Simple List Imputer ¶

[2]:
```python
from sklearn.impute import SimpleImputer
import numpy as np

data = np.array([
    [1, 2, np.nan ],
    [4, np.nan, 6],
    [np.nan, 8, 9]
])

imputer = SimpleImputer(strategy='mean', fill_value =1)

imputed_data = imputer.fit_transform(data)

print(data)
print(imputed_data)
```

```
[[ 1.   2. nan]
 [ 4. nan  6.]
 [nan  8.  9.]]
[[1.  2.  7.5]
 [4.  5.  6. ]
 [2.5 8.  9. ]]
```

KNN Imputer

[5]:
```python
from sklearn.impute import KNNImputer
import numpy as np

data = np.array([
    [2, 4, np.nan ],
    [5, 1, 6 ],
    [np.nan, 5, 7 ],
    [9, 8, 9 ]
])

knn_imputer = KNNImputer(n_neighbors=2)

imputed_data = knn_imputer.fit_transform(data)

print(data)
print(imputed_data)
```

```
[[ 2.  4. nan]
 [ 5.  1.  6.]
 [nan  5.  7.]
 [ 9.  8.  9.]]
[[2.  4.  6.5]
 [5.  1.  6. ]
 [5.5 5.  7. ]
 [9.  8.  9. ]]
```

Mean Removal

[11]:
```python
import numpy as np
from sklearn import preprocessing

input_data = np.array([[3,-1.5,3,-6.4], [0,3,-1.3,4.1],[1,2.3,-2.9,-4.3]])
data_standardized = preprocessing.scale(input_data)

print("\n Mean = ", data_standardized.mean(axis = 0))
print(" Std deviation = ",data_standardized.std(axis = 0))
```

```
 Mean =  [ 5.55111512e-17 -3.70074342e-17  0.00000000e+00 -1.85037171e-17]
 Std deviation =  [1. 1. 1. 1.]
```

Min-Max Scale

[13]:
```python
import numpy as np
from sklearn import preprocessing

input_data = np.array([[3,-1.5,3,-6.4], [0,3,-1.3,4.1],[1,2.3,-2.9,-4.3]])

data_scaler = preprocessing.MinMaxScaler(feature_range = (0,1))
data_scaled = data_scaler.fit_transform(input_data)

print("\n Min Max Scaled Data = ",data_scaled)
```

```
 Min Max Scaled Data =  [[1.         0.         1.         0.        ]
 [0.         1.         0.27118644 1.        ]
 [0.33333333 0.84444444 0.         0.2       ]]
```

Normalization 1

[18]:
```python
import numpy as np
from sklearn import preprocessing

input_data = np.array([[3,-1.5,3,-6.4], [0,3,-1.3,4.1],[1,2.3,-2.9,-4.3]])

data_normalized = preprocessing.normalize(input_data, norm = 'l1')

print("\n L1 Normalized Data = ",data_normalized)
```

```
 L1 Normalized Data =  [[ 0.21582734 -0.10791367  0.21582734 -0.46043165]
 [ 0.          0.35714286 -0.1547619   0.48809524]
 [ 0.0952381   0.21904762 -0.27619048 -0.40952381]]
```

Normalization 2

[21]:
```python
import numpy as np
from sklearn import preprocessing

input_data = np.array([[3,-1.5,3, -6.4],[0,3,-1.3,4.1],[1,2.3, -2.9, -4.3]])

data_normalized = preprocessing.normalize(input_data, norm = 'l2')

print("\n L2 Normalized Data = ",data_normalized)
```

```
 L2 Normalized Data =  [[ 0.38345117 -0.19172558  0.38345117 -0.81802916]
 [ 0.          0.57207755 -0.24790027  0.78183932]
 [ 0.17357868  0.39923096 -0.50337816 -0.74638831]]
```

Binarizing

[24]:
```python
import numpy as np
from sklearn.preprocessing import Binarizer

ages= np.array([[15], [22], [18],[30], [16], [25]])

binarizer = Binarizer(threshold = 18)
binary_ages = binarizer.fit_transform(ages)

print(" Original ages: \n",ages)
print(" Binarized ages: \n",binary_ages)
```

```
 Original ages: 
 [[15]
 [22]
 [18]
 [30]
 [16]
 [25]]
 Binarized ages: 
 [[0]
 [1]
 [0]
 [1]
 [0]
 [1]]
```

