

# **Presentation Summary**

- Introduction and approach
- Data preprocessing main features conclusions
- Unsupervised learning
- Supervised learning
- Models measure methods & accuracy
- Conclusions

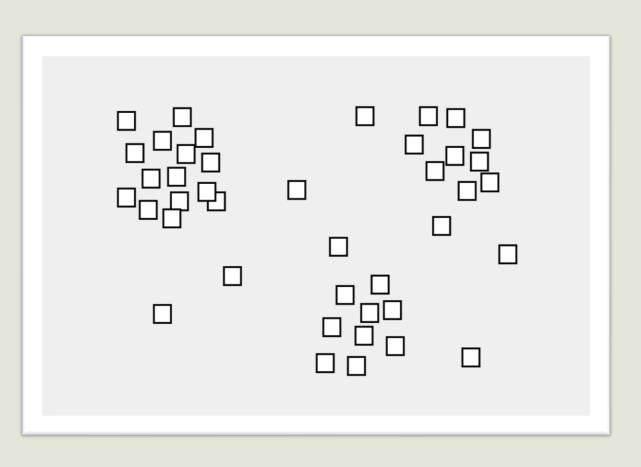
## Introduction and approach

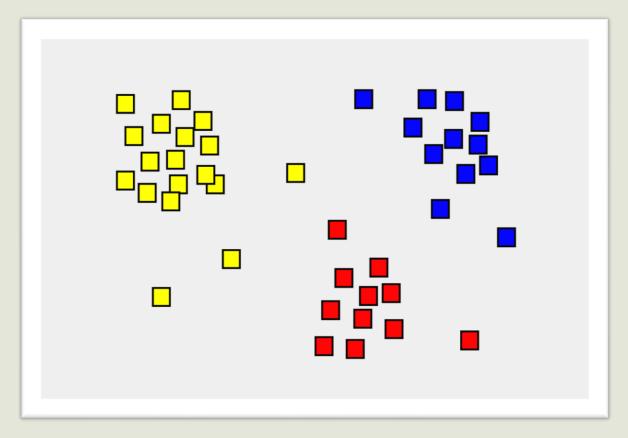
According to the Central Bureau of Statistics the data we are dealing with have high death rate.

We decided to use unsupervised learning to classify the risk groups. Unsupervised learning allows us to automatically identify patterns in the data and classify them into risk groups. This is a more efficient way of dealing with data that has a large rate of sick people, as it requires less manual intervention.

Year	Population	Death count	Death rate
2011	1,952,624	1704	0.0873%
2012	1,990,188	1699	0.0854%
2013	2,030,785	1694	0.0834%
2014	2,070,562	1733	0.0837%
2015	2,110,721	1697	0.0804%
2016	2,150,096	1798	0.0836%
2017	2,191,646	1780	0.0812%
2018	2,229,702	1770	0.0794%
Our data	12,438	197	1.5839%

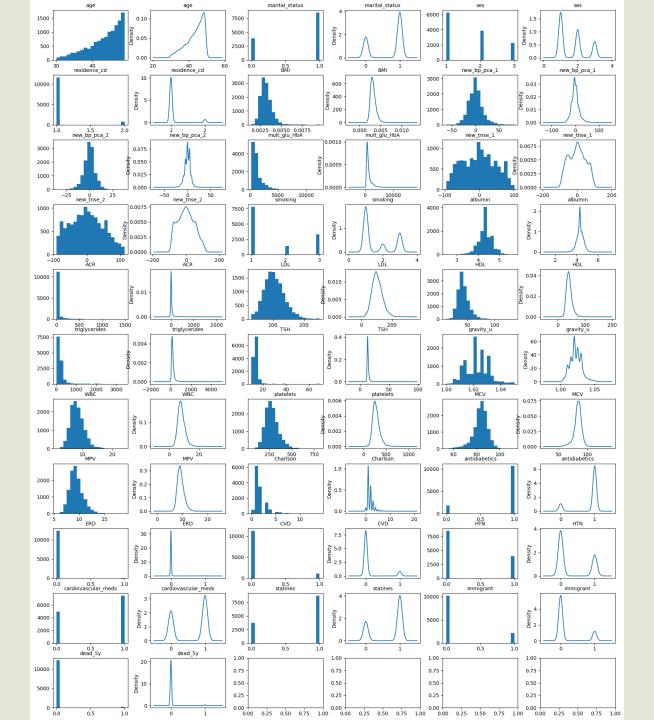
# Supervised VS Unsupervised





# Data preprocessing

- Null including features
- Categorical features (strings)
- Dimensions reduction using correlative features
- Normalize
- Data balancing



# Null including features

#### Discover features

 Finding the null rate in each feature contains null

#### Visualize distribution

- Value count
- Summary of the central tendency, dispersion and shape of a dataset's distribution
- Histogram and density estimation

#### Feature removing

- More than 50% nulls
- More than 50% zeros

#### Feature filling

- The density estimation like normal – we filled with the median because it is noise insensitive
- Fill the previous values for non normal like density estimation

#### **Discover features**

```
nullContainAtt = dict()
print(f'{"Feature":<14} {"Null Values":} {"Null Rate":}')</pre>
for attName in train df.columns:
     train_df[train_df == "NA"] = np.nan
    train_df[train_df == "NaN"] = np.nan
     numOfNulls = train df[attName].isna().sum()
    if numOfNulls > 0:
         nullContainAtt[attName] = [numOfNulls, numOfNulls/numberOfPeople]
         print(f'{bold start}{attName:<14} {numOfNulls:<10}{bu end} {round(</pre>
 train df[train df == "NA"] = np.nan
train df[train df == "NaN"] = np.nan
Feature
               Null Values Null Rate
albumin
               1463
                                 12.0%
alb24h
               8918
                                 72.0%
ACR
               1212
                                 10.0%
gravity_u
               2530
                                 20.0%
nitrites u
               2520
                                 20.0%
leuko u
               2507
                                 20.0%
proteinuria
               2506
                                 20.0%
```

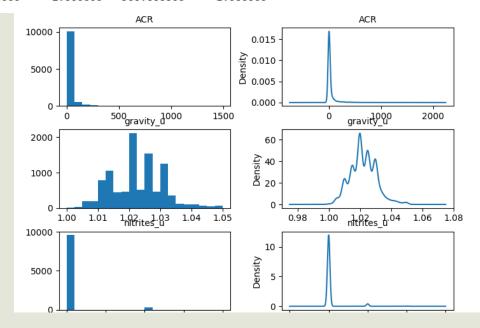
#### Visualize distribution

```
Uniuge values:
ACR
                [6.0, 3.87, 6.35, 5.0, 0.48, 30.0, 4.16, 3.14,...
gravity_u
               [1.025, 1.021, 1.03, 1.017, 1.028, 1.02, nan, ...
nitrites u
                                            [0.0, nan, 1.0, 2.0]
leuko u
               [0.0, 1.0, nan, 75.0, 100.0, 25.0, 250.0, 500....
proteinuria
                                                  [0.0, nan, 1.0]
albumin
               [3.94, 4.5, nan, 4.6, 4.4, 4.8, 4.25, 4.3, 4.1...
dtype: object
                       gravity_u nitrites_u
                ACR
                                                    leuko u proteinuria \
                     9908.000000
count 11226.000000
                                  9918.000000
                                                9931.000000
                                                            9932.000000
          38.829212
                        1.022250
                                     0.037709
                                                  40.851576
                                                                0.050745
mean
std
         117.962082
                        0.008226
                                      0.202310
                                                 120.788718
                                                                0.219488
                                      0.000000
                                                                0.000000
min
           0.000000
                        1.000000
                                                   0.000000
25%
                                      0.000000
                                                                0.000000
           3.870000
                        1.016000
                                                   0.000000
50%
           7.000000
                                     0.000000
                        1.021000
                                                   0.000000
                                                                0.000000
75%
          21.000000
                        1.028000
                                      0.000000
                                                                0.000000
                                                   0.000000
        1494.570000
                        1.050000
                                      2.000000
                                                 500.000000
                                                                1.000000
max
```

