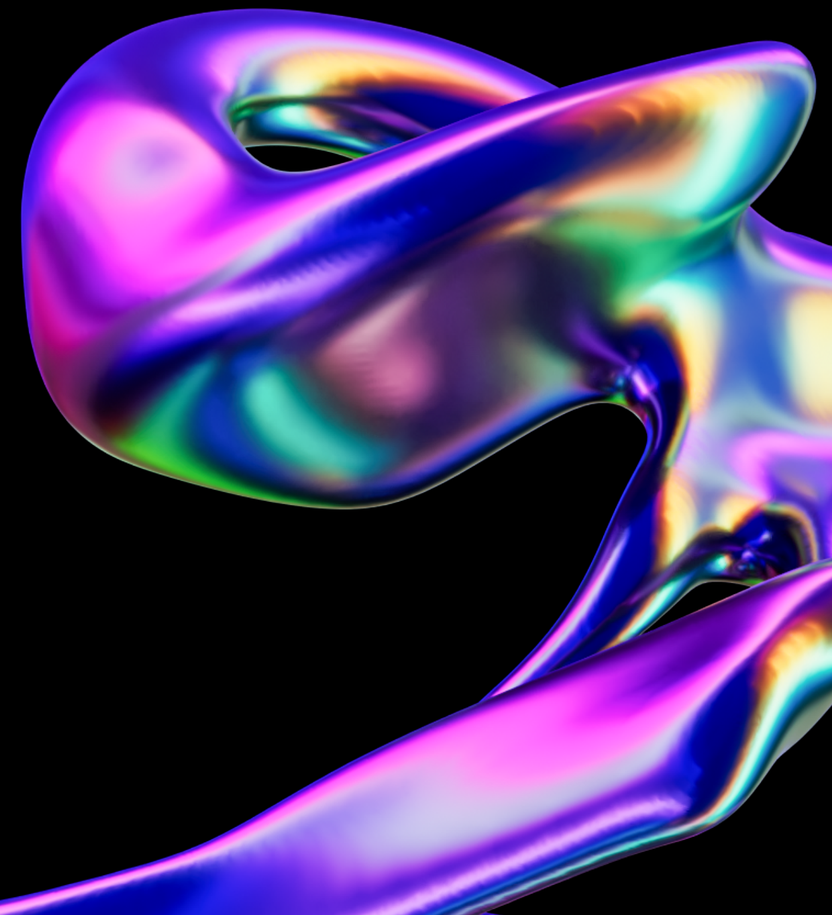
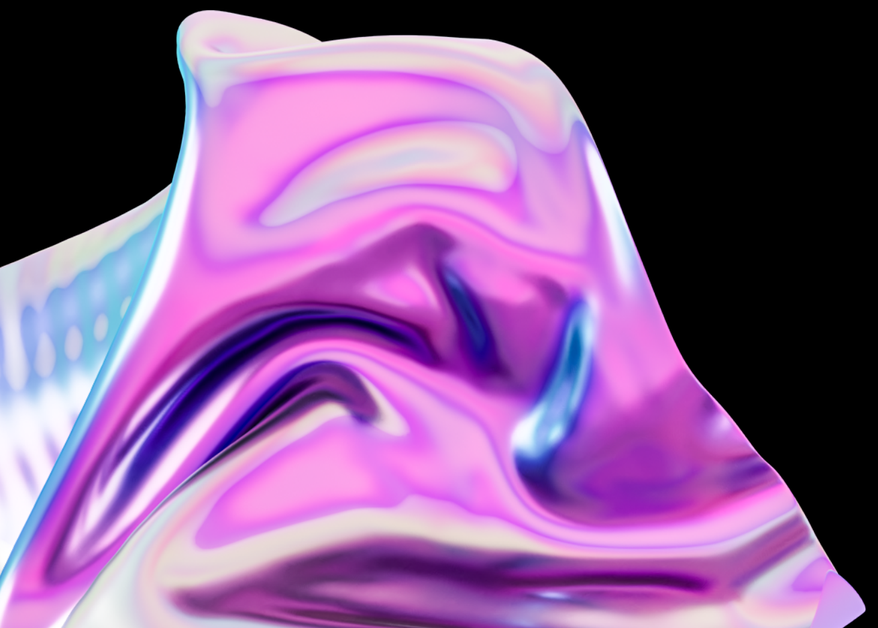
# Machine Learning Project

# Agenda

- **NUMERICAL**
  - INTRODUCTION
  - DATASET
  - DESCRIBTION
  - PREPROSSESING
  - ML MODELS
- **IMAGE**
  - INTRODUCTION
  - DATASET
  - DESCRIBTION
  - PREPROSSESING
  - ML MODELS



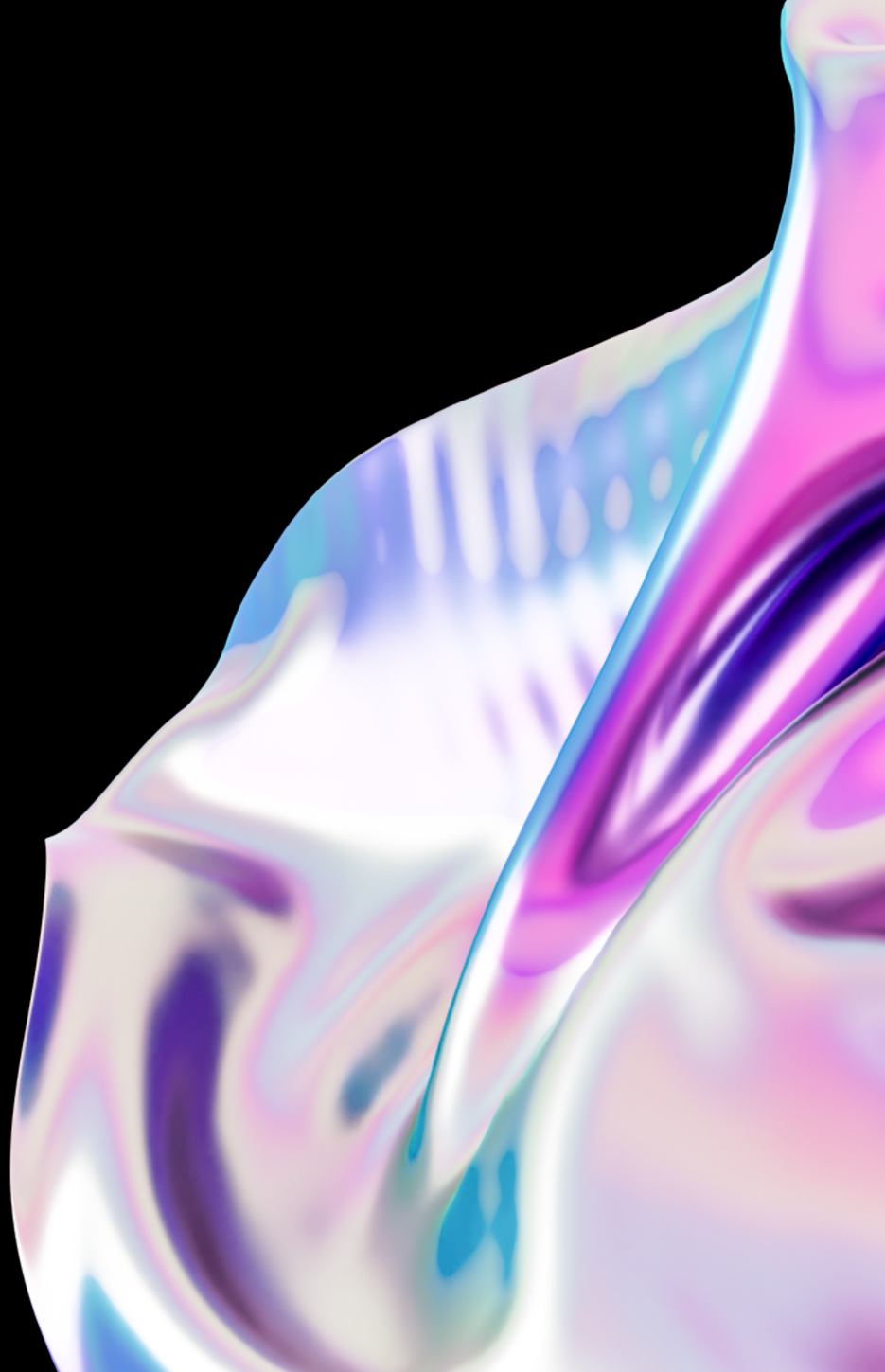


# 1

## House Sales in King County, USA

# Introduction

- IN THIS REPORT, WE PRESENT A MACHINE LEARNING MODEL IS DEVELOPED FOR PREDICTING HOUSE PRICES USING LINEAR REGRESSION AND KNN.

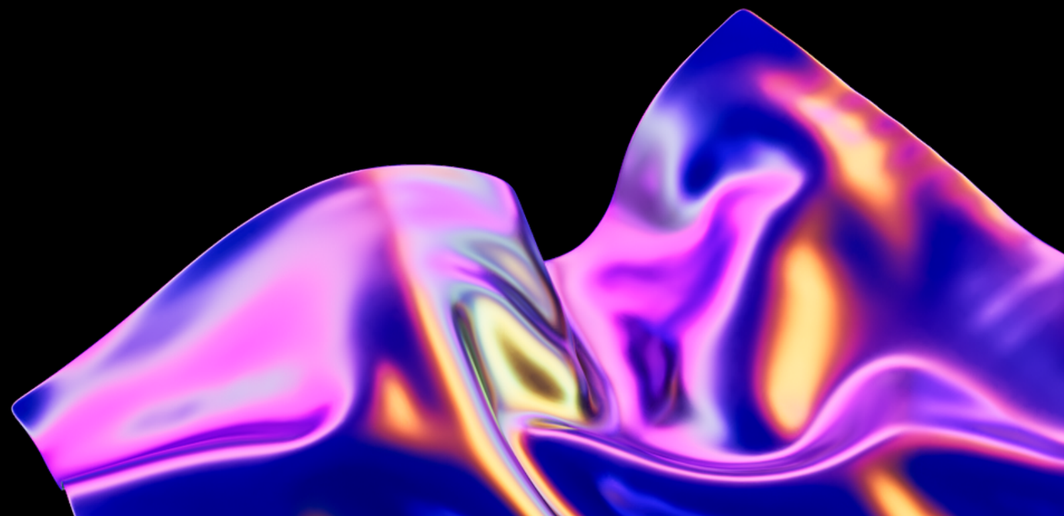


# Numerical dataSet

House Sales in King County, USA

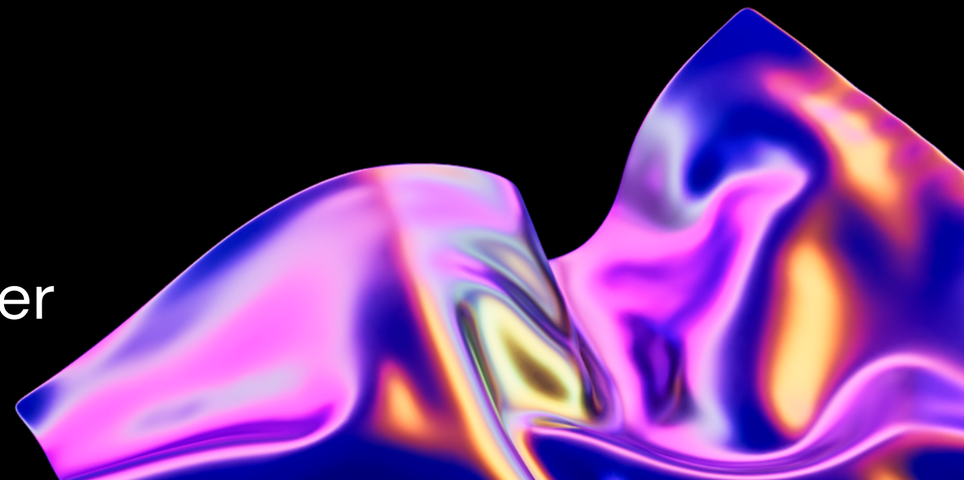
# Data Describtion

- THIS DATASET CONTAINS HOUSE SALE PRICES FOR KING COUNTY, WHICH INCLUDES SEATTLE. IT INCLUDES HOMES SOLD BETWEEN MAY 2014 AND MAY 2015.



# Features Description

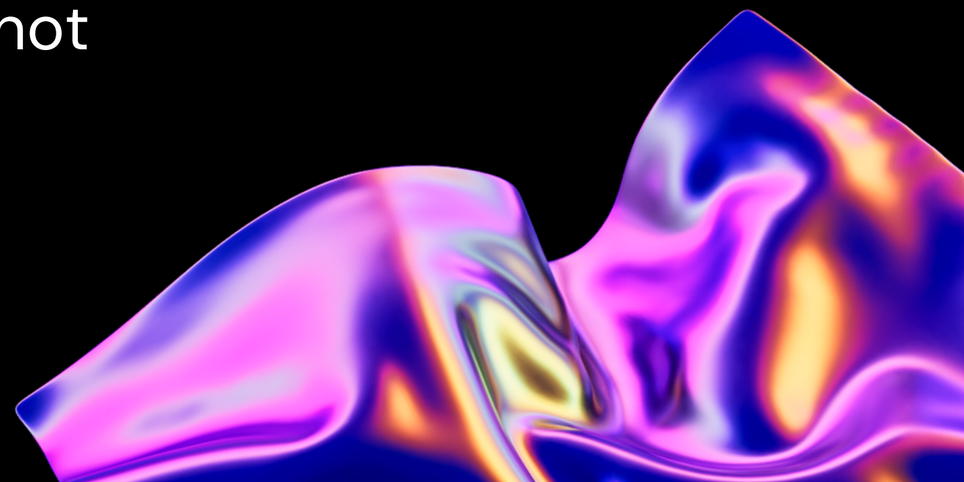
- **Id** : Unique ID for each home sold
- **Date** : Date of the home sale
- **Price** :  Price of each home sold
- **Bedrooms** : Number of bedrooms
- **Bathrooms**  : Number of bathrooms, where accounts for a room with a toilet but no shower



- **sqft_living** : Square footage of the apartments interior living space

- **sqft_lot** : Square footage of the land space

- **waterfront** : A dummy variable for whether the apartment was overlooking the waterfront or not

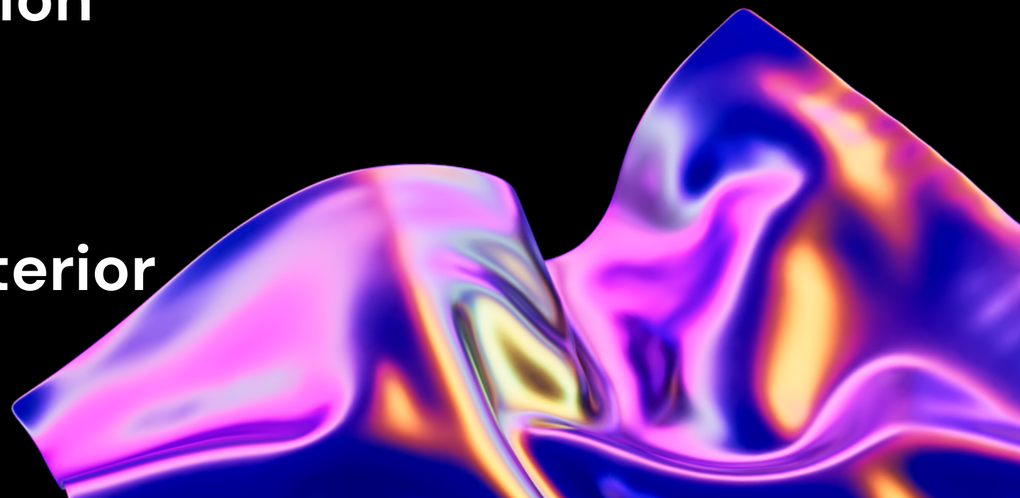


- View : An index from 0 to 4 of how good the view of the property was

- Condition : An index from 1 to 5 on the condition of the apartment

- Grade :  An index from 1 to 13, where 1–3 falls short of building construction and design, 7 has an average level of construction and design, and 11–13 have a high quality level of construction and design

- Sqft_above : The square footage of the interior housing space that is above ground level



- Sqft_basement : The square footage of the interior housing space that is below ground level

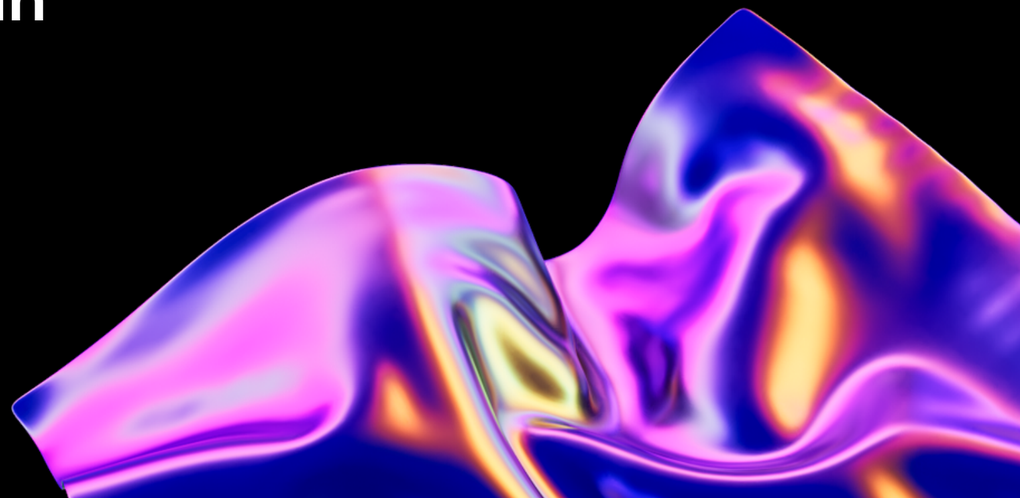
- Yr_built : The year the house was initially built

- Yr_renovated : Square footage of the land space

- Zipcode : What zipcode area the house is in

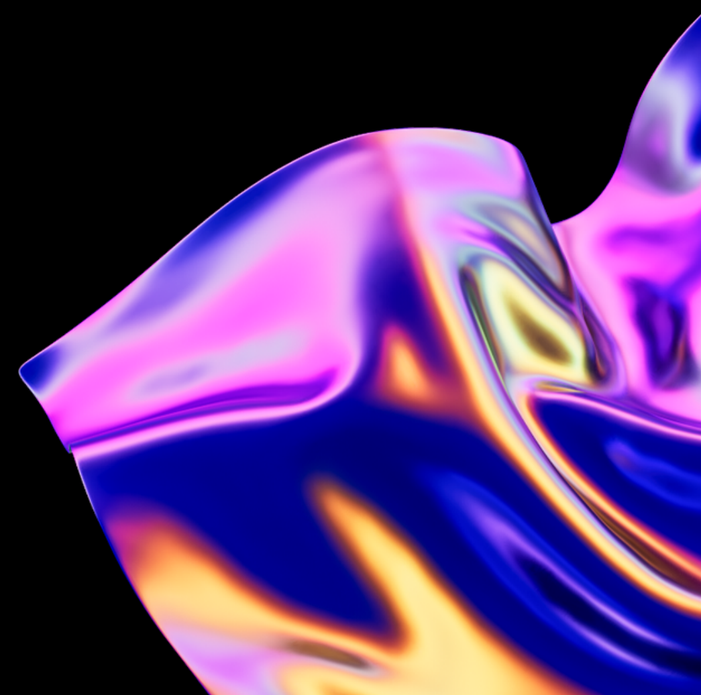
- Lat : Lattitude

- Lang: Longitude



# Data Before Preprocessing

- **SIZE OF DATA** : 2.52 MB
- **NUMBER OF FEATURES** : 21 FEATURE
- **NUMBER OF SAMPLES** : 21613 SAMPLE



# Data Preprocessing

```
[54]: df["yr_renovated"].value_counts().reset_index()
```

[54]:

| | yr_renovated | count |
|---|---|---|
| 0 | 0 | 20699 |
| 1 | 2014 | 91 |
| 2 | 2013 | 37 |
| 3 | 2003 | 36 |
| 4 | 2005 | 35 |
| ... | ... | ... |
| 65 | 1951 | 1 |
| 66 | 1959 | 1 |
| 67 | 1948 | 1 |
| 68 | 1954 | 1 |
| 69 | 1944 | 1 |

```
df["waterfront"].value_counts().reset_index()
```

[52]:

| | waterfront | count |
|---|---|---|
| 0 | 0 | 21450 |
| 1 | 1 | 163 |

```
[53]: df["view"].value_counts().reset_index()
```

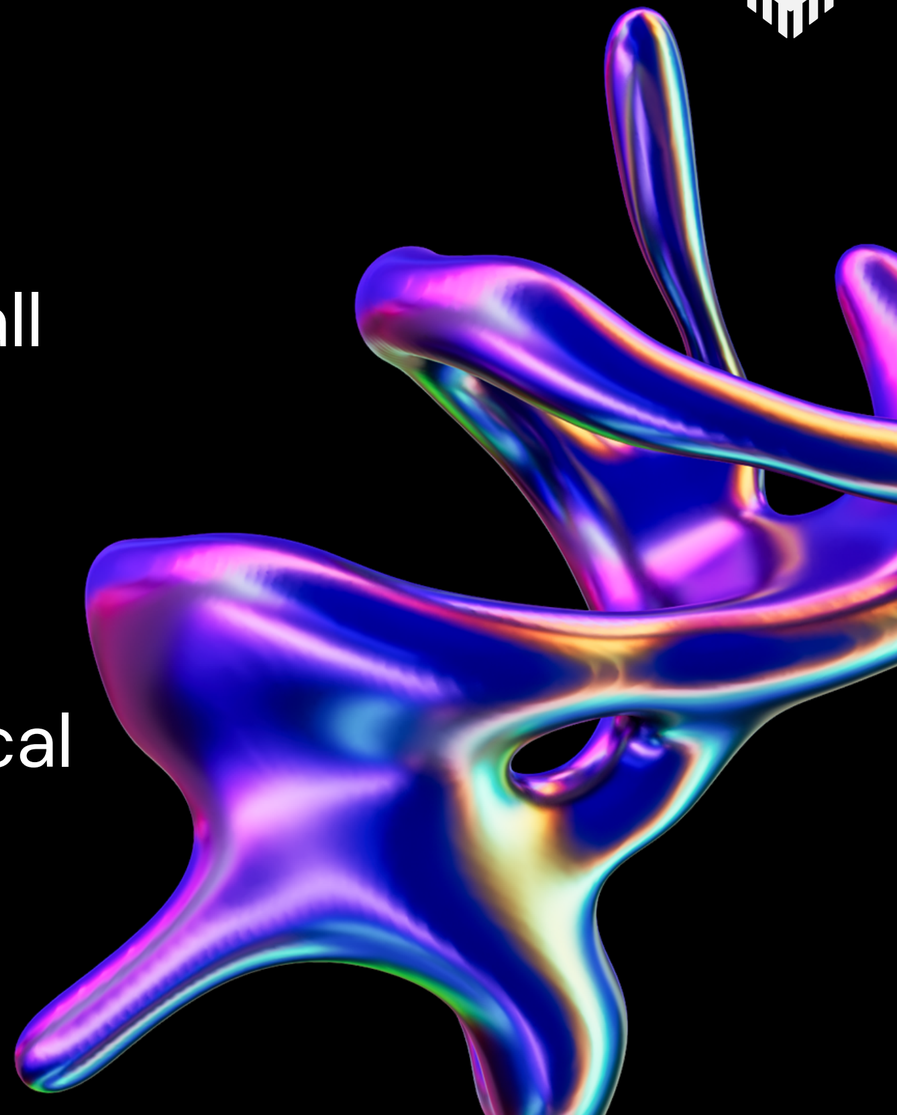
[53]:

| | view | count |
|---|---|---|
| 0 | 0 | 19489 |
| 1 | 2 | 963 |
| 2 | 3 | 510 |
| 3 | 1 | 332 |

Most Values of these features (yr_renovated, view, waterfront) are zero ,so we gonna drop them

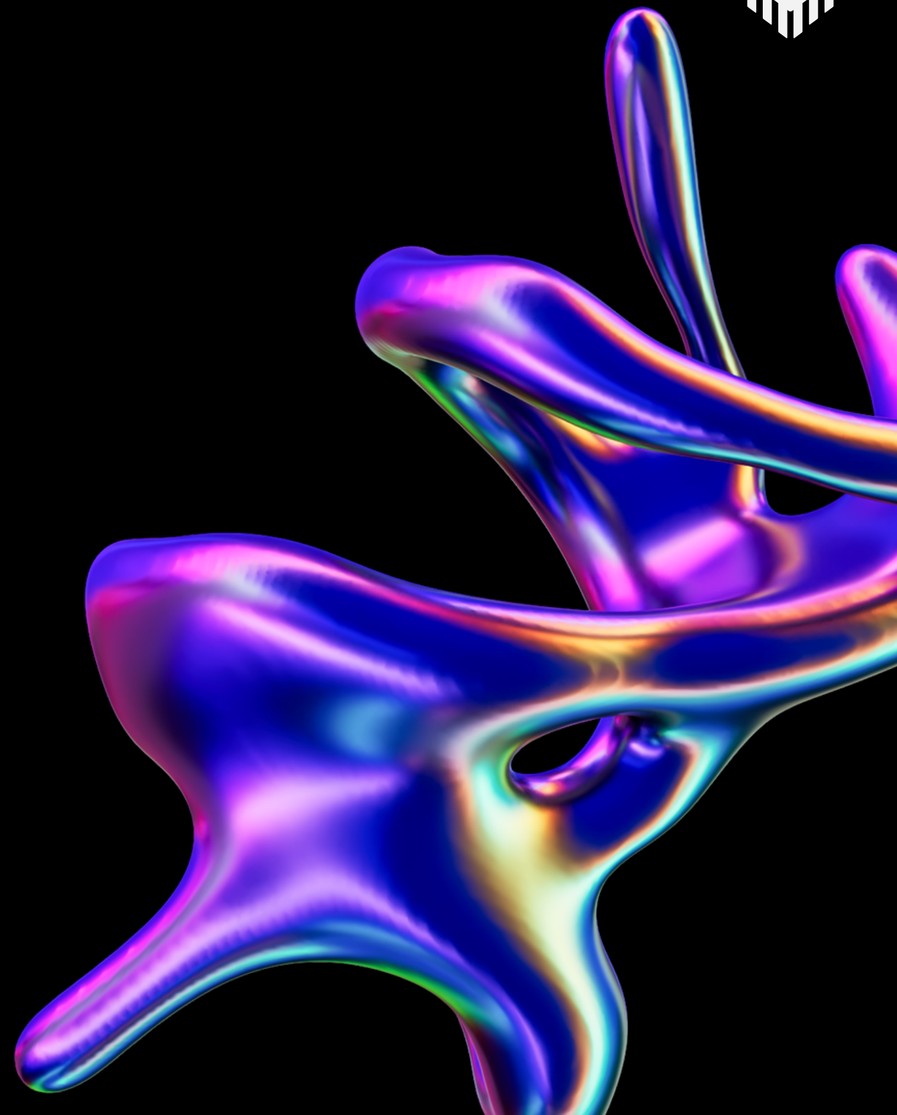
# Standard Scaler

- We scaling data with Standard Scaler To all column and ignore this column:
- "grade", "condition", "floors", "bathrooms", "bedrooms", “date”
- Because values in this column is categorical

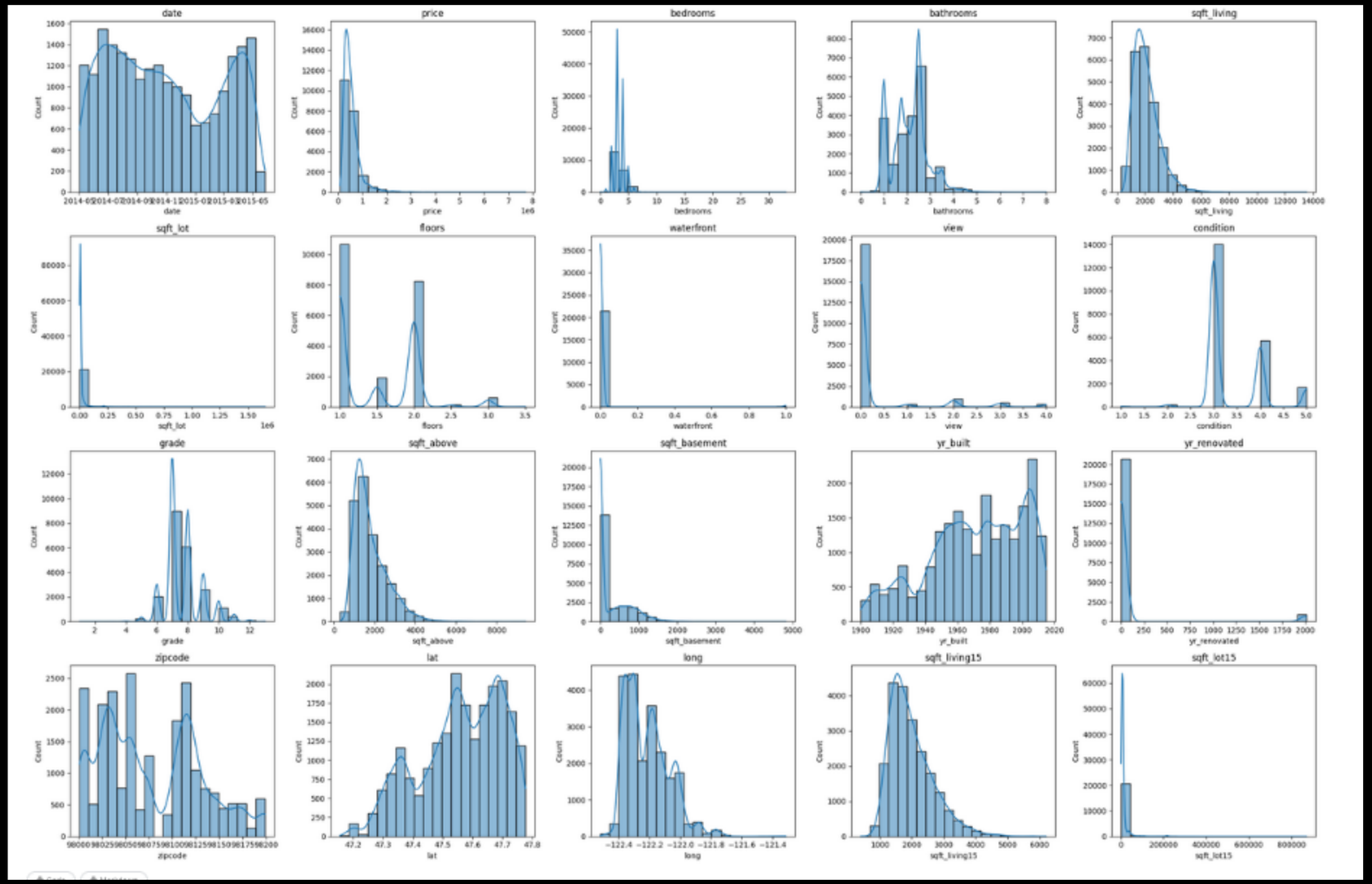


# Transform

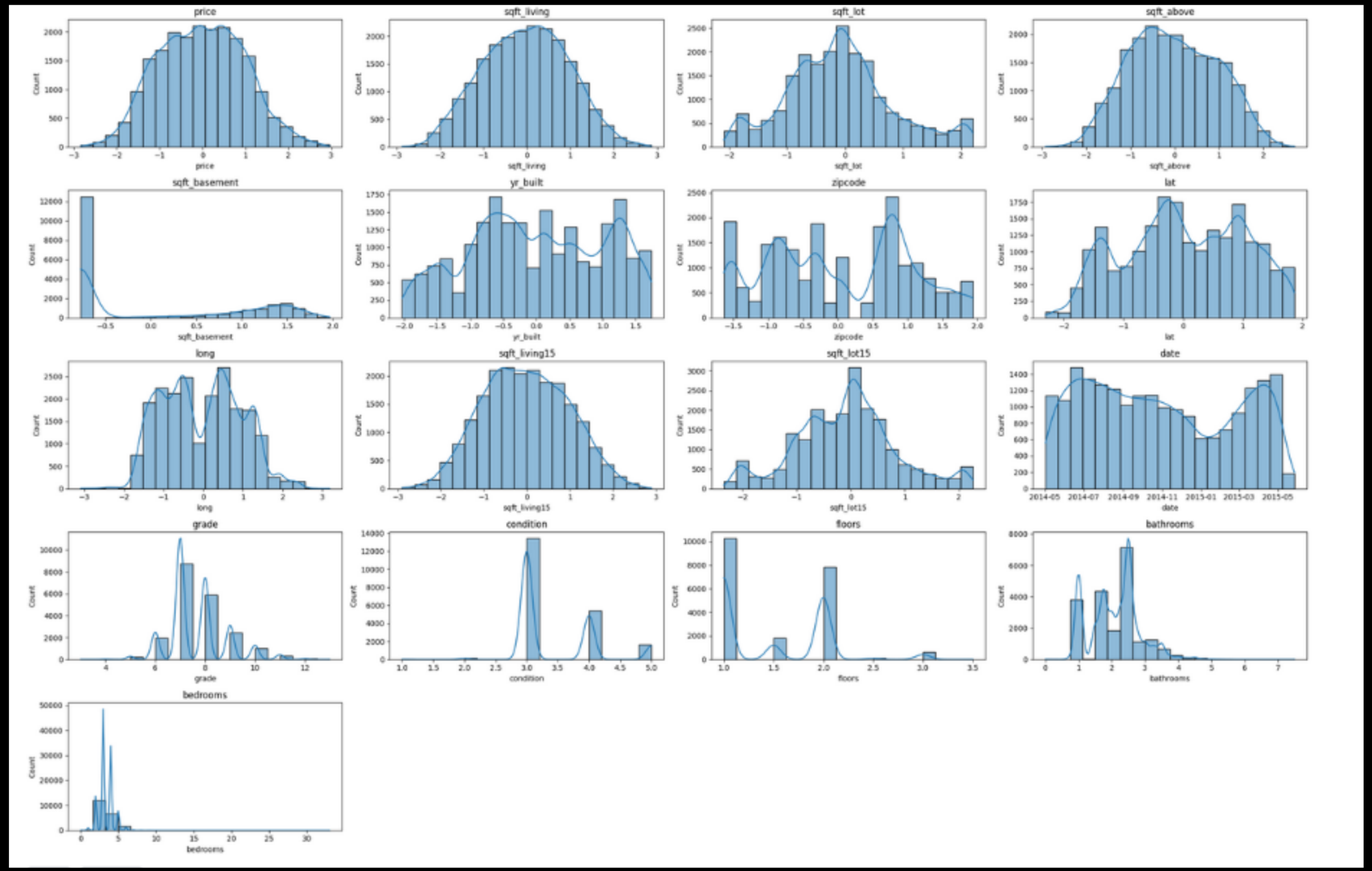
- transform data with power transform (method = yeo-johnson ) to make distribution of data normalized



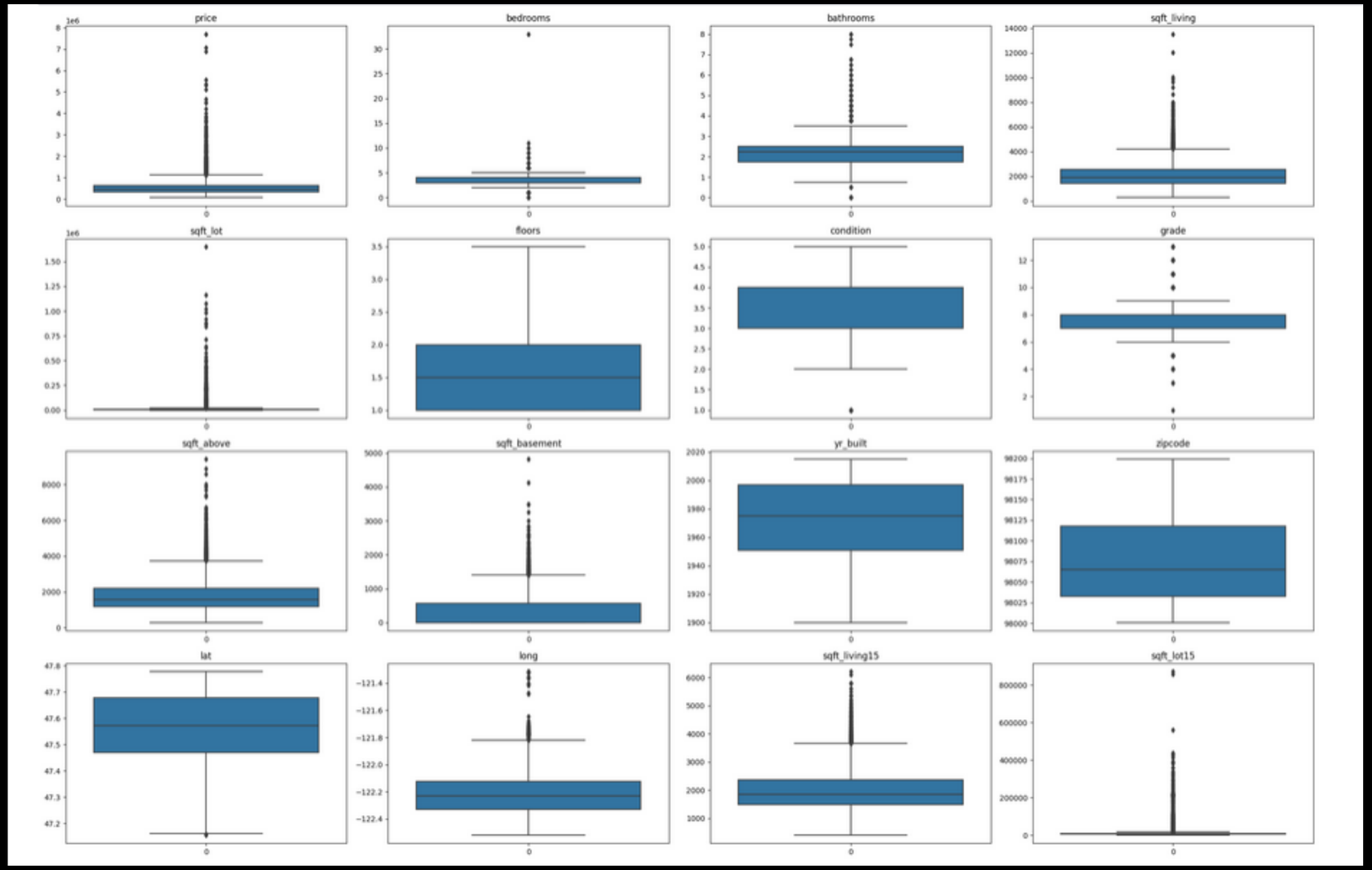
# Data Distribution (Before)



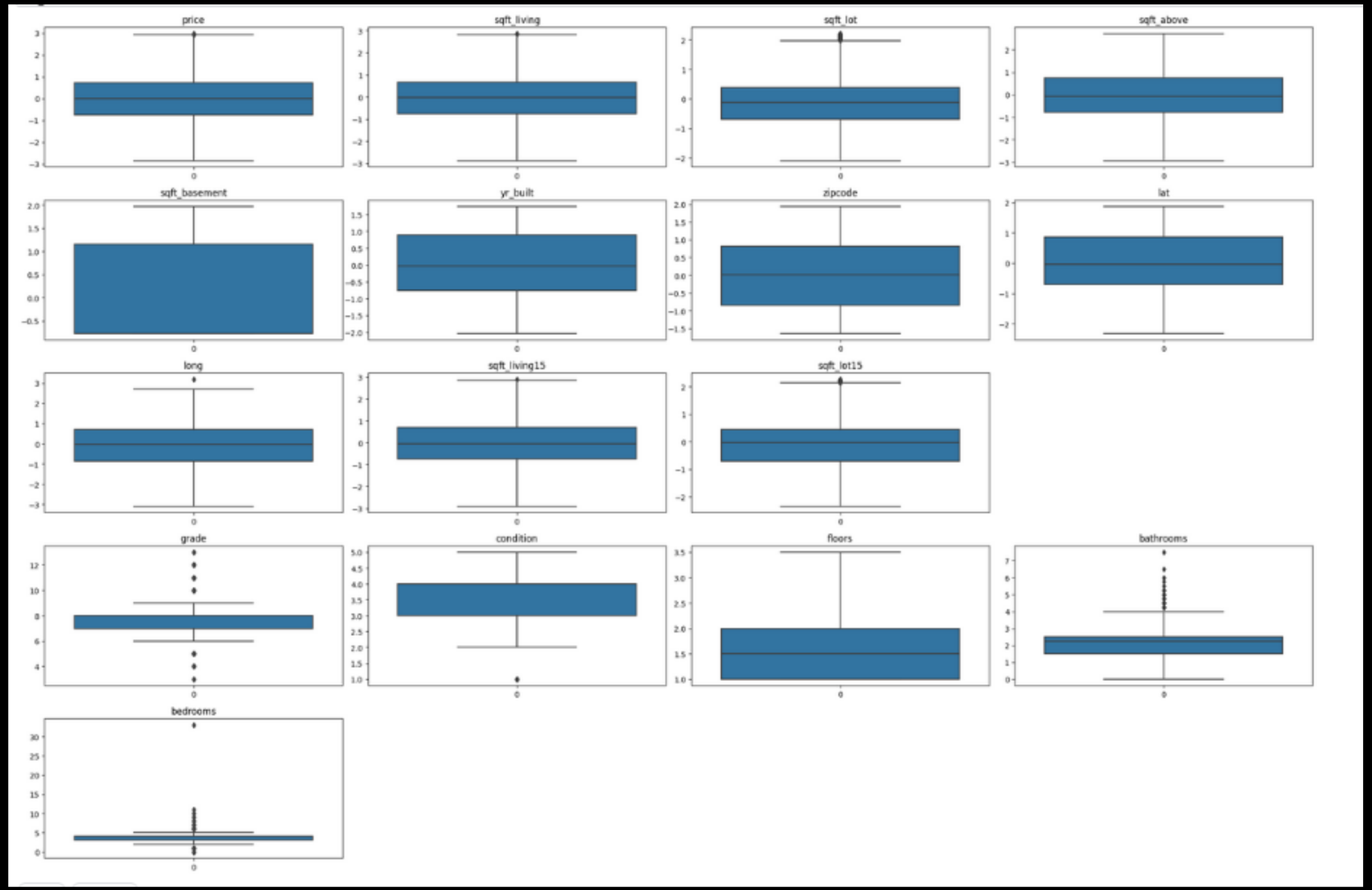
# Data Distribution (after)



# Outliers (Before)

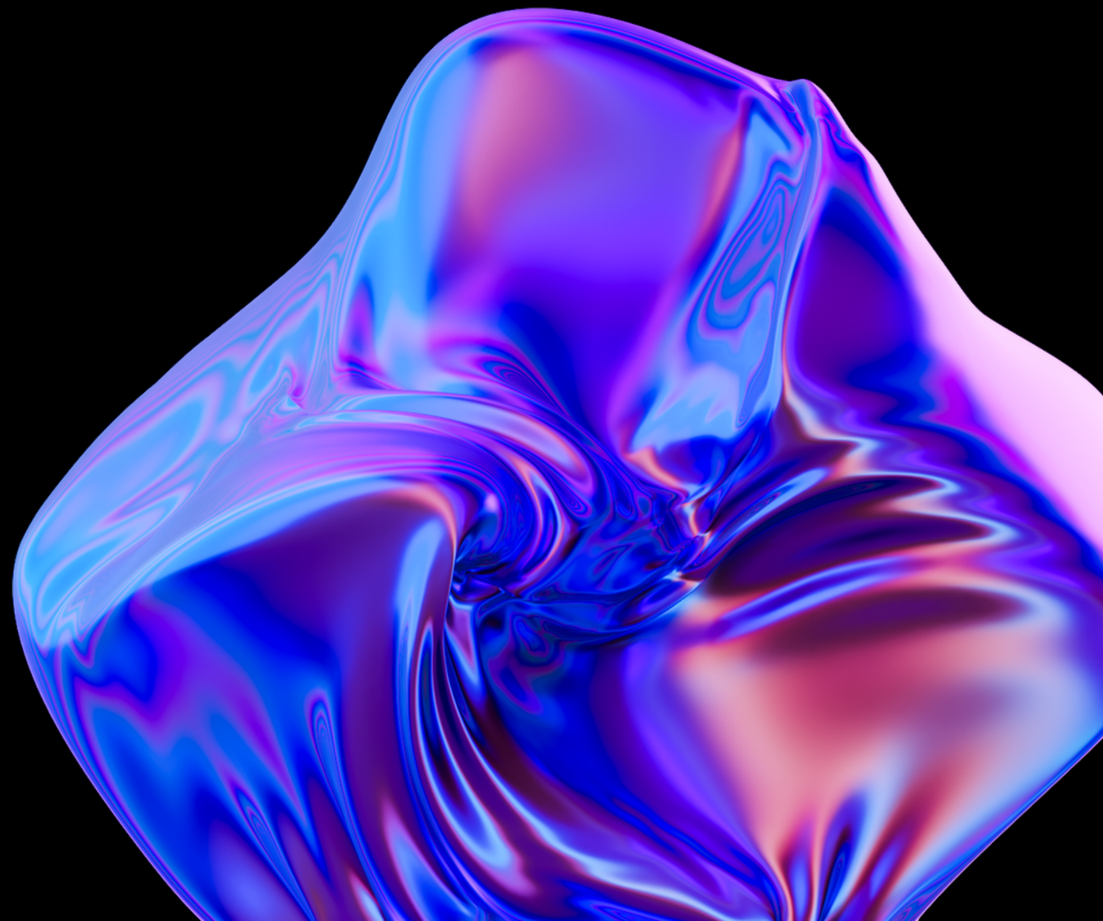


# Outliers (After)



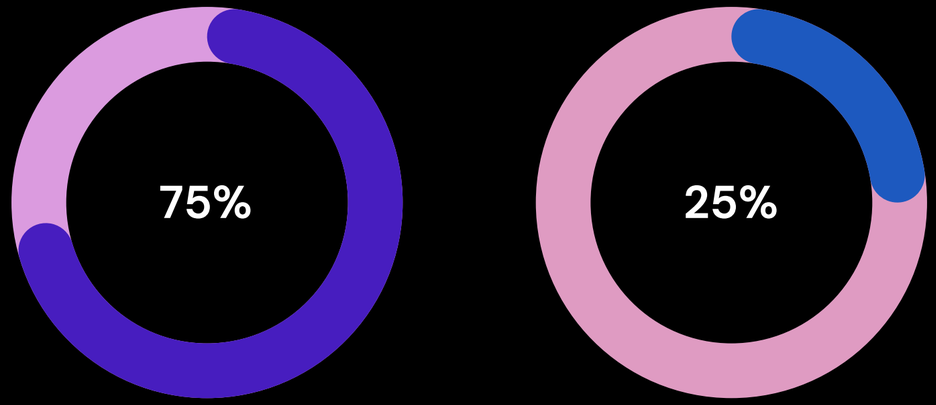
# Data After Preprocessing

- Features : 17
- Samples : 20698



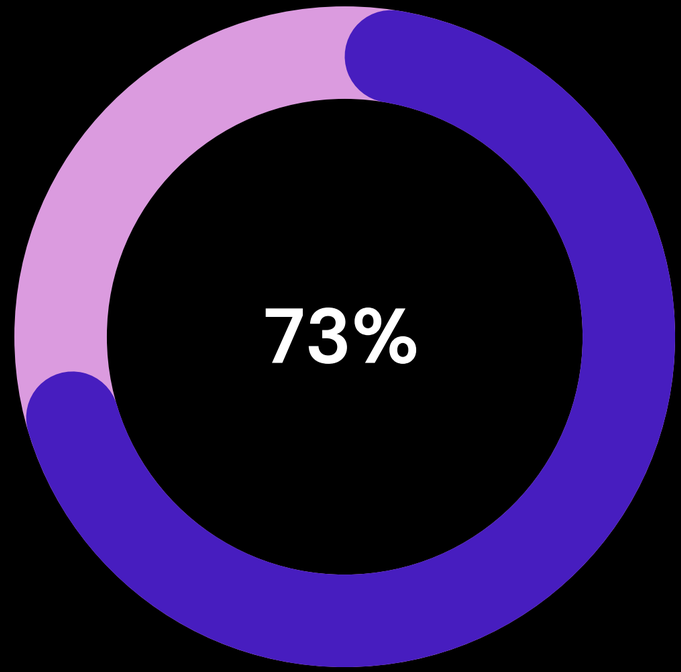
# Split Data

- we split data using  75% for train and  25% for test



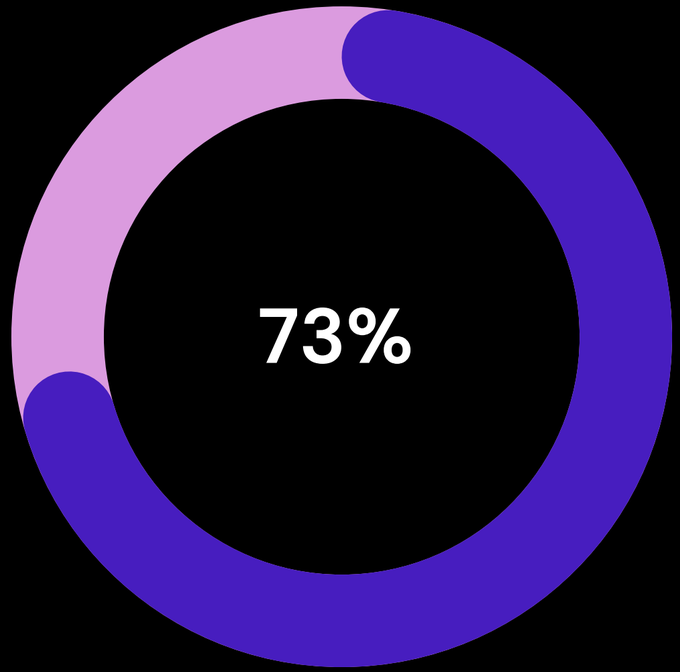
# Linear Regression

- Train accuracy : 72%
- Test accuracy : 73%
- R2_score : 73%
- mean_squared_error : 0.266
- mean_absolute_error : 0.4056



# Cross validation score
# cv = 3

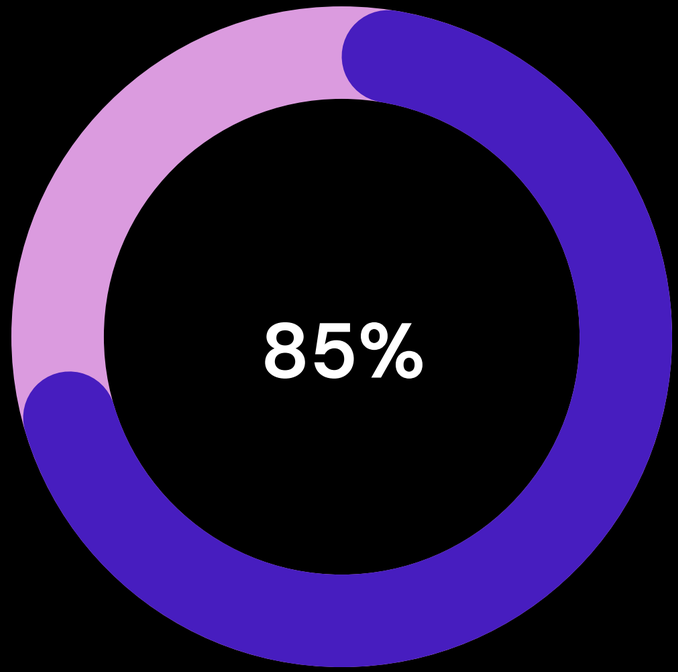
- first : 0.72400891
- second : 0.72103136
- third : 0.73081107



# KNeighbors Regressor

n_neighbors=7, weights="uniform", metric="manhattan"

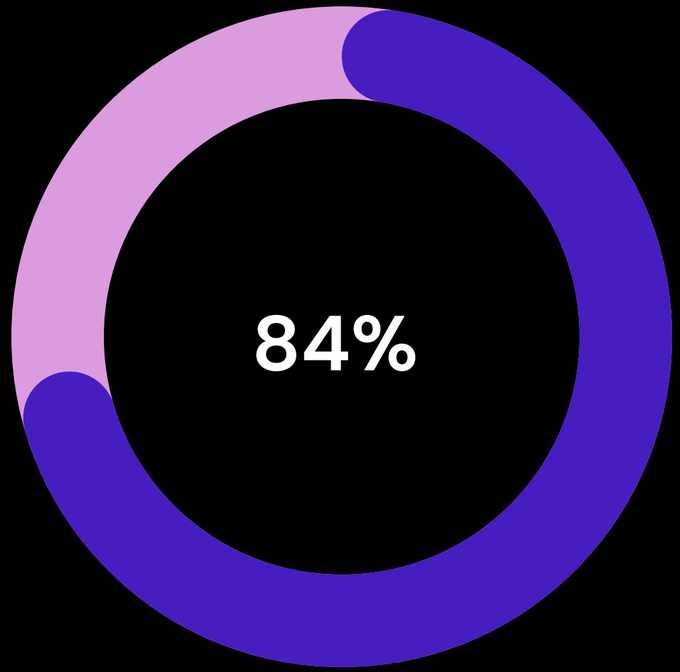
- Train accuracy : 88%
- Test accuracy : 85%
- R2_score : 85%
- mean_squared_error : 0.142
- mean_absolute_error : 0.2731



# Cross validation score

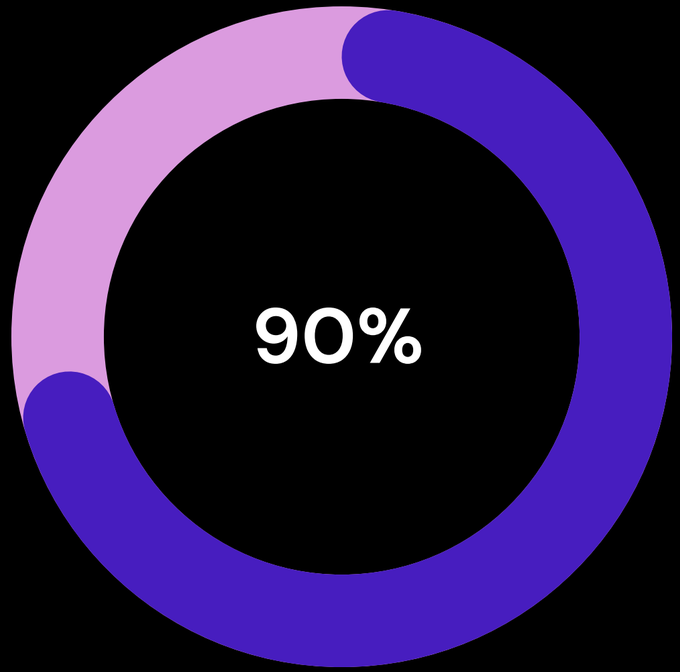
## cv = 5

- first : 0.85222322
- second : 0.84628954
- third : 0.84869059
- forth : 0.85263857
- fifth : 0.84769341



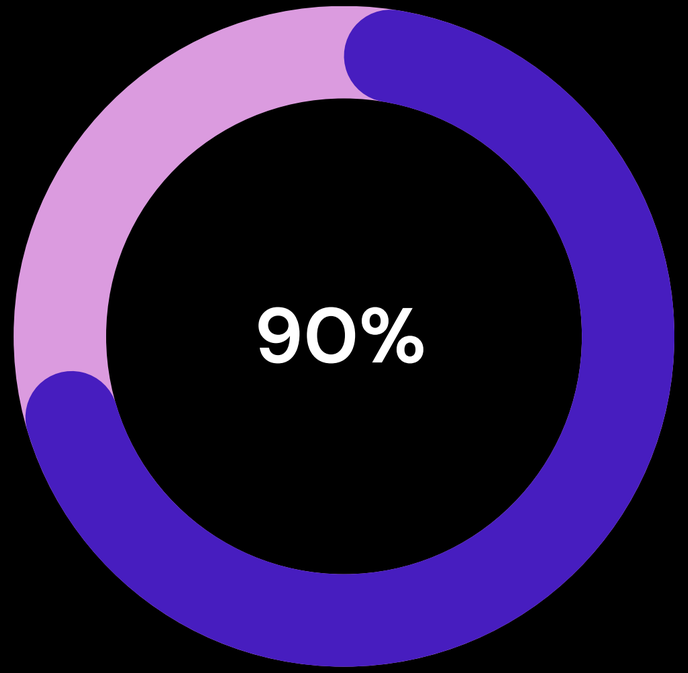
# XGBoost

- train accuracy : 0.95596613
- test accuracy : 0.90185858



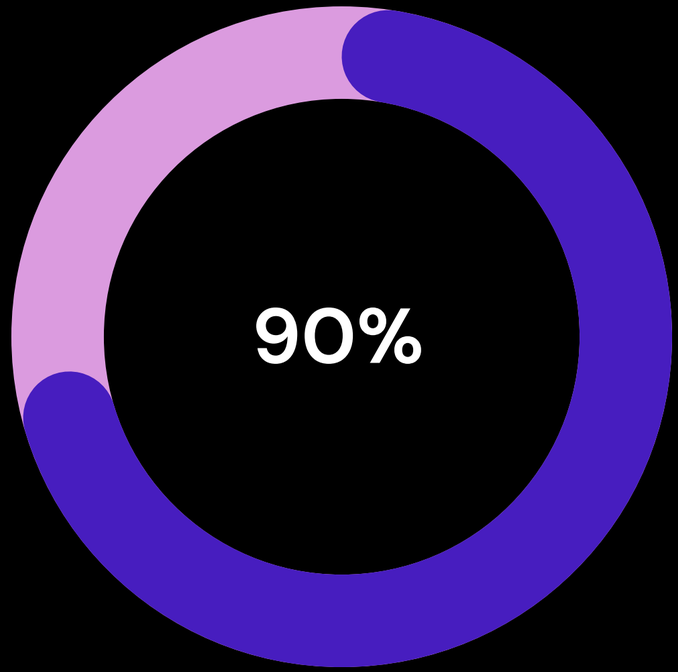
# XGBoost

- first : 0.89670989
- second : 0.89310655
- third : 0.90289082

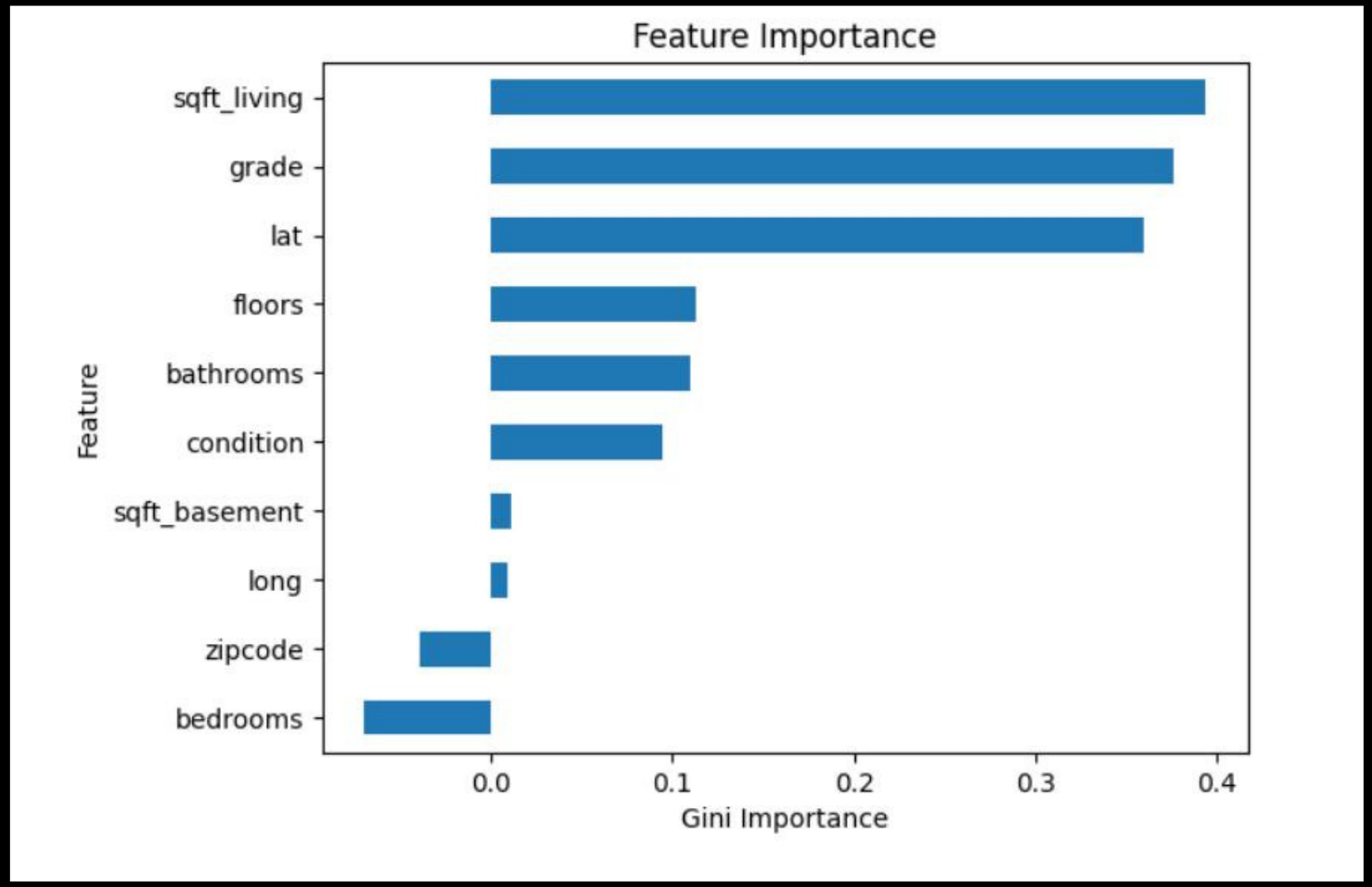


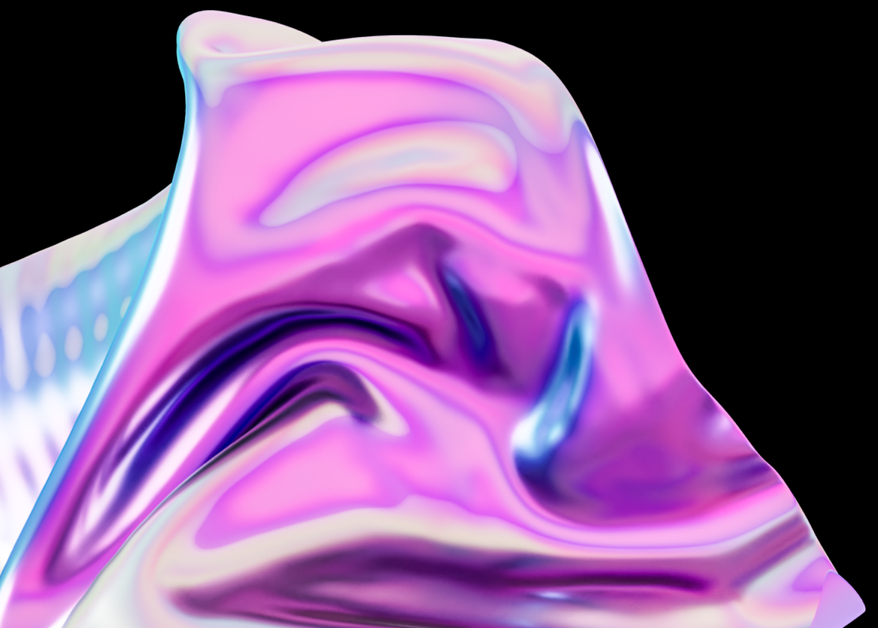
# Cross Validation
# cv=3

- first : 0.89670989
- second : 0.89310655
- third : 0.90289082



# Feature importance

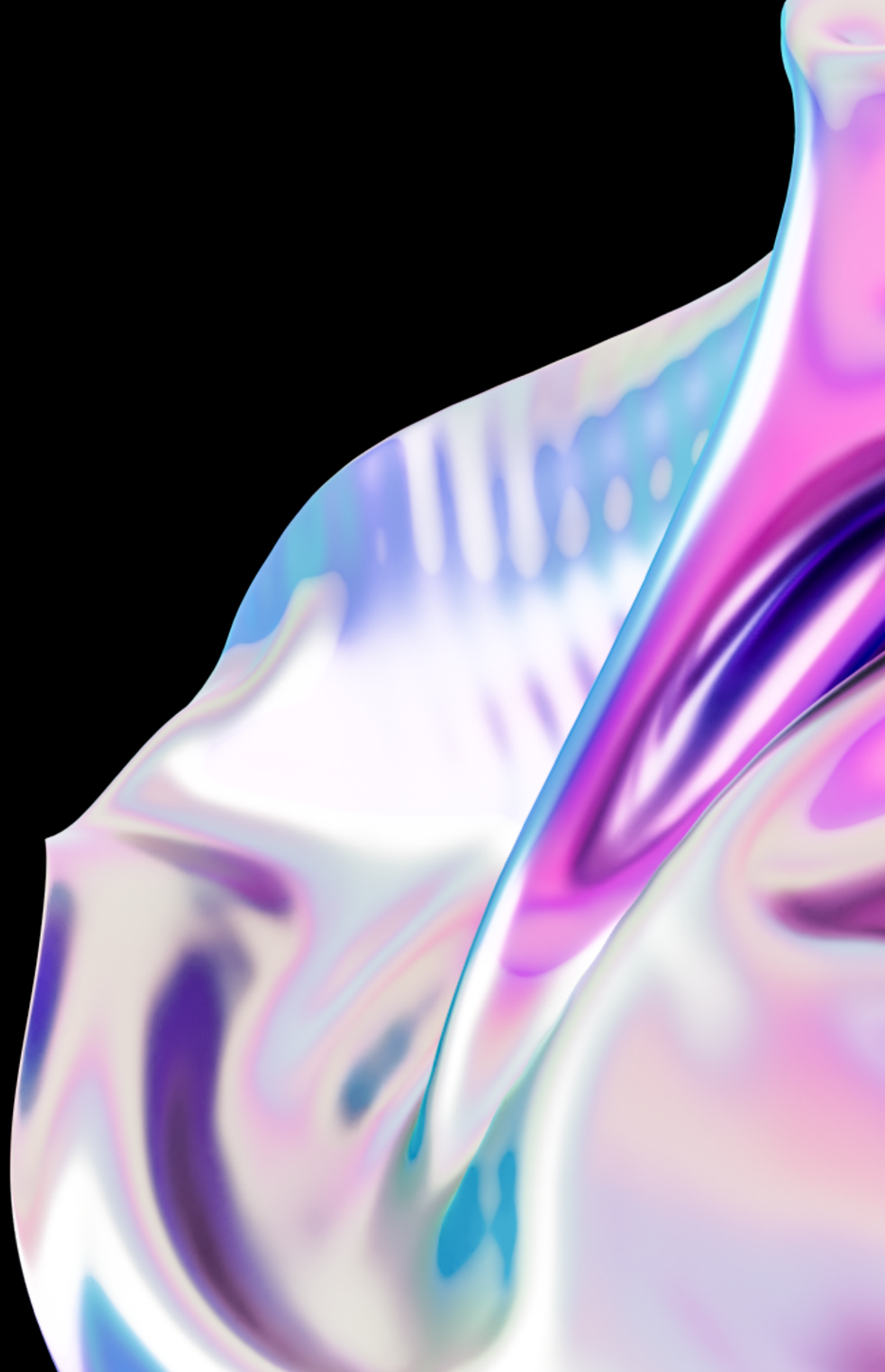




# 2 PlantVillage Dataset

# Introduction

- IN THIS SECTION, WE PRESENT A MACHINE LEARNING MODEL THAT IS DEVELOPED TO DETERMINE IF THIS LEAF  IS HEALTHY OR NOT

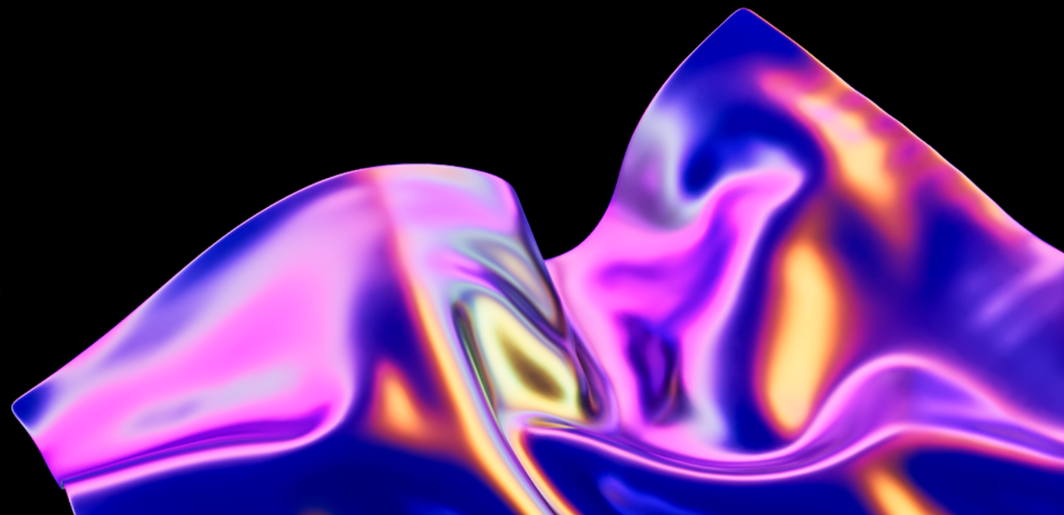


# Image DataSet

PlantVillage Dataset

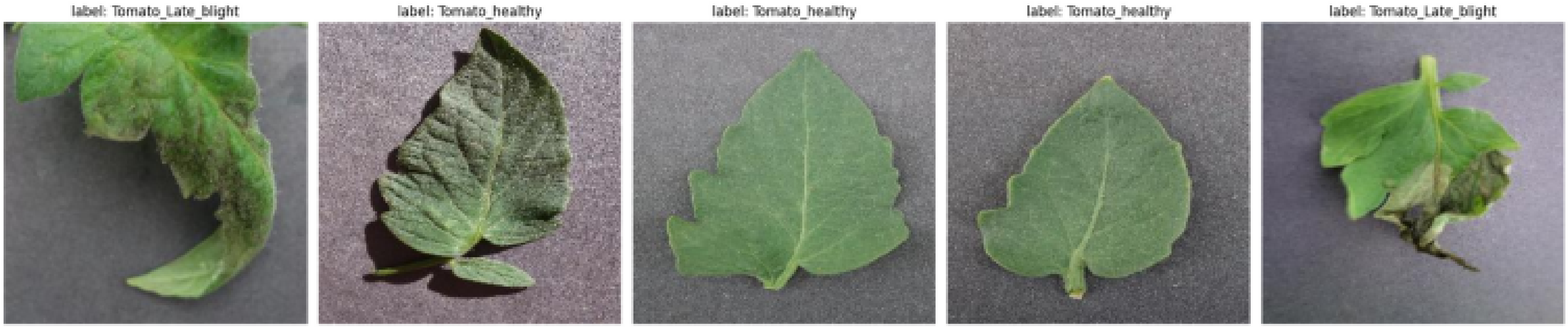
# Data Describtion

- Dataset of diseased plant leaf images and corresponding labels
- Data size : 690 MB
- Classes : 15 class
- image size : Width: 256, Height: 256'



# Feature Selection

- 'PEPPER__BELL___BACTERIAL_SPOT'
- 'PEPPER__BELL___HEALTHY'
- 'POTATO___LATE_BLIGHT'
- 'TOMATO_HEALTHY'
- 'TOMATO_LATE_BLIGHT'



# HOGDescriptor

- Cvtcoler to convert image coler from BGR TO GRAY

- use HOGDescriptor This descriptor is designed to capture the distribution of gradient orientations in an image. It divides the image into small regions called cells and computes histograms of gradient orientations within each cell.

# HOGDescriptor

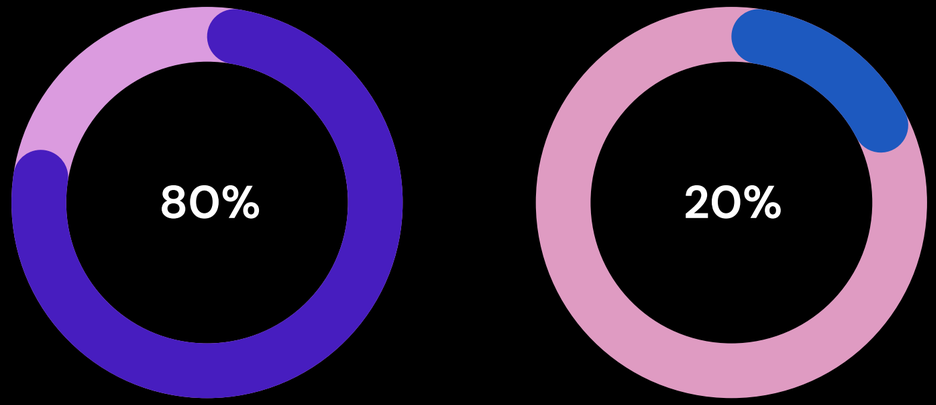
- we us The compute method of the HOG descriptor takes the grayscale image as input and returns a 1D array of HOG features.

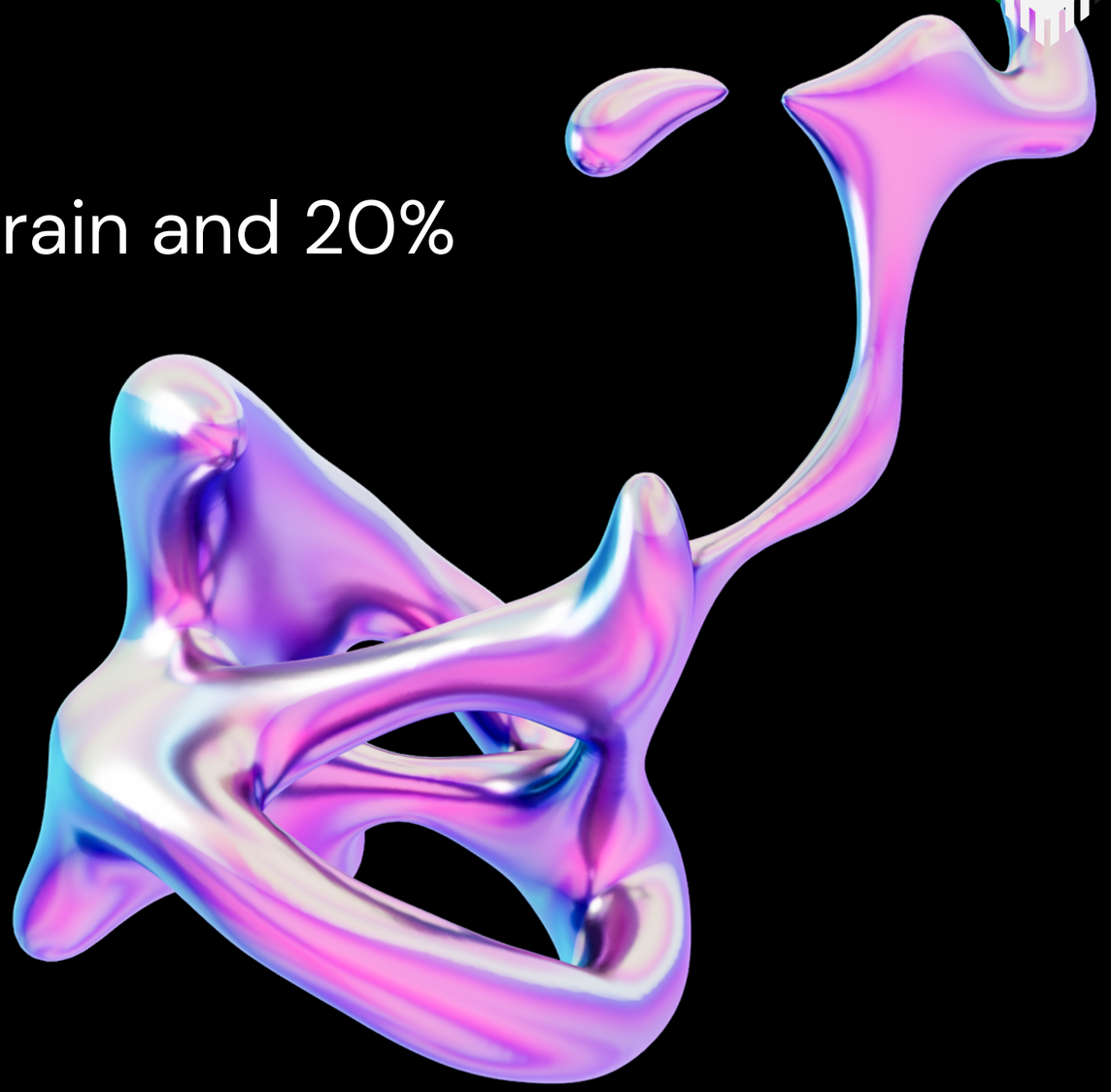
- We The flatten method is used to convert the multi-dimensional array of HOG features into a 1D array

- We shuffled the data and make it as batches batch size is 100
- We convert the image to numpy array
- We using label encoder to transform the target (object column ) to numerical

# Split Data

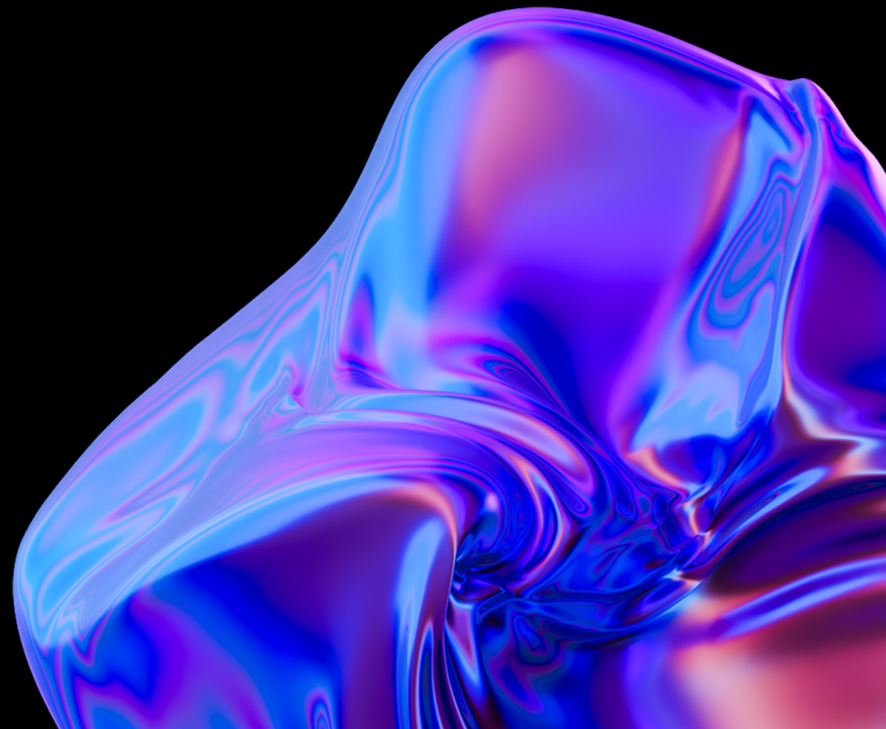
- we split data using  80% for train and 20% for test



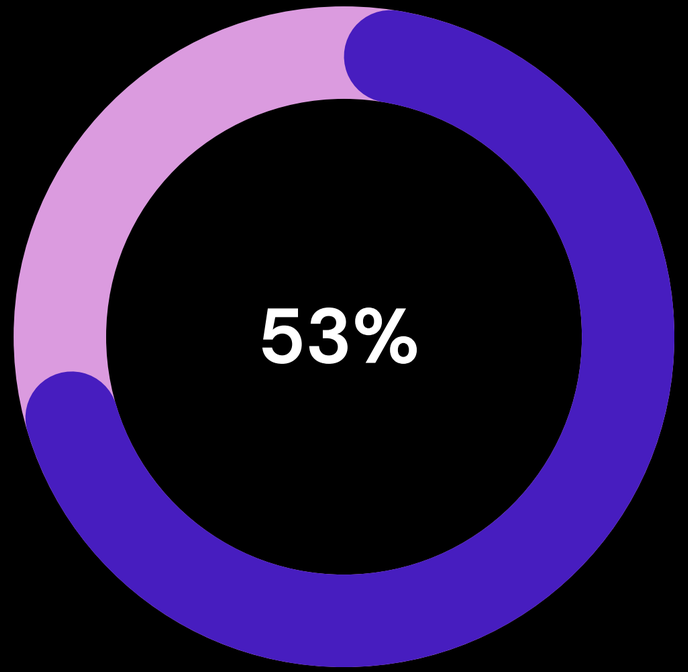
# PIPELINE

- Scale the data using stander scaler
- Add models
  (logistic regression, k-maens)
- Pca to

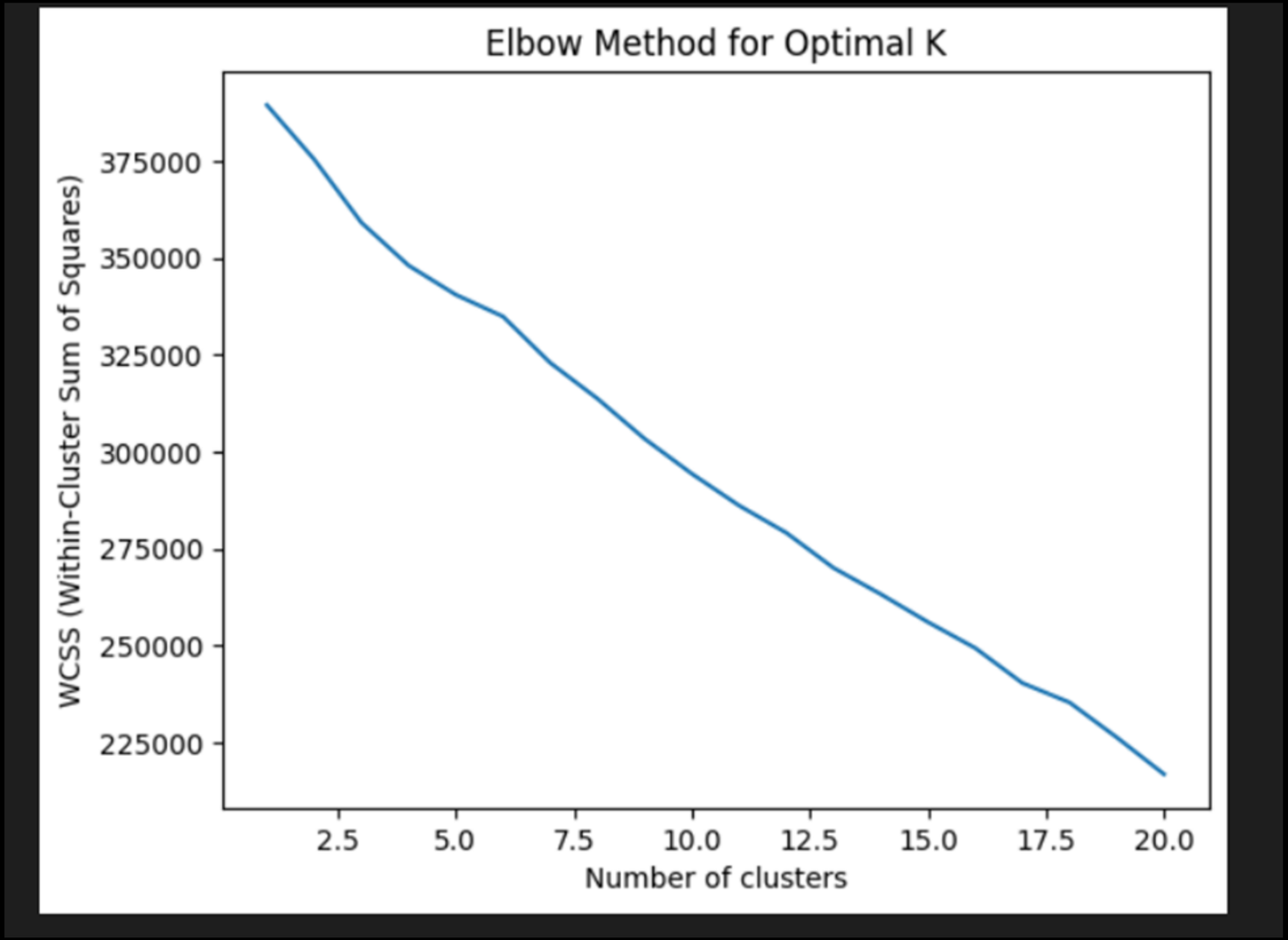


# Logistic Regression

- Accuracy : 53%

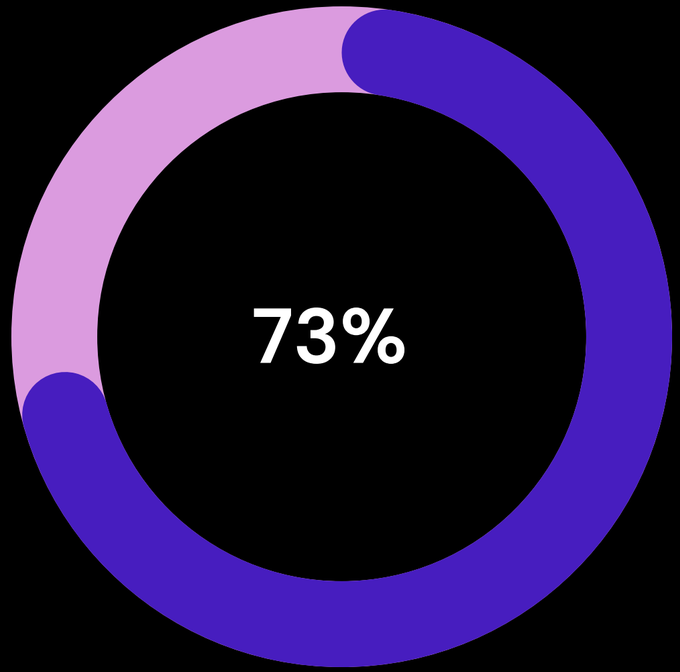


# Elbow method



# K-Means Clustering

- kmeans model with hyperparameter n_clusters=50, n_init=10
- Kmeans accuracy : 73%



# Our team

- abdelhalim ashraf abdelhalim
- eid osama eid
- ramy ibrahim ahmed
- mohey el din maher
- ali adel sayed
- amer medhat ahmed

# Thanks