Applied Machine Learning for Business Analytics

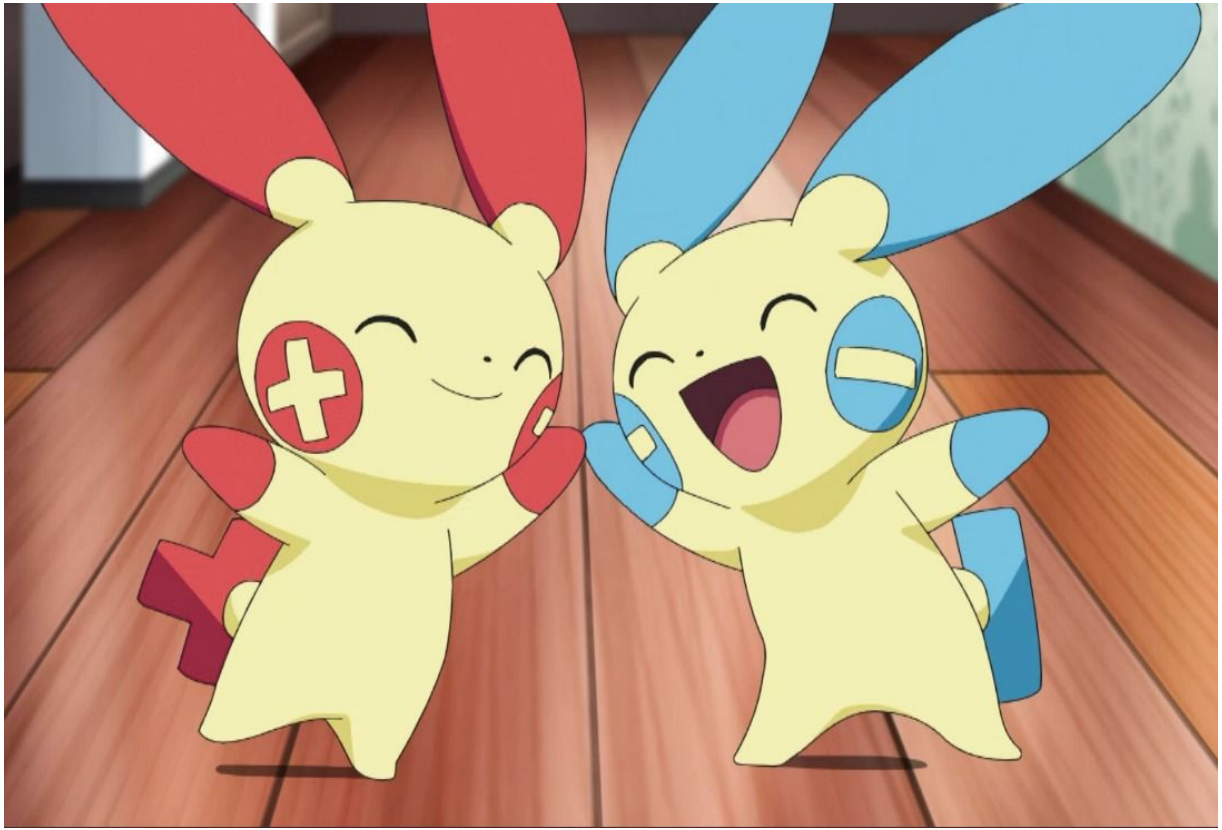
Lecture 2: Training Data Generation

Lecturer: Zhao Rui

Logistics

- HW1 will be due @ Jan 29, 23:59 pm
- For windows users, we did not figure out how to install torch appropriately which would be used in week 8. Therefore, google colab is suggested. Pls let our TA: Cungen know if you need any assistance

Happy Lunar New Year (Year of the Rabbit)



Myth #1: read_csv() is all you need

Course or Kaggle-style Machine Learning Problems:

1.1 Load the data

```
# Load dataset
dataset = pandas.read_csv("../input/Iris.csv")
```

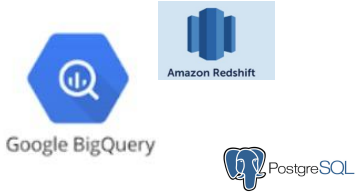
1.2 Manipulating the data

```
print(dataset.head(5))
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

Source: https://www.kaggle.com/code/gilsousa/prediction-iris-dataset

In most cases, (processed) data is tabular. It is usually stored in various kinds of database. Usually, you need to write SQL query



Myth #2: we can get training data easily

Ideally, you would like to write the following SQL query

```
-- SQL query you thought you would write
SELECT
    feature1,
    feature2,
    ...
    featureD,
    label
FROM
    Nonexistent_Table
```

This table does not exist. Instead, we need to write a super complex SQL query, that might manipulate, aggregates and join data from different tables

```
-- SQL query you end up writing
WITH table AS (
     SELECT ...
     FROM ...
     GROUP BY ...
)
,table2 AS (
     SELECT ...
     FROM ...
     GROUP BY ...
)
,table3 AS (
     SELECT ...
     FROM ...
)
SELECT
     feature1,
     feature2,
     ...
