

Made By Gafur Abdirakhman 180107121, Yerkebulan Gylymmadden 180107262, Nurbergen Akbergenov 180107286

Import necessary libraries

In [1]:

```
pip install pyspark
```

Collecting pyspark

 Downloading https://files.pythonhosted.org/packages/45/b0/9d6860891ab14a39d4bddf80ba26ce51c2f9dc4805e5c6978ac0472c120a/pyspark-3.1.1.tar.gz (212.3 MB)

 |██████████| 212.3MB 80kB/s

Collecting py4j==0.10.9

 Downloading https://files.pythonhosted.org/packages/9e/b6/6a4fb90cd235dc8e265a6a2067f2a2c99f0d91787f06aca4bcf7c23f3f80/py4j-0.10.9-py2.py3-none-any.whl (198kB)

 |██████████| 204kB 17.8MB/s

Building wheels for collected packages: pyspark

 Building wheel for pyspark (setup.py) ... done

 Created wheel for pyspark: filename=pyspark-3.1.1-py2.py3-none-any.whl size=212767604 sha256=d34863ca73d0e4f13570b19282889244b5e20288de99ff37e7efc3432c435b89

 Stored in directory: /root/.cache/pip/wheels/0b/90/c0/01de724414ef122bd05f056541fb6a0ecf47c7ca655f8b3c0f

Successfully built pyspark

Installing collected packages: py4j, pyspark

Successfully installed py4j-0.10.9 pyspark-3.1.1

In [2]:

```
import numpy as np

from matplotlib import pyplot as plt

import pyspark
from pyspark.sql import SparkSession
from pyspark.sql.functions import when, lit
from pyspark.sql.types import DoubleType
from pyspark.ml import feature
from pyspark.ml import classification
from pyspark.ml import evaluation
```

Basic info about dataset

Read dataset

In [3]:

```
spark = SparkSession.builder.appName("local").getOrCreate()
df = spark.read.option("header", True).csv("/kidney_disease.csv")
```