Label Encoding using scikit-learn library

[27]:
```python
import pandas as pd
from sklearn import preprocessing

my_data = {
    "Gender": ['F', 'M', 'M', 'F', 'M', 'F', 'M', 'F', 'F', 'M'],
    "Name" : ['Krishna', 'Rudra', 'Hrithik', 'Kiara', 'Dhoni', 'Sara', 'Rajesh', 'Lata', 'Anuska', 'Ratan']
}

blk = pd.DataFrame(my_data)
print("Genuine Data Frame : \n")
print(blk)

my_label = preprocessing.LabelEncoder()
blk['Gender'] = my_label.fit_transform(blk['Gender'])

print(blk['Gender'].unique())
print("Data Frame after Label Encoding: \n")
print(blk)
```

```
Genuine Data Frame : 

  Gender     Name
0      F  Krishna
1      M    Rudra
2      M  Hrithik
3      F    Kiara
4      M    Dhoni
5      F     Sara
6      M   Rajesh
7      F     Lata
8      F   Anuska
9      M    Ratan
[0 1]
Data Frame after Label Encoding: 

   Gender     Name
0       0  Krishna
1       1    Rudra
2       1  Hrithik
3       0    Kiara
4       1    Dhoni
5       0     Sara
6       1   Rajesh
7       0     Lata
8       0   Anuska
9       1    Ratan
```

Label Encoding using Category Codes

[30]:
```python
import pandas as pd

my_data = {
    "Gender": ['F', 'M', 'M', 'F', 'M', 'F', 'M', 'F', 'F', 'M'],
    "Name" : ['Krishna', 'Rudra', 'Hrithik', 'Kiara', 'Dhoni', 'Sara', 'Rajesh', 'Lata', 'Anuska', 'Ratan']
}

blk = pd.DataFrame(my_data)
print(blk.dtypes)
```

```
Gender    object
Name      object
dtype: object
```

[32]:
```python
blk['Gender'] = blk['Gender'].astype('category')
print (blk.dtypes)
```

```
Gender    category
Name        object
dtype: object
```

[34]:
```python
import pandas as pd

my_data = {
    "Gender": ['F', 'M', 'M', 'F', 'M', 'F', 'M', 'F', 'F', 'M' ],
    "Name" : ['Krishna', 'Rudra', 'Hrithik', 'Kiara', 'Dhoni', 'Sara', 'Rajesh', 'Lata', 'Anuska', 'Ratan']
}

blk = pd.DataFrame(my_data)
print("Genuine Data Frame : \n")
print(blk)
blk['Gender'] = blk['Gender'].astype('category')
print("\n Data Frame after Label Encoding using Category codes : \n")
blk['Gender'] = blk['Gender'].cat.codes
print(blk)
```

```
Genuine Data Frame : 

  Gender     Name
0      F  Krishna
1      M    Rudra
2      M  Hrithik
3      F    Kiara
4      M    Dhoni
5      F     Sara
6      M   Rajesh
7      F     Lata
8      F   Anuska
9      M    Ratan

 Data Frame after Label Encoding using Category codes : 

   Gender     Name
0       0  Krishna
1       1    Rudra
2       1  Hrithik
3       0    Kiara
4       1    Dhoni
5       0     Sara
6       1   Rajesh
7       0     Lata
8       0   Anuska
9       1    Ratan
```

Linear Regression

[37]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[43]:
```python
advertising = pd.read_csv("advertising.csv")
print(advertising.head())
```

```
      TV  Radio  News Paper  Sales
0  230.1   37.8        69.2   22.1
1   44.5   39.3        45.1   10.4
2   17.2   45.9        69.3   12.0
3  151.5   41.3        58.5   16.5
4  180.8   10.8        58.4   17.9
```

[45]:
```python
print(advertising.info())
print("Datashape: ",advertising.shape)
```

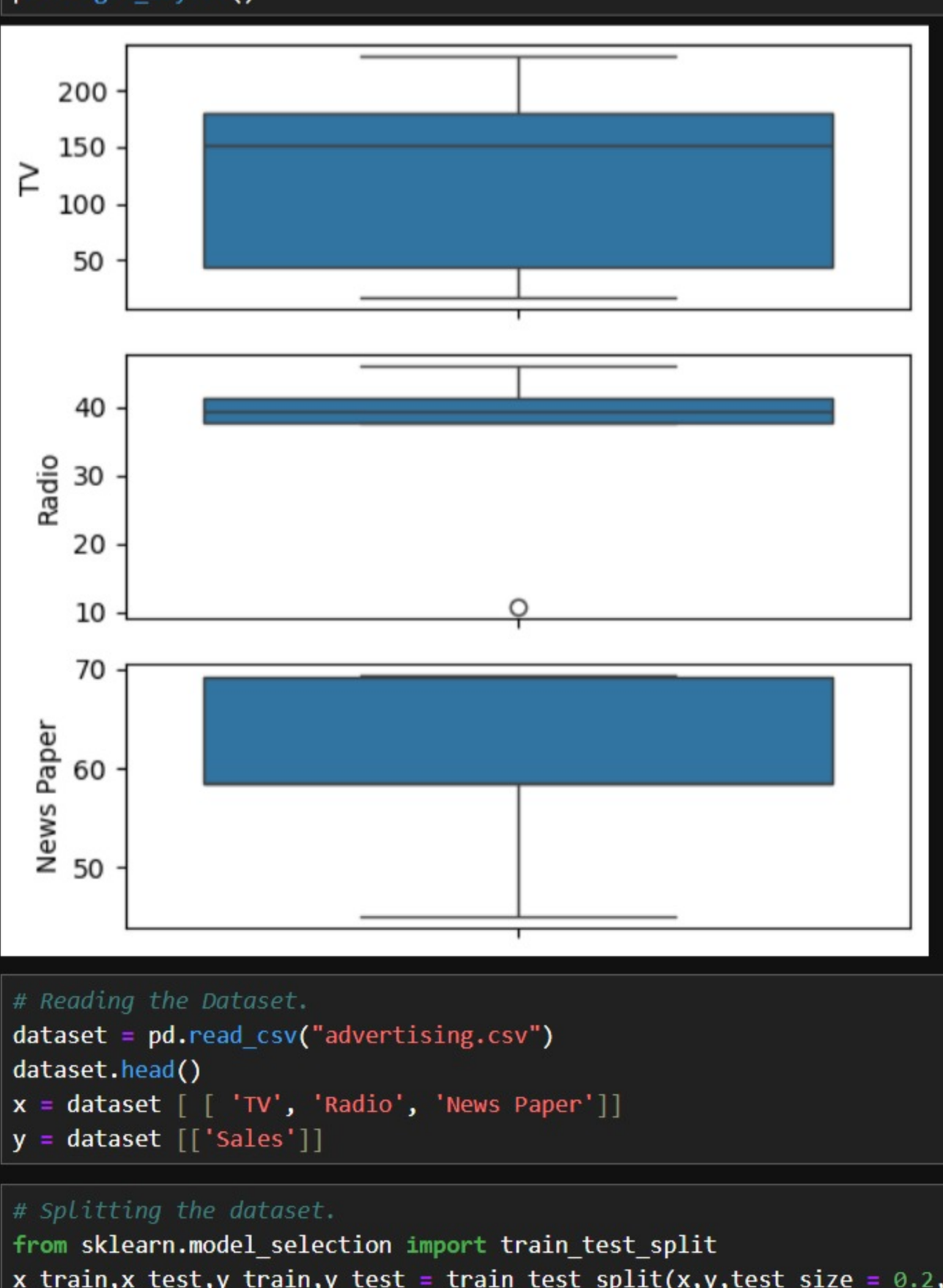
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   TV          5 non-null      float64
 1   Radio       5 non-null      float64
 2   News Paper  5 non-null      float64
 3   Sales       5 non-null      float64
dtypes: float64(4)
memory usage: 292.0 bytes
None
Datashape:  (5, 4)
```

[47]:
```python
# Check Null Values.
print("Checking for null values.")
print(advertising.isnull().sum()*100/advertising.shape[0])
```