#### Feature removing

```
train_df = train_df.drop(["cancer", "alb24h", "nitrites_u", "proteinuria", "leuko_u"], axis=1)
train_df.head()
```

:		age	sex	marital_status	ses	residence_cd	residence	weigh	heigh	ВМІ	bp_sys	 Charlson	framingham_cvd	ant
	0	49	2	U	1	1	urban	84.0	160.0	32.79	116	 1	0.12908	
	1	46	2	М	1	1	urban	67.0	152.0	29.38	113	 1	0.08500	
	2	46	2	U	1	1	urban	62.4	148.0	28.31	133	 1	0.10715	
	3	45	1	U	2	1	urban	73.9	166.0	26.85	100	 1	0.10110	
	4	49	2	U	1	1	urban	53.0	161.0	20.45	110	 1	0.03950	



#### **Feature filling**

```
train_df['gravity_u'] = train_df['gravity_u'].fillna(method='ffill')
train_df['albumin'] = train_df['albumin'].fillna(train_df['albumin'].median())
```

# Categorical features

#### Discover features

• Finding the string contain features

# Look for numerical discrete mapping

 Looking for features that have the same information but in numerical form

#### Convert to numerical

- Understand what are the feature unique values.
- Binary features will map into {0,1}
- Ordered categories will map into discrete numbers from 1 to N groups

#### **Discover features**

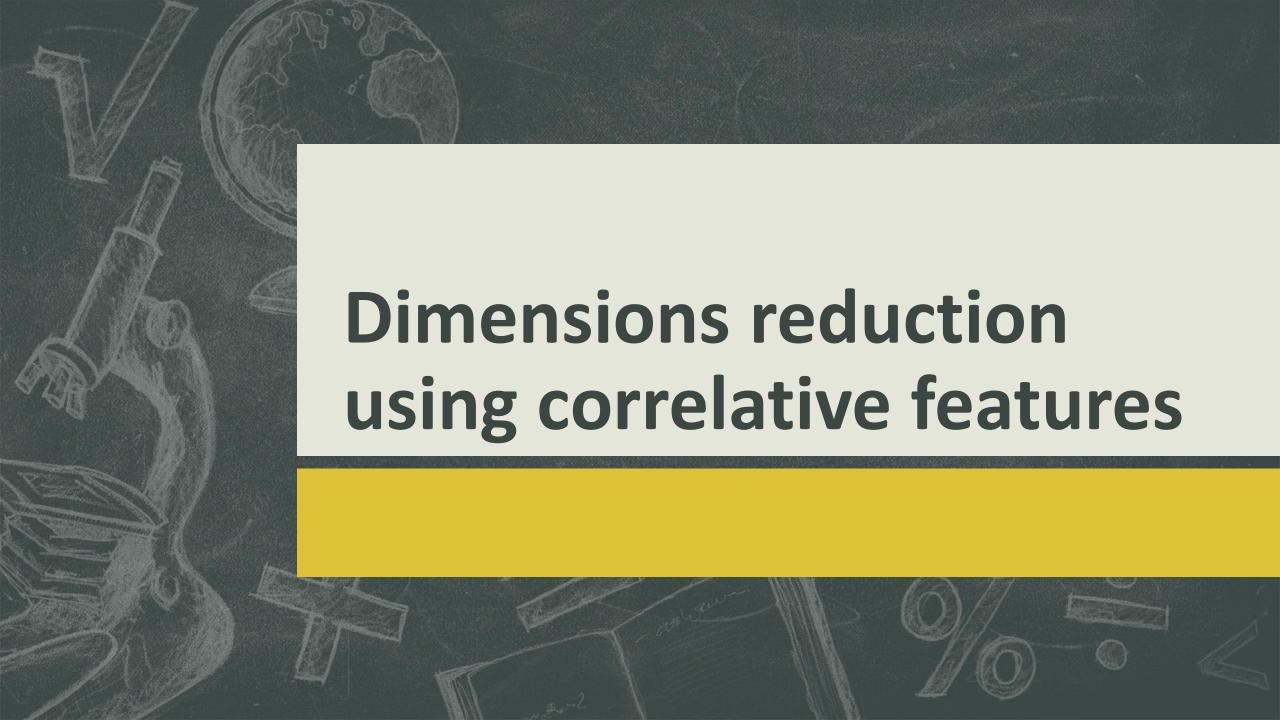
train	_df.select_dt	types(excl	.ude=['in	t64', 'float64
	marital_status	residence	bp_cat	smoking_status
0	U	urban	Normal	non_smoker
1	М	urban	Normal	non_smoker
2	U	urban	Pre-HTN	non_smoker
3	U	urban	Normal	current_smoker
4	U	urban	Normal	non_smoker
12433	U	urban	Pre-HTN	non_smoker
12434	М	urban	Normal	non_smoker
12435	U	urban	Normal	current_smoker
12436	М	urban	Normal	non_smoker
12437	М	urban	Normal	current_smoker
12438 r	ows × 4 colum	ns		

#### **Look for numerical discrete mapping**

#### **Convert to numerical**

```
train_df['marital_status'] = train_df['marital_status'].map({"U" : 0, "M": 1})

train_df['bp_cat'] = train_df['bp_cat'].map({"Normal": 0, "Pre-HTN": 1, "HTN-G1": 2, "HTN-G2": 3, "HTN-G3": 4}).astype(int)
```



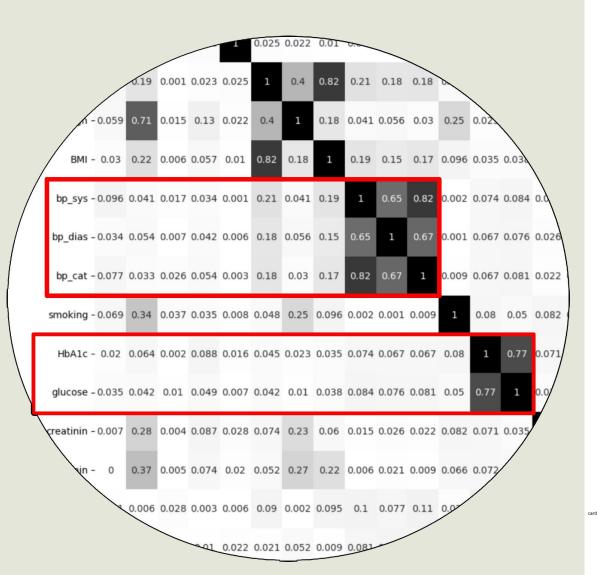
## Correlation matrices

#### Pearson

- Measures the linear association between two variables and is calculated as the covariance of the two variables divided by the product of their standard deviations.
- Used for continuous feature
- -1 to 1 range.
  - -1: perfect negative linear relationship
  - 0: no linear relationship
  - 1: perfect positive linear relationship.

### Spearman

- Rank-based measure of association and is calculated as the Pearson correlation between the ranks of the two variables. It is used when the relationship between the two variables is **not linear**.
- Used when the data is ordinal or has outliers
- -1 to 1 range.
  - -1: perfect negative monotonic relationship
  - 0: no monotonic relationship
  - 1: perfect positive monotonic relationship.



