products| 24|         3| 20000|         [24.0,3.0]|
|   Harsha|   Sales| 21|         1| 15000|         [21.0,1.0]|
|  Shubham|      IT| 23|         2| 18000|         [23.0,2.0]|
|   Mahesh|Products| 35|        10| 40000|        [35.0,10.0]|
|     Ravi| Finance| 34|         8| 38000|         [34.0,8.0]|
|   Ankita| Finance| 36|        12| 50000|        [36.0,12.0]|
+---------+--------+---+----------+------+------------------+
```

In [20]:
```
output.columns
```

Out[20]:
```
['name', 'dept', 'age', 'experience', 'salary', 'independent_feature']
```

In [21]:
```
finalized_data=output.select("independent_feature","salary")
```

In [22]:
```
finalized_data.show()
```

```
+------------------+------+
|independent_feature|salary|
+------------------+------+
|         [31.0,6.0]| 30000|
|         [30.0,4.0]| 25000|
|         [29.0,4.0]| 20000|
|         [24.0,3.0]| 20000|
|         [21.0,1.0]| 15000|
|         [23.0,2.0]| 18000|
|        [35.0,10.0]| 40000|
|         [34.0,8.0]| 38000|
|        [36.0,12.0]| 50000|
+------------------+------+
```

In [28]:
```
#Using Train, Test, Split
from pyspark.ml.regression import LinearRegression
train_data,test_data=finalized_data.randomSplit([0.75,0.25])
regressor=LinearRegression(featuresCol="independent_feature", labelCol="salar
y")
regressor=regressor.fit(train_data)
```

In [29]:
```
#Coefficients
regressor.coefficients
```

Out[29]:
```
DenseVector([-66.8709, 3278.7136])
```

In [30]:
```
#Intercepts
regressor.intercept
```

Out[30]:
```
12017.866258296788
```

In [31]:
```
#Prediction
pred_results=regressor.evaluate(test_data)
```

In [32]:
```
pred_results.predictions.show()
```

```
+-------------------+------+-----------------+
|independent_feature|salary|       prediction|
+-------------------+------+-----------------+
|         [21.0,1.0]| 15000| 13892.2919857072|
|         [31.0,6.0]| 30000|29617.15160796317|
+-------------------+------+-----------------+
```