Module-2
Data Normalization

→ This involves scaling the data to a common range such as between 0 and 1 or -1 and 1. Normalization is often used to handle data with different units and scales.

→ Common normalization techniques include min-max normalization, z-score normalization and decimal scaling.

→ There are mainly four techniques to do Data Normalization or Scaling

1. Min - max Normalization
2. Z-score normalization using mean and standard deviation
3. Z-score using mean and mean absolute deviation.
4. Normalization by decimal scaling

1) Min-max Normalization

$$V = \frac{x - min}{max - min}$$

min = 200 & max = 1000

Data (v)
200
300
400
600
1000

$$V = \frac{200 - 200}{1000 - 200} = 0$$

$$V = \frac{300 - 200}{1000 - 200} = 0.125$$

$$V = \frac{400 - 200}{1000 - 200} = 0.25$$

$$V = \frac{600 - 200}{1000 - 200} = 0.5$$

$$V = \frac{1000 - 200}{1000 - 200} = 1$$

2) Z-score normalization.

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

μ = mean

σ = standard deviation

$$\text{Mean} = \frac{(200 + 300 + 400 + 600 + 1000)}{5} = 500$$

Standard deviation

$$= \sqrt{\frac{\sum (x_i - \mu)^2}{n}}$$

$$= \sqrt{\frac{(200-500)^2 + (300-500)^2 + (400-500)^2 + (600-500)^2 + (1000-500)^2}{5}}$$

$$= 282.8$$

$$V = \frac{(200 - 500)}{282.8} = -1.06$$

$$V = \frac{(300 - 500)}{282.8} = -0.707$$

$$V = \frac{400 - 500}{282.8} = -0.354$$

$$V = \frac{600 - 500}{282.8} = 0.354$$

$$V = \frac{1000 - 500}{282.8} = 1.77$$

Normalized Data (V):

-1.06
-0.707
-0.354
0.354
1.77

Z-score Normalization - Mean Absolute Deviation

$Z = \frac{x - \mu}{A}$. μ = Mean

A = Mean absolute deviations

$$\text{Mean} = \frac{(200 + 300 + 400 + 600 + 1000)}{5} = 500$$

Mean absolute Deviation

$$A = \frac{|200-500| + |300-500| + \ldots + |1000-500|}{5}$$

$$= 240$$

$V = \frac{200 - 500}{240} = -1.25$. μ = Mean = 500

$V = \frac{300 - 500}{240} = -0.833$. A = Mean absolute deviation = 240.

$V = \frac{400 - 500}{240} = -0.417$.

$V = \frac{600 - 500}{240} = 0.417$.

$V = \frac{1000 - 500}{240} = 2.08$.

Normalization By Decimal Scaling

- Find value of j
- The smallest integer j such that

$$\text{Max}\left(\frac{v_i}{10^j}\right) \leq 1$$

- $\frac{200}{10^3} \leq 1 = 0.2$
- $\frac{300}{10^3} = 0.3$
- $\frac{400}{10^3} = 0.4$
- $\frac{600}{10^3} = 0.6$
- $\frac{1000}{10^3} = 1$

$\frac{200}{10^3} < 1$

$\frac{200}{1000} = 0.2$

Binning.

→ Data Binning, also called discrete binning or Bucketing is a data pre-processing technique used to reduce the effects of minor observation errors. It is a form of quantization.

→ The original data values are divided into small intervals known as bins, and then they are replaced by a general value calculated for that bin.

→ This has a smoothing effect on the input data and may also reduce the chances of overfitting in the case of small datasets.

There are 2 methods of dividing data into bins :-

(i) Equal frequency Binning

Bins have an equal frequency.

(ii) Equal width Binning.

Bins have equal width with a range of each bin are defined as [min + w], [min + 2w] - - - [min + nw] where w = (max - min) / (no. of bins).

Equal Frequency.

Input: [5, 10, 11, 13, 15, 35, 50, 55, 72, 92, 204, 215]

Output:

[5, 10, 11, 13]

[15, 35, 50, 55]

[72, 92, 204, 215]

Equal Width:

Input: [5, 10, 11, 13, 15, 35, 50, 55, 72, 92, 204, 215]

Output:

[5, 10, 11, 13, 15, 35, 50, 55, 72]

[92]

[204, 215].

Outliers

→ Outliers are the deviated values or datapoints that are observed too away from other data points in such away that they badly affect the performance of the model.

→ Outliers can be handled with the feature engineering technique

→ ~~Steps~~ Interquartile range method to find outliers

Steps

1. Sort your data from low to high.
2. Identify the first quartile (Q1), the median, and the third quartile (Q3).
3. Calculate your IQR = Q3 − Q1
4. Calculate your upper fence = Q3 + (1.5 * IQR)
5. Calculate your lower fence = Q1 − (1.5 * IQR)
6. Use your fences to highlight any outliers, all values that fall outside your fences. The outliers are any values greater than your upper fence or less than your lower fence

Example:

Given dataset has 11 values. They have couple of extreme values in your dataset. Use the IQR method to check whether they are outliers.

26	37	24	28	35	22	31	53	41	64	29

Step1 :- Sort your data from low to high.

Sort your data in ascending order.

22	24	26	28	29	31	35	37	41	53	64

Step 2 :- Identify the median, the first quartile (Q1), and the third quartile (Q3).

The median is the 6th value. The median value is 31.

The Q1 is the value in the middle of the first half of your dataset, excluding the median.

The first quartile value is 25.

22 24 26 28 29

The Q3 value is in the middle of the second half of your dataset, excluding the median. The third quartile value is 41.

35 37 41 53 64

Step 3:- Calculate your IQR.

The IQR is the range of the middle half of your dataset. Subtract Q1 from Q3 to calculate the IQR.

Formula

IQR = Q3 − Q1

= 41 − 26 = 15

Step 4:- Calculate your upper fence

The upper fence is the boundary around the third quartile. It tells you that any values exceeding the upper fence are outliers

Upper fence = Q3 + (1.5 * IQR)

Upper fence = 41 + (1.5 * 15) = 41 + 22.5

= 63.5

Step 5 :- Calculate your lower fence.

The lower fence is the boundary around the first quartile. Any values less than the lower fence are outliers.

lower fence = Q1 − (1.5 * IQR)

lower fence = 26 − (1.5 * 15)

= 26 − 22.5 = 3.5

Step 6 :- Use your fences to highlight any outliers

~~Go back~~ Highlight any values that are greater than the upper fence or less than your lower fence. These are your outliers.

- upper fence = 63.5
- lower fence = 3.5

22 24 26 28 29 31 35 37 41 53 64

64 in the dataset is the one of the outliers

Different outlier detection techniques

1. Visual inspection :-

(i) Box plots :- Box plots visually represent the distribution of data and highlight any points beyond the whiskers as potential outliers.

(ii) Scatter plots :- Visualizing data in scatter plots can help identify points that are far away from the general pattern.

2. Statistical Methods :-

(i) Z-score :- Calculate the Z-score for each data point. Z-score measures how many standard deviations a datapoint is from the mean. points with a high absolute z-score may be considered outliers. $\left(Z = \frac{x - \mu}{\sigma}\right)$.

(ii) IQR (Interquartile Range)
Calculate the interquartile range (IQR) and consider data points outside the range of 1.5 times the IQR as potential outliers. IQR = Q3 - Q1

3. Density - Based Methods:

- DBSCAN (Density - Based Spatial Clustering of applications with Noise) :- It groups together data points that are close to each other based on a density threshold, considering points with lower density as potential outliers

- LOF (Local outlier factor) :- LOF measures the local density of data points and identifies those with significantly lower density as potential outliers

4. M/c learning models

- Isolation Forest

  It isolates outliers by randomly selecting a feature and then splitting the data points until the outliers are isolated.

- One-class SVM (Support vector Machine)

  Trains on normal datapoints and identifies deviations as potential outliers.

5) Modified Z-score:

MAD (Median Absolute deviation)

Calculate the modified z-score using the median and the MAD, which is less sensitive to extreme values than the standard deviation

Formula MAD = median ($|X_i - \text{median}(X)|$)

Example:

- Dataset : [10, 12, 11, 15, 13, 100]
- Median : 13
- MAD : 1.5
- Modified Z-scores :- [-0.67, -0.33, -0.5, 0, -0.17, 45.33]
- Threshold (eg:- Modified Z>3): Identify 100 as an outlier.

$$\frac{11+15}{2}$$

$$\frac{|10-13|+|12-13|+|11-13|+|15-13|+0+|100-13|}{6}$$

6) Visualization Techniques:

- Histograms:- Visualize the distribution of data to identify any unusual spikes or gaps.
- Q-Q plots:- Compare the quantiles of the data to a theoretical distribution to identify deviations.

7) Distance - Based Methods:-

(i) Mahalanobis distance:-

This accounts for correlations b/w variables and calculates the distance of each point from the centroid. Points with high mahalanobis distances may be outliers.

$$D_M = \sqrt{(x-\mu)^T \Sigma^{-1} (x-\mu)}$$

(ii) K-Nearest neighbors (KNN):-

Identify outliers based on the distance to their k-Nearest neighbours

Q:- Consider the following dataset of exam scores and set the threshold $z = 2$ to identify outliers

Dataset : [72, 85, 88, 95, 68, 90, 78, 110]

Step 1:- Calculate Mean & standard deviation

$$\mu = \frac{72+85+88+95+68+90+78+110}{8} = 86.$$

$$\sigma = \sqrt{\frac{\sum (x_i - \mu)^2}{N}}$$

$$\sigma = \sqrt{\frac{(72-86)^2 + (85-86)^2 + \cdots + (110-86)^2}{8}}$$

$$\sigma = \sqrt{\frac{1740}{8}} \approx 13.19.$$

$z = 2$

Step 2:

Calculate Z-score for each data point

$$Z = \frac{X - \mu}{\sigma}$$

$$Z = \left[\frac{72-86}{13.19}, \frac{85-86}{13.19}, \cdots, \frac{110-86}{13.19}\right]$$

$$Z = [-1.06, -0.08, 0.14, 0.69, -1.44, 0.36, -0.19, 2.22].$$

Step3 :- Identify outliers.
Consider a threshold of $z > 2$ or $z < -2$ to identify outliers.
outliers = [110]

Modified z-score

Consider the data with 16 values

29 46 26 26 22 24 19 16 16 15 44 12

8 7 7 6

step1 :- Sort the dataset
step2 :- Find the median = 16.
step3 :- Find the absolute difference b/w each value & the median

for ex :- Absolute diff = $|16 - 16| = 10$

Data	Absolute difference from median
6	10
7	9
7	9
8	8
12	4
14	2
15	1
16	0
16	0
19	3
22	6
24	8
26	10
26	10
29	13
46	30

0.6745 is a constant used to approximate a median-equivalent of standard

Step 4 :- Find median absolute deviation.

which is 7.68 = 8.

confidence interval = $\bar{x} \pm \left(Z \times \frac{\sigma}{\sqrt{n}}\right)$

Step 5 :- Modified z-score for each data value.

Modified z score = $0.6745(x_i - \tilde{x})/MAD$

Modified z score for first data value = $0.6745 * (6-16)/8 = -0.843$

Repeat the formula for every value in the dataset

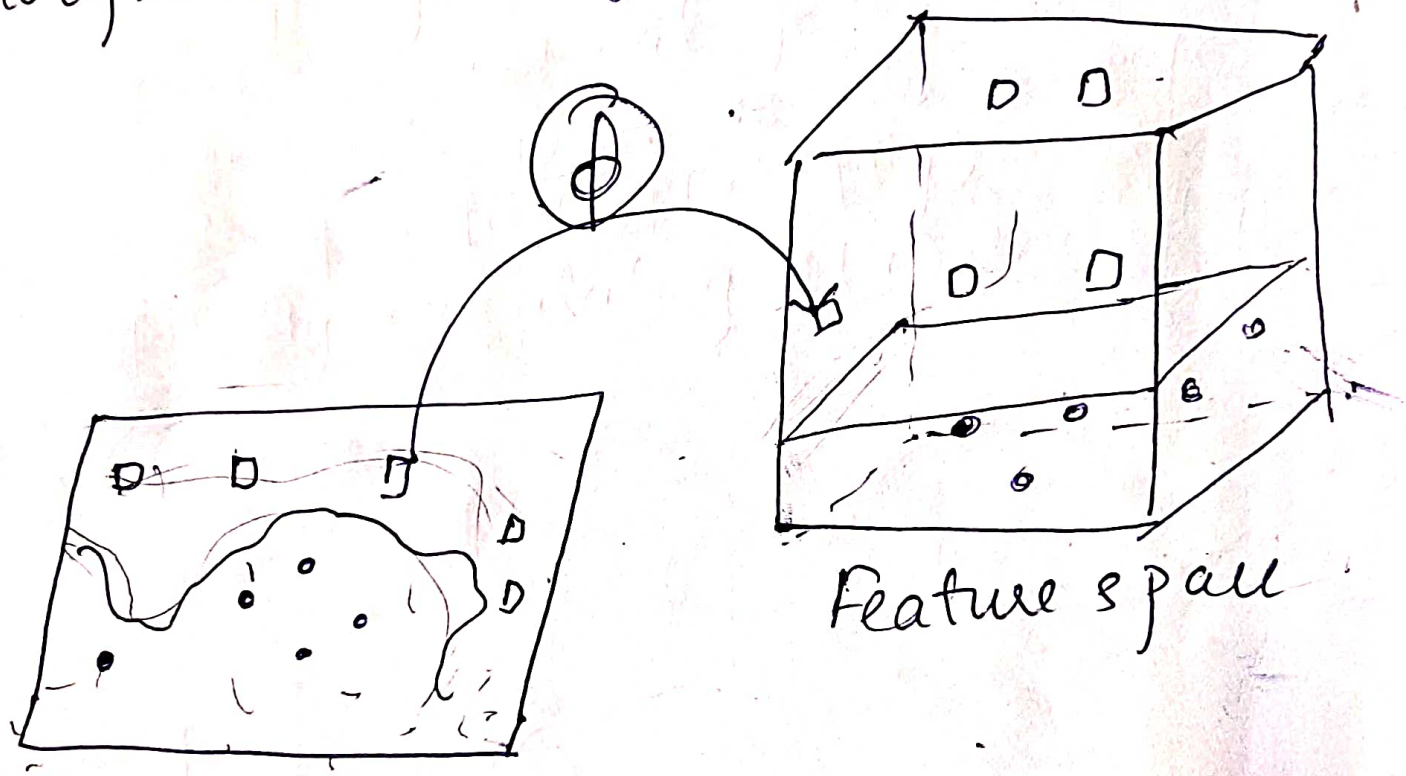
Modified z-score :

-0.843	0.506
-0.759	0.675
-0.759	0.843
-0.675	0.843
-0.337	1.096
-0.169	2.529
-0.084	
0.000	
0.000	
0.253	

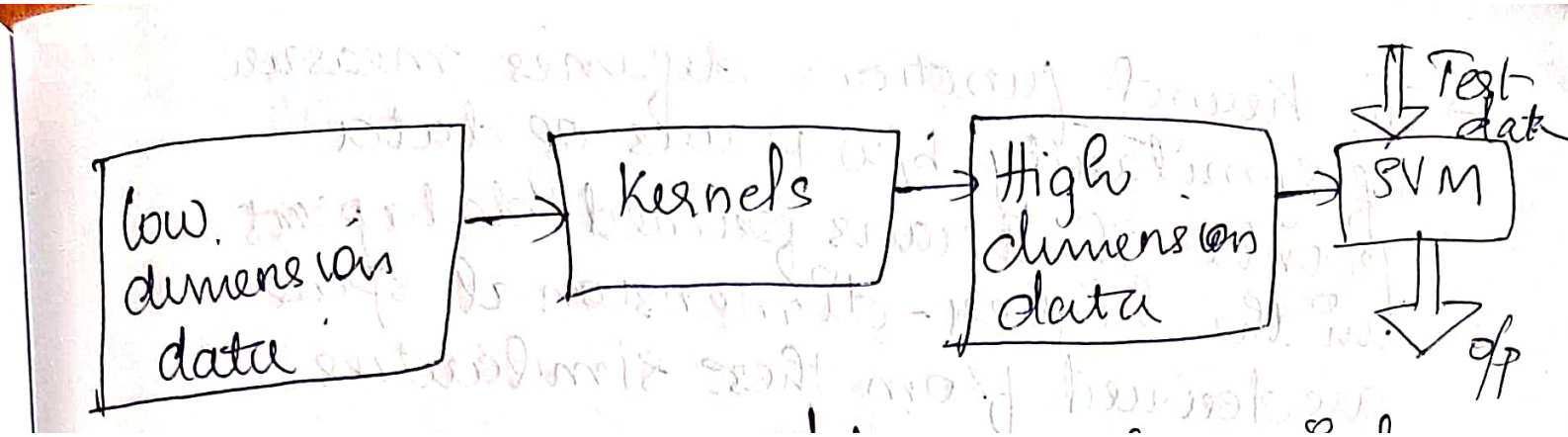
46 is the outlier

Kernel- Induced Feature expansion

Kernels are a set of fxns used to transform data from lower dimension to higher dimension and to manipulate data using dot product at higher dimensions.

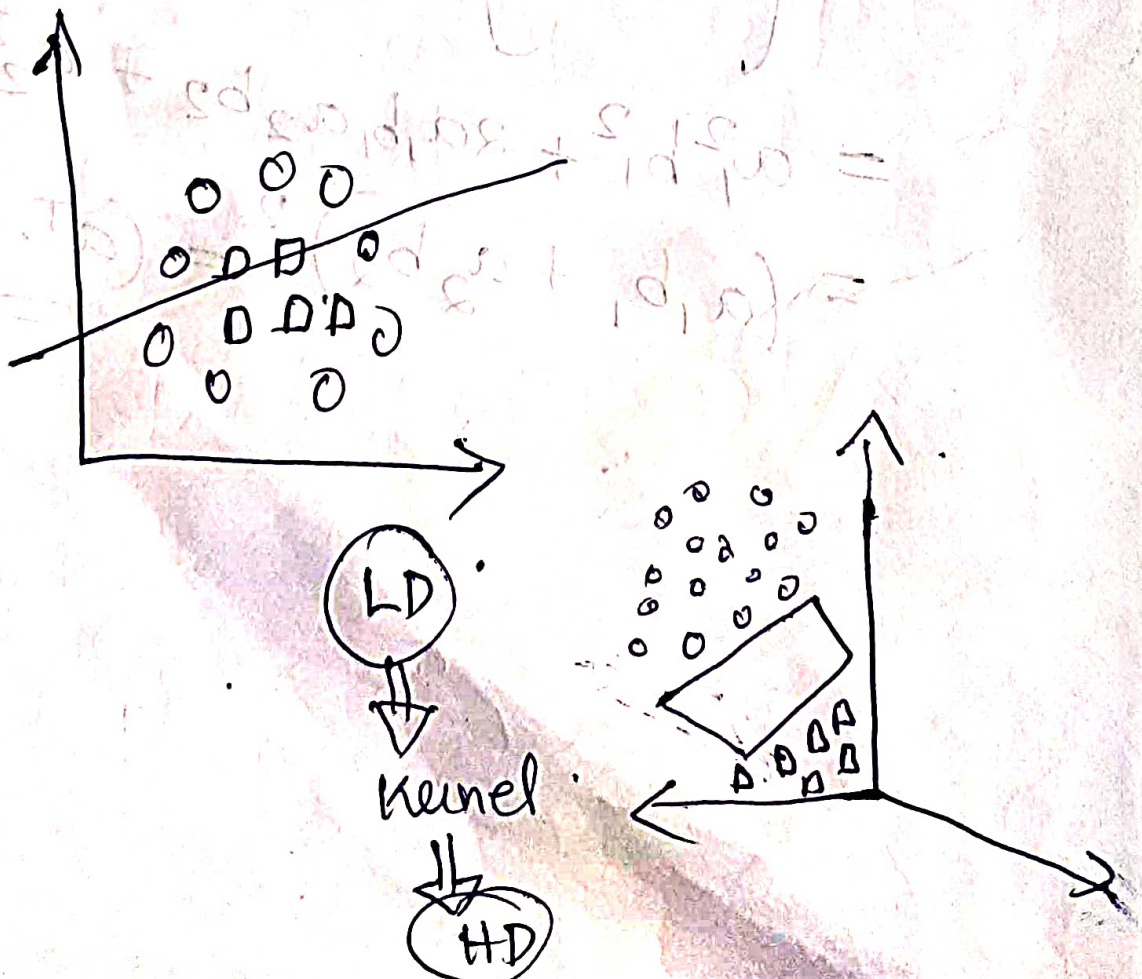


→ The use of kernels is to apply transformation to data and perform classification at the higher dimension as shown in figure.


low dimension data
kernels
High dimension data
SVM
Test data
op

→ The kernel function defines measure of similarity b/w pairs of data points. The transformed datapoints in the higher-dimensional space are derived from these similarities.

→ Mathematically, if $\phi(x)$ represents the mapping of datapoint x into the higher-dimensional space, the kernel function $K(x, y)$ is used to compute the dot product $\phi(x) \cdot \phi(y)$ without explicitly calculating $\phi(x)$ and $\phi(y)$.



Types of Kernels

Linear Kernel :-

Linear kernels are of the type

$$K(x, y) = x^T \cdot y$$

where x and y are two vectors.

Therefore $K(x, y) = \phi(x) \cdot \phi(y) = x^T \cdot y$

polynomial kernel :-

polynomial kernels are of type :-

$$K(x, y) = (x^T y)^q$$

This is called homogeneous kernel.

Here, q is the degree of the polynomial

If $q = 2$, then it is called quadratic kernel.

→ For ~~Gaussian~~ inhomogeneous kernels, this is given as :-

$$K(x, y) = (c + x^T \cdot y)^q$$

Here c is a constant & q is the degree of the polynomial.

- If c is zero & degree is one, then the polynomial kernel is reduced to a linear kernel.
- The value of degree q should be optimal as more degree may lead to overfitting.

a) Consider 2 data points $x = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$ and $y = (2, 3)$ with $c = 1$. Apply linear, homogeneous and inhomogeneous kernels

Soln:

Kernel is given by $K(x, y) = (x^T y)^q$

If $q = 1$, then it is called linear kernel

$$K(x, y) = \left(\begin{pmatrix} 1 \\ 2 \end{pmatrix}^T (2, 3)\right)^1 = 8$$

If $q = 2$

$$K(x, y) = \left(\begin{pmatrix} 1 \\ 2 \end{pmatrix}^T (2\ 3)\right)^2 = 8^2 = 64$$

If $c=1$

$$K(x,y) = (c + x^T y)^q$$

$$= \left(1 + \begin{pmatrix}1\\2\end{pmatrix}^T (2,3)\right)^2 = (1+8)^2 = 81$$

Gaussian Kernel

→ Radial Basis functions (RBFs) are Gaussian kernels are extremely useful in SVM.

