PRACTICAL-6

AIM: Perform different binning techniques to smooth out the noise in the dataset. Make sure that the user should have the choice to apply all the possible techniques. Show the output of different bins. Use histogram to partition the dataset into groups.

Theory:

Binning:

Why is Binning Used?

Binning or discretization is used to transform a continuous or numerical variable into a categorical feature. Binning of continuous variables introduces non-linearity and tends to improve the performance of the model. It can also be used to identify missing values or outliers.

What is the Purpose of Binning Data?

Binning, also called discretization, is a technique for reducing continuous and discrete data cardinality. Binning groups related values together in bins to reduce the number of distinct values.

Example of Binning

Histograms are an example of data binning used to observe underlying distributions. They typically occur in one-dimensional space and equal intervals for ease of visualization.

Approaches to perform smoothing:

1. **Smoothing by bin means:**

➢ In smoothing by bin means, each value in a bin is replaced by the mean value of the bin.

2. Smoothing by bin median:

➢ In this method each bin value is replaced by its bin median value.

3. Smoothing by bin boundary:

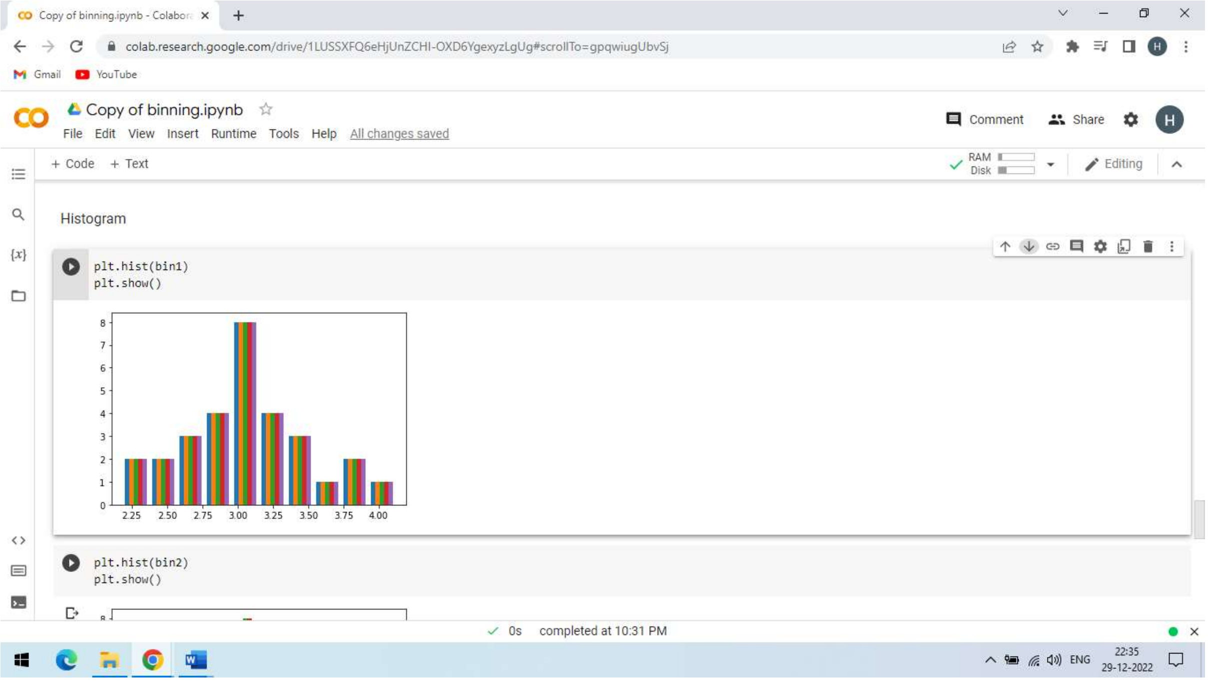
➢ In smoothing by bin boundaries, the minimum and maximum values in a given bin are identified as the bin boundaries. Each bin value is then replaced by the closest boundary value.