In [4]:

```
df.show()
```

```
+---+---+---+---+---+---+---+---+---+---+
| id| age| bp| sg| al| su| rbc| pc| pcc| ba|
| bgr| bu| sc| sod| pot|hemo| pcv| wc| rc|htn| dm|cad|appet| pe|ane|c
| lassification|
+---+---+---+---+---+---+---+---+---+---+
| 0|48.0| 80.0| 1.02| 1.0| 0.0| null| normal|notpresent|notpresent|12
| 1.0| 36.0| 1.2| null|null|15.4| 44| 7800| 5.2|yes|yes| no| good| no| no|
| ckd|
| 1| 7.0| 50.0| 1.02| 4.0| 0.0| null| normal|notpresent|notpresent| n
| ull| 18.0| 0.8| null|null|11.3| 38| 6000|null| no| no| no| good| no| no|
| ckd|
| 2|62.0| 80.0| 1.01| 2.0| 3.0| normal| normal|notpresent|notpresent|42
| 3.0| 53.0| 1.8| null|null| 9.6| 31| 7500|null| no|yes| no| poor| no|yes|
| ckd|
| 3|48.0| 70.0|1.005| 4.0| 0.0| normal|abnormal| present|notpresent|11
| 7.0| 56.0| 3.8|111.0| 2.5|11.2| 32| 6700| 3.9|yes| no| no| poor|yes|yes|
| ckd|
| 4|51.0| 80.0| 1.01| 2.0| 0.0| normal| normal|notpresent|notpresent|10
| 6.0| 26.0| 1.4| null|null|11.6| 35| 7300| 4.6| no| no| no| good| no| no|
| ckd|
| 5|60.0| 90.0|1.015| 3.0| 0.0| null| null|notpresent|notpresent| 7
| 4.0| 25.0| 1.1|142.0| 3.2|12.2| 39| 7800| 4.4|yes|yes| no| good|yes| no|
| ckd|
| 6|68.0| 70.0| 1.01| 0.0| 0.0| null| normal|notpresent|notpresent|10
| 0.0| 54.0|24.0|104.0| 4.0|12.4| 36| null|null| no| no| no| good| no| no|
| ckd|
| 7|24.0| null|1.015| 2.0| 4.0| normal|abnormal|notpresent|notpresent|41
| 0.0| 31.0| 1.1| null|null|12.4| 44| 6900| 5| no|yes| no| good|yes| no|
| ckd|
| 8|52.0|100.0|1.015| 3.0| 0.0| normal|abnormal| present|notpresent|13
| 8.0| 60.0| 1.9| null|null|10.8| 33| 9600| 4.0|yes|yes| no| good| no|yes|
| ckd|
| 9|53.0| 90.0| 1.02| 2.0| 0.0| abnormal|abnormal| present|notpresent| 7
| 0.0|107.0| 7.2|114.0| 3.7| 9.5| 29|12100| 3.7|yes|yes| no| poor| no|yes|
| ckd|
| 10|50.0| 60.0| 1.01| 2.0| 4.0| null|abnormal| present|notpresent|49
| 0.0| 55.0| 4.0| null|null| 9.4| 28| null|null|yes|yes| no| good| no|yes|
| ckd|
| 11|63.0| 70.0| 1.01| 3.0| 0.0| abnormal|abnormal| present|notpresent|38
| 0.0| 60.0| 2.7|131.0| 4.2|10.8| 32| 4500| 3.8|yes|yes| no| poor|yes| no|
| ckd|
| 12|68.0| 70.0|1.015| 3.0| 1.0| null| normal| present|notpresent|20
| 8.0| 72.0| 2.1|138.0| 5.8| 9.7| 28|12200| 3.4|yes|yes| yes| poor|yes| no|
| ckd|
| 13|68.0| 70.0| null|null|null| null| null|notpresent|notpresent| 9
| 8.0| 86.0| 4.6|135.0| 3.4| 9.8|null| null|null|yes|yes| yes| poor|yes| no|
| ckd|
| 14|68.0| 80.0| 1.01| 3.0| 2.0| normal|abnormal| present| present|15
| 7.0| 90.0| 4.1|130.0| 6.4| 5.6| 16|11000| 2.6|yes|yes|yes| poor|yes| no|
| ckd|
| 15|40.0| 80.0|1.015| 3.0| 0.0| null| normal|notpresent|notpresent| 7
| 6.0|162.0| 9.6|141.0| 4.9| 7.6| 24| 3800| 2.8|yes| no| no| good| no|yes|
| ckd|
| 16|47.0| 70.0|1.015| 2.0| 0.0| null| normal|notpresent|notpresent| 9
| 9.0| 46.0| 2.2|138.0| 4.1|12.6|null| null|null| no| no| no| good| no| no|
| ckd|
| 17|47.0| 80.0| null|null|null| null| null|notpresent|notpresent|11
```

```

4.0| 87.0| 5.2|139.0| 3.7|12.1|null| null|null|yes| no| no| poor| no| no|
ckd|
| 18|60.0|100.0|1.025| 0.0| 3.0|    null|   normal|notpresent|notpresent|26
3.0| 27.0| 1.3|135.0| 4.3|12.7| 37|11400| 4.3|yes|yes|yes| good| no| no|
ckd|
| 19|62.0| 60.0|1.015| 1.0| 0.0|    null|abnormal|   present|notpresent|10
0.0| 31.0| 1.6| null|null|10.3| 30| 5300| 3.7|yes| no|yes| good| no| no|
ckd|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
-----+
-----+
-----+
only showing top 20 rows

```

Brief information

In [5]:

```
df.count() #Number of rows/Length of dataset
```

Out[5]:

400

In [6]:

```
df.printSchema() #Information about type of columns
```

```

root
|-- id: string (nullable = true)
|-- age: string (nullable = true)
|-- bp: string (nullable = true)
|-- sg: string (nullable = true)
|-- al: string (nullable = true)
|-- su: string (nullable = true)
|-- rbc: string (nullable = true)
|-- pc: string (nullable = true)
|-- pcc: string (nullable = true)
|-- ba: string (nullable = true)
|-- bgr: string (nullable = true)
|-- bu: string (nullable = true)
|-- sc: string (nullable = true)
|-- sod: string (nullable = true)
|-- pot: string (nullable = true)
|-- hemo: string (nullable = true)
|-- pcv: string (nullable = true)
|-- wc: string (nullable = true)
|-- rc: string (nullable = true)
|-- htn: string (nullable = true)
|-- dm: string (nullable = true)
|-- cad: string (nullable = true)
|-- appet: string (nullable = true)
|-- pe: string (nullable = true)
|-- ane: string (nullable = true)
|-- classification: string (nullable = true)

```

In [7]:

```
for column in df.columns:  
    if column == 'id':  
        continue  
    df.describe([column]).show() #statistical information
```

```
+-----+-----+
|summary|      age|
+-----+-----+
|  count|     391|
|  mean| 51.48337595907928|
| stddev| 17.16971408926224|
|  min|      11.0|
|  max|      90.0|
+-----+-----+
```

```
+-----+-----+
|summary|      bp|
+-----+-----+
|  count|     388|
|  mean| 76.46907216494846|
| stddev| 13.68363749352527|
|  min|     100.0|
|  max|      90.0|
+-----+-----+
```

```
+-----+-----+
|summary|      sg|
+-----+-----+
|  count|     353|
|  mean|  1.0174079320113256|
| stddev| 0.005716616974376756|
|  min|      1.005|
|  max|      1.025|
+-----+-----+
```

```
+-----+-----+
|summary|      al|
+-----+-----+
|  count|     354|
|  mean| 1.0169491525423728|
| stddev| 1.3526789127628445|
|  min|      0.0|
|  max|      5.0|
+-----+-----+
```

```
+-----+-----+
|summary|      su|
+-----+-----+
|  count|     351|
|  mean| 0.45014245014245013|
| stddev|  1.099191251885407|
|  min|      0.0|
|  max|      5.0|
+-----+-----+
```

```
+-----+-----+
|summary|      rbc|
+-----+-----+
|  count|    248|
|  mean|    null|
| stddev|    null|
|  min|  abnormal|
|  max|   normal|
+-----+-----+
```

```
+-----+-----+
```

```
+-----+-----+
| summary|      pc|
+-----+-----+
|  count|     335|
|  mean|    null|
| stddev|    null|
|  min|abnormal|
|  max|  normal|
+-----+-----+
```

```
+-----+-----+
| summary|      pcc|
+-----+-----+
|  count|     396|
|  mean|    null|
| stddev|    null|
|  min|notpresent|
|  max|   present|
+-----+-----+
```

```
+-----+-----+
| summary|      ba|
+-----+-----+
|  count|     396|
|  mean|    null|
| stddev|    null|
|  min|notpresent|
|  max|   present|
+-----+-----+
```

```
+-----+-----+
| summary|      bgr|
+-----+-----+
|  count|     356|
|  mean|148.0365168539326|
| stddev|79.28171423511773|
|  min|       100.0|
|  max|       99.0|
+-----+-----+
```

```
+-----+-----+
| summary|      bu|
+-----+-----+
|  count|     381|
|  mean|57.425721784776904|
| stddev|50.503005849222504|
|  min|        1.5|
|  max|       98.6|
+-----+-----+
```

```
+-----+-----+
| summary|      sc|
+-----+-----+
|  count|     383|
|  mean|3.0724543080939934|
| stddev| 5.741126066859789|
|  min|        0.4|
|  max|       9.7|
+-----+-----+
```

```
+-----+-----+
| summary|      sod|
+-----+-----+
```

```
+-----+-----+
| count | 313 |
| mean | 137.52875399361022 |
| stddev | 10.408752051798777 |
| min | 104.0 |
| max | 4.5 |
+-----+-----+
```

```
+-----+-----+
| summary | pot |
+-----+-----+
| count | 312 |
| mean | 4.627243589743592 |
| stddev | 3.193904176556695 |
| min | 2.5 |
| max | 7.6 |
+-----+-----+
```

```
+-----+-----+
| summary | hemo |
+-----+-----+
| count | 348 |
| mean | 12.526436781609195 |
| stddev | 2.912586608826765 |
| min | 10.0 |
| max | 9.9 |
+-----+-----+
```

```
+-----+-----+
| summary | pcv |
+-----+-----+
| count | 330 |
| mean | 38.88449848024316 |
| stddev | 8.990104814740933 |
| min | 43 |
| max | 9 |
+-----+-----+
```

```
+-----+-----+
| summary | wc |
+-----+-----+
| count | 295 |
| mean | 8406.122448979591 |
| stddev | 2944.4741904103385 |
| min | 6200 |
| max | 9900 |
+-----+-----+
```

```
+-----+-----+
| summary | rc |
+-----+-----+
| count | 270 |
| mean | 4.707434944237919 |
| stddev | 1.0253232655721791 |
| min | ? |
| max | 8.0 |
+-----+-----+
```

```
+-----+
| summary | htn |
+-----+
```

```
|   count| 398|
|   mean|null|
| stddev|null|
|   min|  no|
|   max| yes|
+-----+---+  
  
+-----+---+
|summary| dm|
+-----+---+
|   count| 398|
|   mean|null|
| stddev|null|
|   min|      no|
|   max| yes|
+-----+---+  
  
+-----+---+
|summary| cad|
+-----+---+
|   count| 398|
|   mean|null|
| stddev|null|
|   min|      no|
|   max| yes|
+-----+---+  
  
+-----+---+
|summary| appet|
+-----+---+
|   count| 399|
|   mean| null|
| stddev|null|
|   min| good|
|   max| poor|
+-----+---+  
  
+-----+---+
|summary| pe|
+-----+---+
|   count| 399|
|   mean|null|
| stddev|null|
|   min|  no|
|   max| yes|
+-----+---+  
  
+-----+---+
|summary| ane|
+-----+---+
|   count| 399|
|   mean|null|
| stddev|null|
|   min|  no|
|   max| yes|
+-----+---+  
  
+-----+-----+
|summary|classification|
+-----+-----+
|   count|          400|
```

```
|   mean|      null|
| stddev|      null|
|   min|      ckd|
|   max|      notckd|
+-----+
```

