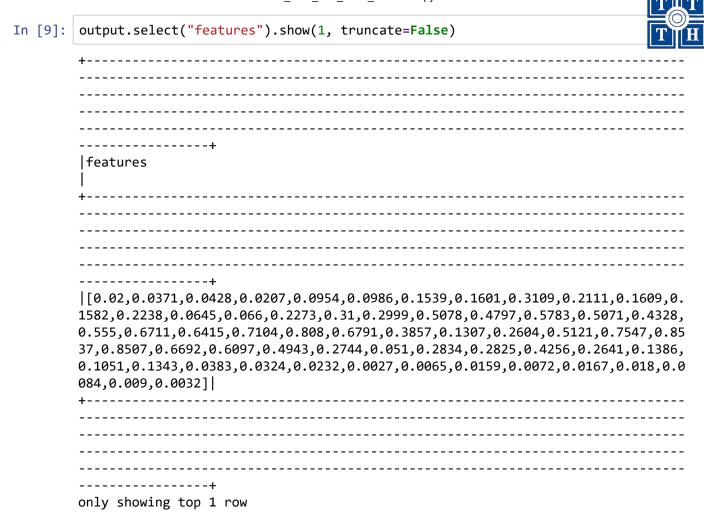


Chapter 13: PCA and Logistic Regression

```
In [1]:
           import findspark
            findspark.init()
 In [2]: from pyspark.sql import SparkSession
 In [3]: | spark = SparkSession.builder.appName('PCA sonar').getOrCreate()
 In [4]: df = spark.read.csv('sonar all data.txt',inferSchema=True,header=False)
In [33]: | str(df.schema.names)
Out[33]: "['_c0', '_c1', '_c2', '_c3', '_c4', '_c5', '_c6', '_c7', '_c8', '_c9',
                     '_c12', '_c13', '_c14', '_c15', '_c16', '_c17', '_c18', '_c19', '_c22', '_c23', '_c24', '_c25', '_c26', '_c27', '_c28', '_c29', '_c32', '_c33', '_c34', '_c35', '_c36', '_c37', '_c38', '_c39', '_c42', '_c43', '_c44', '_c45', '_c46', '_c47', '_c48', '_c49',
            '_c51', '_c52', '_c53', '_c54', '_c55', '_c56', '_c57', '_c58', '_c59', 'labe
            1'1"
 In [6]: | df = df.withColumnRenamed("_c60","label")
 In [7]: | from pyspark.ml.linalg import Vectors
            from pyspark.ml.feature import VectorAssembler
            from pyspark.ml.feature import StandardScaler
            from pyspark.ml.feature import StringIndexer
            from pyspark.ml.feature import PCA
 In [8]: assembler = VectorAssembler(
                 inputCols=['_c%d' % i for i in range(60)],
                outputCol="features")
            output = assembler.transform(df)
```



Scale feature if we need

Label to String

```
In [11]: indexer = StringIndexer(inputCol="label", outputCol="label_idx")
indexed = indexer.fit(output).transform(output)

In [12]: final_data = indexed.select(['std_features', 'label', 'label_idx'])
```



PCA

In [13]: final data.show(3)

```
In [14]: | pca = PCA(k=15, inputCol="std_features", outputCol="pca")
       model = pca.fit(final_data)
In [15]: | model.explainedVariance
Out[15]: DenseVector([0.2035, 0.189, 0.0855, 0.0568, 0.0501, 0.0406, 0.0328, 0.0305, 0.0
       257, 0.0249, 0.0208, 0.019, 0.0175, 0.0154, 0.0143])
       percent = model.explainedVariance
In [16]:
       type(percent)
Out[16]: pyspark.ml.linalg.DenseVector
In [17]: | percent.values.sum()
Out[17]: 0.8261807898020073
In [18]: transformed = model.transform(final_data)
In [19]: transformed.show(3)
               -----+
               std features|label|label idx|
       +-----
       only showing top 3 rows
In [20]: final data = transformed.select("label idx","pca")
```

Logistic Regression

```
In [21]: train_data,test_data = final_data.randomSplit([0.8,0.2])
```



```
In [22]: from pyspark.ml.classification import LogisticRegression
In [23]: logistic = LogisticRegression(featuresCol='pca',
                              labelCol='label idx',
                              predictionCol='prediction')
In [24]: # Fit the model to the data and call this model logisticModel
         logisticModel = logistic.fit(train data)
In [25]: # Create predictions for the testing data and show confusion matrix
         test model = logisticModel.transform(test data)
         test model.groupBy('label idx', 'prediction').count().show()
         +----+
         |label_idx|prediction|count|
         +----+
                                 14
               1.0
                          1.0
               0.0
                          1.0
                                  2
                          0.0
               1.0
                                 10
               0.0
                          0.0
                                 19|
In [26]: # Calculate the elements of the confusion matrix
         TN = test_model.filter('prediction = 0 AND label_idx = prediction').count()
         TP = test model.filter('prediction = 1 AND label idx = prediction').count()
         FN = test model.filter('prediction = 0 AND label idx != prediction').count()
         FP = test_model.filter('prediction = 1 AND label_idx != prediction').count()
In [27]: # Calculate precision and recall
         precision = TP / (TP + FP)
         recall = TP / (TP + FN)
         print('precision = {:.2f}\nrecall = {:.2f}'.format(precision, recall))
         precision = 0.88
         recall
                  = 0.58
In [28]:
         acc =(TP+TN)/test model.count()
Out[28]: 0.73333333333333333
```