Data Science In Practice - Feature engineering

EPF - P2023
Yassine Idrissi - yassine.idrissi@teads.com
Ismail Chafai - ismail.chafai@teads.com

ÉCOLE D'INGÉNIEUR·E·S

IV. Data Science in practice: Feature engineering

1. Introduction
2. Handling numerical features
3. Handling categorical features
4. Using interaction features
5. Feature selection

Introduction

Introduction

- Machine Learning is the process of **fitting** mathematical **models** to **data** in order to derive **insights** or make **predictions**

Introduction

- Machine Learning is the process of **fitting** mathematical **models** to **data** in order to derive **insights** or make **predictions**

- The models take **features** as input, a feature is a **representation** of an aspect of **raw data**

Introduction

- Machine Learning is the process of **fitting** mathematical **models** to **data** in order to derive **insights** or make **predictions**

- The models take **features** as input, a feature is a **representation** of an aspect of **raw data**

- **Feature engineering** is the process of **extracting** features from raw data and transforming them into formats that are best suitable for the **model**

Introduction

- Machine Learning is the process of **fitting** mathematical **models** to **data** in order to derive **insights** or make **predictions**

- The models take **features** as input, a feature is a **representation** of an aspect of **raw data**

- **Feature engineering** is the process of **extracting** features from raw data and transforming them into formats that are best suitable for the **model**

- It’s a very important step in the **ML pipeline**, it enables the model to output predictions of a **better quality**

Introduction

- Example: We are trying to predict the price of a house given its length L and its width W, using a linear model

$$Price = \beta_0 + \beta_1 . L + \beta_2 . W$$

Introduction

- Example: We are trying to predict the price of a house given its length L and its width W, using a linear model

$$Price = \beta_0 + \beta_1 . L + \beta_2 . W$$

- This model would not lead to optimal performances, as we know for a fact that what matters for a house price is its area. By using this information, the model would yield better performances

Introduction

- Example: We are trying to predict the price of a house given its length L and its width W, using a linear model

$$Price = \beta_0 + \beta_1 . L + \beta_2 . W$$

- This model would not lead to optimal performances, as we know for a fact that what matters for a house price is its area. By using this information, the model would yield better performances

$$Price = \beta_0 + \beta_1 . L + \beta_2 . W + \beta_3 . A$$

$$with \quad A = L . W$$

Introduction

- The vast **majority** of time in building a ML pipeline is spent on **data preprocessing** and **feature engineering**

Introduction

- The vast **majority** of time in building a ML pipeline is spent on **data preprocessing** and **feature engineering**

- This has been found to be **the main driver** of model **performance** in many applications

Introduction

- The vast **majority** of time in building a ML pipeline is spent on **data preprocessing** and **feature engineering**

- This has been found to be **the main driver** of model **performance** in many applications

- This process is very **specific** to the **data** at hand and the **model** we are using. It’s quite hard to generalize it across projects

Introduction

- The vast **majority** of time in building a ML pipeline is spent on **data preprocessing** and **feature engineering**

- This has been found to be **the main driver** of model **performance** in many applications

- This process is very **specific** to the **data** at hand and the **model** we are using. It’s quite hard to generalize it across projects

- We will try to cover some of the most common feature engineering techniques

Introduction

- Let’s define some basic concepts

Introduction

- Let’s define some basic concepts
  - **Data** are observations of real-world phenomena (eg: observations of daily stock prices, customer purchase history, weather measurements, …)

Introduction

- Let's define some basic concepts
  - **Data** are observations of real-world phenomena (eg: observations of daily stock prices, customer purchase history, weather measurements, …)
  - A **Task** is a goal we want to achieve or a question we want to answer (eg: which stock options should I buy? How can I better serve my customers? What are the odds of snowing tomorrow?…)

Introduction

- Let's define some basic concepts
  - **Data** are observations of real-world phenomena (eg: observations of daily stock prices, customer purchase history, weather measurements, …)
  - A **Task** is a goal we want to achieve or a question we want to answer (eg: which stock options should I buy? How can I better serve my customers? What are the odds of snowing tomorrow?…)
  - A **Mathematical Model** of data describes the relationships between different aspects of data (eg: what's the mathematic formula that relates a company's earning history and past stock prices to the predicted stock price)

Introduction

- Let's define some basic concepts
  - **Data** are observations of real-world phenomena (eg: observations of daily stock prices, customer purchase history, weather measurements, …)
  - A **Task** is a goal we want to achieve or a question we want to answer (eg: which stock options should I buy? How can I better serve my customers? What are the odds of snowing tomorrow?…)
  - A **Mathematical Model** of data describes the relationships between different aspects of data (eg: what's the mathematic formula that relates a company's earning history and past stock prices to the predicted stock price)
  - A **Feature** is a **numeric** representation of raw data

Introduction

- Some models are **more appropriate** to some features and vice versa

Introduction

- Some models are **more appropriate** to some features and vice versa

- A model trained on the **right features** that are easy to ingest will certainly yield **better performances**

Introduction

- Some models are **more appropriate** to some features and vice versa

- A model trained on the **right features** that are easy to ingest will certainly yield **better performances**

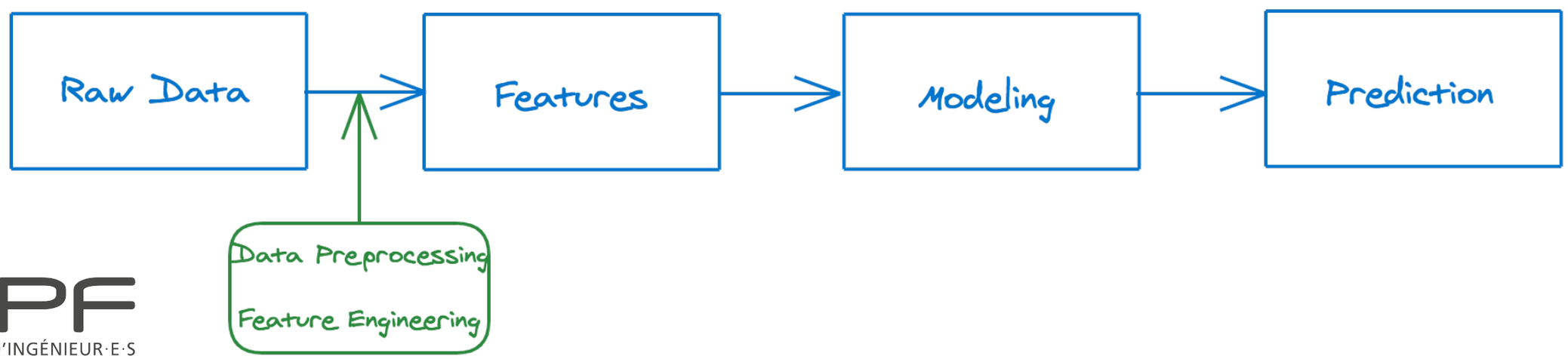
- Feature engineering is the process of **finding the best features** given the **raw data**, the **task** we are trying to achieve and the **model** we decide to use

Introduction

- Some models are **more appropriate** to some features and vice versa

- A model trained on the **right features** that are easy to ingest will certainly yield **better performances**

- Feature engineering is the process of **finding the best features** given the **raw data**, the **task** we are trying to achieve and the **model** we decide to use



Handling numerical features

Handling numerical features

- When dealing with **counts**, we are very likely to encounter extreme values. It’s always a good idea to check the **scale** to decide wether:

Handling numerical features

- When dealing with **counts**, we are very likely to encounter extreme values. It's always a good idea to check the **scale** to decide wether:

  - We keep the data as raw numbers

Handling numerical features

- When dealing with **counts**, we are very likely to encounter extreme values. It's always a good idea to check the **scale** to decide wether:

  - We keep the data as raw numbers
  - Convert them into **binary** values (binarization)

Handling numerical features

- When dealing with **counts**, we are very likely to encounter extreme values. It's always a good idea to check the **scale** to decide wether:

  - We keep the data as raw numbers
  - Convert them into **binary** values (binarization)
  - **Bin** them into **coarser** granularity

Handling numerical features

- When dealing with **counts**, we are very likely to encounter extreme values. It's always a good idea to check the **scale** to decide wether:

  - We keep the data as raw numbers
  - Convert them into **binary** values (binarization)
  - **Bin** them into **coarser** granularity
  - Apply some **numerical transformation** (log, power, …)

Handling numerical features

- When dealing with **counts**, we are very likely to encounter extreme values. It's always a good a idea to check the **scale** to decide wether:
  - We keep the data as raw numbers
  - Convert them into **binary** values (binarization)
  - **Bin** them into **coarser** granularity
  - Apply some **numerical transformation** (log, power, …)
- Let's look at a few examples

Handling numerical features

- Example 1: Binarization

Handling numerical features

- Example 1: Binarization

- Given a dataset containing the full listening histories of one million users on a streaming platform, the task is to build a recommender system to recommend songs to users

Handling numerical features

- Example 1: Binarization

- Given a dataset containing the full listening histories of one million users on a streaming platform, the task is to build a recommender system to recommend songs to users

- We want to predict how much a user will like a particular song. As the data contains the listen counts for each couple (userId, songId), we can be tempted to try predicting the listen count

Handling numerical features

- Example 1: Binarization

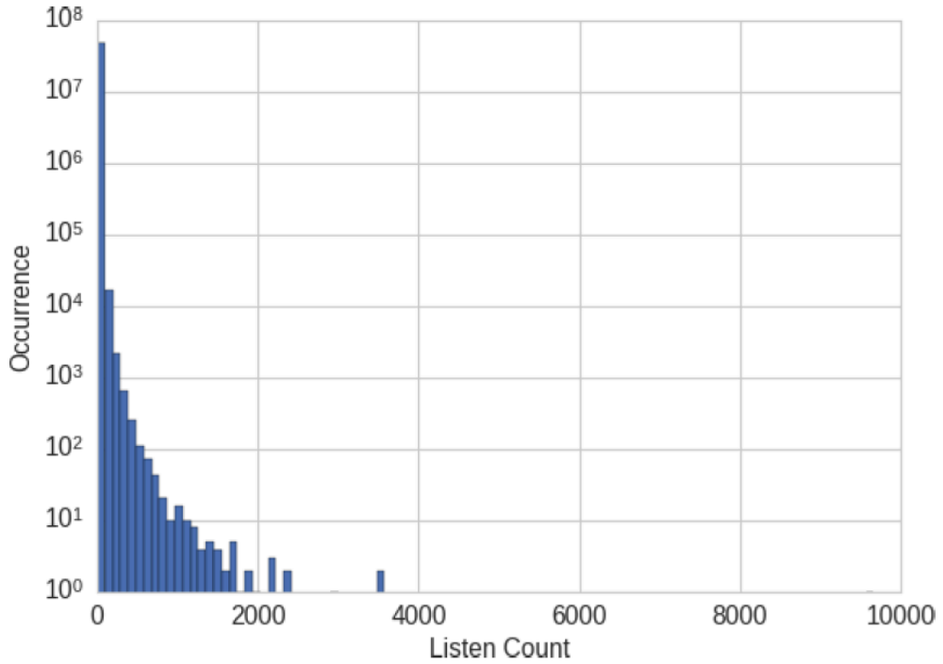
- Given a dataset containing the full listening histories of one million users on a streaming platform, the task is to build a recommender system to recommend songs to users

- We want to predict how much a user will like a particular song. As the data contains the listen counts for each couple (userId, songId), we can be tempted to try predicting the listen count

- Does a large listen count mean the user really likes the song and low listen count mean the user does not like the song?

Handling numerical features

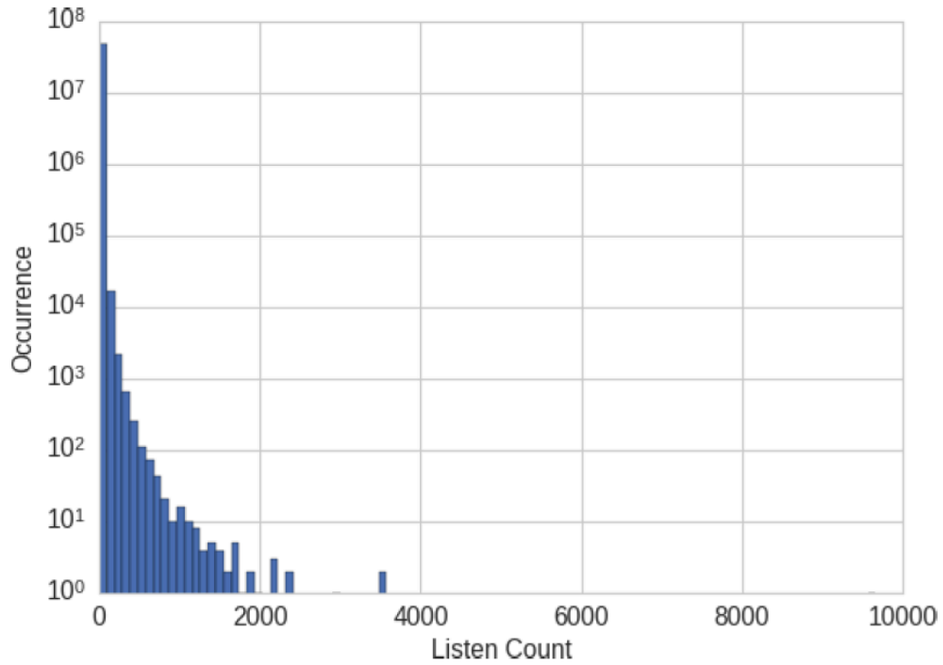
- Example 1: Binarization



Histogram of listen counts in the dataset

Handling numerical features

- Example 1: Binarization



Histogram of listen counts in the dataset

- The data shows that 99% of the counts are below 24, there are some counts in the thousands, these data points are called **outliers**

Handling numerical features

- Example 1: Binarization

- We can see that the listen counts is not a robust measure of user taste. Some users might put their favorite musics in infinite loop, others might like to listen to them on a less frequent basis

Handling numerical features

- Example 1: Binarization

- We can see that the listen counts is not a robust measure of user taste. Some users might put their favorite musics in infinite loop, others might like to listen to them on a less frequent basis

- A more robust measure of user taste is to **binarize (0/1)** these counts: If the user listened to a song more than once, we can count it as the user liked this song

Handling numerical features

- Example 2: Quantization/Binning

Handling numerical features

- Example 2: Quantization/Binning

- Given a dataset containing user reviews of business, we want to predict the rating a user might give to a business

Handling numerical features

- Example 2: Quantization/Binning

- Given a dataset containing user reviews of business, we want to predict the rating a user might give to a business

- The review count might be an interesting feature. There is usually a **strong correlation** between the popularity and good ratings. How can we use this count?