Data clean

In [8]:

```
#check unique values for classification
#and to make sure that in future we will be able to use Machine Learning algorithms
df.select('classification').distinct().collect()
#as you can see it gives 3 values which are both of them is equivalent, but with /t
```

Out[8]:

```
[Row(classification='notckd'),
 Row(classification='ckd'),
 Row(classification='ckd\t')]
```

In [9]:

```
df.filter(df['classification']=="ckd\t").show()
#so, we are going to replace those 2 rows to ckd's rows.
```

```
+---+---+---+---+---+---+---+---+---+---+---+
-----+-----+-----+-----+-----+-----+-----+
-----+
| id| age| bp| sg| al| su| rbc| pc| pcc| ba| bgr|
bu| sc| sod| pot| hemo| pcv| wc| rc| htn| dm| cad| appet| pe| ane| classificat
ion|
+---+---+---+---+---+---+---+---+---+---+---+
-----+-----+-----+-----+-----+-----+-----+
-----+
| 37|72.0|80.0|null|null|null| null| null|notpresent|notpresent|137.0|
65.0|3.4|141.0|4.7| 9.7| 28|6900| 2.5|yes|yes| no| poor| no|yes|
ckd |
|230|65.0|60.0|1.01| 2.0| 0.0|normal|abnormal| present|notpresent|192.0|
17.0|1.7|130.0|4.3|null|null|9500|null|yes|yes| no| poor| no| no|
ckd |
+---+---+---+---+---+---+---+---+---+---+---+
-----+-----+-----+-----+-----+-----+-----+
-----+

```

In [10]:

```
#Let's do it with additional Libraries such as when and lit which are helps us with rep
lacing values
df = df.withColumn('classification', when(df['classification']=='ckd\t',
lit("ckd")).otherwise(df['classification']))
```

