PRELAB #2

modeling

FOCUS: analyzing and organizing raw data & preparing it for the next stage of ML process.

DATA MATRIX

- Data can come in many different formats.
- Data generally isn't stored in a format that's compatible for most ML models.
- **MODEL DEVELOPER:**
  - ↳ Transform the data appropriately
  - ↳ Prepare data for model performance where prediction, accuracy & generalization are your goals.
- Two mathematical constructs that represent data:
  1. Vectors = 1D array (EX. commonly used in Numpy)
  2. Matricies = 2D array (EX. set of N same size array stacked on top of each other.)

EX) Here displays a typical Pandas DataFrame. It has N rows & K columns where each row & column is a unique array. In our work we'll perform operations on both the column & rows arrays.

NOTE: only in SUPERVISED LEARNING.

Features | Label

Merchant Distance	Transaction Amount	Merchant Code (Type)	Is Fraud
5	39.01	1	0
2	112.81	1	1
8	4.99	1	0
6	1115.67	1	1
0.5	97.96	4	1
10.11	100	3	0
4	1.15	1	0
29.79	5.87	1	0
45.47	15.93	2	1
4	2500	1	0
0.95	25.66	2	0
21.33	2	5	0

Examples — Each row represents individual entities/units of the analysis.

Ex. when we do predictions on a trained model, we'll do them on a row ARRAY.

Ex. When we do data preparation work we often do operations on individual COLUMN ARRAY

- Defining what makes your tables' examples should be done in the early problem formulation stage.
- Your data preparation should then be centered on finding the appropriate sample of these units & defining the attributes associated w/ these units.
- With the units defined we will then define the columns [FEATURES]
  - ↳ The process of defining & coding up the features = FEATURE ENGINEERING
- LABEL is the item we're trying to predict.
  - ↳ During SUPERVISED LEARNING our goal is to ensure we have consistent set of FEATURES & LABEL for each EXAMPLE.

What we should consider when modeling: how many features & records are sufficient?

- ↳ GENERAL RULE: more is better of each.
- ↳ RIGHT ANSWER: Perform empirical evaluations.

WHO/WHAT ARE WE MODELING?

- Define your UNIT ANALYSIS in your INITIAL PROBLEM STATEMENT.
- The UNIT ANALYSIS needs to be considered in ALL STEPS in the MODEL BUILDING PROCESS.

EX) PROBLEM STATEMENT: FRAUD DETECTION: What's the likelihood that this transaction is fraud?
- ↳ MARKETING: Will this reader click on the ad I'm showing?
- ↳ RECOMMENDATIONS: What are the best tweets to display to this Twitter user?
- ↳ TELEMETRY: What's the likelihood that this engine will fail in the next 24 hrs?

Each can be solved with binary classification models & each has different unit of analysis that's specified in the PROBLEM STATEMENT.

- The data matrix will build to train a ML model will contain SAMPLE of these UNITS.
- Each SAMPLE are called EXAMPLES.
- The Database we'll be working with will probably have special ID columns that correspond to your UNIT OF ANALYSIS.
  - ↳ All of our DF/tables should include this ID/set of IDs.
  - ↳ These IDs are useful when we want to merge/join individual DF together & they'll be used as JOINT KEYS.

REMEMBER: ID COLUMNS ≠ FEATURES

SAMPLING TECHNIQUES

SAMPLING: The process of extracting subsets of examples from available universe & data.

↳ CONCERNS:

1. The population we want to manage
2. Managing the number of examples we'll use to model

↳ DEFINING SAMPLE POPULATION:

- More descriptive, though, than just the unit.
- Are there any attributes of those units that should be considered when sampling them from larger pool of units.
  - ↳ This should be specified during PROBLEM FORMULATION

UNIT ANALYSIS = 🔲 SAMPLING = 🔲

EX) We'll add extra attribute that'll serve as filter on full set of units.
These attributes should be tied to the core problem you're trying to solve.

PROBLEM STATEMENT: FRAUD DETECTION: What's the likelihood that this transaction is fraud?
- ↳ MARKETING: Will this reader click on the ad I'm showing?
- ↳ RECOMMENDATIONS: What are the best tweets to display to this Twitter user?
- ↳ TELEMETRY: What's the likelihood that this engine will fail in the next 24 hrs?

↓ INSTEAD OF

↳ FRAUD DETECTION: What's the likelihood of ALL transactions being fraud?

↓ With sampling

↳ FRAUD DETECTION: What's the likelihood that a web-based transaction is fraud?
↳ MARKETING: Will this non-subscribing reader click on the ad I'm showing? ⟹ subscriptions are FALSE
↳ RECOMMENDATION: What are the best tweets to display to this new Twitter user?

- Oftentimes you don't want your model to apply to every person in every situation.
- Sometimes you can have different models for different segments of the population.
- Your PROBLEM STATEMENT may dictate using different sampling populations.
- Build a model that's specific to each of these populations.
- WHY BE PRECISE W/ SAMPLE POPULATION? ⟹ CRUCIAL: Generalization
  - ↳ A model trained on a given sample whose population is defined by having certain attributes may not generalize & perform well on a sample with different attributes.

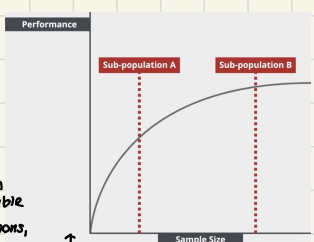
IID (independent & identically distributed) SAMPLES: When we expect 2 independent samples to have the same distribution of features.
- ↳ We can achieve this by using a RANDOM NUMBER GENERATOR to select examples from a population.