Handling numerical features

- Example 2: Quantization/Binning

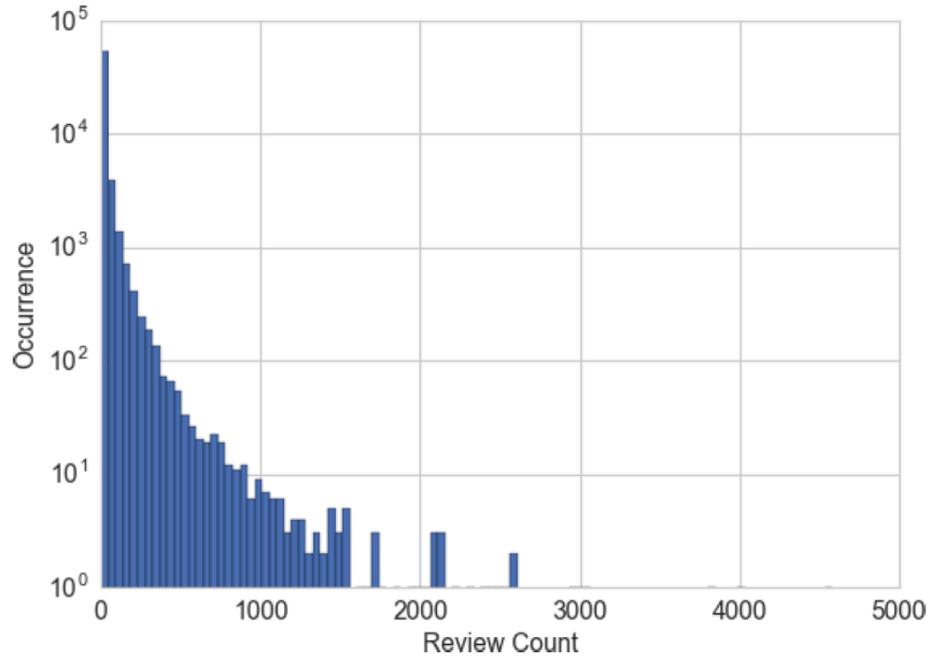
- Given a dataset containing user reviews of business, we want to predict the rating a user might give to a business

- The review count might be an interesting feature. There is usually a **strong correlation** between the popularity and good ratings. How can we use this count?

- Let’s take a look at the data

Handling numerical features

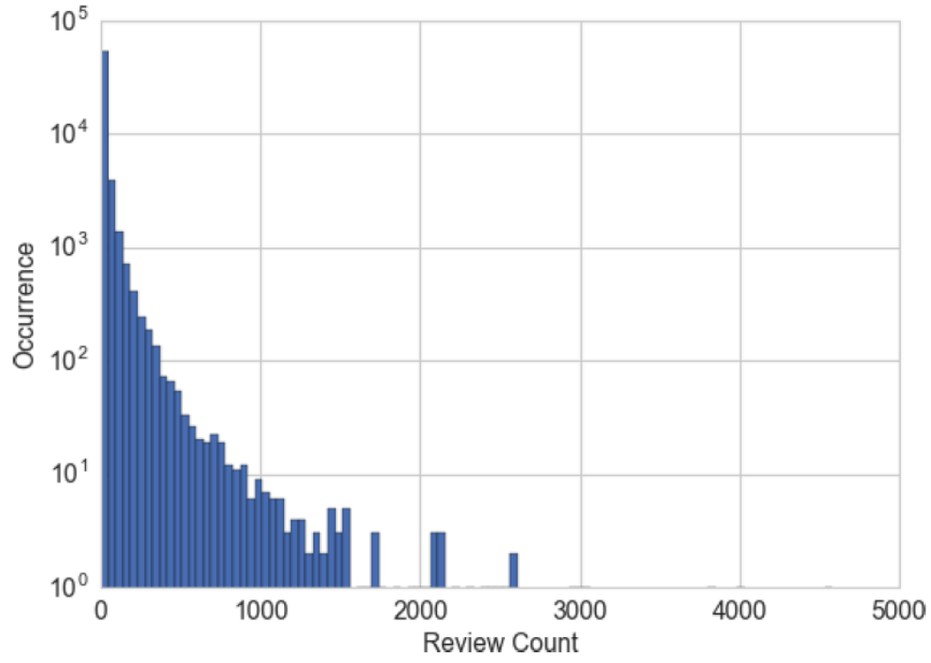
- Example 2: Quantization/Binning



Histogram of review counts in the dataset

Handling numerical features

- Example 2: Quantization/Binning



Histogram of review counts in the dataset

- The raw counts span over several order of **magnitudes**, this can be problematic for linear models as only **one coefficient** is associated to the feature. This coefficient should work for all the orders of magnitude

Handling numerical features

- Example 2: Quantization/Binning

Handling numerical features

- Example 2: Quantization/Binning
- One solution is to **group** the counts into **bins**, this is called **quantization**

Handling numerical features

- Example 2: Quantization/Binning
- One solution is to **group** the counts into **bins**, this is called **quantization**
- Quantization maps a **continuous** number to a **discrete** one

Handling numerical features

- Example 2: Quantization/Binning
- One solution is to **group** the counts into **bins**, this is called **quantization**
- Quantization maps a **continuous** number to a **discrete** one
- We have to decide how quantize data:

Handling numerical features

- Example 2: Quantization/Binning
- One solution is to **group** the counts into **bins**, this is called **quantization**
- Quantization maps a **continuous** number to a **discrete** one
- We have to decide how quantize data:
    1. Fixed width binning (0-9 => 1, 10-19 => 2, 20-29 => 3, …)

Handling numerical features

- Example 2: Quantization/Binning
- One solution is to **group** the counts into **bins**, this is called **quantization**
- Quantization maps a **continuous** number to a **discrete** one
- We have to decide how quantize data:
    1. Fixed width binning (0-9 => 1, 10-19 => 2, 20-29 => 3, …)
    2. Custom-designed ranges (0-12 => 1, 12-17 => 2, 18-24 => 3, …)

Handling numerical features

- Example 2: Quantization/Binning
- One solution is to **group** the counts into **bins**, this is called **quantization**
- Quantization maps a **continuous** number to a **discrete** one
- We have to decide how quantize data:

1. Fixed width binning (0-9 => 1, 10-19 => 2, 20-29 => 3, …)
2. Custom-designed ranges (0-12 => 1, 12-17 => 2, 18-24 => 3, …)
3. Quantile binning: dividing the data into equal portions by using the quantiles of the distribution

Handling numerical features

- Example 3: Log transform

Handling numerical features

- Example 3: Log transform

- Log function is the inverse of the Exp function, it's defined as:

$$log_a(a^x) = x$$

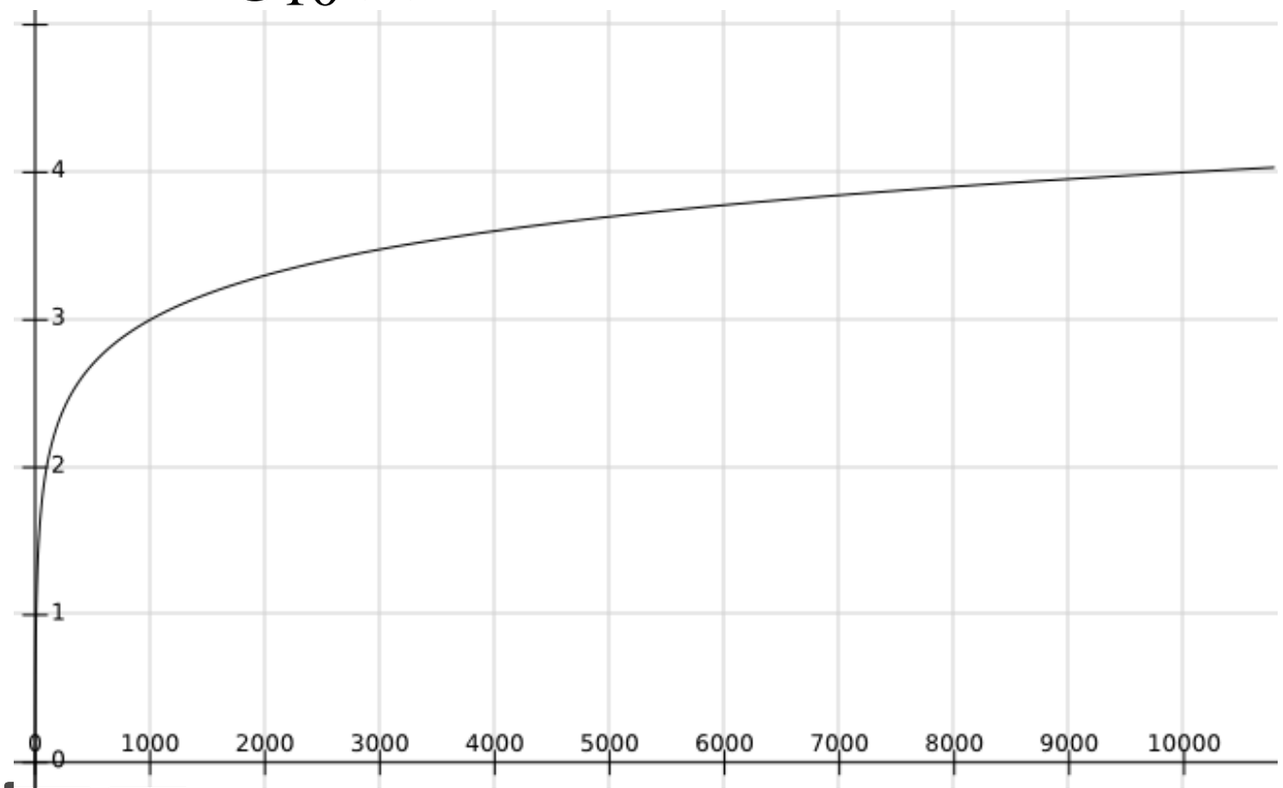
Handling numerical features

- Example 3: Log transform

- Log function is the inverse of the Exp function, it's defined as:

$$log_a(a^x) = x$$

- Ex: $log_{10}(x)$



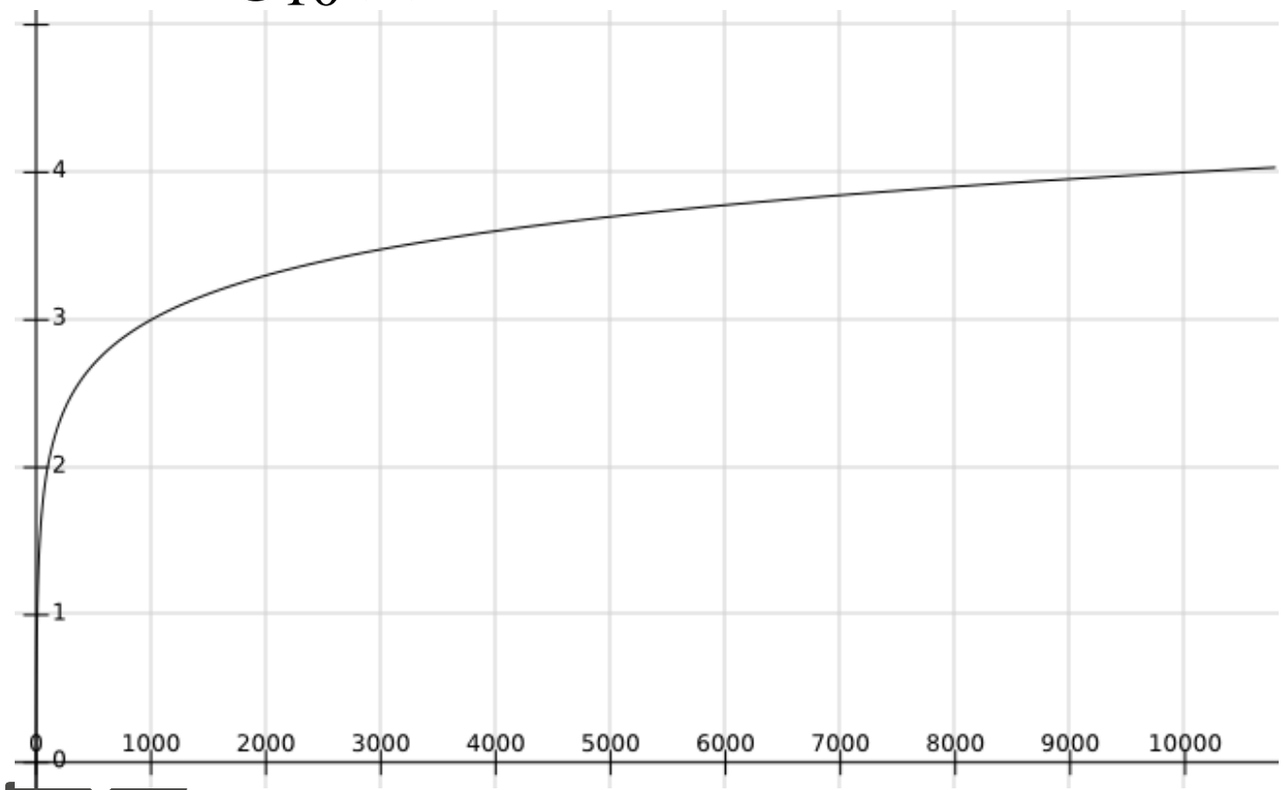
Handling numerical features

- Example 3: Log transform

- Log function is the inverse of the Exp function, it's defined as:

$$log_a(a^x) = x$$

- Ex: $log_{10}(x)$



We can see that this function maps:

- The range of]0,1] to]-inf,0]
- The range of [1,10] to [0,1]
- The range of [10,100] to [1,2]
- …

➡ The log functions compresses the range of large numbers and expands the range of small numbers

➡ It's a powerful tool to deal with positive numbers with a heavy-tailed distribution

Handling numerical features

- Example 3: Log transform
- Given a dataset including features about news articles, we want to predict the popularity of a given article on social media

Handling numerical features

- Example 3: Log transform

- Given a dataset including features about news articles, we want to predict the popularity of a given article on social media

