```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
import pandas as pd
import io
#data = pd.read_csv(io.BytesIO(uploaded['housing.csv']))
data = pd.read_csv('/content/sample_data/housing.csv')
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object 
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

```
data.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	populatic
0	-122.23	37.88	41.0	880.0	129.0	322
1	-122.22	37.86	21.0	7099.0	1106.0	2401
2	-122.24	37.85	52.0	1467.0	190.0	496
3	-122.25	37.85	52.0	1274.0	235.0	558
4	-122.25	37.85	52.0	1627.0	280.0	565

Encoding

(1)Label encoder

(2)Onehot encoder

```
from sklearn.preprocessing import LabelEncoder , OneHotEncoder
data['median_house_value'].value_counts()
```

```
500001.0    965
137500.0    122
162500.0    117
```

```
112500.0    103
187500.0     93
           ...
303200.0      1
307900.0      1
383200.0      1
360800.0      1
405500.0      1
Name: median_house_value, Length: 3842, dtype: int64
```

```
le=LabelEncoder()
data['median_house_value']=le.fit_transform(data['median_house_value'])
data['median_house_value'].value_counts()
```

```
3841    965
959     122
1209    117
710     103
1459     93
       ...
3172      1
3275      1
3204      1
3091      1
2119      1
Name: median_house_value, Length: 3842, dtype: int64
```

```
le.classes_
```

```
array([ 14999.,  17500.,  22500., ..., 499100., 500000., 500001.])
```

```
data['ocean_proximity'].value_counts()
```

```
<1H OCEAN     9136
INLAND        6551
NEAR OCEAN    2658
NEAR BAY      2290
ISLAND           5
Name: ocean_proximity, dtype: int64
```

```
one_hot = OneHotEncoder()
transformed_data = one_hot.fit_transform(data['ocean_proximity'].values.reshape(-1,1)).toa
one_hot.categories_
```

```
[array(['<1H OCEAN', 'INLAND', 'ISLAND', 'NEAR BAY', 'NEAR OCEAN'],
       dtype=object)]
```

```
transformed_data = pd.DataFrame(transformed_data ,
                                columns = ['<1H OCEAN', 'INLAND', 'ISLAND', 'NEAR BAY','NE
transformed_data.head()
```

	<1H OCEAN	INLAND	ISLAND	NEAR BAY	NEAR OCEAN
0	0.0	0.0	0.0	1.0	0.0
1	0.0	0.0	0.0	1.0	0.0
2	0.0	0.0	0.0	1.0	0.0

```
transformed_data.iloc[90, ]
```

```
<1H OCEAN     0.0
INLAND        0.0
ISLAND        0.0
NEAR BAY      1.0
NEAR OCEAN    0.0
Name: 90, dtype: float64
```

```
data['median_house_value'][90]
```

```
1209
```

Normalization & Standardization

```
# consider only numerical columns
```

```
numeric_columns = [c for c in data.columns if data[c].dtype != np.dtype('O')]
numeric_columns
```

```
['longitude',
 'latitude',
 'housing_median_age',
 'total_rooms',
 'total_bedrooms',
 'population',
 'households',
 'median_income',
 'median_house_value']
```

```
len(numeric_columns) , len(data.columns)
```

```
(9, 10)
```

```
numeric_columns.remove('longitude')
numeric_columns.remove('latitude')
```

```
temp_data = data[numeric_columns]
temp_data
```

	housing_median_age	total_rooms	total_bedrooms	population	households	me
0	41.0	880.0	129.0	322.0	126.0	
1	21.0	7099.0	1106.0	2401.0	1138.0	
2	52.0	1467.0	190.0	496.0	177.0	
3	52.0	1274.0	235.0	558.0	219.0	
4	52.0	1627.0	280.0	565.0	259.0	
...	...	...	...	...	...	
20635	25.0	1665.0	374.0	845.0	330.0	
20636	18.0	697.0	150.0	356.0	114.0	
20637	17.0	2254.0	485.0	1007.0	433.0	
20638	18.0	1860.0	409.0	741.0	349.0	

Normalization

20640 rows × 7 columns

```
from sklearn.preprocessing import StandardScaler , MinMaxScaler
import warnings
warnings.filterwarnings('ignore')
normalizer = MinMaxScaler()
temp_data.dropna(axis = 1 , inplace = True)
normalized_data = normalizer.fit_transform(temp_data)
pd.DataFrame(normalized_data , columns = temp_data.columns)
```

	housing_median_age	total_rooms	population	households	median_income	mec
0	0.784314	0.022331	0.008941	0.020556	0.539668	
1	0.392157	0.180503	0.067210	0.186976	0.538027	
2	1.000000	0.037260	0.013818	0.028943	0.466028	
3	1.000000	0.032352	0.015555	0.035849	0.354699	
4	1.000000	0.041330	0.015752	0.042427	0.230776	
...	...	...	...	...	...	
20635	0.470588	0.042296	0.023599	0.054103	0.073130	
20636	0.333333	0.017676	0.009894	0.018582	0.141853	
20637	0.313725	0.057277	0.028140	0.071041	0.082764	
20638	0.333333	0.047256	0.020684	0.057227	0.094295	
20639	0.294118	0.070782	0.038790	0.086992	0.130253	

20640 rows × 6 columns

Standardization

```
standard_scaler = StandardScaler()
standardized_data = standard_scaler.fit_transform(temp_data)
pd.DataFrame(standardized_data , columns = temp_data.columns)
```

	housing_median_age	total_rooms	population	households	median_income	mec
0	0.982143	-0.804819	-0.974429	-0.977033	2.344766	
1	-0.607019	2.045890	0.861439	1.669961	2.332238	
2	1.856182	-0.535746	-0.820777	-0.843637	1.782699	
3	1.856182	-0.624215	-0.766028	-0.733781	0.932968	
4	1.856182	-0.462404	-0.759847	-0.629157	-0.012881	
...	...	...	...	...	...	
20635	-0.289187	-0.444985	-0.512592	-0.443449	-1.216128	
20636	-0.845393	-0.888704	-0.944405	-1.008420	-0.691593	
20637	-0.924851	-0.174995	-0.369537	-0.174042	-1.142593	
20638	-0.845393	-0.355600	-0.604429	-0.393753	-1.054583	
20639	-1.004309	0.068408	-0.033977	0.079672	-0.780129	

20640 rows × 6 columns

Handling With Missing Values

```
data.isnull().sum()
```

```
longitude               0
latitude                0
housing_median_age      0
total_rooms             0
total_bedrooms        207
population              0
households              0
median_income           0
median_house_value      0
ocean_proximity         0
dtype: int64
```

```
# here I Will show you imputing values in Null columns only for 'agent' column
data['total_bedrooms'].isnull().sum()
```

```
207
```

Simple Imputer

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan , strategy='mean')
agent_col = imputer.fit_transform(data['total_bedrooms'].values.reshape(-1,1))
pd.DataFrame(agent_col).isnull().sum()
pd.DataFrame(agent_col).isnull().sum()
```

```
0    0
dtype: int64
```

```
data['total_bedrooms'].isnull().sum()
```

```
207
```

Discretization

```
from sklearn.preprocessing import KBinsDiscretizer
temp_data.head()
```

	housing_median_age	total_rooms	population	households	median_income	median_ho
0	41.0	880.0	322.0	126.0	8.3252	
1	21.0	7099.0	2401.0	1138.0	8.3014	
2	52.0	1467.0	496.0	177.0	7.2574	
3	52.0	1274.0	558.0	219.0	5.6431	
4	52.0	1627.0	565.0	259.0	3.8462	

Quantile Discretization Transform

```
trans = KBinsDiscretizer(n_bins =10 , encode = 'ordinal' , strategy='quantile')
new_data = trans.fit_transform(temp_data)
pd.DataFrame(new_data,columns = temp_data.columns )
```

Uniform Discretization Transform

```
trans = KBinsDiscretizer(n_bins =10 , encode = 'ordinal' , strategy='uniform')
new_data = trans.fit_transform(temp_data)

pd.DataFrame(new_data,columns = temp_data.columns )
```

	housing_median_age	total_rooms	population	households	median_income	mec
0	7.0	0.0	0.0	0.0	5.0	
1	3.0	1.0	0.0	1.0	5.0	
2	9.0	0.0	0.0	0.0	4.0	
3	9.0	0.0	0.0	0.0	3.0	
4	9.0	0.0	0.0	0.0	2.0	
...	...	...	...	...	...	
20635	4.0	0.0	0.0	0.0	0.0	
20636	3.0	0.0	0.0	0.0	1.0	
20637	3.0	0.0	0.0	0.0	0.0	
20638	3.0	0.0	0.0	0.0	0.0	
20639	2.0	0.0	0.0	0.0	1.0	

20640 rows × 6 columns

KMeans Discretization Transform

```
trans = KBinsDiscretizer(n_bins =10 , encode = 'ordinal' , strategy='kmeans')
new_data = trans.fit_transform(temp_data)

pd.DataFrame(new_data,columns = temp_data.columns )
```

	housing_median_age	total_rooms	population	households	median_income	mec
0	7.0	0.0	0.0	0.0	6.0	
1	3.0	4.0	3.0	4.0	6.0	
2	9.0	1.0	0.0	0.0	6.0	
3	9.0	0.0	0.0	0.0	4.0	
4	9.0	1.0	0.0	1.0	3.0	
...	...	...	...	...	...	
20635	4.0	1.0	1.0	1.0	0.0	
20636	3.0	0.0	0.0	0.0	1.0	
20637	3.0	1.0	1.0	2.0	0.0	
20638	3.0	1.0	0.0	1.0	0.0	
20639	2.0	2.0	1.0	2.0	1.0	

20640 rows × 6 columns

✓ 1s completed at 8:05 PM ● ×