     featureD,
     label
FROM
    table1
    LEFT JOIN table2
    INNER JOIN table3
```

Automate SQL query execution with Python (via DB client libraries)

Generate training data: back-and-forth

- Generate training data is the first but crucial step in all ML projects
- Key steps in the process:
  - Data labeling
  - EDA
  - Preprocessing & Feature Engineering
  - Splitting
  - Augmentation
- In the whole process, we should also prevent data leakage

Agenda

1. Labeling
2. EDA
3. Preprocessing
4. Splitting
5. Augmentation
6. Data Leakage

1. Labeling

Labeling

- Labeling is the process of identifying the outputs for the inputs that are worth prediction/modeling
- Three main approaches:
  - Human annotation
  - Labeling function
  - ML-based Approaches: weak supervision, semi supervision, active learning, transfer learning

Human annotation

- Labeling in real-word is a workflow
  - Decide what needs to be labeled
  - Design the labeling interface
  - Set clear labeling instructions

Multiple open-source libraries

1. NLP Data: Doccano
2. Computer Vision: Labellmg

Labeling function

- Human Annotation
  - Expensive: require subject matter expertise (SME)
  - Non-private: data privacy
  - Slow: human efforts can not be scaled
  - Non-adaptive: every change requires re-labeling the dataset
- Programmatic labeling
  - We can encode SMEs' heuristics as functions and use them to label training data automatically
  - Snorkel is the data-centric AI platform that enable the process to generate training datasets without human efforts

Snorkel

https://snorkel.ai/

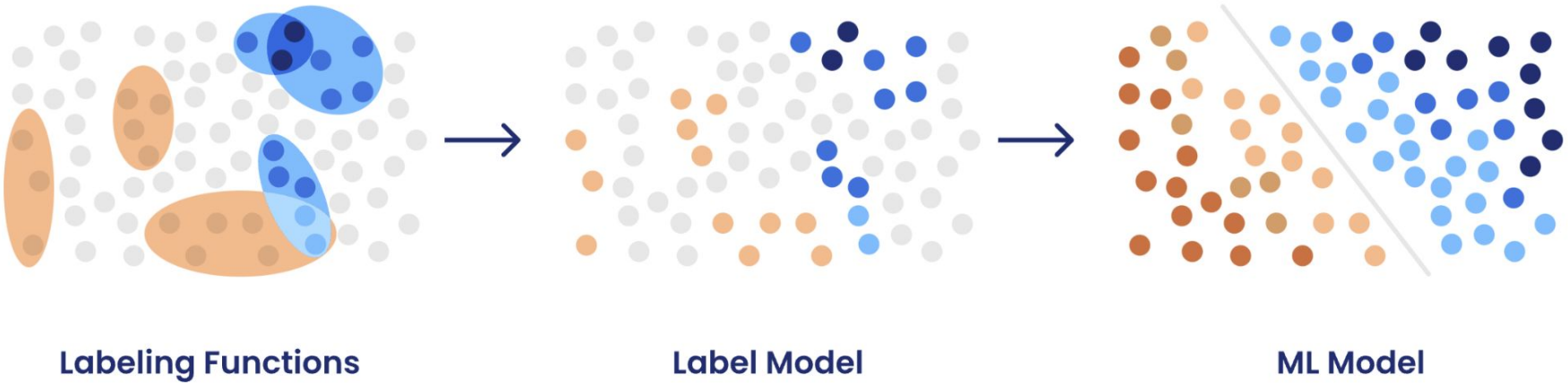
Many types of heuristics can be encoded

Pattern Matching	If a phrase like “send money” is in email
Boolean Search	If unknown_sender AND foreign_source
DB Lookup	If sender is in our Blacklist.db
Heuristics	If SpellChecker finds 3+ spelling errors
Legacy System	If LegacySystem votes spam
Third Party Model	If BERT labels an entity “diet”
Crowd Labels	If Worker #23 votes spam

Source: https://snorkel.ai/programmatic-labeling/

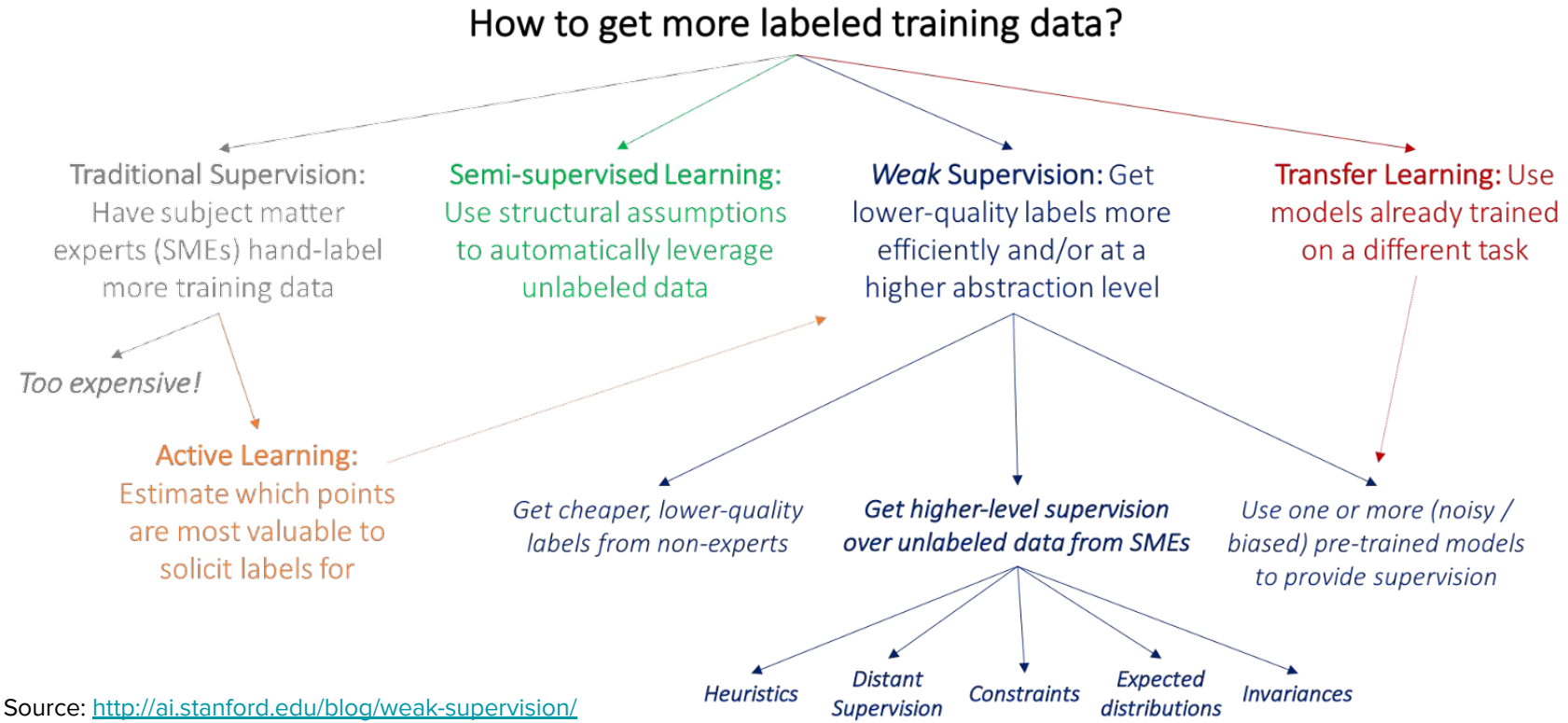
Labeling function

Labeling functions do not need to be comprehensive or cover your entire dataset.



Source: https://snorkel.ai/programmatic-labeling/

ML-based approaches for labeling



Source: http://ai.stanford.edu/blog/weak-supervision/

Active learning

- Active learning is trying to increase the efficiency of the human labeling process by selecting sub samples to be labeled
- The process usually work as follow:
  - Start with a small, initial dataset and label them to train the model
  - Ask the trained model to predict on some unlabeled data
  - Decide which new data points to be labeled based on various kinds of metrics:
    - Uncertainty measurement (entropy)
    - Candidate models' disagreement
  - Repeat until the desired performance is achieved

Active learning

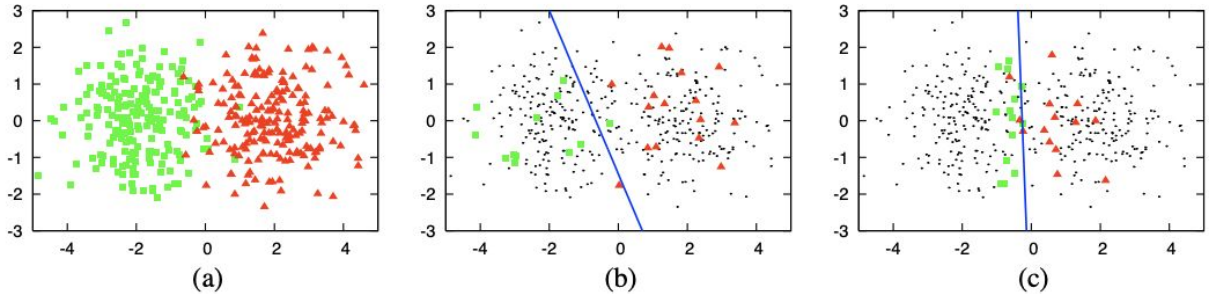


Figure 2: An illustrative example of pool-based active learning. (a) A toy data set of 400 instances, evenly sampled from two class Gaussians. The instances are represented as points in a 2D feature space. (b) A logistic regression model trained with 30 labeled instances randomly drawn from the problem domain. The line represents the decision boundary of the classifier (70% accuracy). (c) A logistic regression model trained with 30 actively queried instances using uncertainty sampling (90%).

