

The focus for implementing the ICP 6 is to execute MLlib is Apache Spark's scalable machine learning library, with APIs in Java, Scala, Python, and R.

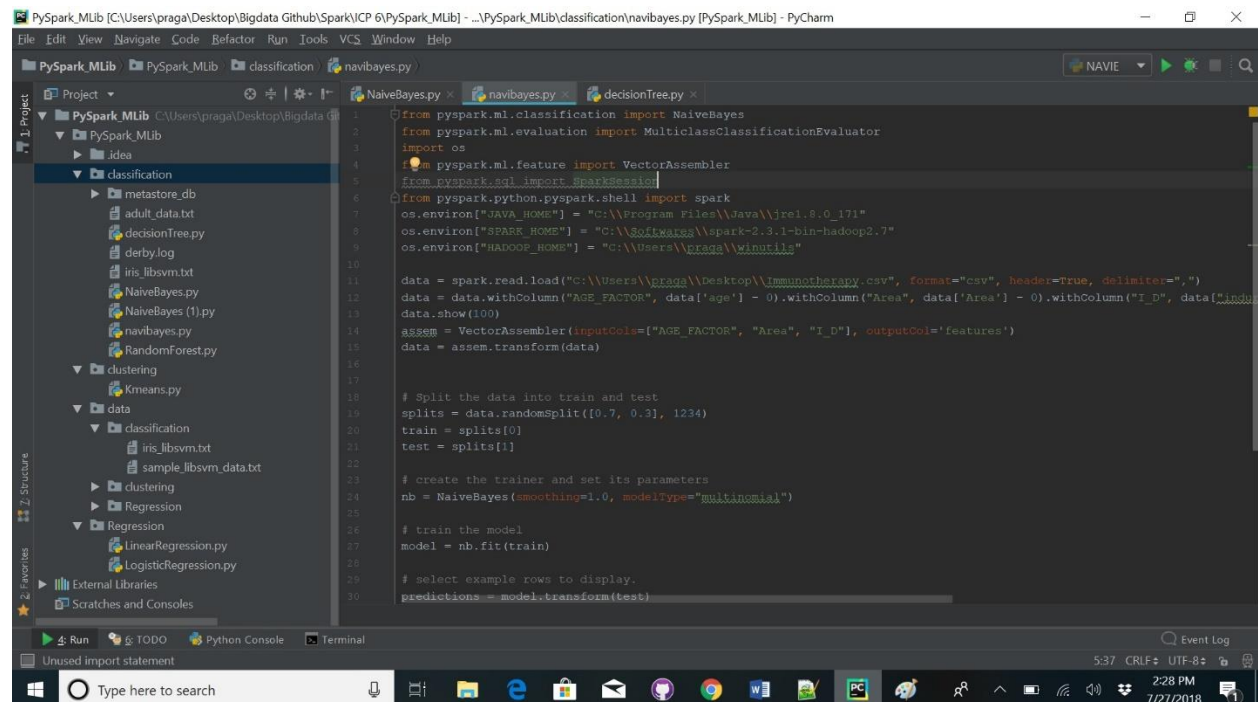
Implemented the **Naive Bayes classifier** on the csv data set. The data set which is used for the execution of the classifier is Immunotherapy.

Columns - sex, age, Time, Number_of_Warts, Type, Area, induration_diameter, (7 - features)
Result_of_Treatment(label)

Algorithm: Naive Bayes classifier, In machine learning, naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

This dataset contains information about wart treatment results of 90 patients using immunotherapy. This dataset has 7 different features like sex, age, time etc., and one label which has two categories (wart treatment curing the disease = 1, wart treatment not curing the disease = 0). In the below code we have only considered three features sex, age and induration_diameter features to predict the Result_of_Treatment. And the cross-validation parameter is set to 60,40 and 90,10 are used to test and train the model.

Code Snippet:



```

1 from pyspark.ml.classification import NaiveBayes
2 from pyspark.ml.evaluation import MulticlassClassificationEvaluator
3 import os
4 from pyspark.ml.feature import VectorAssembler
5 from pyspark.sql import SparkSession
6 from pyspark.python.pyspark.shell import spark
7 os.environ["JAVA_HOME"] = "C:\\Program Files\\Java\\jre1.8.0_171"
8 os.environ["SPARK_HOME"] = "C:\\Software\\spark-2.3.1-bin-hadoop2.7"
9 os.environ["HADOOP_HOME"] = "C:\\Users\\praga\\winutils"
10
11 data = spark.read.load("C:\\Users\\praga\\Desktop\\Immunotherapy.csv", format="csv", header=True, delimiter=",")
12 data = data.withColumn("AGE_FACTOR", data['age'] - 0).withColumn("Area", data['Area'] - 0).withColumn("I_D", data["induration_diameter"] - 0)
13 data.show(100)
14 assembler = VectorAssembler(inputCols=["AGE_FACTOR", "Area", "I_D"], outputCol='features')
15 data = assembler.transform(data)
16
17 # Split the data into train and test
18 splits = data.randomSplit([0.7, 0.3], 1234)
19 train = splits[0]
20 test = splits[1]
21
22 # create the trainer and set its parameters
23 nb = NaiveBayes(smoothing=1.0, modelType="multinomial")
24
25 # train the model
26 model = nb.fit(train)
27
28 # select example rows to display.
29 predictions = model.transform(test)
  
```

Output Screenshots:

60 and 40 split ratios -Test set accuracy 0.375

The screenshot shows the PyCharm IDE with the `NaiveBayes.py` file open. The code defines a `data` matrix and splits it into training and testing sets using a 60-40 ratio. A `NaiveBayes` model is trained and then used to predict the test set. The output console displays the test set accuracy as 0.375.

```
data.show(100)
data = VectorAssembler(inputCols=["AGE_FACTOR", "Area", "I_D"], outputCol='features')
data = assem.transform(data)

# Split the data into train and test
splits = data.randomSplit([0.6, 0.4], 123)
train = splits[0]
test = splits[1]

# create the trainer and set its parameters
nb = NaiveBayes(smoothing=1.0, modelType="multinomial")

# train the model
model = nb.fit(train)
```

Test set accuracy = 0.375
Process finished with exit code 0

70 and 30 split ratios -Test set accuracy -0.4375

The screenshot shows the PyCharm IDE with the `NaiveBayes.py` file open. The code defines a `data` matrix and splits it into training and testing sets using a 70-30 ratio. A `NaiveBayes` model is trained and then used to predict the test set. The output console displays the test set accuracy as -0.4375.

```
# train the model
model = nb.fit(train)

# select example rows to display.
predictions = model.transform(test)
predictions.show(100)

# compute accuracy on the test set
evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction",
                                              metricName="accuracy")

accuracy = evaluator.evaluate(predictions)
print("Test set accuracy = " + str(accuracy))
```

Test set accuracy = -0.4375
SUCCESS: The process with PID 7208 (child process of PID 7876) has been terminated.
SUCCESS: The process with PID 7876 (child process of PID 27180) has been terminated.
SUCCESS: The process with PID 27180 (child process of PID 27828) has been terminated.
Process finished with exit code 0