```
!pip install pyspark
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting pyspark
  Downloading pyspark-3.3.1.tar.gz (281.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 281.4/281.4 MB 3.7 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting py4j==0.10.9.5
  Downloading py4j-0.10.9.5-py2.py3-none-any.whl (199 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 199.7/199.7 KB 9.0 MB/s eta 0:00:00
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.3.1-py2.py3-none-any.whl size=281845512 sha256=ecf286b27c06d165cb3c44bbd36c566154f4
  Stored in directory: /root/.cache/pip/wheels/43/dc/11/ec201cd671da62fa9c5cc77078235e40722170ceba231d7598
Successfully built pyspark
Installing collected packages: py4j, pyspark
Successfully installed py4j-0.10.9.5 pyspark-3.3.1
```

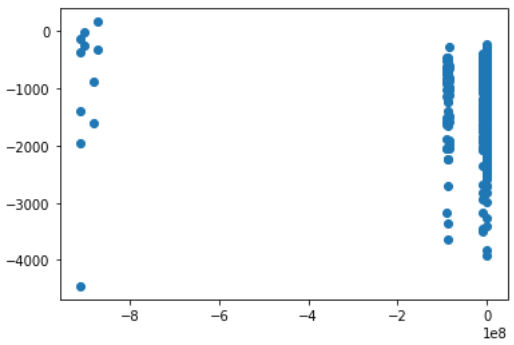
```
from pyspark.sql import SparkSession

spark = SparkSession.builder.getOrCreate()
```

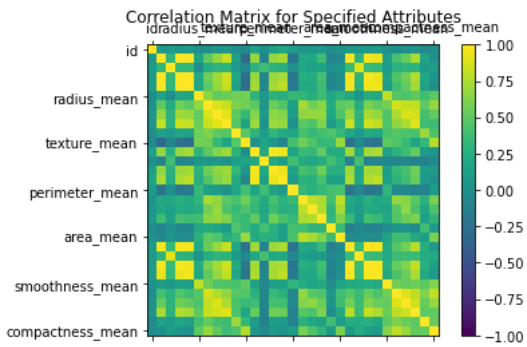
```
df = spark.read.csv("/content/drive/MyDrive/Memoire/data.csv", inferSchema = True, header = True)
df.show(5)
```

```
----+----------+----------------+-----------------+--------------+--------------------+--------------+----------------------+----+
orst|area_worst|smoothness_worst|compactness_worst|concavity_worst|concave points_worst|symmetry_worst|fractal_dimension_worst|_c32|
----+----------+----------------+-----------------+--------------+--------------------+--------------+----------------------+----+
34.6|    2019.0|          0.1622|           0.6656|        0.7119|              0.2654|        0.4601|                0.1189|null|
58.8|    1956.0|          0.1238|           0.1866|        0.2416|               0.186|         0.275|               0.08902|null|
52.5|    1709.0|          0.1444|           0.4245|        0.4504|               0.243|        0.3613|               0.08758|null|
3.87|     567.7|          0.2098|           0.8663|        0.6869|              0.2575|        0.6638|                 0.173|null|
52.2|    1575.0|          0.1374|            0.205|           0.4|              0.1625|        0.2364|               0.07678|null|
----+----------+----------------+-----------------+--------------+--------------------+--------------+----------------------+----+
```

```
import vizualiz
vizualiz.pca_viz(df)
```




```
vizualiz.plot_corr_matrix(df)
```




```
import mylib
train, test = mylib.process1(spark, "/content/drive/MyDrive/Memoire/data.csv")
```

```
train.show(3)
```

```
+--------------------+-----+
|            features|label|
+--------------------+-----+
|[15.46,19.48,101....|  1.0|
|[12.89,13.12,81.8...|  0.0|
|[12.94,16.17,83.1...|  0.0|
+--------------------+-----+
only showing top 3 rows
```

```
import logisticReg
model = logisticReg.logisticRegress(train)
pred = model.transform(test)
pred.show(3)
```

```
+--------------------+-----+--------------------+--------------------+----------+
|            features|label|       rawPrediction|         probability|prediction|
+--------------------+-----+--------------------+--------------------+----------+
|[14.96,19.1,97.03...|  0.0|[0.33562795650608...|[0.58312811193440...|       0.0|
|[20.26,23.03,132....|  1.0|[-8.3103904173978...|[2.45887526690253...|       1.0|
|[22.27,19.67,152....|  1.0|[-18.229529480162...|[1.21064254995793...|       1.0|
+--------------------+-----+--------------------+--------------------+----------+
only showing top 3 rows
```

```
import evaluation
evaluation.evaluate(pred)
```

```
Prediction Accuracy:  0.9866756009806882
Confusion Matrix:
[[114   0]
 [  6  59]]
```

```
import linearSVC
model1 = linearSVC.linearSVC(train)
pred = model1.transform(test)
pred.show(3)
```

```
+--------------------+-----+--------------------+----------+
|            features|label|       rawPrediction|prediction|
+--------------------+-----+--------------------+----------+
|[14.96,19.1,97.03...|  0.0|[0.29874844102278...|       0.0|
|[20.26,23.03,132....|  1.0|[-2.8804020293767...|       1.0|
|[22.27,19.67,152....|  1.0|[-6.5825620117094...|       1.0|
+--------------------+-----+--------------------+----------+
only showing top 3 rows
```

```
import evaluation
evaluation.evaluate(pred)
```

```
Prediction Accuracy:  0.9869800479667896
Confusion Matrix:
[[114   0]
 [  8  57]]
```

```
import decisionTree
model2 = decisionTree.decisionTreeClassifier(train)
pred = model2.transform(test)
pred.show(3)
```

```
+--------------------+-----+-------------+--------------------+----------+
|            features|label|rawPrediction|         probability|prediction|
+--------------------+-----+-------------+--------------------+----------+
|[14.96,19.1,97.03...|  0.0|   [22.0,1.0]|[0.95652173913043...|       0.0|
|[20.26,23.03,132....|  1.0|  [0.0,122.0]|           [0.0,1.0]|       1.0|
|[22.27,19.67,152....|  1.0|  [0.0,122.0]|           [0.0,1.0]|       1.0|
+--------------------+-----+-------------+--------------------+----------+
only showing top 3 rows
```

```
import evaluation
evaluation.evaluate(pred)
```

```
Prediction Accuracy:  0.9457433103855466
Confusion Matrix:
[[112   2]
 [  7  58]]
```

```
import randomForest
model3 = randomForest.randomForestClassifier(train)
```

```
pred = model3.transform(test)
pred.show(3)
```

