BIOS 7747: Machine Learning for Biomedical Applications

Feature exploration, pre-processing and normalization

Antonio R. Porras (antonio.porras@cuanschutz.edu)

Department of Biostatistics and Informatics
Colorado School of Public Health
University of Colorado Anschutz Medical Campus

Outline

- Feature exploration and data cleaning
- Feature distributions
- Feature interactions
- Mitigating outlier effects and multicolinearity
- Feature scaling

Feature exploration and data cleaning

- Types of features:

<u>Numerical</u>

- Continuous
- Discrete

<u>Categorial</u>

- Ordinal
- Nominal

Feature exploration and data cleaning

❑ Encoding categorical variables

- Label encoding (`sklearn.preprocessing.LabelEncoder`)

Cancer stage	Encoded value
Stage I	0
Stage II	1
Stage III	2

Often not recommended

- Distances cannot be assumed in most ordinal variables
- Order cannot be assumed in nominal variables

Feature exploration and data cleaning

❑ Encoding categorical variables

- One hot encoding (`sklearn.preprocessing.OneHotEncoder`)

Sample	Cancer stage
0	Stage III
1	Stage I
2	Stage II

Sample	Stage I	Stage II	Stage III
0	0	0	1
1	1	0	0
2	0	1	0

It can increase substantially the number of features:

$$M = M_{numerical} + \sum_{\forall i \in categorical} N^{i}_{classes}$$

Feature exploration and data cleaning

❑ Encoding categorical variables

- Binary encoding: every feature is coded as a binary number with a fixed number of digits. Each digit is a feature to consider in the model.

Sample	Cancer stage
0	Stage III
1	Stage I
2	Stage II

Sample	Feature 1	Feature 2
0	1	0
1	0	0
2	0	1

It requires less features than one-hot-encoding

$$M = M_{numerical} + \sum_{\forall i \in categorical} \log_2\left(N^i_{classes}\right)$$

Feature exploration and data cleaning

❑ Encoding categorical variables

- Feature hashing: Convert labels to “words” with predefined fixed size

1	A,B,C,D,E,F,G
2	H,I,J,K,L,M,N
3	O,P,Q,R,S,T
4	U,V,W,X,Y,Z

	1	2	3	4
Mary →	1	1	1	1
John →	0	3	1	0
Jennifer →	3	4	1	0

Great to standardize representations using low number of variables

Feature collision may happen

Feature exploration and data cleaning

- Missing values

  - Common reasons:
    - Incomplete data (e.g., prefer not to answer, data transfer errors)
    - Human error (e.g., forgot to annotate, incorrect annotations)
    - Study design (e.g., data does not apply)

  - Classification [D.B. Rubin, 1976]
    - Missing completely at random (MCAR): missing values are not related to the observations (the probability of having missing value is equation for all samples). Unusual in biomedicine.
    - Missing at random (MAR): the probability of missing value is a function of another variable (e.g., male are less likely to answer about mental health questions in a survey).
    - Missing not at random (MNAR): there is no insight about the probability of missing data.

Feature exploration and data cleaning

- Missing values
  - **Sample dropping** (Pandas' *dropna()* function):
    - MCAR: it may not affect predictions when "sufficient" data are available
      - But it may result in insufficient data
    - MAR: it can introduce biases that could potentially be identified
      - Example: male do not have mental illness problems
    - MNAR: it can introduce biases that are hard to identify

Feature exploration and data cleaning

- Missing values
  - **Data imputation** (*sklearn.impute*)
    - Normal imputation: most likely value assuming a normal distribution
      - Numerical data: mean / median value
      - Categorial data: mode

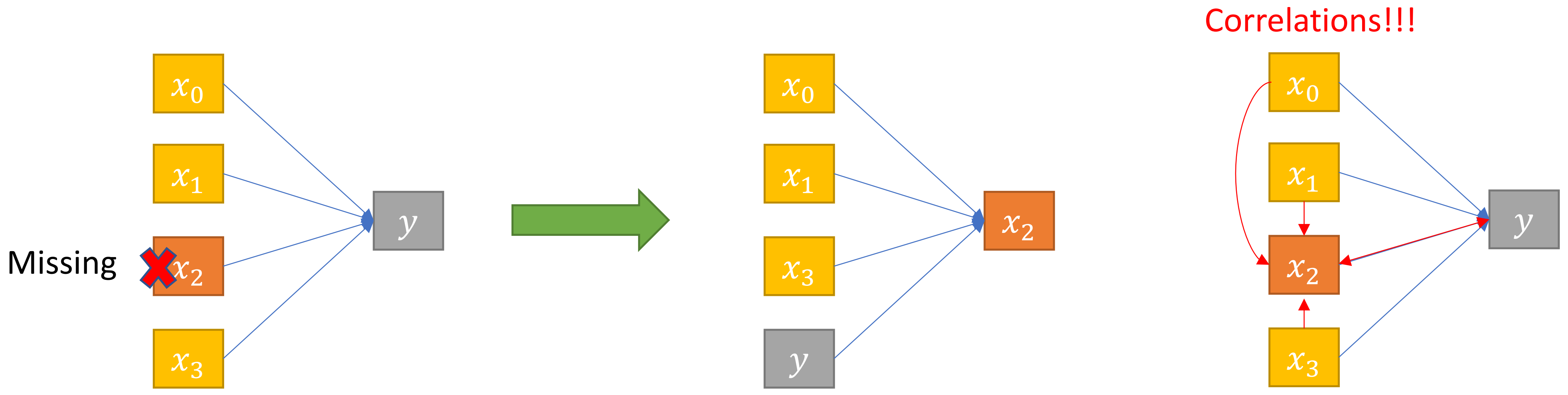
missing feature values

    - Class label imputation: normal imputation using same-class samples
    - Model-based imputation: two-step approach
      1. Train a model to predict the missing value using samples with non-missing values
      2. Predict the missing values

Feature exploration and data cleaning

❑ Missing values

- **Data imputation**
  - <u>Model-based imputation</u> (ext.): regression



Feature exploration and data cleaning

- Missing values

  - **Data imputation**

    - Model-based imputation (ext.):

      - Multiple imputation by chained equations (MICE)

        1. Perform a simple imputation (e.g., mean, median, mode)

        2. For each cycle *c*:
           - For each variable *v* with missing values (ascending order based on number of missing values):
             2.a. Use all variables except for *v* to predict the dependent missing values *y*
             2.b. Update previously missing values for *v* using a model trained with all other variables