Source: https://burrsettles.com/pub/settles.activelearning.pdf

modAL

Modular Active Learning framework for Python3

https://github.com/modAL-python/modAL

Semi supervision

- Use structural assumptions to label a large amount of unlabeled data together with a small amount of labeled data
  - Find the similarity between the unlabeled data and the labeled data -> Clustering algorithms
  - Generate labels of unlabeled data -> Self training

Semi supervision: Self training

- Train the model on the available labeled data (a small set)
- Use this model to generate predictions for unlabeled data
- Use predictions with **high raw probabilities** as labels (confident samples)
- Repeat step 1 with new labeled data

It is also called **pseudo labeling** (tricks to win Kaggle Competition):
https://www.kaggle.com/code/cdeotte/pseudo-labeling-qda-0-969

Weak supervision

- Leverage noisy, imprecise sources to create labels
- Labeling functions belong to weak supervision

Transfer learning

- Apply model trained in one domain to another domain
  - From image classification to image tagging
- The details would be covered in the following lectures

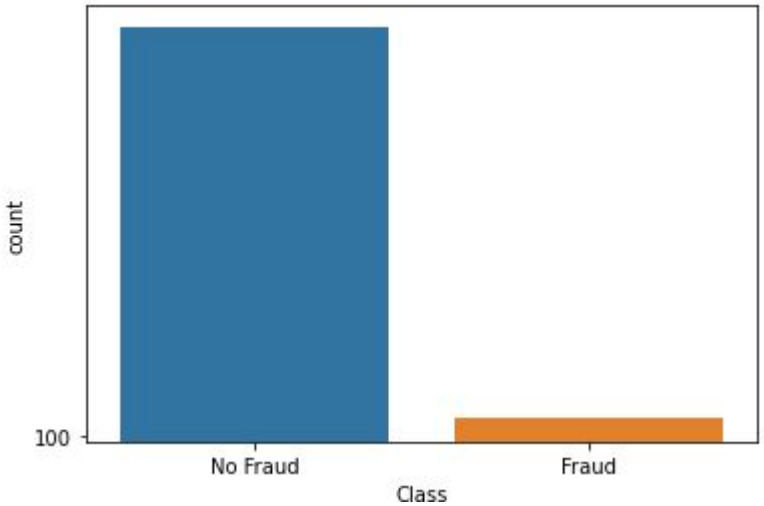
2. EDA

Exploratory data analysis

EDA is used to gain more understandings from our datasets.

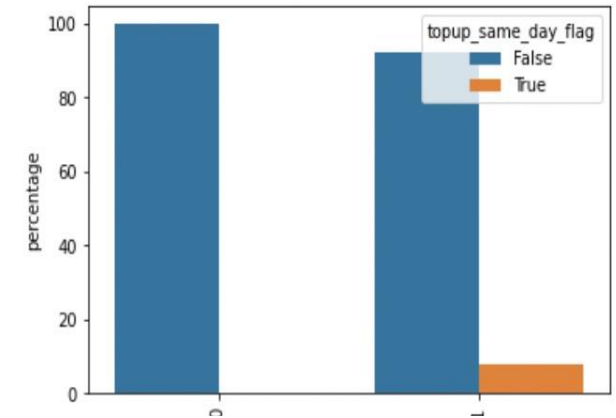
- It is the cyclical process that can be done at any steps of the machine learning projects' lifecycle.
- Use EDA to answer important questions and to make it easier to extract insight

Do we have imbalanced problems?

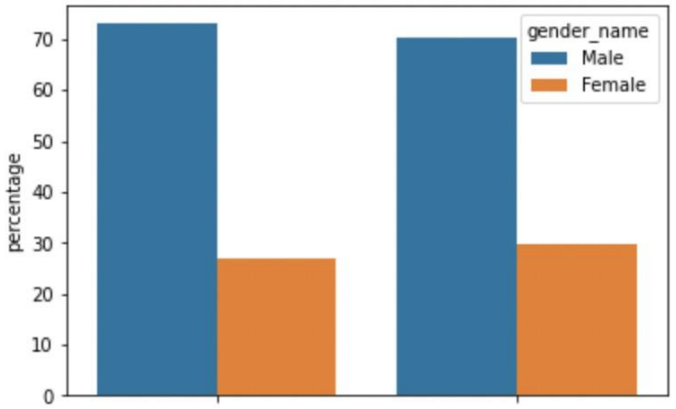


Should we use this feature?

