Proiect Big Data

1. Introducere **(Link)**

1.1. Prezentarea succinta a setului de date

Setul de date ales de mine a fost extras dintr-o bază de date unde au fost analizate datele a 1177 de pacienți cu insuficiență cardiacă. Acesta a fost realizat pentru a descrie rata mortalități pacienților aflați în unitățiile de la terapie intensivă din cauza problemelor cardiace. În urma analizei acestui dataset, am extras următoarele date: *caracteristici demografice* (varsta la momentul internării în spital, sex, etnie, greutate și înalțime), *semne vitale* (frecvență cardiacă[HR], tensiunea arterială sistolică[SBP], tensiunea arterială diastolică[DBP], tensiunea arterială medie[MBP], frecventa respiratorie[RR], temperatura corpului[BT], saturatia pulsului de oxygen[SPO2], urină), *comorbidități* (hipertensiune arterială, fibrilație atrială, boala cardiacă ischemică, diabet zaharat[DM], depresie, anemie hipoferică[HA], hiperlipidemie, boală renală cronică[CKD], boală pulmonară obstructivă cronică[BPOC]) și *variabile de laborator* (hematocrit, globule roșii, hemoglobina corpusculară medie [MCH], concentrația medie a hemoglobinei corpusculare [MCHC], volumul corpuscular mediu [MCV], lățimea distribuției globulelor roșii [RDW], numărul de trombocite, globule albe, neutrofile, bazofile, limfocite, timp de protrombină [PT], raport internațional normalizat [INR], NT-proBNP, creatin kinază, creatinină, azot ureic din sânge [BUN], glucoză, potasiu, sodiu, calciu, clorură, magneziu, gap anionic, bicarbonat, lactate, concentrația ionilor de hidrogen [pH], presiunea parțială a CO2 în sângele arterial și LVEF).

Caracteristicile demografice și semnele vitale extrase au fost înregistrate în primele 24 de ore de la fiecare internare și au fost măsurate variabilele de laborator pe toată durata șederii la unitățiile de la terapie intensivă. Comorbiditățile au fost identificate folosind codurile ICD-9. Pentru variabile de laborator au fost realizate măsurători multiple.

1.2. Enuntarea obiectivelor

Rezultatul primar al acestui proiect a fost mortalitatea în spital, definită ca starea vitală la momentul externării din spital la supraviețuitori și non-supraviețuitori. Dintre cele 1.177 de internări, mortalitatea în spital a fost de aproximativ 13,52%.

2. Procesarea datelor

Install Library

[1]
```
!pip install pyspark
!pip install numpy
!pip install pandas
!pip install matplotlib
!pip install seaborn
!pip install plotly
!pip install sklearn
```

```
Requirement already satisfied: pyspark in c:\users\bibinu\anaconda3\lib\site-packages (3.2.1)
Requirement already satisfied: py4j==0.10.9.3 in c:\users\bibinu\anaconda3\lib\site-packages (from pyspark) (0.10.9.3)
Requirement already satisfied: numpy in c:\users\bibinu\anaconda3\lib\site-packages (1.21.5)
Requirement already satisfied: pandas in c:\users\bibinu\anaconda3\lib\site-packages (1.4.2)
Requirement already satisfied: numpy>=1.18.5 in c:\users\bibinu\anaconda3\lib\site-packages (from pandas) (1.21.5)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\bibinu\anaconda3\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\bibinu\anaconda3\lib\site-packages (from pandas) (2021.3)
Requirement already satisfied: six>=1.5 in c:\users\bibinu\anaconda3\lib\site-packages (from python-dateutil>=2.8.1->pandas)
```

Loading Library

[2]
```
from time import time

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

from pyspark.sql import SparkSession
from pyspark.sql.functions import stddev,countDistinct,count,avg,col
from pyspark.sql.functions import when
from pyspark.sql.functions import mean

from pyspark.sql.types import IntegerType
from sklearn.impute import SimpleImputer
```

Spark Session

[3]
```
spark = SparkSession.builder.appName("Big_Data_Project").getOrCreate()
```

Saving Data as PySpark Dataframe

[4]
```
data_as_spark = spark.read.csv('original_data_null.csv',header=True)
data_as_spark_1 = spark.read.csv('original_data_.csv',header=True)
```

Data Analysis

Data Columns

[5]
```
data_as_spark.columns
```

```
['group',
 'ID',
 'outcome',
 'age',
 'gender',
 'BMI',
 'hypertensive',
 'atrialfibrillation',
 'CHD_with_no_MI',
 'diabetes',
 'deficiencyanemias',
 'depression',
 'Hyperlipemia',
 'Renull_failure',
 'COPD',
 'heart_rate',
 'Systolic_blood_pressure',
 'Diastolic_blood_pressure',
 'Respiratory_rate',
 'temperature',
 'SPO2',
 'Urine_output',
 'hematocrit',
 'hematocrit',
 'RBC',
 'MCH',
 'MCHC',
 'MCV',
 'RDW',
 'Leucocyte',
 'Platelets',
 'Neutrophils',
 'Basophils',
 'Lymphocyte',
 'PT',
 'INR',
 'NT-proBNP',
 'Creatine_kise',
 'Creatinine',
 'Urea_nitrogen',
 'glucose',
 'Blood_potassium',
 'Blood_sodium',
 'Blood_calcium',
 'Chloride',
 'Anion_gap',
 'Magnesium_ion',
 'PH',
 'Bicarbote',
 'Lactic_acid',
 'PCO2',
 'EF']
```

Data Scheme

[6]
```
data_as_spark.printSchema()
```

```
root
 |-- group: string (nullable = true)
 |-- ID: string (nullable = true)
 |-- outcome: string (nullable = true)
 |-- age: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- BMI: string (nullable = true)
 |-- hypertensive: string (nullable = true)
 |-- atrialfibrillation: string (nullable = true)
 |-- CHD_with_no_MI: string (nullable = true)
 |-- diabetes: string (nullable = true)
 |-- deficiencyanemias: string (nullable = true)
 |-- depression: string (nullable = true)
 |-- Hyperlipemia: string (nullable = true)
 |-- Renull_failure: string (nullable = true)
 |-- COPD: string (nullable = true)
```

Data Visualization

[7]
```
data_as_spark.createOrReplaceTempView("results")
sql_results_data_as_spark = spark.sql("select age Varsta,gender Sex,heart_rate Ritm_cardiac,depression Depresie,temperature Temperatura,Respiratory_rate Frecventa_respiratorie,Blood_calcium Calciu,Blood_potassium Potasiu,Blood_sodium Sodiu  from results")
sql_results_data_as_spark.show()
```

Varsta	Sex	Ritm_cardiac	Depresie	Temperatura	Frecventa_respiratorie	Calciu	Potasiu	Sodiu
72	1	68.83783784	0	36.71428571	16.62162162	7.463636364	4.816666667	138.75
75	2	101.3703704	0	36.68253968	20.85185185	8.1625	4.45	138.8888889
83	2	72.31818182	0	36.4537037	23.64	8.266666667	5.825	140.7142857
43	2	94.5	0	36.28703704	21.85714286	9.476923077	4.386666667	138.5
75	2	67.92	0	36.76190476	21.36	8.733333333	4.783333333	136.6666667
76	1	74.18181818	0	35.26666667	20.54545455	8.466666667	4.075	136.25
72	1	69.63636364	0	35.6031746	19.14814815	8.775	4.606666667	144.1333333
83	2	84.66666667	0	36.67361111	18.4	9.171428571	4.2375	140
61	2	91.91666667	0	37.1031746	18.58333333	9.44375	4.718181818	141.0909091
67	1	75.08333333	0	36.86111111	18.125	8.15	3.87	142.3
70	2	95.62962963	0	37.5555563	17.48148148	8.45	4.409090909	140.2727273
83	2	65.16	0	36.47777778	17.4	9.1	4.033333333	140.3333333
77	2	78.83333333	0	36.41666667	15.83333333	8.233333333	3.88	141.7
83	1	65.86956522	0	36.15740741	25.43478261	8.716666667	4.34	137.9
69	2	98.54411765	0	36.50925926	34.69343066	9.22	3.971428571	145.4285714
87	2	73.48	0	36.93333333	20.69230769	9.02	4.685714286	139.1428571
83	2	83.69230769	0	36.92222214	15.65217391	8.888888889	4.790909091	132.8181818
56	2	64.6	1	36.69444444	16.07142857	7.7	4.6	135.8333333
45	2	82	0	36.85185185	28.15384615	8.814285714	5.3	136.6363636
89	2	70.08333333	0	36.00793651	25.58333333	8.566666667	4	137.7777778

only showing top 20 rows

Data Summary

[8]

```
sql_describe_data_as_spark = spark.sql("select age Varsta,gender Sex,heart_rate Ritm_cardiac,temperature Temperatura,Blood_calcium Calciu,Blood_potassium Potasiu from results")
sql_describe_data_as_spark.summary().show()
```

summary	Varsta	Sex	Ritm_cardiac	Temperatura	Calciu	Potasiu
count	1177	1177	1177	1177	1177	1177
mean	74.05522514868309	1.5250637213254035	84.57584840750846	36.67728595220208	8.500894160843536	4.176646395857267
stddev	13.434060756372105	0.49958368741589076	16.018701499654277	0.6075583791550683	0.5722625957038024	0.41483603133775254
min	19	1	100.0357143	33.25	10.00769231	3
25%	65.0	1.0	72.36	36.28571429	8.145454545	3.9
50%	77.0	2.0	83.60869565	36.65079389	8.5	4.115384615
75%	85.0	2.0	95.89285714	37.02222222	8.86875	4.4
max	99	2	null	null	null	6.566666667

Visualization of age range

[9]
```
data_as_spark.createOrReplaceTempView("pacients")
sql_age_range_data_as_spark = spark.sql("select min(age) Varsta_minima , max(age) Varsta_maxima from pacients")
sql_age_range_data_as_spark.show()
```

```
+-------------+-------------+
|Varsta_minima|Varsta_maxima|
+-------------+-------------+
|           19|           99|
+-------------+-------------+
```

Age distribution

[10]
```
data_as_spark.createOrReplaceTempView("age_distrib")
sql_age_data_as_spark = spark.sql("SELECT `19-40`,`41-60`,`61-80`,`81-99` FROM ( select count(ID) `19-40` from age_distrib where age>=19 AND age<=40) JOIN (select count(ID) `41-60` from age_distrib where age>=41 AND age<=60) JOIN (select count(ID) `61-80` from age_distrib where age>=61 AND age<=80) JOIN (select count(ID) `81-99` from age_distrib where age>=81 AND age<=99)")
sql_age_data_as_spark.show()
```

```
+-----+-----+-----+-----+
|19-40|41-60|61-80|81-99|
+-----+-----+-----+-----+
|   20|  171|  517|  469|
+-----+-----+-----+-----+
```

Gender distribution

[11]
```
data_as_spark.createOrReplaceTempView("age_distrib")
sql_gender_data_as_spark = spark.sql("SELECT `No. Women`,`No. Men` FROM ( select count(ID) `No. Women` from age_distrib where gender==1) JOIN (select count(ID) `No. Men` from age_distrib where gender==2)")
sql_gender_data_as_spark.show()
```

```
+---------+-------+
|No. Women|No. Men|
+---------+-------+
|      559|    618|
+---------+-------+
```

Survivors distribution

[12]
```
sql_survivors_data_as_spark = spark.sql("SELECT `Survivors`,`Others` FROM ( select count(ID) `Survivors` from age_distrib
where outcome==0) JOIN (select count(ID) `Others` from age_distrib where outcome==1)")
sql_survivors_data_as_spark.show()
```

```
+---------+------+
|Survivors|Others|
+---------+------+
|     1017|   158|
+---------+------+
```

Data processing

Columns processing

Un prim pas pe care il consider important este redenumirea coloanelor setului de date. Unele dintre acestea contin anumite denumiri care nu sunt tocmai cunoscute. Pentru a observa care dintre coloane sunt importante, voi redenumi fiecare head cu o denumire mai sugestiva. Pentru a vizualiza denumirile initiale ale coloanelor, puteti observa rezultatul celulei [5].

[13]
```
data_as_spark=data_as_spark.withColumnRenamed('group','Grup')
data_as_spark=data_as_spark.withColumnRenamed('ID','Numar')
data_as_spark=data_as_spark.withColumnRenamed('outcome','Stare')
data_as_spark=data_as_spark.withColumnRenamed('age','Varsta')
data_as_spark=data_as_spark.withColumnRenamed('gender','Sex')
data_as_spark=data_as_spark.withColumnRenamed('BMI','Indice_de_masa_corporala')
data_as_spark=data_as_spark.withColumnRenamed('hypertensive','Hipertensiv')
data_as_spark=data_as_spark.withColumnRenamed('atrialfibrillation','Fibrilatie_atriala')
data_as_spark=data_as_spark.withColumnRenamed('CHD_with_no_MI','Boala_coronariana')
data_as_spark=data_as_spark.withColumnRenamed('diabetes','Diabet')
data_as_spark=data_as_spark.withColumnRenamed('deficiencyanemias','Anemii_deficitare')
data_as_spark=data_as_spark.withColumnRenamed('depression','Depresie')
data_as_spark=data_as_spark.withColumnRenamed('Hyperlipemia','Hiperlipemie')
data_as_spark=data_as_spark.withColumnRenamed('Renull_failure','Insuficienta_renala')
data_as_spark=data_as_spark.withColumnRenamed('COPD','Boala_pulmonara_obstructiva_cronica')
```

Pentru a vizualiza denumirile actualizate ale coloanelor, consultati notebook-ul atasat acestui document

[14]
```
data_as_spark.columns
```

```
['Grup',
 'Numar',
 'Stare',
 'Varsta',
 'Sex',
 'Indice_de_masa_corporala',
 'Hipertensiv',
 'Fibrilatie_atriala',
 'Boala_coronariana',
 'Diabet',
 'Anemii_deficitare',
 'Depresie',
 'Hiperlipemie',
 'Insuficienta_renala',
 'Boala_pulmonara_obstructiva_cronica',
 'Frecventa_cardiaca',
 'Tensiune_arteriala_sistolica',
 'Tensiune_arteriala_diastolica',
 'Frecventa_respiratorie',
 'Temperatura',
 'Saturatia_pulsului_de_oxigen',
```

Processing of the column 'Sex' values

[15]
```
data_as_spark = data_as_spark.withColumn("Sex",when(data_as_spark.Sex == "1","Female").when(data_as_spark.Sex == "2","Male").otherwise("null"))
```

Modifications

[16]
```
data_as_spark.createOrReplaceTempView("dataset")
sql_results_data_as_spark = spark.sql("select Varsta,Sex,Frecventa_cardiaca,Depresie,Temperatura,Frecventa_respiratorie from dataset")
sql_results_data_as_spark.show()
```

```
+------+------+------------------+--------+-----------+----------------------+
|Varsta|   Sex|Frecventa_cardiaca|Depresie|Temperatura|Frecventa_respiratorie|
+------+------+------------------+--------+-----------+----------------------+
|    72|Female|       68.83783784|       0|36.71428571|            16.62162162|
|    75|  Male|       101.3703704|       0|36.68253968|            20.85185185|
|    83|  Male|       72.31818182|       0| 36.4537037|                 23.64|
|    43|  Male|              94.5|       0|36.28703704|            21.85714286|
|    75|  Male|            67.92|       0|36.76190476|                 21.36|
|    76|Female|       74.18181818|       0|35.26666667|            20.54545455|
|    72|Female|       69.63636364|       0| 35.6031746|            19.14814815|
|    83|  Male|       84.66666667|       0|36.67361111|                  18.4|
|    61|  Male|       91.91666667|       0| 37.1031746|            18.58333333|
|    67|Female|       75.08333333|       0|36.86111111|                18.125|
|    70|  Male|       95.62962963|       0| 37.5555563|            17.48148148|
|    83|  Male|            65.16|       0|36.47777778|                  17.4|
|    77|  Male|       78.83333333|       0|36.41666667|            15.83333333|
|    83|Female|       65.86956522|       0|36.15740741|            25.43478261|
|    69|  Male|       98.54411765|       0|36.50925926|            34.69343066|
|    87|  Male|             73.48|       0|36.93333333|            20.69230769|
|    83|  Male|       83.69230769|       0|36.92222214|            15.65217391|
|    56|  Male|              64.6|       1|36.69444444|            16.07142857|
|    45|  Male|               82|       0|36.85185185|            28.15384615|
|    89|  Male|       70.08333333|       0|36.00793651|            25.58333333|
+------+------+------------------+--------+-----------+----------------------+
only showing top 20 rows
```

Processing of the column 'Stare' values

[17]
```
data_as_spark = data_as_spark.withColumn("Stare",when(data_as_spark.Stare == "0","alive").when(data_as_spark.Stare == "1","dead").otherwise("null"))
```

Modifications

[18]
```
data_as_spark.createOrReplaceTempView("dataset")
sql_results_data_as_spark = spark.sql("select Varsta,Sex,Stare,Frecventa_cardiaca,Depresie,Temperatura from dataset")
sql_results_data_as_spark.show()
```

```
+------+------+-----+------------------+--------+-----------+
|Varsta|   Sex|Stare|Frecventa_cardiaca|Depresie|Temperatura|
+------+------+-----+------------------+--------+-----------+
|    72|Female|alive|       68.83783784|       0|36.71428571|
|    75|  Male|alive|       101.3703704|       0|36.68253968|
|    83|  Male|alive|       72.31818182|       0| 36.4537037|
|    43|  Male|alive|              94.5|       0|36.28703704|
|    75|  Male|alive|             67.92|       0|36.76190476|
|    76|Female|alive|       74.18181818|       0|35.26666667|
|    72|Female|alive|       69.63636364|       0| 35.6031746|
|    83|  Male|alive|       84.66666667|       0|36.67361111|
|    61|  Male|alive|       91.91666667|       0| 37.1031746|
|    67|Female|alive|       75.08333333|       0|36.86111111|
|    70|  Male|alive|       95.62962963|       0| 37.5555563|
|    83|  Male|alive|             65.16|       0|36.47777778|
|    77|  Male|alive|       78.83333333|       0|36.41666667|
|    83|Female|alive|       65.86956522|       0|36.15740741|
|    69|  Male|alive|       98.54411765|       0|36.50925926|
|    87|  Male|alive|             73.48|       0|36.93333333|
|    83|  Male| dead|       83.69230769|       0|36.92222214|
|    56|  Male|alive|              64.6|       1|36.69444444|
|    45|  Male|alive|                82|       0|36.85185185|
|    89|  Male|alive|       70.08333333|       0|36.00793651|
+------+------+-----+------------------+--------+-----------+
only showing top 20 rows
```

