

```
from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive
```

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

```
df = pd.read_csv('/content/drive/MyDrive/adjusted/Iris.csv')
```

df.head()

	5.1	3.5	1.4	0.2	0
0	4.9	3.0	1.4	0.2	0
1	4.7	3.2	1.3	0.2	0
2	4.6	3.1	1.5	0.2	0
3	5.0	3.6	1.4	0.2	0
4	5.4	3.9	1.7	0.4	0

Next steps: [Generate code with df](#) [New interactive sheet](#)

df.dtypes

	0
5.1	float64
3.5	float64
1.4	float64
0.2	float64
0	int64

dtype: object

df.columns

Index(['5.1', '3.5', '1.4', '0.2', '0'], dtype='object')
--

df.isnull().sum()

	0
5.1	0
3.5	0
1.4	0
0.2	0
0	0

dtype: int64

```
missing_percentage = df.isnull().mean() * 100
threshold = 30
columns_to_drop = missing_percentage[missing_percentage > threshold].index
df = df.drop(columns=columns_to_drop)
print(f"Columns dropped: {list(columns_to_drop)}")
print(f"Remaining columns: {df.columns}")

Columns dropped: []
Remaining columns: Index(['5.1', '3.5', '1.4', '0.2', '0'], dtype='object')
```

```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
for column in df.select_dtypes(include=['object']).columns:
    df[column] = label_encoder.fit_transform(df[column])
df.head()
```

	5.1	3.5	1.4	0.2	0
0	4.9	3.0	1.4	0.2	0
1	4.7	3.2	1.3	0.2	0
2	4.6	3.1	1.5	0.2	0
3	5.0	3.6	1.4	0.2	0
4	5.4	3.9	1.7	0.4	0

Next steps: [Generate code with df](#) [New interactive sheet](#)

```
for column in df.select_dtypes(include=['float64']).columns:
    mean_value = df[column].mean()
    df[column] = df[column].fillna(mean_value)
```

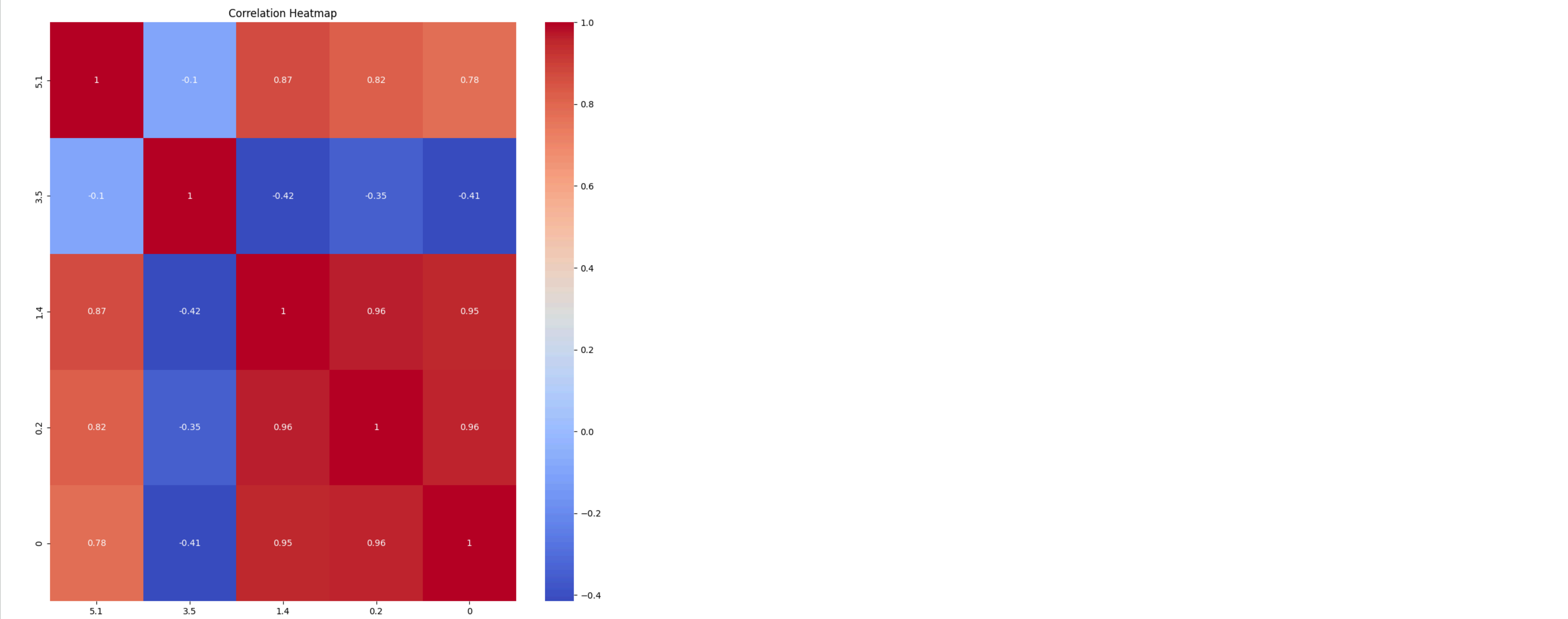
```
for column in df.select_dtypes(include=['object']).columns:
    mode_value = df[column].mode()[0]
    df[column] = df[column].fillna(mode_value)
```

df.isnull().sum()