- Predict Tranx. Probabilities for Onboarding Users



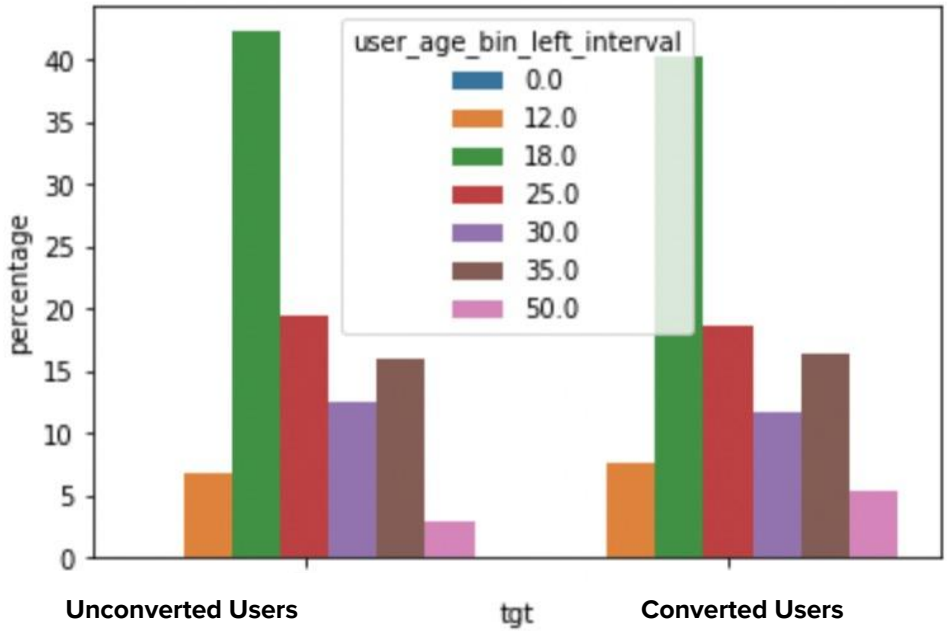
Unconverted Users Converted Users



Unconverted Users Converted Users

EDA for continuous features

User Age Bin (legends = left interval limit, unit in years)



3. Preprocessing

Recall that computers only understand numbers

Preprocessing

Data preprocessing has two kinds of processes:

- Preparation
  - Data Cleaning
  - Missing Values
  - Outlier Removing
  - Feature engineering
- Transformation
  - Scaling
  - Encoding

In those steps, we should be careful about curse of dimensionality and data leakage

3. 1 Preparation

Data cleaning

Based on domain expertise and EDA, we apply constraints on data to make it easier for the following machine learning model to learn the pattern:

- Image: Crop, resize, clip
- Text: lower, stem, lemmatize, regex, remove stopwords

```
"""Clean raw text."""
# Lower
if lower:
    text = text.lower()

# Remove stopwords
if len(stopwords):
    pattern = re.compile(r'\b(' + r"|".join(stopwords) + r")\b\s*")
    text = pattern.sub('', text)
```

Data missing

- Data missing has different reasons
  - Missing at random (MAR)
  - Missing not at random (MNAR)
  - Missing completely at random (MCAR)

https://en.wikipedia.org/wiki/Missing_data

ID	Age	Gender	Annual income	Job	Buy?
1		A	150,000	Engineer	No
2	27	B	50,000	Teacher	No
3		A	100,000		Yes
4	40	B		Engineer	Yes
5	35	B		Doctor	Yes
6		A	50,000	Teacher	No

MAR

Missing at random – the missing data is related to another observed variable

ID	Age	Gender	Annual income	Job	Buy?
1		A	150,000	Engineer	No
2	27	B	50,000	Teacher	No
3		A	100,000		Yes
4	40	B		Engineer	Yes
5	35	B		Doctor	Yes
6		A	50,000	Teacher	No

MNAR

Missing not at random – the data missing is related to the value itself

ID	Age	Gender	Annual income	Job	Buy?
1		A	150,000	Engineer	No
2	27	B	50,000	Teacher	No
3		A	100,000		Yes
4	40	B	($350,0000?)	Engineer	Yes
5	35	B	($350,0000?)	Doctor	Yes
6		A	50,000	Teacher	No

MCAR

Missing completely at random – there is no pattern to which values are missing

ID	Age	Gender	Annual income	Job	Buy?
1		A	150,000	Engineer	No
2	27	B	50,000	Teacher	No
3		A	100,000		Yes
4	40	B		Engineer	Yes
5	35	B		Doctor	Yes
6		A	50,000	Teacher	No

Handling missing values

- Deletion – removing data with missing entries
- Imputation – filling missing fields with certain values

Handling missing values

- Deletion
  - Column deletion – remove columns with too many missing entries
    - drawbacks – even if half the values are missing, the remaining data still potentially useful information for predictions
    - e.g. even if over half the column for ‘Marital status’ is missing, marital status is still highly correlated with house purchasing
  - Row deletion

Marital status
Married
Single
Single

Handling missing values

- Row deletion
  - Good for: data missing completely at random (MCAR) and few values missing

ID	Age	Gender	Annual income	Job	Buy?
1	39	A	150,000	Engineer	No
2	27	B	50,000	Teacher	No
3		A	100,000		Yes
4	40	B	75,000	Engineer	Yes
5	35	B	35,000	Doctor	Yes
6	32	A	50,000	Teacher	No
7	33	B	60,000	Teacher	No
8	20	B	10,000	Student	No

Handling missing values

- Row deletion
  - Bad when many examples have missing fields

ID	Age	Gender	Annual income	Job	Buy?
1		A	150,000	Engineer	No
2	27	B	50,000	Teacher	No
3		A	100,000		Yes
4	40	B		Engineer	Yes
5	35	B		Doctor	Yes
6		A	50,000	Teacher	No
7	33	B	60,000	Teacher	No
8	20	B	10,000	Student	No

Handling missing values

- Row deletion
  - Bad for: missing data at random (MAR)
  - Can potentially bias data – we've accidentally removed all examples with gender 'A'

ID	Age	Gender	Annual income	Job	Buy?
1		A	150,000	Engineer	No
2	27	B	50,000	Teacher	No
3		A	100,000		Yes
4	40	B		Engineer	Yes
5	35	B		Doctor	Yes
6		A	50,000	Teacher	No
7	33	B	60,000	Teacher	No
8	20	B	10,000	Student	No

Handling missing values

- Row deletion
  - Bad for: missing values are not at random (MNAR)
  - Missing information is information itself

ID	Age	Gender	Annual income	Job	Buy?
1		A	150,000	Engineer	No
2	27	B	50,000	Teacher	No
3		A	100,000		Yes
4	40	B	($350,000?)	Engineer	Yes
5	35	B	($350,000?)	Doctor	Yes
6		A	50,000	Teacher	No
7	33	B	60,000	Teacher	No
8	20	B	10,000	Student	No

Handling missing values: Imputation

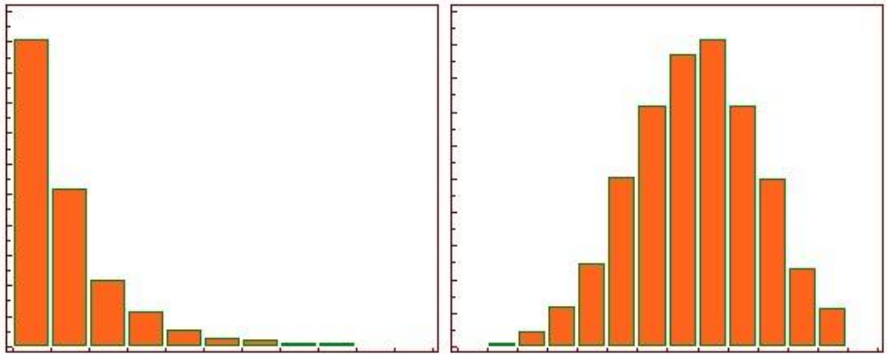
- Fill missing fields with certain values
  - Defaults
    - E.g. 0, or the empty string, etc.