#### The method used: Pearson

.002 0.006 0.048 0.003 0.069 0.059 0.03 0.096 0.034 0.077 0.069 0.02 0.035 0.007 0 0.011 0.017 0.008 0.023 0.011 0.021 0.052 0.013 0.032 0.013 0.049 0.02 0.077 0.39 0.047 0.018 0.076 0.11 0.15 0.077 0.01 0.001 0.092 0.016 0.19 0.71 0.22 0.041 0.054 0.033 0.34 0.064 0.042 0.28 0.37 0.006 0.023 0 0.31 0.11 0.21 0.11 0.037 0.51 0.29 0.2 0.054 0.035 0.45 0.015 0.011 0.11 0.01 0.015 0.1 0.032 0.007 marital\_status - 0.006 0.001 1 0.16 0.022 0.001 0.015 0.006 0.017 0.007 0.026 0.037 0.002 0.01 0.004 0.005 0.028 0.015 0.003 0.015 0.025 0.024 0.011 0.036 0.004 0.015 0.002 0.017 0.027 0.013 0.027 0.009 0.021 0.009 0.006 0 0 0.068 0.11 ses - 0.048 0.092 0.16 1 0.036 0.023 0.13 0.057 0.034 0.042 0.054 0.035 0.088 0.049 0.087 0.074 0.003 0.01 0.011 0.002 0.004 0.019 0.005 0.003 0.021 0.051 0.078 0.018 0.004 0.037 0.037 0.019 0.016 0.01 0.013 0.007 0.22 0.023 residence\_cd - 0.003 0.016 0.022 0.036 1 0.025 0.022 0.01 0.001 0.006 0.003 0.008 0.016 0.007 0.028 0.02 0.02 0.022 0.022 0.022 0.022 0.022 0.023 0.004 0.006 0.021 0.001 0.008 0 0 0.014 0.001 0.015 0.002 0.008 0.009 0.024 0.019 0.011 0.002 0.004 0.002 weigh - 0.063 0.19 0.001 0.023 0.025 1 0.4 0.82 0.21 0.18 0.18 0.048 0.045 0.042 0.074 0.052 0.09 0.021 0.047 0.022 0.12 0.066 0.051 0.13 0.14 0.028 0.05 0.03 0.062 0.14 0.034 0.014 0.041 0.15 0.13 0.031 0.076 0.006 heigh - 0.059 0.71 0.015 0.13 0.022 0.4 1 0.18 0.041 0.056 0.03 0.25 0.23 0.01 0.23 0.27 0.02 0.052 0.025 0. 0.19 0.15 0.17 0.096 0.035 0.038 0.06 0.22 0.095 0.009 0.034 0.068 0.08 0.16 0.006 0.12 0.065 0.12 0.14 0.012 0.048 0.032 0.04 0.015 0.003 0.15 0.14 0.003 0.044 0.008 bp\_sys - 0.096 0.041 0.017 0.034 0.001 0.21 0.041 0.19 1 0.65 0.82 0.002 0.074 0.084 0.015 0.006 0.1 0.081 0.04 0.036 0.093 0.043 0.008 0.039 0.066 0.007 0.054 0.005 0.036 0.33 0.009 0.014 0.04 0.26 0.19 0.054 0.037 0.055 tp\_diss - 0.034 0.054 0.007 0.042 0.006 0.18 0.056 0.15 0.65 1 0.05 1 0.05 0.1 0.05 0.1 0.05 0.01 0.067 0.076 0.026 0.021 0.077 0.096 0.053 0.033 0.099 0.018 0.015 0.042 0.08 0.007 0.036 0.006 0.024 0.23 0.013 0.008 0.016 0.19 0.13 0.056 0.028 0.021 bp\_cat - 0.077 0.033 0.026 0.054 0.003 0.18 0.03 0.17 0.82 0.67 1 0.009 0.067 0.081 0.022 0.009 0.11 0.088 0.042 0.023 0.099 0.047 0.002 0.038 0.058 0.002 0.048 0.009 0.039 0.3 0.013 0.002 0.052 0.25 0.19 0.029 0.042 0.055 smoking - 0.069 0.34 0.037 0.035 0.008 0.048 0.25 0.096 0.002 0.001 0.009 1 0.08 0.05 0.082 0.066 0.024 0.057 0.058 0.17 0.095 0.081 0.051 0.2 0.21 0.073 0.18 0.047 0.073 0.55 0.036 0.003 0.14 0.015 0.047 0.092 0.004 0.038 HbAlc - 0.02 0.064 0.002 0.088 0.016 0.045 0.023 0.035 0.074 0.067 0.067 0.067 0.06 1 0.77 0.071 0.072 0.13 0.18 0.1 0.084 0.19 0.047 0.12 0.076 0.2 0.070 0.079 0.072 0.026 0.16 0.24 0.035 0.034 0.03 0.058 0.062 0.035 0.018 creatinin - 0.007 0.28 0.004 0.087 0.028 0.074 0.23 0.06 0.015 0.026 0.022 0.082 0.071 0.035 1 0.012 0.079 0.008 0.019 0.098 0.057 0.001 0.025 0.002 0.032 0.089 0.09 0.01 0.16 0.12 0.07 0.26 0.079 0.025 0.012 0.026 0.027 0.045 0 0.37 0.005 0.074 0.02 0.052 0.27 0.22 0.006 0.021 0.009 0.066 0.072 0.048 0.012 1 0.062 0.033 0.009 0.091 0.002 0.14 0.033 0.065 0.21 0.14 0.12 0.004 0.056 0.11 0.005 0.035 0.016 0.023 0.019 0.067 0.003 0.03 ACR - 0.011 0.006 0.028 0.003 0.006 0.09 0.002 0.095 0.1 0.077 0.11 0.024 0.13 0.13 0.079 0.062 1 0.053 0.014 0.023 0.11 0.054 0.004 0.061 0.028 0.024 0.043 0.01 0.16 0.074 0.033 0.068 0.084 0.084 0.084 0.12 0.008 0.006 0.082 cholesterol\_total - 0.017 0.023 0.015 0.01 0.022 0.021 0.052 0.001 0.052 0.009 0.081 0.096 0.088 0.057 0.18 0.19 0.008 0.033 0.053 1 0.8 0.22 0.4 0.034 0.035 0.047 0.085 0.065 0.021 0.001 0.026 0.027 0.026 0.002 0.031 0.002 0.029 0.1 0.018 0.036 0 0.003 0.011 0.022 0.047 0.025 0.034 0.04 0.053 0.042 0.058 0.1 0.099 0.019 0.009 0.014 0.08 1 0.16 0.017 0.041 0.029 0.009 0.083 0.041 0.035 0.002 0.058 0.19 0.058 0.01 0.059 0.039 0.076 0.054 0.016 0.031 HDL - 0.023 0.31 0.015 0.002 0.025 0.22 0.27 0.068 0.036 0.033 0.023 0.07 0.084 0.094 0.098 0.091 0.023 0.22 0.16 1 0.026 0.037 0.056 0.08 0.18 0.1 0.024 0.007 0.042 0.26 0.082 0.011 0.087 0.029 0.035 0.067 0.06 0.015 triglycerides - 0.011 0.11 0.025 0.004 0.004 0.12 0.079 0.08 0.099 0.099 0.099 0.099 0.099 0.095 0.19 0.22 0.057 0.002 0.11 0.4 0.017 0.26 1 0.017 0.028 0.12 0.11 0.006 0.002 0.014 0.057 0.3 0.045 0.012 0.067 0.062 0.052 0.1 0.013 0.009 TSH-0.021 0.21 0.024 0.019 0.006 0.066 0.15 0.16 0.043 0.018 0.047 0.081 0.047 0.05 0.001 0.14 0.054 0.034 0.041 0.037 0.017 1 0.028 0.005 0.026 0.092 0.41 0.015 0.061 0.081 0.024 0.043 0.016 0.069 0.057 0.059 0.004 0.057 gravity\_u - 0.052 0.11 0.011 0.005 0.021 0.051 0.083 0.006 0.008 0.015 0.002 0.051 0.051 0.051 0.051 0.051 0.051 0.051 0.051 0.051 0.051 0.051 0.051 0.051 0.051 0.051 0.051 0.051 0.051 0.051 0.051 0 WBC - 0.013 0.037 0.036 0.003 0.001 0.13 0.028 0.12 0.039 0.042 0.038 0.2 0.076 0.054 0.002 0.065 0.061 0.047 0.009 0.08 0.12 0.005 0.013 1 0.11 0.3 0.011 0.016 0.052 0.14 0.04 0.024 0.06 0.053 0.058 0.07 0.006 0.031 RBC - 0.032 0.51 0.004 0.021 0.008 0.14 0.36 0.065 0.066 0.08 0.058 0.21 0.2 0.15 0.032 0.21 0.28 0.085 0.086 0.08 0.058 0.21 0.2 0.15 0.032 0.21 0.028 0.085 0.083 0.18 0.11 0.026 0.092 0.11 1 0.15 0.28 0.039 0.006 0.27 0.18 0.043 0.053 0.018 0.008 0.044 0.062 0.011 platelets - 0.013 0.29 0.015 0.051 0 0 0.028 0.23 0.12 0.007 0.007 0.002 0.073 0.007 0.005 0.089 0.14 0.024 0.065 0.041 0.1 0.006 0.092 0.026 0.3 0.15 1 0.12 0.3 0.005 0.098 0.003 0.002 0.005 0.029 0.034 0.008 0.007 0.005 MCV - 0.049 0.2 0.002 0.078 0.014 0.05 0.14 0.15 0.14 0.05 0.04 0.14 0.05 0.04 0.18 0.079 0.06 0.09 0.12 0.043 0.021 0.035 0.024 0.002 0.11 0.08 0.011 0.28 0.12 1 0.064 0.004 0.14 0.017 0.018 0.017 0.018 0.017 0.051 0.078 0.014 MPV - 0.02 0.054 0.017 0.018 0.001 0.03 0.012 0.005 0.006 0.009 0.047 0.072 0.053 0.01 0.004 0.01 0.002 0.007 0.014 0.015 0.031 0.016 0.039 0.3 0.064 1 0.014 0.034 0.029 0.014 0.003 0.002 0.004 0.006 0.013 0.029 Charlson - 0.077 0.035 0.027 0.004 0.015 0.062 0.024 0.048 0.036 0.024 0.039 0.073 0.026 0.023 0.16 0.056 0.16 0.026 0.058 0.042 0.057 0.061 0.009 0.052 0.006 0.005 0.004 0.014 1 0.09 0.063 0.12 0.31 0.12 0.14 0.058 0.022 0.068 framingham\_cvd - 0.39 0.45 0.013 0.037 0.002 0.14 0.29 0.032 0.33 0.23 0.3 0.55 0.16 0.14 0.19 0.055 0.16 0.14 0.12 0.11 0.074 0.27 0.19 0.26 0.3 0.081 0.045 0.14 0.27 0.098 0.14 0.034 0.09 1 0.061 0.005 0.17 0.16 0.2 0.18 0.037 0.055 antidiabetics - 0.047 0.015 0.027 0.037 0.008 0.034 0.004 0.04 0.009 0.013 0.013 0.013 0.036 0.24 0.19 0.07 0.005 0.033 0.026 0.058 0.082 0.045 0.024 0.029 0.04 0.018 0.003 0.017 0.029 0.063 0.061 1 0.024 0.006 0.012 0.087 0.19 0.032 0.048 ERD - 0.018 0.011 0.009 0.019 0.009 0.014 0.003 0.015 0.014 0.008 0.002 0.003 0.035 0.025 0.06 0.035 0.068 0.002 0.01 0.011 0.012 0.043 0.035 0.024 0.043 0.002 0.018 0.014 0.12 0.005 0.024 1 0.05 0.015 0.017 0.001 0.023 0.053 CVD - 0.076 0.11 0.021 0.016 0.024 0.041 0.076 0.003 0.04 0.016 0.052 0.14 0.034 0.015 0.079 0.016 0.084 0.031 0.059 0.087 0.067 0.016 0.022 0.06 0.053 0.005 0.017 0.003 0.31 0.17 0.006 0.05 1 0.12 0.16 0.12 0.16 0.10 0.008 0.06 HTN - 0.11 0.001 0.009 0.01 0.019 0.15 0.002 0.15 0.02 0.15 0.26 0.19 0.25 0.015 0.03 0.033 0.025 0.023 0.084 0.002 0.039 0.029 0.062 0.069 0.006 0.053 0.018 0.029 0.051 0.002 0.12 0.16 0.012 0.015 0.12 1 0.43 0.085 0.03 0.025 cardiovascular meds - 0.16 0.015 0.006 0.013 0.011 0.13 0.001 0.14 0.19 0.13 0.09 0.047 0.058 0.042 0.012 0.019 0.12 0.029 0.076 0.035 0.052 0.057 0.002 0.058 0.008 0.034 0.034 0.034 0.004 0.14 0.2 0.087 0.017 0.16 0.43 statines - 0.15 0.1 0 0.007 0.002 0.031 0.062 0.003 0.054 0.036 0.029 0.092 0.062 0.049 0.026 0.067 0.008 0.1 0.054 0.067 0.0 0.059 0.006 0.07 0.044 0.008 0.051 0.006 0.058 0.18 0.19 0.001 0.1 0.085 0.18 1 0.002 0.002 dead 5y - 0.01 0.007 0.11 0.023 0.002 0.006 0.006 0.006 0.006 0.006 0.005 0.021 0.055 0.036 0.018 0.012 0.045 0.03 0.082 0.036 0.031 0.019 0.009 0.057 0.002 0.031 0.011 0.002 0.014 0.029 0.068 0.055 0.048 0.053 0.06 0.025 0.006 0.002 0.03

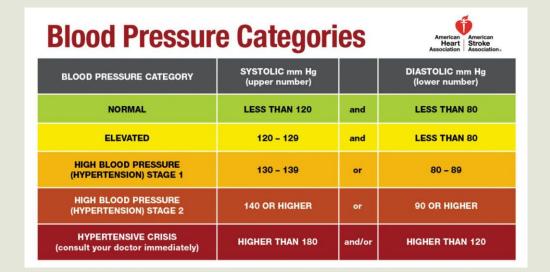
## Continuous Dimension Reduction examples

**BP features:** we can see high correlation between `bp\_sys`, `bp\_dias` and `bp\_cat`.

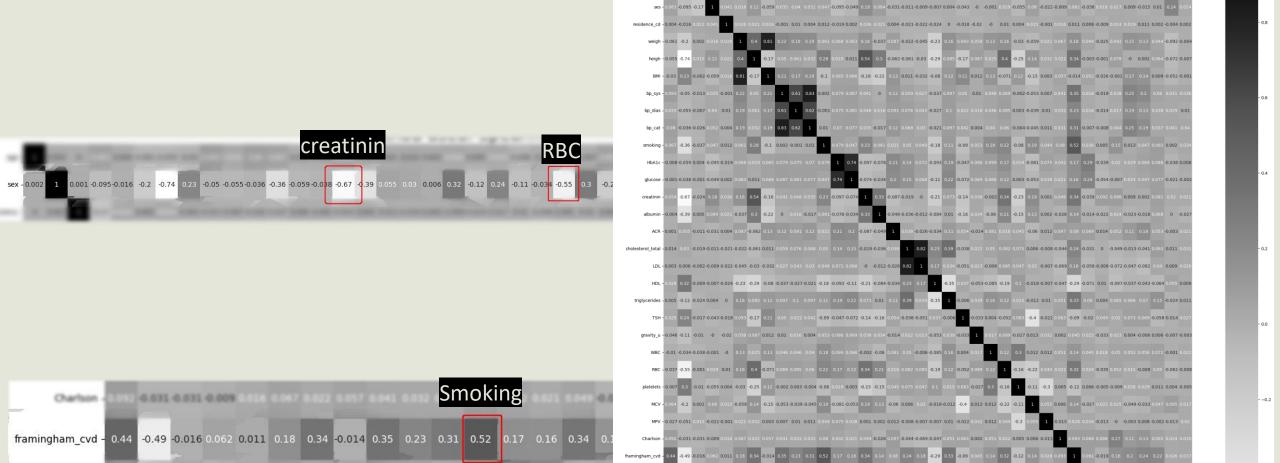
Why we used PCA:

- 1. Dimension reduction less features == less computing power.
- 2. Creating new two orthogonal features the new features have no correlation.