        3. Evaluate convergence criterion (e.g., number of iterations, convergence of variable distributions…)

      - K-Nearest Neighbors and other clustering techniques

Feature exploration and data cleaning

❑ Missing values

- **Incorporating missing values in model**
  - Adding a present/missing value feature k

$$y = \theta_0 + \theta_1 x_0 + \theta_2 x_1 + k\theta_2 x_2 + \theta_3 x_3$$

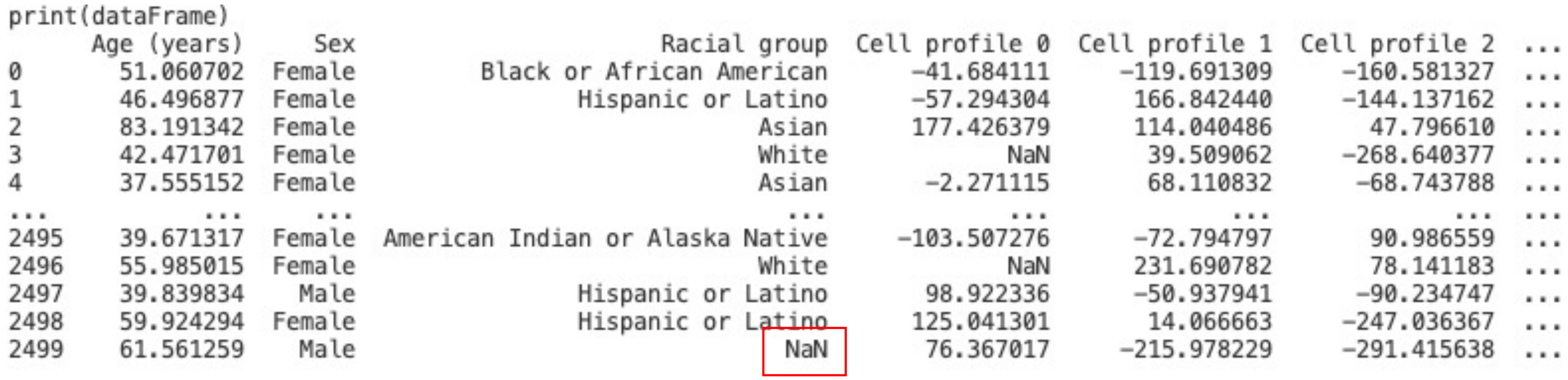
  - The zero contribution to the gradient from missing parameters in a group of samples may bias the optimization algorithm

Note: missing values may not be missing

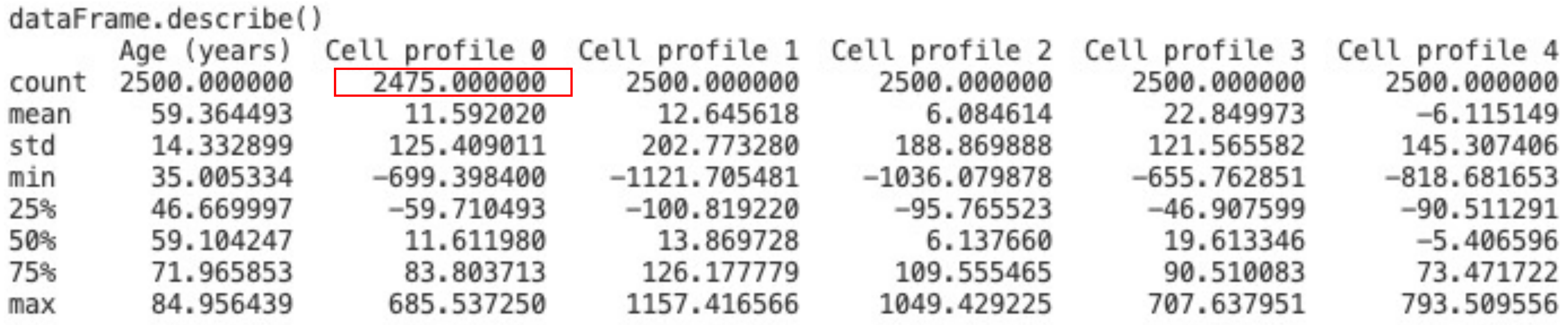
- Datasets often present values of 0 or default value as n/a or empty.
- Comprehensive visual data exploration is important before automated analysis.

Feature exploration and data cleaning

- Data exploration (with Pandas): data summary



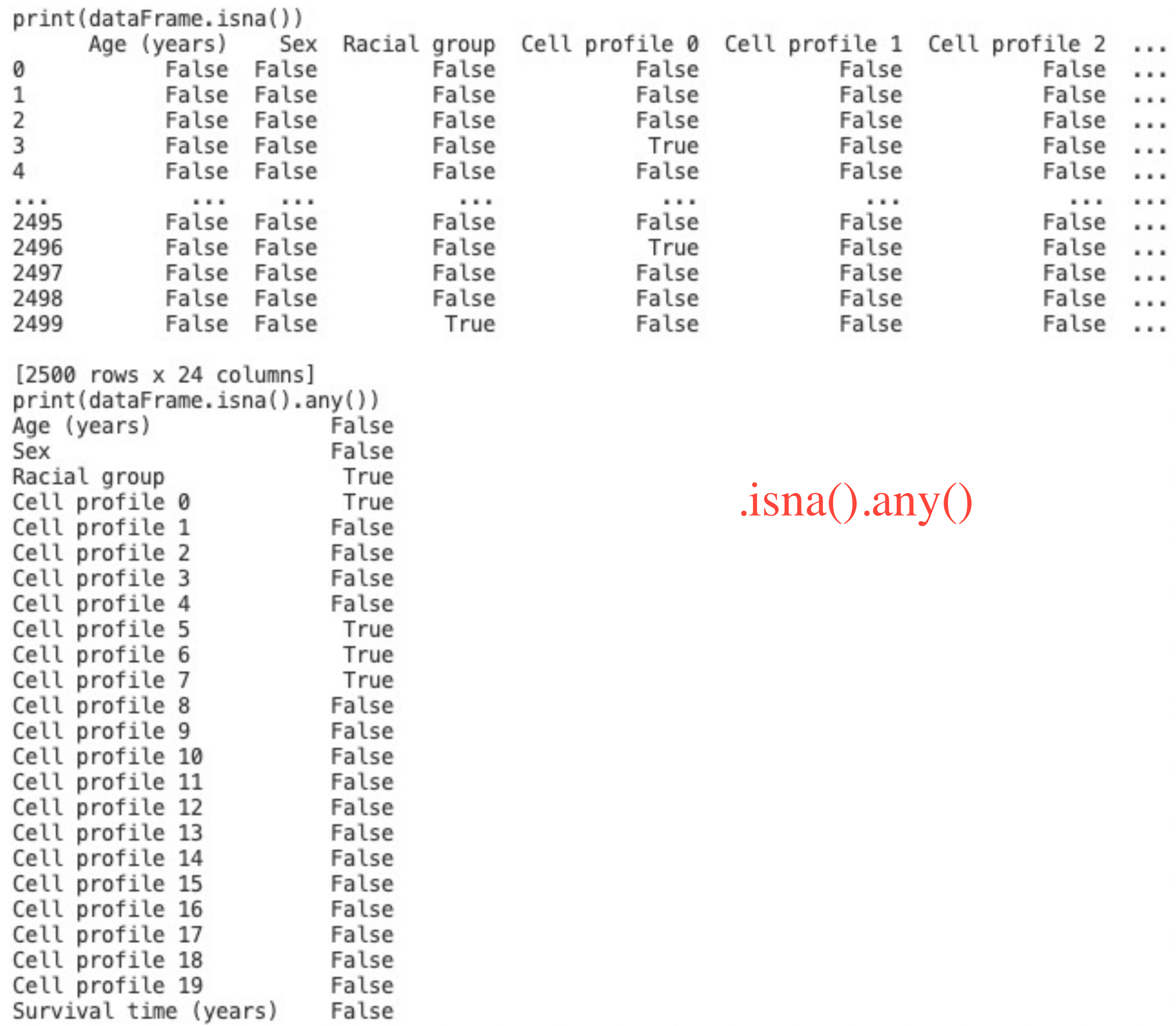
```
print(dataFrame)
       Age (years)     Sex                        Racial group  Cell profile 0  Cell profile 1  Cell profile 2  ...
0        51.060702  Female           Black or African American      -41.684111     -119.691309     -160.581327  ...
1        46.496877  Female                  Hispanic or Latino      -57.294304      166.842440     -144.137162  ...
2        83.191342  Female                               Asian      177.426379      114.040486       47.796610  ...
3        42.471701  Female                               White             NaN       39.509062     -268.640377  ...
4        37.555152  Female                               Asian       -2.271115       68.110832      -68.743788  ...
...            ...     ...                                 ...             ...             ...             ...  ...
2495     39.671317  Female  American Indian or Alaska Native     -103.507276      -72.794797       90.986559  ...
2496     55.985015  Female                               White             NaN      231.690782       78.141183  ...
2497     39.839834    Male                  Hispanic or Latino       98.922336      -50.937941      -90.234747  ...
2498     59.924294  Female                  Hispanic or Latino      125.041301       14.066663     -247.036367  ...
2499     61.561259    Male                                 NaN       76.367017     -215.978229     -291.415638  ...
```



```
dataFrame.describe()
       Age (years)  Cell profile 0  Cell profile 1  Cell profile 2  Cell profile 3  Cell profile 4
count  2500.000000     2475.000000     2500.000000     2500.000000     2500.000000     2500.000000
mean     59.364493       11.592020       12.645618        6.084614       22.849973       -6.115149
std      14.332899      125.409011      202.773280      188.869888      121.565582      145.307406
min      35.005334     -699.398400    -1121.705481    -1036.079878     -655.762851     -818.681653
25%      46.669997      -59.710493     -100.819220      -95.765523      -46.907599      -90.511291
50%      59.104247       11.611980       13.869728        6.137660       19.613346       -5.406596
75%      71.965853       83.803713      126.177779      109.555465       90.510083       73.471722
max      84.956439      685.537250     1157.416566     1049.429225      707.637951      793.509556
```