  - Statistical measures – mean, median, mode
    - e.g. if a day in July is missing its temperature value, fill it with the median temperature in July

Outliers removing

- Formulate assumptions about what would be the normal expected value
  - Like remove the data whose feature value is over 2 standard deviations

```
df[np.abs(df.A - df.A.mean()) <= (2 * df.A.std())]
```

  - Sometimes, values might not be outliers when the transformation is applied (like log-transform)



What is feature engineering

- Feature engineering:
  - Extract features to use in your model
  - How to represent examples by the feature vectors?

Feature engineering

- Core Question:
  - What properties of x **might be** relevant for predicting y?

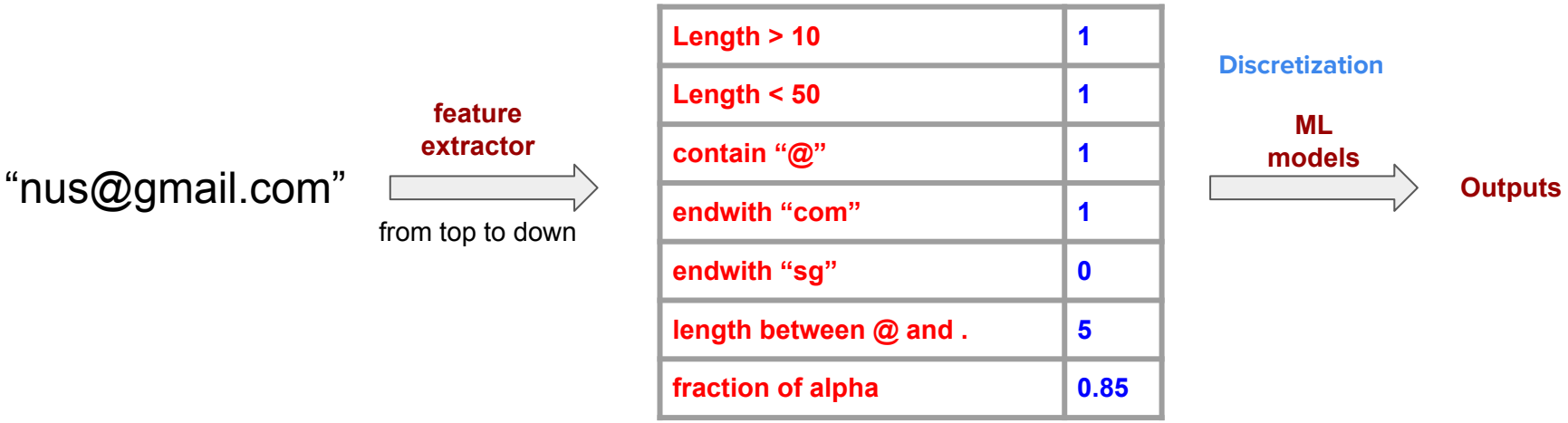
A “Real” machine learning task

- Example Task: Predict y, whether a string x is an email address
  - x: “diszr@nus.edu.sg” **y:1**
  - x: “nusmsba” **y:0**
  - x: “@trump” **y:0**
- Question: What properties of x might be relevant for predicting y?
- Feature extractor: Given input x, output a set of (feature name, feature value) pairs

“nus@gmail.com”  A **fixed-length** vector

Feature engineering

- Question: What properties of x might be relevant for predicting y?
- Feature extractor: Given input x, output a set of (feature name, feature value) pairs



Can we use length directly?

Engineered features

- For text data: BoW Models

Stopword removal I have a dog. He's sleeping.

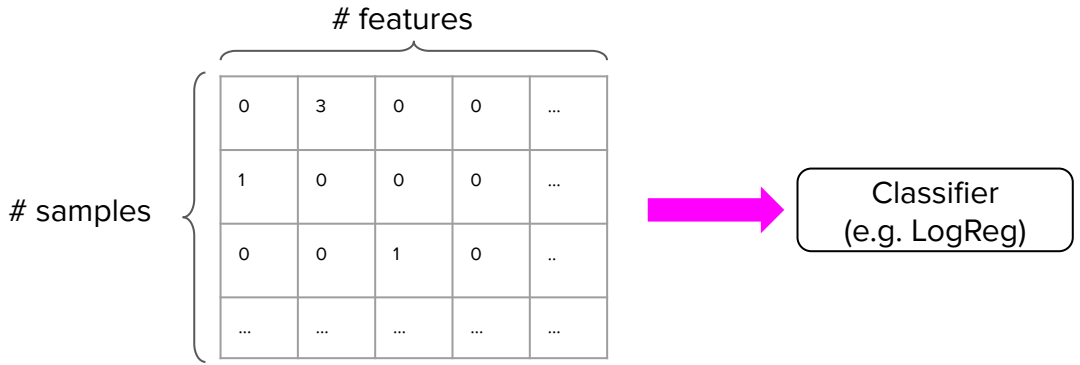
Lemmatization I have dog. He's sleeping.

Contraction I have dog. He's sleep.

Punctuation I have dog. He is sleep.

Lowercase I have dog He is sleep

N-gram i have dog he is sleep

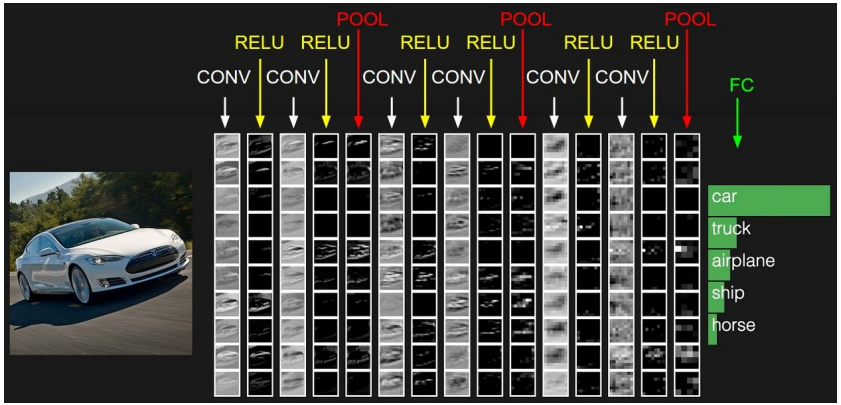


Features

I	you	have	dog	cat	he	she	is	they	sleep	I, have	have, dog	good, dog	...
1	0	1	1	0	1	0	1	0	1	1	1	0	...

Representation learning

- Using Deep Learning Approach:
  - CNN, RNN, Attention Models
  - Learn representations from text, image, video, audio signals



http://cs231n.github.io/convolutional-networks/

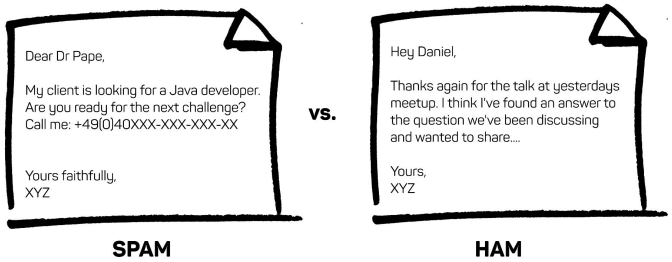
Feature engineering

- In papers, deep learning papers promise no more feature engineering
  - We are still very far from that point
  - Deep learning are not the first choice in industry for many applications

Spam classification

Except BoW Features:

- Post repetitiveness
- Language detection, typos, abnormal punctuations, ratio uppercase/lowercase
- IP, other users from the same IP
- Blacklisted links
- Targeted users
- ...