SAMPLING STEPS:

1. Define the population of interests, including selecting the unit & key attributes to filter on.
2. Random selection from that population, where the size is determined by what's available and/or what's necessary for strong generalization performance.

GENERAL THUMB RULE: more examples lead to better models

HOWEVER: there may be diminishing returns ∴ we want sample size vs. performance to look like:



We may want to deliberately invest in more examples from subpopulation A to lift performance. Then it would be comparable to Group B.

Always aim to build models that treat people in different protected classes equally. It's possible that bc certain classes are underrepresented in the data, the models are inaccurate for those particular group. If data is available then evaluate the model on different groups to test for this. If the model underperforms for smaller subpopulations, one trustworthy way to solve this is to resample the training data so that each group is equally represented.

DATA PREPARATION TECHNIQUES

In real life data is not always suitable for training out-of-the-box. There can be missing values, bad formatting, outliers, and data redundancy are some common problems a ML engineer has to overcome.

Below is an example dataset that contains some of the issues we just discussed.

Sample dataset

Name	Income	Credit Score	Occupation	Job Sector	Loan Status
John Doe	$76,000	650	Engineer	Engineering	Good
Gill Bates	$85,000	760	Nurse	Healthcare	Defaulted
Jane Doe	"95000.00"	0	Banker	Financial	Good
John Doe	$76,000	650	Engineer	Engineering	Good
Melon Usk		810	Flight Attendant	Transportation	Excellent
Barren Wuffet	5000/mo	35000	Contractor	Construction	Defaulted

① The entry for John Doe is repeated
② Income format is not standardized
③ Credit score contains 2 outliers: 0 & 35,000 likely to be errors as typical range is between 350 to 850.
④ The entry for Melon Usk contains a missing value.
⑤ Occupation & Job section are somewhat redundant as they tell similar stories.

KEY POINTS:

- Essential that raw data go through pre-processing steps such as cleaning, featuring engineering, and outlier handling.
- Data matrix is the ideal input format for ML.
  - ↳ The 1st N-1 column [NOTE: N = size] in a data matrix are features & the last column is the label.
  - ↳ Each row in a data matrix is a data point.
- EDA (Exploratory Data Analysis) helps engineers and scientists understand the basic properties of a dataset including its shape, statistical properties, and whether it contains outliers or not.

DATA UNDERSTANDING:

DEFINE THE PROBLEM: This is step zero of any ML project. Before we perform any action, we need to first answer the who and what we are modeling. This can be answered by defining the problem statement & breaking it down into units of analysis.

EXPLORATORY DATA ANALYSIS: EDA is the act of "poking around" the data for any obvious insights. This entails bringing in the data (or a subset thereof) into our coding environment, performing descriptive summary using common statistical libraries & making plots such as scatterplots and histograms for any obvious insight.

VISUALIZATION: Helps us understand our data better. Helps us determine the skew of data, shape of its distribution, and if any outlier is present.

DATA PREPARATION:

SAMPLING: Once we have a concrete idea of the problem we're trying to solve, we need to ensure our data is meaningful in the context of the problem. This can be thought of as having the data drawn from the same environment as production.

We want to ensure that our data points are independentially & identically distributed (IID), meaning that they should be drawn from the same distribution and are independent from each other.

SPECIFYING LABEL: The training dataset for supervised learning must have some "ground truth" (label) associated with each data point. The label is what we're trying to predict & it's the goal of our ML model.

THE DATA MATRIX: The underlying code for most ML models is optimized to process data in an N-dimension table AKA. data matrix, where each row is example/data point, the last column is label, and each remaining columns is features.

DATA CLEANING: Most data is inherently "dirty". This can be caused by problems such as system error, data corruption, and poor quality control. In data cleansing, we ensure our data does not contain missing values or unwanted outliers. When these conditions are encountered, we perform actions s.a. dropping the row or winsorization in order to ensure our data matrix is clean & free of data that are non-representative of the problem we're trying to solve.

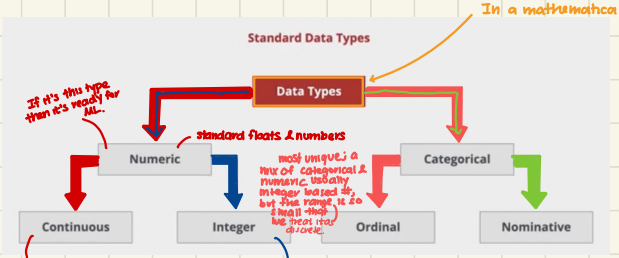
MODELING:

FEATURE ENGINEERING: The data available to us may not always be readily suitable for training a ML model. Sometimes this is due to formatting incompatibility, other times the models themselves are optimized for certain types of input. Feature engineering concerns with bringing data into the right format & representations required by the intended ML model.

CREATE LABELS & FEATURES

Continue our discussion on data preparation and focus on the columns of our matrix.

Different common DATATYPES:



- All ML methods ultimately operate on numbers so you can think of data prep as simply as creating a matrix full of numbers.
- Different datatypes have implications on how you might prepare FEATURES and how you can use them as LABELS for supervised learning tasks.
- Ex) categorical – ordinal : surveys where people use 5 or 10 rating scale to express an opinion about something.
  - ↳ ORDINAL types are numbers ∴ can be used directly as FEATURES.
  - ↳ Best for ordinal features to be converted to binary indicator features
  - ↳ PREDICTING ORDINAL: use both regression & classification.
    - ⇓ You may get values that don't make sence
    - → can help you avoid this problem.