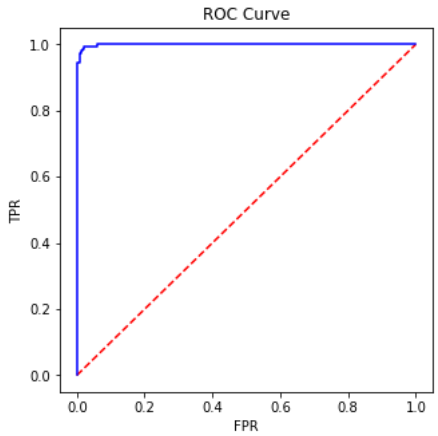
```
+--------------------+-----+-------------+-------------+----------+
|            features|label|rawPrediction|  probability|prediction|
+--------------------+-----+-------------+-------------+----------+
|[14.96,19.1,97.03...|  0.0|[326.0,174.0]|[0.652,0.348]|       0.0|
|[20.26,23.03,132....|  1.0|  [0.0,500.0]|    [0.0,1.0]|       1.0|
|[22.27,19.67,152....|  1.0|  [0.0,500.0]|    [0.0,1.0]|       1.0|
+--------------------+-----+-------------+-------------+----------+
only showing top 3 rows
```

```
import evaluation
evaluation.evaluate(pred)
```

```
Prediction Accuracy:  0.9821164798089573
Confusion Matrix:
[[112   2]
 [  5  60]]
```

```
import matplotlib.pyplot as plt

plt.figure(figsize=(5,5))
plt.plot([0, 1], [0, 1], 'r--')
plt.plot(model.summary.roc.select('FPR').collect(),
         model.summary.roc.select('TPR').collect(), color='b', label='lr')
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC Curve')
plt.show()
print('Training set areaUnderROC: ' + str(model.summary.areaUnderROC))
```




```
Training set areaUnderROC: 0.9990201842053693
```