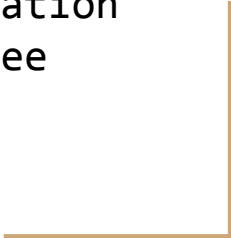




# U:Pass Week 4

36106: Machine Learning  
Algorithms and Application  
By Marisara Satrulee



# Feature Engineering for ML

- Imputation
- Discretization
- Categorical Encoding
- Feature Splitting
- Handling Outliers
- Variable Transformations
- Scaling
- Feature Creation in ML

# Instructions

Go to

**[www.menti.com](https://www.menti.com)**

Enter the code

**8796 0143**



Or use QR code

# Week 4 - Code snippets



[https://github.com/merrymira/UPASS\\_ML\\_WEEK4](https://github.com/merrymira/UPASS_ML_WEEK4)

# Improve MSE (adding zip code)

Explored the regression problem in house prices' prediction, using the USA\_Housing dataset.

```
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Avg. Area Income                      5000 non-null   float64
1   Avg. Area House Age                   5000 non-null   float64
2   Avg. Area Number of Rooms             5000 non-null   float64
3   Avg. Area Number of Bedrooms          5000 non-null   float64
4   Area Population                       5000 non-null   float64
5   Price                                 5000 non-null   float64
6   Address                               5000 non-null   object
dtypes: float64(6), object(1)
```

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Zip Code
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1715
1	79248.642455	6.002900	6.730821	3.09	40173.072174	2321
2	61287.067179	5.865890	8.512727	5.13	36882.159400	292
3	63345.240046	7.188236	5.586729	3.26	34310.242831	2125
4	59982.197226	5.040555	7.839388	4.23	26354.109472	441

# Comparing the performance

RangeIndex: 5000 entries, 0 to 4999

Data columns (total 5 columns):

#	Column	Non-Null Count	Dtype
0	Avg. Area Income	5000 non-null	float64
1	Avg. Area House Age	5000 non-null	float64
2	Avg. Area Number of Rooms	5000 non-null	float64
3	Avg. Area Number of Bedrooms	5000 non-null	float64
4	Area Population	5000 non-null	float64

dtypes: float64(5)

RangeIndex: 5000 entries, 0 to 4999

Data columns (total 6 columns):

#	Column	Non-Null Count	Dtype
0	Avg. Area Income	5000 non-null	float64
1	Avg. Area House Age	5000 non-null	float64
2	Avg. Area Number of Rooms	5000 non-null	float64
3	Avg. Area Number of Bedrooms	5000 non-null	float64
4	Area Population	5000 non-null	float64
5	Zip Code	5000 non-null	int64

dtypes: float64(5), int64(1)

RMSE: 1010<sup>92</sup>.70158252279

MAE: 813<sup>59</sup>.27022837057

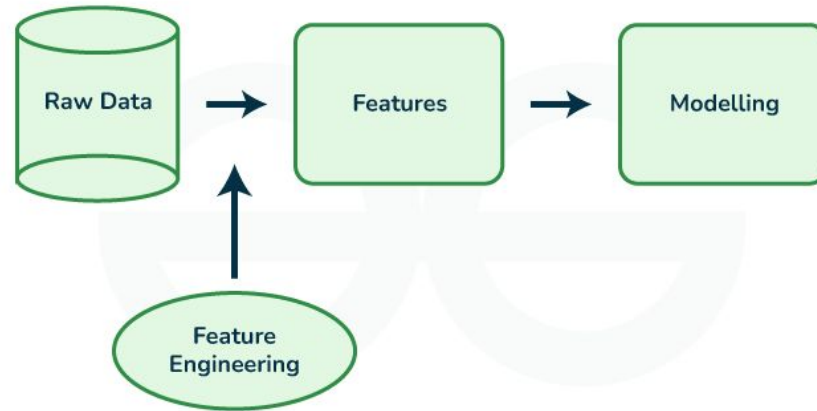
RMSE: 1010<sup>73</sup>.0030082884

MAE: 813<sup>56</sup>.94721294698

The performance is slightly **improved**.

# What is Feature Engineering?

Feature engineering is the process of **transforming raw data into features** that are suitable for machine learning models.



# Feature Engineering for ML

- Imputation
- Discretization
- Categorical Encoding
- Feature Splitting
- Handling Outliers
- Variable Transformations
- Scaling
- Feature Creation in ML



# Type of Features in ML

123

Numerical Data



Categorical Data



Time Series Data

[text]

Text Data

# Imputation

Imputation is for handling missing values in data.

123

Numerical Data

Missing numerical variables are generally replaced by  
**Mean or Median.**



Categorical Data

Missing categorical variables are generally replaced by  
**Mode.**

# Discretization

Discretization involves grouping data values into bins.


Artist Name	Artist Reputation	Height	Width	Weight	Material	Price Of Sculpture	Base Shipping Price	International	Express Shipment	Installation Included
Billy Jenkins	0.26	17.0	6.000000	4128.000000	Brass	13.91	16.27	Yes	Yes	No
Jean Bryant	0.28	3.0	3.000000	61.000000	Brass	6.83	15.00	No	No	No
Laura Miller	0.07	8.0	5.000000	237.000000	Clay	4.96	21.18	No	No	No
Robert Chaires	0.12	9.0	9.617647	400694.821918	Aluminium	5.81	16.31	No	No	No
Rosalyn Krol	0.15	17.0	6.000000	324.000000	Aluminium	3.18	11.94	Yes	Yes	Yes

Metal or Nonmetal

# Categorical Encoding

Beware of One-Hot Encoding(OHE)'s disadvantage :

OHE could dramatically increase the number of features and result in highly correlated features.



Color	green	red	yellow
red	0	1	0
red	0	1	0
yellow	0	0	1
green	1	0	0
yellow	0	0	1

```
from sklearn.preprocessing import OneHotEncoder

color = [["red"], ["red"], ["yellow"], ["green"], ["yellow"]]

ohe = OneHotEncoder()

ohe.fit_transform(color).toarray()

array([[0., 1., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
       [1., 0., 0.],
       [0., 0., 1.]])
```

# Feature Splitting

Transport	Fragile	Customer Information	Remote Location	Scheduled Date	Delivery Date	Customer Location	Cost	Material_cat
Airways	No	Working Class	No	06/07/15	06/03/15	New Michelle, OH 50777	5.649995	Non-Metal
Roadways	No	Working Class	No	03/06/17	03/05/17	New Michaelport, WY 12072	5.081156	Non-Metal
Roadways	Yes	Working Class	Yes	03/09/15	03/08/15	Bowmanshire, WA 19241	5.045294	Non-Metal
Roadways	No	Wealthy	Yes	05/24/15	05/20/15	East Robyn, KY 86375	5.088584	Non-Metal
Airways	No	Working Class	No	12/18/16	12/14/16	Aprilside, PA 52793	5.076610	Non-Metal

Split to 3 new columns of  
Year - Month - Day

# Handling Outliers

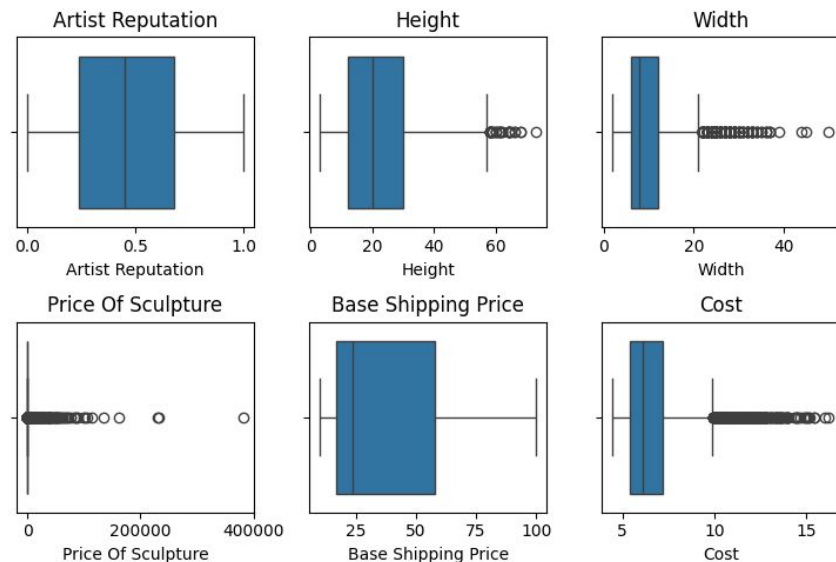
Outliers are unusually high or low values in the dataset. To address this, the methods includes Remove, Replace, or Capping the values. However, it's important to keep in mind that this approach could result in a data loss. Recommending to remove **less than 10%** from the total rows.

By removing or capping outliers, you need to calculate lower and upper bounds as follow:

```
# Remove outliers
Q1 = train_rm_out['Height'].quantile(0.25)
Q3 = train_rm_out['Height'].quantile(0.75)

IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
```



# Handling Outliers

The various methods of handling outliers include:

- **Removal:** Removing the values that are lower than the lower bound or higher than the higher bound

```
train_rm_out = train_rm_out[(train_rm_out['Height'] >= lower_bound) & (org_train['Height'] <= upper_bound)]
```

- **Replacing values:** Treating outliers as missing values and replaced by using appropriate imputation.

```
train_rp_out['Height'] = np.where((train_rp_out['Height'] < lower_bound) | (train_rp_out['Height'] > upper_bound), train_rp_out['Height'].median(), train_rp_out['Height'])
```

- **Capping:** Capping the maximum and minimum values and replacing them with an arbitrary value.

```
train_cp_out['Height'] = train_cp_out['Height'].clip(lower_bound, upper_bound)
```

# Handling Outliers

**Comparing** the shape of the dataset after performing Remove, Replace, Capping methods on outliers

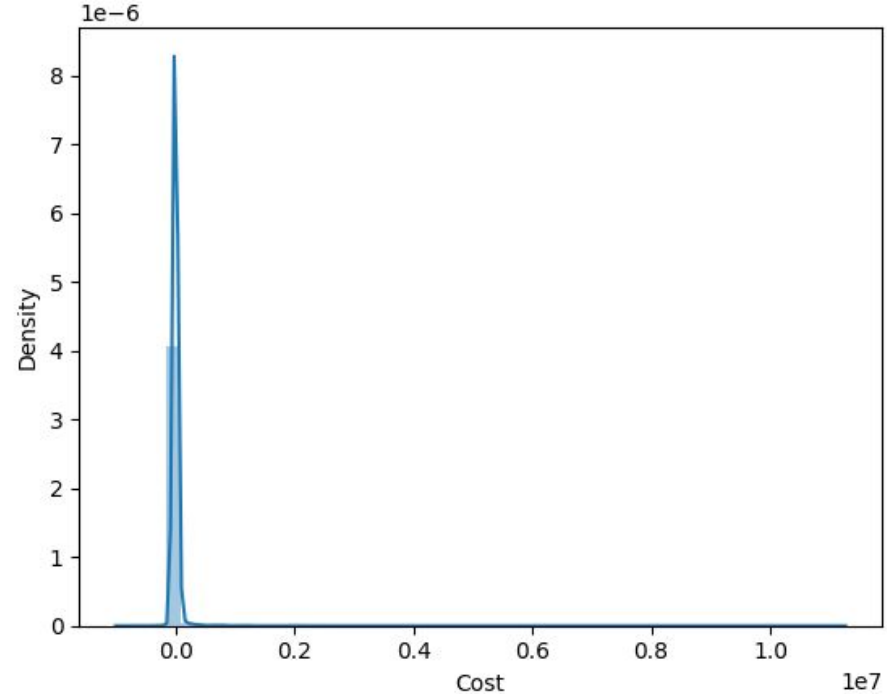
```
train_rm_out.shape, train_rp_out.shape, train_cp_out.shape  
  
((6095, 20), (6500, 20), (6500, 20))
```



# Variable Transformations

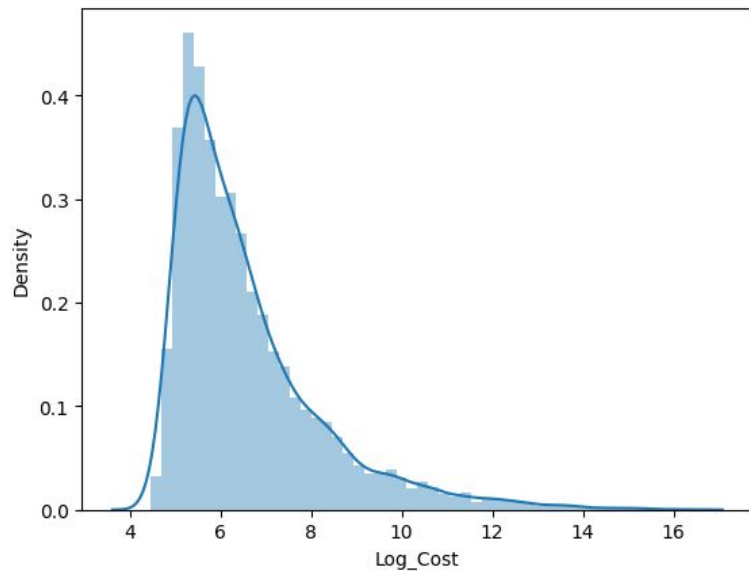
Variable transformation techniques help with normalizing skewed data.

