

**Sri Sivasubramaniya Nadar College of Engineering, Chennai**  
 (An autonomous Institution affiliated to Anna University)

|                     |                                       |                 |                      |
|---------------------|---------------------------------------|-----------------|----------------------|
| Degree & Branch     | B.E. Computer Science & Engineering   | Semester        | VI                   |
| Subject Code & Name | UCS2612 & Machine Learning Laboratory |                 |                      |
| Academic year       | 2025-2026 (Even)                      | Batch:2023-2027 | Due date: 27/01/2026 |
| Name:               | Nanda Kumar B                         | Roll No:        | 3122235001701        |

**Experiment 1 : Working with python packages-Numpy, Scipy, Scikit-learn, Matplotlib**

Aim: To study and explore the python library such as Pandas, Numpy, matplotlib, Scikit-learn, and Scipy to understand the data science and machine learning and understand how the different datasets are analyzed, and mapped to different machine learning models using exploratory data analysis technique.

**Libraries used:**

- **NumPy:** Used for numerical computations and efficient handling of multi-dimensional arrays.
- **Pandas:** Used for data manipulation, cleaning, and analysis using DataFrames.
- **Matplotlib:** Used for creating visualizations such as line graphs, bar charts, and histograms.
- **Seaborn:** Used for advanced statistical data visualization with attractive and informative plots.
- **Scikit-learn:** Used for implementing machine learning algorithms, model training, evaluation, preprocessing, and feature selection.
- **SciPy:** Used for scientific and technical computing, including optimization, and statistical functions.

#### 0.0.1 Loan Amount Prediction

```
[1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df = pd.read_csv("dataset/loan_train.csv")
df
```

```
[1]:      Customer ID          Name Gender  Age  Income (USD) \
0        C-36995  Frederica Shealy     F   56    1933.05
1        C-33999  America Calderone    M   32    4952.91
2        C-3770    Rosetta Verne     F   65    988.19
3        C-26480      Zoe Chitty     F   65       NaN
4        C-23459     Afton Venema     F   31    2614.77
...
29995    C-43723  Angelyn Clevenger    M   38    4969.41
```

|       |         |                   |   |    |         |
|-------|---------|-------------------|---|----|---------|
| 29996 | C-32511 | Silas Slaugh      | M | 20 | 1606.88 |
| 29997 | C-5192  | Carmelo Lone      | F | 49 | NaN     |
| 29998 | C-12172 | Carolann Osby     | M | 38 | 2417.71 |
| 29999 | C-33003 | Bridget Garibaldi | F | 63 | 3068.24 |

|       | Income Stability | Profession           | Type of Employment    | \              |
|-------|------------------|----------------------|-----------------------|----------------|
| 0     | Low              | Working              | Sales staff           |                |
| 1     | Low              | Working              |                       | NaN            |
| 2     | High             | Pensioner            |                       | NaN            |
| 3     | High             | Pensioner            |                       | NaN            |
| 4     | Low              | Working              | High skill tech staff |                |
| ...   | ...              | ...                  |                       | ...            |
| 29995 | Low              | Commercial associate |                       | Managers       |
| 29996 | Low              | Working              |                       | Laborers       |
| 29997 | Low              | Working              |                       | Sales staff    |
| 29998 | Low              | Working              |                       | Security staff |
| 29999 | High             | Pensioner            |                       | NaN            |

|       | Location   | Loan Amount Request (USD) | ... | Credit Score | \   |
|-------|------------|---------------------------|-----|--------------|-----|
| 0     | Semi-Urban | 72809.58                  | ... | 809.44       |     |
| 1     | Semi-Urban | 46837.47                  | ... | 780.40       |     |
| 2     | Semi-Urban | 45593.04                  | ... | 833.15       |     |
| 3     | Rural      | 80057.92                  | ... | 832.70       |     |
| 4     | Semi-Urban | 113858.89                 | ... | 745.55       |     |
| ...   | ...        | ...                       | ... | ...          | ... |
| 29995 | Urban      | 76657.90                  | ... | 869.61       |     |
| 29996 | Semi-Urban | 66595.14                  | ... | 729.41       |     |
| 29997 | Urban      | 81410.08                  | ... | NaN          |     |
| 29998 | Semi-Urban | 142524.10                 | ... | 677.27       |     |
| 29999 | Rural      | 156290.54                 | ... | 815.44       |     |

|       | No. of Defaults | Has Active Credit Card | Property ID | Property Age | \   |
|-------|-----------------|------------------------|-------------|--------------|-----|
| 0     | 0               | NaN                    | 746         | 1933.05      |     |
| 1     | 0               | Unpossessed            | 608         | 4952.91      |     |
| 2     | 0               | Unpossessed            | 546         | 988.19       |     |
| 3     | 1               | Unpossessed            | 890         | NaN          |     |
| 4     | 1               | Active                 | 715         | 2614.77      |     |
| ...   | ...             | ...                    | ...         | ...          | ... |
| 29995 | 0               | Unpossessed            | 566         | 4969.41      |     |
| 29996 | 0               | Inactive               | 175         | 1606.88      |     |
| 29997 | 0               | Active                 | 959         | NaN          |     |
| 29998 | 1               | Unpossessed            | 375         | 2417.71      |     |
| 29999 | 0               | Active                 | 344         | 3068.24      |     |

|   | Property Type | Property Location | Co-Applicant | Property Price | \ |
|---|---------------|-------------------|--------------|----------------|---|
| 0 | 4             | Rural             | 1            | 119933.46      |   |
| 1 | 2             | Rural             | 1            | 54791.00       |   |

|       |     |            |     |           |
|-------|-----|------------|-----|-----------|
| 2     | 2   | Urban      | 0   | 72440.58  |
| 3     | 2   | Semi-Urban | 1   | 121441.51 |
| 4     | 4   | Semi-Urban | 1   | 208567.91 |
| ...   | ... | ...        | ... | ...       |
| 29995 | 4   | Urban      | 1   | 111096.56 |
| 29996 | 3   | Urban      | 1   | 73453.94  |
| 29997 | 1   | Rural      | 1   | 102108.02 |
| 29998 | 4   | Urban      | 1   | 168194.47 |
| 29999 | 3   | Rural      | 1   | 194512.60 |

Loan Sanction Amount (USD)

|       |           |
|-------|-----------|
| 0     | 54607.18  |
| 1     | 37469.98  |
| 2     | 36474.43  |
| 3     | 56040.54  |
| 4     | 74008.28  |
| ...   | ...       |
| 29995 | 68992.11  |
| 29996 | 46616.60  |
| 29997 | 61057.56  |
| 29998 | 99766.87  |
| 29999 | 117217.90 |

[30000 rows x 24 columns]

```
[2]: print("Info \n", df.info())
df.describe().T
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 24 columns):
 #   Column           Non-Null Count  Dtype  
 --- 
 0   Customer ID      30000 non-null   object 
 1   Name              30000 non-null   object 
 2   Gender            29947 non-null   object 
 3   Age               30000 non-null   int64  
 4   Income (USD)     25424 non-null   float64
 5   Income Stability 28317 non-null   object 
 6   Profession        30000 non-null   object 
 7   Type of Employment 22730 non-null   object 
 8   Location           30000 non-null   object 
 9   Loan Amount Request (USD) 30000 non-null   float64
 10  Current Loan Expenses (USD) 29828 non-null   float64
 11  Expense Type 1      30000 non-null   object 
 12  Expense Type 2      30000 non-null   object 
 13  Dependents         27507 non-null   float64
 14  Credit Score       28297 non-null   float64
```

```

15 No. of Defaults           30000 non-null int64
16 Has Active Credit Card   28434 non-null object
17 Property ID               30000 non-null int64
18 Property Age              25150 non-null float64
19 Property Type              30000 non-null int64
20 Property Location          29644 non-null object
21 Co-Applicant              30000 non-null int64
22 Property Price             30000 non-null float64
23 Loan Sanction Amount (USD) 29660 non-null float64
dtypes: float64(8), int64(5), object(11)
memory usage: 5.5+ MB

```

Info

None

|                             | count      | mean          | std          | min        | \ |
|-----------------------------|------------|---------------|--------------|------------|---|
| Age                         | 30000.0    | 40.092300     | 16.045129    | 18.00      |   |
| Income (USD)                | 25424.0    | 2630.574417   | 11262.723830 | 377.70     |   |
| Loan Amount Request (USD)   | 30000.0    | 88826.333855  | 59536.949605 | 6048.24    |   |
| Current Loan Expenses (USD) | 29828.0    | 400.936876    | 242.545375   | -999.00    |   |
| Dependents                  | 27507.0    | 2.253027      | 0.951162     | 1.00       |   |
| Credit Score                | 28297.0    | 739.885381    | 72.163846    | 580.00     |   |
| No. of Defaults             | 30000.0    | 0.193933      | 0.395384     | 0.00       |   |
| Property ID                 | 30000.0    | 501.934700    | 288.158086   | 1.00       |   |
| Property Age                | 25150.0    | 2631.119440   | 11322.677000 | 377.70     |   |
| Property Type               | 30000.0    | 2.460067      | 1.118562     | 1.00       |   |
| Co-Applicant                | 30000.0    | -4.743867     | 74.614593    | -999.00    |   |
| Property Price              | 30000.0    | 131759.680252 | 93549.548104 | -999.00    |   |
| Loan Sanction Amount (USD)  | 29660.0    | 47649.342208  | 48221.146686 | -999.00    |   |
|                             | 25%        | 50%           | 75%          | max        |   |
| Age                         | 25.0000    | 40.000        | 55.0000      | 65.00      |   |
| Income (USD)                | 1650.4575  | 2222.435      | 3090.5925    | 1777460.21 |   |
| Loan Amount Request (USD)   | 41177.7550 | 75128.075     | 119964.6050  | 621497.82  |   |
| Current Loan Expenses (USD) | 247.6675   | 375.205       | 521.2925     | 3840.88    |   |
| Dependents                  | 2.0000     | 2.000         | 3.0000       | 14.00      |   |
| Credit Score                | 681.8800   | 739.820       | 799.1200     | 896.26     |   |
| No. of Defaults             | 0.0000     | 0.000         | 0.0000       | 1.00       |   |
| Property ID                 | 251.0000   | 504.000       | 751.0000     | 999.00     |   |
| Property Age                | 1650.4500  | 2223.250      | 3091.4075    | 1777460.21 |   |
| Property Type               | 1.0000     | 2.000         | 3.0000       | 4.00       |   |
| Co-Applicant                | 1.0000     | 1.000         | 1.0000       | 1.00       |   |
| Property Price              | 60572.1600 | 109993.610    | 178880.7200  | 1077966.73 |   |
| Loan Sanction Amount (USD)  | 0.0000     | 35209.395     | 74261.2500   | 481907.32  |   |

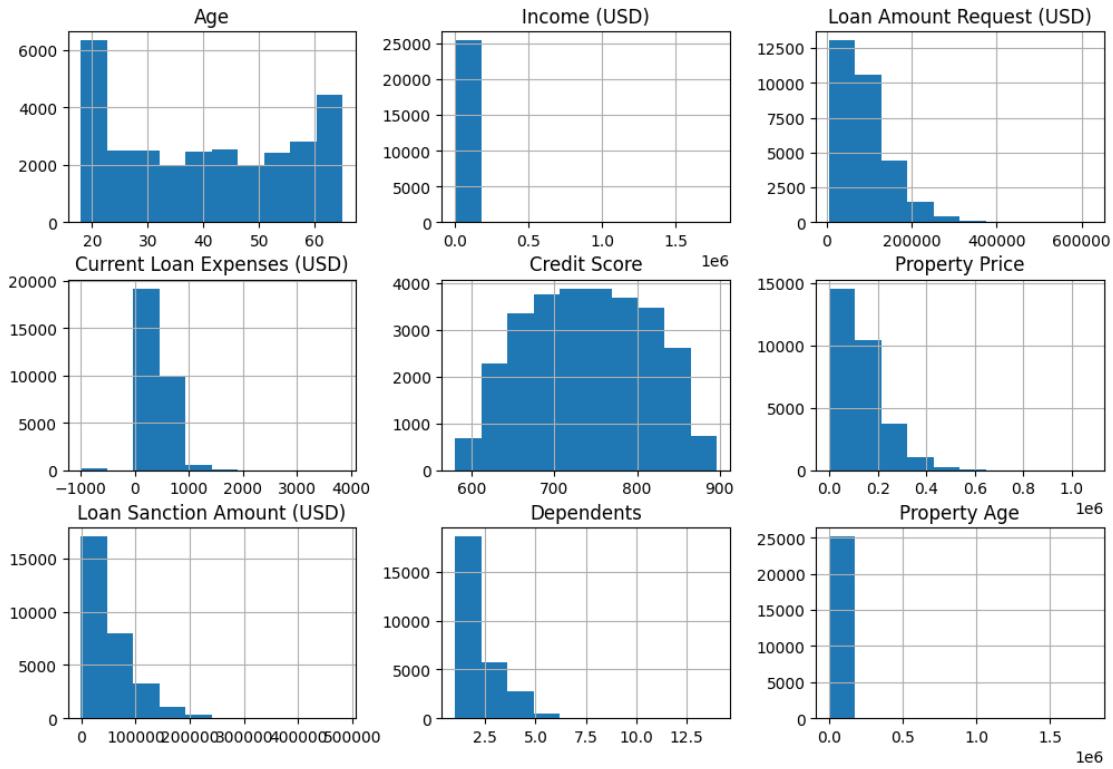
[3]: df.isnull().sum()

```
[3]: Customer ID          0
      Name                0
      Gender              53
      Age                 0
      Income (USD)        4576
      Income Stability    1683
      Profession          0
      Type of Employment   7270
      Location             0
      Loan Amount Request (USD) 0
      Current Loan Expenses (USD) 172
      Expense Type 1       0
      Expense Type 2       0
      Dependents           2493
      Credit Score          1703
      No. of Defaults      0
      Has Active Credit Card 1566
      Property ID           0
      Property Age          4850
      Property Type          0
      Property Location      356
      Co-Applicant           0
      Property Price          0
      Loan Sanction Amount (USD) 340
      dtype: int64
```

### Exploratory Data Analysis

```
[4]: numcols = ['Age', 'Income (USD)', 'Loan Amount Request (USD)',  
             'Current Loan Expenses (USD)', 'Credit Score',  
             'Property Price', 'Loan Sanction Amount (USD)', 'Dependents', 'Property Age']  
  
df[numcols].hist(figsize=(12,8))  
plt.suptitle("Histograms of Numerical Features")  
plt.show()
```

Histograms of Numerical Features



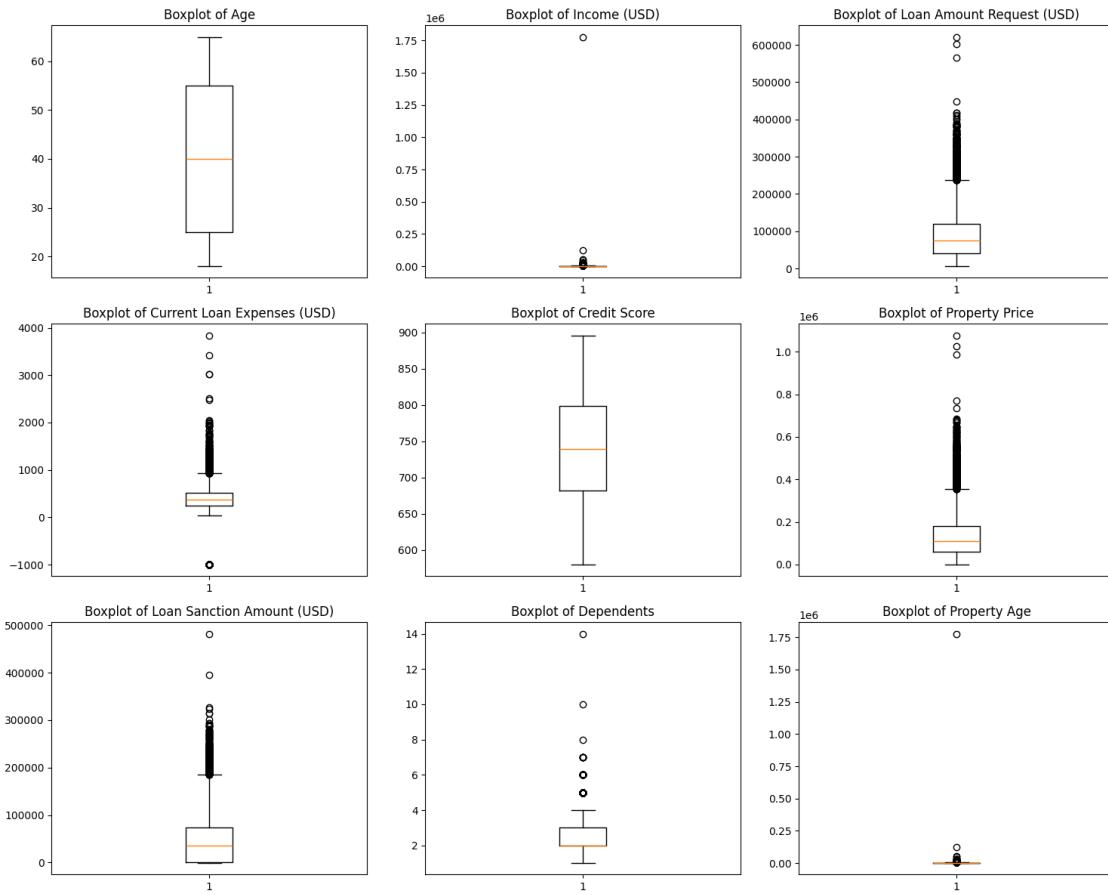
```
[5]: import matplotlib.pyplot as plt

num_cols = [
    'Age', 'Income (USD)', 'Loan Amount Request (USD)',
    'Current Loan Expenses (USD)', 'Credit Score',
    'Property Price', 'Loan Sanction Amount (USD)',
    'Dependents', 'Property Age'
]

fig, axes = plt.subplots(3, 3, figsize=(15, 12)) # 3x3 grid
axes = axes.flatten() # flatten to easily loop

for ax, col in zip(axes, num_cols):
    ax.boxplot(df[col].dropna())
    ax.set_title(f"Boxplot of {col}")

plt.tight_layout()
plt.show()
```



```
[6]: fig, axes = plt.subplots(2, 3, figsize=(24, 8))
axes = axes.flatten()

# Histogram
axes[0].hist(df["Loan Sanction Amount (USD)"], bins=25)
axes[0].set_title("Loan Sanction Amount Distribution")
axes[0].set_xlabel("Loan Sanction Amount")
axes[0].set_ylabel("Frequency")

axes[1].scatter(df["Loan Amount Request (USD)"], df["Loan Sanction Amount (USD)"])
axes[1].set_xlabel("Loan Amount Request (USD)")
axes[1].set_ylabel("Loan Sanction Amount (USD)")
axes[1].set_title("Loan Request vs Loan Sanction")

df["Gender"].value_counts().plot(kind = 'bar', ax = axes[2])
axes[2].set_title("Gender Distribution")
axes[2].set_xlabel("Gender")
axes[2].set_ylabel("Count")
```

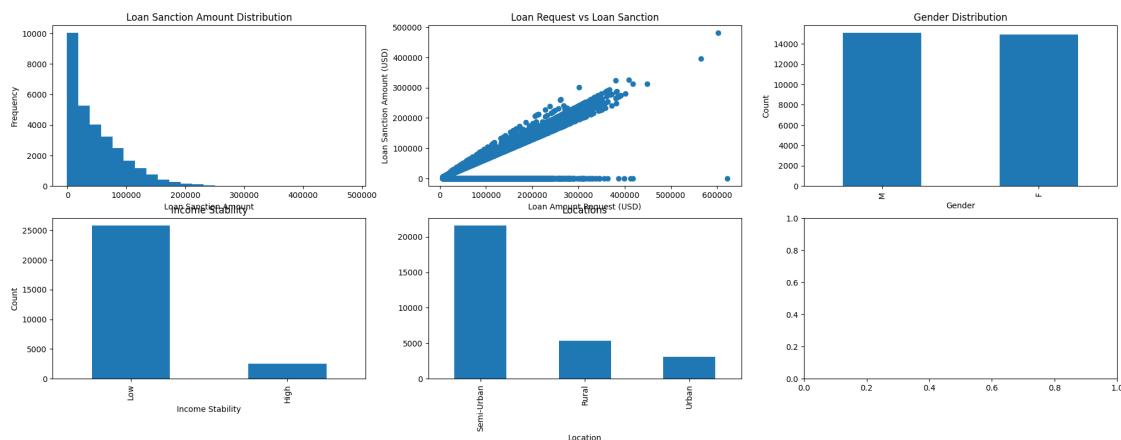
```

df[["Income Stability"]].value_counts().plot(kind = 'bar', ax = axes[3])
axes[3].set_title("Income Stability")
axes[3].set_xlabel("Income Stability")
axes[3].set_ylabel("Count")

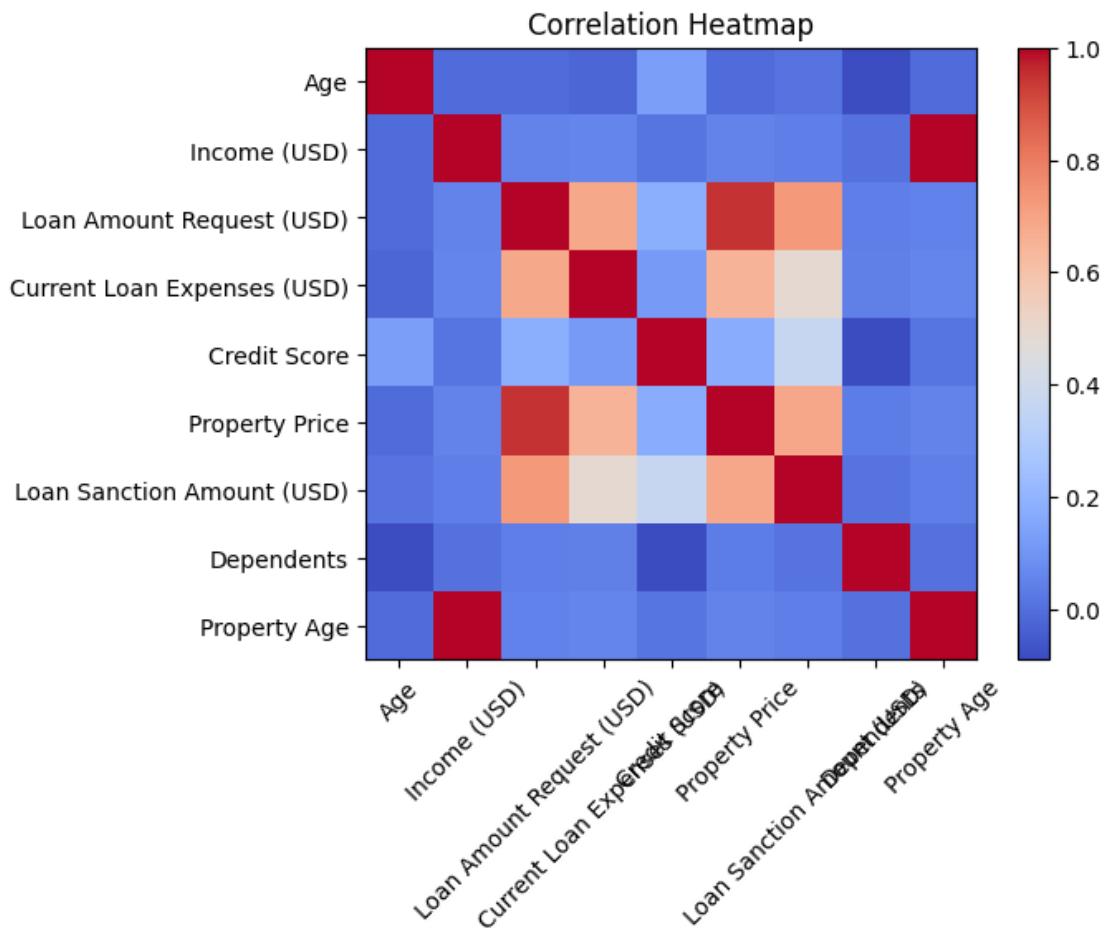
df[["Location"]].value_counts().plot(kind = 'bar', ax = axes[4])
axes[4].set_title("Locations")
axes[4].set_ylabel("Count")

```

[6]: Text(0, 0.5, 'Count')



[7]: corr = df[numcols].corr()
plt.imshow(corr, cmap='coolwarm')
plt.colorbar()
plt.xticks(range(len(corr.columns)), corr.columns, rotation=45)
plt.yticks(range(len(corr.columns)), corr.columns)
plt.title("Correlation Heatmap")
plt.show()



### 0.0.2 Iris

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

df=pd.read_csv('dataset/Iris.csv')
df.head()
```

|   | Id | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm | Species     |
|---|----|---------------|--------------|---------------|--------------|-------------|
| 0 | 1  | 5.1           | 3.5          | 1.4           | 0.2          | Iris-setosa |
| 1 | 2  | 4.9           | 3.0          | 1.4           | 0.2          | Iris-setosa |
| 2 | 3  | 4.7           | 3.2          | 1.3           | 0.2          | Iris-setosa |
| 3 | 4  | 4.6           | 3.1          | 1.5           | 0.2          | Iris-setosa |
| 4 | 5  | 5.0           | 3.6          | 1.4           | 0.2          | Iris-setosa |

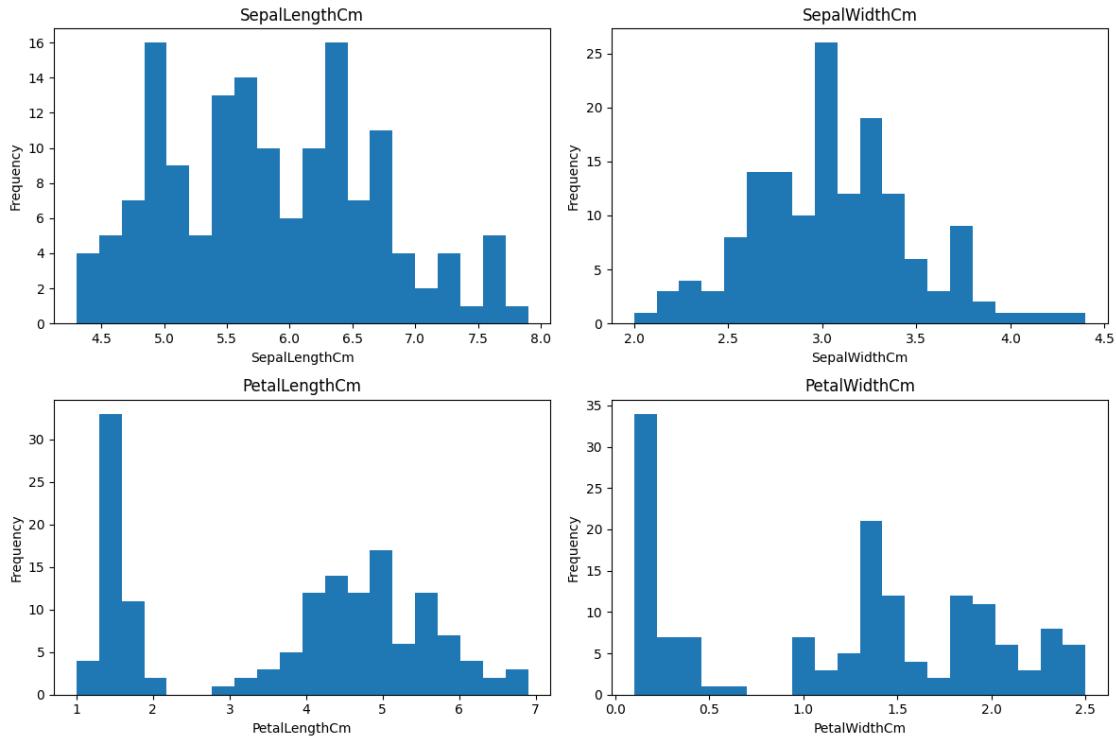
```
[9]: df.describe().T
```

```
[9]:
```

|               | count | mean      | std       | min | 25%   | 50%   | 75%    | max   |
|---------------|-------|-----------|-----------|-----|-------|-------|--------|-------|
| Id            | 150.0 | 75.500000 | 43.445368 | 1.0 | 38.25 | 75.50 | 112.75 | 150.0 |
| SepalLengthCm | 150.0 | 5.843333  | 0.828066  | 4.3 | 5.10  | 5.80  | 6.40   | 7.9   |
| SepalWidthCm  | 150.0 | 3.054000  | 0.433594  | 2.0 | 2.80  | 3.00  | 3.30   | 4.4   |
| PetalLengthCm | 150.0 | 3.758667  | 1.764420  | 1.0 | 1.60  | 4.35  | 5.10   | 6.9   |
| PetalWidthCm  | 150.0 | 1.198667  | 0.763161  | 0.1 | 0.30  | 1.30  | 1.80   | 2.5   |

```
[10]:
```

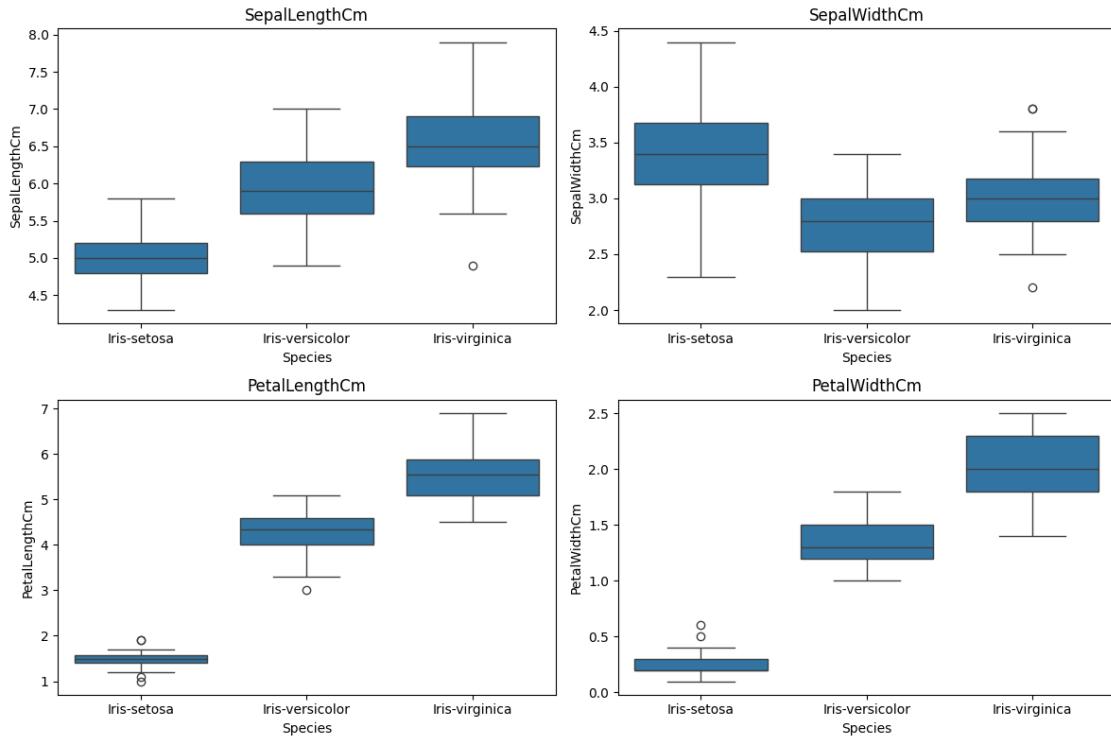
```
features = [  
    'SepalLengthCm',  
    'SepalWidthCm',  
    'PetalLengthCm',  
    'PetalWidthCm'  
]  
  
plt.figure(figsize=(12, 8))  
  
for i, col in enumerate(features, 1):  
    plt.subplot(2, 2, i)  
    plt.hist(df[col], bins=20)  
    plt.title(col)  
    plt.xlabel(col)  
    plt.ylabel('Frequency')  
  
plt.tight_layout()  
plt.show()
```



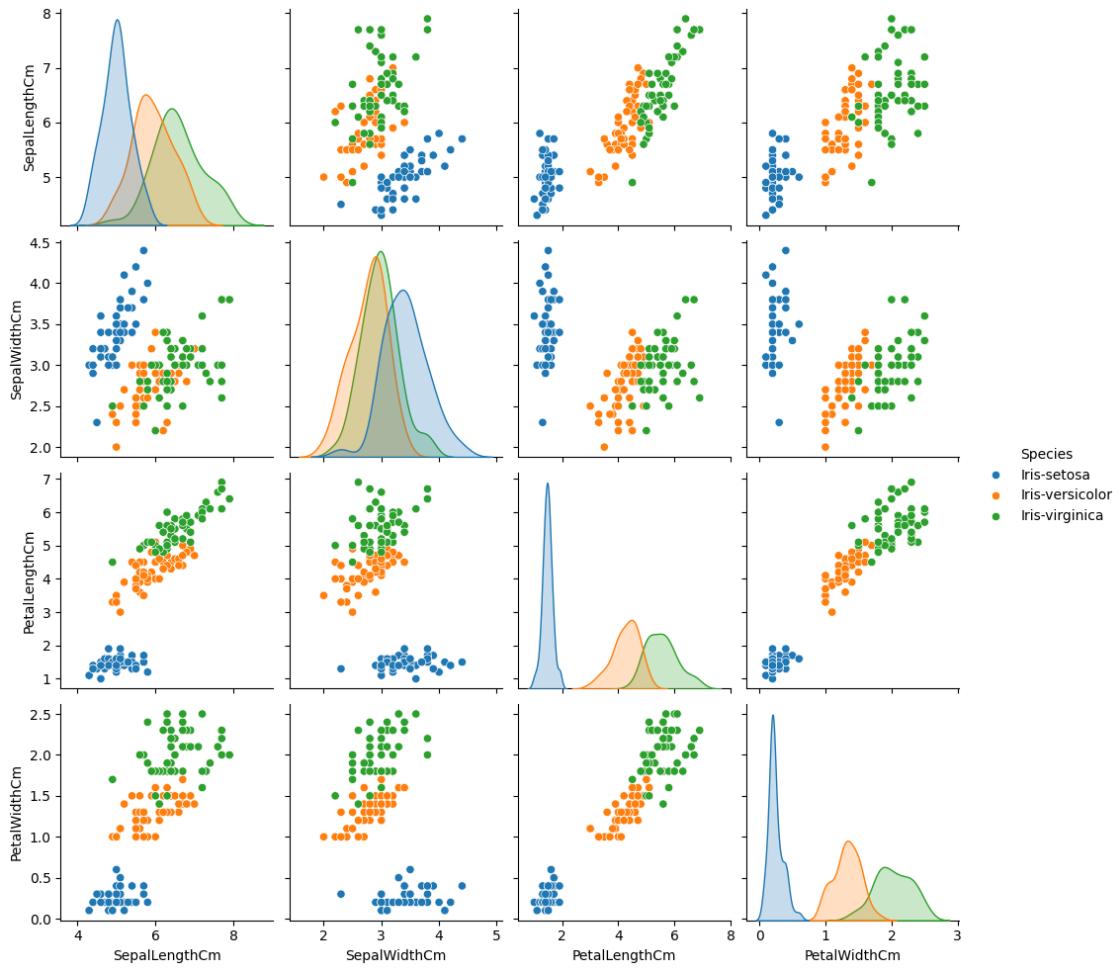
```
[11]: plt.figure(figsize=(12, 8))

for i, col in enumerate(features, 1):
    plt.subplot(2, 2, i)
    sns.boxplot(x='Species', y=col, data=df)
    plt.title(col)

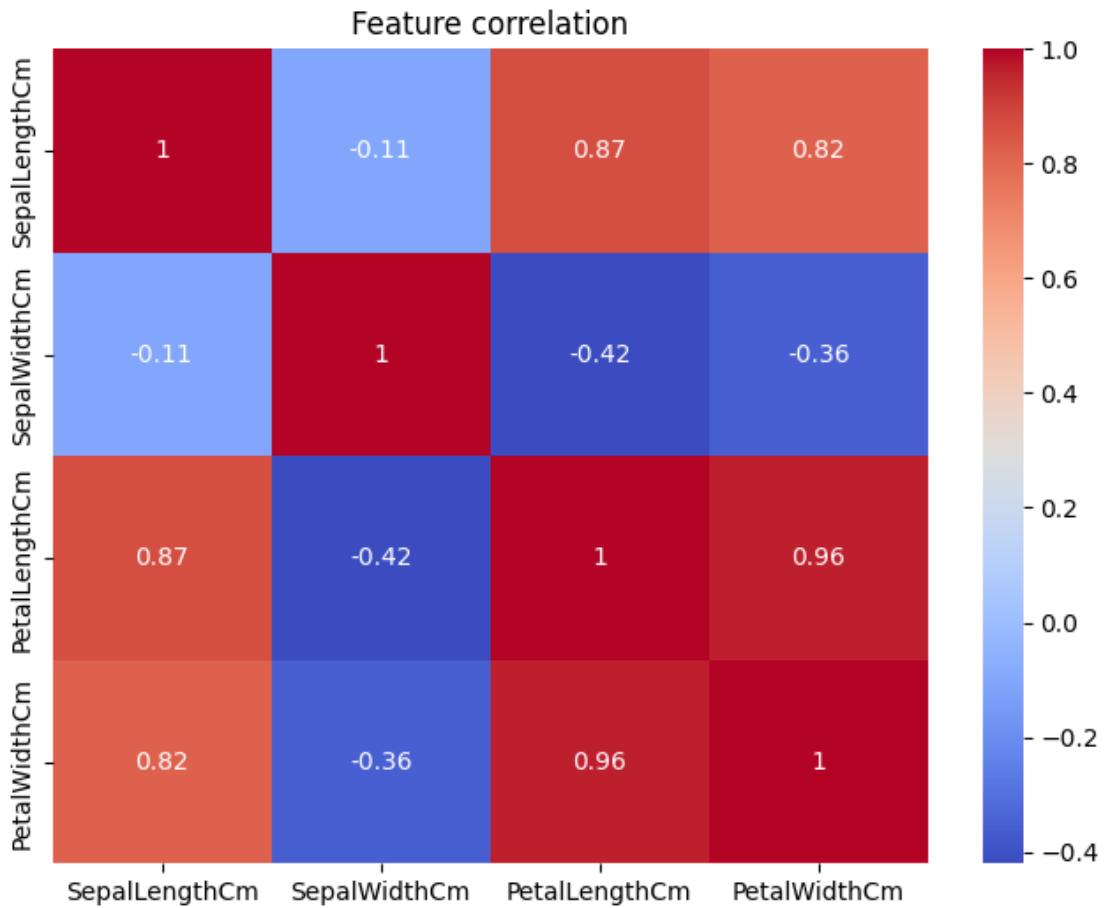
plt.tight_layout()
plt.show()
```



```
[12]: sns.pairplot(df.drop('Id', axis = 1), hue='Species')
plt.show()
```



```
[13]: plt.figure(figsize=(8,6))
sns.heatmap(df.drop(['Species', 'Id'], axis=1).corr(),
            annot=True, cmap='coolwarm')
plt.title("Feature correlation")
plt.show()
```



### 0.0.3 Hand Written Character Recognition

```
[45]: import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os
import math

df = pd.read_csv("datasets/english.csv")
```

```
[46]: print("Shape:", df.shape)
print("Info\n", df.info())
print("Head\n", df.head())

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

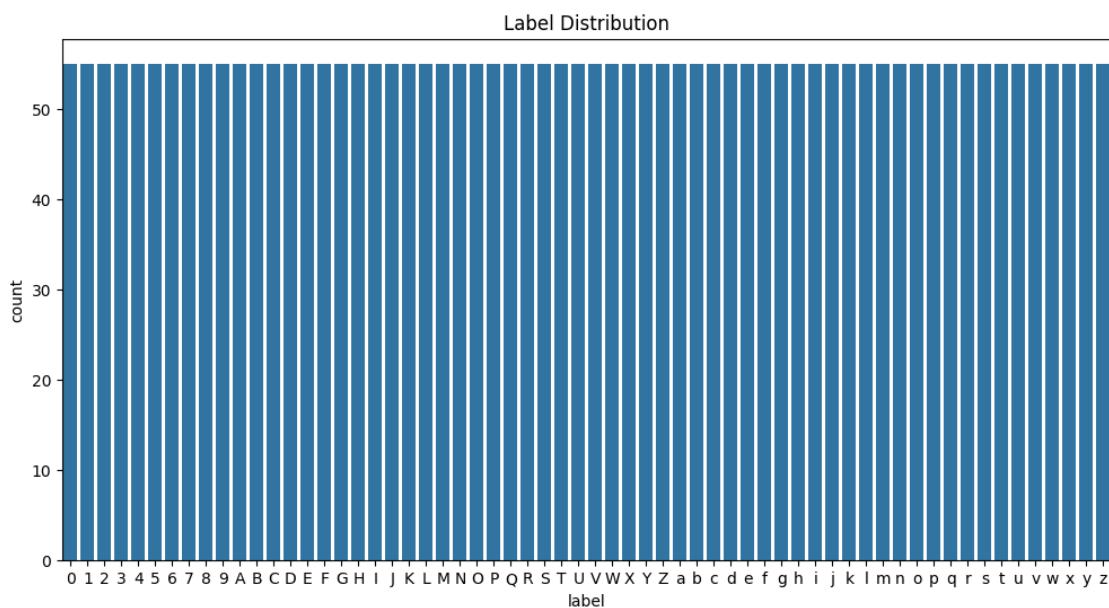
Shape: (3410, 2)  
<class 'pandas.core.frame.DataFrame'>

```

RangeIndex: 3410 entries, 0 to 3409
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
--- 
 0   image    3410 non-null   object  
 1   label    3410 non-null   object  
dtypes: object(2)
memory usage: 53.4+ KB
Info
None
Head
      image  label
0  Img/img001-001.png  0
1  Img/img001-002.png  0
2  Img/img001-003.png  0
3  Img/img001-004.png  0
4  Img/img001-005.png  0
image      0
label      0
dtype: int64

```

```
[47]: plt.figure(figsize=(12,6))
sns.countplot(data=df, x="label", order=df["label"].value_counts().index)
plt.title("Label Distribution")
plt.show()
```



```
[48]: from PIL import Image

plt.figure(figsize=(10, 6))

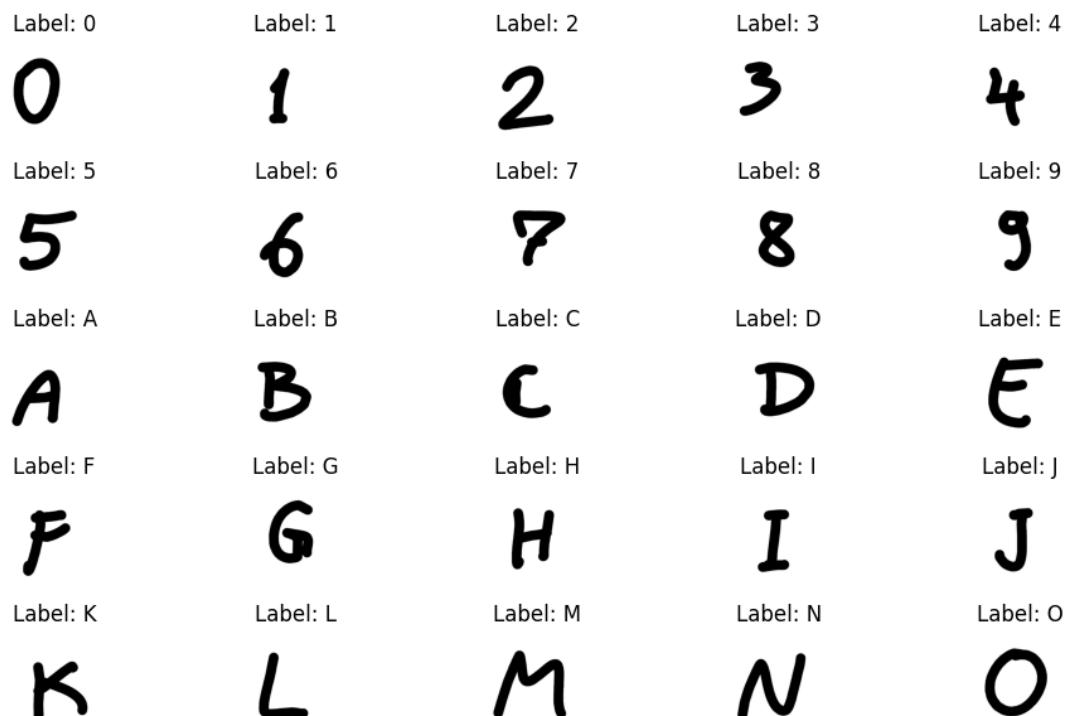
unique = df.groupby('label').first().reset_index().head(25)

for i, row in unique.iterrows():
    img_path = row['image']
    label = row['label']

    img = Image.open(f"datasets/{img_path}")

    plt.subplot(5, 5, i+1)
    plt.imshow(img, cmap='gray')
    plt.title(f"Label: {label}")
    plt.axis('off')

plt.tight_layout()
plt.show()
```



```
[49]: from PIL import Image

heights = []
```

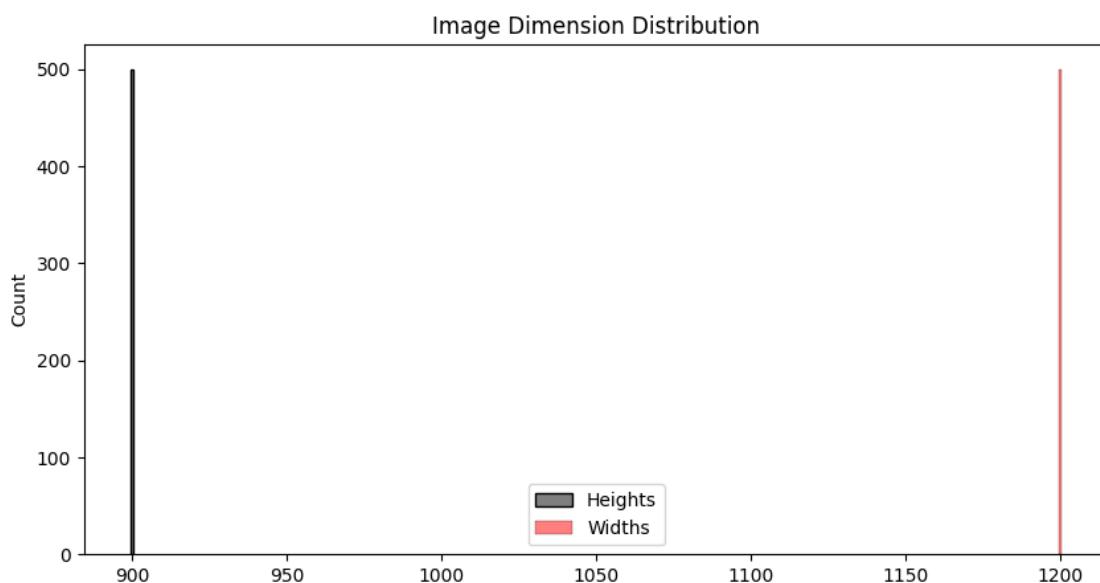
```

widths = []

for img_path in df['image'][:500]:
    img = Image.open(f"datasets/{img_path}")
    w, h = img.size
    widths.append(w)
    heights.append(h)

plt.figure(figsize=(10,5))
sns.histplot(heights, kde=True, color="black", label="Heights")
sns.histplot(widths, kde=True, color="red", label="Widths")
plt.legend()
plt.title("Image Dimension Distribution")
plt.show()

```



```
[14]: import pandas as pd

df = pd.read_csv("dataset/diabetes.csv")
```

```
[17]: print("Shape:", df.shape)
print("Info\n", df.info())
df.head()
```

```

Shape: (768, 9)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
```

```
#   Column           Non-Null Count   Dtype  
---  --  
0   Pregnancies      768 non-null    int64  
1   Glucose          768 non-null    int64  
2   BloodPressure    768 non-null    int64  
3   SkinThickness    768 non-null    int64  
4   Insulin          768 non-null    int64  
5   BMI              768 non-null    float64 
6   DiabetesPedigreeFunction 768 non-null    float64 
7   Age              768 non-null    int64  
8   Outcome          768 non-null    int64  
dtypes: float64(2), int64(7)
```

memory usage: 54.1 KB

Info

None

```
[17]:   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin  BMI  \ 
0            6        148            72            35            0  33.6
1            1         85            66            29            0  26.6
2            8        183            64             0            0  23.3
3            1         89            66            23            94  28.1
4            0        137            40            35           168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                  0.627    50       1
1                  0.351    31       0
2                  0.672    32       1
3                  0.167    21       0
4                  2.288    33       1
```

```
[18]: df.isnull().sum()
```

```
Pregnancies      0
Glucose          0
BloodPressure    0
SkinThickness    0
Insulin          0
BMI              0
DiabetesPedigreeFunction 0
Age              0
Outcome          0
dtype: int64
```

```
[19]: df.describe().T
```

```
count      mean       std      min     25%  \
Pregnancies 768.0  3.845052  3.369578  0.000  1.000000
Glucose      768.0 120.894531 31.972618  0.000 99.000000
```

|                          |       |           |            |        |          |
|--------------------------|-------|-----------|------------|--------|----------|
| BloodPressure            | 768.0 | 69.105469 | 19.355807  | 0.000  | 62.00000 |
| SkinThickness            | 768.0 | 20.536458 | 15.952218  | 0.000  | 0.00000  |
| Insulin                  | 768.0 | 79.799479 | 115.244002 | 0.000  | 0.00000  |
| BMI                      | 768.0 | 31.992578 | 7.884160   | 0.000  | 27.30000 |
| DiabetesPedigreeFunction | 768.0 | 0.471876  | 0.331329   | 0.078  | 0.24375  |
| Age                      | 768.0 | 33.240885 | 11.760232  | 21.000 | 24.00000 |
| Outcome                  | 768.0 | 0.348958  | 0.476951   | 0.000  | 0.00000  |

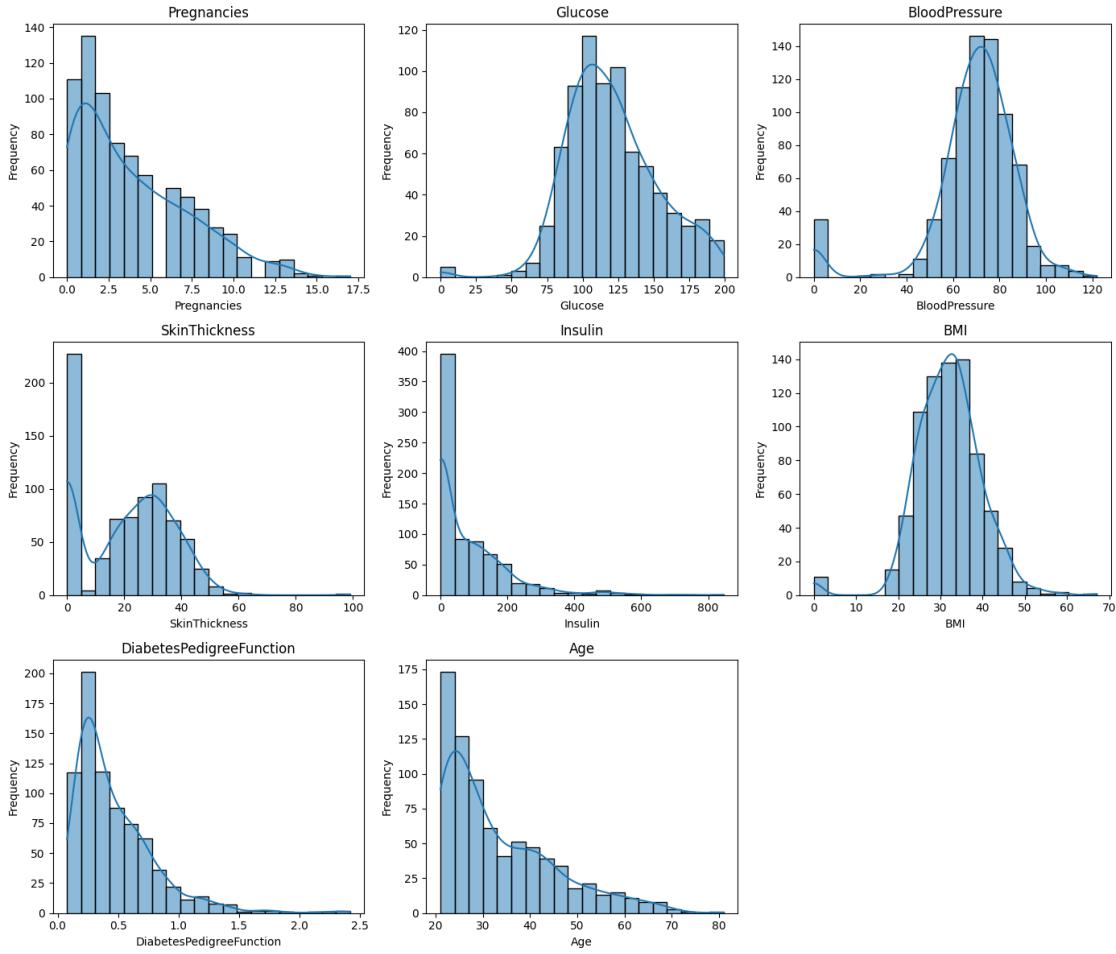
|                          | 50%      | 75%       | max    |
|--------------------------|----------|-----------|--------|
| Pregnancies              | 3.0000   | 6.00000   | 17.00  |
| Glucose                  | 117.0000 | 140.25000 | 199.00 |
| BloodPressure            | 72.0000  | 80.00000  | 122.00 |
| SkinThickness            | 23.0000  | 32.00000  | 99.00  |
| Insulin                  | 30.5000  | 127.25000 | 846.00 |
| BMI                      | 32.0000  | 36.60000  | 67.10  |
| DiabetesPedigreeFunction | 0.3725   | 0.62625   | 2.42   |
| Age                      | 29.0000  | 41.00000  | 81.00  |
| Outcome                  | 0.0000   | 1.00000   | 1.00   |

```
[20]: features = df.drop('Outcome', axis=1).columns

plt.figure(figsize=(14, 12))

for i, col in enumerate(features, 1):
    plt.subplot(3, 3, i)
    sns.histplot(df[col], bins=20, kde=True)
    plt.title(col)
    plt.xlabel(col)
    plt.ylabel("Frequency")

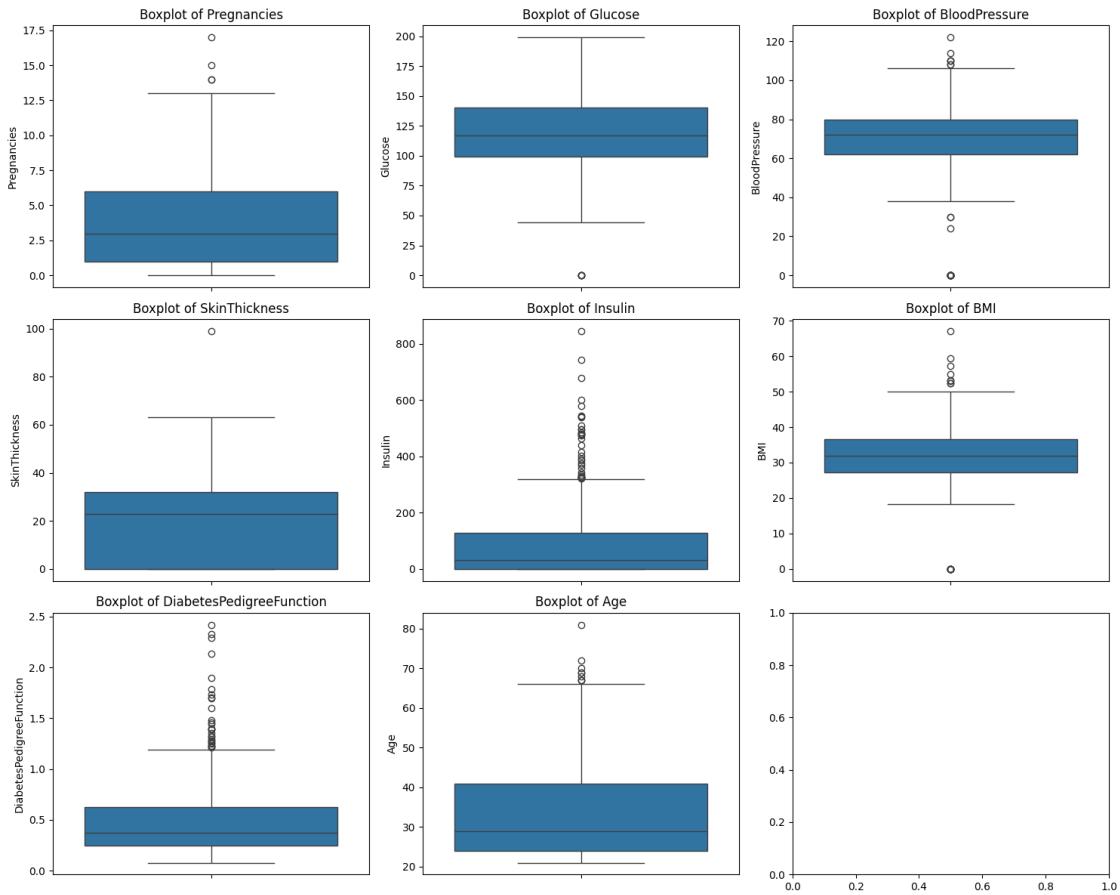
plt.tight_layout()
plt.show()
```



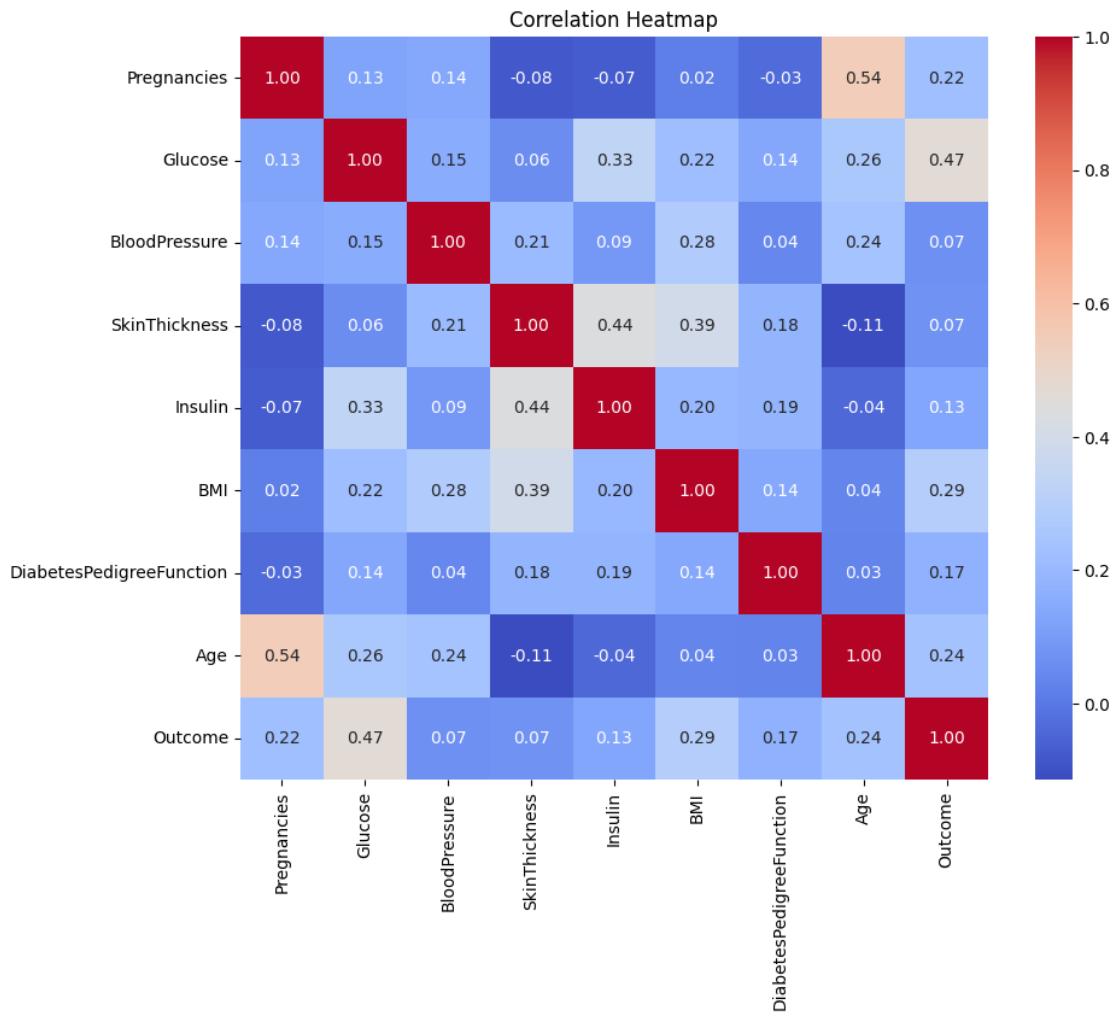
```
[21]: fig, axes = plt.subplots(3, 3, figsize=(15, 12))
axes = axes.flatten()

for i, col in enumerate(features):
    sns.boxplot(y=df[col], ax=axes[i])
    axes[i].set_title(f'Boxplot of {col}')

plt.tight_layout()
plt.show()
```



```
[22]: plt.figure(figsize=(10,8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Heatmap")
plt.show()
```



#### 0.0.4 Email Classification

```
[23]: import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

df = pd.read_csv("dataset/email.csv")
df.head()
```

```
[23]:   Category                         Message
0    ham  Go until jurong point, crazy.. Available only ...
1    ham                      Ok lar... Joking wif u oni...
2   spam  Free entry in 2 a wkly comp to win FA Cup fina...
3    ham  U dun say so early hor... U c already then say...
4    ham  Nah I don't think he goes to usf, he lives aro...
```

```
[24]: print("Info\n", df.info())
df.describe().T
```

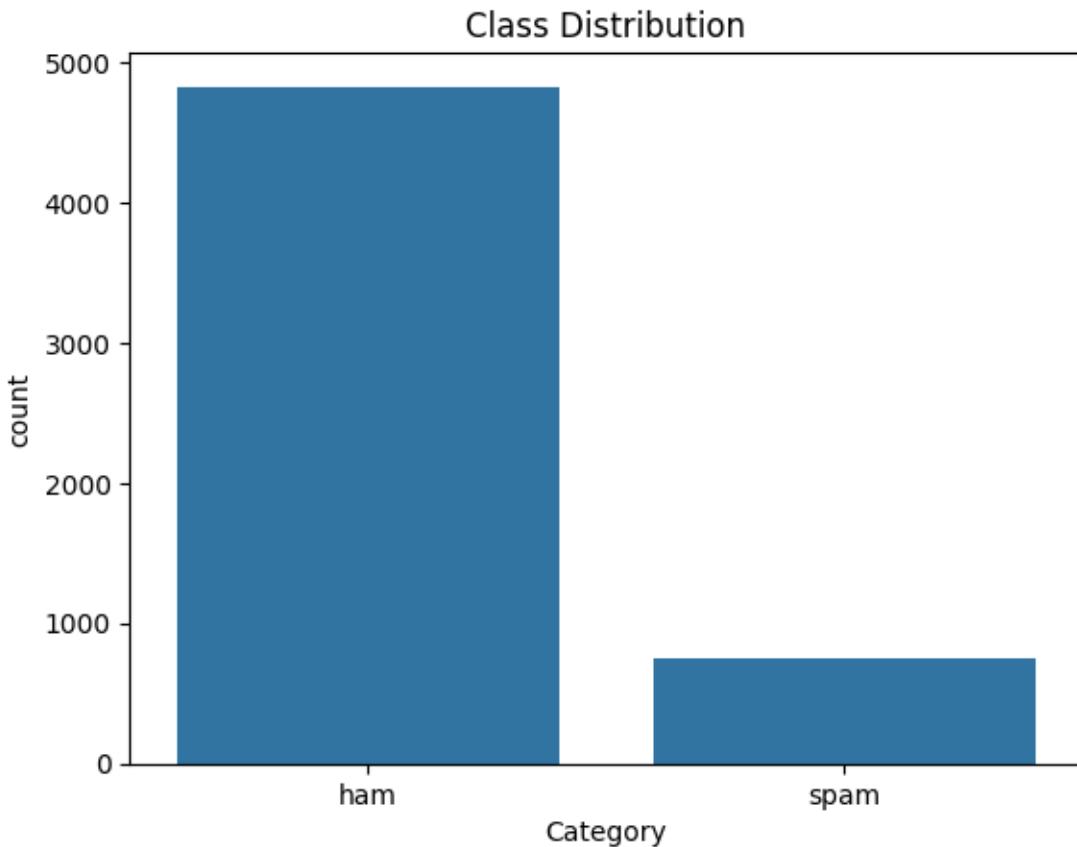
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype  
---  --          -----          ---  
 0   Category    5572 non-null   object  
 1   Message     5572 non-null   object  
dtypes: object(2)
memory usage: 87.2+ KB
Info
None
```

```
[24]:      count unique           top freq
Category  5572      2             ham 4825
Message   5572  5157 Sorry, I'll call later 30
```

```
[25]: print("Null Values\n", df.isnull().sum())
print("Counts of spam and Ham \n", df['Category'].value_counts())
```

```
Null Values
Category      0
Message       0
dtype: int64
Counts of spam and Ham
Category
ham        4825
spam       747
Name: count, dtype: int64
```

```
[26]: sns.countplot(x='Category', data=df)
plt.title("Class Distribution")
plt.show()
```



```
[27]: import re

def clean_text(text):
    text = text.lower()
    text = re.sub(r"http\S+", "", text)
    text = re.sub(r"[^a-zA-Z\s]", "", text)
    return text

df['clean_text'] = df['Message'].apply(clean_text)
```

```
[28]: df['word_count'] = df['clean_text'].apply(lambda x: len(x.split()))
df['char_count'] = df['clean_text'].apply(len)
```

```
[29]: df.head()
```

```
[29]:   Category          Message \
0      ham  Go until jurong point, crazy.. Available only ...
1      ham                  Ok lar... Joking wif u oni...
2     spam  Free entry in 2 a wkly comp to win FA Cup fina...
3      ham  U dun say so early hor... U c already then say...
```

```
4      ham Nah I don't think he goes to usf, he lives aro...
```

|   | clean_text  | word_count | char_count |
|---|---|------------|------------|
| 0 | go until jurong point crazy available only in ... | 20         | 102        |
| 1 | ok lar joking wif u oni                           | 6          | 23         |
| 2 | free entry in a wkly comp to win fa cup final...  | 25         | 124        |
| 3 | u dun say so early hor u c already then say       | 11         | 43         |
| 4 | nah i dont think he goes to usf he lives aroun... | 13         | 59         |

```
[30]: from collections import Counter

all_words = " ".join(df['clean_text']).split()
common_words = Counter(all_words).most_common(20)
common_words
```

```
[30]: [('i', 2249),
 ('to', 2248),
 ('you', 2128),
 ('a', 1448),
 ('the', 1331),
 ('u', 1154),
 ('and', 971),
 ('is', 892),
 ('in', 889),
 ('me', 791),
 ('my', 759),
 ('for', 711),
 ('your', 677),
 ('it', 622),
 ('of', 622),
 ('call', 578),
 ('have', 573),
 ('on', 536),
 ('that', 514),
 ('are', 490)]
```

```
[31]: spam_words = " ".join(df[df['Category']=="spam"]['clean_text']).split()
ham_words = " ".join(df[df['Category']=="ham"]['clean_text']).split()

print("Spam words most common \n\n", Counter(spam_words).most_common(15))
print("\nNot Spam common words \n", Counter(ham_words).most_common(15))
```

```
Spam words most common
```

```
[('to', 686), ('a', 380), ('call', 347), ('you', 287), ('your', 263), ('free',
219), ('for', 204), ('the', 201), ('now', 190), ('or', 188), ('is', 157), ('u',
153), ('txt', 151), ('on', 145), ('ur', 144)]
```

Not Spam common words

```
[('i', 2195), ('you', 1841), ('to', 1562), ('the', 1130), ('a', 1068), ('u', 1001), ('and', 849), ('in', 817), ('me', 761), ('my', 748), ('is', 735), ('it', 594), ('of', 525), ('for', 507), ('that', 488)]
```

## Results and Discussions:

| Dataset                           | Type of ML Task      | Suitable ML Algorithm  |
|-----------------------------------|----------------------|--|
| Iris Dataset                      | Classification       | K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Decision Tree |
| Loan Amount Prediction            | Regression           | Linear Regression, Decision Tree Regressor, Random Forest Regressor    |
| Diabetes Prediction               | Classification       | Logistic Regression, Support Vector Machine, Random Forest             |
| Email Spam Detection              | Text Classification  | Naive Bayes, Support Vector Machine, Logistic Regression               |
| Handwritten Character Recognition | Image Classification | Convolutional Neural Network (CNN), Support Vector Machine             |

## Learning Practices

- Understood the fundamentals of data analysis using Python libraries such as NumPy, Pandas, Matplotlib, and Seaborn.
- Learnt to analyze datasets using descriptive statistics and exploratory data analysis techniques.
- Learnt to visualize data distributions and relationships using appropriate plots and charts.
- Learnt to identify suitable machine learning tasks such as classification and regression for different datasets.