Feature engineering

- For complex tasks, number of features can go up to millions or billions!
- Lots of ML production work involves coming up with new features
  - Fraudsters come up with new techniques very fast, so need to come up with new features very fast to counter
- Often require subject matter expertise
- Good Habits: Know your data
  - Visualize: Plot Histograms, Rank Most to least common value
  - Debug: Duplicate examples? Missing Values? Outliers? Data Agrees with dashboards? Training and Validation data similar?
  - Monitor: Feature quantiles

3. 2 Transformation

Scaling

- Necessary for some models where the scale of feature affects the process
- Only learn “parameters” from **train split** and apply to **all splits**

Which ml models require feature scaling: https://www.quora.com/Which-machine-learning-algorithms-require-feature-scaling

Types of scaling

scaling type	use case
min/max normalization	Any -- no assumptions about variables
z-score normalization	When variables follow a normal distribution
log scaling	When variables follow an exponential distribution
binning	convert a continuous feature into categorical using bins

Feature scaling

- Min-max Scaler:

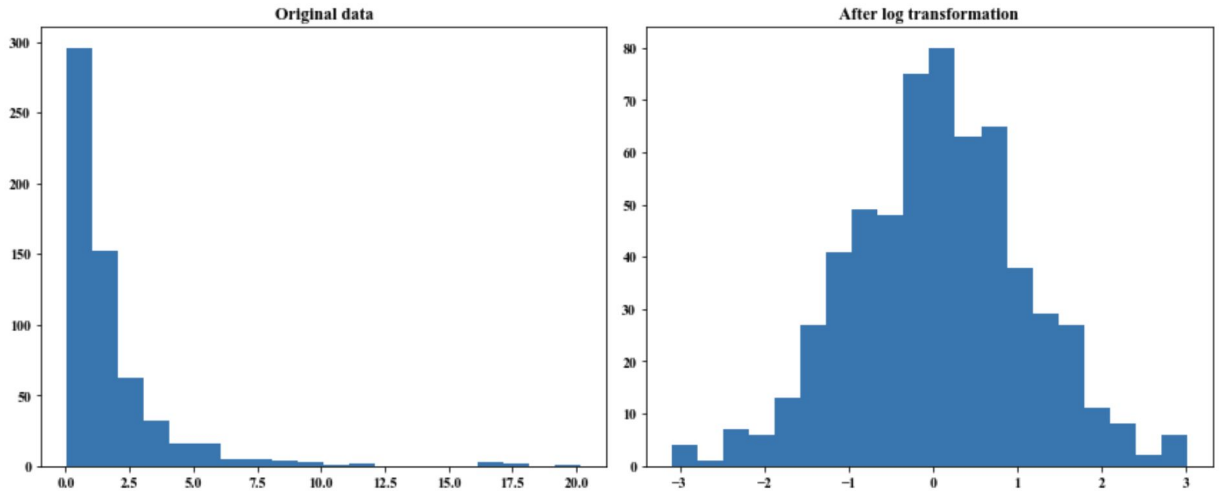
$$\hat{x} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

- Z-score transformation:

$$\hat{x} = \frac{x - x_{mean}}{\sigma}$$

Log scaling

- Help with skewed data
- Often gives performance gain



Binning

- Turning a continuous feature into a discrete feature (quantization)
- Create buckets for different ranges
  - Incorporate knowledge/expertise about each variable by constructing specific buckets
- Examples
  - Income
    - Lower income: x < $35,000
    - Middle income: $35,000 <= x < $100,000
    - High income: x >= $100,000
  - Age
    - Minors: x < 18
    - College: 18 <= x < 22
    - Young adult: 22 <= x < 30
    - 30 <= x < 40
    - 40 <= x < 65
    - Seniors: x >= 65

Encoding

- Label: unique index for categorical value
- One-hot: convert categorical value into binary vector
- Embeddings: dense vectors capturing context

Curse of dimensionality

High dimensionality alway occur with sparsity. The dimensionality come from two aspects:

- Number of features
  - Number of features
  - PCA can be used to linearly project the data into a lower dimensional space
- Feature value dimensionality:
  - The number of unique values per that feature

Feature value dimensionality

When a feature has lots of unique values and few data points for each unique value across the whole dataset. For example, userid for each user, URL for each webpage. The solution would be encoding

- Encode those less frequent features into more frequent features:
  - Binning
  - Extract general attributes
    - Userid -> user profile data
    - URL -> domain
  - High frequent n-grams for BOW

Global vs Local preprocessing

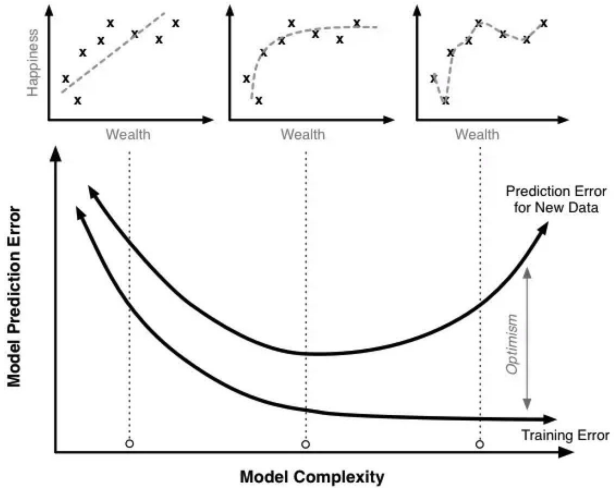
Preprocessing steps can be categorized into:

- Global
  - Do not depend on the dataset
  - E.g. lower casing text, removing stop words
- Local
  - The parameters in the processing steps are learned/obtained only from the training **split**
  - E.g. build vocabulary, scaling

4. Splitting

Generalization

- In ML, a model is used to fit the data
- Once trained, the model is applied upon new data
- Generalization is the prediction capability of the model on live/new data



We need an **unbiased measuring approach** to determine the performance of our models.