→ The RBF funcn is shown as below:-

$$K(x,y) = e^{\frac{-(x-y)^2}{2\sigma^2}}$$

σ is the constant

→ Here, σ is an important parameter. If σ is small, then the RBF is similar to linear SVM and if σ is large, then the kernel is influenced by more support vectors.

a) Consider 2 data points $x=(1,2)$ and $y=(2,3)$ with $\sigma=1$. Apply RBF kernel and find the value of RBF kernel for these points.

$$K(x,y) = e^{\frac{-(x-y)^2}{2\sigma^2}}$$

- Substitute the value of x and y in RBF kernel.
- The squared distance b/w the points (1,2) and (2,3) is given as:-

$$(1-2)^2+(2-3)^2 = 2$$

- If $\sigma=1$, then $K(x,y) = e^{\{\frac{-2}{2}\}} = e^{-1}$

$$= 0.3679$$

Sigmoid Kernel

→ The sigmoid kernel is given as:-

$$K(x_i, y_i) = \tanh(K x_i y_j - \sigma)$$

tanh is the hyperbolic tangent function is often used to ensure that the o/p of the kernel function is between -1 and 1.

K and σ are parameters.

→ The sigmoid kernel is less commonly used compared to other kernels like the

linear, polynomial or Radial basis function (RBF) kernels.

→ The choice of the kernel depends on the nature of the data and the problem at hand. and different kernels may perform better for diff types of datasets.

IMPUTATION TECHNIQUES

IMPUTATION

Imputation is a technique used in feature engineering to handle missing or incomplete data by filling in the missing values with estimated or predicted values. It is a crucial step in the data preprocessing pipeline, as many machine learning algorithms cannot handle missing data.

Here are some common methods for imputation in feature engineering:

. **Mean/Median Imputation:**

- Replace missing values with the mean (for numerical data) or median (robust to outliers) of the respective feature. This is a simple and quick method, but it assumes that the missing values are missing completely at random.

2. **Mode Imputation:**
   - For categorical features, you can replace missing values with the most frequent category (mode) of that feature.
3. **Forward Fill/Backward Fill:**
   - For time-series data, missing values can be filled using the values from the previous (forward fill) or next (backward fill) time point.
4. **Linear Regression Imputation:**
   - Use linear regression models to predict missing values based on other features. This method assumes a linear relationship between the feature with missing values and other relevant features.
5. **K-Nearest Neighbors (KNN) Imputation:**
   - Estimate missing values based on the values of their k-nearest neighbors. This method is useful when there is a local structure in the data.

6. **Deep Learning Approach**
7. **Multiple Imputation**

```
Original Dataset:
|   X   |   Y   |
|-------|-------|
|  2.5  |   3   |
|   4   |   5   |
|  NaN  |   7   |
|   8   |   1   |
|  6.5  |   2   |
```

In this dataset, the value in the "X" column for the third row is missing (represented as NaN). We want to impute this missing value using mean or median imputation.

Mean Imputation:

Calculate the mean of the non-missing values in the "X" column:

makefile Copy code

```
Mean_X = (2.5 + 4 + 8 + 6.5) / 4 = 5
```

mathematica

Copy code

```
Original Dataset:
|   A   |   B   |
|-------|-------|
|  Red  |   3   |
| Green |   5   |
|  NaN  |   7   |
| Blue  |   1   |
| Green |   2   |
```

In this dataset, the value in the "A" column for the third row is missing (represented as NaN). We want to impute this missing value using mode imputation.

Mode Imputation:

Calculate the mode (most frequent category) of the non-missing values in the "A" column:

makefile

Copy code

```
Mode_A = Green
```

```
Original Time-Series Dataset:
| Time  | Value |
|-------|-------|
|   1   |   2   |
|   2   |   NaN |
|   3   |   5   |
|   4   |   NaN |
|   5   |   8   |
```

In this dataset, the values for time points 2 and 4 are missing (represented as NaN). Forward fill involves filling missing values with the most recent observed value.

Applying forward fill:

sql Copy code

```
| Time  | Value |
|-------|-------|
|   1   |   2   |
|   2   |   2   |  <- Forward fill with the value from time point 1
|   3   |   5   |
|   4   |   5   |  <- Forward fill with the value from time point 3
|   5   |   8   |
```

Backward Fill:

Alternatively, backward fill involves filling missing values with the next observed value. Applying backward fill to the original dataset:

```sql
| Time  | Value |
|-------|-------|
|   1   |   2   |
|   2   |   5   |  <- Backward fill with the value from time point 3
|   3   |   5   |
|   4   |   8   |  <- Backward fill with the value from time point 5
|   5   |   8   |
```

Now, the missing values at time points 2 and 4 have been filled using the next observed values.

Deep Learning-Based Imputation:

1. **Prepare Data:**
   - Split the dataset into two parts: one with complete data (training set) and another with missing values (imputation set).

sql Copy code

```
Training Set:
|   X   |   Y   |
|-------|-------|
|  2.5  |   3   |
|   4   |   5   |
|   8   |   1   |
|  6.5  |   2   |

Imputation Set:
|   X   |   Y   |
|-------|-------|
|  NaN  |   7   |
```

2. **Build Neural Network:**
   - Design a neural network architecture suitable for the task. In this example, a simple feedforward neural network with one hidden layer is used.

python Copy code

```python
# Example code using TensorFlow/Keras
from tensorflow import keras
from tensorflow.keras import layers

model = keras.Sequential([
    layers.Dense(10, activation='relu', input_shape=(1,)),
    layers.Dense(1)
])

model.compile(optimizer='adam', loss='mean_squared_error')
```

3. **Train Neural Network:**
   - Train the neural network using the training set, where "Y" is the input, and "X" is the target variable.

```python
model.fit(training_set['Y'], training_set['X'], epochs=100, batch_size=1)
```

4. **Predict Missing Values:**
   - Use the trained neural network to predict the missing values in the imputation set based on the available values of "Y."

```python
imputation_set['X'] = model.predict(imputation_set['Y'])
```

The neural network will output predicted values for the missing "X" values in the imputation set.

```mathematica
Imputed Imputation Set:
|         X          |   Y   |
|--------------------|-------|
|  Predicted Value   |   7   |
```

Linear Regression Imputation:

1. **Prepare Data:**
   - Split the dataset into two parts: one with complete data (training set) and another with missing values (imputation set).

sql

Copy code

```
Training Set:
|   X   |   Y   |
|-------|-------|
|  2.5  |   3   |
|   4   |   5   |
|   8   |   1   |
|  6.5  |   2   |

Imputation Set:
|   X   |   Y   |
|-------|-------|
|  NaN  |   7   |
```

2. **Train a Linear Regression Model:**
   - Use the training set to train a linear regression model. The model learns the linear relationship between "X" and "Y."

python Copy code

```python
from sklearn.linear_model import LinearRegression

# Assuming 'training_set' is a DataFrame with 'X' and 'Y' columns
X_train = training_set[['Y']]
y_train = training_set['X']

lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
```

3. **Predict Missing Values:**
   - Use the trained linear regression model to predict the missing values in the imputation set. Use the available values in "Y" as input to predict the missing values in "X."

```python
# Assuming 'imputation_set' is a DataFrame with 'X' and 'Y' columns
X_imputation = imputation_set[['Y']]
predicted_values = lr_model.predict(X_imputation)

imputation_set['X'] = predicted_values
```

The linear regression model predicts values for the missing "X" based on the observed linear relationship in the training set.

```mathematica
Imputed Imputation Set:
|         X          |   Y   |
|--------------------|-------|
|  Predicted Value   |   7   |
```