Data Preprocessing And Cleaning

01

Data Preprocessing

Data Preprocessing

- Often, you cannot directly feed a dataframe into a model
- You would often need explore the data using visualizations
- And also get an understanding of the problem being solved and the data collection methods
- Then judge the right way to preprocess the data

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	pre
0	5285000	4600	2	2	1	yes	no	no	no	yes	2	
1	3675000	5640	2	1	1	no	no	no	no	no	0	
2	4200000	3520	3	1	2	yes	no	no	no	no	0	
3	2275000	1836	2	1	1	no	no	yes	no	no	0	
4	3570000	3150	3	1	2	yes	no	yes	no	no	0	

Data Preprocessing

- Two types of variables: numerical and categorical
- (you can also have image and text data but we'll talk about it later when we get to CV and NLP)
- The classification depends on the data type of the variables
- For most machine learning pipelines, we need to convert categorical data into numerical data
- You can do it using the OneHotEncoder or the LabelEncoder

One-Hot Encoding

- Convert a categorical column into n numerical columns, where n is the number of classes in the column
- For each data point, only the column of the corresponding class has value 1; the others have value 0

Color
Red
Red
Yellow
Green
Yellow

Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1

Image source:
https://www.kaggle.com/code/dansbecker/using-categorical-data-with-one-hot-encoding

Label Encoding

- For every class in the categorical column, map the class value into an integer
- Make a numerical column that include the corresponding integers

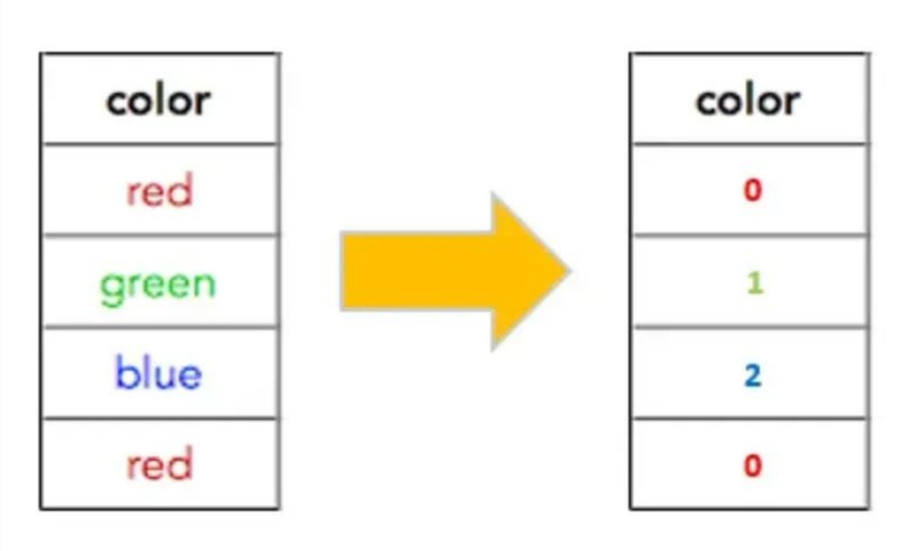


Image source:
https://medium.com/@sunnykumar1516/what-is-label-encoding-application-of-label-encoder-in-machine-learning-and-deep-learning-models-c593669483ed

Pros and Cons

- One-hot encoding:
  Advantage: does not assume ordinal relationship between different classes
  Disadvantage: creates many columns, making the data too high-dimensional

- Label encoding:
  Advantage: does not increase dimensionality of data
  Disadvantage: assumes ordinal relationship between different classes, which may not hold in reality

Normalization

- Often, it is helpful to normalize data to a specific range of values
- This includes the minimum, maximum, mean, or standard deviation of the data
- Especially important for neural networks!