Partition our whole training data

The performance on this set would be checked to tune model hyperparameters

Training Set

Validation Set

Test Set

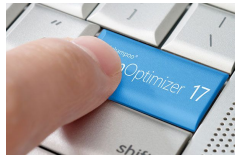
The model will use it to optimize its model internal weights

Our final measure of how the model may behave on new, unseen data

Hyperparameters

- Machine learning algorithms usually have two kinds of weights:
  - ***Parameters***: learned by data during training such as slope of linear regression, layer weights of neural networks
  - **Hyperparameters**: left to us to select beforehand such as K in KNN, number of layers in neural networks

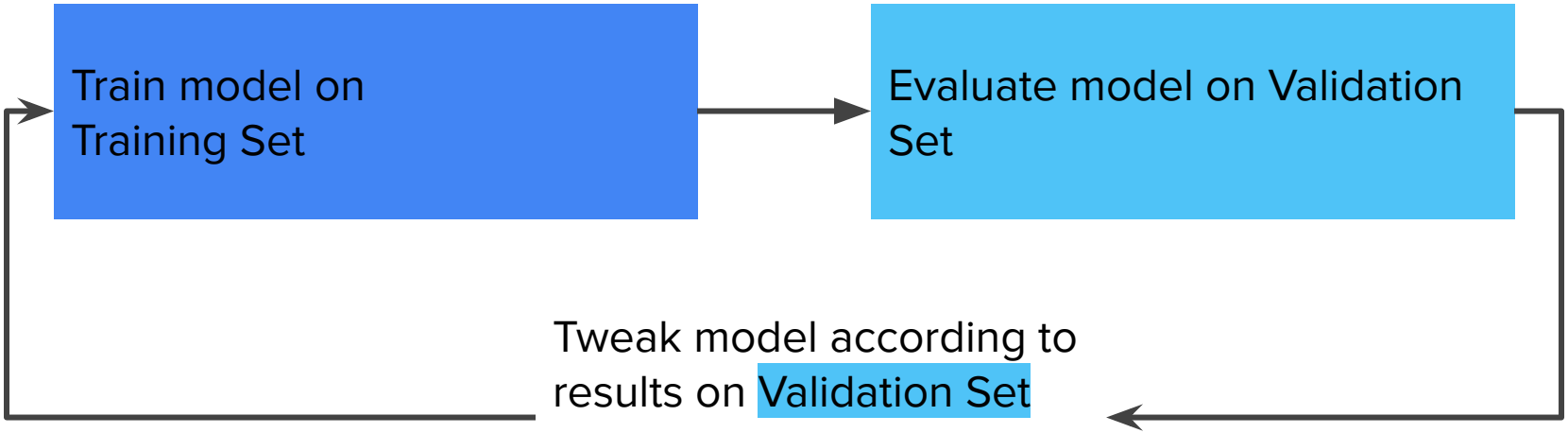
Hyperparameters



Parameters

Scores

Better workflow: use a validation set



Pick model that does best on Validation Set
Confirm results on Test Set (one time assessment of the model)

Proper data splits

- The dataset should be representative of data we will address
- Equal class distribution across all splits
- Shuffle data randomly but be careful about data leaks
- The order of splits to prevent data leakage:
    - Preprocessing (global) -> Splitting -> Preprocessing (local)

5. Augmentation

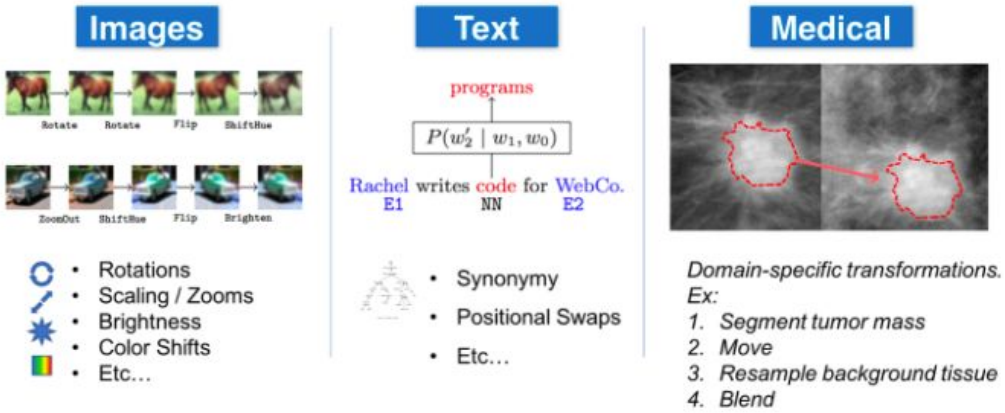
Data augmentation

Using existing samples, we generate synthetic, yet realistic samples. Here, realistic means that the augmented data samples should be close to the samples that our deployed model could address in production

- Split data firstly
- Augment the training split only
- QA
    - It is important to validate the quality of augmented data. Data validation test or specific tests can be applied to check

Data augmentation

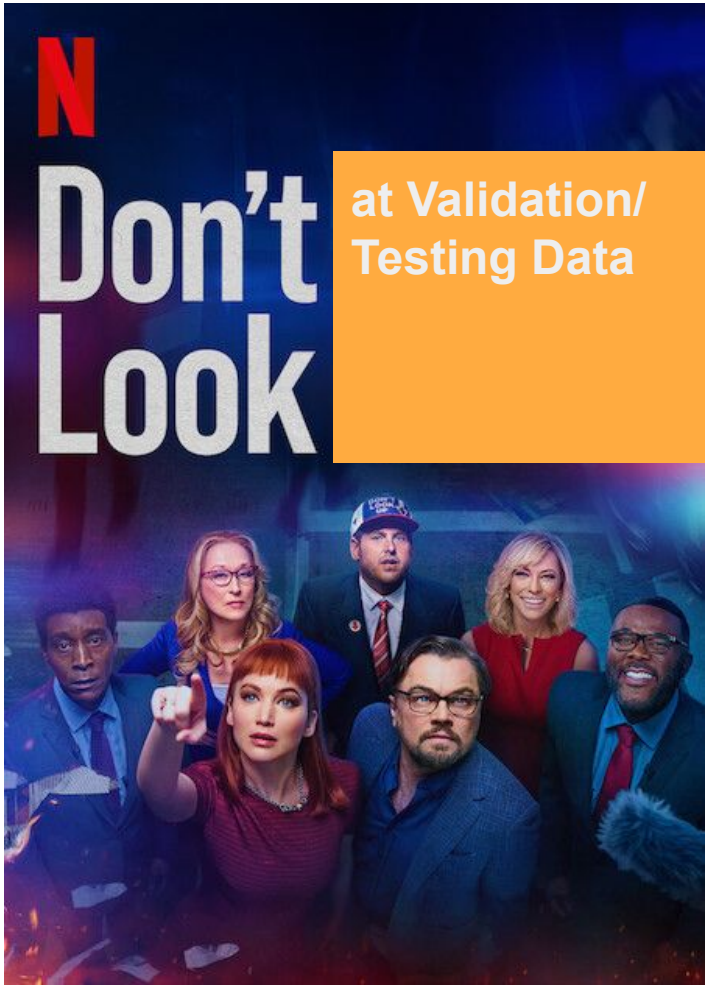
- For structured data (tabular data):
  - Add random noise
  - Synthetic oversampling (SMOTE)
- For unstructured data (like text & images):



Libraries

- NLP
  - NLPAug
  - TextAttack
  - TextAugment
- CV
  - Imgaug
  - Albumentations
  - Augmentor
- Other
  - Snorkel: weak supervision
  - DeltaPy: tabular data
  - Audiomentations: audio data
  - Tsaug: time series data

6. Data Leakage


N
Don't Look
at Validation/
Testing Data

Data leakage

It happens when the training data contains information about labels, but similar data is not available when the model is used for prediction

- Leakage makes the model to look accurate on the training set while the model will perform poorly in production
- There are two main types of leakage: train-test contamination and target leakage

Train-test contamination

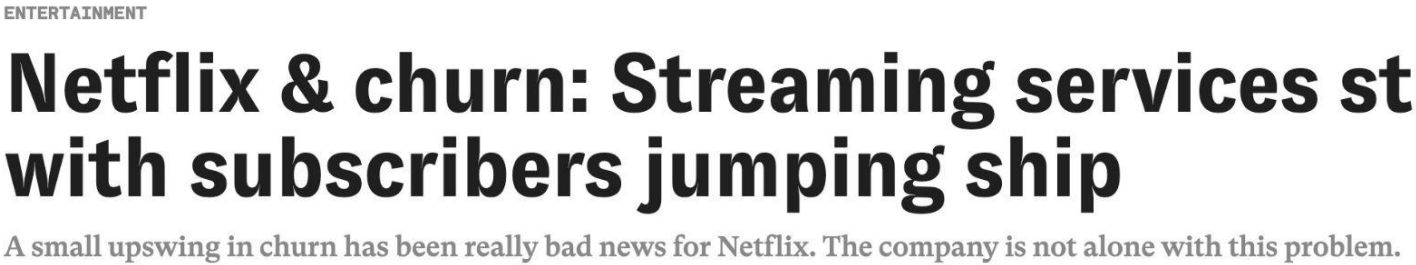
- Training-test contamination: we are not careful to distinguish training data from validation data
  - Oversampling before splits
    - Training data may overlap with testing data