- We want to use the word count of an article as a feature

Handling numerical features

- Example 3: Log transform

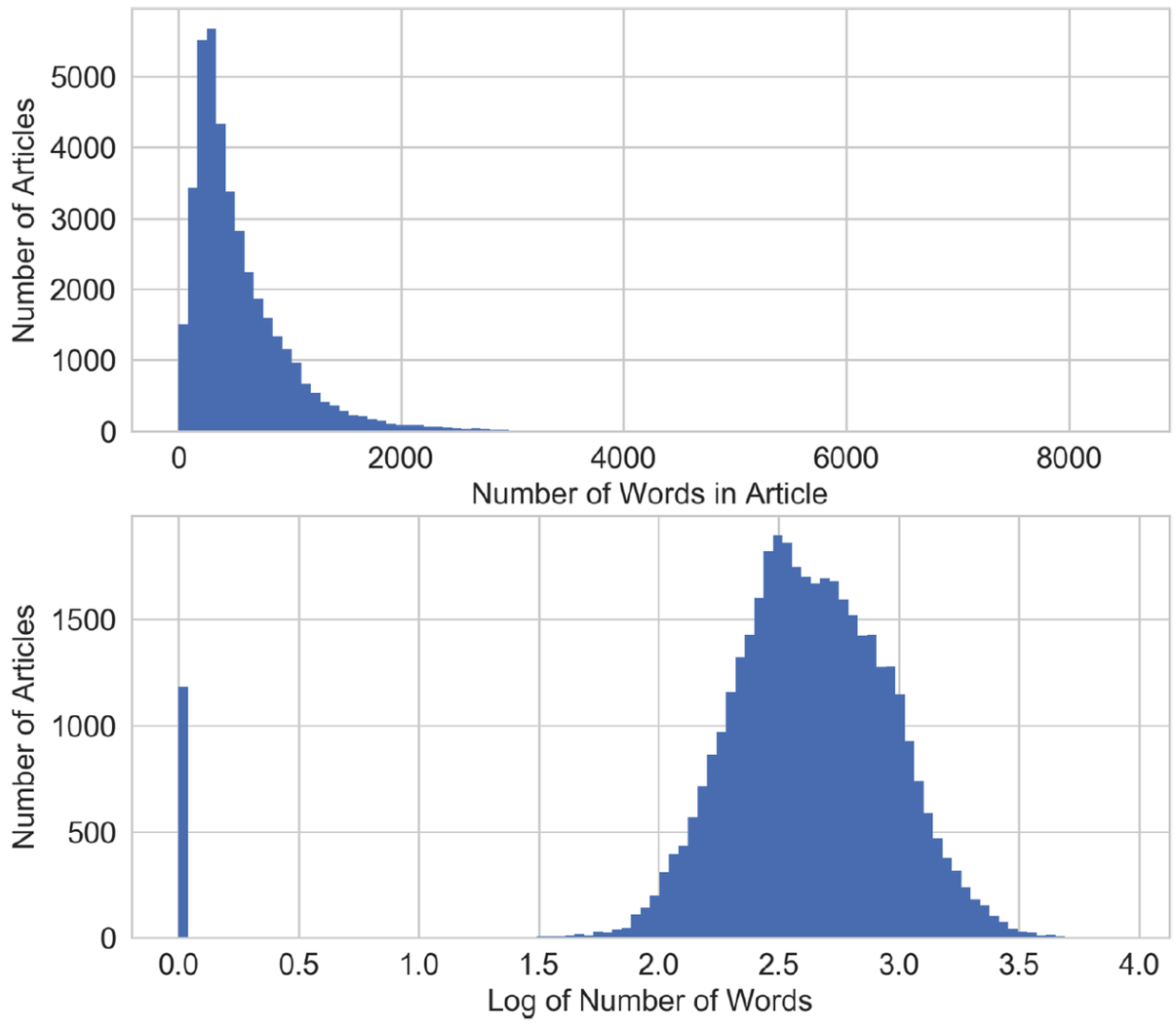
- Given a dataset including features about news articles, we want to predict the popularity of a given article on social media

- We want to use the word count of an article as a feature

- The raw distribution is **skewed** to the left, we need to apply a log transformation

Handling numerical features

- Example 3: Log transform



Comparison of word counts in news articles before (top) and after (bottom) log transformation

Handling numerical features

- Example 4: Min-Max Scaling

Handling numerical features

- Example 4: Min-Max Scaling

- Min-Max scaling is used to **squeeze** all feature values to the [0,1] range by applying the following transformation:

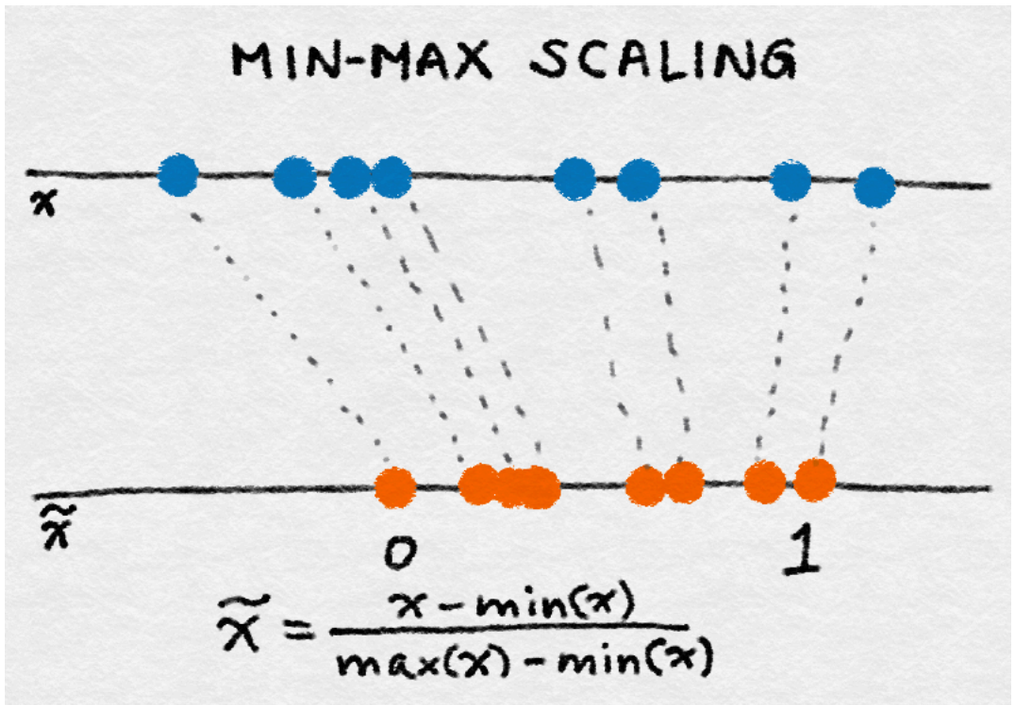
$$\tilde{X} = \frac{X - min(X)}{max(X) - min(X)}$$

Handling numerical features

- Example 4: Min-Max Scaling

- Min-Max scaling is used to **squeeze** all feature values to the [0,1] range by applying the following transformation:

$$\tilde{X} = \frac{X - min(X)}{max(X) - min(X)}$$



Handling numerical features

- Example 5: Standardization

Handling numerical features

- Example 5: Standardization

- Standardization is used to transform a feature into a feature having a mean of 0 and a variance of 1 by applying:

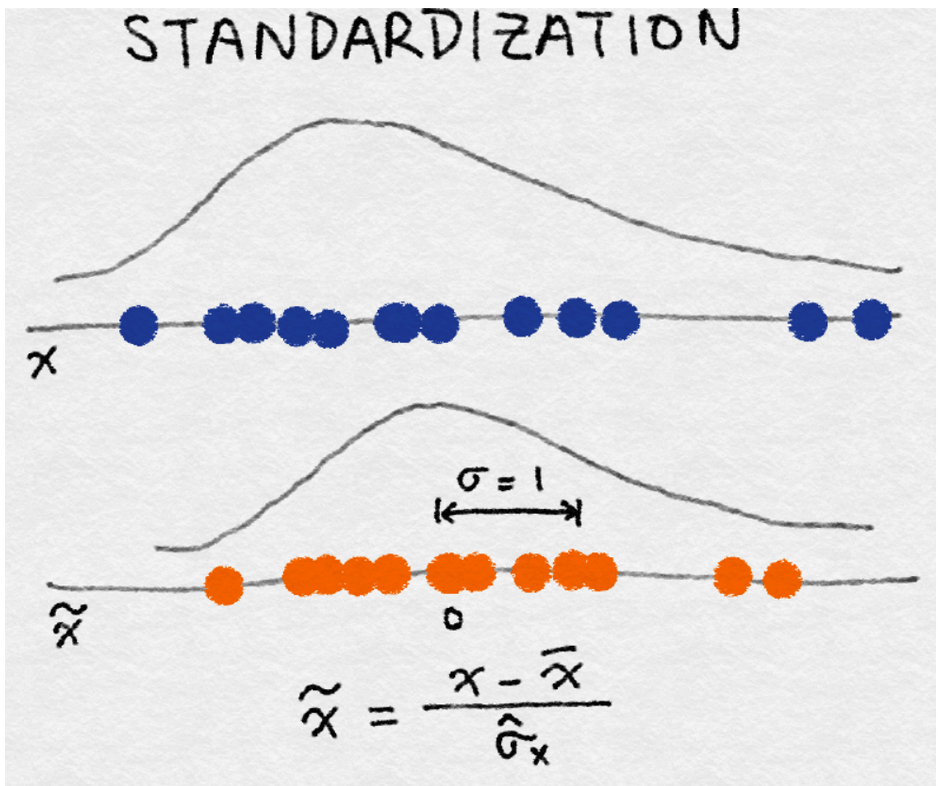
$$\tilde{X} = \frac{X - mean(X)}{\sqrt{var(X)}}$$

Handling numerical features

- Example 5: Standardization

- Standardization is used to transform a feature into a feature having a mean of 0 and a variance of 1 by applying:

$$\tilde{X} = \frac{X - mean(X)}{\sqrt{var(X)}}$$



Handling numerical features

- We've seen 5 different techniques to deal with numerical features. There are many other techniques that are worth trying too (polynomial transformations, box-cox, …)

Handling numerical features

- We've seen 5 different techniques to deal with numerical features. There are many other techniques that are worth trying too (polynomial transformations, box-cox, …)

- Numerical feature scaling is very useful in datasets containing features that **differ wildly in scale**, especially if we're using **linear models**. Random Forests on the other hand are quite robust to features' scale differences

Handling numerical features

- We've seen 5 different techniques to deal with numerical features. There are many other techniques that are worth trying too (polynomial transformations, box-cox, …)

- Numerical feature scaling is very useful in datasets containing features that **differ wildly in scale**, especially if we're using **linear models**. Random Forests on the other hand are quite robust to features' scale differences

- It can also be shown that numerical feature scaling can bring **more stability** and **quicker convergence** for gradient descent

Handling categorical features

Handling categorical features

- A **categorical** variable is used to represent **categories**

Handling categorical features

- A **categorical** variable is used to represent **categories**
- Ex: Cities, seasons of the year, …

Handling categorical features

- A **categorical** variable is used to represent **categories**
- Ex: Cities, seasons of the year, …
- The values may be represented numerically. But unlike numerical variables, the values of a categorical **cannot be ordered** with respect to one another

Handling categorical features

- A **categorical** variable is used to represent **categories**

- Ex: Cities, seasons of the year, …

- The values may be represented numerically. But unlike numerical variables, the values of a categorical **cannot be ordered** with respect to one another

- It doesn’t matter how different two values are, but only that they are different

Handling categorical features

- A **categorical** variable is used to represent **categories**
- Ex: Cities, seasons of the year, …
- The values may be represented numerically. But unlike numerical variables, the values of a categorical **cannot be ordered** with respect to one another
- It doesn’t matter how different two values are, but only that they are different
- Ex: A stock price is numerical, a company id is categorical

Handling categorical features

- To use categorical features into ML models, we need encode them into numbers

Handling categorical features

- To use categorical features into ML models, we need encode them into numbers

- There are many techniques to achieve this:

Handling categorical features

- To use categorical features into ML models, we need encode them into numbers

- There are many techniques to achieve this:
  - One-hot encoding / Dummy coding

Handling categorical features

- To use categorical features into ML models, we need encode them into numbers

- There are many techniques to achieve this:
  - One-hot encoding / Dummy coding
  - Feature hashing

Handling categorical features

- To use categorical features into ML models, we need encode them into numbers

- There are many techniques to achieve this:
  - One-hot encoding / Dummy coding
  - Feature hashing
  - Bin counting

Handling categorical features

- **One-hot encoding**: using a **group of bits**, each bit represents a possible category

Handling categorical features

- **One-hot encoding**: using a **group of bits**, each bit represents a possible category

- A categorical variable with k categories is represented as a feature vector of size k

Handling categorical features

- **One-hot encoding**: using a **group of bits**, each bit represents a possible category

- A categorical variable with k categories is represented as a feature vector of size k

- Ex: One-hot-encoding a categorical variable with 3 categories

City	City_1	City_2	City_3
Montpellier	1	0	0
Paris	0	1	0
Marseille	0	0	1

Handling categorical features

- **Dummy-coding** is the same as One-hot encoding but uses one less bit

Handling categorical features

- **Dummy-coding** is the same as One-hot encoding but uses one less bit

- A categorical variable with k categories is represented as a feature vector of size k-1

Handling categorical features

- **Dummy-coding** is the same as One-hot encoding but uses one less bit

- A categorical variable with k categories is represented as a feature vector of size k-1

- Ex: One-hot-encoding a categorical variable with 3 categories

City	City_1	City_2
Montpellier	1	0
Paris	0	1
Marseille	0	0

Handling categorical features

- **One-hot encoding** and **Dummy-coding** are very similar

Handling categorical features

- **One-hot encoding** and **Dummy-coding** are very similar

- We can show that One-hot encoding is **redundant**. It introduces a **collinearity issue** that leads to a **non-uniqueness** of the solution

Handling categorical features

- **One-hot encoding** and **Dummy-coding** are very similar

- We can show that One-hot encoding is **redundant**. It introduces a **collinearity issue** that leads to a **non-uniqueness** of the solution

- Dummy-coding removes this **redundancy**, however it doesn’t handle well **missing data**. In One-hot encoding missing data is assigned to the all-zero vector. In Dummy-coding the all-zero vector is already taken by the reference

Handling categorical features

- **One-hot encoding** and **Dummy-coding** are very similar

- We can show that One-hot encoding is **redundant**. It introduces a **collinearity issue** that leads to a **non-uniqueness** of the solution

- Dummy-coding removes this **redundancy**, however it doesn't handle well **missing data**. In One-hot encoding missing data is assigned to the all-zero vector. In Dummy-coding the all-zero vector is already taken by the reference

- These 2 methods are not enough to deal with **large categorical variables**, we need some alternatives

Handling categorical features

- In internet applications, the number of categories in a categorical feature can be **very large**. Using plain One-hot encoding would yield to **very large datasets**, thus requiring more computing power