Processing of the column 'Hipertensiv' values

[19]
```
data_as_spark = data_as_spark.withColumn("Hipertensiv",when(data_as_spark.Hipertensiv == "0","nu").when(data_as_spark.Hipertensiv == "1","da").otherwise("null"))
```

Modifications

[20]
```
data_as_spark.createOrReplaceTempView("dataset")
sql_results_data_as_spark = spark.sql("select Varsta,Sex,Hipertensiv,Frecventa_cardiaca,Depresie,Temperatura from dataset")
sql_results_data_as_spark.show()
```

```
+------+------+-----------+------------------+--------+-----------+
|Varsta|   Sex|Hipertensiv|Frecventa_cardiaca|Depresie|Temperatura|
+------+------+-----------+------------------+--------+-----------+
|    72|Female|         nu|       68.83783784|       0|36.71428571|
|    75|  Male|         nu|       101.3703704|       0|36.68253968|
|    83|  Male|         nu|       72.31818182|       0| 36.4537037|
|    43|  Male|         nu|              94.5|       0|36.28703704|
|    75|  Male|         da|             67.92|       0|36.76190476|
|    76|Female|         da|       74.18181818|       0|35.26666667|
|    72|Female|         da|       69.63636364|       0| 35.6031746|
|    83|  Male|         da|       84.66666667|       0|36.67361111|
|    61|  Male|         da|       91.91666667|       0| 37.1031746|
|    67|Female|         da|       75.08333333|       0|36.86111111|
|    70|  Male|         da|       95.62962963|       0| 37.5555563|
|    83|  Male|         da|             65.16|       0|36.47777778|
|    77|  Male|         da|       78.83333333|       0|36.41666667|
|    83|Female|         da|       65.86956522|       0|36.15740741|
|    69|  Male|         da|       98.54411765|       0|36.50925926|
|    87|  Male|         da|             73.48|       0|36.93333333|
|    83|  Male|         da|       83.69230769|       0|36.92222214|
|    56|  Male|         da|              64.6|       1|36.69444444|
|    45|  Male|         da|                82|       0|36.85185185|
|    89|  Male|         nu|       70.08333333|       0|36.00793651|
+------+------+-----------+------------------+--------+-----------+
only showing top 20 rows
```

Luand in considerare ca mai sunt inca 8 coloane de acest tip si ca formatarea acestora se va realiza in acelasi fel nu voi mai adauga celule in urma carora se vor modifica datele. Acestea se pot gasi in notebook-ul atasat documentului.

In acest moment trebuie sa ne ocupam de valorile null pe care le gasim in setul nostru de date. Daca incercam sa stergem toate randurile ce contin valori null vom observa conform celulei de mai jos ca vom ramanem cu 427 de linii ceea ce inseamna ca am sterge 750 de linii. Acest lucru nu este tocmai in regula, deoarece pierdem un procent foarte mare de date.

[37]
```
data_as_spark.createOrReplaceTempView("dataset")
sql_results_data_as_spark = spark.sql("select Varsta,Sex,Hipertensiv,Frecventa_cardiaca,Depresie,Temperatura from dataset")
sql_results_data_as_spark.show()
```

```
+-------+------------------+------------------+------------------+--------------------+------------------+
|summary|           Varsta|               Sex|Frecventa_cardiaca|            Depresie|       Temperatura|
+-------+------------------+------------------+------------------+--------------------+------------------+
|  count|               427|               427|               427|                 427|               427|
|   mean| 72.47540983606558| 1.515222482435597| 85.05205210149882|0.13114754098360656|36.709099645737695|
| stddev|13.416625853897502|0.5003544608930822|16.279529219221338|0.33795761084070836|  0.65745153466054|
|    min|                35|                 1|         100.09375|                   0|       34.32407407|
|    25%|              63.0|               1.0|       72.51851852|                 0.0|       36.27777778|
|    50%|              74.0|               2.0|       84.54166667|                 0.0|       36.66666667|
|    75%|              84.0|               2.0|       96.43478261|                 0.0|       37.07777778|
|    max|              99|                 2|       99.96666667|                   1|       39.13247842|
+-------+------------------+------------------+------------------+--------------------+------------------+
```

Pentru a micsora numarul de date pierdute in urma stergeri valorilor null, voi aborda urmatoarea strategie. Celulele null ce apartin de coloanele cu date valorice vor fi inlocuite cu media coloanei respective. In urma acestui pas, vom sterge restul celulelor null. Coloanele esentiale ce contin valori numerice sunt:

- 1.Indice_de_masa_corporala
- 2.Frecventa_cardiaca
- 3.Tensiune_arteriala_sistolica
- 4.Tensiune_arteriala_diastolica
- 5.Frecventa_respiratorie
- 6.Temperatura
- 7.Saturatia_pulsului_de_oxigen
- 8.Urina
- 9.Volumul_celule_rosii_din_sange
- 10.Globule_rosii
- 11.Hemoglobina_corpusculara_medie
- 12.Concentratia_medie_a_hemoglobinei_corpusculare
- 13.Volumul_corpuscular_mediu
- 14.Latimea_distributiei_globulelor_rosii
- 15.Leucocite
- 16.Trombocite
- 17.Neurofile
- 18.Globule_albe
- 19.Limfocite
- 20.Timp_de_protrombina
- 21.Raport_internationalizat_normalizat
- 22.NT_proBNP
- 23.Creatin_kinaza
- 24.Creatina
- 25.Nitrogen_ureic
- 26.Glucoza
- 27.Potasiu_din_sange
- 28.Sodiu_din_sange
- 29.Calciu_din_sange
- 30.Clorura
- 31.Interval_anionic
- 32.Ioni_de_magneziu
- 33.Concentratia_ionilor_de_oxigen
- 34.Bicarbote
- 35.Acid_lactic
- 36.Presiunea_partiala_a_dioxidului_de_carbon
- 37.Fractie_de_eliminare

Processing of the column 'Indice_de_masa_corporala' values

[38]
```
#Calculam media coloanei 'Indice_de_masa_corporala'
mean_val= data_as_spark.select(mean(data_as_spark['Indice_de_masa_corporala'])).collect()
mean_val
#Afisam valoarea medie a coloanei 'Indice_de_masa_corporala'
mean_val[0][0]
#Modificam valorile de null
data_as_spark = data_as_spark.withColumn("Indice_de_masa_corporala",when(data_as_spark.Indice_de_masa_corporala ==
"null",mean_val[0][0]).otherwise(data_as_spark.Indice_de_masa_corporala))
data_as_spark.createOrReplaceTempView("dataset")
sql_results_data_as_spark = spark.sql("select Sex,Indice_de_masa_corporala,Depresie,Temperatura from dataset")
sql_results_data_as_spark.show()
```

```
+------+-----------------------+--------+-----------+
|   Sex|Indice_de_masa_corporala|Depresie|Temperatura|
+------+-----------------------+--------+-----------+
|Female|             37.58817943|      nu|36.71428571|
|  Male|       30.18827765159043|      nu|36.68253968|
|  Male|             26.57263379|      nu| 36.4537037|
|  Male|             83.26462934|      nu|36.28703704|
|  Male|             31.82484194|      nu|36.76190476|
|Female|             24.26229342|      nu|35.26666667|
|Female|             39.66742627|      nu| 35.6031746|
|  Male|             22.31111111|      nu|36.67361111|
|  Male|             19.99224315|      nu| 37.1031746|
|Female|             45.03203011|      nu|36.86111111|
|  Male|             50.46121203|      nu| 37.5555563|
|  Male|             25.39189649|      nu|36.47777778|
|  Male|             22.69896194|      nu|36.41666667|
|Female|             33.89105707|      nu|36.15740741|
|  Male|                     20|      nu|36.50925926|
|  Male|             35.19894167|      nu|36.93333333|
|  Male|       30.18827765159043|      nu|36.92222214|
|  Male|              27.8516182|      da|36.69444444|
|  Male|             91.17665294|      nu|36.85185185|
|  Male|       30.18827765159043|      nu|36.00793651|
+------+-----------------------+--------+-----------+
only showing top 20 rows
```

Processing of the column 'Frecventa_cardiaca' values

[39]
```
# Calculam media coloanei 'Frecventa_cardiaca'
mean_val= data_as_spark.select(mean(data_as_spark['Frecventa_cardiaca'])).collect()
mean_val
#Afisam valoarea medie a coloanei 'Frecventa_cardiaca'
mean_val[0][0]
#Modificam valorile de null
data_as_spark = data_as_spark.withColumn("Frecventa_cardiaca",when(data_as_spark.Frecventa_cardiaca ==
"null",mean_val[0][0]).otherwise(data_as_spark.Frecventa_cardiaca))
data_as_spark.createOrReplaceTempView("dataset")
sql_results_data_as_spark = spark.sql("select Sex,Frecventa_cardiaca,Depresie,Temperatura from dataset")
sql_results_data_as_spark.show()
```

```
+------+------------------+--------+-----------+
|   Sex|Frecventa_cardiaca|Depresie|Temperatura|
+------+------------------+--------+-----------+
|Female|       68.83783784|      nu|36.71428571|
|  Male|       101.3703704|      nu|36.68253968|
|  Male|       72.31818182|      nu| 36.4537037|
|  Male|              94.5|      nu|36.28703704|
|  Male|             67.92|      nu|36.76190476|
|Female|       74.18181818|      nu|35.26666667|
|Female|       69.63636364|      nu| 35.6031746|
|  Male|       84.66666667|      nu|36.67361111|
|  Male|       91.91666667|      nu| 37.1031746|
|Female|       75.08333333|      nu|36.86111111|
|  Male|       95.62962963|      nu| 37.5555563|
|  Male|             65.16|      nu|36.47777778|
|  Male|       78.83333333|      nu|36.41666667|
|Female|       65.86956522|      nu|36.15740741|
|  Male|       98.54411765|      nu|36.50925926|
|  Male|             73.48|      nu|36.93333333|
|  Male|       83.69230769|      nu|36.92222214|
|  Male|              64.6|      da|36.69444444|
|  Male|                82|      nu|36.85185185|
|  Male|       70.08333333|      nu|36.00793651|
+------+------------------+--------+-----------+
only showing top 20 rows
```

Processing of the column 'Tensiune_arteriala_sistolica' values

[40]

```
# Calculam media coloanei 'Tensiune_arteriala_sistolica'
mean_val= data_as_spark.select(mean(data_as_spark['Tensiune_arteriala_sistolica'])).collect()
mean_val
#Afisam valoarea medie a coloanei 'Tensiune_arteriala_sistolica'
mean_val[0][0]
#Modificam valorile de null
data_as_spark = data_as_spark.withColumn("Tensiune_arteriala_sistolica",when(data_as_spark.Tensiune_arteriala_sistolica
== "null",mean_val[0][0]).otherwise(data_as_spark.Tensiune_arteriala_sistolica))
data_as_spark.createOrReplaceTempView("dataset")
sql_results_data_as_spark = spark.sql("select Sex,Tensiune_arteriala_sistolica,Depresie,Temperatura from dataset")
sql_results_data_as_spark.show()
```

```
+------+---------------------------+--------+-----------+
|   Sex|Tensiune_arteriala_sistolica|Depresie|Temperatura|
+------+---------------------------+--------+-----------+
|Female|                 155.8666667|      nu|36.71428571|
|  Male|                         140|      nu|36.68253968|
|  Male|                 135.3333333|      nu| 36.4537037|
|  Male|                       126.4|      nu|36.28703704|
|  Male|                      156.56|      nu|36.76190476|
|Female|                       118.1|      nu|35.26666667|
|Female|                 106.5652174|      nu| 35.6031746|
|  Male|                 141.1304348|      nu|36.67361111|
|  Male|                 98.43478261|      nu| 37.1031746|
|Female|                         122|      nu|36.86111111|
|  Male|                 149.0357143|      nu| 37.5555563|
|  Male|                 103.2608696|      nu|36.47777778|
|  Male|                 126.9032258|      nu|36.41666667|
|Female|                 112.1428571|      nu|36.15740741|
|  Male|                      107.36|      nu|36.50925926|
|  Male|                 159.6956522|      nu|36.93333333|
|  Male|                 157.2894737|      nu|36.92222214|
|  Male|                      113.28|      da|36.69444444|
|  Male|                      162.24|      nu|36.85185185|
|  Male|                 112.4166667|      nu|36.00793651|
+------+---------------------------+--------+-----------+
only showing top 20 rows
```

Luand in considerare ca mai sunt inca 34 coloane de acest tip si ca formatarea acestora se va realiza in acelasi fel nu voi mai adauga celule in urma carora se vor modifica datele. Acestea se pot gasi in notebook-ul atasat documentului.

In urma acestor modificari pe care le-am realizat, toate coloanele ce contin valori numerice nu mai contin valori null. In continuare, vom transforma valorile null ramase in coloanele ce nu contin valori numerice in string-uri de forma " " pentru a putea folosi functia care ne va sterge randurile cu valori null. Coloana pe care o mai avem de modificat este Stare care mai contine 2 valori null.

[75]
```
data_as_spark = data_as_spark.withColumn("Stare",when(data_as_spark.Stare == "null","").otherwise(data_as_spark.Stare))
```

Delete rows that contains “ ” type of strings

[76]
```
data_as_spark=data_as_spark.where(data_as_spark.Stare !="")
```

In urmatoarea celula, intalnim numarul de linii ce se afla in setul meu de date ales in urma tuturor modificariilor efectuate. Conform outputului acesteia, solutia aleasa de mine de a gestiona valorile null ale dataframe-ului s-a dovedit a fi eficienta .

[77]
```
data_as_spark.count()
```

```
1175
```

In urma analizarii rezultatelor obtinute, coloana Grup contine 2 valori si anume 1 si 2. Aceste valori nu ne sunt favorabile rezultatelor pe care le cautam. Ca urmare, vom sterge aceasta coloana din setul nostru de date.

[78]
```
data_as_spark= data_as_spark.drop("Grup")
```

Transform Pyspark Dataframe to Pandas Dataframe

[79]
```
data_as_csv = data_as_spark.toPandas()
```

Saving the processed dataset

[80]
```
data_as_csv.to_csv('processed_data.csv')
```

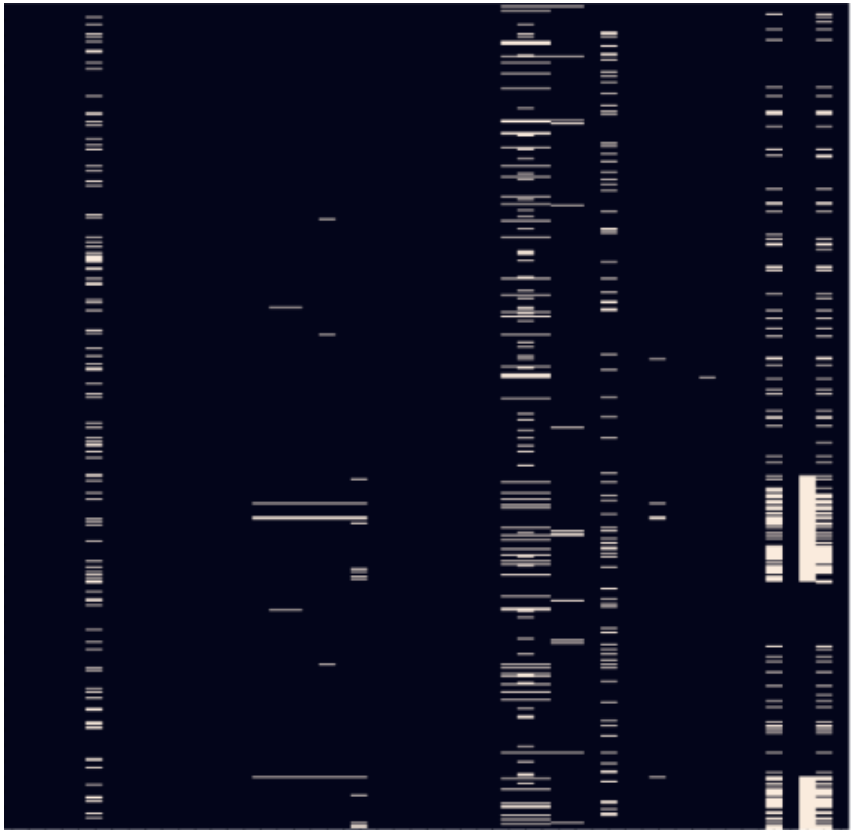
Visualization of the dataset using matplotlib & seaborn library

Vom crea 2 seturi de date ce ne vom ajuta sa vizualizam datele. Urmatorul set de date ne va ajuta sa vizualizam datele initiale. Anumite particularitatii din setul de date initial ne pot ajuta sa plotam rezultatele esentiale in functie de caz.

[81]
```
data_as_csv_1 = data_as_spark_1.toPandas()
```

Un prim lucru pe care il putem vizualiza folosind aceste 2 librari sunt rezultatele cu privire la modificari efectuate mai sus. Output-ul urmatoarei celule prezinta setului de date initial care continea o serie foarte mare de date lipsa.

