

# Naive Bayes

Naïve Bayes is a simple learning algorithm that utilizes Bayes rule together with a strong assumption that the attributes are conditionally independent, given the class. While this independence assumption is often violated in practice, naïve Bayes nonetheless often delivers competitive classification accuracy. Coupled with its computational efficiency and many other desirable features, this leads to naïve Bayes being widely applied in practice.

```
In [6]: import findspark
import pyspark
findspark.init()
```

```
In [7]: from pyspark.sql import SparkSession
from pyspark import SparkContext, SparkConf

spark = SparkSession.builder.appName('abc').getOrCreate()
sc = spark.sparkContext
```

Get IRIS Data

```
In [8]: from sklearn import datasets
import pandas as pd

# Load iris dataset
iris = datasets.load_iris()
irisDF = pd.DataFrame(iris.data, columns=iris.feature_names)
irisDF['target']=iris.target
irisDF.head(5)
```

```
Out[8]:
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

```
In [9]: df = spark.createDataFrame(irisDF)
df.show(5)
```

```
+-----+-----+-----+-----+-----+
|sepal length (cm)|sepal width (cm)|petal length (cm)|petal width (cm)|target|
+-----+-----+-----+-----+-----+
|          5.1|          3.5|          1.4|          0.2|      0|
|          4.9|          3.0|          1.4|          0.2|      0|
|          4.7|          3.2|          1.3|          0.2|      0|
|          4.6|          3.1|          1.5|          0.2|      0|
|          5.0|          3.6|          1.4|          0.2|      0|
+-----+-----+-----+-----+-----+
```

only showing top 5 rows

```
In [10]: from pyspark.ml.feature import RFormula

df2 = df.select(df["sepal length (cm)"].alias("sepal length"), df["sepal width (cm)"].alias("sepal width"), df["petal length (cm)"].alias("petal length"), df["petal width (cm)"].alias("petal width"))
#df2.show()

formula = RFormula(
    formula="target ~ sepal length + sepal width + petal length + petal width",
    featuresCol="features",
    labelCol="label")

output = formula.fit(df2).transform(df2)
output2=output.select("features", "label")
output2.show(5)
```

```
+-----+-----+
|      features|label|
+-----+-----+
|[5.1,3.5,1.4,0.2]| 0.0|
|[4.9,3.0,1.4,0.2]| 0.0|
|[4.7,3.2,1.3,0.2]| 0.0|
|[4.6,3.1,1.5,0.2]| 0.0|
|[5.0,3.6,1.4,0.2]| 0.0|
+-----+-----+
```

only showing top 5 rows

## Split to train and test

```
In [11]: train, test = output2.randomSplit([0.5, 0.5], seed=12345)
print('train:', train.count())
print('test:', test.count())
```

```
train: 68
test: 82
```

In [12]: `from pyspark.ml.classification import NaiveBayes`

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

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

# select example rows to display.
predictions = model.transform(test)
predictions.show(5)
```

features	label	rawPrediction	probability	prediction
[4.4,2.9,1.4,0.2]	0.0	[-10.796464806398...	[0.65738757331102...	0.0
[4.6,3.6,1.0,0.2]	0.0	[-10.934359345537...	[0.80167566255078...	0.0
[4.7,3.2,1.3,0.2]	0.0	[-11.145608801297...	[0.72355115765638...	0.0
[4.8,3.0,1.4,0.1]	0.0	[-10.835664246941...	[0.71513135355031...	0.0
[4.8,3.1,1.6,0.2]	0.0	[-11.685317645028...	[0.66837922051078...	0.0

only showing top 5 rows

In [15]: `from pyspark.ml.evaluation import MulticlassClassificationEvaluator`  
`evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="accuracy")`  
`accuracy = evaluator.evaluate(predictions)`  
`print("Test set accuracy = " + str(accuracy))`

Test set accuracy = 0.6219512195121951

In [ ]: