

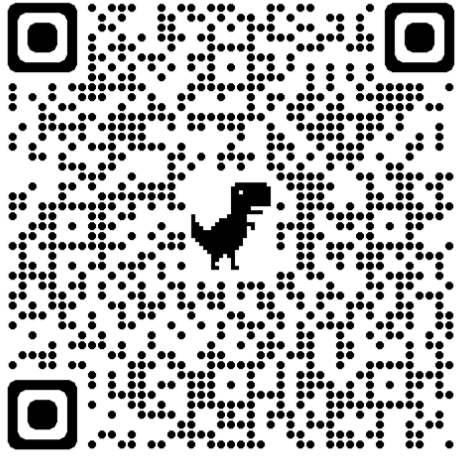


An AI Singapore Student Chapter

# Advanced ML Workshop

Day 1





Scan the QR code to mark your  
attendance



Attendance



# Learning Objectives



Creating ML Pipelines using Scikit-Learn package

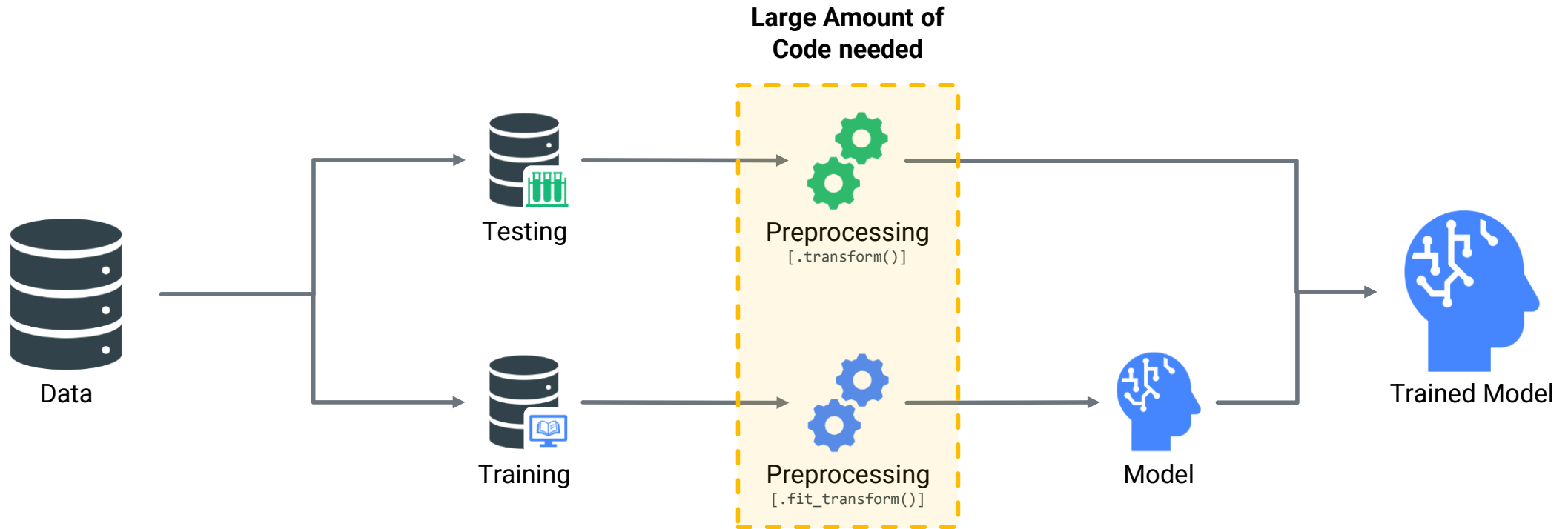


Implementing Data pre-processing techniques into pipelines:  
Encoding, Scaling, Imputation



Familiarize with additional pre-processing techniques:  
Log/Power Transformer, Column Transformer, Imbalanced Classes

# Data Preparation



# Potential Issues

**Large amount of code can:**



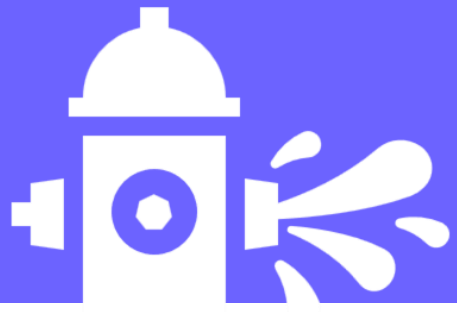
Result in errors when code is edited



Prone to data leakage as one might accidentally apply data pre-processing on entire dataset



Difficulty in debugging



# Data Leakage


**Data Leakage can occur when:**



Missing values are imputed with mean of Entire dataset before splitting



Standardization is applied on the entire dataset before splitting



Data leakage would result in  
unreliability in the test scores

Importance





# Analogy

## AI Model

- Model learns key traits (mean) and patterns (standardization)

## Student

- Students are given tips and hints before taking the test



# Pipelines





Pipelines consist of a sequence of steps to execute onto the dataset



What are Pipelines



# Benefits



Calling fit and predict once on your data



Avoids data leakage



Lesser amount of code which allows easier editing



# Knowledge Check

> Which of the following will cause data leakage?

- A. `StandardScaler().fit(X, y)`
- B. `StandardScaler().fit(X_train, y_train)`
- C. `OneHotEncoder().fit(X, y)`
- D. `OneHotEncoder().fit(X_train, y_train)`



# Pipeline Steps



Series of transformers, imputers and scalars and models



Executed sequentially. Sequence of steps is very important



2D array consisting an array of arrays which consists of the step name and the function

```
my_pipe = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),
    ('standardization', StandardScaler()),
    ('lr', LogisticRegression())
])
```

# Functions



`.fit()`

Apply all the pre-processing steps one after the other on the **entire** dataset parsed



`.predict()`

Transforms data and predict with the final estimator

Issue:

**Different** features have  
**different** characteristics  
and require **different** ways  
to handle it



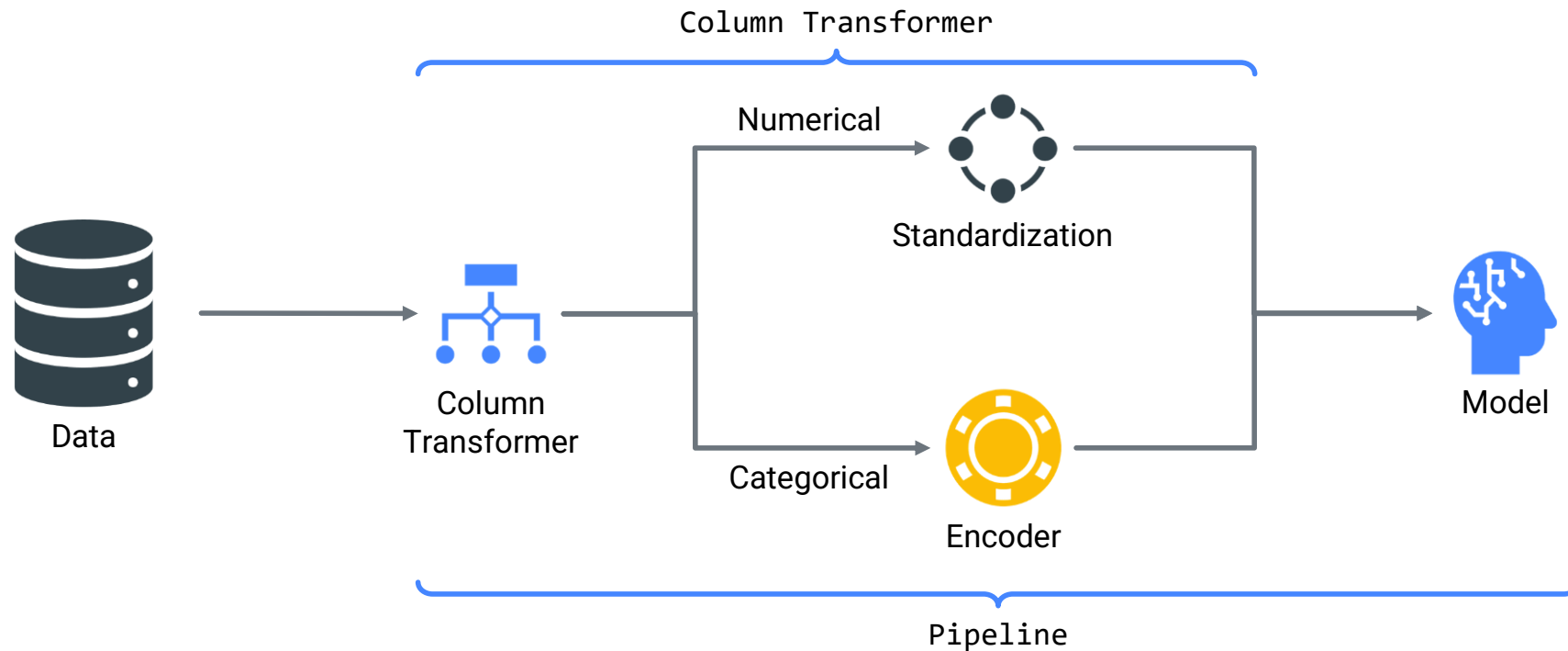
Applies pre-processing on specific  
column specified

ColumnTransformer





# Column Transformer in Pipeline



# Using Column Transformer

The diagram illustrates the components of the `ColumnTransformer` constructor call in the provided code snippet. Four labels with arrows point to specific parts of the code:

- Name**: Points to the variable `my_pipe`.
- Transformer**: Points to the `transformers` argument.
- Column(s)**: Points to the `column` argument.
- Passthrough /Drop**: Points to the `remainder='passthrough'` argument.

```
my_pipe = ColumnTransformer(transformers=[
    ('standardization', StandardScaler(), column),
], remainder='passthrough')
```



# Knowledge Check

> What are Machine Learning Pipelines?

- A. Pipelines help to transport water from one end to another.
- B. Pipelines are code that is shaped like a pipeline.
- C. Pipelines contains a sequence of steps that it will execute for your dataset.
- D. Pipelines are functions to use with no initialization to pre-process your data.

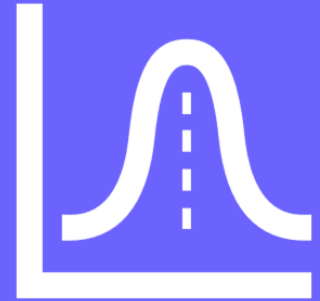


# Knowledge Check

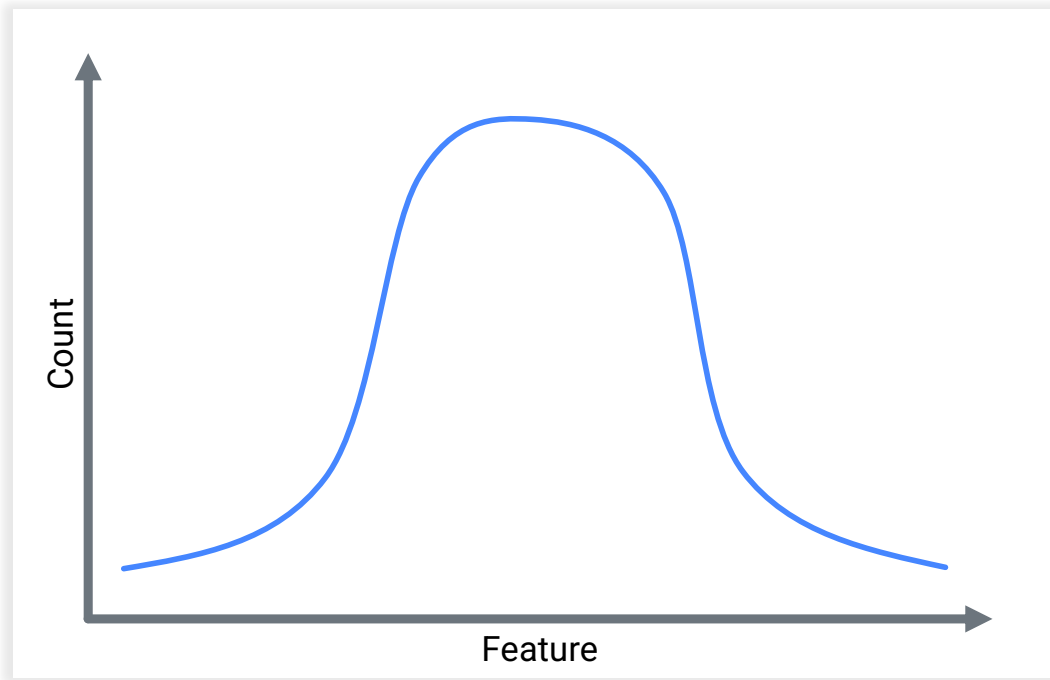
> The sequence of a Machine Learning Pipeline matters

- A. True
- B. False

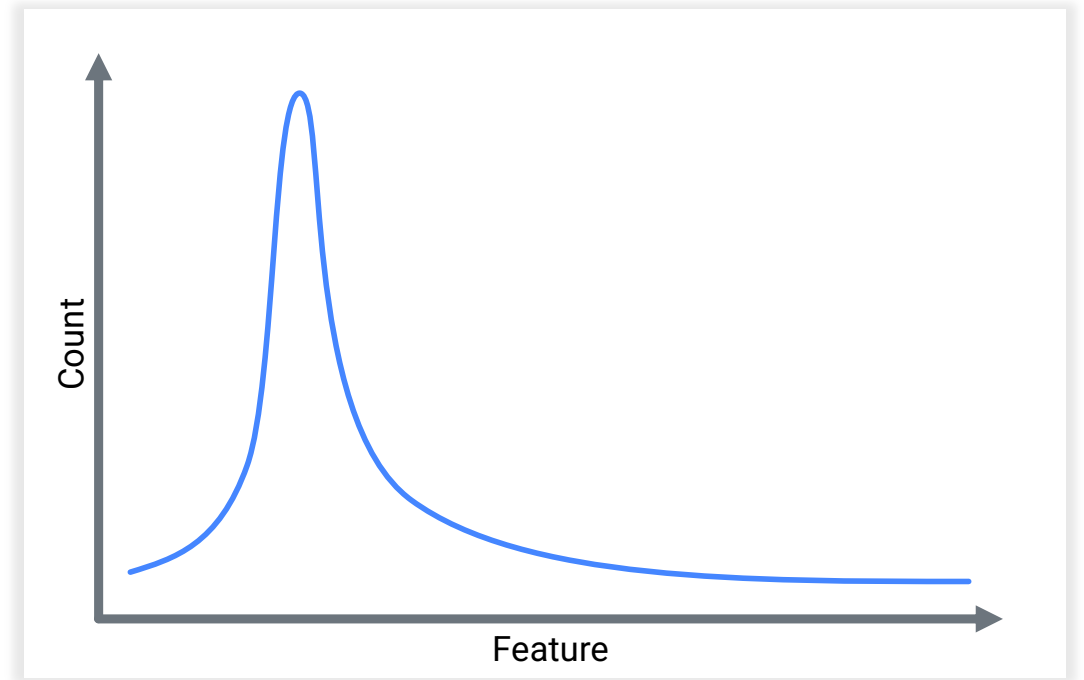
# Log Transform



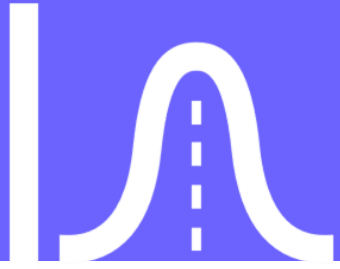
# Normalised Distribution



“Un-skewed” Distribution



Skewed Distribution



# Purpose



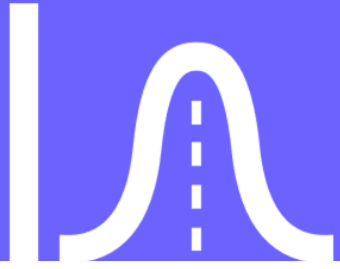
Models work best with an “un-skewed” distribution



“Un-skews” data



Keeping the relative distance between data points the same  
(crucial for Parametric and distance based models)



# np.log

## np.log

- Does not have any system in place
- Can consider np.log2 or np.log10

## np.log1p

- Adds 1 to all data in the selected columns
- Allows tiny non-zero values to still be scaled



# Using log transform

```
ft = FunctionTransformer(func=np.log1p)
```



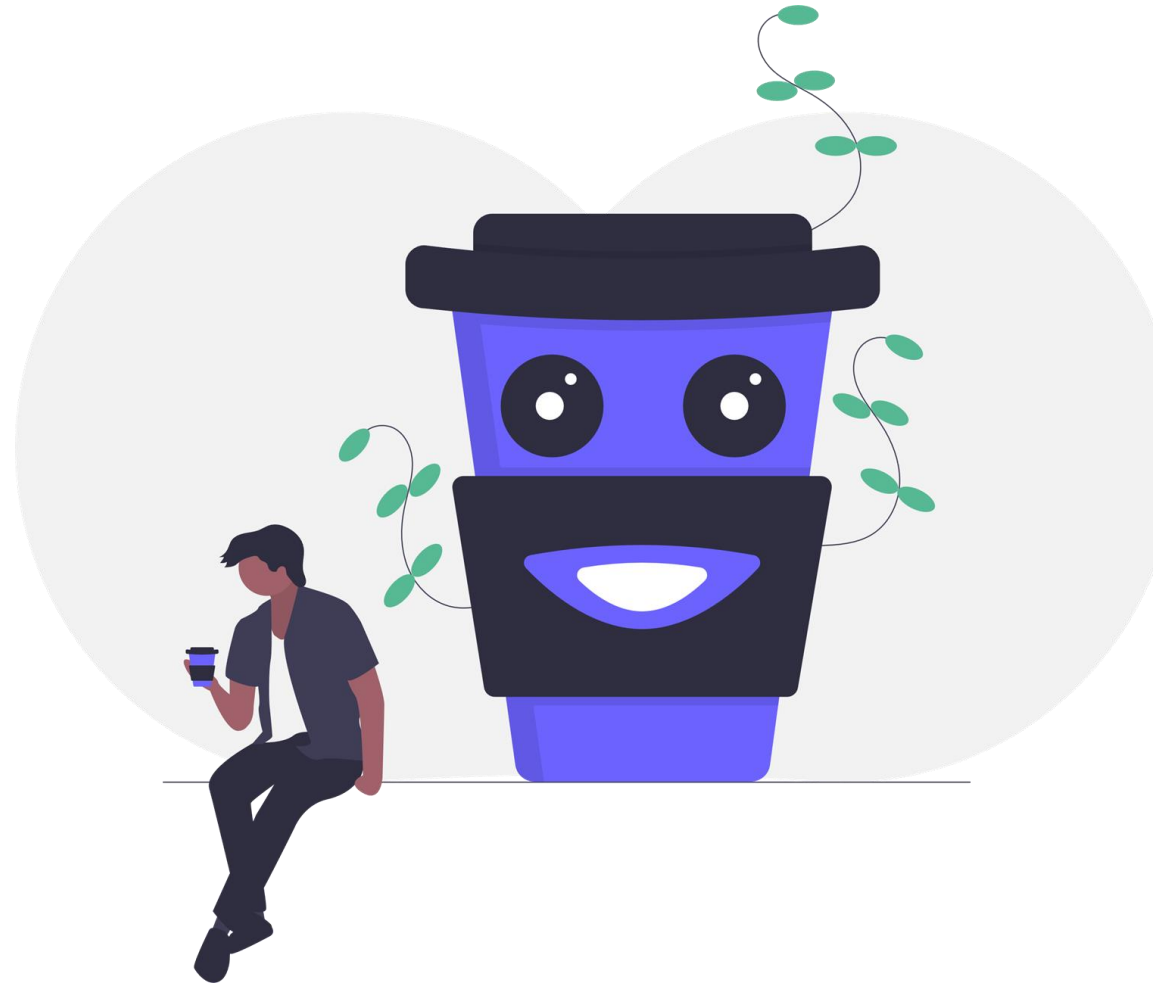
# Knowledge Check

> You should use `np.log` instead of `np.log1p` when there's many 0 values.