	0
5.1	0
3.5	0
1.4	0
0.2	0
0	0

dtype: int64

```
plt.figure(figsize=(12, 12))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



```
df['0'].value_counts()
```

count	
0	
1	50
2	50
0	49

dtype: int64

```
df['0'].value_counts().plot(kind='bar', color=[ 'purple', 'pink','red'])
plt.xticks([0, 1, 2], ['Setosa (0)', 'Versicolor (1)', 'virginica(2)'], rotation=0)
plt.title('Iris Species Count')
plt.xlabel('Species')
plt.ylabel('Count')
plt.grid(axis='y')
plt.show()
```



```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(6, 4))

sns.barplot(
    x='0',
    y='5.1',
    data=df,
    estimator='mean'
)

plt.title("Average Sepal Length by Iris Class")
plt.xlabel("Class")
plt.ylabel("Average Sepal Length")
plt.grid(axis='y')

plt.show()
```



```
cat_cols = [col for col in df.columns if df[col].dtype == 'object']
num_cols = [col for col in df.columns if df[col].dtype != 'object']

print("Categorical columns:", cat_cols)
print("Numeric columns:", num_cols)

for col in cat_cols:
    print(f"{col} has {df[col].unique()} values")

Categorical columns: []
Numeric columns: ['5.1', '3.5', '1.4', '0.2', '0']
```

```
plt.figure(figsize=(8, 6))
sns.histplot(df['5.1'], bins=15, kde=True)
plt.title('Distribution of Feature 5.1')
plt.xlabel('5.1 Value')
plt.ylabel('Frequency')
plt.show()
```



```
plt.figure(figsize=(6, 4))

sns.barplot(
    x='0',
    y='1.4',
    data=df,
    estimator='mean'
)

plt.title("Average of Feature 1.4 by Class")
plt.xlabel("Class")
plt.ylabel("Average 1.4 Value")
plt.grid(axis='y')

plt.show()
```



```
df.describe()
```

	5.1	3.5	1.4	0.2	0	
count	149.000000	149.000000	149.000000	149.000000	149.000000	
mean	5.848322	3.051007	3.774497	1.205369	1.006711	
std	0.828594	0.433499	1.759651	0.761292	0.817847	
min	4.300000	2.000000	1.000000	0.100000	0.000000	
25%	5.100000	2.800000	1.600000	0.300000	0.000000	
50%	5.800000	3.000000	4.400000	1.300000	1.000000	
75%	6.400000	3.300000	5.100000	1.800000	2.000000	
max	7.900000	4.400000	6.900000	2.500000	2.000000	

```
df.shape

(149, 5)
```

```
import pandas as pd
import matplotlib.pyplot as plt

class_col = df.columns[-1]
petal_len_col = df.columns[2]
print("Using class column:", class_col)
print("Using petal-length column:", petal_len_col)

if 'petal_size' not in df.columns:median_val = pd.to_numeric(df[petal_len_col], errors='coerce').median()
df['petal_size'] = df[petal_len_col].apply(
    lambda x: 'Large' if pd.to_numeric(x, errors='coerce') > median_val else 'Small'
)

ctab = pd.crosstab(df[class_col], df['petal_size'])
print("\nCrosstab:\n", ctab)

colors = plt.cm.tab10.colors[:ctab.shape[1]]
ax = ctab.plot(kind='bar', figsize=(8,5), color=colors, edgecolor='black', alpha=0.9)

# إضافة الأرقام فوق الأعمدة
for p in ax.patches:height = p.get_height()
if not (height is None) and not (pd.isna(height)):ax.annotate(
    str(int(height)),
    (p.get_x() + p.get_width() / 2, height),
    ha='center',
    va='bottom',
    fontsize=10,
    xytext=(0, 3),
    textcoords='offset points'
)

plt.title("Petal Size Distribution by Class")
plt.xlabel(f"Class ({class_col})")
plt.ylabel("Count")
plt.legend(title="Petal Size")
plt.grid(axis='y', linestyle='--', alpha=0.4)
plt.tight_layout()
plt.show()
```



```
species_counts = df['0'].value_counts()

plt.figure(figsize=(5,5))
plt.pie(
    species_counts,
    labels=species_counts.index,
    autopct='%1.1f%%',
    startangle=90
)
plt.title("Species Distribution")
plt.show()
```

