NAME: Meith Navlakha
SAP: 60004190068
BDI Practical

EXPERIMENT 8

AIM: Execute any two ML Algorithms using Apache Spark MLlib and compare the results.

THEORY:

Spark MLlib:

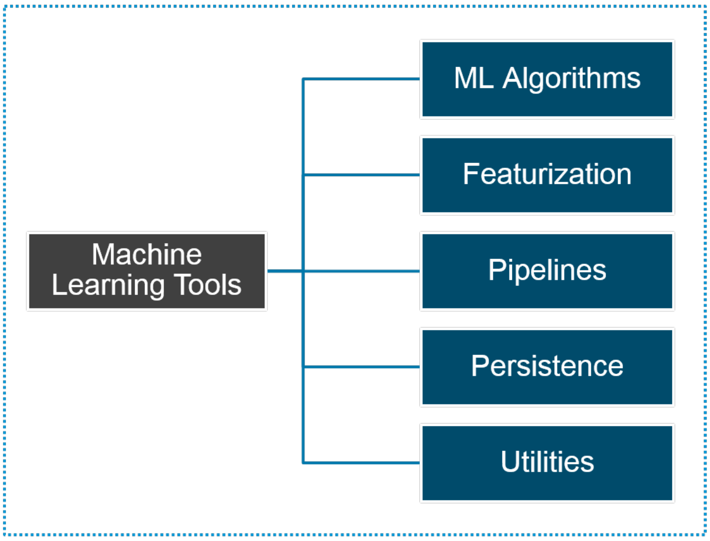
Spark MLlib is Apache Spark's Machine Learning component. MLlib consists of popular algorithms and utilities. MLlib in Spark is a scalable Machine learning library that discusses both high-quality algorithm and high speed. The machine learning algorithms like regression, classification, clustering, pattern mining, and collaborative filtering. Lower level machine learning primitives like generic gradient descent optimization algorithm are also present in MLlib.

spark.mllib contains the original API built on top of RDDs. It is currently in maintenance mode.

spark.ml provides higher level API built on top of DataFrames for constructing ML pipelines. spark.ml is the primary Machine Learning API for Spark at the moment.

Spark MLlib Tools :

- **ML Algorithms:** ML Algorithms form the core of MLlib. These include common learning algorithms such as classification, regression, clustering and collaborative filtering.
- **Featurization:** Featurization includes feature extraction, transformation, dimensionality reduction and selection.
- **Pipelines:** Pipelines provide tools for constructing, evaluating and tuning ML Pipelines.
- **Persistence:** Persistence helps in saving and loading algorithms, models and Pipelines.
- **Utilities:** Utilities for linear algebra, statistics and data handling.



NAME: Meith Navlakha
SAP: 60004190068
BDI Practical

MLlib Algorithms:

The popular algorithms and utilities in Spark MLlib are:

- Basic Statistics
- Regression
- Classification
- Recommendation System
- Clustering
- Dimensionality Reduction
- Feature Extraction
- Optimization

CODE:

```
# install pyspark
!pip install pyspark

# initialise session
from pyspark.context import SparkContext
from pyspark.sql.session import SparkSession
from pyspark.sql import SQLContext
sc = SparkContext('local')
spark = SparkSession(sc)
sqlContext = SQLContext(sc)
```

1) Regeression

```
df = sqlContext.read.format('com.databricks.spark.csv').options(header = 'true', inferschema=
'true').load('healthcare-dataset-stroke-data.csv')
df.take(1)

pd_df = df.toPandas()
print(pd_df)

pd_df.dtypes

df.cache()
df.printSchema()
df.cache()
df.printSchema()
df.head()

from pyspark.ml.feature import VectorAssembler
vectorAssembler = VectorAssembler(inputCols = ['age', 'avg_glucose_level', 'hypertension',
'heart_disease', 'stroke'], outputCol = 'features')
tdf = vectorAssembler.transform(df)
```

```
print(tdf)
tdf = tdf.select(['features', 'stroke'])
tdf.show(3)


splits = tdf.randomSplit([0.7, 0.3])
train_df = splits[0]
test_df = splits[1]


from pyspark.ml.regression import LinearRegression

lr = LinearRegression(featuresCol = 'features', labelCol='stroke', maxIter=10, regParam=0.3,
elasticNetParam=0.8)
lr_model = lr.fit(train_df)
print("Coefficients: " + str(lr_model.coefficients))
print("Intercept: " + str(lr_model.intercept))


trainingSummary = lr_model.summary
print("RMSE: %f" % trainingSummary.rootMeanSquaredError)
print("r2: %f" % trainingSummary.r2)
```