In [11]:

```
#we already replaced it, so, let's see  
df.filter(df['classification']=="ckd\\t").show()
```

```
+---+---+---+---+---+---+---+---+---+---+---+  
| id|age| bp| sg| al| su|rbc| pc|pcc| ba|bgr| bu| sc|sod|pot|hemo|pcv| wc|  
rc|htn| dm|cad|appet| pe|ane|classification|  
+---+---+---+---+---+---+---+---+---+---+---+  
|-----+-----+-----+-----+-----+-----+-----+  
+---+---+---+---+---+---+---+---+---+---+---+  
|-----+-----+-----+-----+-----+-----+-----+  
+---+---+---+---+---+---+---+---+---+---+---+
```

In [12]:

```
#so, we have only 2 values in classification column  
df.select('classification').distinct().collect()
```

Out[12]:

```
[Row(classification='notckd'), Row(classification='ckd')]
```

In [13]:

```
#drop all rows with nan  
df = df.na.drop()
```

In [14]:

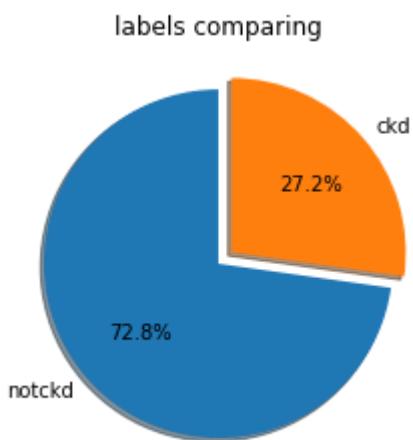
```
#let's check and count data which are missing values for each columns
for column in df.columns:
    print(column, "\n", "Missing values: ", df.filter(df[column].isNull()).count())
```

```
id
Missing values: 0
age
Missing values: 0
bp
Missing values: 0
sg
Missing values: 0
al
Missing values: 0
su
Missing values: 0
rbc
Missing values: 0
pc
Missing values: 0
pcc
Missing values: 0
ba
Missing values: 0
bgr
Missing values: 0
bu
Missing values: 0
sc
Missing values: 0
sod
Missing values: 0
pot
Missing values: 0
hemo
Missing values: 0
pcv
Missing values: 0
wc
Missing values: 0
rc
Missing values: 0
htn
Missing values: 0
dm
Missing values: 0
cad
Missing values: 0
appet
Missing values: 0
pe
Missing values: 0
ane
Missing values: 0
classification
Missing values: 0
```

labels comparing

In [15]:

```
a = df.select('classification').collect()
notckd = 0
ckd = 0
for i in a:
    if i[0] == 'ckd':
        ckd += 1
    else:
        notckd += 1
labels = 'notckd', 'ckd'
sizes = [notckd * 100 / (notckd + ckd), ckd * 100 / (notckd + ckd)]
explode = (0, 0.1)
fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
         shadow=True, startangle=90)
ax1.set_title('labels comparing')
plt.show() #Below, we can see that our label of the cleaned dataset is unbalanced, so accuracy metrics will be useless to estimate the model in the future.
```



Implementing Machine Learning algorithms

In [16]:

```
data = df.select('age', 'bp', 'sg', 'al', 'su', 'bgr', 'bu', 'sc', 'sod', 'pot', 'hemo',
                 'pcv', 'wc', 'rc', 'classification') #select columns which can be converted from string to double
```

In [17]:

```
for column in data.columns:
    if column == 'classification':
        break
    data = data.withColumn(column, data[column].cast(DoubleType())) #change type from string to double
```

In [18]:

```
colors = data.select('classification').collect()
c = ['blue' if str(i[0]) == 'notckd' else 'red' for i in colors] #Make list of color to
makr classification plot
```