**Trail and Error:** `glucose` and `HbA1c` features have 0.77 correlation rate, by using trail and error approach we found out that multiply those features correlates well with both features and we can drop both and use the new feature







antidiabetics

-0.015

0.038 0.008

0.003

-0.007

FD CFT 4.039 4.0

antidiabetics - 0.035 -0.015 0.027 -0.038 0.008 0.044 -0.003 0.052 0.016 0.016 -0.007 0.036 0.29 0.24 -0.039 -0.014 0.089 -0.031 -0.058 -0.071 0.08 -0.02 0.08

019 -0.007 -0.11 0.024 0.002 -0.004 -0.007 -0.001 0.036 0.01 0.04 0.034 0.008 -0.002 0.021 -0.027 0.021 0.031 0.02

-0.015 0.006 -0.015 0.011 0.13 0.002 0.14 0.2 0.13 0.19 0.

HTN - 0.11 0.001-0.009 0.009 0.019 0.15 -0 0.17 0.25 0.19 0.25 0.013 0.029 0.024 0.009 0.023 0.11 0.013-0.047-0.037 0.066 0.073 0.004

PD - 0.01 -0.011 -0.009 -0.01 -0.009 -0.025 -0.001 -0.026 -0.019 -0.014 -0.008 0.005 -0.019 -0.024 0.022 -0.014 0.08 0.01 0.004 0.049 -0.033 0.018 -0.035 -0.005 0.022 -0.013 0.086 -0.019 -0.024 1 0.005 0.05 0.024 0.014 0.0

1.1 0 001 0.002 0.044 0.064 0.009 0.06 0.038 0.037 0.092 0.088 0.077 0.011 0.068 0.053 0.093 0.04 0.064 0.15 0.058 0.006 0.071 0.05 0.011 0.

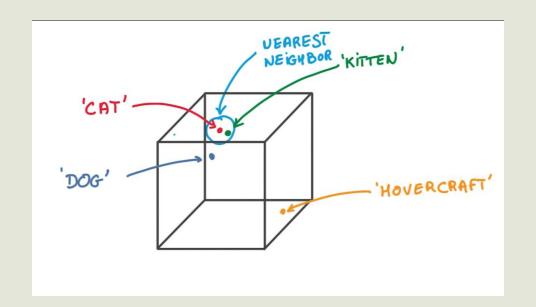
102 0.068 0.24 0.004-0.092-0.072-0.052 0.031 0.029 0.041 0.002-0.038-0.021 0.02 0 0.003 0.011 0.009 0.055 0.024 0.014 0.007 0.001 0.062 0.004 0.

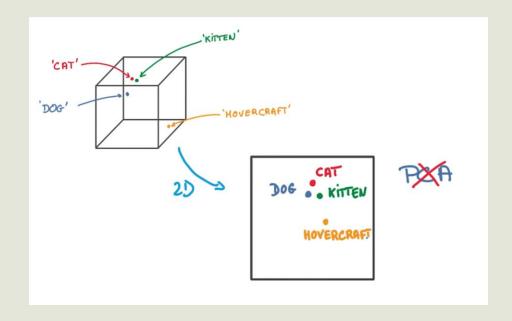
47 0.064 0.047 0.002 -0.018 0.18 -0.041-0.082-0.043 0.07 0.06

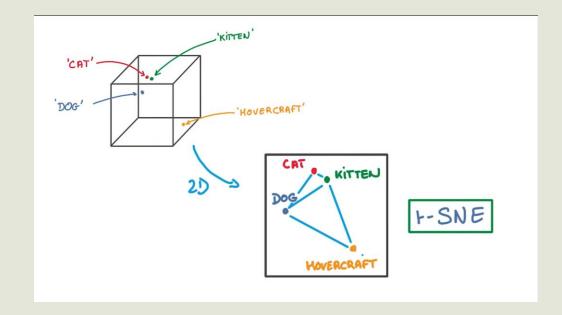
26 0.009 0.011 0.027 -0.003 0.022 -0.008 -0.005 0

# District Dimension Reduction examples

- TSNE (t-distributed Stochastic Neighbor Embedding):
  - 1. Compute pairwise similarities calculate the pairwise similarities between all data points in the high-dimensional space. This can be done using a similarity measure such as the Euclidean or Gaussian kernel.
  - 2. Initialize the low-dimensional representation The next step is to randomly initialize the low-dimensional representation of the data points. This is typically done in a 2D or 3D plot.
  - 3. Optimize the cost function The main objective of t-SNE is to minimize a cost function that measures the difference between the pairwise similarities in the high-dimensional space and the pairwise similarities in the low-dimensional space. The cost function is optimized using gradient descent or similar optimization techniques.
  - 4. Compute the low-dimensional representation After the cost function is minimized, the final low-dimensional representation of the data is obtained. This representation can then be plotted in a 2D or 3D scatter plot to visualize the relationships between the data points.