2) Classification

```
df = sqlContext.read.format('com.databricks.spark.csv').options(header = 'true', inferschema =
'true' ).load( 'salary.csv' )
df.take(1)

df.cache()
df.printSchema()

df.dtypes

from pyspark.ml.feature import StringIndexer

indexer0 = StringIndexer(inputCol="workclass", outputCol="workclassEncoded")
indexed0 = indexer0.fit(df).transform(df)
indexer1 = StringIndexer(inputCol="education", outputCol="educationEncoded")
indexed1 = indexer1.fit(indexed0).transform(indexed0)
indexer2 = StringIndexer(inputCol="occupation", outputCol="occupationEncoded")
indexed2 = indexer2.fit(indexed1).transform(indexed1)
indexer3 = StringIndexer(inputCol="sex", outputCol="sexEncoded")
indexed3 = indexer3.fit(indexed2).transform(indexed2)
indexer4 = StringIndexer(inputCol="native-country", outputCol="countryEncoded")
indexed4 = indexer4.fit(indexed3).transform(indexed3)
indexer5 = StringIndexer(inputCol="salary", outputCol="salaryEncoded")
indexed5 = indexer5.fit(indexed4).transform(indexed4)
indexed5.show()
```

NAME: Meith Navlakha
SAP: 60004190068
BDI Practical

```
from pyspark.ml.feature import VectorAssembler
vectorAssembler = VectorAssembler(inputCols = ['age', 'workclassEncoded', 'educationEncoded', 'occupationEncoded', 'sexEncoded', 'countryEncoded', 'education-num', 'capital-gain', 'hours-per-week', 'capital-loss'], outputCol = 'features')
tdf = vectorAssembler.transform(indexed5)
print(tdf)
tdf = tdf.select(['features', 'salaryEncoded'])
tdf.show(3)

train, test = tdf.randomSplit([0.7, 0.3], seed = 2018)
print("Training Dataset Count: " + str(train.count()))
print("Test Dataset Count: " + str(test.count()))

from pyspark.ml.classification import LogisticRegression
from pyspark.ml.classification import LinearSVC
lsvc = LinearSVC(featuresCol = 'features', labelCol = 'salaryEncoded', maxIter=10, regParam=0.1)
lr = LogisticRegression(featuresCol = 'features', labelCol = 'salaryEncoded', maxIter=10)
lsvcModel = lsvc.fit(train)
lrModel = lr.fit(train)

print("Coefficients: " + str(lsvcModel.coefficients))
print("Intercept: " + str(lsvcModel.intercept))

import matplotlib.pyplot as plt
trainingSummary = lrModel.summary
roc = trainingSummary.roc.toPandas()
plt.plot(roc['FPR'],roc['TPR'])
plt.ylabel('False Positive Rate')
plt.xlabel('True Positive Rate')
plt.title('ROC Curve')
plt.show()
print('Training set areaUnderROC: ' + str(trainingSummary.areaUnderROC))
```

NAME: Meith Navlakha
SAP: 60004190068
BDI Practical

OUTPUT:

1) Regression

```
         id  gender   age  hypertension  heart_disease ever_married  \
0      9046    Male  67.0             0              1          Yes
1     51676  Female  61.0             0              0          Yes
2     31112    Male  80.0             0              1          Yes
3     60182  Female  49.0             0              0          Yes
4      1665  Female  79.0             1              0          Yes
...     ...     ...   ...           ...            ...          ...
5105  18234  Female  80.0             1              0          Yes
5106  44873  Female  81.0             0              0          Yes
5107  19723  Female  35.0             0              0          Yes
5108  37544    Male  51.0             0              0          Yes
5109  44679  Female  44.0             0              0          Yes

          work_type Residence_type  avg_glucose_level   bmi   smoking_status  \
0           Private          Urban             228.69  36.6  formerly smoked
1     Self-employed          Rural             202.21   N/A     never smoked
2           Private          Rural             105.92  32.5     never smoked
3           Private          Urban             171.23  34.4           smokes
4     Self-employed          Rural             174.12    24     never smoked
...             ...            ...                ...   ...              ...
5105        Private          Urban              83.75   N/A     never smoked
5106  Self-employed          Urban             125.20    40     never smoked
5107  Self-employed          Rural              82.99  30.6     never smoked
5108        Private          Rural             166.29  25.6  formerly smoked
5109       Govt_job          Urban              85.28  26.2          Unknown
```

```
id                     int32
gender                object
age                  float64
hypertension           int32
heart_disease          int32
ever_married          object
work_type             object
Residence_type        object
avg_glucose_level    float64
bmi                   object
smoking_status        object
stroke                 int32
dtype: object
```