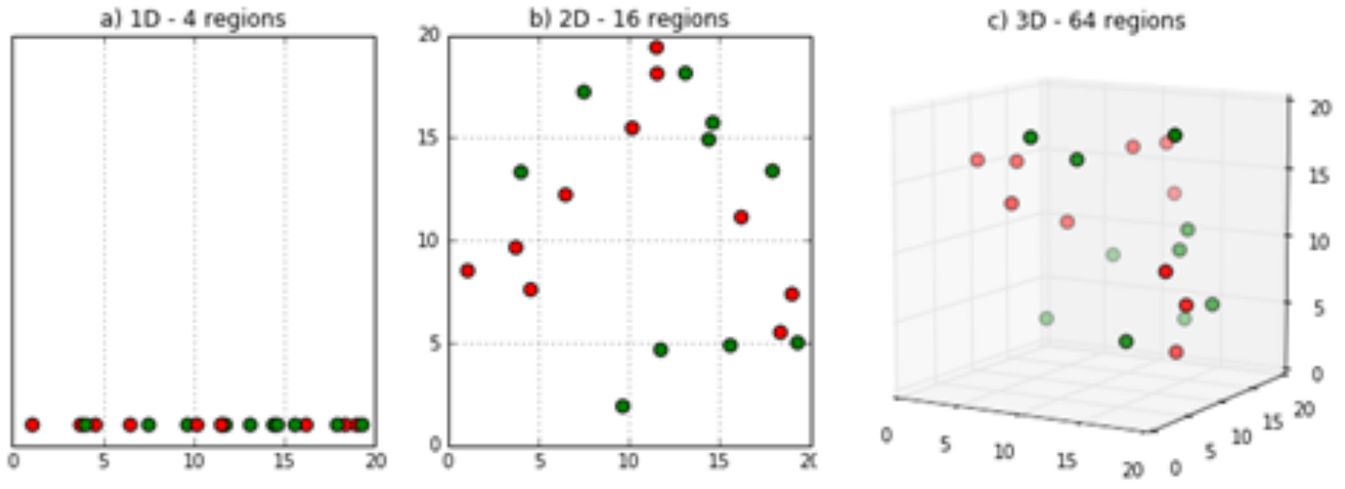
Handling categorical features

- In internet applications, the number of categories in a categorical feature can be **very large**. Using plain One-hot encoding would yield to **very large datasets**, thus requiring more computing power

- This can lead to a common issue known as the **curse of dimensionality**: As the volume of the features space grows, the available data becomes **more sparse**

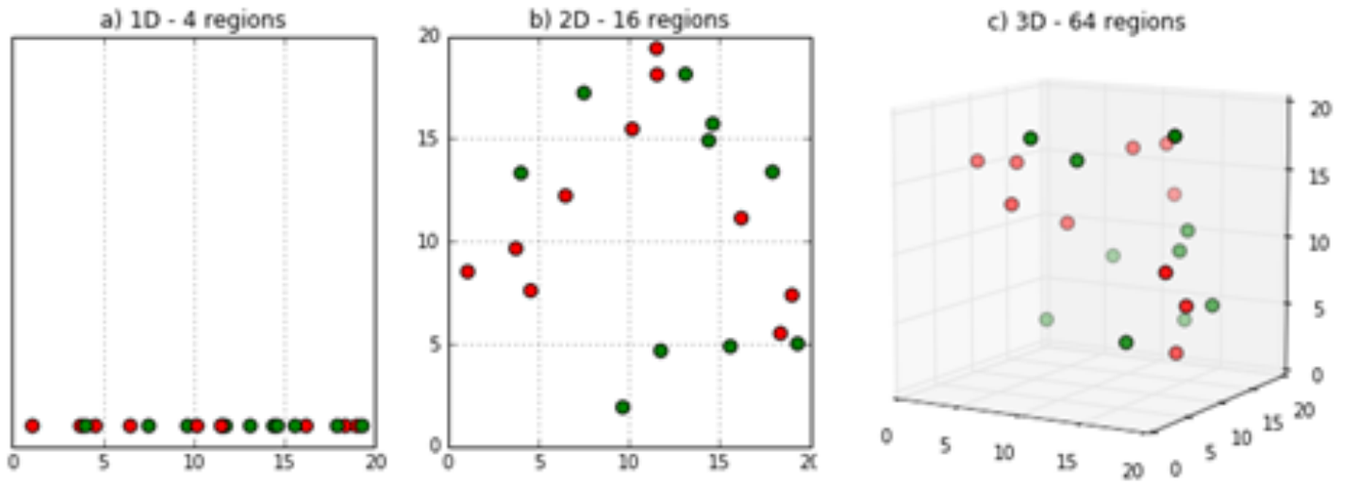
Handling categorical features

- Illustration of the curse of dimensionality



Handling categorical features

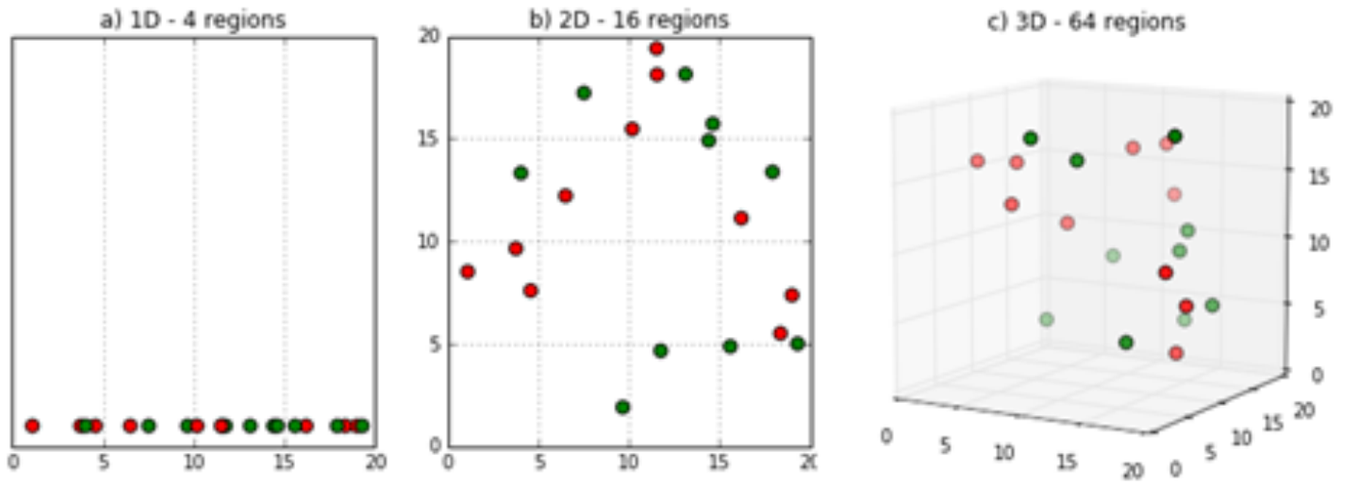
- Illustration of the curse of dimensionality



- As the **dimension** of the space **grows**, the given data fills **less and less space**

Handling categorical features

- Illustration of the curse of dimensionality

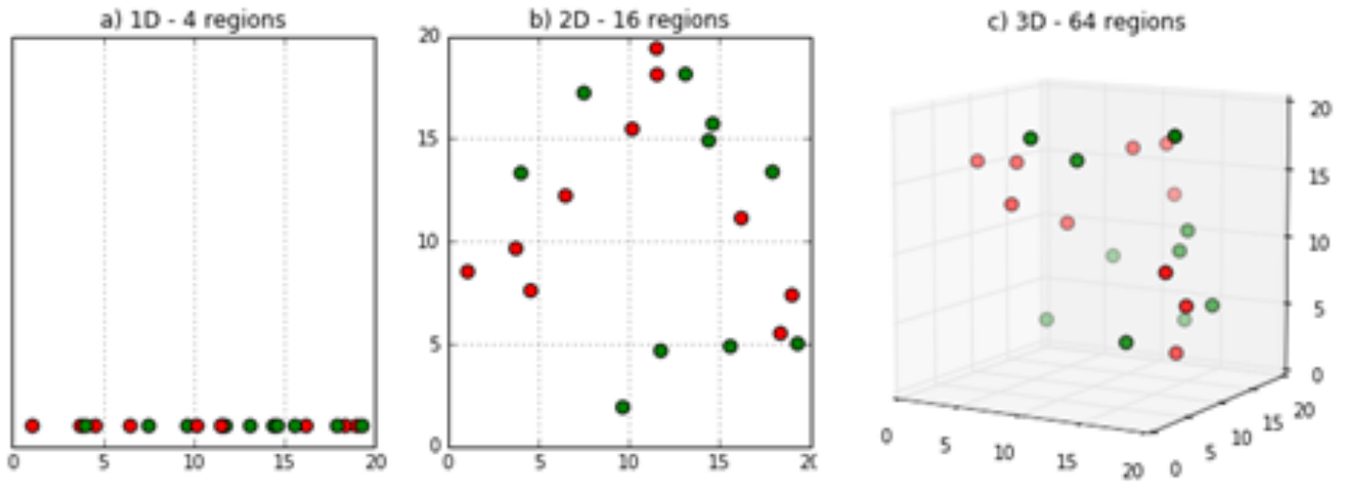


- As the **dimension** of the space **grows**, the given data fills **less and less space**

- As a consequence we will need **more data** to obtain **statistical significance**

Handling categorical features

- Illustration of the curse of dimensionality



- As the **dimension** of the space **grows**, the given data fills **less and less space**
- As a consequence we will need **more data** to obtain **statistical significance**
- Fortunately, there are some techniques to deal with very large categories: hashing, bin counting, …

Handling categorical features

- **Hashing**: using a **deterministic** hash function that maps a potentially **unbounded integer** to a **finite integer** range [1,M]

Handling categorical features

- **Hashing**: using a **deterministic** hash function that maps a potentially **unbounded integer** to a **finite integer** range [1,M]

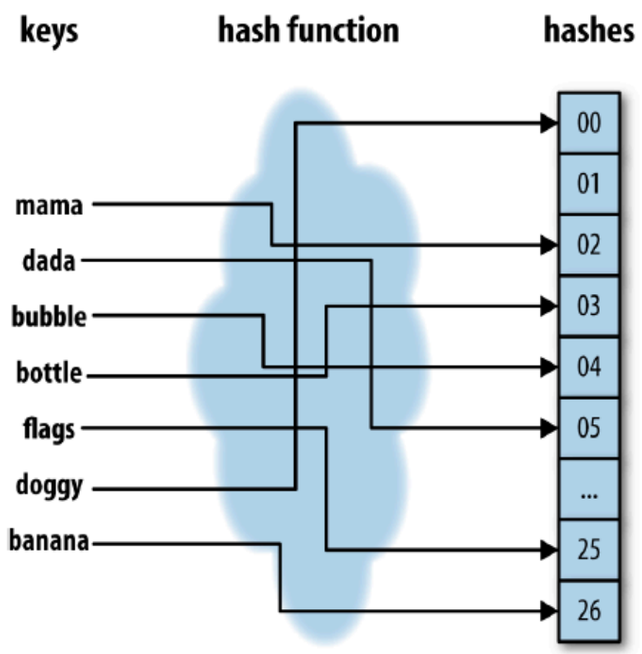
- It can be seen as way to **compress** a large feature vector into an M-dimensional vector by using the following transformation

Handling categorical features

- **Hashing**: using a **deterministic** hash function that maps a potentially **unbounded integer** to a **finite integer** range [1,M]

- It can be seen as way to **compress** a large feature vector into an M-dimensional vector by using the following transformation

$$f(X) = \textbf{HASH}(X)\,\%\,M$$



Handling categorical features

- Since the input space is potentially larger than the output space, multiple numbers can get mapped to the same output. This is called a **collision**

Handling categorical features

- Since the input space is potentially larger than the output space, multiple numbers can get mapped to the same output. This is called a **collision**

- The larger the output space size M, the lesser the collisions are, the higher the consumption of memory during the training

Handling categorical features

- Since the input space is potentially larger than the output space, multiple numbers can get mapped to the same output. This is called a **collision**

- The larger the output space size M, the lesser the collisions are, the higher the consumption of memory during the training

- Choosing the right M can be seen as a **tradeoff between memory consumption and collision rate**

Handling categorical features

- Since the input space is potentially larger than the output space, multiple numbers can get mapped to the same output. This is called a **collision**

- The larger the output space size M, the lesser the collisions are, the higher the consumption of memory during the training

- Choosing the right M can be seen as a **tradeoff between memory consumption and collision rate**

- We can show that a controlled collision rate does not deteriorate the model's performances

Handling categorical features

- One-hot encoding / dummy coding
  - Not scalable: cannot support high-cardinality attributes

Handling categorical features

- One-hot encoding / dummy coding
  - Not scalable: cannot support high-cardinality attributes
  - Not efficient: large value index dictionary must be retained

Handling categorical features

- One-hot encoding / dummy coding
  - Not scalable: cannot support high-cardinality attributes
  - Not efficient: large value index dictionary must be retained
  - Not flexible: only linear models can benefit from it

Handling categorical features

- One-hot encoding / dummy coding
  - Not scalable: cannot support high-cardinality attributes
  - Not efficient: large value index dictionary must be retained
  - Not flexible: only linear models can benefit from it
  - Not adaptive: doesn’t support drift in attribute values

Handling categorical features

- Hashing trick:

  + Scalable: no mapping tables

Handling categorical features

- Hashing trick:
  - + Scalable: no mapping tables
  - + Efficient: low cost, preserves sparsity

Handling categorical features

- Hashing trick:
  - \+ Scalable: no mapping tables
  - \+ Efficient: low cost, preserves sparsity
  - – Not flexible: only linear learners can benefit from it

Handling categorical features

- Hashing trick:

  + Scalable: no mapping tables

  + Efficient: low cost, preserves sparsity

  – Not flexible: only linear learners can benefit from it

  ± Adaptive: new values are ok, no temporal effect

Handling categorical features

- An alternative solution is called **Bin Counting**

Handling categorical features

- An alternative solution is called **Bin Counting**

- Core idea: Instead of using the value of the categorical variable as the feature, use **statistics of the target under that values** (the conditional probability, counts of classes, odds ratios, …)

Handling categorical features

- An alternative solution is called **Bin Counting**

- Core idea: Instead of using the value of the categorical variable as the feature, use **statistics of the target under that values** (the conditional probability, counts of classes, odds ratios, …)

- Example: Click prediction with Bin Counting

Handling categorical features

- An alternative solution is called **Bin Counting**

- Core idea: Instead of using the value of the categorical variable as the feature, use **statistics of the target under that values** (the conditional probability, counts of classes, odds ratios, …)

- Example: Click prediction with Bin Counting

  ➡ We want to predict wether a user will click on an advertisement or not, given some categorical features (userId, adId, websiteId…)

Handling categorical features

- An alternative solution is called **Bin Counting**

- Core idea: Instead of using the value of the categorical variable as the feature, use **statistics of the target under that values** (the conditional probability, counts of classes, odds ratios, …)

- Example: Click prediction with Bin Counting

  ➡ We want to predict wether a user will click on an advertisement or not, given some categorical features (userId, adId, websiteId…)

  ➡ These features can have extremely large cardinalities (thousands, millions…)

Handling categorical features

- Example: Click prediction with Bin Counting

userId	adId	…	click
9cee4db5	026244a3303d	…	1
43e6b34c	026244a3303d	…	1
bf7c2071	7cf137c0d1d2	…	0
d47ee3d0	de13539e3365	…	0
d47ee3d0	de13539e3365	…	0
43e6b34c	7cf137c0d1d2	…	1
9cee4db5	8a14d3c7ebfe	…	1
9cee4db5	026244a3303d	…	1
…	…	…	
bf7c2071	8a14d3c7ebfe	…	0