- A. True
- B. False

# Break

10 minutes



# Imbalanced Classes

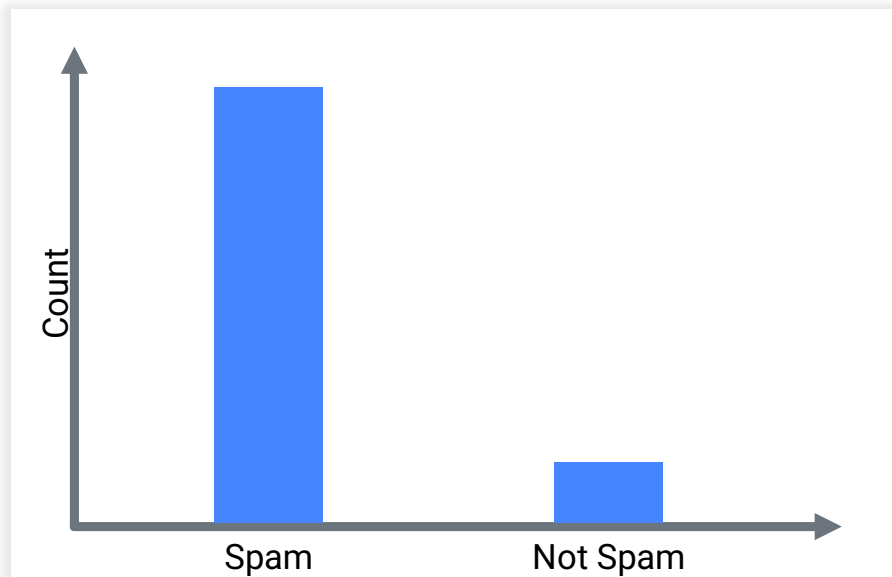




When total number of data for each class is marginally different



**What is Imbalance Class**



Example of dataset with imbalance class

Number of spam emails is significantly higher

# Issue

**Imbalance classes  
causes model to only  
predict the majority  
class**



# Solution



Under sample



Over sample





Under Sample

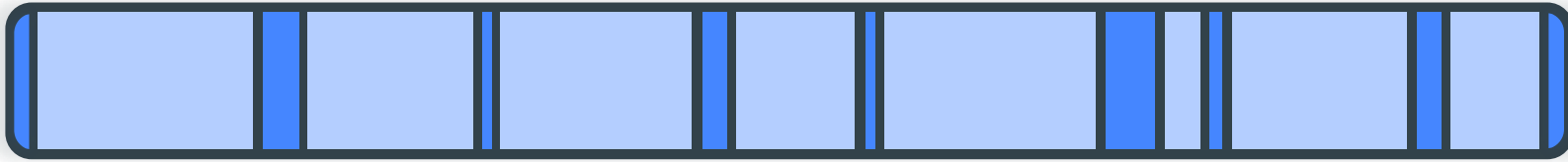
Take small samples from majority class

Majority class will have same size as minority

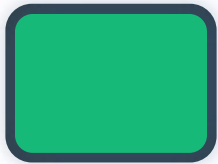
Causes us to mistakenly discard useful data

# Under Sample

spam



not spam



Before

# Under Sample

spam



not spam



After



Over sample

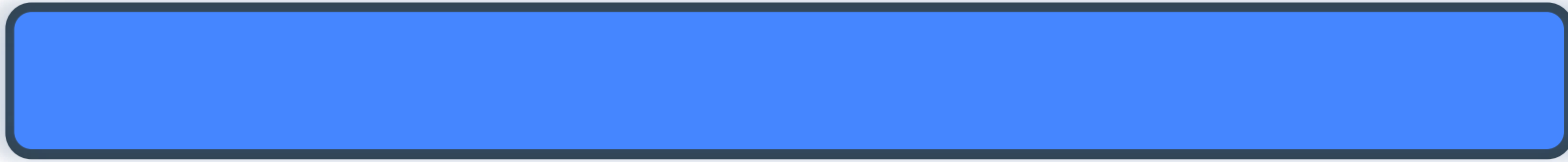
Generate synthetic data for minority class

Size of minority class will be same as majority

Potentially causes model to overfit

# Over Sample

spam



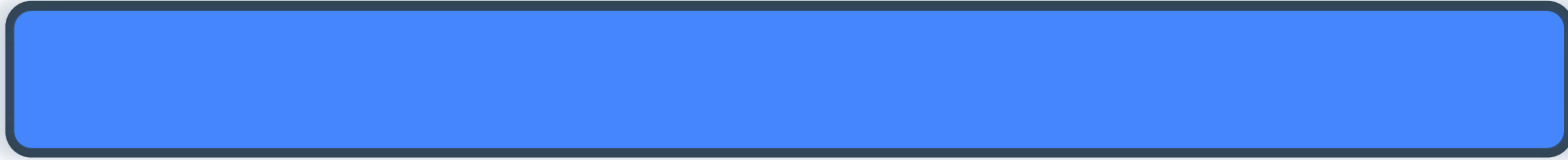
not spam



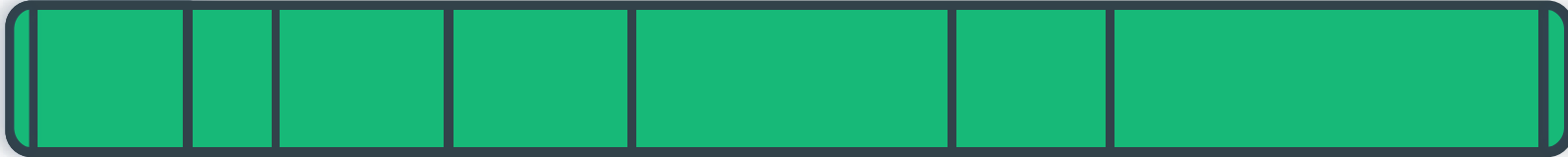
Before

# Over Sample

spam



not spam



After



# Notes



Do try both methods to see which one performs better



Make sure to over/under sample training data **ONLY**



This is because testing data is evaluated with metrics such as accuracy, precision and f1-score