ONE HOT CODING

- CATEGORICAL — NOMINATIVE
  - ↳ A categorical variable → often represented as strings. ⟹ Most ML algorithm can't operate on strings ∴ CONVERT TO NUMERIC
    - → Database systems often encode discrete categories as integers. ⟹ DON'T TREAT THEM AS PURE #s.
  - ↳ Nominative datatypes are typical basis for classification.
    - ↳ When used as LABELS we call the individual values CLASSES.
    - ↳ Our model either directly predicts the CLASS or produces a score that represents the likelihood of belonging to a specific class.

SPECIFYING A LABEL

When building our data matrix for ML applications, we invest time in creating columns. Most columns represents FEATURES in process called FEATURE ENGINEERING.

In SUPERVISED LEARNING we first define the LABEL which nearly every aspect of the development process is centered around the label.

EASIER CASES:
- → There's a clear set of discrete choices to model
- → Each example maps cleanly to a single class
- → The label is directly observable

HARDER CASES:
- → The problem goal is subjective
- → There's multiple relevant labels to consider
- → The ideal label is harder to work with.
- → We don't directly observe the label we want to predict.

REMEMBER: Defining LABEL is tightly coupled with PROBLEM STATEMENT.

Ex) Labeling for social network feed recommendations

PROBLEM GOAL: to maximize long term user satisfaction ⟹ SUBJECTIVE + HARD TO MEASURE AT SCALE ∴ we rely on PROXIES.
↳ This is what keeps the social network relevant & entertaining ∴ how we keep our customers

⇓ Primary proxy

User retention after 90 days ⟹ DIFFICULT TO WORK WITH + LONG TIME TO MEASURE + NOT HELPFUL MAKING SHORT-TERM DECISIONS

⇓ Proxy of the proxy

We can choose between DWELL TIME, LIKES, COMMENTS, etc. various short term engagement

REMEMBER: NO right answer on how to label a problem

LABELING TRICKS TO SIMPLIFY THE PROBLEM : Take harder problem & simplifying so it's easier to work with.

① Grouping ordinal labels into high/low:

Ex) Predicting ordinal outcome like an online review from 1-5. We can care about high vs. low ratings.
Map the ratings to a high vs. low scale & use BINARY CLASSIFICATION.

↳ rating = 4 OR 5, label = 1
↳ rating = otherwise, label = 0

There can be SUBJECTIVITY when defining what's high/low but core idea is SIMPLIFYING the problem.

② Converting time-to-event predictions to classification tasks.

Ex) Predicting the time something might happen.
To approach the problem: convert to BINARY CLASSIFICATION

↳ whether or customer will still be a customer after some pre-specified # of days
↳ We'll lose some GRANULARITY in our predictions but problem will be easier to solve.

REMEMBER: changing label = changing problem statement.

REMEMBER: subjectivity is an inevitable part in design decisions ∴ always solicit feedback to ensure you're still align with the spirit of the OG problem.

pandas.Categorical(): Represents a categorical variable in classic R/S-plus fashion.

```
pandas.Categorical(values, categories=None, ordered=None, dtype=None, fastpath=False, copy=True)
```

- values: list-like — The values of the categorical.
- categories: index-like [OPTIONAL] — The unique categories for this categorical.
- ordered: bool; DEFAULT=False — Whether this categorical is treated as a ordered categorical. TRUE = resulting categorical will be ordered
- fastpath, copy: OPTIONAL

pandas.get_dummies(): converts categorical variable into dummy/indicator variables

```
pandas.get_dummies(data, prefix=None, prefix_sep="_", dummy_na=False, columns=None, sparse=False, drop_first=False, dtype=None)
```

- data: array-like, series or DF — Data in which to get dummy indicators
- prefix: str, list of str, dict of str — DEFAULT=None — String to append DF column names
- prefix_sep: str — If appending prefix
- dummy_na: bool — DEFAULT=False — Add column to indicate NaNs
- columns: list-like — column names in the DF to be encoded
- sparse, drop_first, dtype

INTRODUCTION TO FEATURE ENGINEERING

GOAL: prepare our data for ML model to train on.

- Not much concerned about cleanliness of the data (Ex. outliers, missing data)
- FOCUS: mapping the appropriate predictive/causal concepts into a data representation & transforming it into a format that can be easily consumed by intended ML model.

ESSENTIALLY we're answering: HOW DO WE GET FROM SOME DATA SOURCE TO A COLUMN IN A DATA MATRIX?

↳ 2 strategies: ① Select the right data to be our features.

↳ STEP 1: Selection, filtering, and fetching of the desired data from some data source into our ML environment.
  ↳ Machine's condition = FEATURE
  ↳ Operational status = LABEL

↳ STEP 2: Bring data into our ML environment.

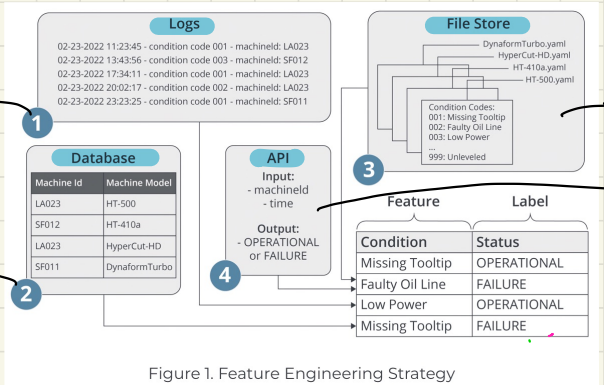


Figure 1. Feature Engineering Strategy

- We can repeat this step to bring in other features for constructing our data matrix.

② FEATURE TRANSFORMATION: Transform the data into a format that's suitable for the intended ML model.