[82]
```
plt.figure(figsize=(10, 10))
sns.heatmap(data_as_csv_1.isnull(), cbar=False)
```



Urmatorul output al celulei urmatoare ne prezinta rezultatele cu privire la completarea setului nostru de date

[83]
```
plt.figure(figsize=(10, 10))
sns.heatmap(data_as_csv.isnull(), cbar=False)
```

Descrierea sumativa a datelor

[84]
```
data_as_csv.describe().style.background_gradient(cmap = 'copper')
```

	Numar	Stare	Varsta	Sex	Indice_de_masa_corporala	Hipertensiv	Fibrilatie_atriala	Boala_coronariana	Diabet	Anemii_deficitare	Depresie
count	1175	1175	1175	1175	1175	1175	1175	1175	1175	1175	1175
unique	1175	2	68	2	933	2	2	2	2	2	2
top	125047	alive	89	Male	30.18827765159043	da	nu	nu	nu	nu	nu
freq	1	1017	140	617	214	844	645	1074	681	776	1035

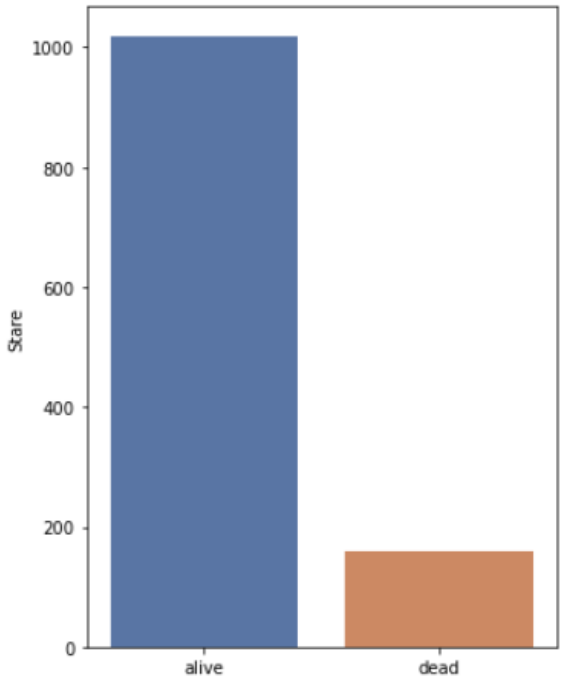
In urma modificariilor pe care le-am efectua asupra setului de date ales de mine am observat faptul ca anumite coloane sunt mai semnificative decat altele pentru analiza noastra. Aceste coloane sunt fie etichetate intr-un anumit fel, fie contin valori numerice. Conform scopului urmarit de aceasta analiza, vom considera a fi semnificative urmatoarele coloane pe care le voi grupa mai jos in functie de natura acestora.

- Coloane categorice
  - Stare
  - Sex
  - Diabet
  - Depresie
- Coloane numerice
  - Indice_de_masa_corporala
  - Frecventa_cardiaca
  - Frecventa_respiratorie
  - Temperatura
  - Tensiunea_arteriala_sistolica / Tensiunea_arteriala_diastolica
  - Numarul de globule albe si rosii
  - Cantitatea de leucocite,trombocite,neurofile, limfocite
  - Cantitatea de Glucoza, Potasiu, Sodiu, Calciu, Clorura, Magneziu

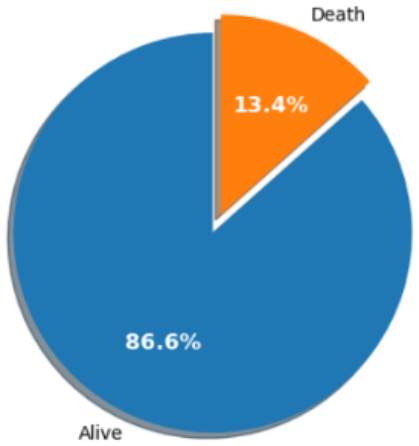
In continuare voi prezenta o serie de grafice unde se poate observa detalii cu privire la variabilelor mentionate mai sus.

Distributia coloanei Stare

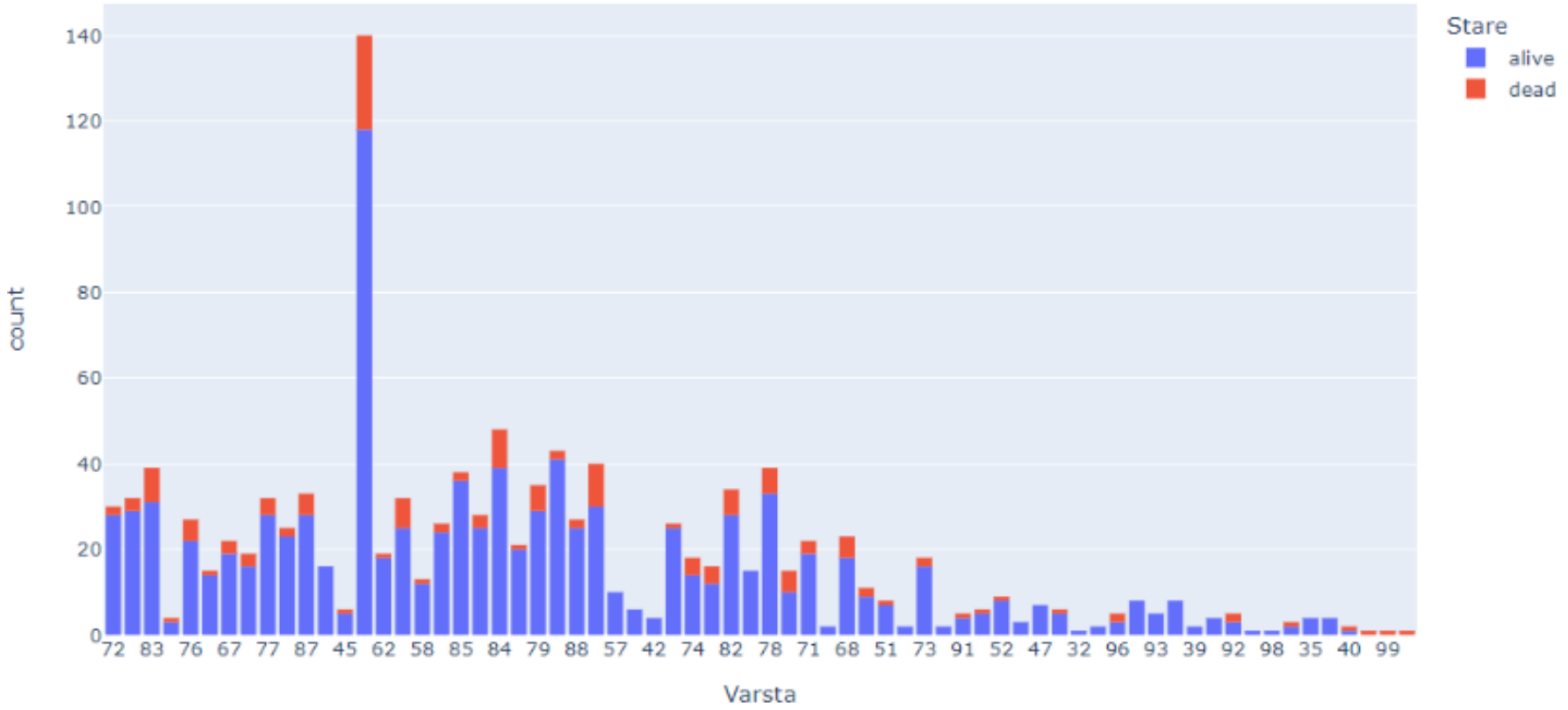
[85]
```
plt.figure(figsize=(5,7))
data=data_as_csv['Stare'].value_counts()[0:2]
sns.barplot(x=data.index,y=data,palette='deep')
```



[86]
```
fig, ax = plt.subplots(figsize=(5,5), dpi=100)
patches, texts, autotexts = ax.pie(data_as_csv_1['outcome'].value_counts(), autopct= '%1.1f%%',
shadow=True, startangle=90, explode=(0.1, 0), labels=['Alive','Death'])
plt.setp(autotexts, size=12, color = 'white', weight='bold')
autotexts[1].set_color('white');
plt.title('Distribuția rezultatelor', fontsize=10)
plt.show()
```



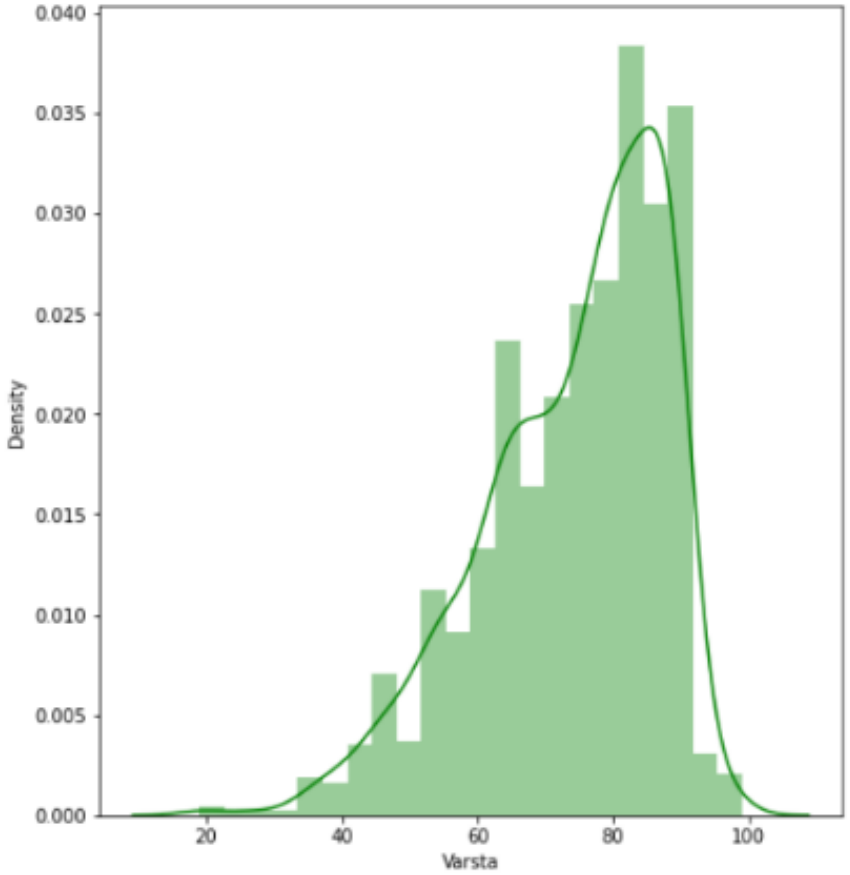
[87]
```
fig = px.histogram(data_as_csv, x="Varsta",color="Stare", hover_data=data_as_csv.columns)
fig.show()
```



Distributia coloanei Varsta

[88]

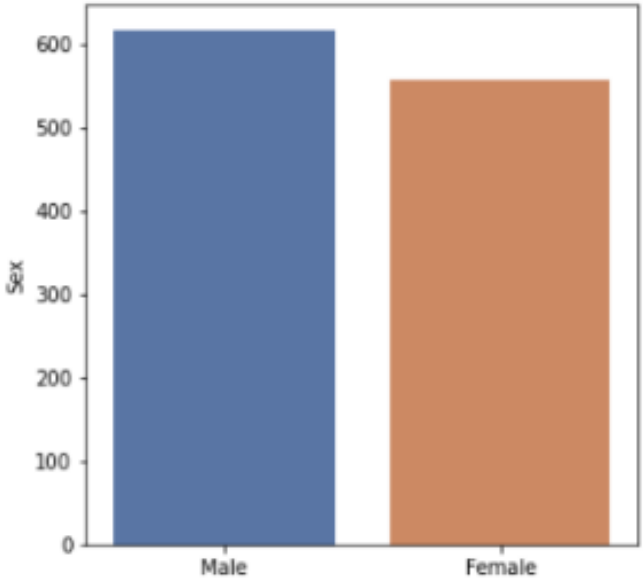
```
plt.figure(figsize = (30, 20))
plotnumber = 1
cat_cols = ['Varsta']
for column in cat_cols:
    if plotnumber <= 2:
        ax = plt.subplot(3, 5, plotnumber)
        sns.distplot(data_as_csv[column],color='green')
        plt.xlabel(column)
    plotnumber += 1
plt.tight_layout()
plt.show()
```



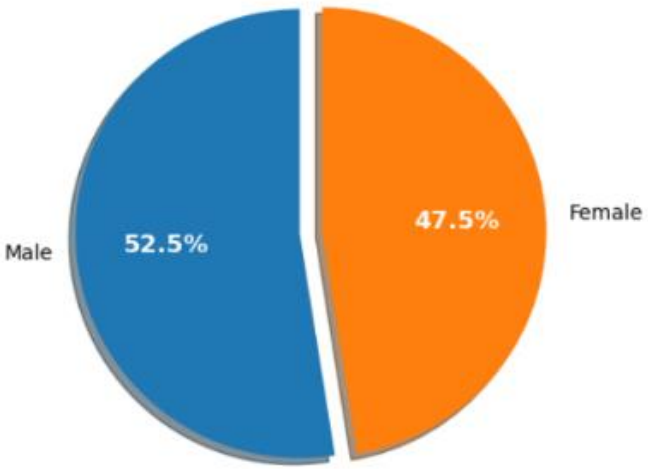
Distributia coloanei Sex

[89]

```
plt.figure(figsize=(5,5))
data=data_as_csv['Sex'].value_counts()[0:2]
sns.barplot(x=data.index,y=data,palette='deep')
```

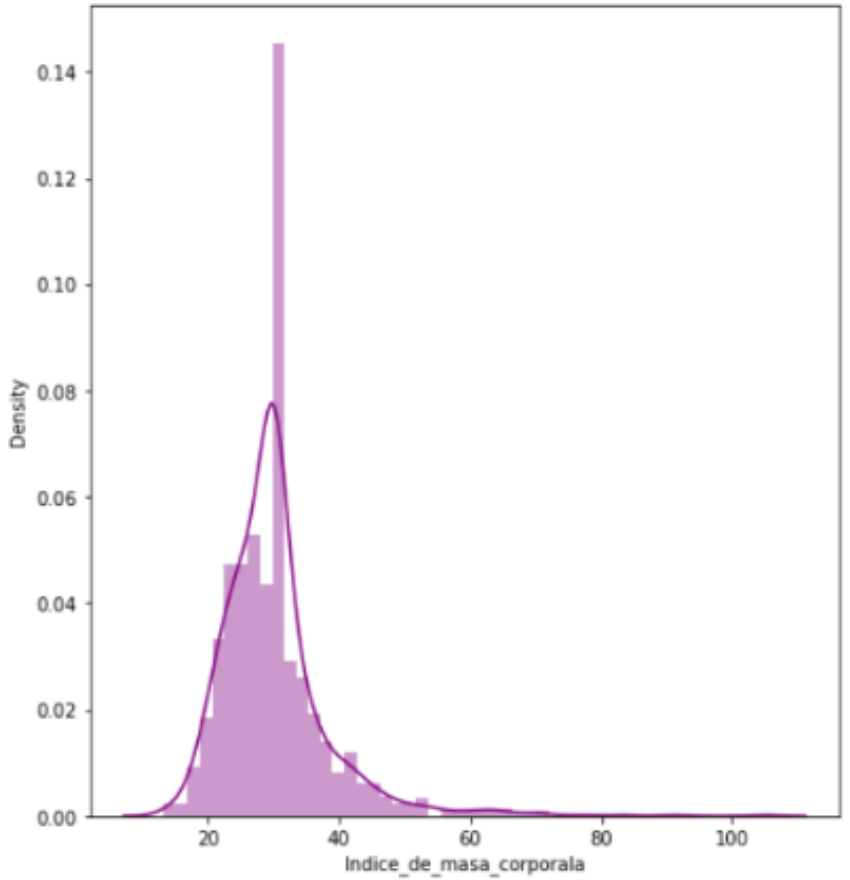


[90]
```
fig, ax = plt.subplots(figsize=(5,5), dpi=100)
patches, texts, autotexts = ax.pie(data_as_csv_1['gender'].value_counts(), autopct=
'%1.1f%%', shadow=True, startangle=90, explode=(0.1, 0), labels=['Male','Female'])
plt.setp(autotexts, size=12, color = 'white', weight='bold')
autotexts[1].set_color('white');
plt.title('Distribuția genurilor', fontsize=10)
plt.show()
```



Distributia coloanei Indice_de_masa_corporala

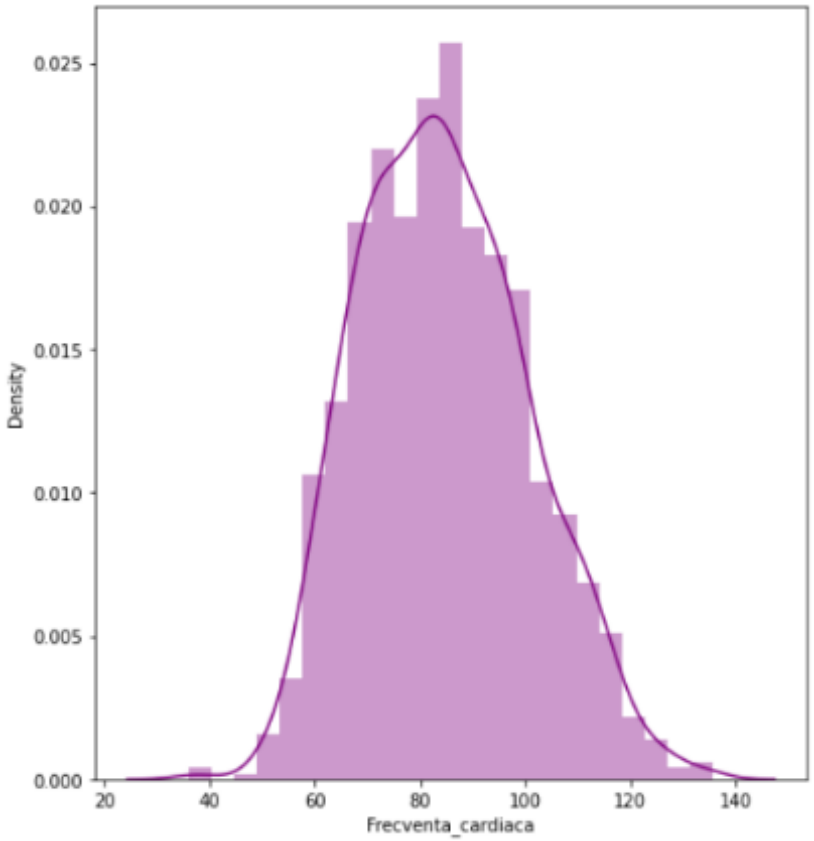
[91]
```
plt.figure(figsize = (30, 20))
plotnumber = 1
cat_cols = ['Indice_de_masa_corporala']
for column in cat_cols:
    if plotnumber <= 2:
        ax = plt.subplot(3, 5, plotnumber)
        sns.distplot(data_as_csv[column],color='purple')
        plt.xlabel(column)
    plotnumber += 1
plt.tight_layout()
plt.show()
```