Code:

```
import numpy as np
import math
from sklearn.datasets import load_iris
from sklearn import datasets, linear_model, metrics
import matplotlib.pyplot as plt
# load iris data set
dataset = load_iris()
print(dataset)
a = dataset.data
b = np.zeros(150)
print(dataset)
# take 1st column among 4 column of data set
for i in range (150):
  b[i]=a[i,1]
#sort the array
b=np.sort(b)
```

```
print(b.size)
# create bins
bin1=np.zeros((30,5))
bin2=np.zeros((30,5))
bin3=np.zeros((30,5))
print(bin1.size)
# Bin mean
for i in range (0,150,5):
  k=int(i/5)
  mean=(b[i] + b[i+1] + b[i+2] + b[i+3] + b[i+4])/5
  for j in range(5):
    bin1[k,j]=mean
print("Bin Mean: \n",bin1)
# Bin boundaries
for i in range (0,150,5):
  k=int(i/5)
  for j in range (5):
    if (b[i+j]-b[i]) < (b[i+4]-b[i+j]):
      bin2[k,j]=b[i]
    else:
      bin2[k,j]=b[i+4]
print("\n")
print("Bin Boundaries: \n",bin2)
# Bin median
for i in range (0,150,5):
  k=int(i/5)
  for j in range (5):
   bin3[k,j]=b[i+2]
print("Bin Median: \n",bin3)
```

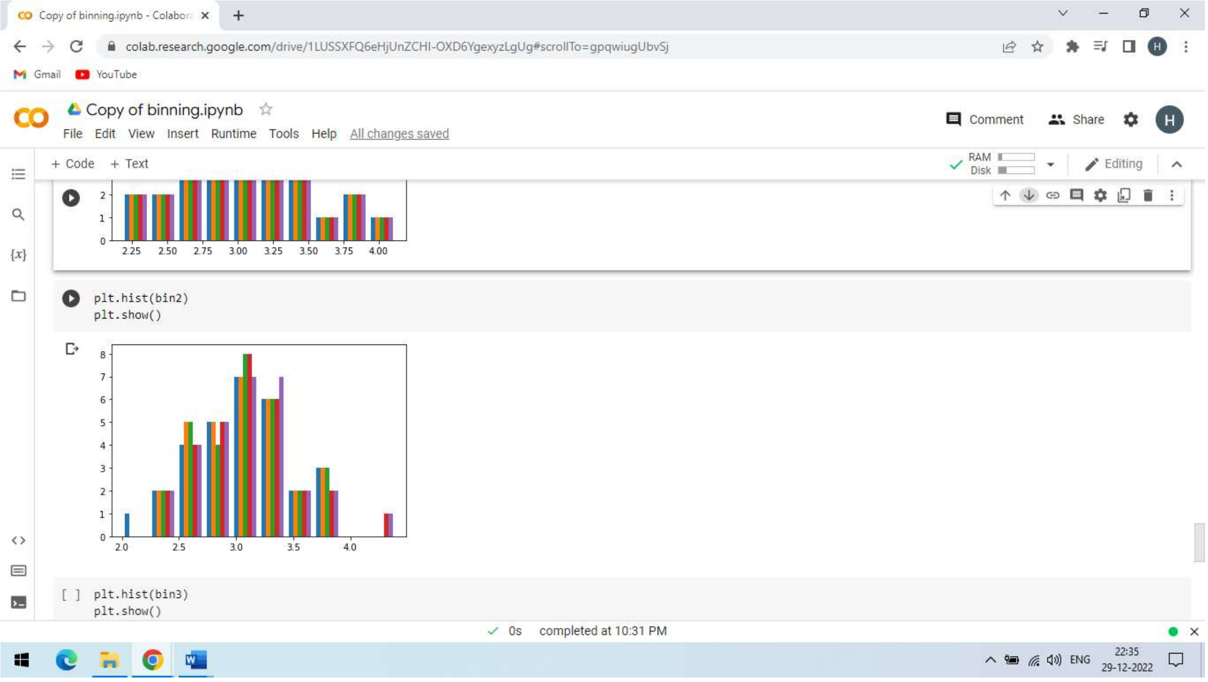
Output:


Copy of binning.ipynb - Colabor
colab.research.google.com/drive/1LUSSXFQ6eHjUnZCHI-OXD6YgexyzLgUg#scrollTo=gpqwiugUbvSj
Gmail
YouTube
Copy of binning.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
0
1
2
3
4
5
6
7
8
2.25
2.50
2.75
3.00
3.25
3.50
3.75
4.00
✓ 0s completed at 10:31 PM
ENG
22:35
29-12-2022

Histogram

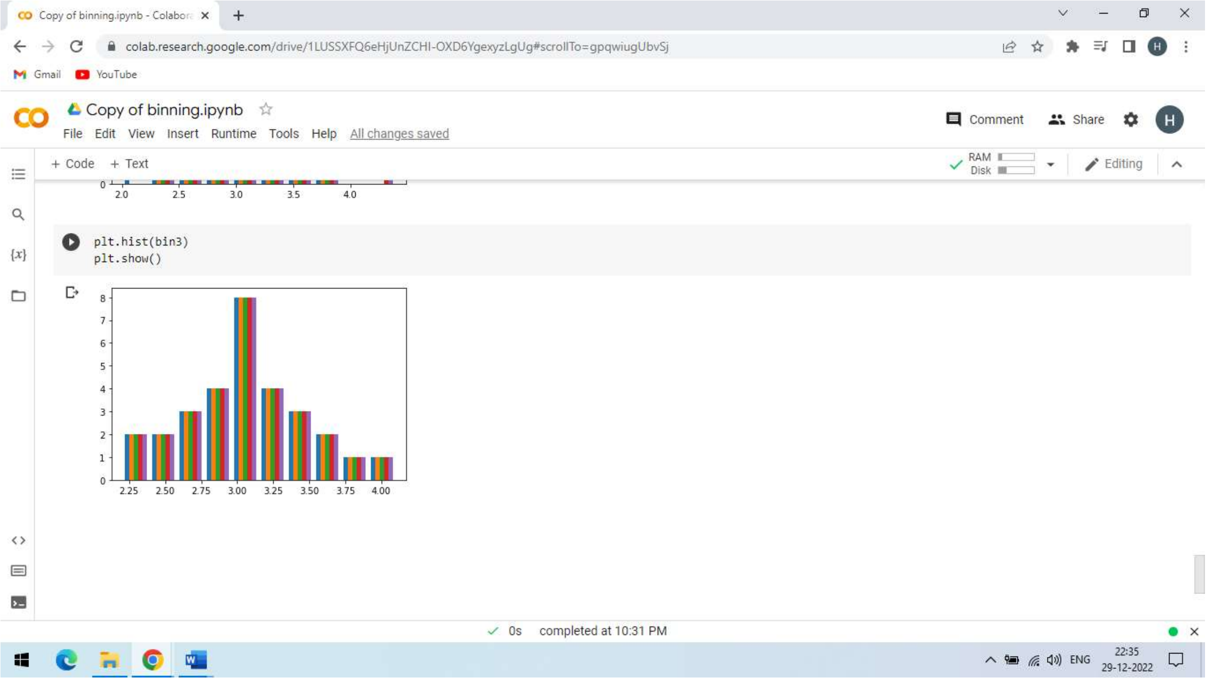
```
plt.hist(bin1)
plt.show()
```

```
plt.hist(bin2)
plt.show()
```


Copy of binning.ipynb - Colabor
colab.research.google.com/drive/1LUSSXFQ6eHjUnZCHI-OXD6YgexyzLgUg#scrollTo=gpqwiugUbvSj
Gmail
YouTube
Copy of binning.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
0
1
2
2.25
2.50
2.75
3.00
3.25
3.50
3.75
4.00
0
1
2
3
4
5
6
7
8
2.0
2.5
3.0
3.5
4.0
✓ 0s completed at 10:31 PM
ENG
22:35
29-12-2022

```
plt.hist(bin2)
plt.show()
```

```
[ ] plt.hist(bin3)
    plt.show()
```



```
plt.hist(bin3)
plt.show()
```

✓ 0s completed at 10:31 PM