- Course-BTech Type - AI Core-1
- Course Code - CSET211 Course Name - Statistical Machine Learning
- Year - Second Semester - ODD
- Date - 05/08/2024 Batch - 23 CSE 3rd Semester

Enrolment Number:- E23CSEU0677

- Name:- Pulkit Malik

Section 1: Numpy

```
#1
import numpy as np
print(np.__version__)
```

```
1.26.4
```

```
#2
array = np.arange(15)
odd_numbers = array[array % 2 != 0]
print(odd_numbers)
```

```
[ 1  3  5  7  9 11 13]
```

```
#3
array = np.arange(15)
boolean_array = array > 5
print(boolean_array)
```

```
[False False False False False False  True  True  True  True  True  True
  True  True  True]
```

```
#4
array = np.random.randint(1, 100, size=10)
max_value = np.max(array)
print("Array:", array)
print("Maximum value:", max_value)
```

```
Array: [ 2 91 51 91 69 93 49 54 17 66]
Maximum value: 93
```

```
#5
array = np.random.randint(1, 100, size=10)
min_index = np.argmin(array)
print("Array:", array)
print("Index of the minimum value:", min_index)
```

```
Array: [38 36 59 10 48 98 44 89 41 23]
Index of the minimum value: 3
```

```
#6
array_2d = np.random.rand(3, 3)
print("2D Array:")
print(array_2d)
```

```
2D Array:
[[0.06751134 0.98620527 0.83135756]
 [0.21726811 0.24508101 0.80419548]
 [0.63051281 0.86290523 0.54111448]]
```

```
#7
array_2d = np.zeros((4, 4))
np.fill_diagonal(array_2d, 5)
print("2D Array with diagonal filled with 5:")
print(array_2d)
```

```
2D Array with diagonal filled with 5:
[[5. 0. 0. 0.]
```

```
      [0. 5. 0. 0.]
      [0. 0. 5. 0.]
      [0. 0. 0. 5.]]
```

```
#8
array = np.array([[1, 2, 3, 4],
                  [5, 6, 7, 8],
                  [9, 10, 11, 12]])
row_sums = np.sum(array, axis=1)
print("Array:")
print(array)
print("\nSum of each row:")
print(row_sums)
```

```
Array:
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]

Sum of each row:
[10 26 42]
```

```
#9
array = np.array([[1, 2, 3, 4],
                  [5, 6, 7, 8],
                  [9, 10, 11, 12]])
flipped_array = np.fliplr(array)
print("Original Array:")
print(array)
print("\nHorizontally Flipped Array:")
print(flipped_array)
```

```
Original Array:
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]

Horizontally Flipped Array:
[[ 4  3  2  1]
 [ 8  7  6  5]
 [12 11 10  9]]
```

```
#10
array = np.array([[1, 2, 3, 4],
                  [5, 6, 7, 8],
                  [9, 10, 11, 12]])
cumulative_product = np.cumprod(array, axis=1)
print("Original Array:")
print(array)
print("\nCumulative Product of Each Row:")
print(cumulative_product)
```

```
Original Array:
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]

Cumulative Product of Each Row:
[[    1     2     6    24]
 [    5    30   210  1680]
 [    9    90   990 11880]]
```

```
#11
data = np.array([[1.0, 2.0, np.nan],
                 [4.0, np.nan, 6.0],
                 [7.0, 8.0, 9.0]])
print("Original Dataset:")
print(data)
column_means = np.nanmean(data, axis=0)
inds = np.where(np.isnan(data))
data[inds] = np.take(column_means, inds[1])
print("\nCleaned Dataset:")
print(data)
```

```
Original Dataset:
[[ 1.  2. nan]
 [ 4. nan  6.]
 [ 7.  8.  9.]]

Cleaned Dataset:
[[1.  2.  7.5]
 [4.  5.  6. ]
 [7.  8.  9. ]]
```

```
#12
data = np.array([['Alice', 'Engineer', 'New York'],
                 ['Bob', 'Doctor', 'Chicago'],
                 ['Charlie', 'Engineer', 'San Francisco'],
                 ['David', 'Artist', 'New York'],
                 ['Eve', 'Doctor', 'Chicago']])
text_column = data[:, 1]
unique_values = np.unique(text_column)
print("Unique values in the text column:")
print(unique_values)
```

```
Unique values in the text column:
['Artist' 'Doctor' 'Engineer']
```

Section 2 : Pandas

```
#1
import pandas as pd
url = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/master/diamonds.csv'
df = pd.read_csv(url)
print(df)
```

```
       carat        cut color clarity  depth  table  price     x     y     z
0       0.23      Ideal     E     SI2   61.5   55.0    326  3.95  3.98  2.43
1       0.21    Premium     E     SI1   59.8   61.0    326  3.89  3.84  2.31
2       0.23       Good     E     VS1   56.9   65.0    327  4.05  4.07  2.31
3       0.29    Premium     I     VS2   62.4   58.0    334  4.20  4.23  2.63
4       0.31       Good     J     SI2   63.3   58.0    335  4.34  4.35  2.75
...      ...        ...   ...     ...    ...    ...    ...   ...   ...   ...
53935   0.72      Ideal     D     SI1   60.8   57.0   2757  5.75  5.76  3.50
53936   0.72       Good     D     SI1   63.1   55.0   2757  5.69  5.75  3.61
53937   0.70  Very Good     D     SI1   62.8   60.0   2757  5.66  5.68  3.56
53938   0.86    Premium     H     SI2   61.0   58.0   2757  6.15  6.12  3.74
53939   0.75      Ideal     D     SI2   62.2   55.0   2757  5.83  5.87  3.64

[53940 rows x 10 columns]
```

```
#2

selected_columns = df[['carat', 'cut', 'price']]
print(selected_columns)
```

```
       carat        cut  price
0       0.23      Ideal    326
1       0.21    Premium    326
2       0.23       Good    327
3       0.29    Premium    334
4       0.31       Good    335
...      ...        ...    ...
53935   0.72      Ideal   2757
53936   0.72       Good   2757
53937   0.70  Very Good   2757
53938   0.86    Premium   2757
53939   0.75      Ideal   2757

[53940 rows x 3 columns]
```

```
#3
import pandas as pd
df.loc[:, 'Price-Quality'] = df['price'].astype(str) + '-' + df['clarity'].astype(str)
print(df['Price-Quality'])
```

```
0         326-SI2
1         326-SI1
2         327-VS1
3         334-VS2
4         335-SI2
           ...
53935    2757-SI1
53936    2757-SI1
53937    2757-SI1
53938    2757-SI2
53939    2757-SI2
Name: Price-Quality, Length: 53940, dtype: object
```

```
#4
unique_values = df.nunique()
print(unique_values)
```

```
carat              273
cut                  5
color                7
clarity              8
depth              184
table              127
price            11602
x                  554
y                  552
z                  375
Price-Quality    24827
dtype: int64
```

```
#5
numerical_summary = df.describe()
print(numerical_summary)
```

```
              carat         depth         table         price             x  \
count  53940.000000  53940.000000  53940.000000  53940.000000  53940.000000
mean       0.797940     61.749405     57.457184   3932.799722      5.731157
std        0.474011      1.432621      2.234491   3989.439738      1.121761
min        0.200000     43.000000     43.000000    326.000000      0.000000
25%        0.400000     61.000000     56.000000    950.000000      4.710000
50%        0.700000     61.800000     57.000000   2401.000000      5.700000
75%        1.040000     62.500000     59.000000   5324.250000      6.540000
max        5.010000     79.000000     95.000000  18823.000000     10.740000

                 y             z
count  53940.000000  53940.000000
mean       5.734526      3.538734
std        1.142135      0.705699
min        0.000000      0.000000
25%        4.720000      2.910000
50%        5.710000      3.530000
75%        6.540000      4.040000
max       58.900000     31.800000
```

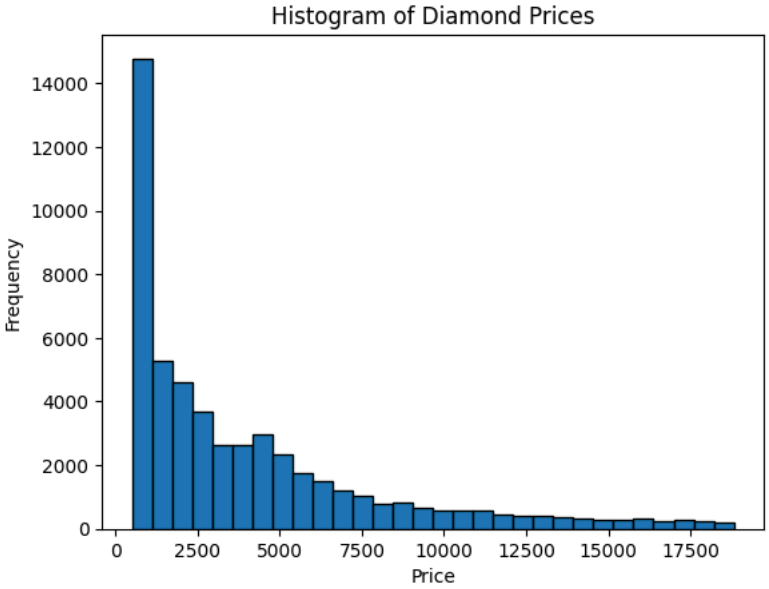
```
#6
df = df[df['price'] >= 500]
df.reset_index(drop=True, inplace=True)
print(df)
```

```
        carat        cut color clarity  depth  table  price     x     y     z  \
0        0.35      Ideal     I     VS1   60.9   57.0    552  4.54  4.59  2.78
1        0.30    Premium     D     SI1   62.6   59.0    552  4.23  4.27  2.66
2        0.30      Ideal     D     SI1   62.5   57.0    552  4.29  4.32  2.69
3        0.30      Ideal     D     SI1   62.1   56.0    552  4.30  4.33  2.68
4        0.42    Premium     I     SI2   61.5   59.0    552  4.78  4.84  2.96
...       ...        ...   ...     ...    ...    ...    ...   ...   ...   ...
52206    0.72      Ideal     D     SI1   60.8   57.0   2757  5.75  5.76  3.50
52207    0.72       Good     D     SI1   63.1   55.0   2757  5.69  5.75  3.61
52208    0.70  Very Good     D     SI1   62.8   60.0   2757  5.66  5.68  3.56
52209    0.86    Premium     H     SI2   61.0   58.0   2757  6.15  6.12  3.74
52210    0.75      Ideal     D     SI2   62.2   55.0   2757  5.83  5.87  3.64

      Price-Quality
0          552-VS1
1          552-SI1
2          552-SI1
3          552-SI1
4          552-SI2
...            ...
52206     2757-SI1
52207     2757-SI1
52208     2757-SI1
52209     2757-SI2
52210     2757-SI2

[52211 rows x 11 columns]
```

```
#7
import pandas as pd
import matplotlib.pyplot as plt
df['price'].plot(kind='hist', bins=30, edgecolor='black')
plt.title('Histogram of Diamond Prices')
plt.xlabel('Price')
plt.ylabel('Frequency')
plt.show()
```


Histogram of Diamond Prices
Frequency
Price
0
2000
4000
6000
8000
10000
12000
14000
0
2500
5000
7500
10000
12500
15000
17500