Normalization

- MinMaxScaler: normalizes data linearly to a scale of 0 to 1
- StandardScaler: normalizes data linearly to a mean of 0 and a standard deviation of 1

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

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

Image source: https://www.digitalocean.com/community/tutorials/normalize-data-in-python
https://www.digitalocean.com/community/tutorials/standardscaler-function-in-python

Normalization

- MinMaxScaler:
  Advantages: ensures the data is of a uniform absolute scale
  Disadvantages: can be easily affected by outliers

- StandardScaler:
  Advantages: controls the mean and standard deviation of the data, which is a bit more robust than just taking the range
  Disadvantages: still easily affected by outliers (due to mean calculation)

Data Leakage

- To ensure a reliable model evaluation you would NOT want information from the training set to “leak” into the validation set
- This includes data entries (which can happen if you have duplicate entries in the dataset) and statistical information

Data Leakage

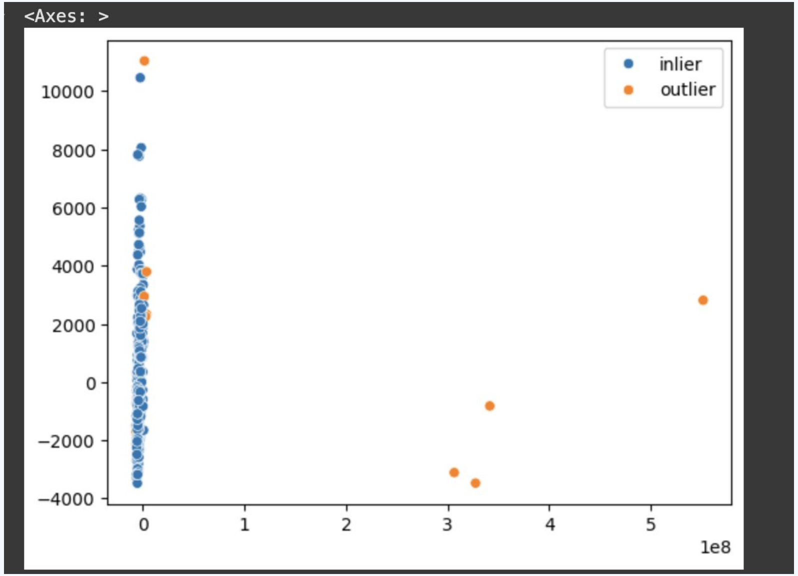
- Preprocessing steps should be done after data splitting
- Parameters (such as averages or standard deviations) in preprocessing steps should be obtained from the training set and then directly applied to the validation set

Try It Out!

- Experience the simple data preprocessing pipeline in this notebook!
- https://www.kaggle.com/code/carsoncheng/data-preprocessing-salary/edit

Outliers

- Sometimes there are outliers in the data
- They can stem from interesting phenomena that you can actually model (e.g., anomalies) or data input errors / errors in experiment procedures



Outliers

- Outliers can affect statistical analysis like analyzing the mean
- It can also affect modeling
- Identify outliers with data visualization

Outliers

- Try to look into the reason for outliers
- Drop (or correct if you can) outliers if it's a data input error
- Don't drop outliers if it is a potentially significant result especially in safety-critical applications
- https://medium.com/@abhaysingh71711/the-impact-of-outliers-on-data-when-to-remove-and-when-to-retain-fb6e474ddbd8

Missing Values

- Sometimes there are missing values in the data
- 1. Use models that can natively handle missing values
- 2. Impute missing values
- 3. Drop rows or columns

Missing Values

- Models that natively handle missing values: random forests, XGBoost, bayesian models,...

- Models that do NOT natively handle missing values: linear models, neural networks, ...

Missing Values

- Before deciding on whether to include missing values in the modeling, take note of the following types of missing data:
- MCAR: missing values completely random
- MAR: missing values depend on some observed variables and are otherwise random
- MNAR: missing values depend on the unobserved value itself
- https://medium.com/@sujathamudadla1213/what-are-the-differences-between-mcar-mar-and-mnar-missing-data-and-why-do-they-matter-for-aaa884938a8e

Missing Values

- MCAR vs not MCAR: Little's MCAR test, see if missingness of one variable is correlated with other variables
- MAR vs MNAR: requires domain knowledge (e.g., knowledge about data collection methods)

Missing Values

- For MCAR, you can simply drop rows with missing data or impute with mean or median of the non-missing values
- In other cases, the simple methods can introduce biases in the modeling towards cases where data is more likely to be complete
- For MNAR data, the methods involved can be very advanced (and require good domain knowledge of your study), if you are curious you can check related content online

Missing Values

- If a column has so much missing data (and is not particularly useful from the start), you can drop the column

Class Imbalance

- When dealing with classification, always note how many samples are in each class
- This is one of the most important steps in the EDA (exploratory data analysis)
- If the number of samples in one class is much greater than the number of samples in the other, the dataset exhibits a large degree of class imbalance
- This affects metrics such as accuracy; in spam detection, 99% accuracy is not good if it only learns to predict “not spam”

Class Imbalance

- Solution 1: set “class_weights” variables on the loss function
- Mispredictions in the minority class are penalized more heavily to compensate for the few number of samples of the minority class

Class Imbalance

- Solution 2: sample data from the dataset so that the training process becomes class-balanced
- Undersample majority classes during training, or oversample minority classes during training (e.g,. WeightedRandomSampler, SMOTE)

Class Imbalance

- For undersampling, all data from the majority classes are still used for training, but each majority class data sample is sampled less frequently
- For oversampling, it can work the same way as the aforementioned technique, but you can also create synthetic data with techniques like SMOTE or (for image data) generative models; may work better if you only have a few samples for the minority class

Low-Data Scenario

- Few-shot learning approaches are used
- Prototypical networks: selects “class prototypes” from averaging all samples in the class, and then classifying points based on which prototype it’s closest to

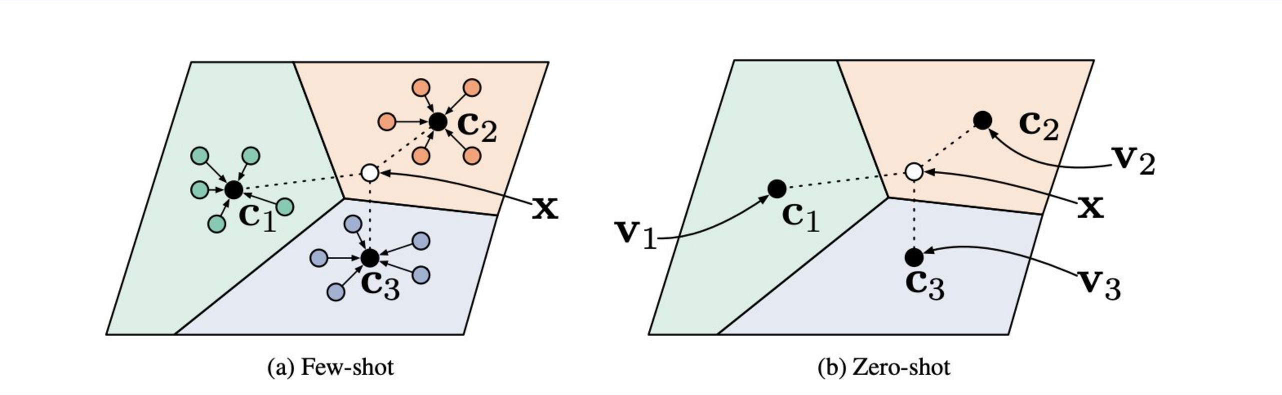
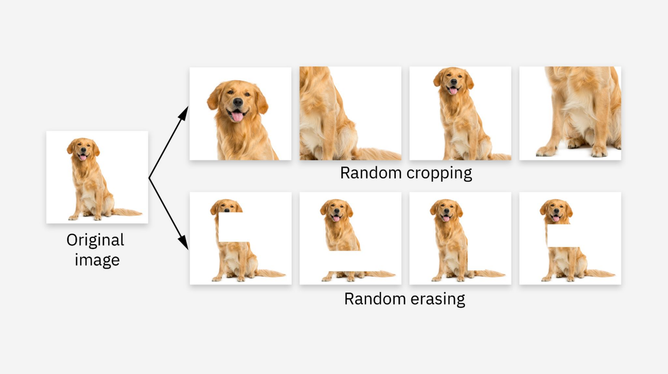