In [19]:

```
data.show()
```

age	bp	sg	al	su	bgr	bu	sc	sod	pot	hemo	pcv	wc	r	c	classification
48.0	70.0	1.005	4.0	0.0	117.0	56.0	3.8	111.0	2.5	11.2	32.0	6700.0	3.	9	ckd
53.0	90.0	1.02	2.0	0.0	70.0	107.0	7.2	114.0	3.7	9.5	29.0	12100.0	3.	7	ckd
63.0	70.0	1.01	3.0	0.0	380.0	60.0	2.7	131.0	4.2	10.8	32.0	4500.0	3.	8	ckd
68.0	80.0	1.01	3.0	2.0	157.0	90.0	4.1	130.0	6.4	5.6	16.0	11000.0	2.	6	ckd
61.0	80.0	1.015	2.0	0.0	173.0	148.0	3.9	135.0	5.2	7.7	24.0	9200.0	3.	2	ckd
48.0	80.0	1.025	4.0	0.0	95.0	163.0	7.7	136.0	3.8	9.8	32.0	6900.0	3.	4	ckd
69.0	70.0	1.01	3.0	4.0	264.0	87.0	2.7	130.0	4.0	12.5	37.0	9600.0	4.	1	ckd
73.0	70.0	1.005	0.0	0.0	70.0	32.0	0.9	125.0	4.0	10.0	29.0	18900.0	3.	5	ckd
73.0	80.0	1.02	2.0	0.0	253.0	142.0	4.6	138.0	5.8	10.5	33.0	7200.0	4.	3	ckd
46.0	60.0	1.01	1.0	0.0	163.0	92.0	3.3	141.0	4.0	9.8	28.0	14600.0	3.	2	ckd
56.0	90.0	1.015	2.0	0.0	129.0	107.0	6.7	131.0	4.8	9.1	29.0	6400.0	3.	4	ckd
48.0	80.0	1.005	4.0	0.0	133.0	139.0	8.5	132.0	5.5	10.3	36.0	6200.0	4.	0	ckd
59.0	70.0	1.01	3.0	0.0	76.0	186.0	15.0	135.0	7.6	7.1	22.0	3800.0	2.	1	ckd
63.0	100.0	1.01	2.0	2.0	280.0	35.0	3.2	143.0	3.5	13.0	40.0	9800.0	4.	2	ckd
56.0	70.0	1.015	4.0	1.0	210.0	26.0	1.7	136.0	3.8	16.1	52.0	12500.0	5.	6	ckd
71.0	70.0	1.01	3.0	0.0	219.0	82.0	3.6	133.0	4.4	10.4	33.0	5600.0	3.	6	ckd
73.0	100.0	1.01	3.0	2.0	295.0	90.0	5.6	140.0	2.9	9.2	30.0	7000.0	3.	2	ckd
71.0	60.0	1.015	4.0	0.0	118.0	125.0	5.3	136.0	4.9	11.4	35.0	15200.0	4.	3	ckd
52.0	90.0	1.015	4.0	3.0	224.0	166.0	5.6	133.0	47.0	8.1	23.0	5000.0	2.	9	ckd
50.0	90.0	1.01	2.0	0.0	128.0	208.0	9.2	134.0	4.8	8.2	22.0	16300.0	2.	7	ckd

only showing top 20 rows

In [20]:

```
assembler = feature.VectorAssembler(inputCols=['age', 'bp', 'sg', 'al', 'su', 'bgr', 'bu', 'sc', 'sod', 'pot', 'hemo', 'pcv', 'wc', 'rc'],outputCol="features") #collect all columns to one feature
data = assembler.transform(data)
```

StandardScaler

In [21]:

```
scaler = feature.StandardScaler(inputCol="features", outputCol="SSout",
                                 withStd=True, withMean=True)
scalerModel = scaler.fit(data)
scaledData = scalerModel.transform(data)
```

PCA implementing

In [22]:

```
pca = feature.PCA(k=2, inputCol="SSout", outputCol="pcaFeatures")
model = pca.fit(scaledData)

result = model.transform(scaledData).select("pcaFeatures", "classification")
```