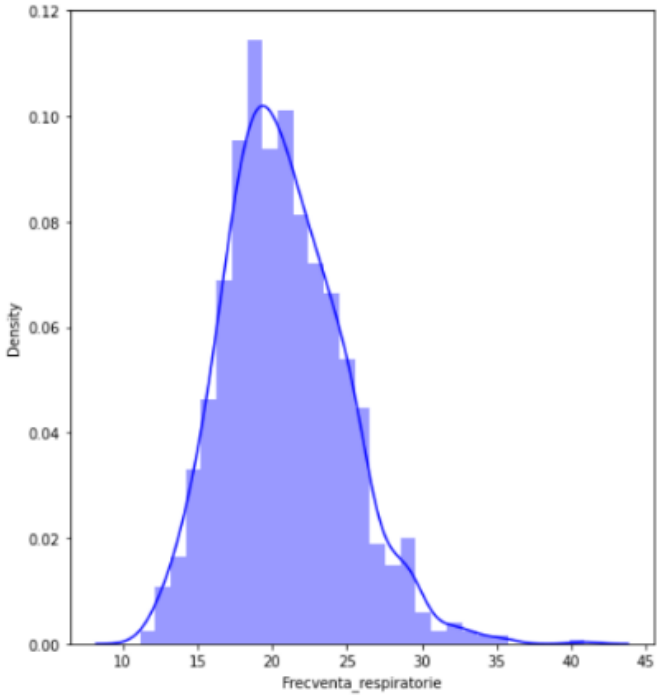
Distributia coloanei Frecventa_cardiaca

[96]

```
plt.figure(figsize = (30, 20))
plotnumber = 1
cat_cols = ['Frecventa_cardiaca']
for column in cat_cols:
    if plotnumber <= 2:
        ax = plt.subplot(3, 5, plotnumber)
        sns.distplot(data_as_csv[column],color='purple')
        plt.xlabel(column)
    plotnumber += 1
plt.tight_layout()
plt.show()
```



Distributia coloanei Frecventa_respiratorie



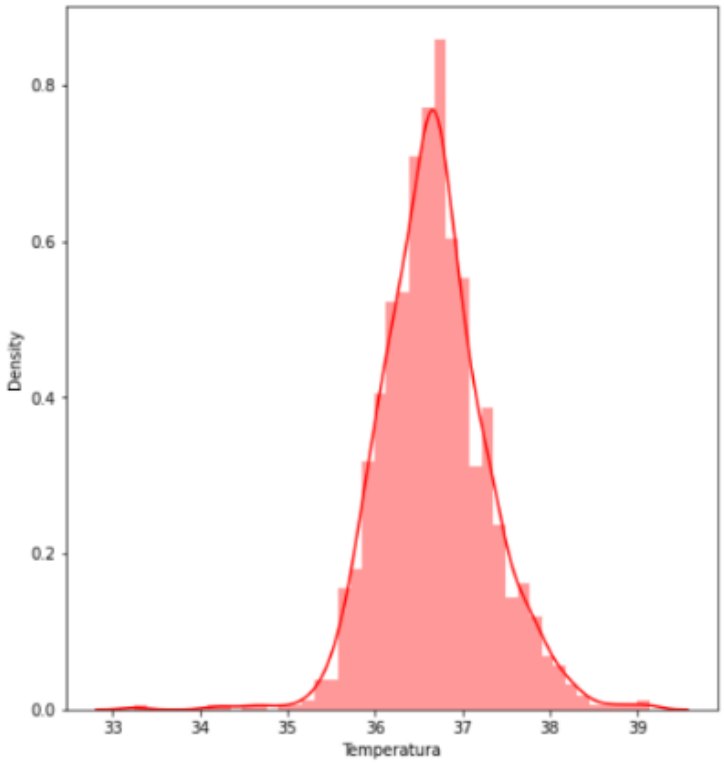
[97]

```
plt.figure(figsize = (30, 20))
plotnumber = 1
cat_cols = ['Frecventa_respiratorie']
for column in cat_cols:
    if plotnumber <= 2:
        ax = plt.subplot(3, 5, plotnumber)
        sns.distplot(data_as_csv[column],color='blue')
        plt.xlabel(column)
    plotnumber += 1
plt.tight_layout()
plt.show()
```

Distributia coloanei Temperatura

[98]

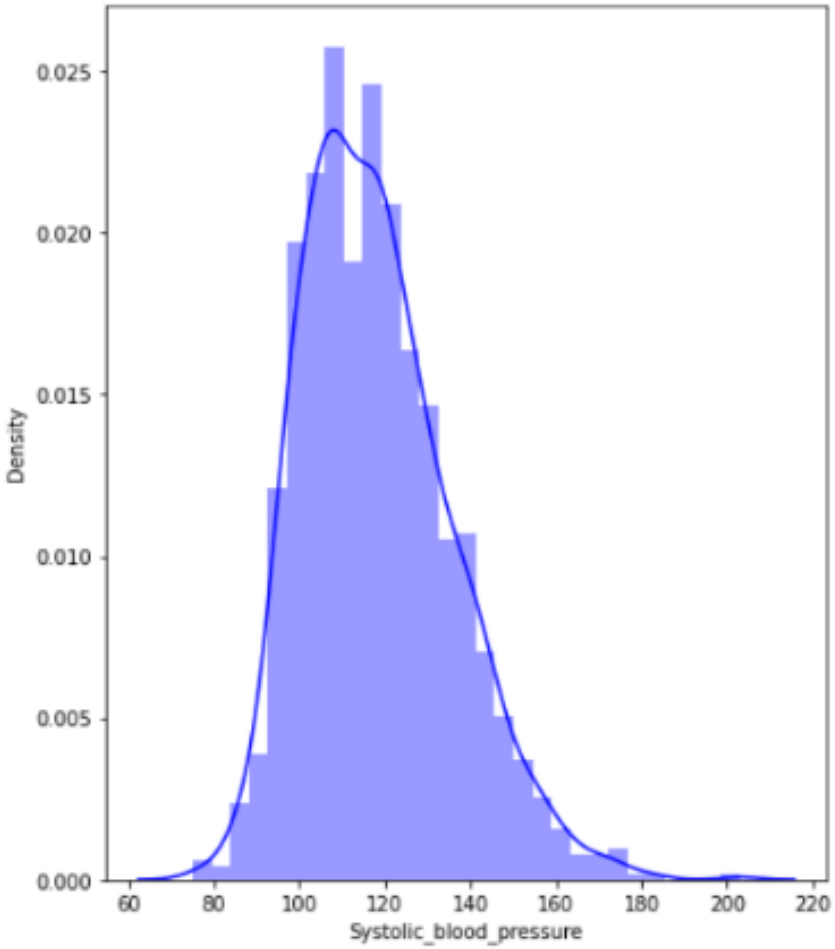
```
plt.figure(figsize = (30, 20))
plotnumber = 1
cat_cols = ['Temperatura']
for column in cat_cols:
    if plotnumber <= 2:
        ax = plt.subplot(3, 5, plotnumber)

sns.distplot(data_as_csv[column],color='red')
        plt.xlabel(column)
    plotnumber += 1
plt.tight_layout()
```

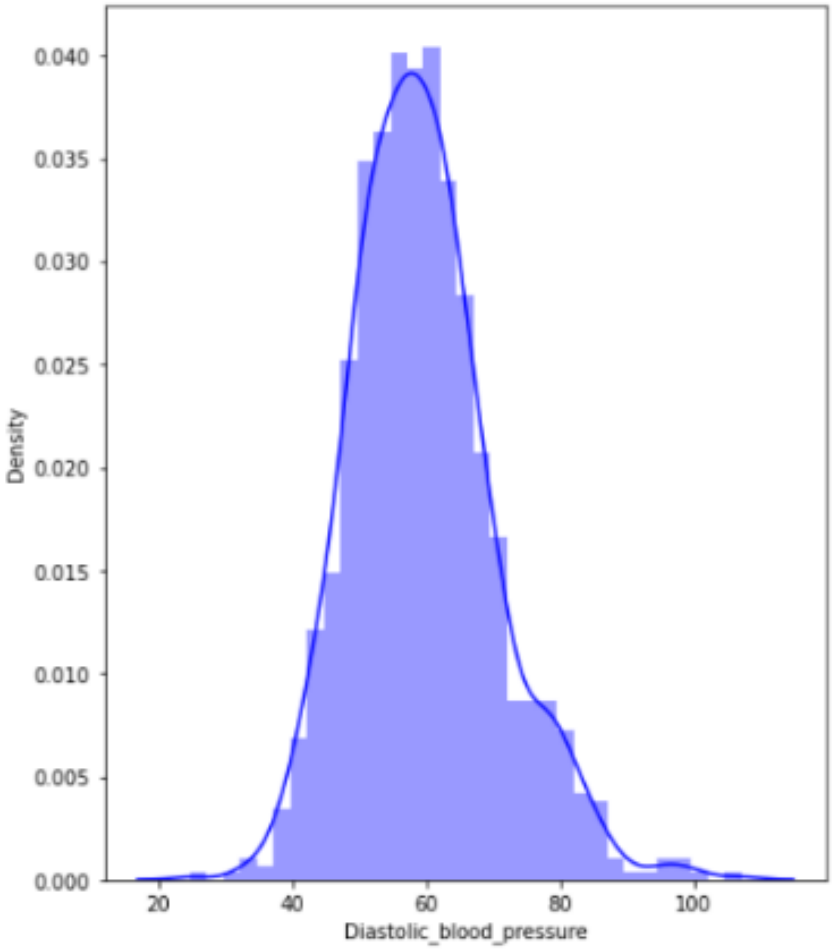


Distributia coloanei Tensiune_arteriala si Tensiune_arteriala_diastolica

[99]

```
plt.figure(figsize = (30, 20))
plotnumber = 1
cat_cols = ['Systolic_blood_pressure','Diastolic_blood_pressure']
for column in cat_cols:
    if plotnumber <= 2:
        ax = plt.subplot(3, 5, plotnumber)
        sns.distplot(data_as_csv_1[column],color='blue')
        plt.xlabel(column)
    plotnumber += 1
plt.tight_layout()
plt.show()
```





Luand in considerare ca acestea au fost cateva dintre cele mai importante variabile nu voi mai adauga celulele in urma carora se vor afisa distributiile variabilelor ramase. Acestea se pot gasi in notebook-ul atasat documentului.

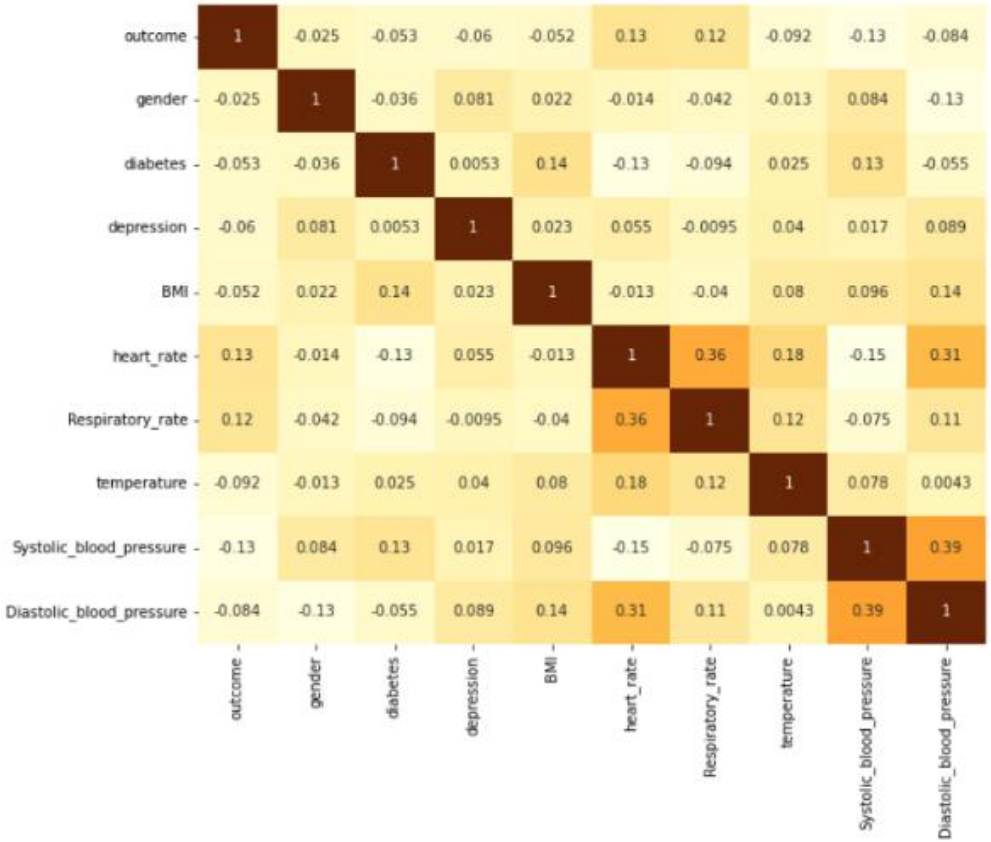
In continuare, voi prezenta o serie de ploturi ce descriu corelatia dintre variabile.

Corelatia dintre variabile

[106]

```
from sklearn.impute import SimpleImputer
dataframe = pd.read_csv('original_data_.csv')
label_x = dataframe.drop(columns='outcome')
label_y = dataframe[['outcome']]
simpleImp = SimpleImputer(missing_values=np.nan,
strategy='mean')
column = label_x.select_dtypes(include='float64').columns
simpleImp.fit(label_x[column])
label_x[column] = simpleImp.transform(label_x[column])
SimpleImp =  SimpleImputer(missing_values=np.nan,
strategy="most_frequent")
SimpleImp.fit(label_y)
label_y = SimpleImp.transform(label_y)
label_y = pd.DataFrame(label_y, columns=['outcome'],
dtype='int64')
df = label_x.copy()
df['outcome'] = label_y
df.groupby(by='group').describe().round().T
col = ['outcome', 'gender', 'diabetes','depression', 'BMI',
'heart_rate',
```

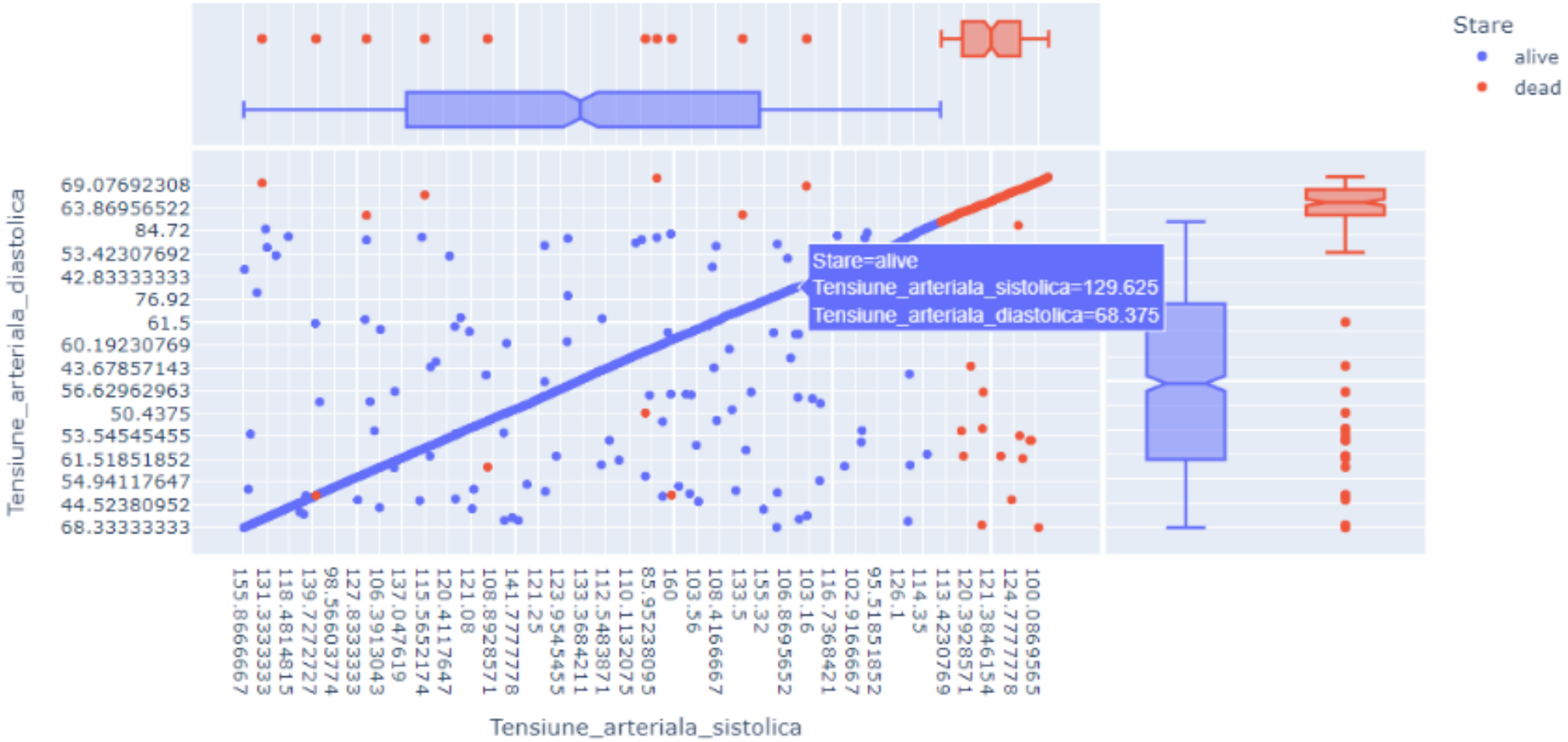
Dintre toate variabilele, putem observa ca exista cateva care sunt puternic corelate. Mai jos, vom putea vizualiza legaturile dintre variabilele din setul de date ales de mine. Mai jos as dori sa enumar primele *5* cele mai corelate perechi de variabile pentru a putea vizualiza mult mai bine legatura dintre acestea.