↳ The types of feature transformation to apply depends on the model.

↳ common feature transformation techniques: BINARY INDICATOR, ONE HOT ENCODING, FUNCTIONAL TRANSFORMATION

PROCESS OF FEATURE ENGINEERING

2 STRATEGIES:

① Mapping predictive or causal concepts to data representation

↳ Aggregate various data assets into a single modeling matrix

↳ Converting data with different formats & locations into a common structure.

↳ create conceptual model of problem to help guide you.

NOTE: performance usually comes from 1st strategy.
If your not capturing the right concepts in your data ⇒ no algorithm is going to make a good prediction.

② Manipulating data so that's appropriate for common ML APIs.

↳ 4 main techniques: 1) Work converting log or database information into concepts you think would make for good features.
2) ONE HOT ENCODING
↳ categorical values absolutely needs to be converted to numeric forms before it can be used as a feature
↳ Not optional when you have categorical or text or string data.

DOMAIN-DRIVEN FEATURE ENGINEERING

Cases where feature engineering isn't important:

EXAMPLE 1 IMAGE RECOGNITION
EXAMPLE 2 LANGUAGE TRANSLATIONS

Both involve DEEP NEURAL NETWORKS where they have the property of they can automatically learn feature concepts from RAW DATA.

PRINCIPLES: when we map concepts to data representations

① THINK LIKE A SCIENTIST: Based on your knowledge of the event you're trying to model, identify likely causal factors & test them empirically.

- The best features in the model are believed to cause the outcome.
- Don't limit yourself to proven causes.

② SEEK THE WISDOM OF DOMAIN EXPERTS: Interview domain experts to gain insight into hypothesized or known causal factors.

- You can understand what features NOT to build → legal + system constraints?
- Knowledge on how to approach the problem

③ PRIORITIZE & ITERATE: Plan your feature engineering tasks in advance & prioritize those with highest expected likelihood to be predictive.

- use your collective intuition & knowledge of the domain & plan ahead.
- create a list of all the features you plan to build & seek input on their expected value.
- Then build the features & evaluate them as you go along.

EX) TWITTER RECOMMENDATIONS ON WHO TO FOLLOW

- LABEL: follow/not follow after being recommended.
- we're creating concepts that either cause/highly correlated with the outcome of interest.

EXAMPLE CONCEPTS FOR THE FEATURE ENGINEER PROCESS:

Why do I follow someone? ← Think about your own experience; what thought process do you have when you decide to follow someone.

✗ ↳ Do I find the person interesting? ← we want to AVOID abstract & subjective concepts.
We want to define factors that are specific enough that can be coded up in an unambiguous way.

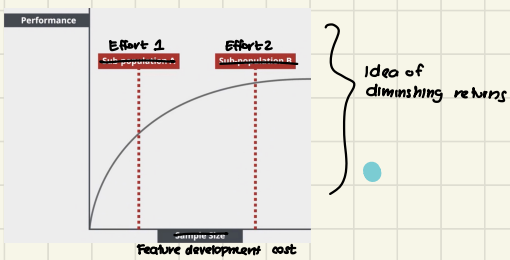
✓ ↳ The person posts about topics that I'm interested in.
↳ Represent it as a match between topics that they post & topics I explicitly followed.

✓ ↳ This person follows me.
↳ Assume that I'd reciprocate.

✓ ↳ People I follow also follow this person.
↳ Friends I consider interesting also consider someone else interesting. I would as well.

✓ ↳ The person is very popular.
↳ other people find interesting than I might as well.

- These examples meet our condition & we can translate those features that make it into our data matrix.
- If we plot feature development cost w/ model performance then:
  ↳ once we hit those diminishing returns, we can then evaluate how much additional effort do we really want to put into the feature engineering.



FEATURE TRANSFORMATIONS

Different ML models are intended & optimized for different inputs. This means that ML engineers need to transform the features into the appropriate format or values in order to train models properly.

ALL OF THE ALGORITHM REQUIRE ANY CATEGORICAL VALUES TO BE CONVERTED TO NUMERICAL FORMAT

↓

POSSIBLE TRANSFORMATION TECHNIQUES:

BINARY INDICATOR: In some cases, it makes sense to represent numeric or categorical data as binary values.

- BENEFIT: makes the data smaller == lower computing cost
  makes our final model simpler == desirable if we want our model to ultimately be more general than specific
- EX) Cast movie rating greater than or equal to 3 to Good & movie rating less than 3 to bad.
  EX) Cast glucose level of 100 or higher as Abnormal and below 100 as Normal.
  EX) Group countries into English Speaking or Non-English Speaking.

ONE HOT ENCODING: Popular method for categorical data. ML algorithms operate on numerical inputs ∴ we have to transform categorical data into some form of numerical representation. The idea is to convert K categories into an array of K binary values prior to being consumed by a ML model.

- EX) Feature called weather which has 5 distinct values: Sunny, Overcast, Rain, Snow
  2 other features: elevation & zip code ← BOTH have numerical types ∴ don't need to be one-hot encoded

To one-hot encode weather feature: → Replace our feature (column) weather w/ 4 new features (columns)
→ One new feature (column) for every distinct value in the weather column.
→ Each new feature would have a binary value: 1/0.
↳ 1 = every row in which it appeared in the OG weather column
↳ 0 = otherwise

Data before One-Hot Encoding

Elevation	Zip code	Weather
1000	92651	Sunny
40	12834	Overcast
500	85532	Rain
232	43645	Snow
157	23426	Rain

One-Hot Encoding

Data after One-Hot Encoding

Elevation	Zip code	Sunny	Overcast	Rain
1000	92651	1	0	0
40	12834	0	1	0
500	85532	0	0	1
232	43645	0	0	0
157	23426	0	0	1

Figure 1. One-Hot Encoding

FUNCTIONAL TRANSFORMATIONS: We apply a function to convert one numeric value to another based on some function.