# Samplers



## RandomUnderSampler

A naïve technique that randomly deletes data points in the majority class. In other words, it selects majority class data points that closest to a minority class



## RandomOverSampler

Duplicates some of the original samples of the minority classes





# SMOTE



Stands for Synthetic Minority Over-Sampling Technique



Method in imblearn library

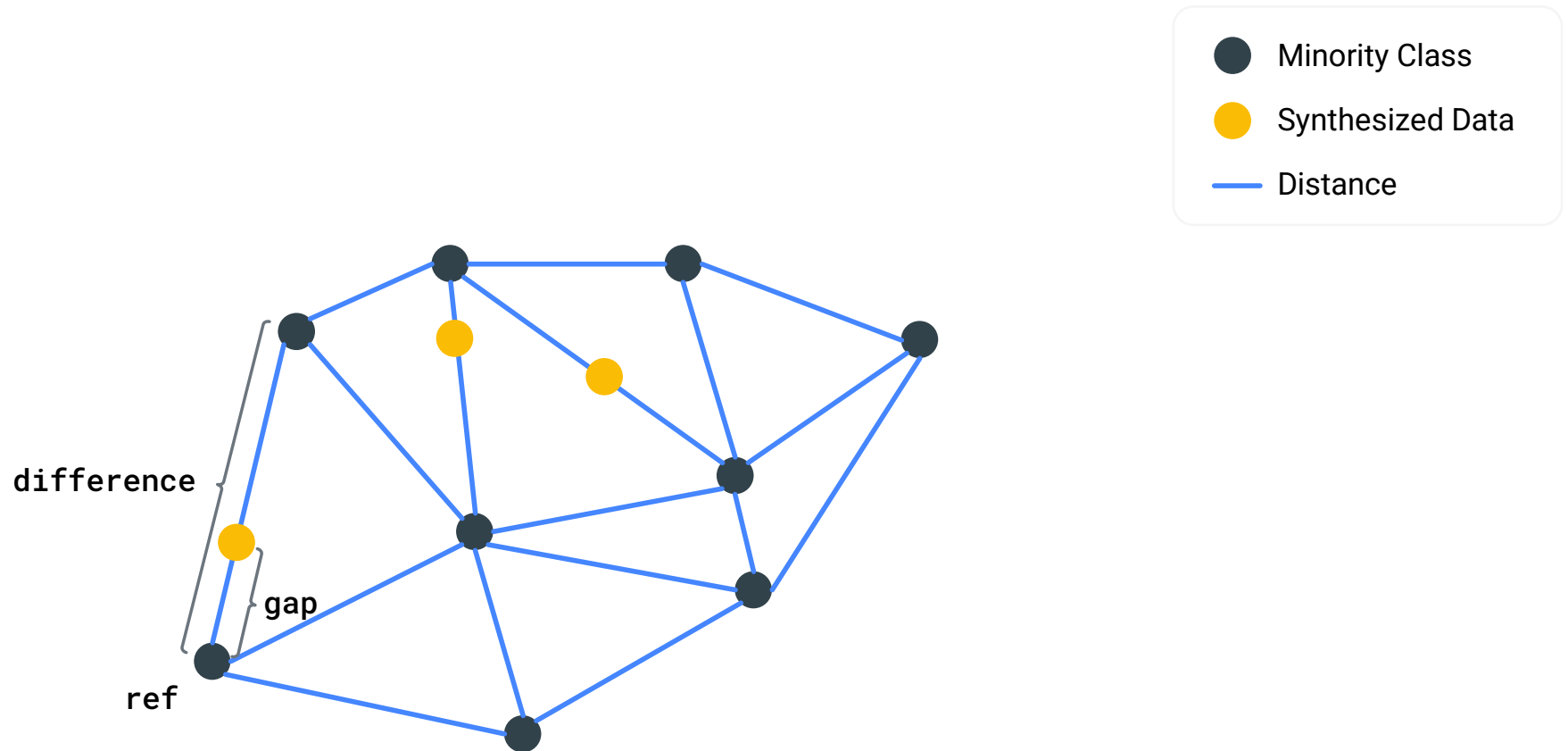


Allows you to increase size of minority class in balanced way



Calculates synthesized data

# How SMOTE works



$$\text{Synthesized data} = \text{ref} + \text{gap} * \text{difference}$$



# ADASYN



Stands for **Adaptive Synthetic**



Increases minority class in a balanced way



Similar to SMOTE, however ADASYN's is dependent on density of minority class in the region



# Comparing the Samplers

## Random Sampler

- Does not require data from other classes when selecting data points
- More prone to overfitting as it duplicates data points

## SMOTE & ADASYN

- Need data from other classes due to its formula
- Less prone to overfitting as it creates data points



# Knowledge Check

> What is the difference between data and synthetic data?

- A. There is no difference
- B. Data comes from the original dataset, while synthetic data is generated based on other data points
- C. Data is taken from the dataset, while synthetic data is generated randomly



# Knowledge Check

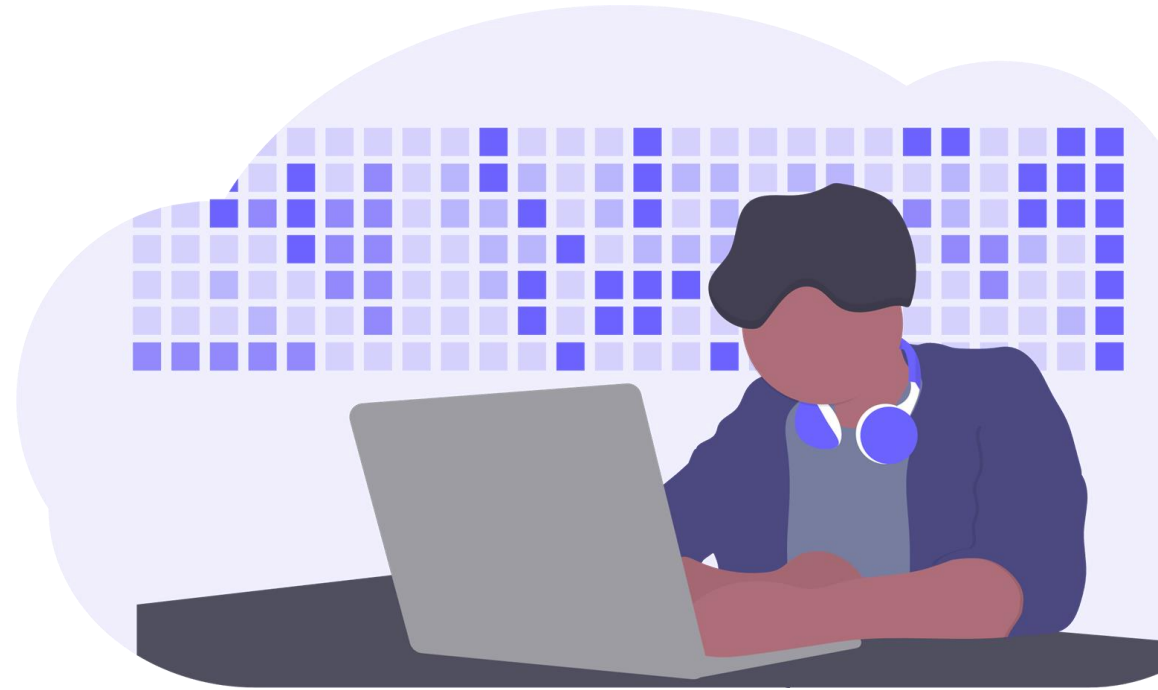
> If you want to retain as much data as possible, yet need the classes to be balanced, which approach should you take

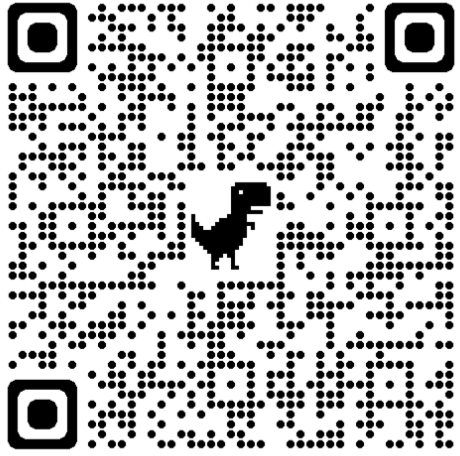
- A. Over sampling
- B. Under sampling



An AI Singapore Student Chapter

# Thank You





Scan the QR code to mark your  
attendance



**Attendance**