Tensiune_arteriala_diastolica - Indice_de_masa_corporala (0.14)
Temperatura - Frecventa_cardiaca (0.18)
Tensiune_arteriala_diastolica - Frecventa_cardiaca (0.31)
Frecventa_cardiaca - Frecventa_respiratorie (0.36)
Tensiune_arteriala_sistolica - Tensiune_arteriala_diastolica (0.39)

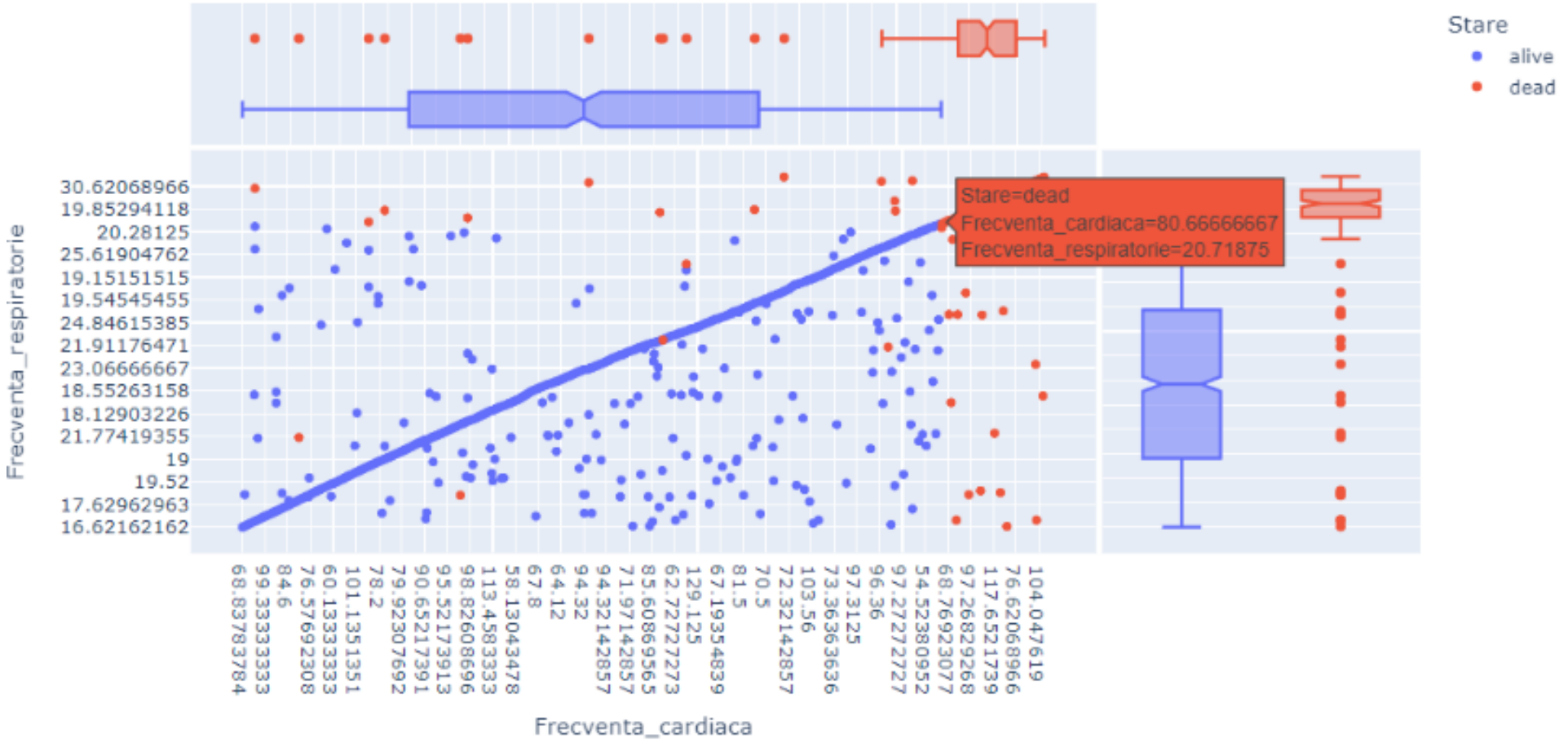
Tensiunea_arteriala_sistolica si Tensiunea_arteriala_diastolica

[107]
```
px.scatter(data_as_csv, x="Tensiune_arteriala_sistolica",
y="Tensiune_arteriala_diastolica",color="Stare",  marginal_y="box", marginal_x="box")
```



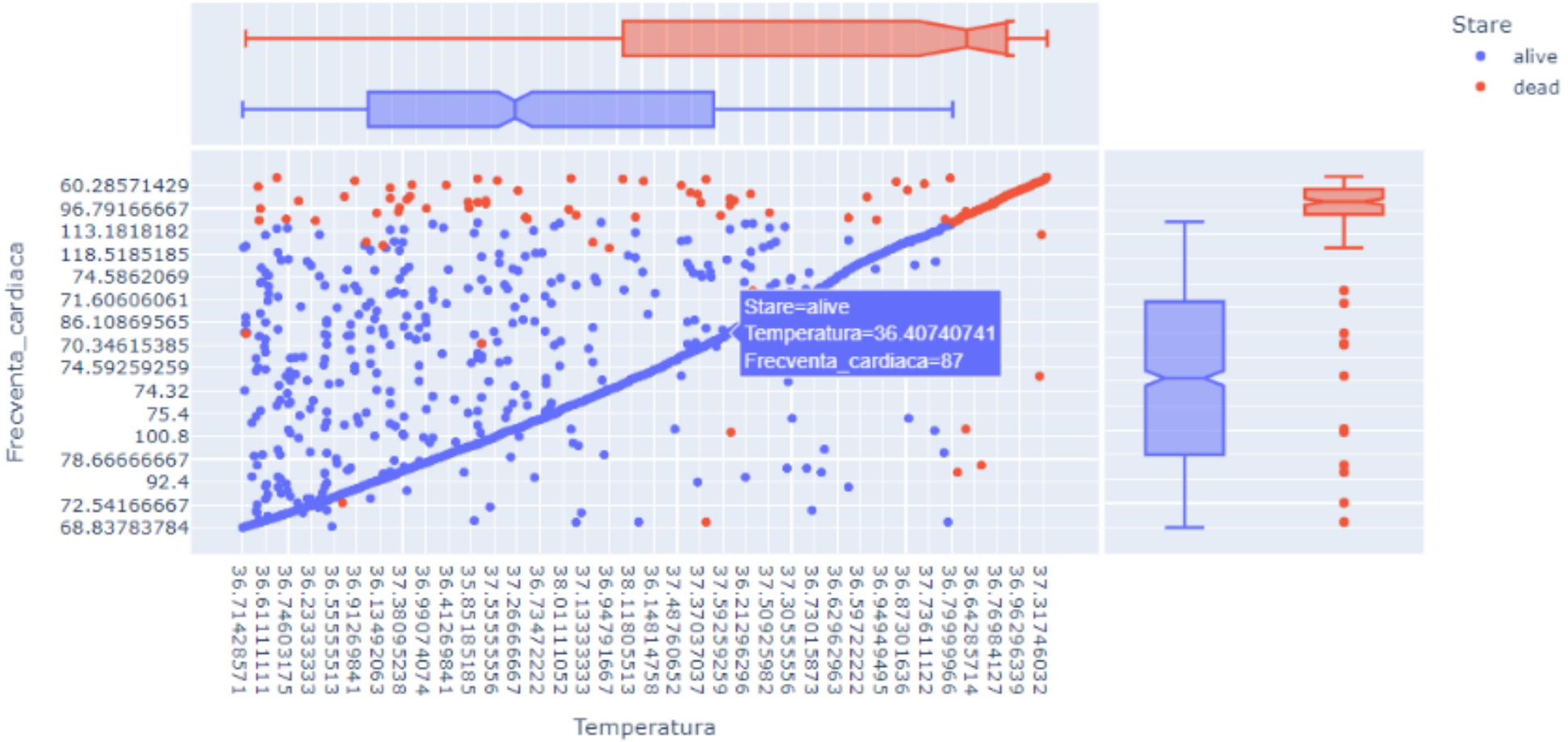
Frecventa_cardiaca si Frecventa_respiratorie

[108]
```
px.scatter(data_as_csv, x="Frecventa_cardiaca",
y="Frecventa_respiratorie",color="Stare",  marginal_y="box",
marginal_x="box")
```



Temperatura si Frecventa_cardiaca

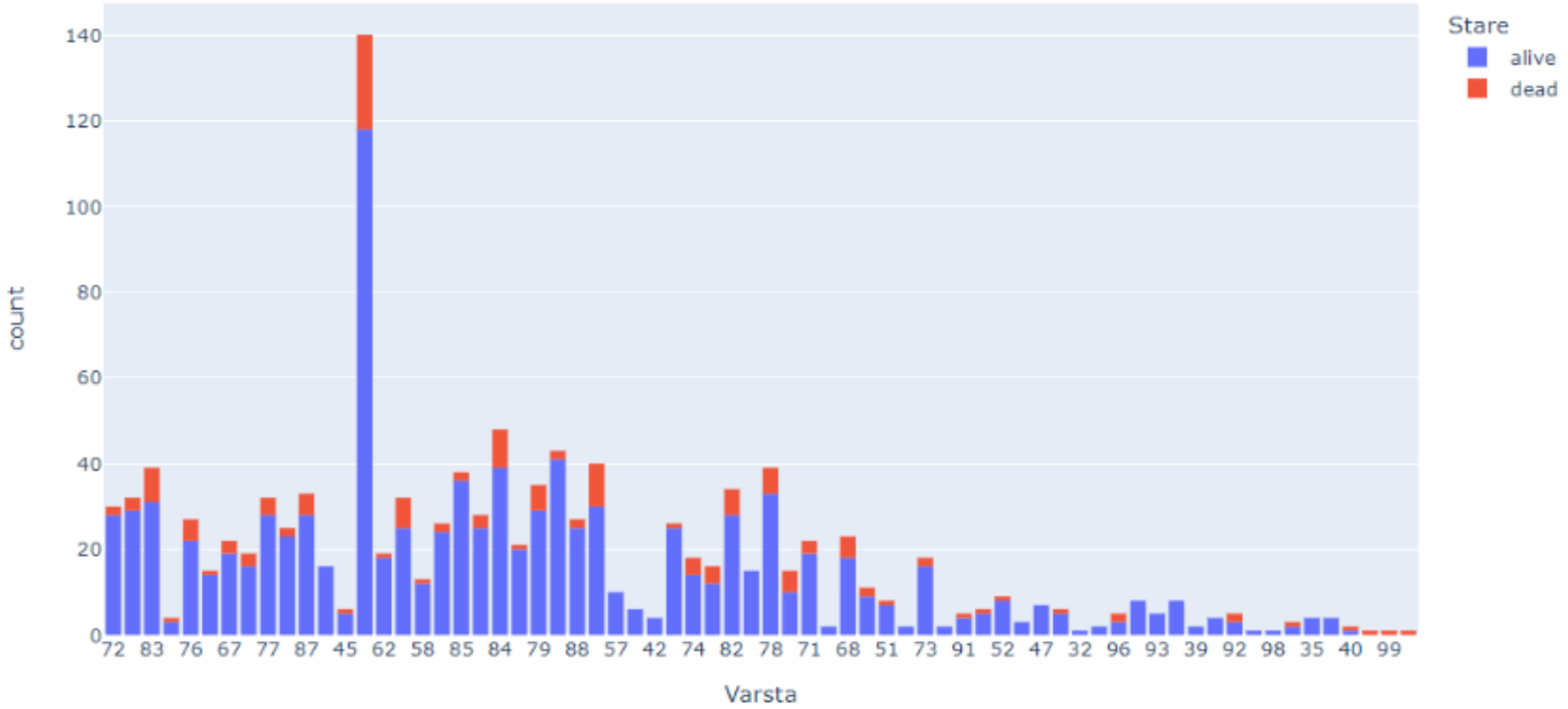
[110]
```
px.scatter(data_as_csv, x="Temperatura",
y="Frecventa_cardiaca",color="Stare",  marginal_y="box", marginal_x="box")
```



Vizualizarea variabilelor necorelate si impactul acestora asupra rezultatului final

Distributia pacientilor in functie de starea de sanatate si varsta

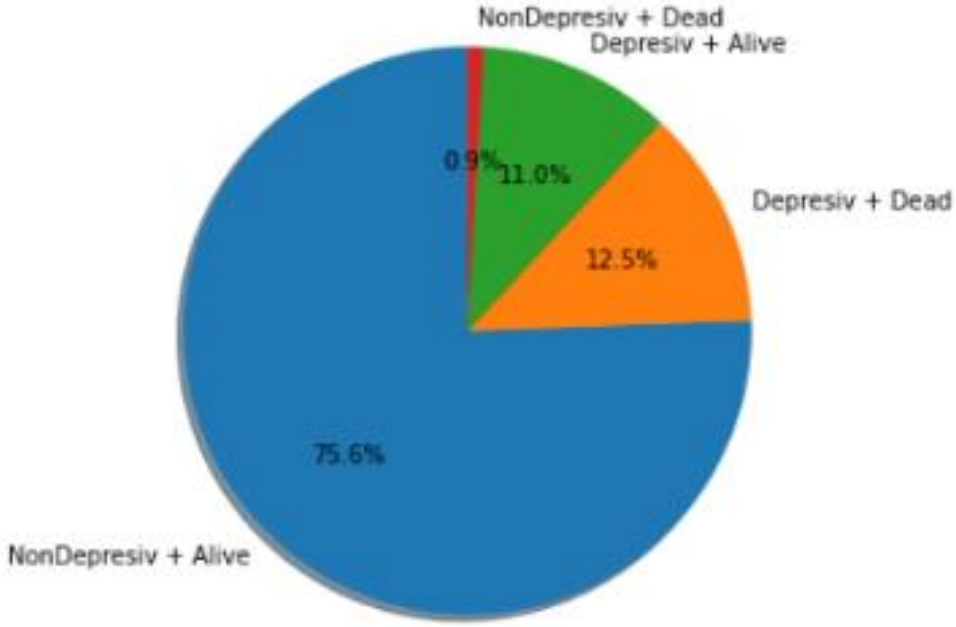
[112]
```
fig = px.histogram(data_as_csv, x="Varsta",color="Stare", hover_data=data_as_csv.columns)
fig.show()
```



Impactul depresiei asupra starii pacientilor

[113]

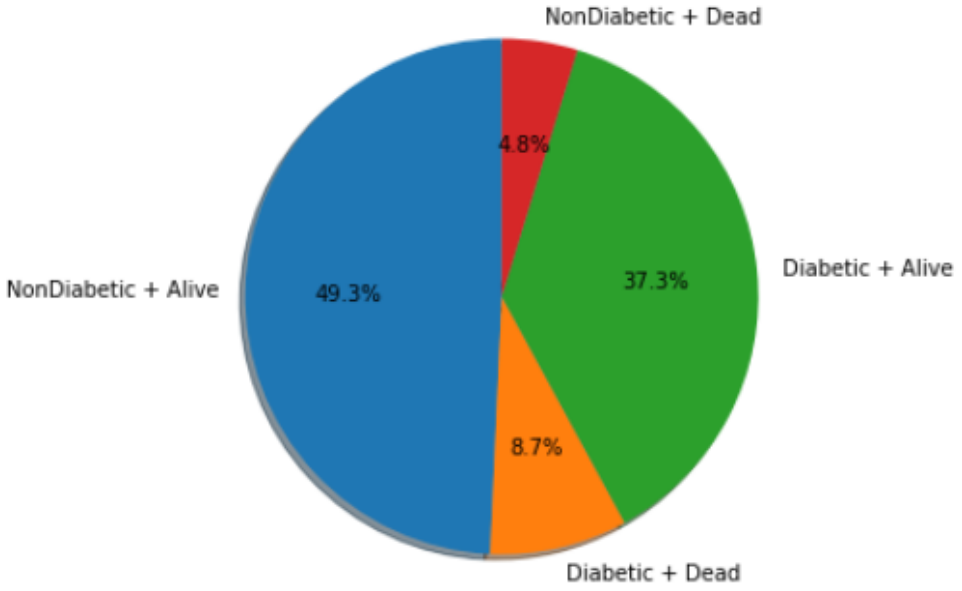
```
df = dataframe
fig = plt.figure(figsize=(12,7))
#ax1 = fig.add_axes([0,0,1,1])
ax2 = fig.add_axes([0.1,0.4, 0.4,0.6])
lab = ['NonDepresiv + Alive', 'Depresiv + Dead', 'Depresiv + Alive',
'NonDepresiv + Dead']
#ax1.bar(df['hypertensive'].unique(),df['hypertensive'].value_count
s(sort=False), width=0.5)
#ax1.set_xticks([0,1])

ax2.pie(df.groupby(by=['depression', 'outcome']).outcome.count(),
autopct= '%1.1f%%', shadow=True,startangle=90, labels=lab)

plt.title('Impactul depresiei asupra starii pacientilor')
plt.show()
```