- EX) Converting a feature to log scale or to the square.
- EX)

Feature	Label
-3	8.99
-1.98	4
-1	0.98
0.1	0
1.1	1
2	4
3	8.99

Square()

Feature	Label
9	8.99
3.9204	4
1	0.98
0.01	0
1.21	1
4	4
9	8.99

Figure 2. Feature Transformation Using Square

← our model more complex to capture the relationship among labels & features:
↳ more data to train
↳ slower in making predictions

transforming the feature values so that the relationship is linear
↳ simplify the model
↳ improve prediction speed.

INTERACTION TERMS: In some cases the combined effect of one or more features leads to a strong predictor than each feature alone. In such cases we can form additional feature by multiplying the individual features (or other mathematical operations) to arrive at a 3rd feature.

X1	X2
1	2
4	6
2	3

X1	X2	X1X2
1	2	2
4	6	24
2	3	6

Figure 3. New interaction terms created by multiplying X1 and X2

BINNING: Sometimes we want to convert numerical values into discrete bins when the ordinal nature of the numerical values themselves aren't necessarily indicative of that label.

- Binning is best visualized in HISTOGRAMS
- Useful in reducing the complexity of the model to achieve GENERALIZATION
- Ex) Binning the weights of individuals into groups in an attempt to predict their risk of heart diseases.

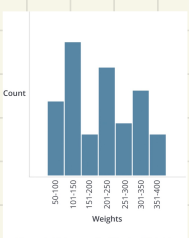


Figure 4. Histogram For Binning Weights

SCALING: Some models perform better when features are scaled to some standard range, while others have no problems accepting an input of a wide range.

- Most common types of scaling approaches:
  - ↳ STANDARD SCALER: transform the values of a feature to have a mean 0 & standard deviation of 1.
  - ↳ MIN-MAX SCALER / MIN-MAX NORMALIZATION: transform the values of a feature to a range between some min and some max value (oftentimes 0 & 1), while keeping their relative distance the same.

SUMMARY:

Expected input and output data format for various feature transformation techniques.

Technique	Input	Output
Binary Indicator	Categorical or Numeric	Binary (0/1)
One-Hot Encoding	Categorical	Binary (0/1)
Functional Transformation	Numeric	Numeric
Interaction Terms	Numeric	Numeric
Binning	Numeric	Categorical
Scaling	Numeric	Numeric

INTRODUCTION TO EXPLORATORY DATA ANALYSIS

Assume we're at the point where we have a representative sample of users & have a structured dataset that includes features, and if appropriate, a label.

Approach our data investigation phase with 2 main goals:

① Ensure we have high-quality data
↳ Bad quality data: missing values, outliers, etc.

② Delievering insights so we can be knowledgable about the problem.
↳ EXPLORATORY DATA ANALYSIS (EDA):

- Explore your data by asking several key questions about your data, which are oriented towards supervised learning.

Made w/ direct utility to the model-vetting process

↳ How is the data distributed? ⟹ informs about outliers & skews we should address
↳ What features are redundant? ⟹ informs what features we may or may not cut.
↳ How do different features correlate with our label? ⟹ informs about features to keep & offers high expectations on how modeling might be working.

GOAL: We want to end up with well distributed independent features.

When examining/working with data:

STEP 1: Look at it
↳ You're given a flat text file
↳ Before loading it into python, check for a few things!

USING PANDAS TO INSPECT YOUR DATA

When performing EDA our main goals is to check for data quality issues. (missing values, outliers)

↳ Pandas method: `pd.dataframe.describe`

EX)

```
In [27]: describe_vars = ['isbuyer','buy_freq','expected_time_buy','uniq_urls','num_checkins','y_buy']
         df[describe_vars].describe()
```

← filtered to a limited set of columns

Out[27]:

	isbuyer	buy_freq	expected_time_buy	uniq_urls	num_checkins	y_buy
count	54584.000000	2327.000000	54584.000000	54584.000000	54584.000000	54584.000000
mean	0.042632	1.240653	-0.198040	86.569343	720.657592	0.004635
std	0.202027	0.782228	4.997792	61.969765	1275.727306	0.067924
min	0.000000	1.000000	-181.923800	-1.000000	1.000000	0.000000
25%	0.000000	1.000000	0.000000	30.000000	127.000000	0.000000
50%	0.000000	1.000000	0.000000	75.000000	319.000000	0.000000
75%	0.000000	1.000000	0.000000	155.000000	802.000000	0.000000
max	1.000000	15.000000	84.285710	206.000000	37091.000000	1.000000

binary label

The count of records that have non-null values

Tells us over 95% of the column values are missing.

2 columns have have negative values. It can be an encoding value error.

We'd usually do some pre-processing to address this particular concern

STANDARD DISTRIBUTION STATISTICS

Examining this, the average is incredibly low

GENERALLY, if your binary label is < less than 1% of the time ⇒ CLASS IMBALANCE PROBLEM

mean is greater than the median ⇒ HIGHLY SKEWED COLUMN

⇓

RELATED TO HAVING OUTLIERS

VISUALIZE YOUR DATA: UNVARIANT PLOTTING

Plotting your data gives a lot of information that summary statistics don't.

Using plots helps us understand 2 basic things:

① How is a given feature distributed? [UNIVARIANT INSIGHT]

↳ Tells us the range of the data, central tendency, skew, and variance.