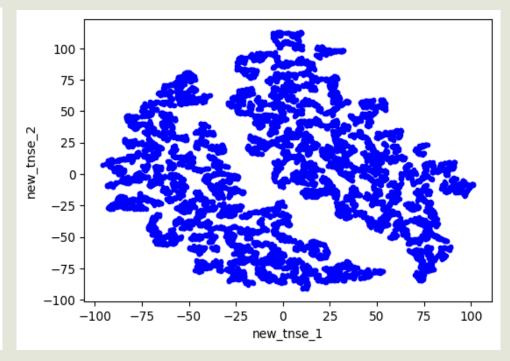




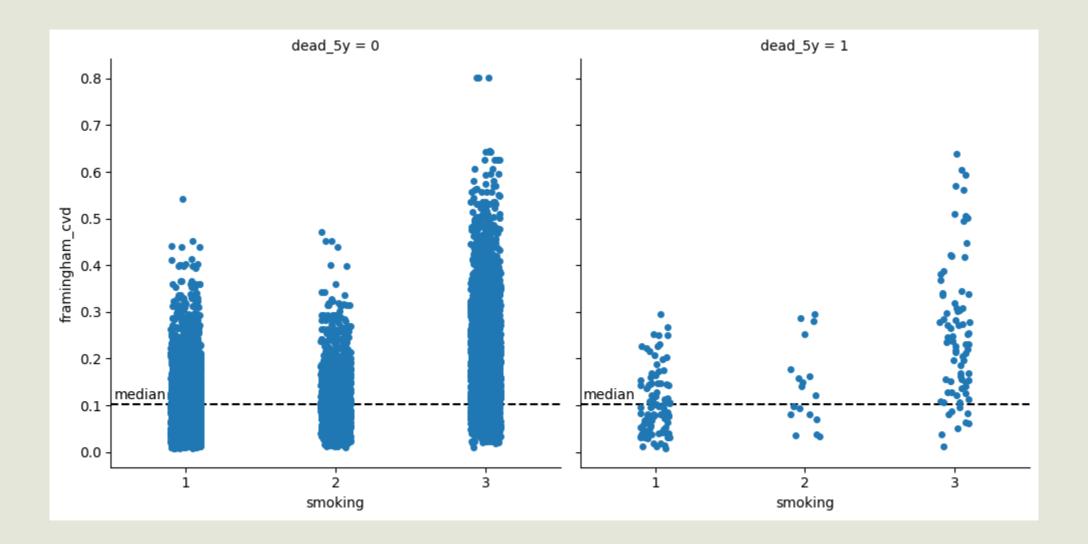
#### Original features

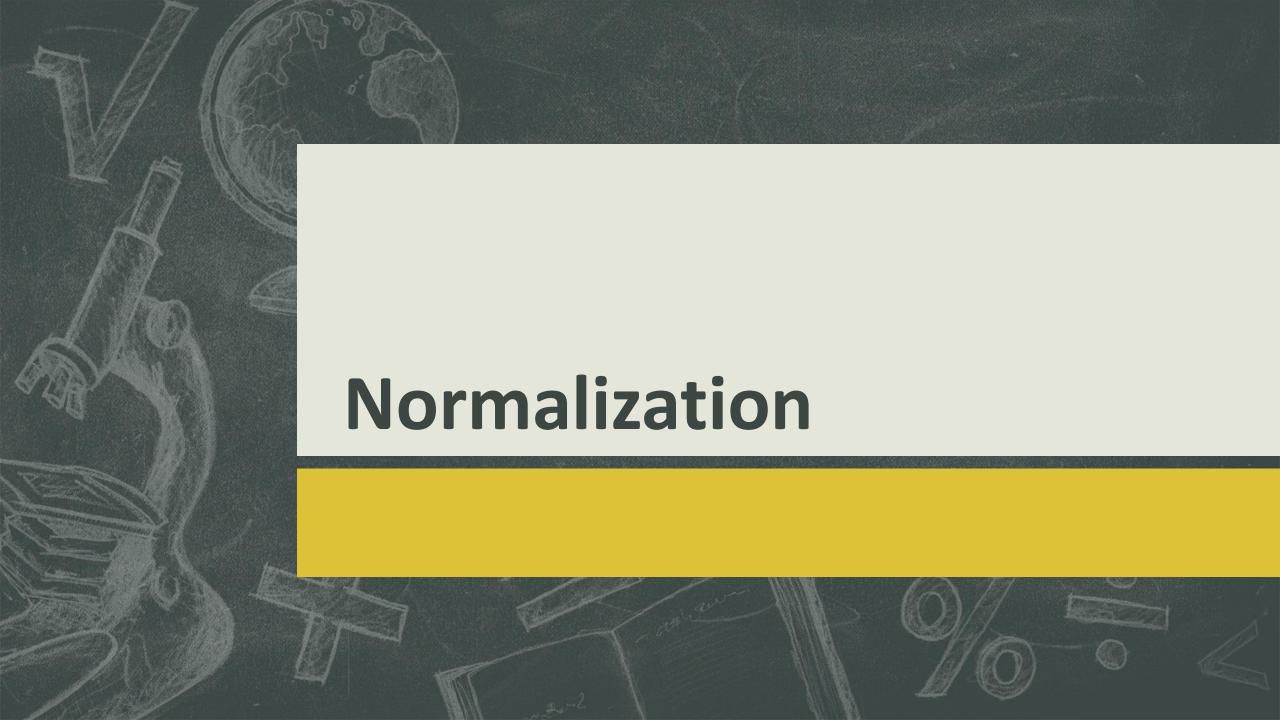
#### 

#### TNSE new features



```
tsne = TSNE(2)
p = tsne.fit_transform(train_df[['sex', 'RBC', 'creatinin']])
index = min(col_dict['RBC'], col_dict['creatinin'])
train_df.insert(index, 'new_tnse_1', 0)
train_df.insert(index+1, 'new_tnse_2', 0)
train_df['new_tnse_1'] = p[:, 0]
train_df['new_tnse_2'] = p[:, 1]
```



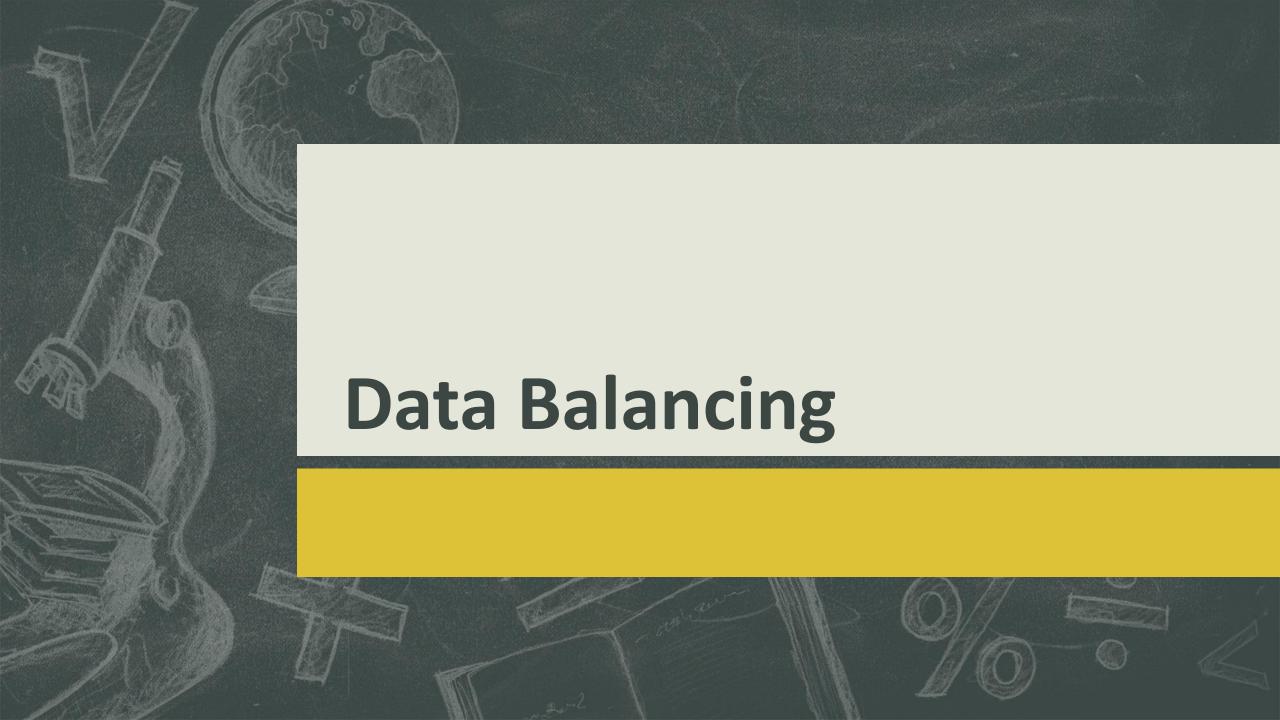


## Min-Max Normalization

- Also referred as Feature Scaling, performs linear transformation on the data.
- This technique gets all the scaled data in the range (0, 1).
- Disadvantage: Does not handle outliers very well.