Feature exploration and data cleaning

❑ Data exploration (with Pandas): missing values



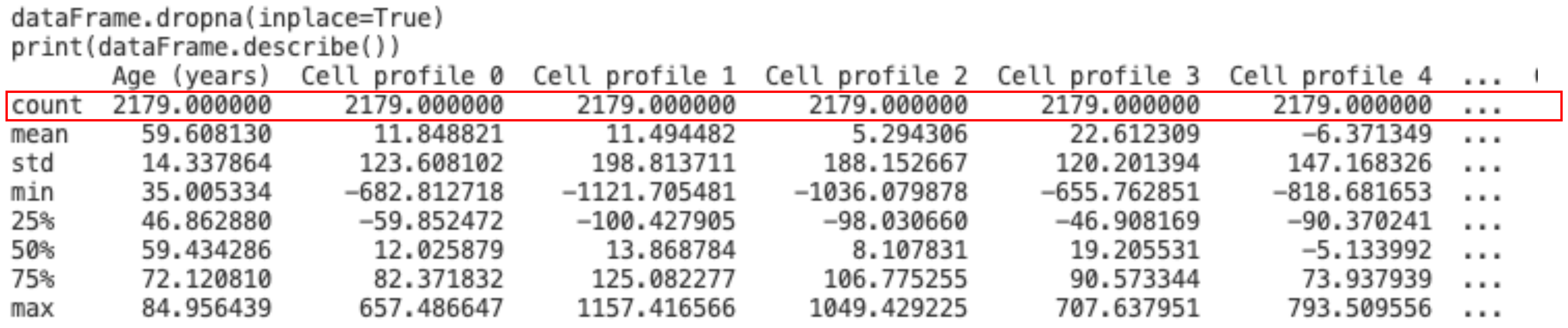
```
print(dataFrame.isna())
       Age (years)    Sex  Racial group  Cell profile 0  Cell profile 1  Cell profile 2  ...
0            False  False         False           False           False           False  ...
1            False  False         False           False           False           False  ...
2            False  False         False           False           False           False  ...
3            False  False         False            True           False           False  ...
4            False  False         False           False           False           False  ...
...            ...    ...           ...             ...             ...             ...  ...
2495         False  False         False           False           False           False  ...
2496         False  False         False            True           False           False  ...
2497         False  False         False           False           False           False  ...
2498         False  False         False           False           False           False  ...
2499         False  False          True           False           False           False  ...

[2500 rows x 24 columns]
print(dataFrame.isna().any())
Age (years)           False
Sex                   False
Racial group           True
Cell profile 0         True
Cell profile 1        False
Cell profile 2        False
Cell profile 3        False
Cell profile 4        False
Cell profile 5         True
Cell profile 6         True
Cell profile 7         True
Cell profile 8        False
Cell profile 9        False
Cell profile 10       False
Cell profile 11       False
Cell profile 12       False
Cell profile 13       False
Cell profile 14       False
Cell profile 15       False
Cell profile 16       False
Cell profile 17       False
Cell profile 18       False
Cell profile 19       False
Survival time (years)  False
```

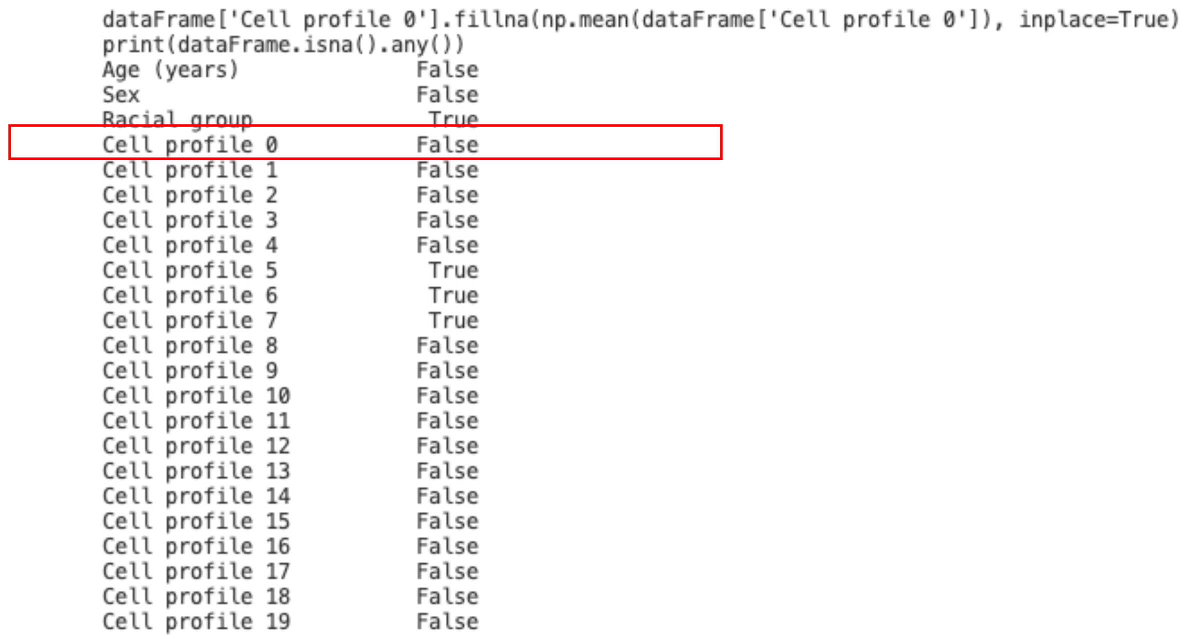
.isna().any()