↳ Knowing this will tell us what preparation techniques we might need to use.

② Know how two or more features are distributed together.

↳ Tells if 2 features are correlated or redundant.

PLOTS TO UNDERSTAND UNIVARIANT DISTRIBUTIONS

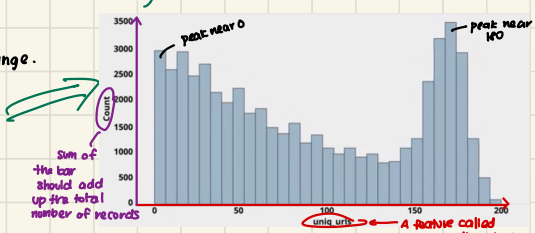
① HISTOGRAM: x axis = range of data (specific features)
y axis = how much of the data is present in that range.

↳ Perfect to understand if there's a heavy skew

↳ Works for both numeric & categorical data

↳ Using SEABORN and MATPLOTLIB to produce these samples.

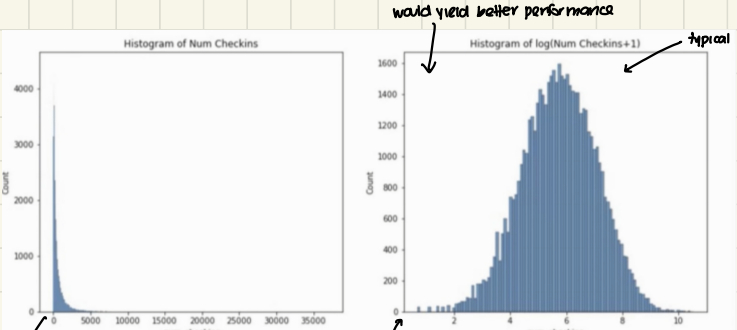
EX) This shows that we have a multi-modal distribution



Sum of the bar should add up the total number of records

A feature called uniq_urls with values from 0–200.

EX)



would yield better performance

typical bell curve for normal distribution

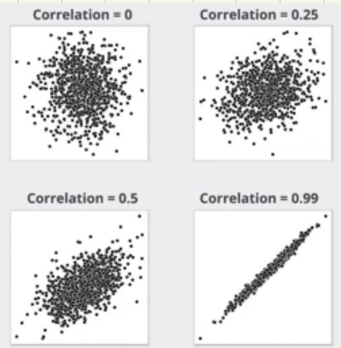
Most data is in the first 5% or 10%

when we take the log of the feature

- This is a good example of a feature with which we may want to form a functional transformation w/ the function here being the natural log.
- When a feature is skewed like this, some modeling algorithms will make predictions on higher values of the feature.

VISUALIZE YOUR DATA: BIVARIATE PLOTTING

- Visualize relationships between 2 data columns. Our **GOAL** is to understand if 2 columns are correlated with each other.
- **GENERAL RULE:** don't use highly correlated features in a model.
- Our plotting efforts will center on the bivariate relationship between a single feature and the label of your problem.
  - ↳ will help us explain how our model is working



NUMERIC DATA:

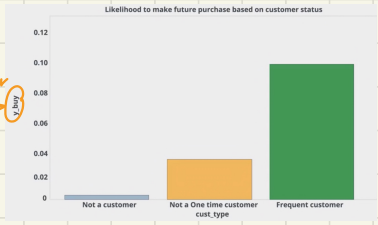
① **SCATTER PLOT:**
- ↳ Analyze the **CORRELATION** between x and y axis.
  - ↳ GENERAL RULE: more dispersion = less correlated
  - ↳ Building & selecting features we want ~0 correlation between individual features. ~1 or ~-1 between features and labels.
  - ↳ NOTE: Don't solely rely on correlational statistic

② **BAR PLOT**

HACK: By binning the data and plotting, we can compute label rates, get the error bars, and visualize the underlying trend.

BIVARIANT PLOTS: WHEN ONE OF THE VARIABLES IS A LABEL

EX)



Label average for each value of the categorical feature

HIGHLY PREDICTIVE FEATURE

- **FEATURE** is CATEGORICAL
- **LABEL** is a BINARY OUTCOME
- This is a **BARPLOT**
- Seaborn automatically include **ERROR BARS** to show that the differences are statistically significant.
- Plots like this helps build insight around a particular problem.

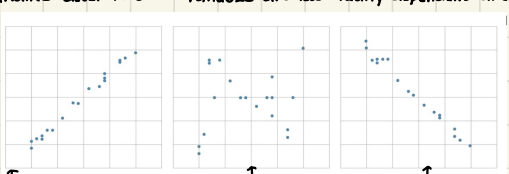
CORRELATION, COVARIANCE, & MUTUAL INFORMATION

Both **covariance and correlation** find the linear dependencies between variables.

As part of the EDA we're interested in the dependencies between various random variables. Which is commonly achieved by calculating the correlation & mutual information between these variables.

CORRELATION & COVARIANCE:

- For 2 randomly distributed variables the covariance formula is: $Cov(x,y) = \frac{\sum_{i=0}^{N}(x_i-\bar{x})(y_i-\bar{y})}{N-1}$ — Tells us whether two features are directly dependent or inversely dependent on each other.
  - ↳ Larger magnitude of covariance = variables are highly independent on each other
  - ↳ covariance closer to 0 = variables are less linearly dependent on each other.