Image source: https://arxiv.org/pdf/1703.05175

Low-Data Scenario

- Data augmentation: apply transformations or add noise to the data to “expand” the training dataset
- Synthetic data generation: generates new data that matches the distribution of known training data; works best when you have prior knowledge of what is in the dataset



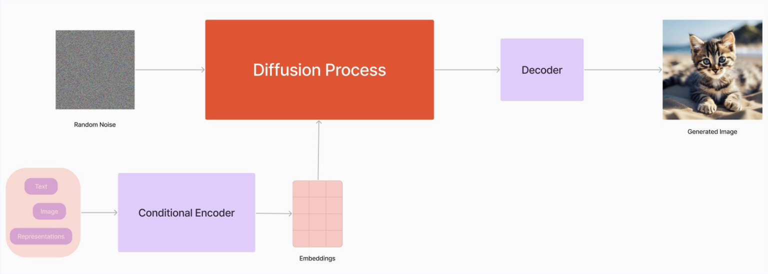


Image source: https://www.ibm.com/think/topics/data-augmentation
https://huggingface.co/learn/computer-vision-course/unit10/datagen-diffusion-models

Low-Data Scenario

- They (especially data augmentation) also work well for moderate to high-data scenarios but is especially useful for low-data scenarios to prevent your models from overfitting to the small dataset

Feature Engineering

- Sometimes a data point conveys more information to the model if you combine multiple features together

- e.g., for predicting taxi fares from the start and end point, you can calculate the distance between the two points to get a rough idea

Feature Engineering

- Check out the data preprocessing pipelines and the feature engineering process in https://www.kaggle.com/code/carsoncheng/taxi-fare-prediction

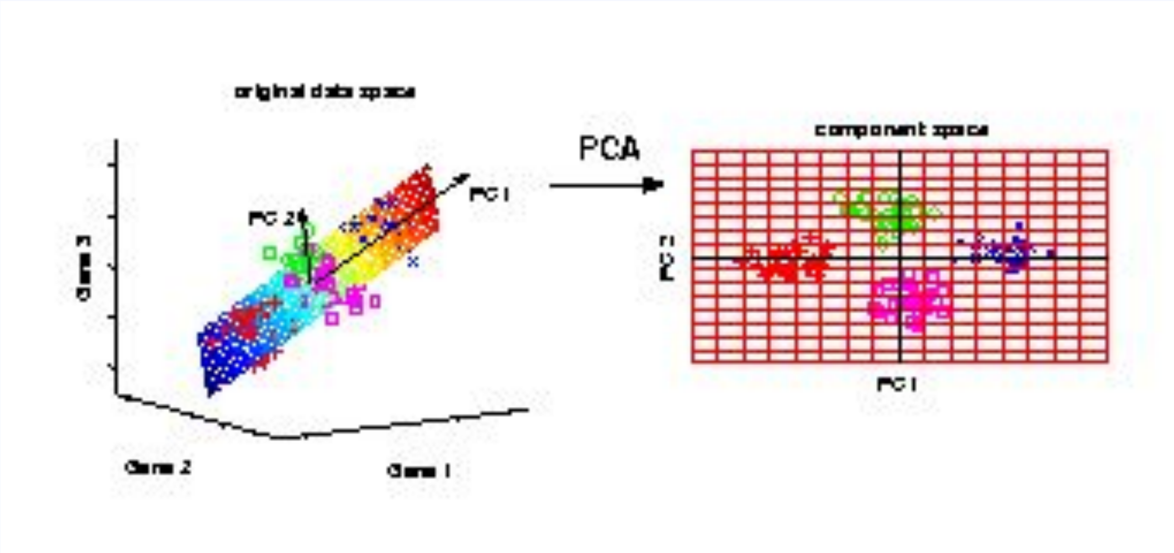
02

Data Visualization

You can enter a subtitle here if you need it

Dimensionality Reduction

- Sometimes you want to visualize data points with high-dimensional features(more than 1)
- Then reduce it to 2 dimensions to enable it to be visualized



Dimensionality Reduction

- PCA: linear method; can handle linear dependencies but cannot model nonlinear relationships;
- TSNE: nonlinear method; can handle nonlinear relationships but takes longer to run on large datasets