Impactul diabetului asupra starii pacientilor

[114]

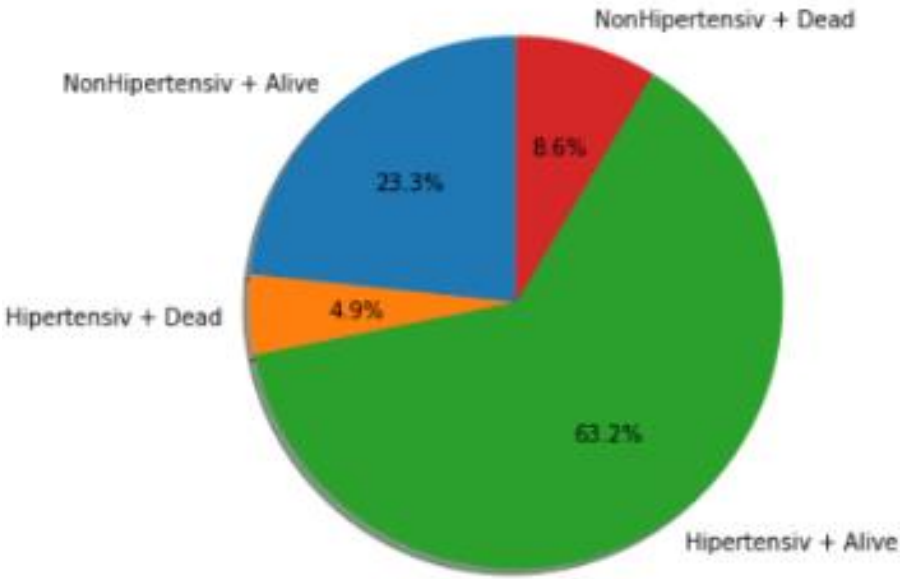
```
df = dataframe
fig = plt.figure(figsize=(12,7))
#ax1 = fig.add_axes([0,0,1,1])
ax2 = fig.add_axes([0.1,0.4, 0.4,0.6])
lab = ['NonDiabetic + Alive', 'Diabetic + Dead', 'Diabetic +
Alive', 'NonDiabetic + Dead']
#ax1.bar(df['hypertensive'].unique(),df['hypertensive'].val
ue_counts(sort=False), width=0.5)
#ax1.set_xticks([0,1])

ax2.pie(df.groupby(by=['diabetes',
'outcome']).outcome.count(), autopct= '%1.1f%%',
shadow=True,startangle=90, labels=lab)

plt.title('Impactul diabetului asupra starii pacientilor')
plt.show()
```



Impactul hipertensivitatii asupra starii pacientilor

[116]

```
df = dataframe
fig = plt.figure(figsize=(12,7))
#ax1 = fig.add_axes([0,0,1,1])
ax2 = fig.add_axes([0.1,0.4, 0.4,0.6])
lab = ['NonHipertensiv + Alive', 'Hipertensiv + Dead',
'Hipertensiv + Alive', 'NonHipertensiv + Dead']
#ax1.bar(df['hypertensive'].unique(),df['hypertensive'].val
ue_counts(sort=False), width=0.5)
#ax1.set_xticks([0,1])

ax2.pie(df.groupby(by=['hypertensive','outcome']).outcom
e.count(), autopct= '%1.1f%%', shadow=True,startangle=90,
labels=lab)

plt.title('Impactul hipertensivitatii asupra starii
pacientilor')
plt.show()
```



3. Metode Machine Learning

Inainte de a incepe sa construim modele pe care le vom folosi pentru a ne oferi o solutie la problema noastra mai avem o serie de pasi pe care trebuie sa ii parcurgem. Inainte de a incepe sa analizam modelele trebuie sa observam ca setul nostru de date se imparte in 2 seturi de date in functie de natura valorilor si anume categorice si numerice.

Pentru a rezolva problema propusa si anume de a prezice valoarea tinta Stare, voi folosi o serie de modele care sa fie potrivite tipurilor mele de date dar si rezultatelor pe care le caut

Setul de date ale de mine poate fi descris in felul urmator din punctul de vedere al tipurilor de date.

DATA = CATEGORICAL DATA + NUMERIC DATA

In urmatorul tabel voi scrie care au fost modele ales de mine si predictia acestora in functie de rezultatele obtinute dar si de totalitatea datelor folosite

DATA / MODEL	CATEGORICAL DATA	NUMERIC DATA
Naïve Bayes Model	0,847082	
Linear SVM Model	0,866451	
Logistic Regression Model	0,877450	
Artificial Neural Network Model	1	

Modelul Naïve Bayes

Am ales sa folosesc acest model pentru foarte scalabil, necesitând un număr de parametri liniari in functie de numărul de variabile (features/predictors). Antrenamentul se realizeaza prin intermediul probabilitatii maxime care necesită timp linear si care este mai eficienta decât aproximarea iterativă care este folosită in alte tipuri de clasificatoare.

Transformarea setului de date pentru a se potrivi modelului

[117]

```
# Salvam datele intr-o alta variabila
naive_bayes_data =
data_as_spark.select("Numar","Sex","Hipertensiv","Fibrilatie_atriala","Boala_coronariana","Diabet","Anemii_deficitare","Depresie",
"Hiperlipemie","Insuficienta_renala","Boala_pulmonara_obstructiva_cronica","Stare")
# In cazul in care dorim sa micsoram numarul de valori introduse
#naive_bayes_data = naive_bayes_data.limit(300)

#1 Modificarea coloanei Stare
naive_bayes_data = naive_bayes_data.withColumn("Stare",when(naive_bayes_data.Stare ==
"alive","yes").when(naive_bayes_data.Stare == "dead","no").otherwise("null"))
#2 Modificarea coloanei Sex
naive_bayes_data = naive_bayes_data.withColumn("Sex",when(naive_bayes_data.Sex ==
"Female","female").when(naive_bayes_data.Sex == "Male","male").otherwise("null"))
#3 Modificarea coloanei Hipertensiv
naive_bayes_data = naive_bayes_data.withColumn("Hipertensiv",when(naive_bayes_data.Hipertensiv ==
"nu","x_hipertensiv").when(naive_bayes_data.Hipertensiv == "da","hipertensiv").otherwise("null"))
#4 Modificarea coloanei Fibrilatie_atriala
naive_bayes_data = naive_bayes_data.withColumn("Fibrilatie_atriala",when(naive_bayes_data.Fibrilatie_atriala ==
"nu","x_fibrilatie_atriala").when(naive_bayes_data.Fibrilatie_atriala == "da","fibrilatie_atriala").otherwise("null"))
#5 Modificarea coloanei Boala_coronariana
naive_bayes_data = naive_bayes_data.withColumn("Boala_coronariana",when(naive_bayes_data.Boala_coronariana ==
"nu","x_boala_coronariana").when(naive_bayes_data.Boala_coronariana == "da","boala_coronariana").otherwise("null"))
#6 Modificarea coloanei Diabet
naive_bayes_data = naive_bayes_data.withColumn("Diabet",when(naive_bayes_data.Diabet ==
"nu","x_diabet").when(naive_bayes_data.Diabet == "da","diabet").otherwise("null"))
#7 Modificarea coloanei Anemii_deficitare
naive_bayes_data = naive_bayes_data.withColumn("Anemii_deficitare",when(naive_bayes_data.Anemii_deficitare ==
"nu","x_anemii_deficitare").when(naive_bayes_data.Anemii_deficitare == "da","anemii_deficitare").otherwise("null"))
```

```
#8 Modificarea coloanei Depresie
naive_bayes_data = naive_bayes_data.withColumn("Depresie",when(naive_bayes_data.Depresie ==
"nu","x_depresie").when(naive_bayes_data.Depresie == "da","depresie").otherwise("null"))
#9 Modificarea coloanei Hiperlipemie
naive_bayes_data = naive_bayes_data.withColumn("Hiperlipemie",when(naive_bayes_data.Hiperlipemie ==
"nu","x_hiperlipemie").when(naive_bayes_data.Hiperlipemie == "da","hiperlipemie").otherwise("null"))
#10 Modificarea coloanei Insuficienta_renala
naive_bayes_data = naive_bayes_data.withColumn("Insuficienta_renala",when(naive_bayes_data.Insuficienta_renala ==
"nu","x_insuficienta_renala").when(naive_bayes_data.Insuficienta_renala == "da","insuficienta_renala").otherwise("null"))
#11 Modificarea coloanei Boala_pulmonara_obstructiva_cronica
naive_bayes_data =
naive_bayes_data.withColumn("Boala_pulmonara_obstructiva_cronica",when(naive_bayes_data.Boala_pulmonara_obstructiva_cro
nica == "nu","x_boala_pulmonara_obstructiva_cronica").when(naive_bayes_data.Boala_pulmonara_obstructiva_cronica ==
"da","boala_pulmonara_obstructiva_cronica").otherwise("null"))
# Afisarea datelor finale
naive_bayes_data.select("Numar","Stare","Sex","Hipertensiv","Fibrilatie_atriala","Boala_coronariana","Diabet").show()
naive_bayes_data.select("Anemii_deficitare","Depresie","Hiperlipemie","Insuficienta_renala","Boala_pulmonara_obstructiva_cronic
a","Stare").show()
naive_bayes_data.count()
```

Numar	Stare	Sex	Hipertensiv	Fibrilatie_atriala	Boala_coronariana	Diabet
1	yes	female	x_hipertensiv	x_fibrilatie_atriala	x_boala_coronariana	diabet
2	yes	male	x_hipertensiv	x_fibrilatie_atriala	x_boala_coronariana	x_diabet
3	yes	male	x_hipertensiv	x_fibrilatie_atriala	x_boala_coronariana	x_diabet
4	yes	male	x_hipertensiv	x_fibrilatie_atriala	x_boala_coronariana	x_diabet
5	yes	male	hipertensiv	x_fibrilatie_atriala	x_boala_coronariana	x_diabet
6	yes	female	hipertensiv	fibrilatie_atriala	x_boala_coronariana	x_diabet
7	yes	female	hipertensiv	x_fibrilatie_atriala	x_boala_coronariana	x_diabet
8	yes	male	hipertensiv	fibrilatie_atriala	x_boala_coronariana	diabet
9	yes	male	hipertensiv	fibrilatie_atriala	x_boala_coronariana	diabet
10	yes	female	hipertensiv	x_fibrilatie_atriala	x_boala_coronariana	diabet
11	yes	male	hipertensiv	x_fibrilatie_atriala	x_boala_coronariana	diabet
12	yes	male	hipertensiv	x_fibrilatie_atriala	x_boala_coronariana	diabet
13	yes	male	hipertensiv	x_fibrilatie_atriala	x_boala_coronariana	x_diabet
14	yes	female	hipertensiv	fibrilatie_atriala	x_boala_coronariana	diabet
15	yes	male	hipertensiv	x_fibrilatie_atriala	x_boala_coronariana	x_diabet
16	yes	male	hipertensiv	x_fibrilatie_atriala	x_boala_coronariana	diabet
17	no	male	hipertensiv	x_fibrilatie_atriala	x_boala_coronariana	x_diabet
18	yes	male	hipertensiv	x_fibrilatie_atriala	x_boala_coronariana	diabet
19	yes	male	hipertensiv	x_fibrilatie_atriala	x_boala_coronariana	x_diabet
20	yes	male	x_hipertensiv	x_fibrilatie_atriala	x_boala_coronariana	x_diabet

Anemii_deficitare	Depresie	Hiperlipemie	Insuficienta_renala	Boala_pulmonara_obstructiva_cronica	Stare
anemii_deficitare	x_depresie	hiperlipemie	insuficienta_renala	x_boala_pulmonara...	yes
anemii_deficitare	x_depresie	x_hiperlipemie	x_insuficienta_re...	boala_pulmonara_o...	yes
anemii_deficitare	x_depresie	x_hiperlipemie	insuficienta_renala	x_boala_pulmonara...	yes
x_anemii_deficitare	x_depresie	x_hiperlipemie	x_insuficienta_re...	x_boala_pulmonara...	yes
anemii_deficitare	x_depresie	x_hiperlipemie	insuficienta_renala	boala_pulmonara_o...	yes
anemii_deficitare	x_depresie	hiperlipemie	insuficienta_renala	boala_pulmonara_o...	yes
x_anemii_deficitare	x_depresie	hiperlipemie	insuficienta_renala	boala_pulmonara_o...	yes
anemii_deficitare	x_depresie	x_hiperlipemie	x_insuficienta_re...	x_boala_pulmonara...	yes
x_anemii_deficitare	x_depresie	x_hiperlipemie	insuficienta_renala	x_boala_pulmonara...	yes
x_anemii_deficitare	x_depresie	x_hiperlipemie	x_insuficienta_re...	x_boala_pulmonara...	yes
x_anemii_deficitare	x_depresie	hiperlipemie	x_insuficienta_re...	x_boala_pulmonara...	yes
anemii_deficitare	x_depresie	hiperlipemie	x_insuficienta_re...	x_boala_pulmonara...	yes
x_anemii_deficitare	x_depresie	x_hiperlipemie	x_insuficienta_re...	boala_pulmonara_o...	yes
x_anemii_deficitare	x_depresie	hiperlipemie	insuficienta_renala	x_boala_pulmonara...	yes
x_anemii_deficitare	x_depresie	x_hiperlipemie	x_insuficienta_re...	x_boala_pulmonara...	yes
x_anemii_deficitare	x_depresie	hiperlipemie	insuficienta_renala	x_boala_pulmonara...	yes
anemii_deficitare	x_depresie	x_hiperlipemie	insuficienta_renala	x_boala_pulmonara...	no
anemii_deficitare	depresie	x_hiperlipemie	insuficienta_renala	x_boala_pulmonara...	yes
anemii_deficitare	x_depresie	x_hiperlipemie	x_insuficienta_re...	x_boala_pulmonara...	yes
anemii_deficitare	x_depresie	x_hiperlipemie	insuficienta_renala	x_boala_pulmonara...	yes

Import Library

[118]
```
from pyspark.ml import Pipeline
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.feature import StringIndexer
from pyspark.sql.functions import *
```

[119]
```
#Salvam datele transformate ca csv
naive_bayes_csv = naive_bayes_data.toPandas()
#Salvam datele local
naive_bayes_csv.to_csv('naive_bayes_data.csv',index=None,header=None)
```

[120]
```
pacients_data = pd.read_csv("naive_bayes_data.csv",header = None)
pacients_data.head()
```

	0	1	2	3	4	5	6	7	8	9
0	1	female	x_hipertensiv	x_fibrilatie_atriala	x_boala_coronariana	diabet	anemii_deficitare	x_depresie	hiperlipemie	insuficienta_renala
1	2	male	x_hipertensiv	x_fibrilatie_atriala	x_boala_coronariana	x_diabet	anemii_deficitare	x_depresie	x_hiperlipemie	x_insuficienta_renala
2	3	male	x_hipertensiv	x_fibrilatie_atriala	x_boala_coronariana	x_diabet	anemii_deficitare	x_depresie	x_hiperlipemie	insuficienta_renala
3	4	male	x_hipertensiv	x_fibrilatie_atriala	x_boala_coronariana	x_diabet	x_anemii_deficitare	x_depresie	x_hiperlipemie	x_insuficienta_renala
4	5	male	hipertensiv	x_fibrilatie_atriala	x_boala_coronariana	x_diabet	anemii_deficitare	x_depresie	x_hiperlipemie	insuficienta_renala

Create Pyspark Dataframe

[121]
```
#Create Dataframe
pacients_df = spark.createDataFrame(pacients_data)
pacients_df.columns
```

```
['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11']
```

Change the columns

[122]
```
pacients_df = pacients_df.select(col("0").alias("numar"),
                                 col("1").alias("sex"),
                                 col("2").alias("tensiune"),
                                 col("3").alias("fibrilatie_atriala"),
                                 col("4").alias("boala_coronariana"),
                                 col("5").alias("diabet"),
                                 col("6").alias("anemii_deficitare"),
                                 col("7").alias("depresie"),
                                 col("8").alias("hiperlipemie"),
                                 col("9").alias("insuficienta_renala"),
                                 col("10").alias("boala_pulmonara_obstructiva_cronica"),
                                 col("11").alias("stare"))
```

Add the index columns

[123]
```
indexers = [StringIndexer(inputCol="numar", outputCol = "numar_index"),
            StringIndexer(inputCol="sex", outputCol = "sex_index"),
            StringIndexer(inputCol="tensiune", outputCol = "tensiune_index"),
            StringIndexer(inputCol="fibrilatie_atriala", outputCol = "fibrilatie_atriala_index"),
            StringIndexer(inputCol="boala_coronariana", outputCol = "boala_coronariana_index"),
            StringIndexer(inputCol="diabet", outputCol = "diabet_index"),
            StringIndexer(inputCol="anemii_deficitare", outputCol = "anemii_deficitare_index"),
            StringIndexer(inputCol="depresie", outputCol = "depresie_index"),
            StringIndexer(inputCol="hiperlipemie", outputCol ="hiperlipemie_index"),
            StringIndexer(inputCol="insuficienta_renala", outputCol = "insuficienta_renala_index"),
            StringIndexer(inputCol="boala_pulmonara_obstructiva_cronica", outputCol
                          ="boala_pulmonara_obstructiva_cronica_index"),
            StringIndexer(inputCol="stare", outputCol = "label")]
```

Create the pipeline

[124]
```
pipeline = Pipeline(stages = indexers)
indexed_pacients_df = pipeline.fit(pacients_df).transform(pacients_df)
```

Visualize the results

[125]
```
indexed_pacients_df.show(5,False)
```

```
+-----+------+-------------+--------------------+-------------------+--------+--------------------+----------+--------------+
|numar|sex   |tensiune     |fibrilatie_atriala  |boala_coronariana  |diabet  |anemii_deficitare   |depresie  |hiperlipemie  |
+-----+------+-------------+--------------------+-------------------+--------+--------------------+----------+--------------+
|1    |female|x_hipertensiv|x_fibrilatie_atriala|x_boala_coronariana|diabet  |anemii_deficitare   |x_depresie|hiperlipemie  |
|2    |male  |x_hipertensiv|x_fibrilatie_atriala|x_boala_coronariana|x_diabet|anemii_deficitare   |x_depresie|x_hiperlipemie|
|3    |male  |x_hipertensiv|x_fibrilatie_atriala|x_boala_coronariana|x_diabet|anemii_deficitare   |x_depresie|x_hiperlipemie|
|4    |male  |x_hipertensiv|x_fibrilatie_atriala|x_boala_coronariana|x_diabet|x_anemii_deficitare|x_depresie|x_hiperlipemie|
|5    |male  |hipertensiv  |x_fibrilatie_atriala|x_boala_coronariana|x_diabet|anemii_deficitare   |x_depresie|x_hiperlipemie|
+-----+------+-------------+--------------------+-------------------+--------+--------------------+----------+--------------+
only showing top 5 rows
```

```
+--------------+-----------------+-------------------------+-----------------------------------------+-----+
|depresie_index|hiperlipemie_index|insuficienta_renala_index|boala_pulmonara_obstructiva_cronica_index|label|
+--------------+-----------------+-------------------------+-----------------------------------------+-----+
|0.0           |1.0              |1.0                      |0.0                                      |0.0  |
|0.0           |0.0              |0.0                      |1.0                                      |0.0  |
|0.0           |0.0              |1.0                      |0.0                                      |0.0  |
|0.0           |0.0              |0.0                      |0.0                                      |0.0  |
|0.0           |0.0              |1.0                      |1.0                                      |0.0  |
+--------------+-----------------+-------------------------+-----------------------------------------+-----+
```

Create the features vectors

[126]
```
vectorAssembler = VectorAssembler(inputCols =
["numar_index","sex_index","tensiune_index","fibrilatie_atriala_index","boala_coronariana_index","diabet_index","anemii_de
ficitare_index","depresie_index","hiperlipemie_index","insuficienta_renala_index","boala_pulmonara_obstructiva_cronica_in
dex"],outputCol = "features")
vindexed_pacients_df = vectorAssembler.transform(indexed_pacients_df)
```

Visualize the results

[127]
```
vindexed_pacients_df.show(5,False)
```

```
+-------------------------+-----------------------------------------+-----+-----------------------------------------+
|insuficienta_renala_index|boala_pulmonara_obstructiva_cronica_index|label|features                                 |
+-------------------------+-----------------------------------------+-----+-----------------------------------------+
|1.0                      |0.0                                      |0.0  |(11,[1,2,5,6,8,9],[1.0,1.0,1.0,1.0,1.0,1.0])|
|0.0                      |1.0                                      |0.0  |(11,[0,2,6,10],[288.0,1.0,1.0,1.0])      |
|1.0                      |0.0                                      |0.0  |(11,[0,2,6,9],[398.0,1.0,1.0,1.0])       |
|0.0                      |0.0                                      |0.0  |(11,[0,2],[509.0,1.0])                   |
|1.0                      |1.0                                      |0.0  |(11,[0,6,9,10],[620.0,1.0,1.0,1.0])      |
+-------------------------+-----------------------------------------+-----+-----------------------------------------+
```

Import the model

[128]
```
from pyspark.ml.classification import NaiveBayes
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
```

Divizarea train-test

[129]
```
splits = vindexed_pacients_df.randomSplit([0.6,0.4])
train_df = splits[0]
test_df = splits[1]
```

Crearea modelului si configurarea parametrilor sai

[130]
```
nb = NaiveBayes(smoothing=1.0,modelType="multinomial")
```

Antrenarea modelului

[131]
```
nbmodel = nb.fit(train_df)
```

Aplicarea modelului si afisarea primelor 5 linii

[132]
```
predictions_df = nbmodel.transform(test_df)
predictions_df.show(5)
```

```
+----------------------------------------+-----+--------------------+--------------------+--------------------+----------+
|boala_pulmonara_obstructiva_cronica_index|label|            features|       rawPrediction|         probability|prediction|
+----------------------------------------+-----+--------------------+--------------------+--------------------+----------+
|                                     0.0|  0.0|(11,[0,2,6,9],[39...|[-24.600809176179...|[0.89987776402336...|       0.0|
|                                     0.0|  0.0|(11,[0,5,6,8],[20...|[-23.107291559380...|[0.93058800221294...|       0.0|
|                                     0.0|  0.0|    (11,[0],[233.0])|[-1.3918074527539...|[0.83794419786539...|       0.0|
|                                     0.0|  0.0|(11,[0,1,7,8,9],[...|[-32.049632375989...|[0.92244492194847...|       0.0|
|                                     0.0|  0.0|(11,[0,1,2,6],[35...|[-24.239149826289...|[0.82190508267950...|       0.0|
+----------------------------------------+-----+--------------------+--------------------+--------------------+----------+
```

Calculul acuratetei pe dataset-ul de test

[133]
```
evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="accuracy")
nbaccuracy = evaluator.evaluate(predictions_df)
print("Test set accuracy = " + str(nbaccuracy))
```

```
Test set accuracy = 0.8888888888888888
```

Modelul Linear SVM

În cazul unei probleme de clasificare liniare, SVM-urile pot realiza eficient o clasificare neliniară folosind ceea ce se numește kernel, mapând implicit intrările lor în spații de caracteristici cu dimensiuni mari.

Import model

[134]
```
from pyspark.ml.classification import LinearSVC
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from sklearn.metrics import confusion_matrix
```

Visualize the features

[135]
```
va_df = vindexed_pacients_df.select(['features', 'label'])
va_df.show(3)
```

```
+--------------------+-----+
|            features|label|
+--------------------+-----+
|(11,[1,2,5,6,8,9]...|  0.0|
|(11,[0,2,6,10],[2...|  0.0|
|(11,[0,2,6,9],[39...|  0.0|
+--------------------+-----+
only showing top 3 rows
```

Divizarea train-test

[136]
```
(train, test) = va_df.randomSplit([0.9, 0.1])
```

Crearea modelului si configurarea parametrilor sai

[137]
```
lsvc = LinearSVC(labelCol="label", maxIter=50,regParam=0.1)
```

Antrenarea modelului

[138]
```
lsvcModel = lsvc.fit(train)
```

Afisarea coeficientilor

[139]
```
print("Coefficients: " + str(lsvcModel.coefficients))
print("Intercept: " + str(lsvcModel.intercept))
```

```
Coefficients: [-3.2161912691296665e-09,-5.886041889810479e-05,1.3432638807815618e-05,4.758202144397335e-05,1.5383848718496854e-
05,-0.0004296554464475632,-2.424787200714913e-05,0.0,-1.994947567246183e-06,-3.7954762669978905e-05,-0.09306299728842779]
Intercept: -1.0000358323594556
```

Predictions

[140]
```
pred = lsvcModel.transform(test)
pred.show(3)
```

```
+--------------------+-----+--------------------+----------+
|            features|label|       rawPrediction|prediction|
+--------------------+-----+--------------------+----------+
|    (11,[0],[267.0])|  0.0|[1.00003669108252...|       0.0|
|(11,[0,1],[240.0,...|  0.0|[1.00009546466425...|       0.0|
|(11,[0,1,2],[248....|  0.0|[1.00008205775498...|       0.0|
+--------------------+-----+--------------------+----------+
only showing top 3 rows
```

Acuratete

[141]
```
evaluator = MulticlassClassificationEvaluator(metricName="accuracy")
acc = evaluator.evaluate(pred)
print(" Accuracy: ", acc)
```

```
Accuracy:  0.8490566037735849
```

Matrix of confusion

[142]
```
y_pred=pred.select("prediction").collect()
y_orig=pred.select("label").collect()
cm = confusion_matrix(y_orig, y_pred)
print("Confusion Matrix:")
print(cm)
```

```
Confusion Matrix:
[[90  0]
 [16  0]]
```

Modelul regresie logistica

Am ales acest model pentru ca se poate descurca foarte bine sa prezica un rezultat binar, cum ar fi da sau nu, pe baza observațiilor anterioare ale unui set de date. Un model de regresie logistică prezice o variabilă de date dependentă analizând relația dintre una sau mai multe variabile independente existente.

Transformarea datelor pentru a se potrivi cu modelul

[143]
```
# Salvam datele intr-o alta variabila
log_regr_data=data_as_spark.select(data_as_spark.Varsta.cast("integer"),"Sex","Depresie",data_as_spark.Frecventa_cardiaca.cast("float"),data_as_spark.Tensiune_arteriala_sistolica.cast("float"),"Diabet",data_as_spark.Temperatura.cast("float"),"Hipertensiv",data_as_spark.Potasiu_din_sange.cast("float"),data_as_spark.Calciu_din_sange.cast("float"),data_as_spark.Frecventa_respiratorie.cast("float"),"Stare")
cols = log_regr_data.columns

#1 Modificarea coloanei Stare
log_regr_data=log_regr_data.withColumn("Stare",when(log_regr_data.Stare=="alive","yes").when(log_regr_data.Stare=="dead","no").otherwise("null"))

#2 Modificarea coloanei Sex
log_regr_data=log_regr_data.withColumn("Sex",when(log_regr_data.Sex=="Female","female").when(log_regr_data.Sex=="Male","male").otherwise("null"))

#3 Modificarea coloanei Depresie
log_regr_data = log_regr_data.withColumn("Depresie",when(log_regr_data.Depresie == "nu","no").when(log_regr_data.Depresie == "da","yes").otherwise("null"))
```

```
#4 Modificarea coloanei Diabet
log_regr_data = log_regr_data.withColumn("Diabet",when(log_regr_data.Diabet == "nu","no").when(log_regr_data.Diabet ==
"da","yes").otherwise("null"))
#5 Modificarea coloanei Hipertensiv
log_regr_data = log_regr_data.withColumn("Hipertensiv",when(log_regr_data.Hipertensiv ==
"nu","no").when(log_regr_data.Hipertensiv == "da","yes").otherwise("null"))

# Afisarea datelor finale
log_regr_data.select("Varsta","Sex","Depresie","Frecventa_cardiaca","Tensiune_arteriala_sistolica","Diabet").show()
log_regr_data.select("Temperatura","Hipertensiv","Potasiu_din_sange","Calciu_din_sange","Frecventa_respiratorie","Stare").show()
log_regr_data.count()
log_regr_data.printSchema()
```

```
+------+------+--------+------------------+----------------------------+------+
|Varsta|   Sex|Depresie|Frecventa_cardiaca|Tensiune_arteriala_sistolica|Diabet|
+------+------+--------+------------------+----------------------------+------+
|    72|female|      no|          68.83784|                   155.86667|   yes|
|    75|  male|      no|         101.37037|                       140.0|    no|
|    83|  male|      no|         72.318184|                   135.33333|    no|
|    43|  male|      no|              94.5|                       126.4|    no|
|    75|  male|      no|             67.92|                      156.56|    no|
|    76|female|      no|         74.181816|                       118.1|    no|
|    72|female|      no|          69.63636|                   106.565216|    no|
|    83|  male|      no|         84.666664|                    141.13043|   yes|
|    61|  male|      no|         91.916664|                    98.434784|   yes|
|    67|female|      no|         75.083336|                       122.0|   yes|
|    70|  male|      no|          95.62963|                    149.03572|   yes|
|    83|  male|      no|             65.16|                    103.26087|   yes|
|    77|  male|      no|         78.833336|                    126.90323|    no|
|    83|female|      no|          65.86957|                    112.14286|   yes|
|    69|  male|      no|          98.54412|                       107.36|    no|
|    87|  male|      no|             73.48|                    159.69565|   yes|
|    83|  male|      no|          83.69231|                    157.28947|    no|
|    56|  male|     yes|              64.6|                       113.28|   yes|
|    45|  male|      no|              82.0|                       162.24|    no|
|    89|  male|      no|         70.083336|                   112.416664|    no|
+------+------+--------+------------------+----------------------------+------+
only showing top 20 rows
```

Temperatura	Hipertensiv	Potasiu_din_sange	Calciu_din_sange	Frecventa_respiratorie	Stare
36.714287	no	4.8166666	7.4636364	16.621622	yes
36.68254	no	4.45	8.1625	20.851852	yes
36.453705	no	5.825	8.266666	23.64	yes
36.287037	no	4.386667	9.476923	21.857143	yes
36.761906	yes	4.7833333	8.733334	21.36	yes
35.266666	yes	4.075	8.466666	20.545454	yes
35.603176	yes	4.6066666	8.775	19.148148	yes
36.67361	yes	4.2375	9.171429	18.4	yes
37.103176	yes	4.7181816	9.44375	18.583334	yes
36.86111	yes	3.87	8.15	18.125	yes
37.555557	yes	4.409091	8.45	17.481482	yes
36.47778	yes	4.0333333	9.1	17.4	yes
36.416668	yes	3.88	8.233334	15.833333	yes
36.157406	yes	4.34	8.716666	25.434782	yes
36.50926	yes	3.9714286	9.22	34.69343	yes
36.933334	yes	4.6857142	9.02	20.692308	yes
36.922222	yes	4.7909093	8.888889	15.652174	no
36.694443	yes	4.6	7.7	16.071428	yes
36.851852	yes	5.3	8.814285	28.153847	yes
36.00794	no	4.0	8.566667	25.583334	yes

only showing top 20 rows

```
root
 |-- Varsta: integer (nullable = true)
 |-- Sex: string (nullable = false)
 |-- Depresie: string (nullable = false)
 |-- Frecventa_cardiaca: float (nullable = true)
 |-- Tensiune_arteriala_sistolica: float (nullable = true)
 |-- Diabet: string (nullable = false)
 |-- Temperatura: float (nullable = true)
 |-- Hipertensiv: string (nullable = false)
 |-- Potasiu_din_sange: float (nullable = true)
 |-- Calciu_din_sange: float (nullable = true)
 |-- Frecventa_respiratorie: float (nullable = true)
 |-- Stare: string (nullable = false)
```

Import library

[144]
```
from pyspark.ml.feature import OneHotEncoder,StringIndexer,VectorAssembler
from pyspark.ml import Pipeline
from pyspark.ml.evaluation import BinaryClassificationEvaluator,MulticlassClassificationEvaluator
```

Categorical columns

[145]
```
categoricalColumns = ['Sex','Depresie','Diabet','Hipertensiv']
stages = []
```

Indexing categorical columns

[146]
```
for categoricalCol in categoricalColumns:
    stringIndexer = StringIndexer(inputCol = categoricalCol, outputCol = categoricalCol + 'Index')
    encoder = OneHotEncoder(inputCols=[stringIndexer.getOutputCol()], outputCols=[categoricalCol + "classVec"])
    stages += [stringIndexer,encoder]
```

Indexing target columns

[147]
```
label_stringIdx = StringIndexer(inputCol = 'Stare', outputCol = 'label')
stages += [label_stringIdx]
```

Indexing numeric columns

[148]
```
numericCols = ['Varsta','Frecventa_cardiaca','Tensiune_arteriala_sistolica','Temperatura','Potasiu_din_sange','Calciu_din_sange','Frecventa_respiratorie']
assemblerInputs = [c + "classVec" for c in categoricalColumns] + numericCols
assembler = VectorAssembler(inputCols=assemblerInputs, outputCol="features")
stages += [assembler]
```

Create pipeline

[149]
```
pipeline = Pipeline(stages = stages)
pipelineModel = pipeline.fit(log_regr_data)
log_regr_data = pipelineModel.transform(log_regr_data)
selectedCols = ['label', 'features'] + cols
log_regr_data = log_regr_data.select(selectedCols)
```

Divizarea train-test

[150]
```
train, test = log_regr_data.randomSplit([0.7, 0.3])
print("Training Dataset Count: " + str(train.count()))
print("Test Dataset Count: " + str(test.count()))
```

```
Training Dataset Count: 809
Test Dataset Count: 366
```

Import model

[151]
```
from pyspark.ml.classification import LogisticRegression
```

Crearea modelului si configurarea parametrilor sai

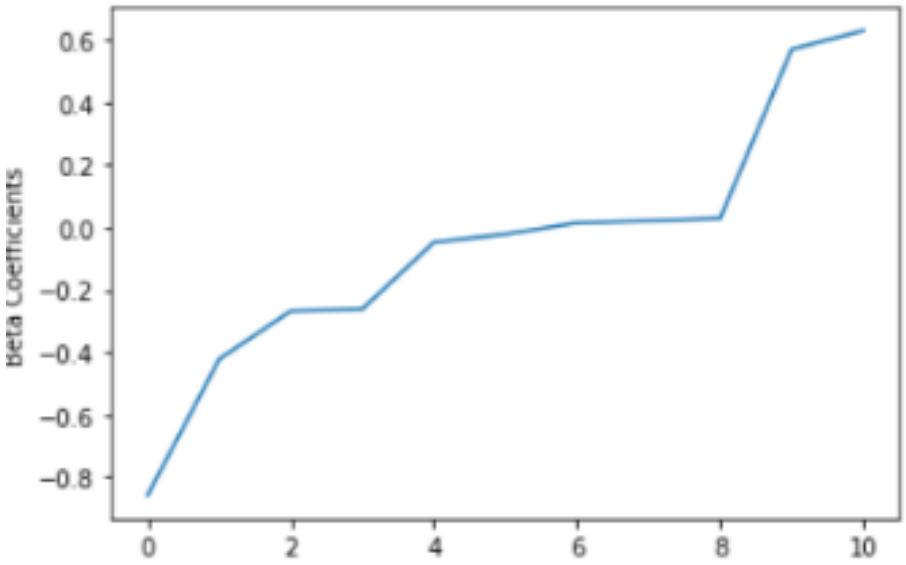
[152]
```
lr = LogisticRegression(featuresCol = 'features', labelCol = 'label', maxIter=10)
```

Antrenarea modelului

[153]
```
lrModel = lr.fit(train)
```

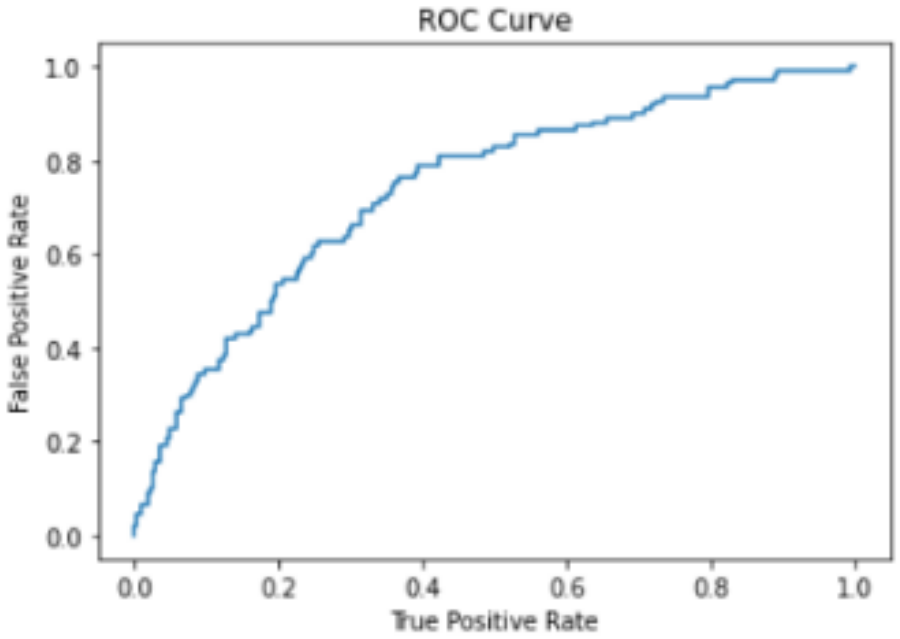
Beta Coefficients

[154]
```
beta = np.sort(lrModel.coefficients)
plt.plot(beta)
plt.ylabel('Beta Coefficients')
plt.show()
```



Plot of the ROC Curve

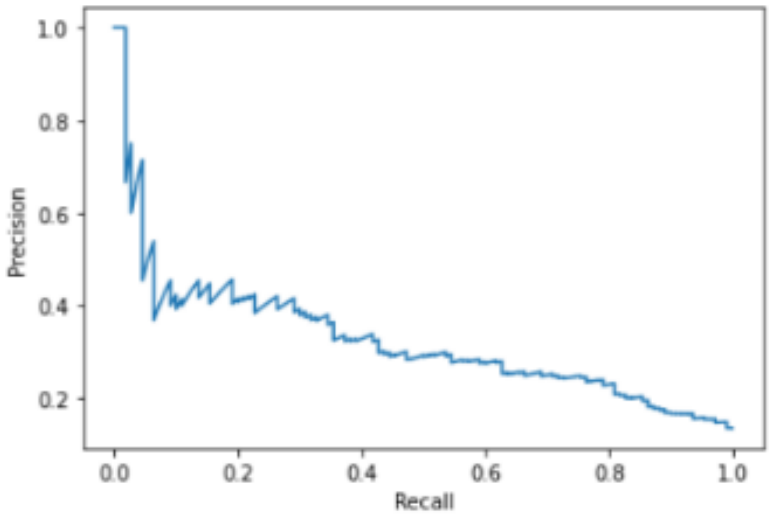
[155]
```
trainingSummary = lrModel.summary
roc = trainingSummary.roc.toPandas()
plt.plot(roc['FPR'],roc['TPR'])
plt.ylabel('False Positive Rate')
plt.xlabel('True Positive Rate')
plt.title('ROC Curve')
plt.show()
print('Training set areaUnderROC: ' +
str(trainingSummary.areaUnderROC))
```



Visualize the precision

[156]

```
pr = trainingSummary.pr.toPandas()
plt.plot(pr['recall'],pr['precision'])
plt.ylabel('Precision')
plt.xlabel('Recall')
plt.show()
```



Visualize the predictions

[157]

```
predictions = lrModel.transform(test)
predictions.select('Varsta', 'Sex', 'label', 'rawPrediction', 'prediction', 'probability').show(10)
```

```
+------+------+-----+--------------------+----------+--------------------+
|Varsta|   Sex|label|       rawPrediction|prediction|         probability|
+------+------+-----+--------------------+----------+--------------------+
|    78|female|  0.0|[2.47739620586088...|       0.0|[0.92254193972512...|
|    32|female|  0.0|[3.19295328407156...|       0.0|[0.96056823397809...|
|    47|female|  0.0|[3.27645215005637...|       0.0|[0.96361208682000...|
|    48|female|  0.0|[2.84264532086691...|       0.0|[0.94493726257457...|
|    85|female|  0.0|[1.70352965676057...|       0.0|[0.84599516634526...|
|    87|female|  0.0|[2.54731474976166...|       0.0|[0.92739290988736...|
|    39|female|  0.0|[3.91176989160257...|       0.0|[0.98038729056891...|
|    71|female|  0.0|[3.02228596113570...|       0.0|[0.95357083797530...|
|    81|female|  0.0|[0.77092600254329...|       0.0|[0.68372117263213...|
|    57|female|  0.0|[2.55310886080411...|       0.0|[0.92778209324832...|
+------+------+-----+--------------------+----------+--------------------+
only showing top 10 rows
```

Area under curve on the test dataset

[158]
```
predictionAndLabels = lrModel.evaluate(train)
evaluator = BinaryClassificationEvaluator()
predictionAndLabels.predictions.show()
```

```
+-----+--------------------+--------------------+----------+
|Stare|       rawPrediction|         probability|prediction|
+-----+--------------------+--------------------+----------+
|  yes|[1.82239750835842...|[0.86085356041197...|       0.0|
|  yes|[2.3863173273743,...|[0.91577796297303...|       0.0|
|  yes|[3.08988583514586...|[0.95647361239933...|       0.0|
|  yes|[2.43250517986687...|[0.91927263870054...|       0.0|
|  yes|[-0.4120204516436...|[0.39842775384312...|       1.0|
|  yes|[2.51820065331896...|[0.92540794442043...|       0.0|
|  yes|[2.86804563191645...|[0.94624402323005...|       0.0|
|  yes|[3.62043473107615...|[0.97392696635478...|       0.0|
|  yes|[3.88944945241336...|[0.97995347850438...|       0.0|
|  yes|[2.25915118828261...|[0.90543698007114...|       0.0|
|  yes|[2.85912434514355...|[0.94578841986533...|       0.0|
|  yes|[2.97960806934808...|[0.95164433884617...|       0.0|
|  yes|[3.49755492427848...|[0.97061811932763...|       0.0|
|  yes|[2.90706330722986...|[0.94819449894279...|       0.0|
|  yes|[2.15754924532702...|[0.89637212108316...|       0.0|
|  yes|[1.71612781627279...|[0.84762940201015...|       0.0|
|  yes|[3.19423581624958...|[0.96061678363889...|       0.0|
|  yes|[2.60280088830868...|[0.93104162267163...|       0.0|
|  yes|[2.29717188099106...|[0.90864254439121...|       0.0|
|  yes|[2.37718310817546...|[0.91507077179442...|       0.0|
+-----+--------------------+--------------------+----------+
```

[159]
```
print('Test Area Under ROC', evaluator.evaluate(predictions))
```

```
Test Area Under ROC 0.7440382599580736
```

Accuracy

[160]
```
# Pentru multiclass
evaluator = MulticlassClassificationEvaluator(predictionCol='prediction',
labelCol='label',metricName='accuracy')
acc = evaluator.evaluate(predictionAndLabels.predictions)
print(" Accuracy: ", acc)
```

```
Accuracy:  0.8640296662546354
```

5. Metode Deep Learning

Modelul Artificial Neural Network

Am ales acest model deoarece se potrivit pentru aceasta problema de clasificare mai ales prin performanta dar si prin complexitatea care ne este oferita in momentul prezicerii coloanelor tinta.

Install TensorFlow

[161]
```
!pip install tensorflow
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: tensorflow in /usr/local/lib/python3.7/dist-packages (2.8.2+zzzcolab20220527125636)
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.21.6)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (4.1.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.46.3)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.1.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages (from tensorflow) (57.4.0)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (3.1.0)
Requirement already satisfied: keras-preprocessing>=1.1.1 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.1.2)
Requirement already satisfied: protobuf<3.20,>=3.9.2 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (3.17.3)
Requirement already satisfied: tensorflow-estimator<2.9,>=2.8 in /usr/local/lib/python3.7/dist-packages (from tensorflow)
```

Import Tensorflow

[162]
```
import tensorflow as tf
```

Reading the data

[163]
```
tensor_data = pd.read_csv('processed_data.csv')
tensor_data.head()
```

	Numar	Stare	Varsta	Sex	Indice_de_masa_corporala	Hipertensiv	Fibrilatie_atriala	Boala_coronariana	Diabet	Anemii_deficitare	...	Sodiu_din_sange
0	1	alive	72	Female	37.588179	nu	nu	nu	da	da	...	138.750000
1	2	alive	75	Male	30.188278	nu	nu	nu	nu	da	...	138.888889
2	3	alive	83	Male	26.572634	nu	nu	nu	nu	da	...	140.714286
3	4	alive	43	Male	83.264629	nu	nu	nu	nu	nu	...	138.500000
4	5	alive	75	Male	31.824842	da	nu	nu	nu	da	...	136.666667

Transformarea datelor pentru a se potrivi modelului

[164]
```
#Modifications of the dataset

#Modifications of the Stare
tensor_data.loc[tensor_data["Stare"] == "alive", "Stare"] = 1
tensor_data.loc[tensor_data["Stare"] == "dead", "Stare"] = 0

#Modifications of the Hipertensiv
tensor_data.loc[tensor_data["Hipertensiv"] == "nu", "Hipertensiv"] = 0
tensor_data.loc[tensor_data["Hipertensiv"] == "da", "Hipertensiv"] = 1

#Modifications of the Fibrilatie_atriala
tensor_data.loc[tensor_data["Fibrilatie_atriala"] == "nu", "Fibrilatie_atriala"] = 0
tensor_data.loc[tensor_data["Fibrilatie_atriala"] == "da", "Fibrilatie_atriala"] = 1
```

```
#Modifications of the Boala_coronariana
tensor_data.loc[tensor_data["Boala_coronariana"] == "nu", "Boala_coronariana"] = 0
tensor_data.loc[tensor_data["Boala_coronariana"] == "da", "Boala_coronariana"] = 1

#Modifications of the Diabet
tensor_data.loc[tensor_data["Diabet"] == "nu", "Diabet"] = 0
tensor_data.loc[tensor_data["Diabet"] == "da", "Diabet"] = 1

#Modifications of the Anemii_deficitare
tensor_data.loc[tensor_data["Anemii_deficitare"] == "nu", "Anemii_deficitare"] = 0
tensor_data.loc[tensor_data["Anemii_deficitare"] == "da", "Anemii_deficitare"] = 1

#Modifications of the Depresie
tensor_data.loc[tensor_data["Depresie"] == "nu", "Depresie"] = 0
tensor_data.loc[tensor_data["Depresie"] == "da", "Depresie"] = 1

#Modifications of the Hiperlipemie
tensor_data.loc[tensor_data["Hiperlipemie"] == "nu", "Hiperlipemie"] = 0
tensor_data.loc[tensor_data["Hiperlipemie"] == "da", "Hiperlipemie"] = 1

#Modifications of the Insuficienta_renala
tensor_data.loc[tensor_data["Insuficienta_renala"] == "nu", "Insuficienta_renala"] = 0
tensor_data.loc[tensor_data["Insuficienta_renala"] == "da", "Insuficienta_renala"] = 1

#Modifications of the Boala_pulmonara_obstructiva_cronica
tensor_data.loc[tensor_data["Boala_pulmonara_obstructiva_cronica"] == "nu", "Boala_pulmonara_obstructiva_cronica"] = 0
tensor_data.loc[tensor_data["Boala_pulmonara_obstructiva_cronica"] == "da", "Boala_pulmonara_obstructiva_cronica"] = 1

X=tensor_data.drop(['Numar','Fibrilatie_atriala','Boala_coronariana','Anemii_deficitare','Hiperlipemie','Insuficienta_renala','Boala_pulmo
nara_obstructiva_cronica','Urina','Saturatia_pulsului_de_oxigen','Volumul_celule_rosii_din_sange','Hemoglobina_corpusculara_medie','C
oncentratia_medie_a_hemoglobinei_corpusculare','Volumul_corpuscular_mediu','Latimea_distributiei_globulelor_rosii','Timp_de_protro
mbina','Raport_internationalizat_normalizat','NT_proBNP','Creatin_kinaza','Creatina','Nitrogen_ureic','Interval_anionic','Ioni_de_magnezi
u','Concentratia_ionilor_de_oxigen','Bicarbote','Acid_lactic','Presiunea_partiala_a_dioxidului_de_carbon','Fractie_de_eliminare'],axis = 1)
y = tensor_data['Stare']
```

Processing the values of 'Sex' column

[165]
```
from sklearn.preprocessing import OneHotEncoder
encoder = OneHotEncoder(drop='first')
encoder_gen = encoder.fit_transform(X[["Sex"]]).toarray()
X[['is_female']] = encoder_gen
X.drop(['Sex'],axis=1,inplace=True)
```

Data Scaling

[166]
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()

standardized_attrs =
scaler.fit_transform(X[['Varsta','Indice_de_masa_corporala','Hipertensiv','Diabet','Depresie','Frecventa_cardiaca','Tensiune_arteriala_sistolica','Tensiune_arteriala_diastolica','Frecventa_respiratorie','Temperatura','Globule_rosii','Leucocite','Trombocite', 'Neurofile', 'Globule_albe', 'Limfocite', 'Glucoza', 'Potasiu_din_sange', 'Sodiu_din_sange', 'Calciu_din_sange', 'Clorura']])
X[['Varsta','Indice_de_masa_corporala','Hipertensiv','Diabet','Depresie','Frecventa_cardiaca','Tensiune_arteriala_sistolica','Tensiune_arteriala_diastolica','Frecventa_respiratorie','Temperatura','Globule_rosii','Leucocite','Trombocite', 'Neurofile', 'Globule_albe', 'Limfocite', 'Glucoza', 'Potasiu_din_sange', 'Sodiu_din_sange', 'Calciu_din_sange', 'Clorura']] =
standardized_attrs
X.describe()
```

	Varsta	Indice_de_masa_corporala	Hipertensiv	Diabet	Depresie	Frecventa_cardiaca	Tensiune_arteriala_sistolica
count	1175.000000	1175.000000	1175.000000	1175.000000	1175.000000	1175.000000	1175.000000
mean	0.687936	0.183849	0.718298	0.420426	0.119149	0.487210	0.336019
std	0.167948	0.092082	0.450020	0.493837	0.324102	0.159898	0.134810
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.575000	0.130155	0.000000	0.000000	0.000000	0.366364	0.238325
50%	0.725000	0.183812	1.000000	0.000000	0.000000	0.479224	0.323438
75%	0.825000	0.204755	1.000000	1.000000	0.000000	0.597936	0.417913
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

Divizarea train-test

[167]
```
from sklearn.model_selection import train_test_split
#X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=101)
```

Import tensorflow libraries for ANN

[168]
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

Crearea modelului si stabilirea parametrilor potriviti

[169]
```
model = Sequential()
model.add(Dense(units=100, activation='relu', input_shape=(23,))) # First layer
model.add(Dense(units=100, activation='relu')) # Second layer
model.add(Dense(units=1, activation='sigmoid')) # Output layer
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 100)               2400

 dense_1 (Dense)             (None, 100)               10100

 dense_2 (Dense)             (None, 1)                 101

=================================================================
Total params: 12,601
Trainable params: 12,601
Non-trainable params: 0
_________________________________________________________________
```

Compilarea modelului

[170]
```
model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

Data conversion to tensor type

[171]
```
X_train =tf.convert_to_tensor(X_train, dtype=tf.float32)
y_train = tf.convert_to_tensor(y_train, dtype=tf.float32)
X_test = tf.convert_to_tensor(X_test, dtype=tf.float32)
y_test = tf.convert_to_tensor(y_test, dtype=tf.float32)
```

Antrenarea modelului

[172]
```
history = model.fit(x=X_train, y=y_train, epochs=100, validation_data=(X_test, y_test),verbose=1,batch_size=128)
```

```
Epoch 98/100
6/6 [==============================] - 0s 8ms/step - loss: 3.2551e-05 - accuracy: 1.0000 - val_loss: 2.9466e-05
y: 1.0000
Epoch 99/100
6/6 [==============================] - 0s 8ms/step - loss: 3.1811e-05 - accuracy: 1.0000 - val_loss: 2.8804e-05
y: 1.0000
Epoch 100/100
6/6 [==============================] - 0s 8ms/step - loss: 3.1120e-05 - accuracy: 1.0000 - val_loss: 2.8169e-05
y: 1.0000
```

Calcularea predictiilor

[173]
```
y_pred = model.predict(X_test).reshape((-1,)) > 0.5
y_pred = y_pred.astype(np.int32)
y_pred[0:5]
```

```
array([1, 1, 1, 1, 1], dtype=int32)
```

Matrix of confusion

[174]
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred)
```

```
array([[ 59,   0],
       [  0, 411]])
```

Accuracy

[175]
```
model.evaluate(X_test, y_test)
```

```
15/15 [==============================] - 0s 2ms/step - loss: 2.8169e-05 - accuracy: 1.0000
[2.816872438415885e-05, 1.0]
```

Loss plot

[176]
```
plt.plot(history.history['loss'],c='cornflowerblue
',label='training loss')
plt.plot(history.history['val_loss'],c='goldenrod',
label='validation loss')
plt.legend()
```