Feature exploration and data cleaning

- Data exploration (with Pandas): missing values



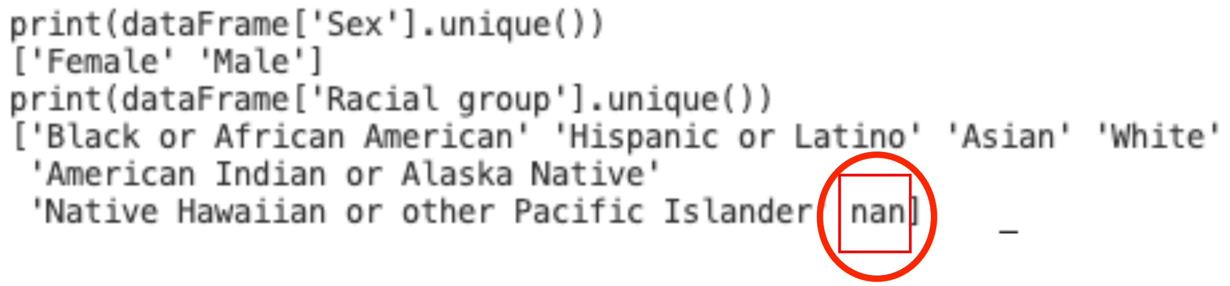
```
dataFrame.dropna(inplace=True)
print(dataFrame.describe())
       Age (years)  Cell profile 0  Cell profile 1  Cell profile 2  Cell profile 3  Cell profile 4  ...
count  2179.000000     2179.000000     2179.000000     2179.000000     2179.000000     2179.000000  ...
mean     59.608130       11.848821       11.494482        5.294306       22.612309       -6.371349  ...
std      14.337864      123.608102      198.813711      188.152667      120.201394      147.168326  ...
min      35.005334     -682.812718    -1121.705481    -1036.079878     -655.762851     -818.681653  ...
25%      46.862880      -59.852472     -100.427905      -98.030660      -46.908169      -90.370241  ...
50%      59.434286       12.025879       13.868784        8.107831       19.205531       -5.133992  ...
75%      72.120810       82.371832      125.082277      106.775255       90.573344       73.937939  ...
max      84.956439      657.486647     1157.416566     1049.429225      707.637951      793.509556  ...
```



```
dataFrame['Cell profile 0'].fillna(np.mean(dataFrame['Cell profile 0']), inplace=True)
print(dataFrame.isna().any())
Age (years)                 False
Sex                         False
Racial group                 True
Cell profile 0              False
Cell profile 1              False
Cell profile 2              False
Cell profile 3              False
Cell profile 4              False
Cell profile 5               True
Cell profile 6               True
Cell profile 7               True
Cell profile 8              False
Cell profile 9              False
Cell profile 10             False
Cell profile 11             False
Cell profile 12             False
Cell profile 13             False
Cell profile 14             False
Cell profile 15             False
Cell profile 16             False
Cell profile 17             False
Cell profile 18             False
Cell profile 19             False
```

Feature exploration and data cleaning

- Data exploration (with Pandas): categorial variables



```
print(dataFrame['Sex'].unique())
['Female' 'Male']
print(dataFrame['Racial group'].unique())
['Black or African American' 'Hispanic or Latino' 'Asian' 'White'
 'American Indian or Alaska Native'
 'Native Hawaiian or other Pacific Islander' nan]
```

Feature distributions

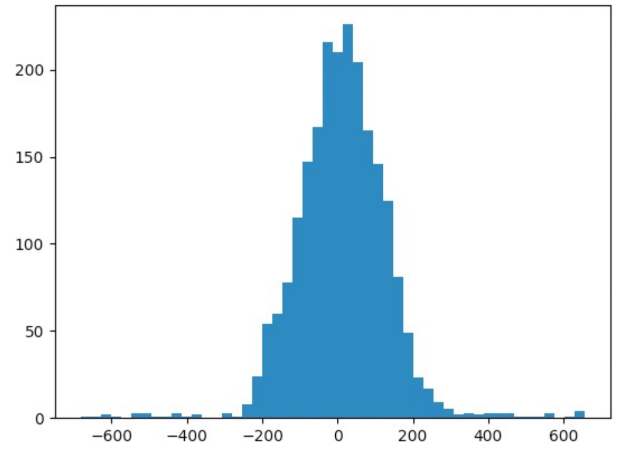
Information Visualization (BIOS 7719)

❑ Numerical features

- Quantitative values: mean, median, IQR, standard deviation, range… (numpy)

- Visualizations (matplotlib.pyplot as plt):

```
plt.hist(dataFrame['Cell profile 0'], bins=50)
plt.show()
```



- Testing normality:
  - Shapiro-Wilk test (scipy.stats) or other available tests

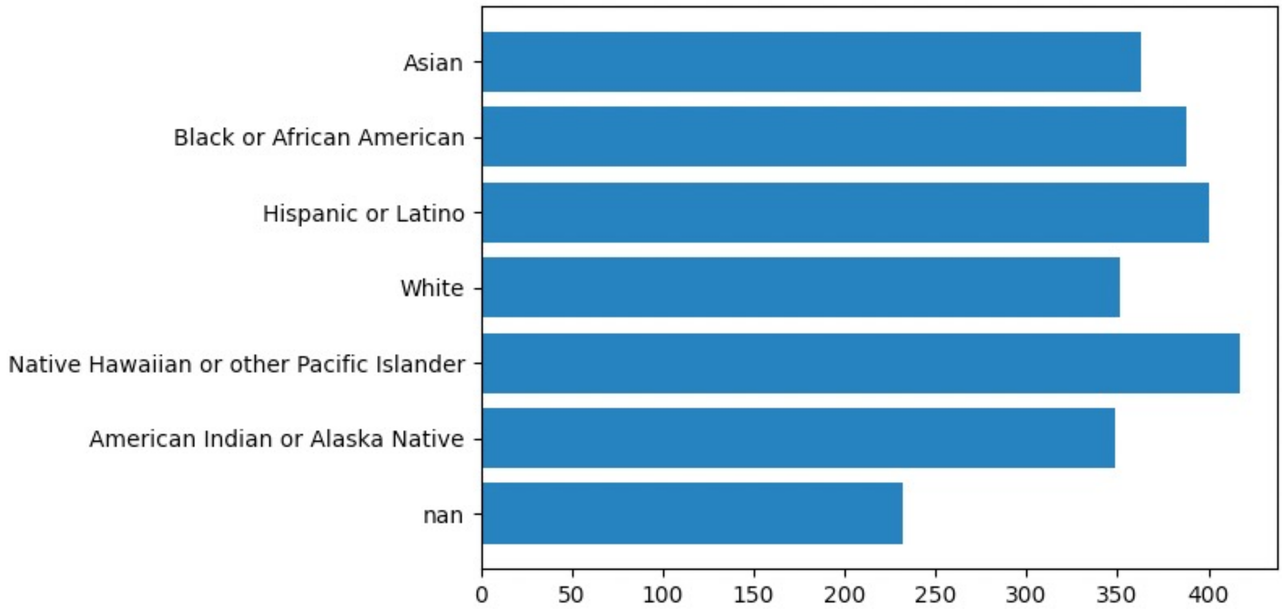
```
print(scipy.stats.shapiro(dataFrame['Cell profile 0']))
ShapiroResult(statistic=0.9458088278770447, pvalue=1.3563603337555129e-27)
```