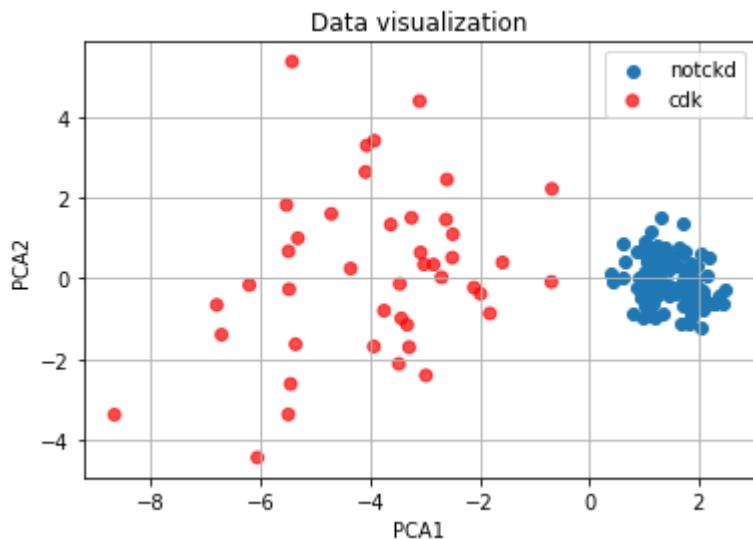
In [23]:

```
X = []
for i in result.select("pcaFeatures").collect():
    X.append([i[0][0], i[0][1]])
X = np.array(X) #Convert updated features from pyspark to numpy array
```

Data visualization

In [24]:

```
x_red = []
red = []
x_blue = []
blue = []
for i in range(len(X)):
    if c[i] == 'red':
        x_red.append(X[i])
        red.append(c[i])
    else:
        x_blue.append(X[i])
        blue.append(c[i])
x_red = np.array(x_red)
red = np.array(red)
x_blue = np.array(x_blue)
blue = np.array(blue)
plt.scatter(x_blue[:,0], x_blue[:,1], label = 'notckd')
plt.scatter(x_red[:,0], x_red[:,1], c=red, alpha=0.7, label = 'cdk')
plt.xlabel('PCA1')
plt.ylabel('PCA2')
plt.grid()
plt.legend()
plt.title('Data visualization')
plt.show()
```



In [25]:

```
result.show()
```

```
+-----+-----+
|      pcaFeatures | classification |
+-----+-----+
|[ -3.4298082810042...|      ckd|
|[ -3.4754560761380...|      ckd|
|[ -3.2446867740835...|      ckd|
|[ -5.4900484862099...|      ckd|
|[ -3.7443007061682...|      ckd|
|[ -2.9799547646873...|      ckd|
|[ -3.9236059743282...|      ckd|
|[ -2.4888323261282...|      ckd|
|[ -2.6950436305699...|      ckd|
|[ -2.4989563640813...|      ckd|
|[ -3.2873235172557...|      ckd|
|[ -3.9313672617946...|      ckd|
|[ -5.4937342909021...|      ckd|
|[ -2.5973258215255...|      ckd|
|[ -0.6788080196123...|      ckd|
|[ -3.0149307317597...|      ckd|
|[ -4.7085724777884...|      ckd|
|[ -2.8411973972218...|      ckd|
|[ -6.7142534874477...|      ckd|
|[ -5.3617086947228...|      ckd|
+-----+
only showing top 20 rows
```

Implementing StringIndexer

In [26]:

```
indexer = feature.StringIndexer(inputCol="classification", outputCol="categoryIndex")
indexed = indexer.fit(result).transform(result)
indexed.show() #Replace words with numbers
```

pcaFeatures	classification	categoryIndex
[-3.4298082810042...	ckd	1.0
[-3.4754560761380...	ckd	1.0
[-3.2446867740835...	ckd	1.0
[-5.4900484862099...	ckd	1.0
[-3.7443007061682...	ckd	1.0
[-2.9799547646873...	ckd	1.0
[-3.9236059743282...	ckd	1.0
[-2.4888323261282...	ckd	1.0
[-2.6950436305699...	ckd	1.0
[-2.4989563640813...	ckd	1.0
[-3.2873235172557...	ckd	1.0
[-3.9313672617946...	ckd	1.0
[-5.4937342909021...	ckd	1.0
[-2.5973258215255...	ckd	1.0
[-0.6788080196123...	ckd	1.0
[-3.0149307317597...	ckd	1.0
[-4.7085724777884...	ckd	1.0
[-2.8411973972218...	ckd	1.0
[-6.7142534874477...	ckd	1.0
[-5.3617086947228...	ckd	1.0

only showing top 20 rows

In [27]:

```
#we split data to train(90%) and test(10%)
train, test = indexed.select("pcaFeatures", "categoryIndex").randomSplit([0.9, 0.1], seed=1)
```