NAME: Meith Navlakha
SAP: 60004190068
BDI Practical

```
root
 |-- id: integer (nullable = true)
 |-- gender: string (nullable = true)
 |-- age: double (nullable = true)
 |-- hypertension: integer (nullable = true)
 |-- heart_disease: integer (nullable = true)
 |-- ever_married: string (nullable = true)
 |-- work_type: string (nullable = true)
 |-- Residence_type: string (nullable = true)
 |-- avg_glucose_level: double (nullable = true)
 |-- bmi: string (nullable = true)
 |-- smoking_status: string (nullable = true)
 |-- stroke: integer (nullable = true)
```

	0	1	2	3	4
summary	count	mean	stddev	min	max
id	5110	36517.82935420744	21161.72162482715	67	72940
gender	5110	None	None	Female	Other
age	5110	43.226614481409015	22.61264672311348	0.08	82.0
hypertension	5110	0.0974559686888454	0.296606674233791	0	1
heart_disease	5110	0.05401174168297456	0.22606298750336554	0	1
ever_married	5110	None	None	No	Yes
work_type	5110	None	None	Govt_job	children
Residence_type	5110	None	None	Rural	Urban
avg_glucose_level	5110	106.14767710371804	45.28356015058193	55.12	271.74
bmi	5110	28.893236911794673	7.85406672968016	10.3	N/A
smoking_status	5110	None	None	Unknown	smokes
stroke	5110	0.0487279843444227	0.21531985698023753	0	1

```
Row(id=9046, gender='Male', age=67.0, hypertension=0, heart_disease=1, ever_married='Yes',
work_type='Private', Residence_type='Urban', avg_glucose_level=228.69, bmi='36.6',
smoking_status='formerly smoked', stroke=1)
```

NAME: Meith Navlakha
SAP: 60004190068
BDI Practical

```
[('id', 'int'),
 ('gender', 'string'),
 ('age', 'double'),
 ('hypertension', 'int'),
 ('heart_disease', 'int'),
 ('ever_married', 'string'),
 ('work_type', 'string'),
 ('Residence_type', 'string'),
 ('avg_glucose_level', 'double'),
 ('bmi', 'string'),
 ('smoking_status', 'string'),
 ('stroke', 'int')]
```

```
DataFrame[id: int, gender: string, age: double, hypertension: int, heart_disease: int,
ever_married: string, work_type: string, Residence_type: string, avg_glucose_level: double,
bmi: string, smoking_status: string, stroke: int, features: vector]
+--------------------+------+
|            features|stroke|
+--------------------+------+
|[67.0,228.69,0.0,...|     1|
|[61.0,202.21,0.0,...|     1|
|[80.0,105.92,0.0,...|     1|
+--------------------+------+
only showing top 3 rows
```

Final Predictions

```
Coefficients: [0.0,0.0,0.0,0.0,0.0]
Intercept: 0.04802021903959562
```

```
RMSE: 0.213809
r2: -0.000000
```

NAME: Meith Navlakha
SAP: 60004190068
BDI Practical

2) Classification

```
[Row(age=39, workclass=' State-gov', fnlwgt=77516, education=' Bachelors',
education-num=13, marital-status=' Never-married', occupation=' Adm-clerical',
relationship=' Not-in-family', race=' White', sex=' Male', capital-gain=2174,
capital-loss=0, hours-per-week=40, native-country=' United-States', salary='
<=50K')]
```

```
root
 |-- age: integer (nullable = true)
 |-- workclass: string (nullable = true)
 |-- fnlwgt: integer (nullable = true)
 |-- education: string (nullable = true)
 |-- education-num: integer (nullable = true)
 |-- marital-status: string (nullable = true)
 |-- occupation: string (nullable = true)
 |-- relationship: string (nullable = true)
 |-- race: string (nullable = true)
 |-- sex: string (nullable = true)
 |-- capital-gain: integer (nullable = true)
 |-- capital-loss: integer (nullable = true)
 |-- hours-per-week: integer (nullable = true)
 |-- native-country: string (nullable = true)
 |-- salary: string (nullable = true)
```