  - Often data are only normally distributed for specific subgroups of samples

Feature distributions

- Categorical features

```
raceLabels = list(set(dataFrame['Racial group']))
raceCounts = [list(dataFrame['Racial group']).count(c) for c in raceLabels]
raceLabels[raceLabels==np.nan] = 'nan' # Converting to string
plt.barh(raceLabels, raceCounts)
plt.show()
```



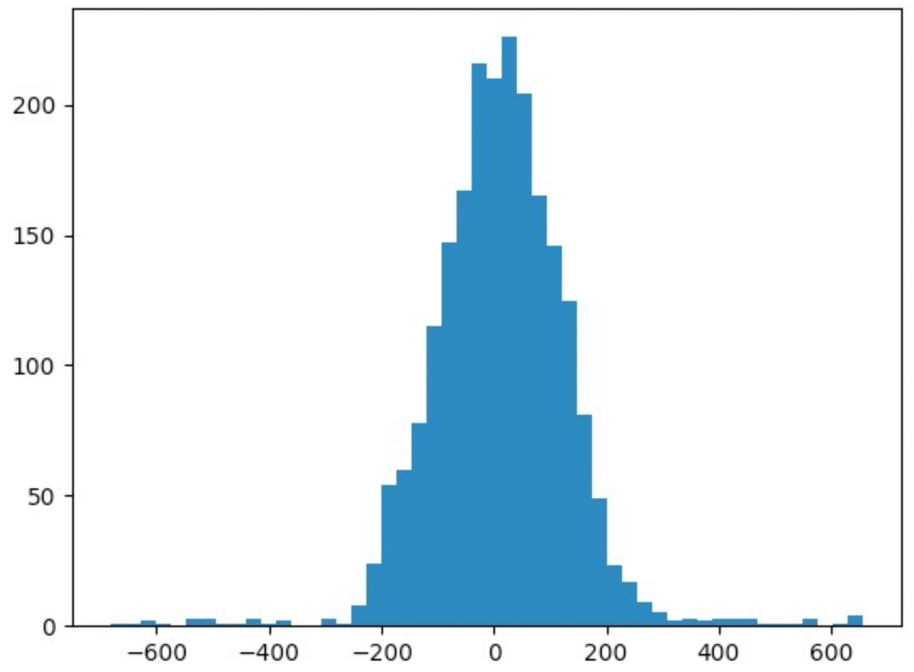
Feature distributions

- Outliers
  - Observations that do not follow the overall patterns in the population
  - They can bias the training process and lead towards suboptimal models
  - Types:
    - Natural: realistic/plausible observations that are uncommon
    - Error:
      - Data entry (human)
      - Measurement (instrument)
      - Experimental (extraction or execution)
      - Sampling (source of information)

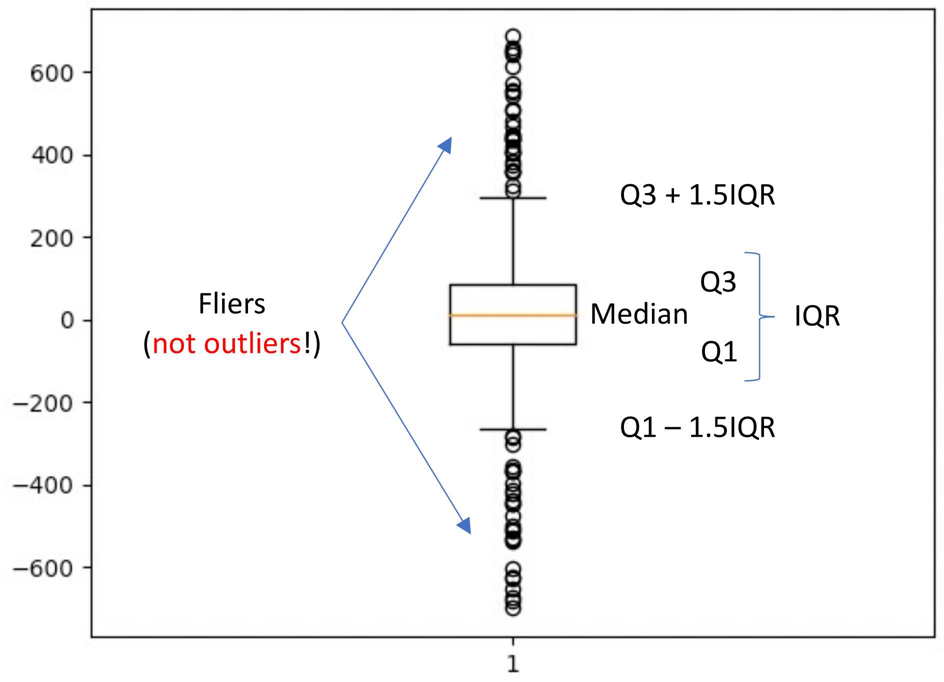
Feature distributions

❑ Outliers

- First step for outlier identification is visualization



```
plt.hist(dataFrame['Cell profile 0'], bins=50)
plt.show()
```



```
plt.boxplot(dataFrame['Cell profile 0’])
plt.show()
```

Feature distributions

❑ Identifying outliers

- *z*-score: assumes normal data distributions. A threshold must be established, usually ≥ 3.

$$\mathrm{z} = \frac{x - \mu}{\sigma}$$

- Thomson's Tau test: Rejection interval: $\frac{t_{\alpha/2}(n-1)}{\sqrt{n}\sqrt{n-2+t^2_{\alpha/2}}}$

- Tukey's "fences": $[Q_1 - k * (Q_3 - Q_1), Q_3 + k * (Q_3 - Q_1)]$
  - For $k = 1.5$: outliers. For 3: extreme values.

- Clustering methods: e.g., KNN, DBScan

[Martin et al, A density-based algorithm for discovering clusters in large spatial databases with noise. AAAI Press. pp. 226–231, 1996]

Feature distributions

- Outliers

  - Distinction between natural and error outliers in biomedicine is very important

  - Underrepresented cases often appear as natural outliers of the normative population
    - There is a high risk of creating biased methods that discriminate specific populations

  - Outlier treatment:

    - Natural outliers: highly encouraged to consider in model training. Consider creating separate models.