```
Checking for null values.
TV            0.0
Radio         0.0
News Paper    0.0
Sales         0.0
dtype: float64
```

[49]:
```python
# Importing Library for Visualization.
# Outlier Analysis.

fig,axs = plt.subplots(3,figsize = (5,5))
plt1 = sns.boxplot(advertising['TV'],ax = axs[0])
plt2 = sns.boxplot(advertising['Radio'],ax = axs[1])
plt3 = sns.boxplot(advertising['News Paper'],ax = axs[2])
plt.tight_layout()
```



[51]:
```python
# Reading the Dataset.
dataset = pd.read_csv("advertising.csv")
dataset.head()
x = dataset [ [ 'TV', 'Radio', 'News Paper']]
y = dataset [['Sales']]
```

[53]:
```python
# Splitting the dataset.
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.2, random_state = 100)
```

[57]:
```python
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

dataset = pd.read_csv("advertising.csv")
dataset.head()
x = dataset[['TV', 'Radio', 'News Paper']]
y = dataset['Sales']
x_train,x_test,y_train, y_test = train_test_split(x,y,test_size = 0.2, random_state = 100)

print("TV Data: \n",x_train.head())
print(x_train.shape)
print(x_test.shape)
print("Sales Data: \n",y_train.head())
print(y_train.shape)
print(y_test.shape)

model = LinearRegression()
model.fit(x_train,y_train)
```

```
TV Data: 
       TV  Radio  News Paper
2   17.2   45.9        69.3
3  151.5   41.3        58.5
4  180.8   10.8        58.4
0  230.1   37.8        69.2
(4, 3)
(1, 3)
Sales Data: 
 2    12.0
3    16.5
4    17.9
0    22.1
Name: Sales, dtype: float64
(4,)
(1,)
```

[57]:
```
LinearRegression
LinearRegression()
```

[59]:
```python
y_pred = model.predict(x_test)
mse = mean_squared_error(y_test,y_pred)
r2 = r2_score(y_test,y_pred)
print("Mean Squared Error: ",mse)
print("R^2 Score : ",r2)
print("Coefficent: ",model.coef_)
print("Intercept : ",model.intercept_)
```

```
Mean Squared Error:  1.7278029673711812
R^2 Score :  nan
Coefficent:  [ 0.04748954 -0.00085181  0.17423743]
Intercept :  -0.8523758451967467
C:\Users\rajpu\anaconda3\Lib\site-packages\sklearn\metrics\_regression.py:1211: UndefinedMetricWarning: R^2 score is not well-defined with less than two sa
mples.
  warnings.warn(msg, UndefinedMetricWarning)
```

Case Study Implementation: Predicting House Prices.

[62]:
```python
# Step 1: Import Libraries.
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Step 2: Create the Dataset.
data = {
    'Size' : [1500, 1600, 1700, 1800, 1900, 2000],
    'Bedrooms' : [3, 3, 4, 4, 5, 5],
    'Age' : [10, 15, 20, 5, 8, 2],
    'Price' : [300000, 320000, 350000, 400000, 450000, 500000]
}

df = pd.DataFrame(data)

# Step 3: Preprocess the Data.
X = df[['Size', 'Bedrooms', 'Age']]
Y = df['Price']

# Step 4: Split the Data into Training and Testing Sets.
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = 0.2, random_state = 42)

# Step 5: Build and Train the Linear Regression Model.
model = LinearRegression()
model.fit(X_train,Y_train)

# Step 6: Make Predictions.
Y_pred = model.predict(X_test)

# Step 7: Evaluate the Model.
mse = mean_squared_error(Y_test,Y_pred)
r2 = r2_score(Y_test,Y_pred)

print("Mean Squared Error: ",mse)
print("R^2 Score : ",r2)

# Output the Coefficients.
print("Coefficients: ",model.coef_)
print("Intercept : ",model.intercept_)
```

```
Mean Squared Error:  1450000000.0000024
R^2 Score :  -13.500000000000023
Coefficients:  [ 5.00000000e+02  5.22568061e-11 -2.91065221e-12]
Intercept :  -499999.99999999965
```

[ ]: