```python
#importing libraries
import numpy as np
import pandas as pd

Data = {'Name':['Alice', 'Bob', 'John','Lisa'],
        'Age':[21,20,22,23],
        'City':['San franscisco','New year','Los angeles','Chicago']}

df = pd.DataFrame(Data)
print(df)
```

```
    Name  Age            City
0  Alice   21  San franscisco
1    Bob   20        New year
2   John   22     Los angeles
3   Lisa   23         Chicago
```

```python
# Handling Missing Values
from sklearn.preprocessing import LabelEncoder
Data = {'Name':['Alice', 'Bob', 'John',None],
        'Age':[21,None,22,23],
        'City':['San franscisco','New year','Los angeles','Chicago']}

df = pd.DataFrame(Data)
print(df)

#Checking for missing value
print("Missing Values:\n",df.isnull())

#Dropping rows with missing values
df_cleaned = df.dropna()

#filling missing values with specified values
mean_values = df['Age'].mean()
df_filled = df.fillna(value={'Name':'unknown',
                             'Age': df['Age'].mean()})

print("\nDataframe after dropping missing values\n",df_cleaned)
print("\nDataframe after filling missing values\n",df_filled)

#encoding Categorical Data
label_encoder = LabelEncoder()
df['Encoded_city'] =
label_encoder.fit_transform(df['City'].astype(str))
print("\n After Encoding City")
print(df)
```

```
    Name   Age            City
0  Alice  21.0  San franscisco
1    Bob   NaN        New year
2   John  22.0     Los angeles
```

```
3   None  23.0         Chicago
Missing Values:
     Name    Age   City
0  False  False  False
1  False   True  False
2  False  False  False
3   True  False  False

Dataframe after dropping missing values
     Name   Age            City
0  Alice  21.0  San franscisco
2   John  22.0     Los angeles

Dataframe after filling missing values
       Name   Age            City
0    Alice  21.0  San franscisco
1      Bob  22.0        New year
2     John  22.0     Los angeles
3  unknown  23.0         Chicago

 After Encoding City
    Name   Age            City  Encoded_city
0  Alice  21.0  San franscisco             3
1    Bob   NaN        New year             2
2   John  22.0     Los angeles             1
3   None  23.0         Chicago             0
```

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

# Encoding Categorical Data
categories = ['red', 'blue', 'green', 'yellow', 'white']

label_encoder = LabelEncoder()
numeric_labels = label_encoder.fit_transform(categories)

print(categories)
print(numeric_labels)

# Creating DataFrame
data = {'Name': ['Alice', 'Bob', 'John', None],
        'Age': [21, None, 22, 23],
        'City': ['San Francisco', 'New York', 'Los Angeles',
'Chicago']}
df = pd.DataFrame(data)

# Encoding 'City' column
label_encoder = LabelEncoder()
df['City_Encoded'] = label_encoder.fit_transform(df['City'])
```

```
X = df[['Age', 'City_Encoded']]
y = df['Name']

# Splitting the data into training and test sets
x_train, x_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

print("Encoded DataFrame:")
print(df)
print("\nX_train:")
print(x_train)
print("\nX_test:")
print(x_test)
```

```
['red', 'blue', 'green', 'yellow', 'white']
[2 0 1 4 3]
Encoded DataFrame:
    Name   Age           City  City_Encoded
0  Alice  21.0  San Francisco             3
1    Bob   NaN       New York             2
2   John  22.0    Los Angeles             1
3   None  23.0        Chicago             0

X_train:
    Age  City_Encoded
3  23.0             0
0  21.0             3
2  22.0             1

X_test:
   Age  City_Encoded
1  NaN             2
```

```
# Independent variables (features)
X = df[['Name', 'Age', 'City']]

# Dependent variable (target)
y = df['City_Encoded']

print("Independent Variables (Features):\n", X)
print("Dependent Variable (Target):\n", y)
```

```
Independent Variables (Features):
     Name   Age           City
0  Alice  21.0  San Francisco
1    Bob   NaN       New York
2   John  22.0    Los Angeles
3   None  23.0        Chicago
Dependent Variable (Target):
 0    3
1    2
```

```
2    1
3    0
Name: City_Encoded, dtype: int32
```

```
from sklearn.preprocessing import StandardScaler
import pandas as pd

Data = {'Name':['Alice', 'Bob', 'John',None],
        'Age':[21,None,22,23],
        'City':['San franscisco','New year','Los angeles','Chicago']}

df = pd.DataFrame(Data)
# Instantiate the StandardScaler
scaler = StandardScaler()

# Fit and transform the 'Age' column
X = df[['Name', 'Age', 'City']]
X['Age'] = scaler.fit_transform(X[['Age']].fillna(X[['Age']].mean()))

print("Scaled Independent Variables (Features):\n", X)
```

```
Scaled Independent Variables (Features):
     Name       Age            City
0  Alice -1.414214  San franscisco
1    Bob  0.000000        New year
2   John  0.000000     Los angeles
3   None  1.414214         Chicago
```

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

df = pd.read_csv('House_data.csv')
print(df)
```

```
                     date         price  bedrooms  bathrooms
sqft_living  \
0     2014-05-02 00:00:00  3.130000e+05       3.0       1.50
1340
1     2014-05-02 00:00:00  2.384000e+06       5.0       2.50
3650
2     2014-05-02 00:00:00  3.420000e+05       3.0       2.00
1930
3     2014-05-02 00:00:00  4.200000e+05       3.0       2.25
2000
4     2014-05-02 00:00:00  5.500000e+05       4.0       2.50
1940
...                   ...           ...       ...        ...
...
4595  2014-07-09 00:00:00  3.081667e+05       3.0       1.75
```

```
1510
4596  2014-07-09 00:00:00  5.343333e+05       3.0       2.50
1460
4597  2014-07-09 00:00:00  4.169042e+05       3.0       2.50
3010
4598  2014-07-10 00:00:00  2.034000e+05       4.0       2.00
2090
4599  2014-07-10 00:00:00  2.206000e+05       3.0       2.50
1490

      sqft_lot  floors  waterfront  view  condition  sqft_above  \
0         7912     1.5           0     0          3        1340
1         9050     2.0           0     4          5        3370
2        11947     1.0           0     0          4        1930
3         8030     1.0           0     0          4        1000
4        10500     1.0           0     0          4        1140
...        ...     ...         ...   ...        ...         ...
4595      6360     1.0           0     0          4        1510
4596      7573     2.0           0     0          3        1460
4597      7014     2.0           0     0          3        3010
4598      6630     1.0           0     0          3        1070
4599      8102     2.0           0     0          4        1490

      sqft_basement  yr_built  yr_renovated                    street
\
0                 0      1955          2005      18810 Densmore Ave N

1               280      1921             0           709 W Blaine St

2                 0      1966             0  26206-26214 143rd Ave SE

3              1000      1963             0           857 170th Pl NE

4               800      1976          1992         9105 170th Ave NE

...             ...       ...           ...                       ...

4595              0      1954          1979            501 N 143rd St

4596              0      1983          2009          14855 SE 10th Pl

4597              0      2009             0          759 Ilwaco Pl NE

4598           1020      1974             0         5148 S Creston St

4599              0      1990             0         18717 SE 258th St


           city  statezip country
0     Shoreline  WA 98133     USA
1       Seattle  WA 98119     USA
```

```
2          Kent  WA 98042     USA
3      Bellevue  WA 98008     USA
4       Redmond  WA 98052     USA
...         ...       ...     ...
4595    Seattle  WA 98133     USA
4596   Bellevue  WA 98007     USA
4597     Renton  WA 98059     USA
4598    Seattle  WA 98178     USA
4599  Covington  WA 98042     USA

[4600 rows x 18 columns]
```

```
print(df.describe())
```

```
              price     bedrooms    bathrooms   sqft_living
sqft_lot  \
count  4.600000e+03  4600.000000  4600.000000   4600.000000
4.600000e+03
mean   5.519630e+05     3.400870     2.160815   2139.346957
1.485252e+04
std    5.638347e+05     0.908848     0.783781    963.206916
3.588444e+04
min    0.000000e+00     0.000000     0.000000    370.000000
6.380000e+02
25%    3.228750e+05     3.000000     1.750000   1460.000000
5.000750e+03
50%    4.609435e+05     3.000000     2.250000   1980.000000
7.683000e+03
75%    6.549625e+05     4.000000     2.500000   2620.000000
1.100125e+04
max    2.659000e+07     9.000000     8.000000  13540.000000
1.074218e+06

            floors   waterfront          view    condition   sqft_above
\
count  4600.000000  4600.000000  4600.000000  4600.000000  4600.000000

mean      1.512065     0.007174     0.240652     3.451739  1827.265435

std       0.538288     0.084404     0.778405     0.677230   862.168977

min       1.000000     0.000000     0.000000     1.000000   370.000000

25%       1.000000     0.000000     0.000000     3.000000  1190.000000

50%       1.500000     0.000000     0.000000     3.000000  1590.000000

75%       2.000000     0.000000     0.000000     4.000000  2300.000000

max       3.500000     1.000000     4.000000     5.000000  9410.000000
```

```
       sqft_basement     yr_built  yr_renovated
count    4600.000000  4600.000000   4600.000000
mean      312.081522  1970.786304    808.608261
std       464.137228    29.731848    979.414536
min         0.000000  1900.000000      0.000000
25%         0.000000  1951.000000      0.000000
50%         0.000000  1976.000000      0.000000
75%       610.000000  1997.000000   1999.000000
max      4820.000000  2014.000000   2014.000000
```

```
print(df.isnull().sum())
```

```
date             0
price            0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
view             0
condition        0
sqft_above       0
sqft_basement    0
yr_built         0
yr_renovated     0
street           0
city             0
statezip         0
country          0
dtype: int64
```

```
new_df = df.drop(labels=['street','city','statezip','country'],axis=1)
new_df =new_df.drop('date',axis=1)
new_df
```

```
              price  bedrooms  bathrooms  sqft_living  sqft_lot  floors
\
0     3.130000e+05       3.0       1.50         1340      7912     1.5

1     2.384000e+06       5.0       2.50         3650      9050     2.0

2     3.420000e+05       3.0       2.00         1930     11947     1.0

3     4.200000e+05       3.0       2.25         2000      8030     1.0

4     5.500000e+05       4.0       2.50         1940     10500     1.0

...            ...       ...        ...          ...       ...     ...
```

```
4595  3.081667e+05       3.0       1.75         1510       6360     1.0
4596  5.343333e+05       3.0       2.50         1460       7573     2.0
4597  4.169042e+05       3.0       2.50         3010       7014     2.0
4598  2.034000e+05       4.0       2.00         2090       6630     1.0
4599  2.206000e+05       3.0       2.50         1490       8102     2.0

      waterfront  view  condition  sqft_above  sqft_basement  yr_built
\
0              0     0          3        1340              0      1955
1              0     4          5        3370            280      1921
2              0     0          4        1930              0      1966
3              0     0          4        1000           1000      1963
4              0     0          4        1140            800      1976
...          ...   ...        ...         ...            ...       ...
4595           0     0          4        1510              0      1954
4596           0     0          3        1460              0      1983
4597           0     0          3        3010              0      2009
4598           0     0          3        1070           1020      1974
4599           0     0          4        1490              0      1990

      yr_renovated
0             2005
1                0
2                0
3                0
4             1992
...            ...
4595          1979
4596          2009
4597             0
4598             0
4599             0

[4600 rows x 13 columns]
```

```
X =
new_df[['bedrooms','bathrooms','sqft_living','sqft_lot','floors','wate
rfront','view','condition']]
y = new_df['price']

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,
random_state=42)
model = LinearRegression()
model.fit(X_train, y_train)
```

```
LinearRegression()
```

```
import matplotlib.pyplot as plt

y_pred = model.predict(X_test)

plt.scatter(y_test,y_pred)
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual Prices vs. Predicted Prices")
plt.show()
```