↑ HIGH POSITIVE COVARIANCE ∴ HIGH LINEAR DEPENDENCY

↑ NEAR 0 COVARIANCE ∴ LITTLE LINEAR DEPENDENCY

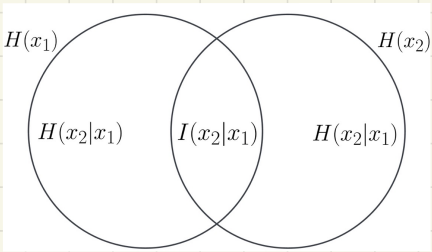
↑ HIGH NEGATIVE COVARIANCE ∴ HIGH LINEAR DEPENDENCY

- Magnitude is unbounded & can be arbitrarily large depending on the data you're working on.
- **PEARSON CORRELATION:** standardizes the range of value for covariance to be always -1 and 1.

- Widely used correlation formula: $Corr(x,y) = \frac{\sum_{i=0}^{N}(x_i-\overline{x})(y_i-\overline{y})}{\sigma_x \sigma_y}$
- By standardizing the covariance we can easily compare the degree of linear dependance between variables.

MUTUAL INFORMATION: Helps us understand the amount of reduction in uncertainty that one random variable provides for another.

- Takes on the range between 0 & 1 & is built upon the concept of uncertainty.
- **CONCEPT OF UNCERTAINTY:** quantified in information theory as **ENTROPY** (denoted as H)
  - ↳ Entropy = 0 ⟹ variable is completely predictable
  - ↳ Entropy = 1 ⟹ variable is completely uncertain
- Calculate mutual information between 2 variables: $I(x_1,x_2) = H(x_1) - H(x_1|x_2) = H(x_2) - H(x_2|x_1)$
  - uncertainty of variables x_1 given variable x_2.



← If we have large overlapping region between $H(x_1)$ and $H(x_2)$ then we have high mutual information.

← If variables are completely non-overlapping, then there's no mutual information between 2 variables.

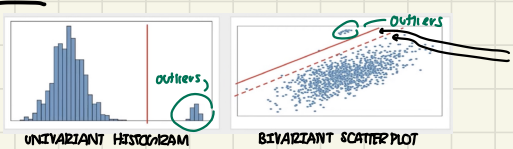
EX)
- Assume $H(x_2)$ is completely uncertain (1)
- Then we introduced $H(x_2)$ which helps bring down the entropy down to a value 0.1 denoted by $H(x_2|x_2)$, meaning our mutual information is: 1-0.1 = 0.9.
  - ↳ bic x_2 greatly reduces the uncertainty of x_1 ∴ there's high mutual information between the 2 variables.

HOW TO USE CORRELATION & MUTUAL INFORMATION

- When we have a large number of features to work with, we only want to choose those that are **MOST RELEVANT** in predicting our label.
- Too many features ⟹ increases computing cost ⟹ reduces generalizability of our model as it would try to learn from features that are less relevant. ↳ NOT OUR GOAL!
- **PROPER APPROACH:** select features w/ highest correlation & mutual information against our label.
  Consider any pairs of features that are highly correlated or have high mutual information to be redundant, in which case we may choose to remove one of the features from our dataset.

OUTLIERS: A datapoint that's far from all other datapoints.

EX)



UNIVARIANT HISTOGRAM — BIVARIANT SCATTER PLOT

- Outliers can be BOTH univariant & multivariant in nature.
- Outliers are subjective; the 2 lines can either present a cutoff point for considering if something is an outlier.
- OUTLIER THRESHOLDS: **selection rule of thumb:** only consider no more than the top 1% to be an outlier.

REASONS WHY THEY'RE PROBLEMATIC: outliers tend to skew mean values towards the outlier & the increase the variance which makes error bars larger.
↳ Any extrapolation around these errors are likely to be error prone.

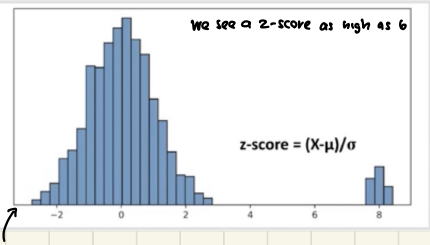
- Outlier can be a clue that something went wrong in the data collection process.
- Researching the outlier can expose some errors in the overall data processing.

METHODS IN DETECTING OUTLIERS:

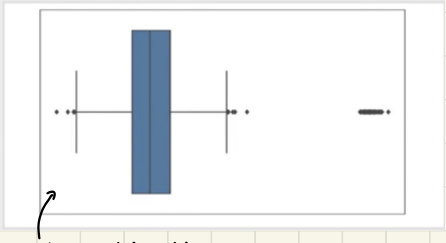
Both use the natural variance of a data & outliers are defined based on how much they deviate from normal variance.

① **Z-Score:** computes z-score for each point and any point with abs(Z) > K is an outlier.
↳ z-score is the point minus the average for that feature divided by the standard deviation.
↳ The absolute value of the z-score is greater than some threshold K ⟹ outlier

② **Interquartile range:** computes IQR as distance between 25th & 75th percentile. Any point that's K times greater than IQR from 25th or 75th percentile is considered an outlier.



- Z-score on the X axis
- Z-score is the result of transforming a variable by subtracting the mean from each point & dividing by the standard deviation.
- In a normally distributed variable, most of the mass should have a z-score less than 3.



- Box and whiskers plot
- The whiskers are tuned to 2x the interquartile range which is between 25th and 70th percentile.

We could want a outlier threshold to determine what's an outlier & what isn't.

WHAT TO DO WITH AN OUTLIER?