  For example: if you have 99 values between 0 and 40, and one value is 100, then the 99 values will all be transformed to a value between 0 and 0.4. That data is just as squished as before!
- We used the method for normal distribution like features.
- Formula:

$$x_{scaled} = rac{x - x_{min}}{x_{max} - x_{min}}$$

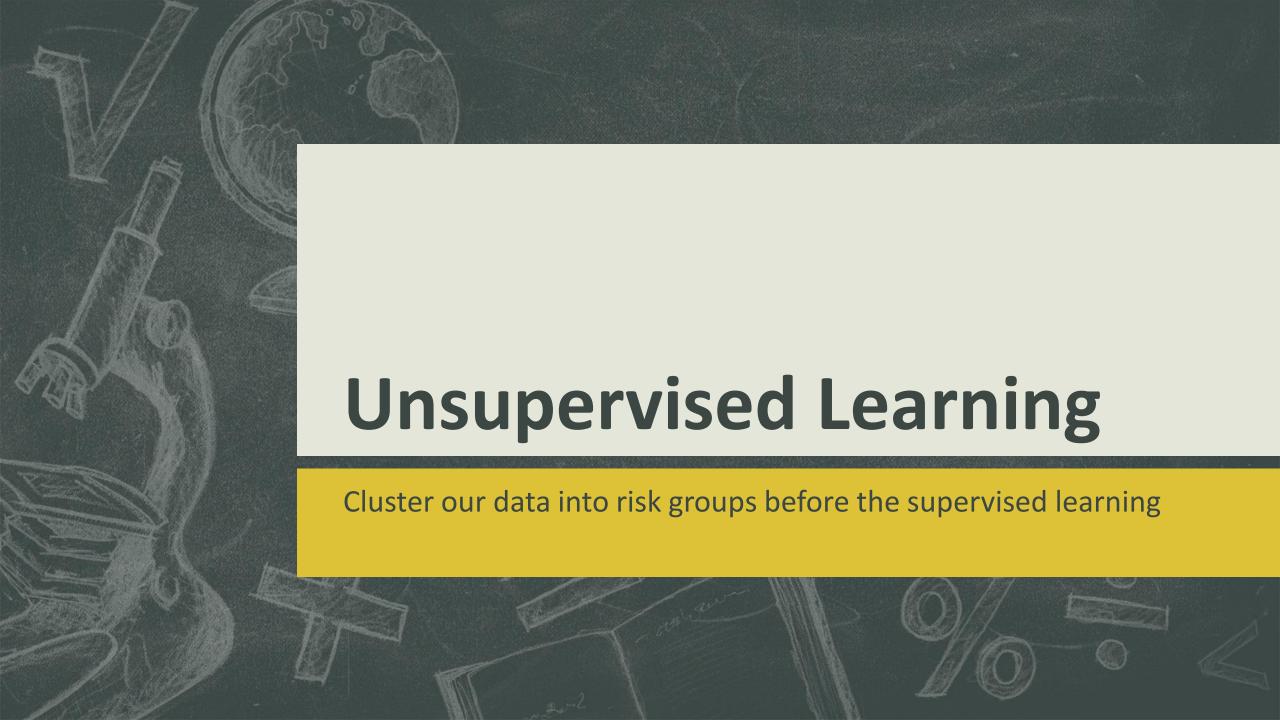


## SMOTE-NC

- SMOTE-NC (Synthetic Minority Over-sampling Technique for Nominal and Continuous) is a technique for oversampling imbalanced datasets in machine learning.
- It combines the Synthetic Minority Over-sampling Technique (SMOTE) with the nearest neighbors correction for continuous features (NC).

Let  $x_i$  be a minority class sample and  $N_i$  be its k nearest neighbors. Let  $d_{ij}$  be the Euclidean distance between  $x_i$  and  $x_j$  for j in  $N_i$ . Then, the weighted average of  $x_i$  and its nearest neighbors is computed as follows:  $x'_i = x_i + r * (x_j - x_i)$ where r is a random number between 0 and 1 and j is randomly selected from  $N_i$ .

SMOTE-NC generates synthetic samples for the minority class by interpolating between minority class samples and their nearest neighbors. The nearest neighbors correction is applied to ensure the generated samples are consistent with the distribution of the continuous features in the minority class.



## K Means & Mini Batch K Means

Model	Test data	Class	Death rate	Total	Alive	Dead
Kmeans	False	0	50.1%	13439	6706	6733
Kmeans	False	1	47.17%	5118	2704	2414
Kmeans	False	2	62.73%	1033	385	648
Kmeans	True	0	1.3%	1697	1675	22
Kmeans	True	1	2.31%	693	677	16
Kmeans	True	2	4.08%	98	94	4

Model	Test data	Class	Death rate	Total	Alive	Dead
Mini Batch Kmeans	False	0	48.31%	4848	2506	2342
Mini Batch Kmeans	False	1	49.78%	13838	6949	6889
Mini Batch Kmeans	False	2	62.39%	904	340	564
Mini Batch Kmeans	True	0	2.17%	644	630	14
Mini Batch Kmeans	True	1	1.36%	1759	1735	24
Mini Batch Kmeans	True	2	4.71%	85	81	4

- We can easily notice that both models created new risk group with much higher risk than the other and the original data.
- It seems like a good direction, so we tested some more models, the following model works best form our perspective.



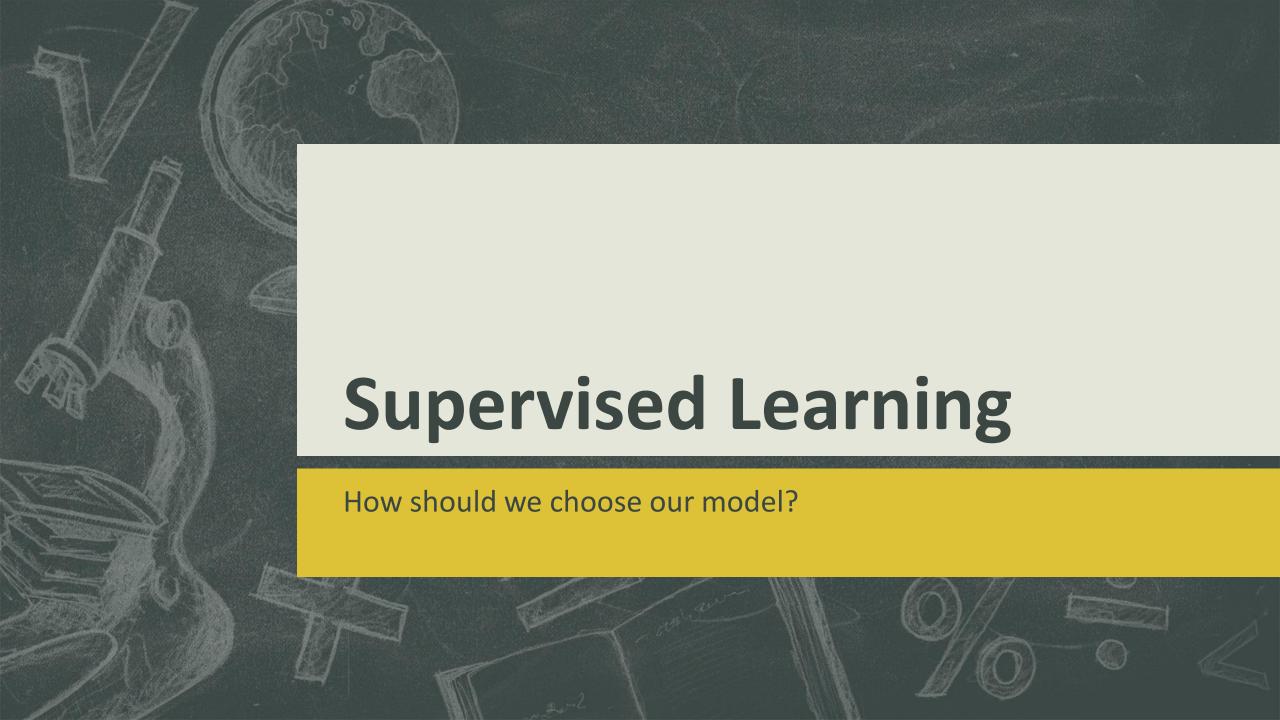
## Gaussian Mixture Model

Model	Test data	Class	Death rate	Total	Alive	Dead
GaussianMixture	False	0	48.83%	17250	8827	8423
GaussianMixture	False	1	72.7%	1883	514	1369
GaussianMixture	False	2	0.0%	451	451	0
GaussianMixture	True	0	1.37%	2265	2234	31
Gaussian Mixture	True	1	6.61%	121	113	8
GaussianMixture	True	2	0.0%	102	102	0

