Data Preprocessing Phase

The goal of this stage is to prepare the data such that you can perform Inferential Stats. In short, you are making the data compatible for the Inferential Stats!

Reason:
1. Every AI engineer expects your data to be COMPLETE (no NANs Strictly)
2. Every Algo in the Inferential Stats expects your data to be completely NUMERIC.

Preprocessing Task (Goal: To make your data COMPLETE and NUMERIC)
1. Check and Handle the Missing Data
2. Check and Handle Categorical Data
3. Check and Handle Ordinal Data
4. Perform Data Standardization(optional)

1. Check and Handle the Missing Data

There are three perspectives to solve the missing data problem
a. Use Stat Approach
b. Use Domain Approach
c. Use Hybrid Approach - Some columns using stat while some using domain if you are aware.

Guidelines to Handling Missing Data (Stats Approach)
==========
a. Numerical Data(ND):
    a. Continuous ND : Replace Missing Values (NaN) with the mean value of the column
    b. Discrete ND : Replace Missing Values (NaN) with the median value of the column

b. Non-Numerical Data:
    Replace Missing Value with the Mode's first value

Guidelines by Prashant Nair to Handling Missing Data (Domain Approach)
==========

Irrespective of the type of the data column, replace Missing Data with the default value as specified by the domain.

real-estate industry in Mumbai India (MMRDA)

Whenever a builder builds a tower/skyscraper/building for residential purpose, it is mandatory to supply parking space to each flat owner
depending on the type of the flat

2BHK ---- 1 Parking Space
3BHK ---- 2 Parking Space
4BHK and Above -- 3 parking Space

Dataset of Building in Mumbai Region
Parking -- NaN(Replace NaN with 1,2,3 depending on type of flat configuration(2BHK ,3BHK, 4BHK))

1	Read Dataset	`data = pd.read_csv('datasample.csv')`

		`data.info()`
		
		`<class 'pandas.core.frame.DataFrame'>`
		`RangeIndex: 10 entries, 0 to 9`
		`Data columns (total 4 columns):`
		` #  Column  Non-Null Count  Dtype`
		`---  ------  --------------  -----`
		` 0  Country  9 non-null  object`
		` 1  Age  9 non-null  float64`
		` 2  Salary  9 non-null  float64`
		` 3  Purchased  10 non-null  object`
		`dtypes: float64(2), object(2)`
		`memory usage: 448.0+ bytes`

2	How many NaNs in each column	`data.isna().sum()`
		
		`Country  1`
		`Age  1`
		`Salary  1`
		`Purchased  0`
		`dtype: int64`

	`# Stat Approach`
	
	`# Country Column -- Use mode`	

	`# Age Column ------ Use mean` `# Salary Column --- USe mean`	

	Lets Handle missing value for country column.
	As country column is a categorical data, the guideline suggest to:
	replace NaN with the mode's first value
		`data['Country'].fillna(`
		`data['Country'].mode()[0] ,`
		`inplace=True)`

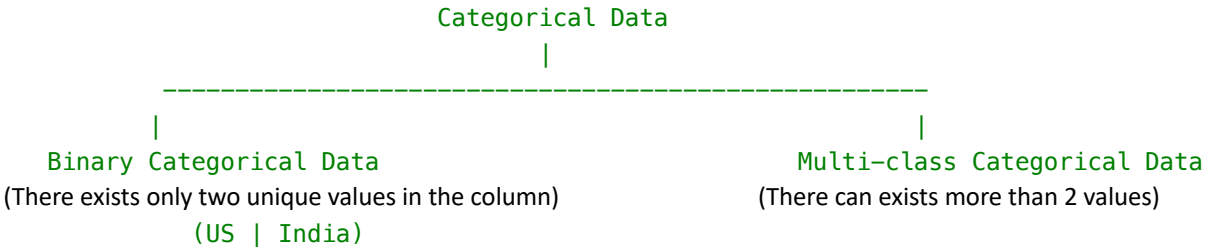
	Lets Handle missing value for Salary column.
	As Salary column is a continuous ND data, the guideline suggest to:
	- replace NaN with the mean value
		`data['Salary'].fillna(`
		`data['Salary'].mean() ,`
		`inplace=True)`

	Lets Handle missing value for Age column. As Age column is a discrete ND data for example purpose, the guideline suggest to - replace NaN with the median value	`data['Age'].fillna(` `data['Age'].median() , inplace=True)`

		`data.info()`
		
		`<class 'pandas.core.frame.DataFrame'>`
		`RangeIndex: 10 entries, 0 to 9`
		`Data columns (total 4 columns):`
		` #  Column  Non-Null Count  Dtype`
		`---  ------  --------------  -----`
		` 0  Country  10 non-null  object`
		` 1  Age  10 non-null  float64`
		` 2  Salary  10 non-null  float64`
		` 3  Purchased  10 non-null  object`
		`dtypes: float64(2), object(2)`
		`memory usage: 448.0+ bytes`

		`data.isna().sum()`
		
		`Country  0`
		`Age  0`
		`Salary  0`
		`Purchased  0`
		`dtype: int64`

Handling Categorical Data



Strategy:

1. Arrange the data in asc order
['India', 'US']
1

Replace data with 0 and 1
based on index loc of list

		Data Country Age Salary Purchased 0 France 44.0 72000.000000 0 1 Spain 27.0 48000.000000 1 2 Germany 30.0 54000.000000 0 3 Spain 38.0 61000.000000 0 4 Germany 40.0 63777.777778 1 5 France 35.0 58000.000000 1 6 Spain 38.0 52000.000000 0 7 France 48.0 79000.000000 1 8 France 50.0 83000.000000 0 9 France 37.0 67000.000000 1

1		`list1 = ['India', 'US']`
		`sorted(list1)`
		
		`['India', 'US']`

		`sorted(data['Purchased'].unique())`

		['No', 'Yes']
		data['Purchased'].replace(['No','Yes'],[0,1] , inplace=True)
	Change categorical columns to numberical One Hot Encoding ---> Creates Dummy Variables	from sklearn.preprocessing import OneHotEncoder ohe = OneHotEncoder(sparse=False) fCountry = ohe.fit_transform(features[:,0].reshape(-1,1)) fCountry array([[1., 0., 0.], [0., 0., 1.], [0., 1., 0.], [0., 0., 1.], [0., 1., 0.], [1., 0., 0.], [0., 0., 1.], [1., 0., 0.], [1., 0., 0.], [1., 0., 0.]])