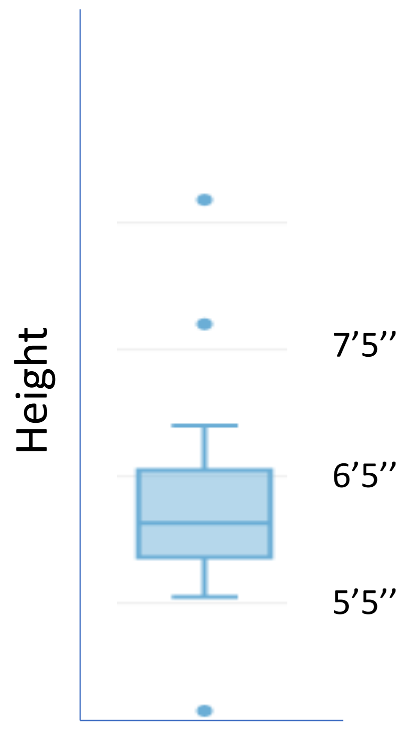
    - Error outliers:

      - Remove samples

      - Use data imputation techniques

Feature distributions

- Outliers
  - There are no robust methods to identify the outlier type and ensure a model free of biases
    - Understanding the dataset is essential



College students?

Basketball team?

Feature interactions

❑ Feature interactions

- Multicollinearity: when an “independent” feature can be predicted to a degree from other independent features.

- Why is it a problem?

  - Linear regression: $\boldsymbol{X} = \begin{pmatrix} 1 & x_0^{(0)} & \dots & x_{M-1}^{(0)} \\ \vdots & & & \vdots \\ 1 & x_0^{(N-1)} & \dots & x_{M-1}^{(N-1)} \end{pmatrix}$ If columns are correlated, the rank of $\boldsymbol{X}$ is lower than $\boldsymbol{M} + \mathbf{1}$. Hence, $\boldsymbol{X}\boldsymbol{X}^T$ does not have an inverse.

  - Individual relationships between the dependent and independent variables cannot be recovered.

$f(x_0, x_1) = \alpha x_0 + \beta x_1$ and $x_0 = \gamma x_1$, then: $f(x_0, x_1) = \alpha x_0 + \beta x_1 = \frac{1}{N} \alpha x_0 + N \beta x_1$

Feature interactions

❑ Detection:

- Coefficients associated with correlated variables usually have high standard errors

- No significant contribution (or extreme contribution) of one variable to the regression model may indicate a collinearity (note that it may also mean lack of correlation or extreme correlation with predicted variable)

- Variance inflation factor for predictor j : $\mathrm{VIF_j} = \frac{1}{1-R_j^2}$
  - A value over 5 may indicate collinearity

[O'Brien, R. M. (2007). "A Caution Regarding Rules of Thumb for Variance Inflation Factors". *Quality & Quantity*. **41** (5): 673–690]

- High condition number
  - Two or more features with similarly high variance may suggest which are the correlated predictors.

- Correlation matrix:
  - It can only evaluate pair-wise relationships and multicollinearity often involves several features.

Feature interactions

- What to do?
  - Multicollinearity does not necessarily bias the predictions but their explanation (the contribution from the colinear variables).
  - Feature dropping: may cause loss of information (lower predictive accuracy) in exchange for more significant coefficient values.
  - Data transformations:
    - Transform the data to a new space where features are uncorrelated.
    - Example: principal component regression:
      - We will see PCA later in the course

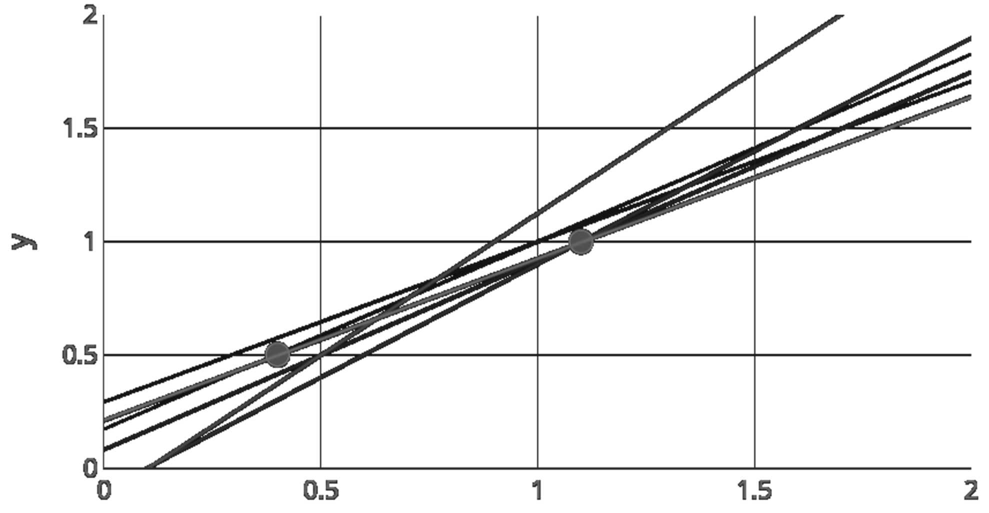
Mitigating multicollinearity and outlier values

- Regularization:
  - Can reduce the effect of both outliers and multicollinearity
  - L2-regularization (aka Ridge)

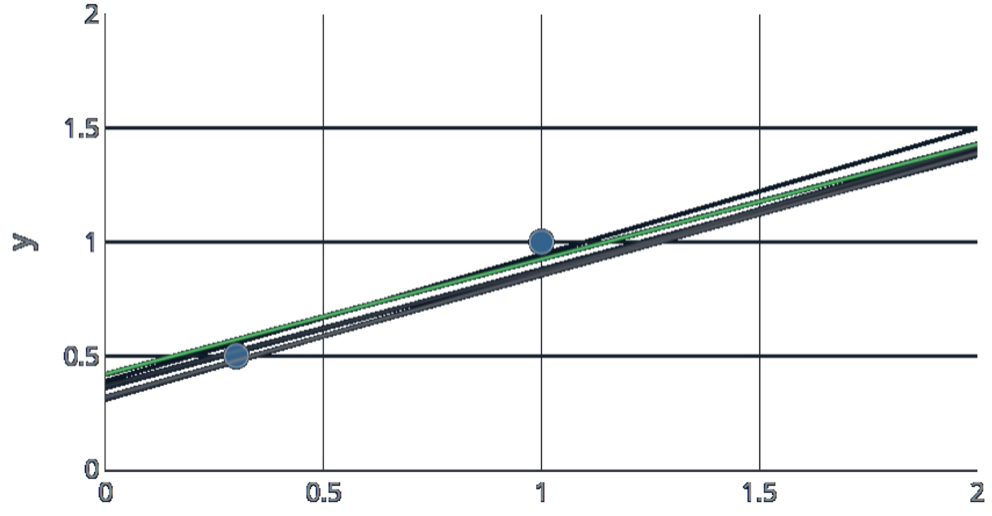
$$\frac{\partial}{\partial \boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{\partial}{\partial \boldsymbol{\theta}} \frac{1}{2}\left((\boldsymbol{X\theta} - \boldsymbol{y})^T(\boldsymbol{X\theta} - \boldsymbol{y}) + \alpha\theta^T\theta\right)$$

$$\boldsymbol{\theta} = \left(\boldsymbol{X}^T\boldsymbol{X} + \alpha I_M\right)^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

Promotes low parameter values, which tends to prevent extreme effects from independent variables



Without regularization



With L2-regularization

Mitigating multicollinearity and outlier values

❑ Regularization:

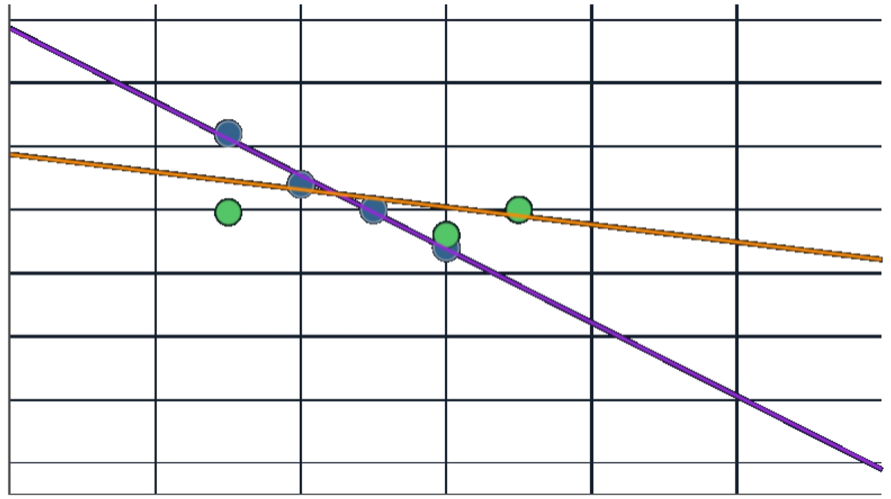
- L1-regularization (aka Lasso)
  - Its origin comes from the introduction of a **soft** threshold to parameters estimates in linear regression

$$\boldsymbol{\theta} = \arg\min \left\{ \frac{1}{2} (\boldsymbol{X\theta} - \boldsymbol{y})^T (\boldsymbol{X\theta} - \boldsymbol{y}) \right\} \quad \text{s.t.} \; \|\boldsymbol{\theta}\|_1 \leq t$$

$$\Downarrow$$

$$\boldsymbol{\theta} = \arg\min \left\{ \frac{1}{2} (\boldsymbol{X\theta} - \boldsymbol{y})^T (\boldsymbol{X\theta} - \boldsymbol{y}) + \alpha \|\boldsymbol{\theta}\|_1 \right\}$$

Lagrangian form (we will see Lagrange multipliers later in the course)



Circles:
- Purple: training
- Green: test

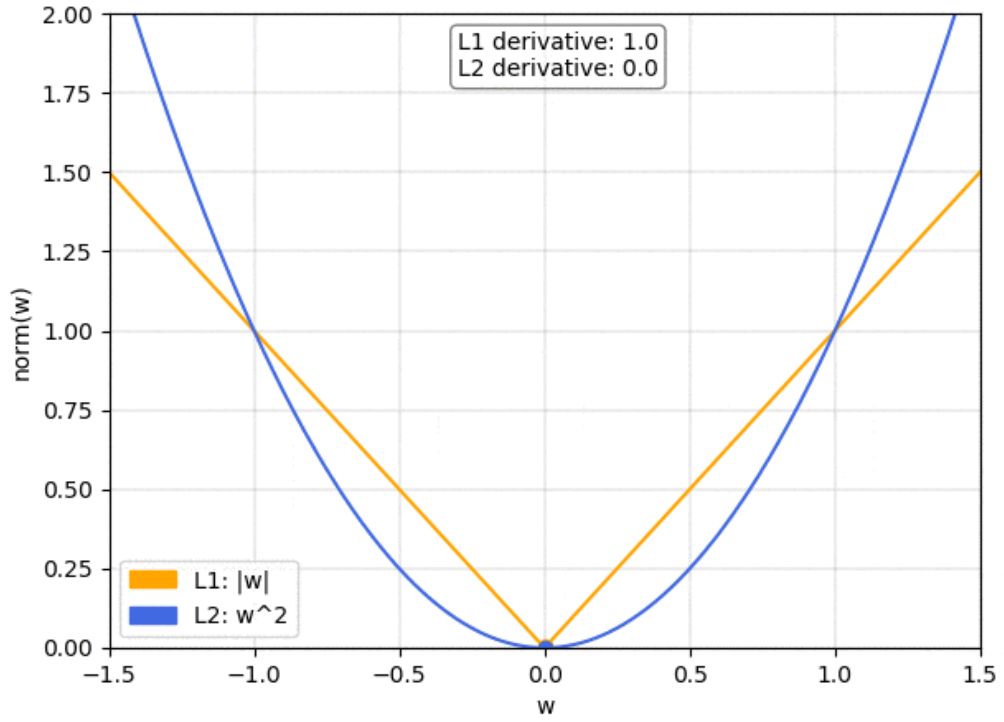
Lines:
- Purple: MSE
- Orange: L1

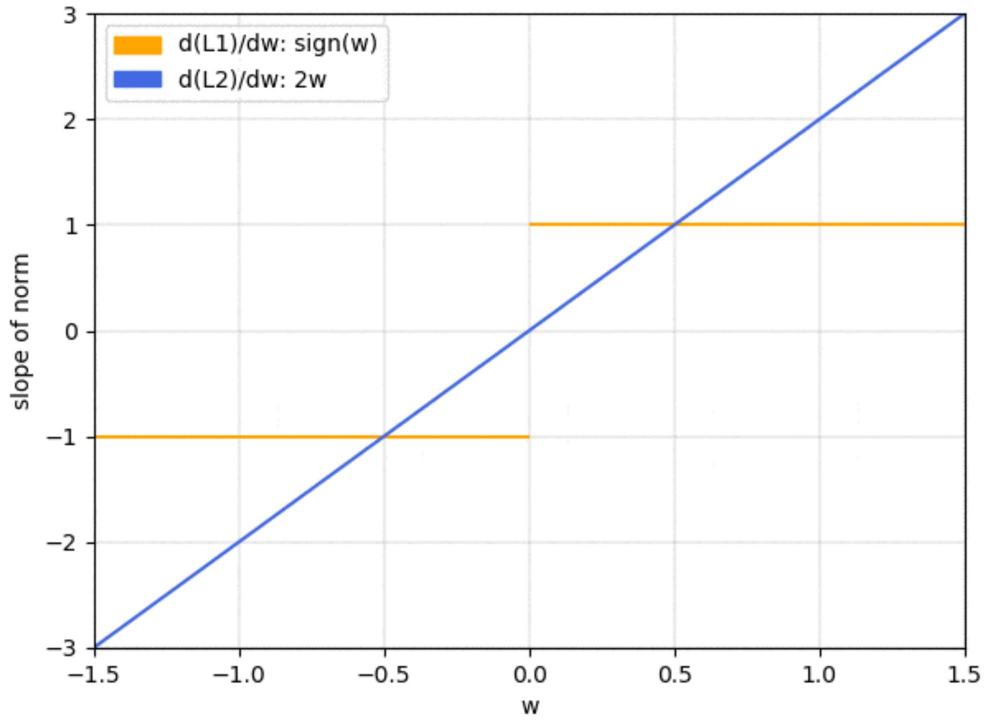
Mitigating multicollinearity and outlier values

- Regularization
  - Gradients: L2 vs. L1 regularization term

L2 $$\frac{1}{2}\alpha \frac{\partial \sum \theta_i^2}{\partial \theta_j} = \alpha\theta_j$$

L1 $$\alpha \frac{\partial \sum |\theta_i|}{\partial \theta_j} = \alpha \operatorname{sign}(\theta_j)$$