Handling categorical features

- Example: Click prediction with Bin Counting

userId	NB click	NB no-click	Conditional click Probability
9cee4db5	11	5	0.6875
43e6b34c	12	6	0.6667
bf7c2071	2	4	0.3334
d47ee3d0	1	3	0.25
…	…	…	…

Handling categorical features

- Example: Click prediction with Bin Counting

userId	NB click	NB no-click	Conditional click Probability
9cee4db5	11	16	0.6875
43e6b34c	12	18	0.6667
bf7c2071	2	6	0.3334
d47ee3d0	1	4	0.25
…	…	…	…

adId	NB click	NB no-click	Conditional click Probability
026244a3303	59	92	0.6413
7cf137c0d1d2	66	94	0.7021
de13539e336	20	42	0.4761
8a14d3c7ebfe	54	78	0.6923
…	…	…	…

Handling categorical features

- Example: Click prediction with Bin Counting

userIdProcessed	adIdProcessed	…	click
0.6875	0.6413	…	1
0.6667	0.6413	…	1
0.3334	0.7021	…	0
0.25	0.4761	…	0
0.25	0.4761	…	0
0.6667	0.7021	…	1
0.6875	0.6923	…	1
0.6875	0.6413	…	1
…	…	…	
0.3334	0.6923	…	0

Handling categorical features

- Example: Click prediction with Bin Counting

Handling categorical features

- Example: Click prediction with Bin Counting
  - If we have 1M+ distinct userIds, and 1M+ distinct adIds, we would end up having 2M+ columns if we use one-hot encoding

Handling categorical features

- Example: Click prediction with Bin Counting
  - If we have 1M+ distinct userIds, and 1M+ distinct adIds, we would end up having 2M+ columns if we use one-hot encoding
  - With Bin Counting, we end up having **only** 2 columns

Handling categorical features

- Example: Click prediction with Bin Counting
  - If we have 1M+ distinct userIds, and 1M+ distinct adIds, we would end up having 2M+ columns if we use one-hot encoding
  - With Bin Counting, we end up having **only** 2 columns
  - In this example, we used the **conditional probability**. We can use other features such as **odds ratios**, log odds ratios, …

Handling categorical features

- Example: Click prediction with Bin Counting
  - If we have 1M+ distinct userIds, and 1M+ distinct adIds, we would end up having 2M+ columns if we use one-hot encoding
  - With Bin Counting, we end up having **only** 2 columns
  - In this example, we used the **conditional probability**. We can use other features such as **odds ratios**, log odds ratios, …

$$\textbf{oddsRatio}(user = 9cee4db5) = \frac{P(click \mid user = 9cee4db5) \quad / \quad P(noclick \mid user = 9cee4db5)}{P(click \mid user \neq 9cee4db5) \quad / \quad P(noclick \mid user \neq 9cee4db5)} = 0.7902$$

Handling categorical features

- Example: Click prediction with Bin Counting
  - If we have 1M+ distinct userIds, and 1M+ distinct adIds, we would end up having 2M+ columns if we use one-hot encoding
  - With Bin Counting, we end up having **only** 2 columns
  - In this example, we used the **conditional probability**. We can use other features such as **odds ratios**, log odds ratios, …

$$\textbf{oddsRatio}(user = 9cee4db5) = \frac{P(click \mid user = 9cee4db5) \quad / \quad P(noclick \mid user = 9cee4db5)}{P(click \mid user \neq 9cee4db5) \quad / \quad P(noclick \mid user \neq 9cee4db5)} = 0.7902$$

  - Odds ratio can be interpreted as « How much more likely is user 9cee4db5 to click than to not click, compared to others »

Handling categorical features

- Bin counting:

  + Scalable: no mapping tables

Handling categorical features

- Bin counting:

  + Scalable: no mapping tables

  + Efficient: low cost, low dimensionality

Handling categorical features

- Bin counting:
  - + Scalable: no mapping tables
  - + Efficient: low cost, low dimensionality
  - + Flexible: works well with all sorts of models

Handling categorical features

- Bin counting:

  + Scalable: no mapping tables

  + Efficient: low cost, low dimensionality

  + Flexible: works well with all sorts of models

  + Adaptive: new values easily added, we can use backoff for infrequent values

Handling categorical features

- Bin counting:

  + Scalable: no mapping tables

  + Efficient: low cost, low dimensionality

  + Flexible: works well with all sorts of models

  + Adaptive: new values easily added, we can use backoff for infrequent values

  ± Risky: should be handled with great caution, can introduce data leakage

Handling categorical features

- Bin counting relies on **historic data** to generate the necessary **statistics**

Handling categorical features

- Bin counting relies on **historic data** to generate the necessary **statistics**
- Can we just use the same dataset to compute the statistics and train the model?

Handling categorical features

- Bin counting relies on **historic data** to generate the necessary **statistics**

- Can we just use the same dataset to compute the statistics and train the model?

  ➡ **Absolutely not**: As the statistics involve the target variable, using the **output** to compute **input** features leads to an **information leakage**

Handling categorical features

- Bin counting relies on **historic data** to generate the necessary **statistics**

- Can we just use the same dataset to compute the statistics and train the model?

➡ **Absolutely not**: As the statistics involve the target variable, using the **output** to compute **input** features leads to an **information leakage**

➡ **Information leakage** happens when some information that the model is **not supposed to have access to becomes available to it**, thus giving the model an **unfair advantage**. For example, if the input variables for a row i contains information about the target variable of row i

Handling categorical features

- Bin counting relies on **historic data** to generate the necessary **statistics**

- Can we just use the same dataset to compute the statistics and train the model?

➡ **Absolutely not**: As the statistics involve the target variable, using the **output** to compute **input** features leads to an **information leakage**

➡ **Information leakage** happens when some information that the model is **not supposed to have access to becomes available to it**, thus giving the model an **unfair advantage**. For example, if the input variables for a row i contains information about the target variable of row i

➡ This can also happen if test data is leaked into training data, or if future data is leaked into past data

Handling categorical features

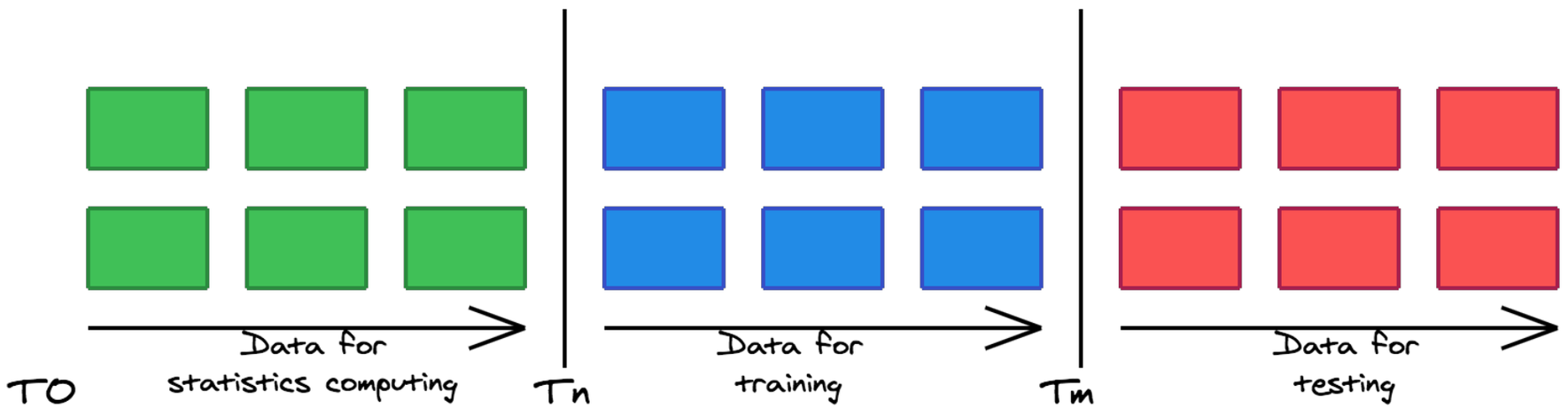
- To **avoid information leakage** while using bin counting, we can:

Handling categorical features

- To **avoid information leakage** while using bin counting, we can:
  - Use **different time windows** for statistics computation, training & testing

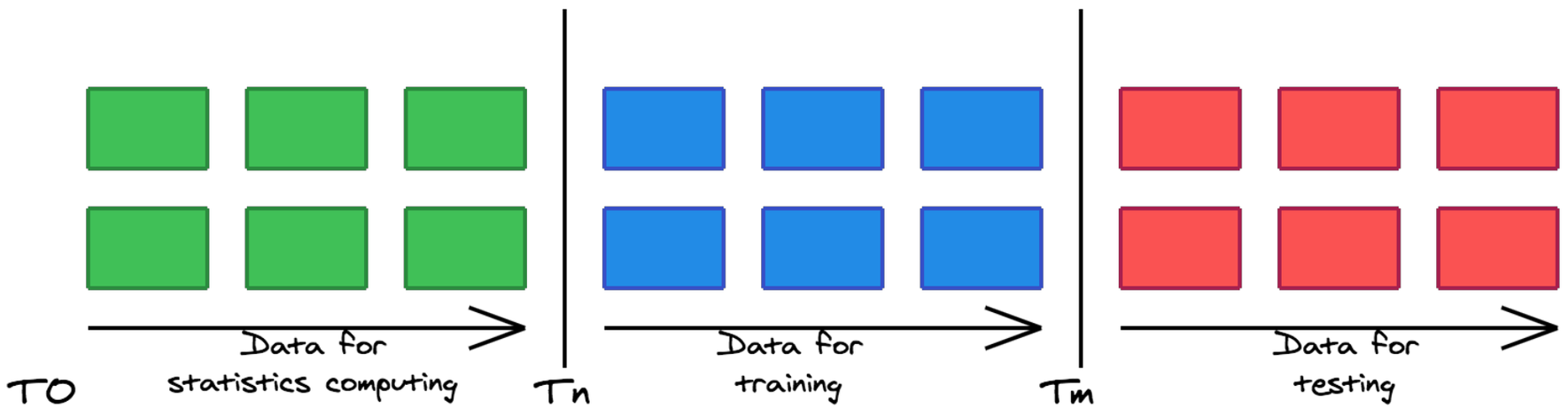
Handling categorical features

- To **avoid information leakage** while using bin counting, we can:
  - Use **different time windows** for statistics computation, training & testing



Handling categorical features

- To **avoid information leakage** while using bin counting, we can:
  - Use **different time windows** for statistics computation, training & testing



  - Use **leave-one-out** counting associated with a **small random noise** to cover for for any potential data-leakage

Handling categorical features

- Summary

Plain one-hot encoding	
Pros	- Easy to implement - Accurate - Feasible for online learning
Cons	- Computationally inefficient - Does not adapt to growing categories - Only feasible for linear models

Handling categorical features

- Summary

Feature hashing	
Pros	- Easy to implement - Makes model training cheaper - Easily adaptable to new categories - Easily handles rare categories
Cons	- Only suitable for linear models - Hashed features are not interpretable - Not always yielding a good accuracy

Handling categorical features

- Summary

Bin counting	
Pros	- No computational burden at training time - Suitable for all sorts of models - Easily adaptable to new categories - Easily handles rare categories with back-off - Interpretable
Cons	- Requires historical data - Not suitable for online learning - High potential of information leakage

Using interaction features

Interaction features

- It often happens that 2 individual features don't have any predictive power when used **separately**, but can be **predictive** when **combined**. *(Eg: Latitude + Month for snow prediction)*

Interaction features

- It often happens that 2 individual features don't have any predictive power when used **separately**, but can be **predictive** when **combined**. *(Eg: Latitude + Month for snow prediction)*

- We can solve that by using **interaction features**, which can be modeled as a **logical AND** between 2 or more features

Interaction features

- It often happens that 2 individual features don't have any predictive power when used **separately**, but can be **predictive** when **combined**. *(Eg: Latitude + Month for snow prediction)*

- We can solve that by using **interaction features**, which can be modeled as a **logical AND** between 2 or more features

- Eg:

userId	adId	userId&adId	click
9cee4db5	026244a3303d	9cee4db5**&**026244a3303d	1
43e6b34c	026244a3303d	43e6b34c**&**026244a3303d	1
bf7c2071	7cf137c0d1d2	bf7c2071**&**7cf137c0d1d2	0
d47ee3d0	de13539e3365	d47ee3d0**&**de13539e3365	0

Interaction features

- Interaction features are most helpful with **linear models**, as by design, they are linear with respect the features and cannot easily grasp feature interactions. We can define interactions as:

Interaction features

- Interaction features are most helpful with **linear models**, as by design, they are linear with respect the features and cannot easily grasp feature interactions. We can define interactions as:

$$y = \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \beta_{1,2}(x_1 \,\&\, x_2) + \beta_{1,3}(x_1 \,\&\, x_3) + \ldots$$

Interaction features

- Interaction features are most helpful with **linear models**, as by design, they are linear with respect the features and cannot easily grasp feature interactions. We can define interactions as:

$$y = \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \beta_{1,2}(x_1 \,\&\, x_2) + \beta_{1,3}(x_1 \,\&\, x_3) + \ldots$$

- **Tree-based models** can approximate these interactions by doing splits based on different features. However, it can be worth it to help the models by feeding them interactions that seem interesting

Feature engineering

- We have presented some of the common feature engineering techniques

Feature engineering

- We have presented some of the common feature engineering techniques

- They don’t apply to any kind of data. **Special** data requires **special** techniques:

Feature engineering

- We have presented some of the common feature engineering techniques

- They don’t apply to any kind of data. **Special** data requires **special** techniques:
  - Temporal data: extract info about the day of the week, the month, temporal averages, …

Feature engineering

- We have presented some of the common feature engineering techniques

- They don’t apply to any kind of data. **Special** data requires **special** techniques:
  - Temporal data: extract info about the day of the week, the month, temporal averages, …
  - Geo-spatial data: compute distances …

Feature engineering

- We have presented some of the common feature engineering techniques

- They don’t apply to any kind of data. **Special** data requires **special** techniques:
  - Temporal data: extract info about the day of the week, the month, temporal averages, …
  - Geo-spatial data: compute distances …

- How can we get feature engineering ideas for our data?

Feature engineering

- We have presented some of the common feature engineering techniques

- They don’t apply to any kind of data. **Special** data requires **special** techniques:
  - Temporal data: extract info about the day of the week, the month, temporal averages, …
  - Geo-spatial data: compute distances …

- How can we get feature engineering ideas for our data?

  ➡ **Exploratory data analysis**

Exploratory Data Analysis

- In each ML project, it is **critical** to have a **deep understanding** of:

Exploratory Data Analysis

- In each ML project, it is **critical** to have a **deep understanding** of:
  - The **properties** of the data: schema, statistical properties …

Exploratory Data Analysis

- In each ML project, it is **critical** to have a **deep understanding** of:
  - The **properties** of the data: schema, statistical properties …
  - The **quality** of the data: missing values, inconsistent data types …

Exploratory Data Analysis

- In each ML project, it is **critical** to have a **deep understanding** of:
  - The **properties** of the data: schema, statistical properties …
  - The **quality** of the data: missing values, inconsistent data types …
  - The **predictive power** of the data: for example, the correlation of features with the target

Exploratory Data Analysis

- In each ML project, it is **critical** to have a **deep understanding** of:
  - The **properties** of the data: schema, statistical properties …
  - The **quality** of the data: missing values, inconsistent data types …
  - The **predictive power** of the data: for example, the correlation of features with the target
- The results of the EDA can influence **subsequent decisions** on how we deal with the task at hand

Exploratory Data Analysis

- The goal of doing statistical analyses of the data is to determine the **quality of the features** and their **predictive power** in contrast with the target value

- The data exploration is conducted from three different angles:
  - **Descriptive**
  - **Correlative**
  - **Contextual**

Exploratory Data Analysis

- **Descriptive** analysis (univariate analysis) provides an understanding of the characteristics of each variable

Exploratory Data Analysis

- **Descriptive** analysis (univariate analysis) provides an understanding of the characteristics of each variable

Attribute type	Statistic/calculation	Details
Common	Data type	Attribute's data type
	Missing values	Percentage of missing values
Numerical	Quantile statistics	Q1, Q2, Q3, Q3, min, max, range
	Descriptive statistics	Mean, mode, standard deviation, median absolute deviation, kurtosis, skewness
	Distribution histogram	Based on the appropriate number of bins
Categorical	Cardinality	Number of unique values for the categorical feature
	Unique counts	Number of occurrences for each unique value of the categorical feature

Exploratory Data Analysis

- **Correlation** analysis (bivariate analysis) examines the **relationship** between two attributes, and determines whether the two are **correlated**. It can be done from two perspectives:

Exploratory Data Analysis

- **Correlation** analysis (bivariate analysis) examines the **relationship** between two attributes, and determines whether the two are **correlated**. It can be done from two perspectives:

  - **Qualitative analysis:** computation of the descriptive statistics of the target numerical or categorical attributes against each unique value of the independent feature

Exploratory Data Analysis

- **Correlation** analysis (bivariate analysis) examines the **relationship** between two attributes, and determines whether the two are **correlated**. It can be done from two perspectives:

  - **Qualitative analysis:** computation of the descriptive statistics of the target numerical or categorical attributes against each unique value of the independent feature

  - **Quantitative analysis:** testing the relationship between X and Y based on a hypothesis-testing framework

Exploratory Data Analysis

- **Contextual** analysis is used to further understand the given dataset and to gain **domain-specific** insights. We can talk about:

Exploratory Data Analysis

- **Contextual** analysis is used to further understand the given dataset and to gain **domain-specific** insights. We can talk about:

  - **Time-based analysis**: in many real-world datasets, the timestamp is one of the **key contextual** information. It's very important to understand the characteristics of the data along with the time dimension with different granularities. Ex: number of records per time interval, number of unique values per time interval, descriptive statistics per time interval, …

Exploratory Data Analysis

- **Contextual** analysis is used to further understand the given dataset and to gain **domain-specific** insights. We can talk about:

  - **Time-based analysis**: in many real-world datasets, the timestamp is one of the **key contextual** information. It's very important to understand the characteristics of the data along with the time dimension with different granularities. Ex: number of records per time interval, number of unique values per time interval, descriptive statistics per time interval, …

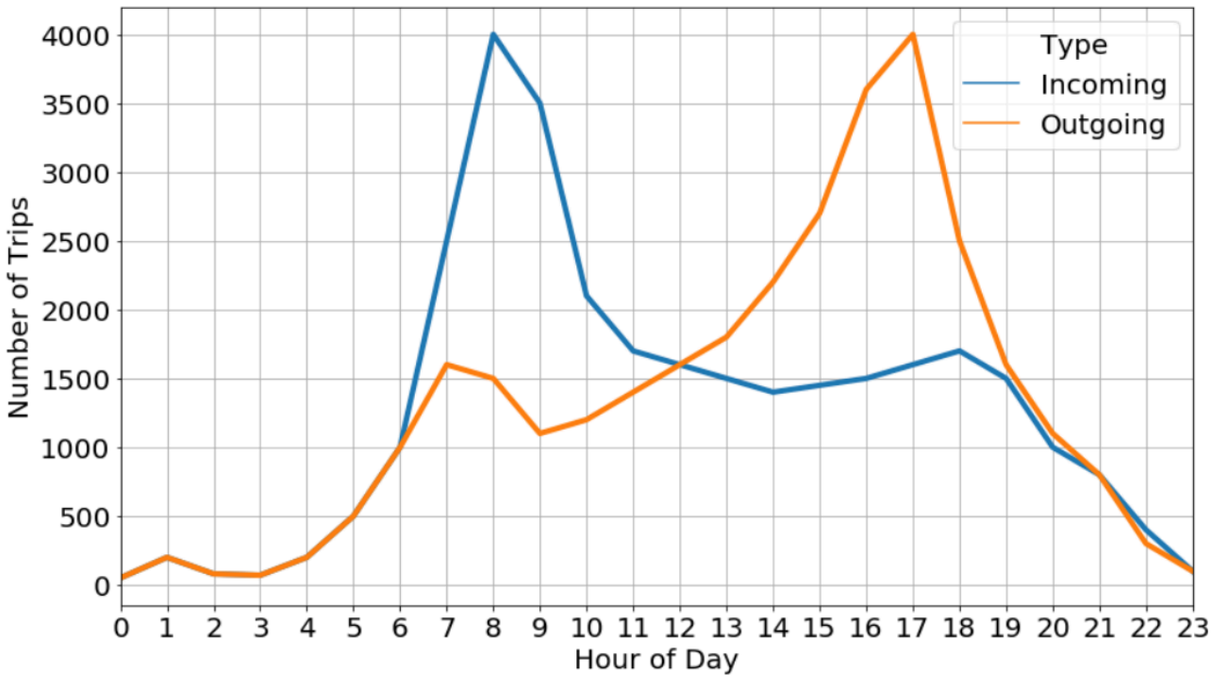
  - **Agent-based analysis**: another common attribute is unique identification (websiteId, userId, …). With these ID attributes, we can perform descriptive statistics per agent. Ex: average number of transaction on a specific website in X months, average number of unique customers per specific store in X months, …

Exploratory Data Analysis

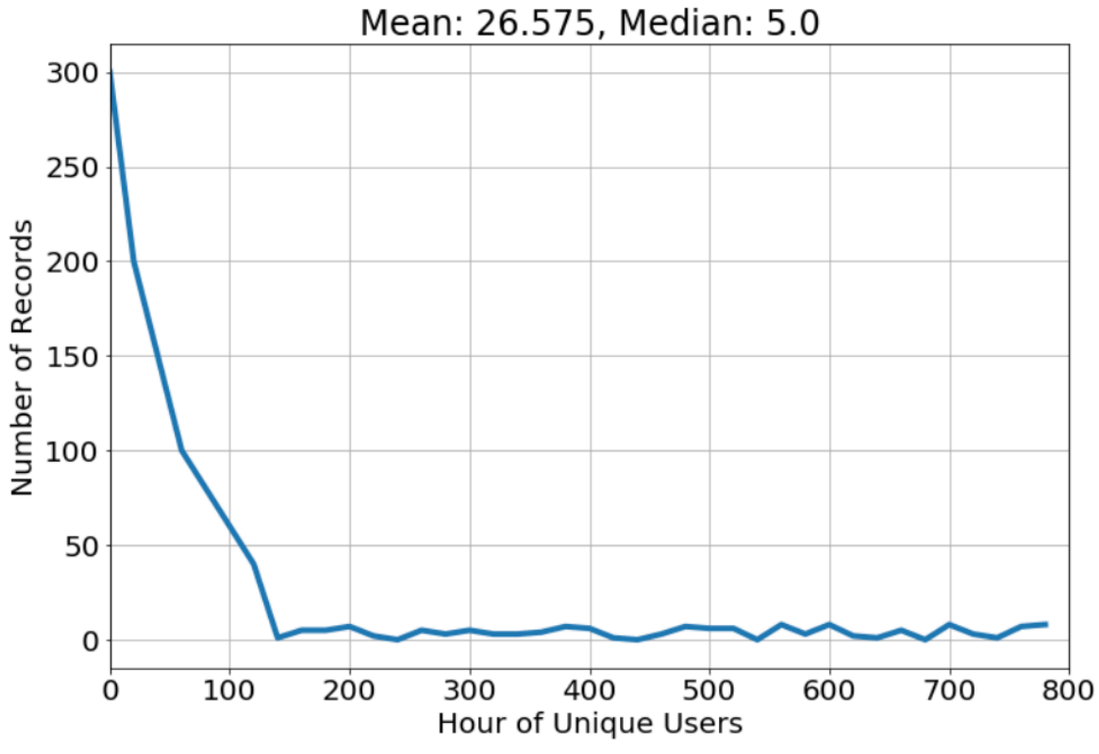
- **Data visualization** is essential in gaining **insights** from the analysis result

Exploratory Data Analysis

- **Data visualization** is essential in gaining **insights** from the analysis result



Average number of train trips per hour originating from and ending at one particular location



Histogram of the number of transactions (combined call, SMS, and internet) made per user in a week

Exploratory Data Analysis

- Let’s illustrate this with a concrete example: The census income dataset

workclass	education	marital_ status	occupation	relationship	race	sex	native_ country	income_ bracket
Private	9th	Married-civ-spouse	Other-service	Wife	Black	Female	United States	<=50K
Private	9th	Married-civ-spouse	Exec-managerial	Wife	Asian Pac Islander	Female	United States	>50K
Private	9th	Married-civ-spouse	Machine-op-inspct	Wife	White	Female	United States	>50K
Private	9th	Married-civ-spouse	Exec-managerial	Wife	White	Female	United States	<=50K
Private	9th	Married-civ-spouse	Tech-support	Wife	White	Female	United States	<=50K

age	functional_weight	education_num	capital_gain	capital_loss	hours_per_week
39	297847	5	3411	0	34
72	74141	5	0	0	48
45	178215	5	0	0	40
31	86958	5	0	0	40
55	176012	5	0	0	23

Exploratory Data Analysis

- **Descriptive** data analysis:

Feature	Type	% Missing
age	Integer	0
workclass	String	0
functional_weight	Integer	0
education	String	0
education_num	Integer	0
marital_status	String	0
occupation	String	0
relationship	String	0
race	String	0
sex	String	0
capital_gain	Integer	0
capital_loss	Integer	0
hours_per_week	Integer	0
native_country	String	0
income_bracket	String	0

Exploratory Data Analysis

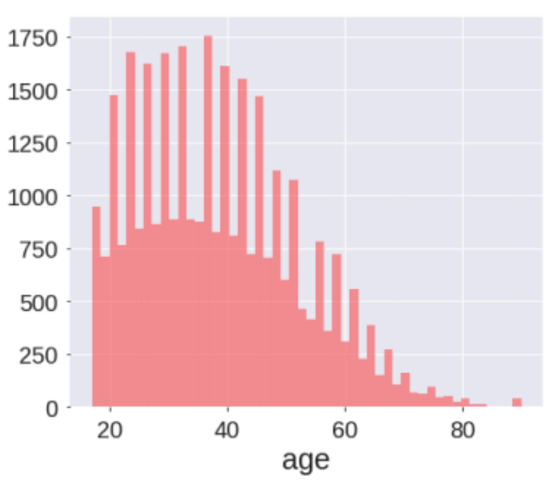
- **Descriptive** data analysis: numerical features

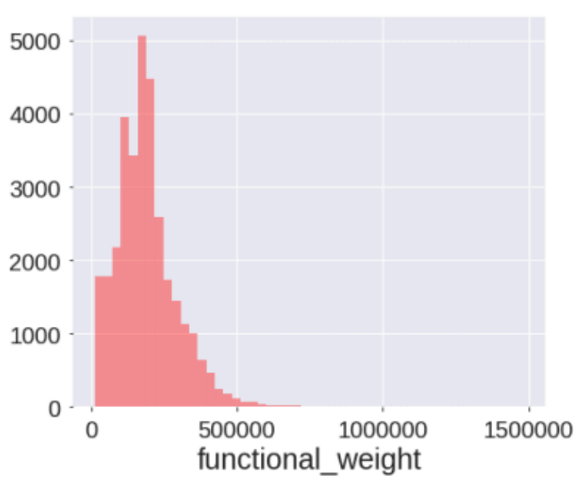
	max	range	IQR	mode	mad	kurtosis	skewness
age	90	73	20	36	11.189182	-0.166127	0.558743
functional_weight	1484705	1472420	119224	123011	77608.21854	6.218811	1.44698
education_num	16	15	3	9	1.903048	0.623444	-0.311676
capital_gain	99999	99999	0	0	1977.373437	154.799438	11.953848
capital_loss	4356	4356	0	0	166.462055	20.376802	4.594629
hours_per_week	99	98	5	40	7.583228	2.916687	0.227643

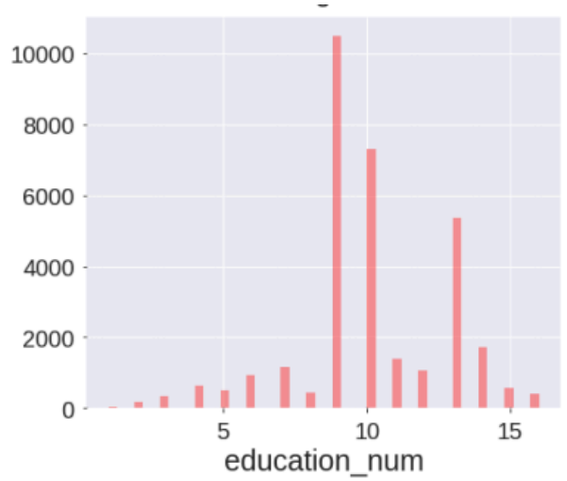
	mean	std	min	25%	50%	75%
age	38.581647	13.640433	17	28	37	48
functional_weight	189778.3665	105549.9777	12285	117827	178356	237051
education_num	10.080679	2.57272	1	9	10	12
capital_gain	1077.648844	7385.292085	0	0	0	0
capital_loss	87.30383	402.960219	0	0	0	0
hours_per_week	40.437456	12.347429	1	40	40	45