Implementing LG, DT, SVM and train them

In [28]:

```
#use three classification algorithms
lr = classification.LogisticRegression(featuresCol = 'pcaFeatures', labelCol = 'categoryIndex', predictionCol="predictions")
lrModel = lr.fit(train)
dt = classification.DecisionTreeClassifier(featuresCol = 'pcaFeatures', labelCol = 'categoryIndex', predictionCol="predictions")
dtModel = dt.fit(train)

svm = classification.LinearSVC(featuresCol = 'pcaFeatures', labelCol = 'categoryIndex', predictionCol="predictions")
svmModel = svm.fit(train)
```

Implementing Metrics

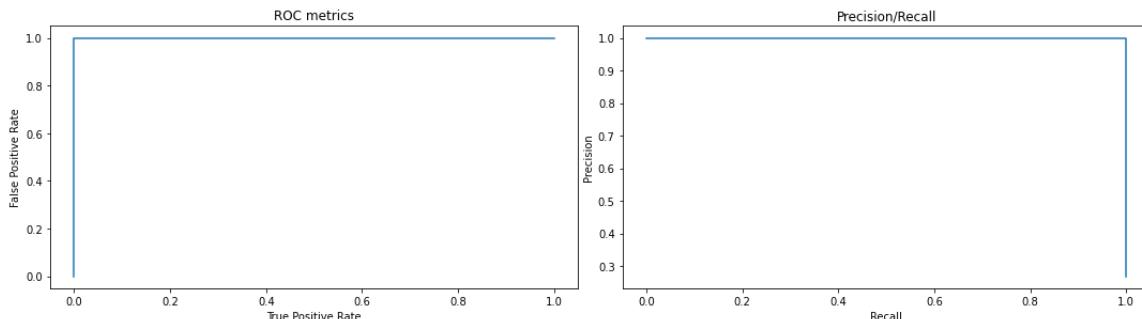
ROC, Precision, Recall

In [29]:

```
#add some metrics
trainingSummary = lrModel.summary
roc = trainingSummary.roc.toPandas()
pr = trainingSummary.pr.toPandas()

fig, (ax1, ax2) = plt.subplots(1, 2)
fig.set_figwidth(15)
fig.set_figheight(4)
fig.tight_layout()
ax1.plot(roc['FPR'],roc['TPR'])
ax1.set_title('ROC metrics')
ax1.set(xlabel = 'True Positive Rate', ylabel = 'False Positive Rate')

ax2.plot(pr['recall'],pr['precision'])
ax2.set_title('Precision/Recall')
ax2.set(xlabel = 'Recall', ylabel = 'Precision')
plt.show()
print('Training set areaUnderROC: ' + str(trainingSummary.areaUnderROC))
```



Training set areaUnderROC: 1.0

Make predictions

In [30]:

```
lr_predictions = lrModel.transform(test)
dt_predictions = dtModel.transform(test)
svm_predictions = svmModel.transform(test)
```

Accuracy

In [31]:

```
evaluator = evaluation.MulticlassClassificationEvaluator(  
    labelCol="categoryIndex", predictionCol="predictions", metricName="accuracy")  
accuracy = evaluator.evaluate(lr_predictions)  
print('Logistic Regression model accuracy = ' + str(accuracy*100) + '%')  
accuracy = evaluator.evaluate(dt_predictions)  
print('Decision Tree model accuracy = ' + str(accuracy*100) + '%')  
accuracy = evaluator.evaluate(svm_predictions)  
print('Support Vector Machine model accuracy = ' + str(accuracy*100) + '%')
```

```
Logistic Regression model accuracy = 100.0%  
Decision Tree model accuracy = 92.3076923076923%  
Support Vector Machine model accuracy = 100.0%
```