age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	salary	workclassEncoded	educationEncoded	occupationEncoded	sexEncoded	countryEncoded	salaryEncoded
39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K	4.0	2.0	3.0	0.0	0.0	0.0
50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K	1.0	2.0	2.0	0.0	0.0	0.0
38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K	0.0	0.0	9.0	0.0	0.0	0.0
53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K	0.0	5.0	9.0	0.0	0.0	0.0
28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K	0.0	2.0	0.0	1.0	9.0	0.0
37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White	Female	0	0	40	United-States	<=50K	0.0	3.0	2.0	1.0	0.0	0.0
49	Private	160187	9th	5	Married-spouse-a...	Other-service	Not-in-family	Black	Female	0	0	16	Jamaica	<=50K	0.0	10.0	5.0	1.0	11.0	0.0

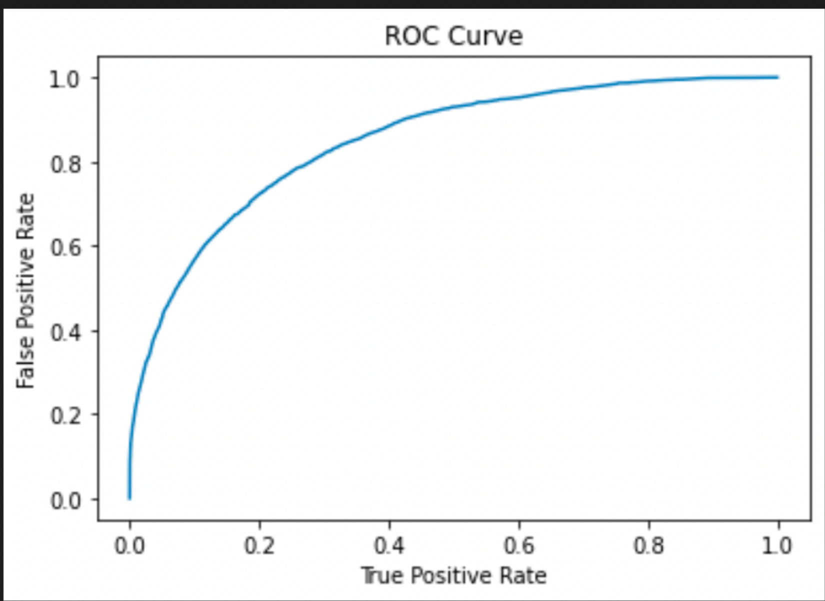
NAME: Meith Navlakha
SAP: 60004190068
BDI Practical

```
DataFrame[age: int, workclass: string, fnlwgt: int, education: string, education-num: int,
marital-status: string, occupation: string, relationship: string, race: string, sex: string,
capital-gain: int, capital-loss: int, hours-per-week: int, native-country: string, salary:
string, workclassEncoded: double, educationEncoded: double, occupationEncoded: double,
sexEncoded: double, countryEncoded: double, salaryEncoded: double, features: vector]
+--------------------+-------------+
|            features|salaryEncoded|
+--------------------+-------------+
|[39.0,4.0,2.0,3.0...|          0.0|
|[50.0,1.0,2.0,2.0...|          0.0|
|(10,[0,3,6,8],[38...|          0.0|
+--------------------+-------------+
only showing top 3 rows
```

```
Training Dataset Count: 22828
Test Dataset Count: 9733
```

Final Predictions

```
Coefficients:
[0.009679140612290072,0.01342892012614891,0.02288921743340083,-0.02078106627747353,-0.2950315
05,0.008667393334614108,0.0003678549301454241]
Intercept: -2.5735276509983613
```




```
Training set areaUnderROC: 0.8475822185984571
```

NAME: Meith Navlakha
SAP: 60004190068

CONCLUSION:

Thus in the experiment, we explored and learnt about pySpark's machine learning library – MLlib which supports Basic Statistics, Regression, Classification, Recommendation System, Clustering and many more. We implemented Linear Regression algorithm on stroke healthcare dataset having an RSME of 0.213 and an R^2 of 0.0 suggesting overfitting. We also implemented Classification on salary dataset giving us an AUC score of 0.847 suggesting a good fit. Thus, in this experimented we have implemented 2 ML algorithms supported by MLlib. The outputs were observed and attached above.