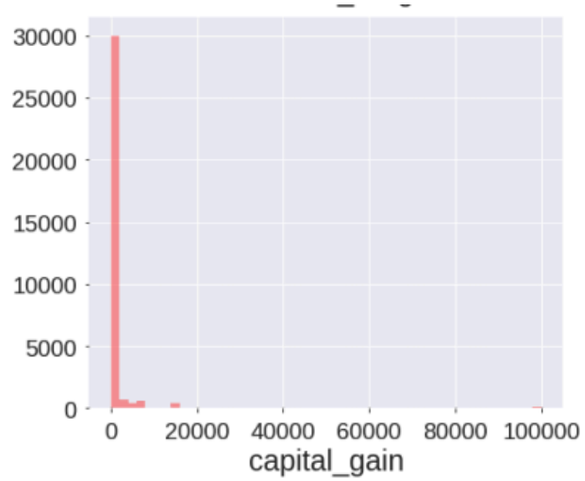
Exploratory Data Analysis

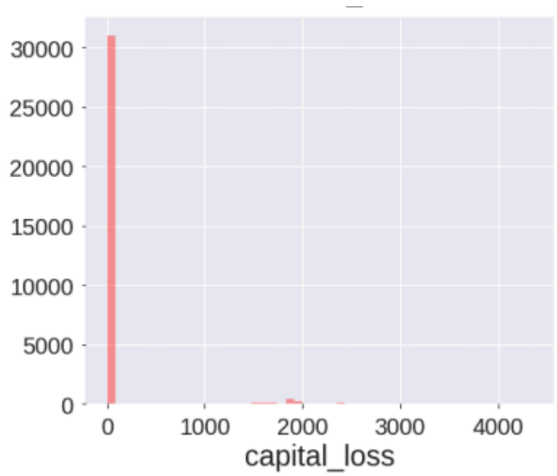
- **Descriptive** data analysis: numerical features

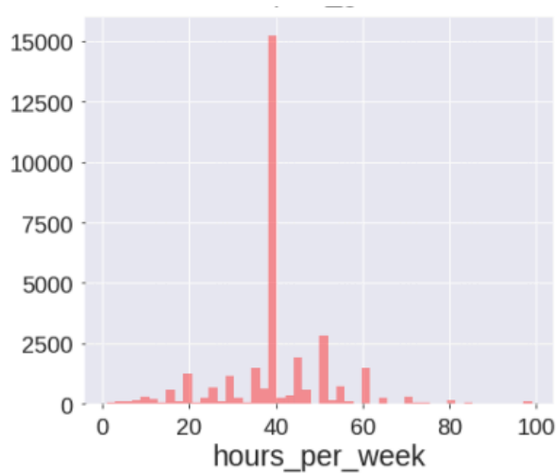






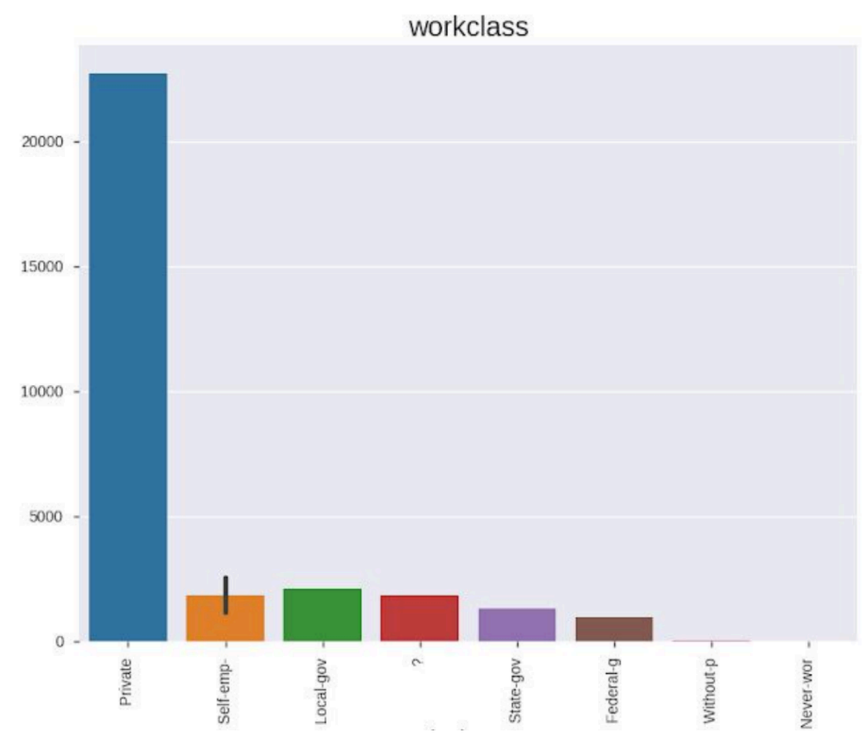


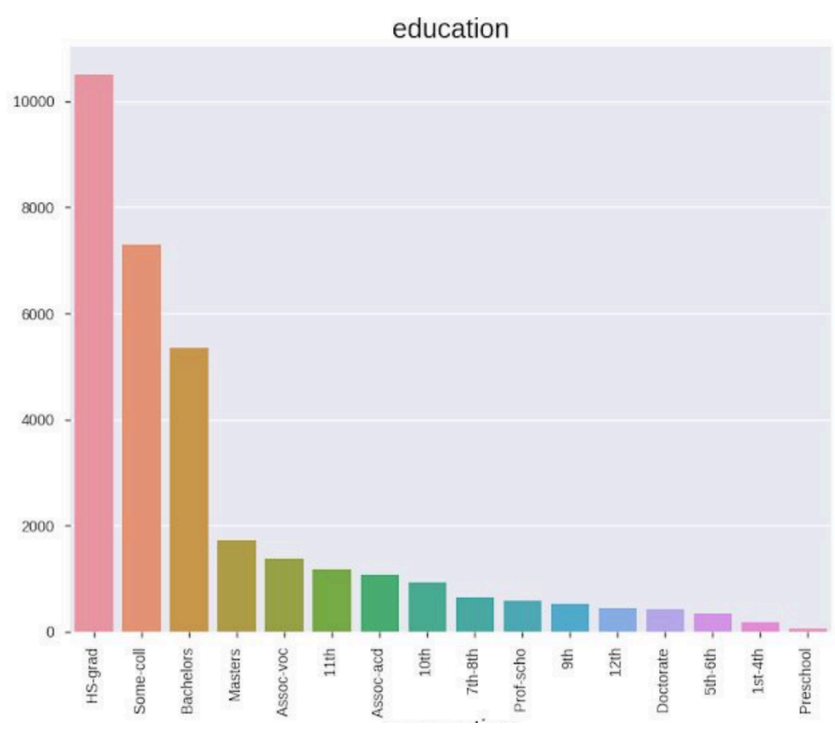


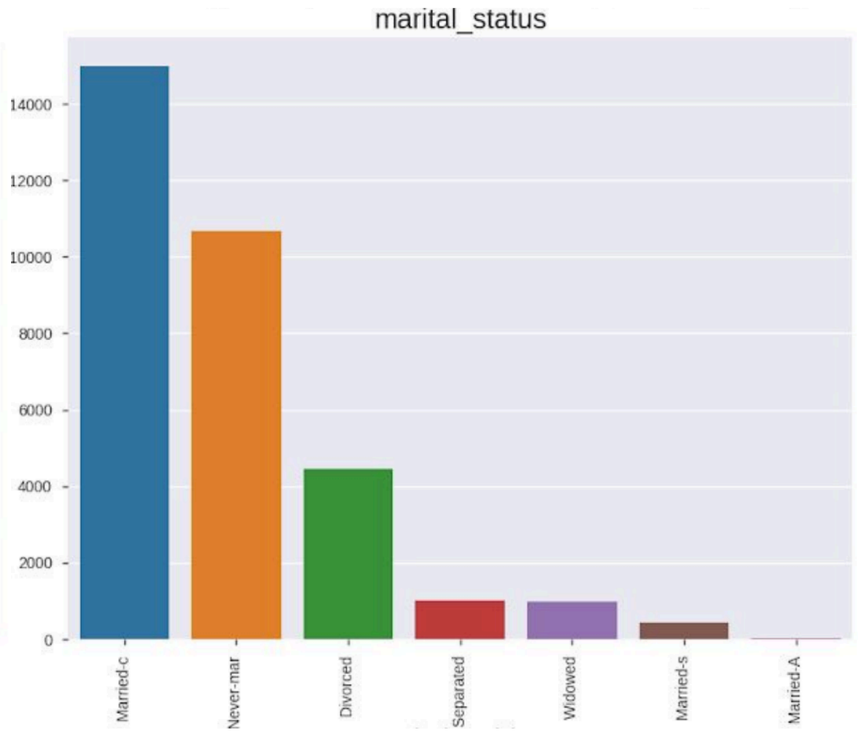


Exploratory Data Analysis

- **Descriptive** data analysis: categorical features

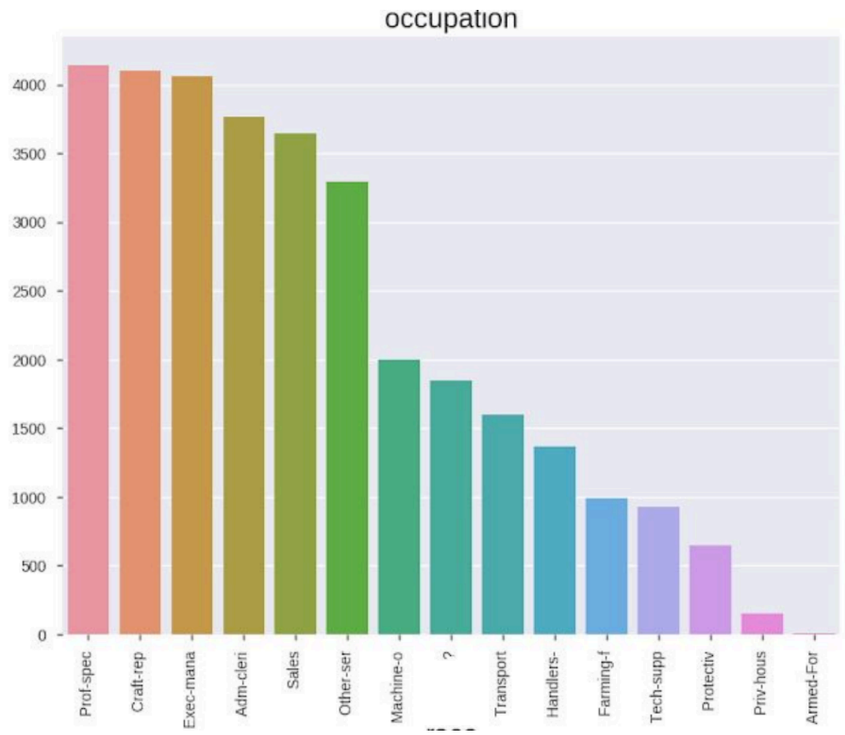


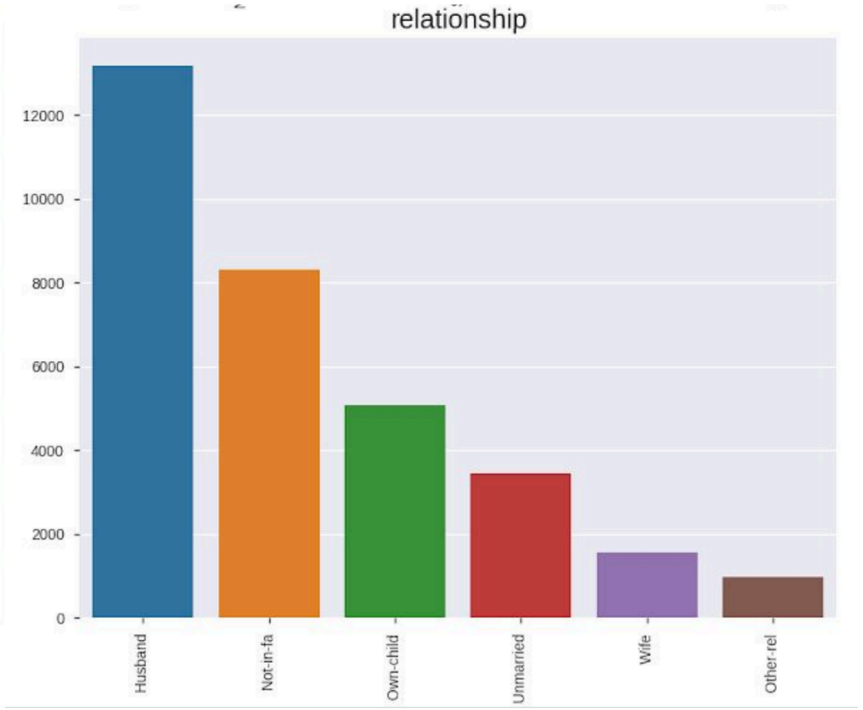


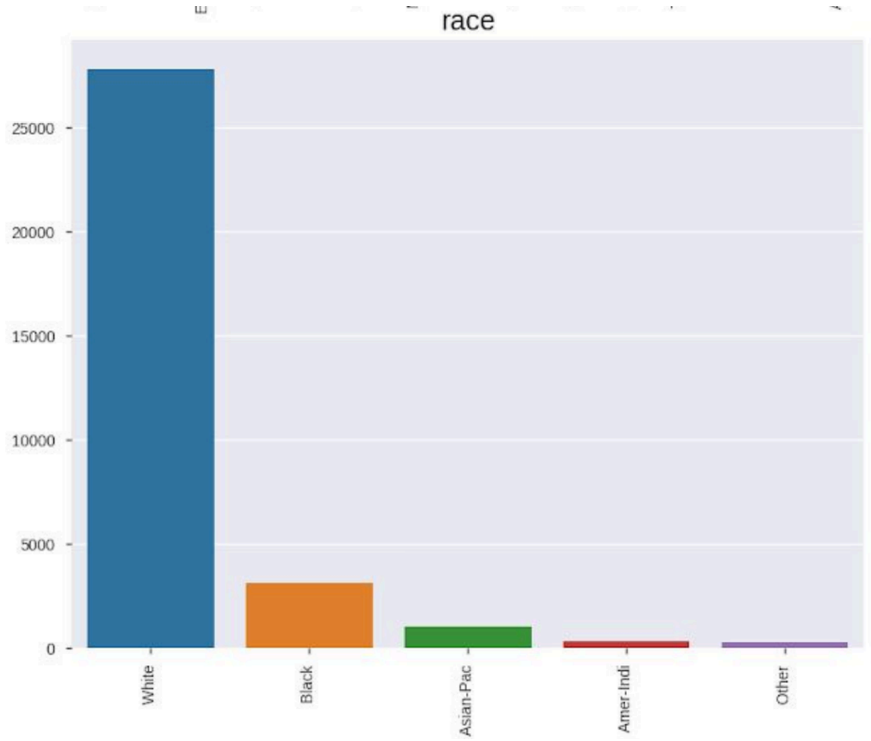


Exploratory Data Analysis

- **Descriptive** data analysis: categorical features







Exploratory Data Analysis

- **Correlation** analysis:

Categorical	Numerical	f-statistic	p-value
income_bracket	age	1886.707314	0.00E+00
income_bracket	functional_weight	2.915594	8.77E-02
income_bracket	education_num	4120.09578	0.00E+00
income_bracket	capital_gain	1709.150064	0.00E+00
income_bracket	capital_loss	754.830452	2.69E-164
income_bracket	hours_per_week	1813.386282	0.00E+00

Most of the numerical features are correlated with the target variable, except for functional_weight ($p > 0.05$)

cat1	cat2	chi_statistic	p_value	DoF
workclass	income_bracket	1045.7086	1.19E-210	18
education	income_bracket	4429.653302	0.00E+00	32
marital_status	income_bracket	6517.741654	0.00E+00	14
occupation	income_bracket	4031.97428	0.00E+00	30
relationship	income_bracket	6699.076897	0.00E+00	12
race	income_bracket	330.920431	4.43E-65	10
sex	income_bracket	1518.88682	0.00E+00	4
native_country	income_bracket	317.230386	8.56E-29	84

All categorical features seem correlated with the target

Feature selection

Feature selection

- As we've seen in the previous sections, there is a very large number of feature transformations we can try for a given model

Feature selection

- As we've seen in the previous sections, there is a very large number of feature transformations we can try for a given model
- Should we just throw all these features at the model?