  - Prepare features on the entire data instead of just training data
    - Create vocab/preprocessing scaler from train+test data
  - Group leakage
    - A patient has 2 CT scans, 1 in train, 1 in test.
- Pipelines should be built to split training and validation data carefully

Target leakage

- Target leakage
  - Some form of the label “slip” into the features
  - This same information is not available during inference
  - It usually happens in the timing order

Target leakage example I: churn prediction

- Build an ML model to predict which users will cancel memberships in the following one week



ENTERTAINMENT

Netflix & churn: Streaming services struggle with subscribers jumping ship

A small upswing in churn has been really bad news for Netflix. The company is not alone with this problem.

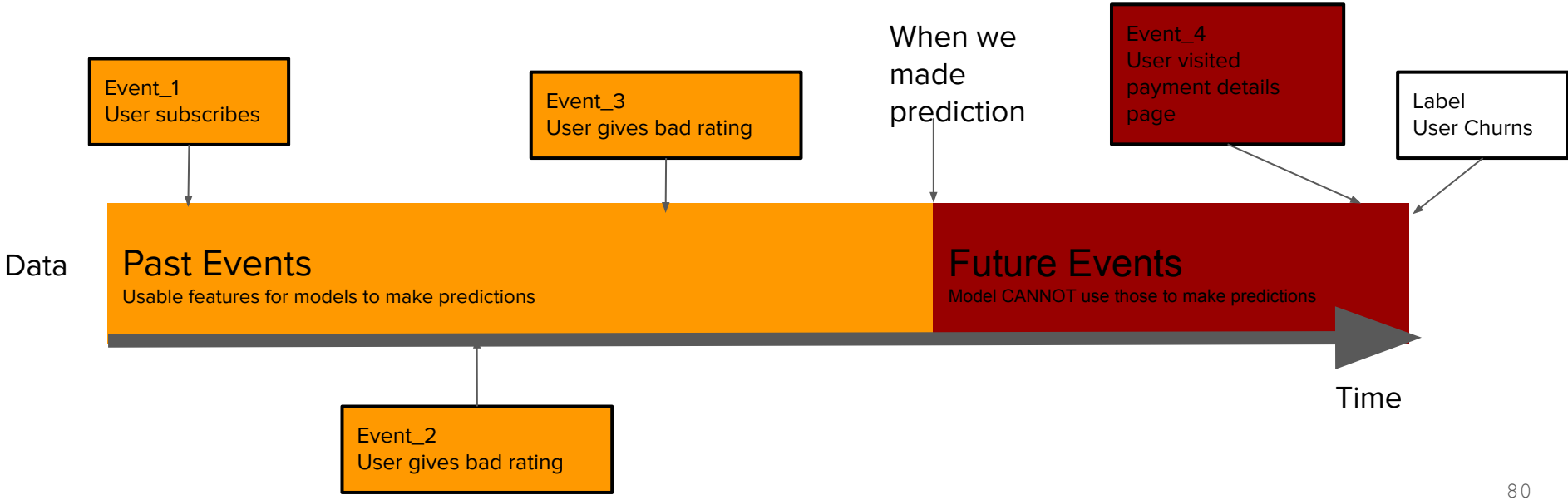
Source: https://www.protocol.com/entertainment/netflix-churn-subscription-services

Churn prediction

- Build an ML model to predict which users will cancel memberships in the following one week
- Features will be created from plenty of historical “events” for each user
- After feature engineering/selection, we found one event: “users visited payment details page” that has 80% correlation with churn label.
- This feature is added. And ML model get over **90%** accuracy in development stage
- However, when the model is deployed in production, the accuracy drops to **70%**

Churn prediction

- Target leakage is happening
  - Event_4: user visited payment details page happens always a few seconds before user churns
  - Event_4 is not available during inference and happens always in the future



Target leakage example II

- Detect Lung Cancer from CT Scans
- Collected from Hospital I
- Performs well on unseen data from I
- Performs poorly on new data from Hospital II

Date
Doctor note
Medical record
Scanner type
CT scan Image

Target leakage example II

- Detect Lung Cancer from CT Scans
- Collected from Hospital I
- Performs well on unseen data from I
- Performs poorly on new data from Hospital II

Date
Doctor note
Medical record
Scanner type
CT scan Image

At hospital I, when doctors suspect that a patient has lung cancer, they send that patient to a higher-quality scanner

How can we prevent data leakage

- Check for duplication between train and valid/test splits
- Use only train splits for feature engineering (model training for sure)
- Check the correlation between feature and label
- Keep asking yourself during model development: can we use this information when the model is deployed to address new samples in production?
- Feature Store can avoid target leakage
    - Point-in-time Joins

Next Class: Neural Networks and Deep Learning