1. Quantile Transformer Scaler
2. Log Transformation *\*Most popular*
3. Power Transformer Scaler

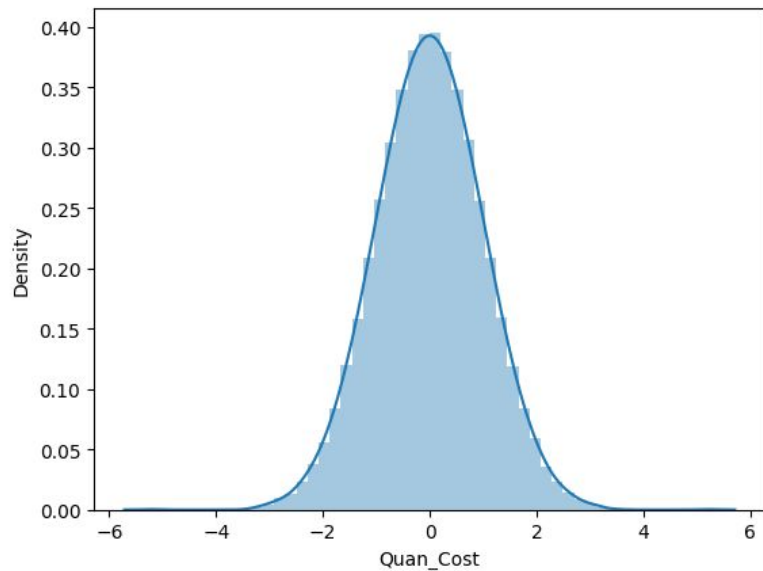


# Variable Transformations

**Applying** Log Transformation



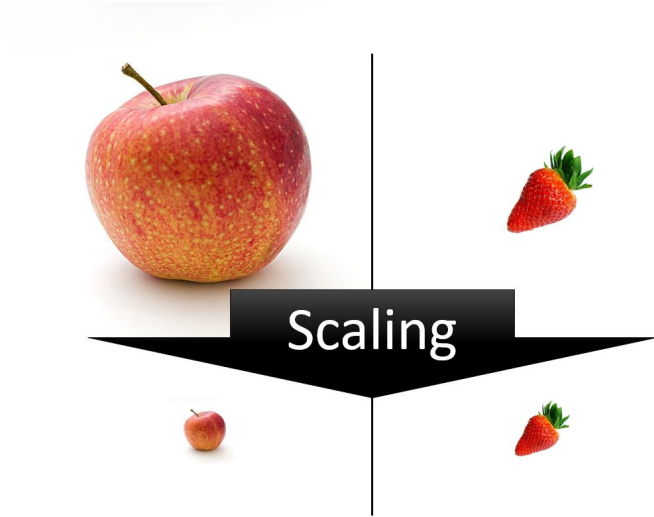
**Applying** Quantile Transformer Scaler



# Scaling

Feature scaling is sometimes referred to as feature normalization.

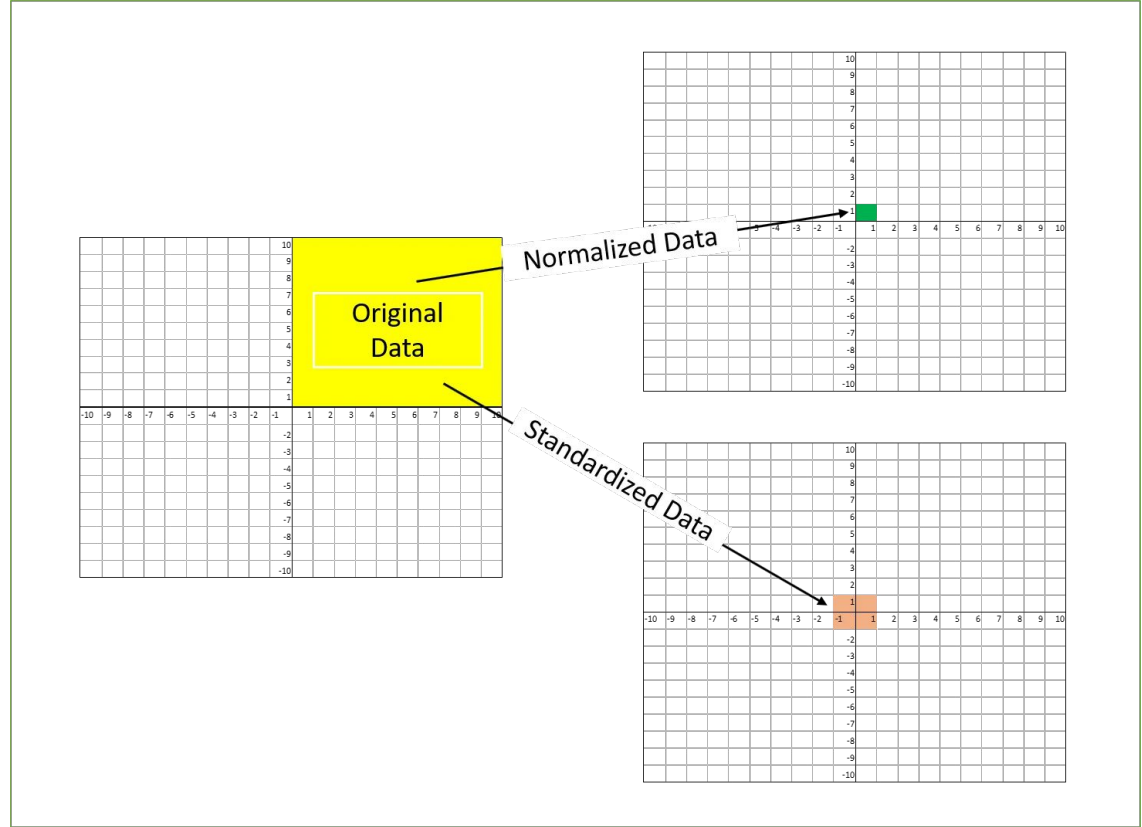
1. MinMax Scaler
2. Standard Scaler *\*Most popular*
3. MaxAbsScaler
4. Robust Scaler
5. Unit Vector Scaler/ Normalizer



# Scaling

Normalization is used when we want to bound our values between two numbers, typically, between  $[0,1]$  or  $[-1,1]$ .

While Standardization transforms the data to have zero mean and a variance of 1, they make our data **unitless**.



# Feature Creation in ML

Feature Creation means creating new features from the existing one.

## Question

What new features do you think we can create, considering the table on the right?

Artist Name	Artist Reputation	Height	Width	Weight	Price Of Sculpture	Base Shipping Price
Billy Jenkins	0.26	17.0	6.000000	4128.000000	13.91	16.27
Jean Bryant	0.28	3.0	3.000000	61.000000	6.83	15.00
Laura Miller	0.07	8.0	5.000000	237.000000	4.96	21.18
Robert Chaires	0.12	9.0	9.617647	400694.821918	5.81	16.31
Rosalyn Krol	0.15	17.0	6.000000	324.000000	3.18	11.94

# Comparing the Performance

Fitting Linear Regression  
with **Original** Training  
Dataset


RMSE: 24.950222678542406

MAE: 20.348215708227503

Fitting Linear Regression  
with **Added Features**  
Training Dataset

RMSE: 25.043947678604418

MAE: 20.424194793991337



The performance is performing **worse!!!**  
What do you think happen in this case?

# When went wrong? Fixing time!!

Data columns (total 19 columns):

#	Column	Non-Null Count	Dtype
0	Artist Name	3362 non-null	object
1	Artist Reputation	3362 non-null	float64
2	Height	3362 non-null	float64
3	Width	3362 non-null	float64
4	Weight	3362 non-null	float64
5	Material	3362 non-null	object
6	Price Of Sculpture	3362 non-null	float64
7	Base Shipping Price	3362 non-null	float64
8	International	3362 non-null	object
9	Express Shipment	3362 non-null	object
10	Installation Included	3362 non-null	object
11	Transport	3362 non-null	object
12	Fragile	3362 non-null	object
13	Customer Information	3362 non-null	object
14	Remote Location	3362 non-null	object
15	Scheduled Date	3362 non-null	object
16	Delivery Date	3362 non-null	object
17	Customer Location	3362 non-null	object
18	Cost	3362 non-null	float64

dtypes: float64(7), object(12)

Data columns (total 31 columns):

#	Column	Non-Null Count	Dtype
0	Artist Name	6500 non-null	object
1	Artist Reputation	6500 non-null	float64
2	Height	6500 non-null	float64
3	Width	6500 non-null	float64
4	Weight	6500 non-null	float64
5	Price Of Sculpture	6500 non-null	float64
6	Base Shipping Price	6500 non-null	float64
7	International	6500 non-null	object
8	Express Shipment	6500 non-null	object
9	Installation Included	6500 non-null	object
10	Transport	6500 non-null	object
11	Fragile	6500 non-null	object
12	Customer Information	6500 non-null	object
13	Remote Location	6500 non-null	object
14	Scheduled Date	6500 non-null	datetime64[ns]
15	Delivery Date	6500 non-null	datetime64[ns]
16	Customer Location	6500 non-null	object
17	Cost	6500 non-null	float64
18	Material_cat	6500 non-null	object
19	Material__Aluminium	6500 non-null	uint8
20	Material__Brass	6500 non-null	uint8
21	Material__Bronze	6500 non-null	uint8
22	Material__Clay	6500 non-null	uint8
23	Material__Marble	6500 non-null	uint8
24	Material__Stone	6500 non-null	uint8
25	Material__Wood	6500 non-null	uint8
26	Scheduled_Year	6500 non-null	int64
27	Scheduled_Month	6500 non-null	int64
28	Scheduled_Day	6500 non-null	int64
29	Log_Cost	6500 non-null	float64
30	Quan_Cost	6500 non-null	float64

dtypes: datetime64[ns](2), float64(9), int64(3), object(10), uint8(7)