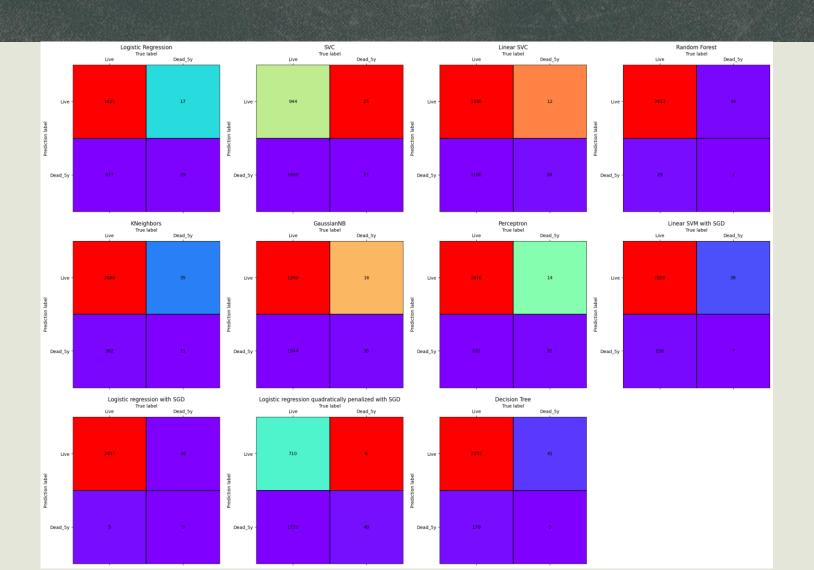
Model	Test data	Class	Death rate	Total	Alive	Dead
GaussianMixture	False	0	47.07%	16948	8971	7977
GaussianMixture	False	1	73.37%	1769	471	1298
Gaussian Mixture	False	2	59.48%	881	357	524
GaussianMixture	True	0	1.54%	2271	2236	35
Gaussian Mixture	True	1	8.26%	109	100	9
GaussianMixture	True	2	1.85%	108	106	2

- The model give us more accurate risk group dividing, we'll use the classes that GMM made and insert it into the data.
- We are trying to use weight for classes to affect the supervised model.





# Confusion Matrices & Related Rates



In an imbalanced dataset, one class typically has a much larger number of instances than the other class, which can lead to a bias towards the majority class.

When using traditional metrics such as accuracy, a model can still have a high accuracy score even if it does not perform well on the minority class. For example, a model that always predicts the majority class will have a high accuracy, even though it does not perform well on the minority class.

	Balanced Accura	асу	False Negative Rate	Accuracy	Precision	Error rate	Score
Model							
Logistic Regression	68.89	9%	36.96%	74.52%	4.49%	25.48%	74.52%
Perceptron	67.7	5%	30.43%	66.00%	3.70%	34.00%	66.00%
Multilayer Percptron	66.8	6%	39.13%	72.63%	4.05%	27.37%	72.63%
Linear SVC	64.3	1%	26.09%	55.06%	2.98%	44.94%	55.06%
GaussianNB	61.2	3%	34.78%	57.40%	2.79%	42.60%	57.40%
Logistic regression quadratically penalized with SGD	58.0	2%	13.04%	30.14%	2.26%	69.86%	30.14%
KNeighbors	54.5	4%	76.09%	84.04%	2.95%	15.96%	84.04%
Linear SVM with SGD	52.7	8%	84.78%	88.95%	2.88%	11.05%	88.95%
Decision Tree	51.9	5%	89.13%	91.52%	2.86%	8.48%	91.52%
Random Forest	51.5	8%	95.65%	97.07%	6.45%	2.93%	97.07%
Logistic regression with SGD	49.9	0%	100.00%	97.95%	0.00%	2.05%	97.95%
SVC	42.1	5%	54.35%	38.79%	1.38%	61.21%	38.79%

# **Model Picking**

- If we look from an unbiased point of view, we should use the model with the highest balanced accuracy, therefore we'll get the best result for any data.
- On the other hand, we are testing death prediction in 5 years, taking this into consider should maybe measure the False Negative Rate.

It can show how many dead in 5 years people the model "missed".

Balanced Accuracy	False Negative Rate	Accuracy	Precision	Error rate	Score
		_			
58.02%	13.04%	30.14%	2.26%	69.86%	30.14%
64.31%	26.09%	55.06%	2.98%	44.94%	55.06%
67.75%	30.43%	66.00%	3.70%	34.00%	66.00%
61.23%	34.78%	57.40%	2.79%	42.60%	57.40%
68.89%	36.96%	74.52%	4.49%	25.48%	74.52%
66.86%	39.13%	72.63%	4.05%	27.37%	72.63%
42.15%	54.35%	38.79%	1.38%	61.21%	38.79%
54.54%	76.09%	84.04%	2.95%	15.96%	84.04%
52.78%	84.78%	88.95%	2.88%	11.05%	88.95%
51.95%	89.13%	91.52%	2.86%	8.48%	91.52%
51.58%	95.65%	97.07%	6.45%	2.93%	97.07%
49.90%	100.00%	97.95%	0.00%	2.05%	97.95%
	58.02% 64.31% 67.75% 61.23% 68.89% 66.86% 42.15% 54.54% 52.78% 51.95% 51.58%	58.02%       13.04%         64.31%       26.09%         67.75%       30.43%         61.23%       34.78%         68.89%       36.96%         66.86%       39.13%         42.15%       54.35%         54.54%       76.09%         52.78%       84.78%         51.95%       89.13%         51.58%       95.65%	58.02%       13.04%       30.14%         64.31%       26.09%       55.06%         67.75%       30.43%       66.00%         61.23%       34.78%       57.40%         68.89%       36.96%       74.52%         66.86%       39.13%       72.63%         42.15%       54.35%       38.79%         54.54%       76.09%       84.04%         52.78%       84.78%       88.95%         51.95%       89.13%       91.52%         51.58%       95.65%       97.07%	58.02%       13.04%       30.14%       2.26%         64.31%       26.09%       55.06%       2.98%         67.75%       30.43%       66.00%       3.70%         61.23%       34.78%       57.40%       2.79%         68.89%       36.96%       74.52%       4.49%         66.86%       39.13%       72.63%       4.05%         42.15%       54.35%       38.79%       1.38%         54.54%       76.09%       84.04%       2.95%         52.78%       84.78%       88.95%       2.88%         51.95%       89.13%       91.52%       2.86%         51.58%       95.65%       97.07%       6.45%	58.02%       13.04%       30.14%       2.26%       69.86%         64.31%       26.09%       55.06%       2.98%       44.94%         67.75%       30.43%       66.00%       3.70%       34.00%         61.23%       34.78%       57.40%       2.79%       42.60%         68.89%       36.96%       74.52%       4.49%       25.48%         66.86%       39.13%       72.63%       4.05%       27.37%         42.15%       54.35%       38.79%       1.38%       61.21%         54.54%       76.09%       84.04%       2.95%       15.96%         52.78%       84.78%       88.95%       2.88%       11.05%         51.95%       89.13%       91.52%       2.86%       8.48%         51.58%       95.65%       97.07%       6.45%       2.93%

## Conclusions

- Imbalanced data provides a lot of difficulties that we must chose a road along the way, it can be a waste of time in some cases.
- In our case the we can notice the tradeoffs in the models and keep it in mind while considering what should we use.
- For future work we should create a new model that will optimize the following Min-Max problem:
  - ➤ Minimize the false negative rate
  - Maximize the balanced accuracy