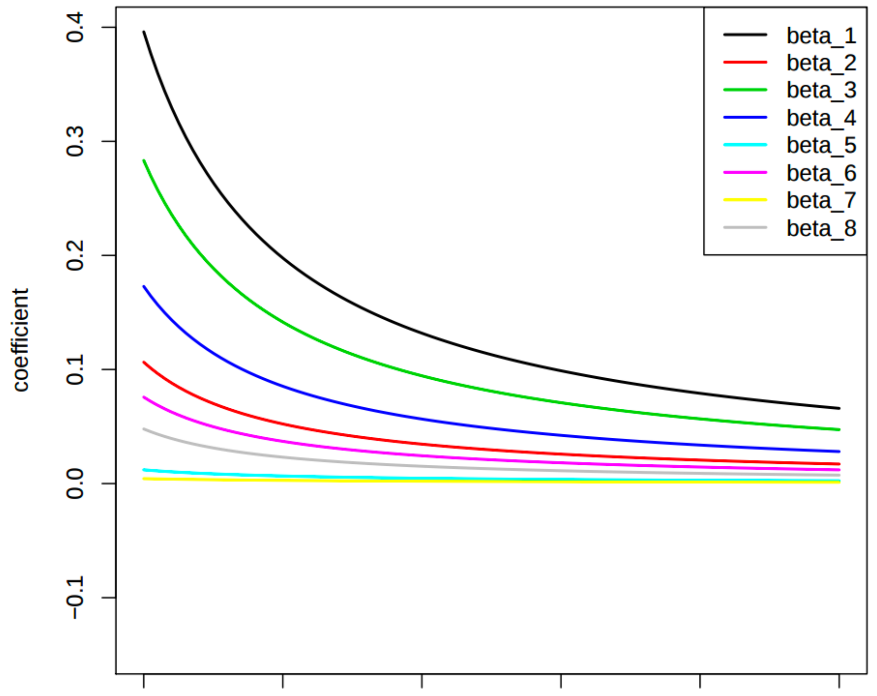
Mitigating multicollinearity and outlier values

- Regularization
  - L2: the gradient depends on the value of each coefficient
    - Coefficients with higher values will provide gradients with higher magnitude.
    - Gradient descent optimization will prioritize decreasing the value of coefficients with higher values
  - L1: the gradient is constant and the same for all coefficients
    - Coefficients with lower values tend to be zeroed first
    - Zeroing coefficients is equivalent to feature selection… but coefficients with lower magnitude will tend to be eliminated first.

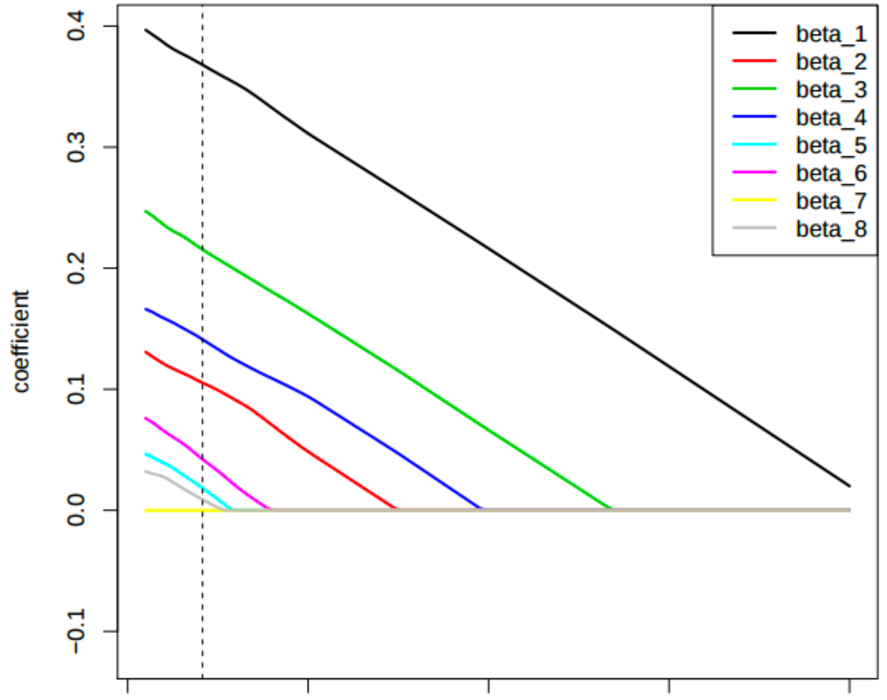
Mitigating multicollinearity and outlier values

- Regularization
  - Gradients: L2 vs. L1 regularization term

L2



L1



Feature scaling

- Feature scaling: converting features values to present specific ranges or distributions

- Why?
  - Parameter updates: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \boldsymbol{\delta\theta}_t = \boldsymbol{\theta}_t - \boldsymbol{\alpha}\frac{\partial}{\partial \boldsymbol{\theta}_t}\boxed{\boldsymbol{J}(\boldsymbol{\theta}_t)}$
  - MSE:

$$\theta_0 \leftarrow \theta_0 - \alpha \sum_{i=0}^{N-1}\left(f\left(\boldsymbol{x}^{(i)};\boldsymbol{\theta}\right) - y^{(i)}\right)$$

$$\theta_1 \leftarrow \theta_1 - \alpha \sum_{i=0}^{N-1}\left(f\left(\boldsymbol{x}^{(i)};\boldsymbol{\theta}\right) - y^{(i)}\right)\boldsymbol{x}_0^{(i)} \longrightarrow$$

$$\theta_2 \leftarrow \theta_2 - \alpha \sum_{i=0}^{N-1}\left(f\left(\boldsymbol{x}^{(i)};\boldsymbol{\theta}\right) - y^{(i)}\right)\boldsymbol{x}_1^{(i)} \longrightarrow$$

Different magnitudes mean different rates of change

  - Features with higher magnitude tend to bias optimization algorithms (not only regression models) towards solutions that prioritize the adjustment of model parameters associated with them

Feature scaling

- Feature normalization (or scaling normalization): converts features to specific ranges

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

`sklearn.preprocessing.`**`MinMaxScaler`**

- Feature standardization (or z-score normalization): converts features to have zero-mean and unit standard deviation

$$x_{scaled} = \frac{x - \bar{x}}{\sigma_x}$$

`sklearn.preprocessing.`**`StandardScaler`**

Feature scaling

- Normalization vs. standardization:
  - Homogeneous ranges vs. homogeneous variance across features
  - Normalized scales vs. normalized distributions
  - Unknown distributions vs. assumed Gaussian distributions
  - Bounded vs. unbounded
  - Highly sensitive vs. robust to outliers
- In general, standardization is preferred on methods that assume normal distributions (e.g., Lasso regression), while normalization is preferred when features present uniform distributions and datasets don’t present extreme outliers (e.g., KNN)

Takeaway points

- First, explore your dataset visually and try to understand it as much as possible
- Proper data encoding and management of missing values and outliers has a significant impact in any machine learning models
- Exploring automated data summaries (e.g., using Pandas) and plotting feature distributions is the first step to identify if certain assumptions can be made
- Try to remove feature dependencies when possible. Slightly lower accuracy may be preferred to improve confidence and reproducibility
- If the dataset may be underpowered or there is evidence of partial multicollinearity and/or possible outliers, consider model regularization
- Normalize or standardize your data