Feature selection

- As we've seen in the previous sections, there is a very large number of feature transformations we can try for a given model

- Should we just throw all these features at the model?

Feature selection

- As we've seen in the previous sections, there is a very large number of feature transformations we can try for a given model

- Should we just throw all these features at the model?

*Hint: **G**arbage In, Garbage Out*

Feature selection

- We should **carefully select** our features because:

Feature selection

- We should **carefully select** our features because:
    - The **more features** we use in a model, the **more complex** it is. Remember: as the model's **complexity** increases, so does its **variance** and so does its tendency to **overfit**, thus **decreasing its performance**

Feature selection

- We should **carefully select** our features because:
    - The **more features** we use in a model, the **more complex** it is. Remember: as the model's **complexity** increases, so does its **variance** and so does its tendency to **overfit**, thus **decreasing its performance**
    - The **more features** we use in a model, the **less explicable/interpretable** it is

Feature selection

- We should **carefully select** our features because:
  - The **more features** we use in a model, the **more complex** it is. Remember: as the model's **complexity** increases, so does its **variance** and so does its tendency to **overfit**, thus **decreasing its performance**
  - The **more features** we use in a model, the **less explicable/interpretable** it is
  - The **more features** we use in a model, the **longer** it takes to train it

Feature selection

- We should **carefully select** our features because:
  - The **more features** we use in a model, the **more complex** it is. Remember: as the model's **complexity** increases, so does its **variance** and so does its tendency to **overfit**, thus **decreasing its performance**
  - The **more features** we use in a model, the **less explicable/interpretable** it is
  - The **more features** we use in a model, the **longer** it takes to train it
  - Simply because sometimes, **less is more**

Feature selection

- Feature selection helps achieving: **Less complexity, less overfitting, more accuracy, more interpretability and faster trainings**

Feature selection

- Feature selection helps achieving: **Less complexity, less overfitting, more accuracy, more interpretability and faster trainings**

- To do so, we can use:

Feature selection

- Feature selection helps achieving: **Less complexity, less overfitting, more accuracy, more interpretability and faster trainings**

- To do so, we can use:
  - Filter methods

Feature selection

- Feature selection helps achieving: **Less complexity, less overfitting, more accuracy, more interpretability and faster trainings**

- To do so, we can use:
  - Filter methods
  - Iterative subset selection methods (forward, backward, …)

Feature selection

- Feature selection helps achieving: **Less complexity, less overfitting, more accuracy, more interpretability and faster trainings**

- To do so, we can use:
  - Filter methods
  - Iterative subset selection methods (forward, backward, …)
  - Shrinkage methods (mainly lasso regularization, ridge regularization, …)

Feature selection

- **Filter methods**: choose features based on their score in various statistical tests for their **correlation** with the target variable

Feature selection

- **Filter methods**: choose features based on their score in various statistical tests for their **correlation** with the target variable

Feature selection

- **Filter methods**: choose features based on their score in various statistical tests for their **correlation** with the target variable

	Continuous Target	Categorical Target
Continuous Feature	Pearson's Correlation	LDA
Categorical Feature	Anova	Chi-Square

Feature selection

- **Filter methods**: choose features based on their score in various statistical tests for their **correlation** with the target variable

	Continuous Target	Categorical Target
Continuous Feature	Pearson's Correlation	LDA
Categorical Feature	Anova	Chi-Square

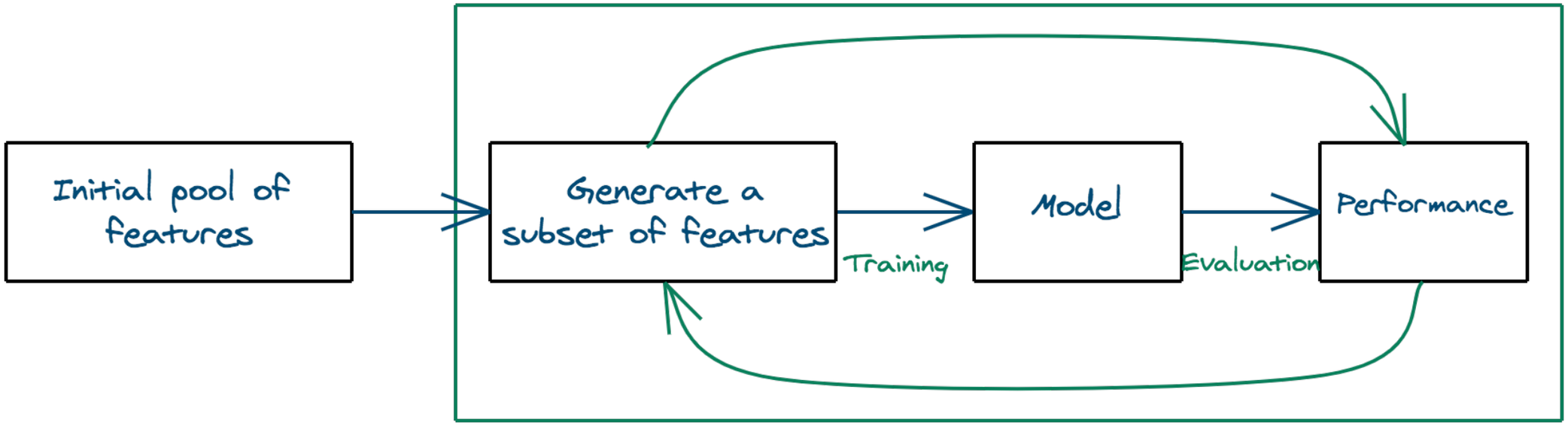
- **Pearson's correlation**: used as a measure of quantifying linear dependence between two variables
- **LDA**: Linear Discriminant Analysis used to find a linear combination of features that characterizes two or more classes
- **Anova**: (Analysis of Variance) A statistical test of wether the means of several classes are equal or not
- **Chi-Square**: used to determine wether a relationship between 2 categorical variables is likely to reflect a real association

Feature selection

- **Iterative subset selection** methods

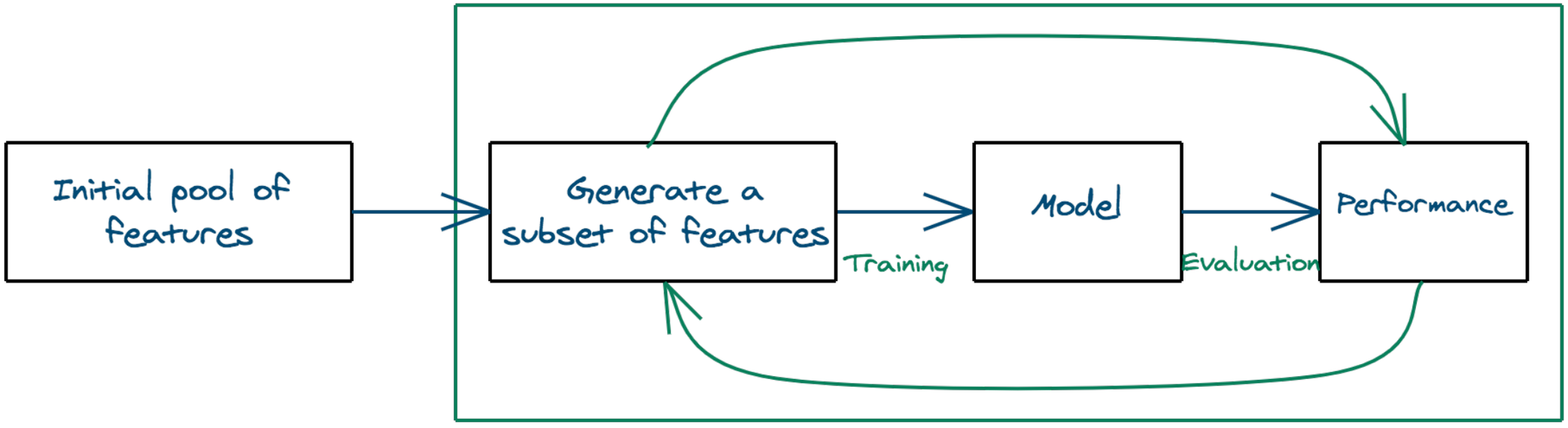
Feature selection

- **Iterative subset selection** methods



Feature selection

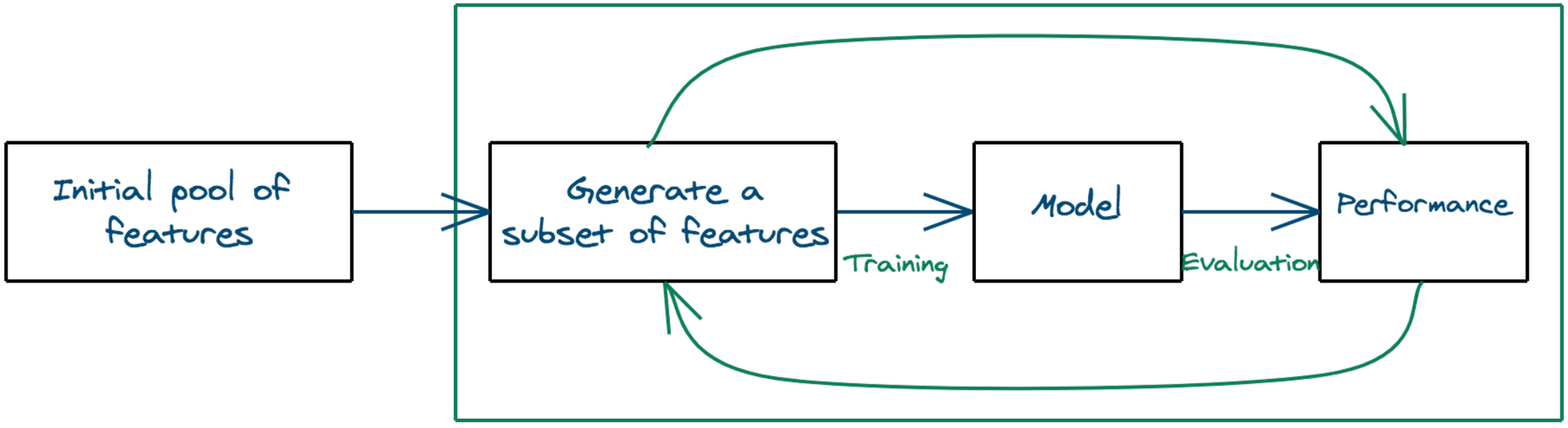
- **Iterative subset selection** methods



- To generate subsets of features, we can either employ:

Feature selection

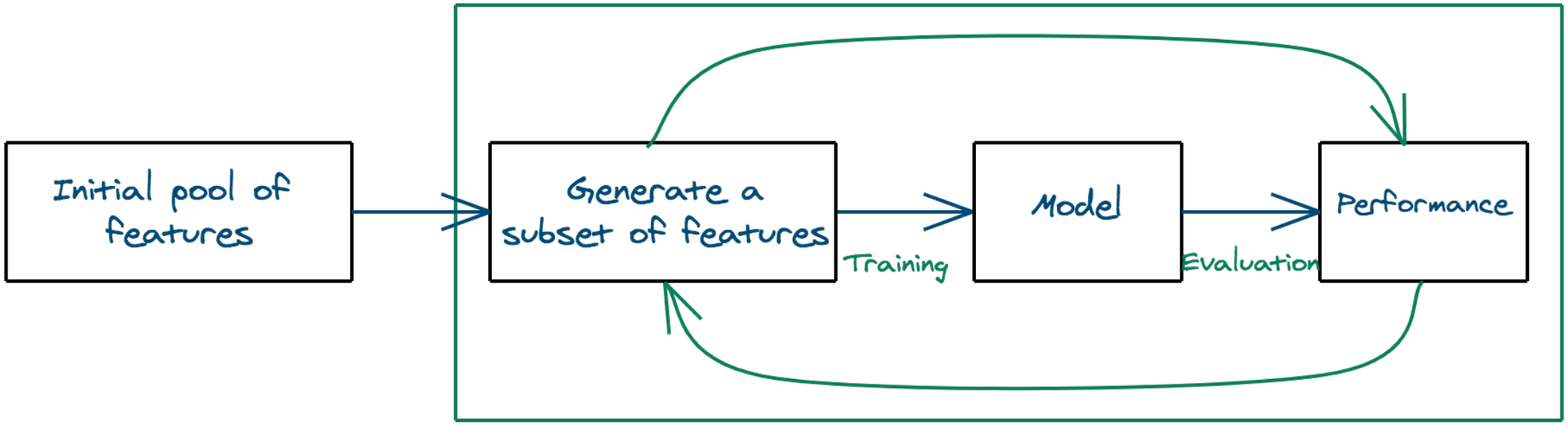
- **Iterative subset selection** methods



- To generate subsets of features, we can either employ:
  - **Forward** selection

Feature selection

- **Iterative subset selection** methods



- To generate subsets of features, we can either employ:
  - **Forward** selection
  - **Backward** Selection

Feature selection

- **Forward** selection:

Feature selection

- **Forward** selection:
  - Start with 0 features, only an intercept

Feature selection

- **Forward** selection:
  - Start with 0 features, only an intercept
  - At each iteration, we **add the feature that yields the best improvement** of the model

Feature selection

- **Forward** selection:
  - Start with 0 features, only an intercept
  - At each iteration, we **add the feature that yields the best improvement** of the model
  - We **repeat** that until **no improvement** is observed

Feature selection

- **Backward** selection:

Feature selection

- **Backward** selection:
  - Start with **all the features**

Feature selection

- **Backward** selection:
  - Start with **all the features**
  - At each iteration, we **drop the feature** whose removal yields the **best improvement** of the model

Feature selection

- **Backward** selection:
  - Start with **all the features**
  - At each iteration, we **drop the feature** whose removal yields the **best improvement** of the model
  - We **repeat** that until **no improvement** is observed