1. Discard those examples [if outlier is rare & you suspect it's driven by an error w/ the data generating process]
2. Winsorization:
   - identify the outliers
   - Replace them with a high but acceptable values

Ex)

COMMON STATISTICS REFRESHER

Mean

A mean is an average of a series of numerical values obtained by adding up the numbers and diving by the number of values. There are other kinds of means (geometric, harmonic) that are calculated differently, but the term "mean" usually refers to the arithmetic mean, which is the sum of all the observations divided by the number of observations.

Median

A median is the middle number in a series of numerical values sorted from lowest to highest. If there is an even number of values in the series, the median is the mean (halfway point) of the two middle numbers.

Mode

A mode is the most frequent value in a series. If a series has more than one mode, it's considered multimodal. Multimodality is best demonstrated with a histogram that has two peaks, where each peak can be considered a "local" mode.

Outlier

An outlier is a data point that differs significantly from other observations. This could be due to error (like putting a decimal in the wrong spot) or a true pattern that needs further investigation. There are different statistical ways of identifying outliers and a common one is to flag values that are less than or greater than $1.5 \times IQR$.

Z-Score

The z-score is a way of translating data to understand how many standard deviations away from the mean each point is. If a data point has a z-score of 1, that means it is 1 standard deviation above the mean. If a data point has a z-score of -1.5, that means it is 1.5 standard deviations below the mean. The z-score for each point is calculated by subtracting a value by the mean and dividing by the standard deviation. This is shown in the equation $Z = \frac{x-\mu}{\sigma}$, where x is the value, μ is the mean, and σ is the standard deviation.

Winsorization

Outliers can be removed from a dataset (and often are), or they can be modified in some way to prevent them from having a disproportional impact on the characteristics you are trying to describe in your data. Winsorization is the method of clamping outlier data points at specific values derived from the data itself, like a percentile. Data can be Winsorized from both tails of its distribution by choosing a lower/upper percentile and capping points at those percentiles.

Univariate

Data is univariate if it contains one variable (uni = 1, variate = variable). An example of univariate data is measured dog heights without any other demographic information (e.g., age, breed).

Multivariate

Data is multivariate if it contains more than one variable. An example of multivariate data is measured dog heights along with age, sex, weight, and breed.

OUTLIER DETECTION METHOD

Possible reasons why outliers occur:

- Human error (Ex. entry error)
- System error (Ex. integer overflow)
- Legitimate value, but is unrepresentative of the typical scenario.

2 common approaches for detecting & separating out outliers:

① Z-score:

The idea of z-score is to take a data point x, subtract it by the mean of the dataset, and then divide it by the standard deviation of the dataset. Typically when a data point has an absolute value of the z-score above 3, it is considered an outlier. In a normal distribution, 99.7% of all points in a dataset are expected to fall within a z-score of 3. Z=3 is also an acceptable threshold, but the MLE can use any threshold and test them.

$$z = \frac{x - \bar{x}}{\sigma}$$

② IQR:

IQR or Interquartile Range seeks to identify a range where the majority of the data points lie.

IQR is defined by the following formula:

IQR = Q3 - Q1

(Q3: median of the upper half of the data set; Q1: median of the lower half of the dataset.)

↳ After finding IQR we then precede to calculate the range of acceptable values by taking:

↳ Q1 + K*IQR to get the LOWEST ACCEPTABLE VALUE

↳ Q3 + K*IQR to get the HIGHEST ACCEPTEABLE VALUE

EX) We have an array: [1, 10, 35, 37, 38, 40, 45, 46, 50, 84, 85]. The median = 40.

↳ Lower half: [1, 10, 35, 37, 38] ; ½ is: 35 = Q1

↳ Upper half: [45, 46, 50, 84, 85] ; ½ is: 50 = Q3

∴ IQR = 50 - 35 = 15

K = 2

∴ K*IQR = 30

∴ 35 - Q1 = 5

50 + Q3 = 80

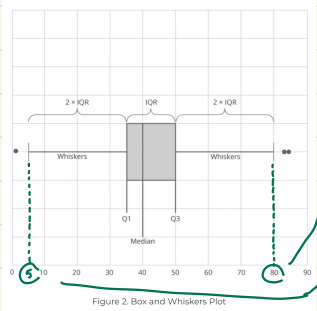


Figure 2. Box and Whiskers Plot

5 & 80 are outliers

MISSING DATA

If there's consistent errors in your data you probably want to trace the source and fix it.

- Common causes for missing data:
  - → system failures
  - → timeouts from logging data
  - → illegal values being passed into formatted data field
  - → incomplete webforms/surveys, etc.

HOW TO WORK WITH MISSING DATA:

- Traditional databases use: null
- Pandas data looks like: NaN (not a number) datatype
- There's a possibility that the missing values are acceptable condition by the system and some developer encoded them in a special way.

To examine if a particular column has missing values (T/F): data_frame.isna().sum() > 0

STEP 1: Check for missing-ness.

↳ Which column has missing values?

↳ How many missing values are there?

→ must address them!

HANDLING MISSING VALUES:

LOWEST EFFORT: Drop the record or column [DELETION]

↳ If missing value count is very low. OR

↳ If there's missing values in the same example ← indicates maybe systematic error in the data collection

MODEST EFFORT: Replace missing value with mean or median [IMPUTATION]

↳ Often choose MEDIAN when data is highly skewed.

MOST EFFORT: Predict missing value with E[x|x'] [INTERPOLATION]

↳ When there's correlation amongst your features, you might be able to predict the real value from the other features.

↳ We treat the future of interest as a LABEL

↳ Treat other features as PREDICTOR FEATURES

SUBJECTIVITY WHEN HANDLING MISSING VALUES:

- Making choices on the method & the values we might use for imputation if we use imputation.