import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

import numpy as np

print("All Modules imported Successfully")

iris = pd.read\_csv("iris.csv")

print (iris.columns)

iris["species"].value\_counts()

sns.set\_style("whitegrid");

sns.FacetGrid(iris, hue="species", height =4) \

   .map(plt.scatter, "sepal\_length", "sepal\_width") \

   .add\_legend();

plt.show();

sns.set\_style("whitegrid");

sns.FacetGrid(iris, hue="species", height =4) \

   .map(plt.scatter, "petal\_length", "petal\_width") \

   .add\_legend();

plt.show();

plt.close()

sns.set\_style("whitegrid")

sns.pairplot(iris, hue="species", height=3)

plt.show()

sns.FacetGrid(iris, hue="species", height=5) \

    .map(sns.distplot, "petal\_length") \

    .add\_legend()

plt.show()

iris\_setosa = iris[iris['species'] == 'setosa']

print(np.mean(iris\_setosa["petal\_length"]))

print(np.std(iris\_setosa["petal\_length"]))

iris\_virginica = iris[iris['species'] == 'virginica']

print(np.mean(iris\_virginica["petal\_length"]))

print(np.std(iris\_virginica["petal\_length"]))

iris\_versicolor = iris[iris['species'] == 'versicolor']

print(np.mean(iris\_versicolor["petal\_length"]))

print(np.std(iris\_versicolor["petal\_length"]))

#Median, Quantiles, Percentiles, IQR.

print("\nMedians:")

print(np.median(iris\_setosa["petal\_length"]))

#Median with an outlier

print(np.median(np.append(iris\_setosa["petal\_length"],50)))

print(np.median(iris\_virginica["petal\_length"]))

print(np.median(iris\_versicolor["petal\_length"]))

print("\nQuantiles:")

print(np.percentile(iris\_setosa["petal\_length"],np.arange(0, 100, 25)))

print(np.percentile(iris\_virginica["petal\_length"],np.arange(0, 100, 25)))

print(np.percentile(iris\_versicolor["petal\_length"], np.arange(0, 100, 25)))

print("\n90th Percentiles:")

print(np.percentile(iris\_setosa["petal\_length"],90))

print(np.percentile(iris\_virginica["petal\_length"],90))

print(np.percentile(iris\_versicolor["petal\_length"], 90))

from statsmodels import robust

print ("\nMedian Absolute Deviation")

print(robust.mad(iris\_setosa["petal\_length"]))

print(robust.mad(iris\_virginica["petal\_length"]))

print(robust.mad(iris\_versicolor["petal\_length"]))

sns.boxplot(x = 'species', y = 'petal\_length', data = iris)

plt.show()

#2D Density plot, contors-plot

sns.jointplot(x="petal\_length", y="petal\_width", data=iris\_setosa, kind="kde");

plt.show();

iris\_virginica\_SW = iris\_virginica.iloc[:,1]

iris\_versicolor\_SW = iris\_versicolor.iloc[:,1]

from scipy import stats

stats.ks\_2samp(iris\_virginica\_SW, iris\_versicolor\_SW)

x = stats.norm.rvs(loc=0.2, size=10)

stats.kstest(x,'norm')

iris.isnull().sum()

data = iris.drop\_duplicates(subset ="species")

data

data.corr(method='pearson')

sns.violinplot(x="species", y="petal\_length", data=iris, size=8)

plt.show()

iris\_setosa = iris.loc[iris["species"] == "setosa"];

iris\_virginica = iris.loc[iris["species"] == "virginica"];

iris\_versicolor = iris.loc[iris["species"] == "versicolor"];

duplicate = iris.drop\_duplicates(subset ="species",)

duplicate

data.corr(method='pearson')

sns.heatmap(iris.corr(method='pearson').drop(

  [], axis=1).drop([], axis=0),

            annot = True);

plt.show()

correlation\_matrix = iris.corr()

plt.figure(figsize=(8, 6))

sns.heatmap(correlation\_matrix, annot=True, cmap="coolwarm", linewidths=0.5)

plt.title("Correlation Heatmap for Iris Dataset")

plt.show()

!pip install sweetviz

import sweetviz as sv

advert\_report = sv.analyze(iris)

#display the report

advert\_report.show\_html('Advertising.html')

#finding outliers and replacing with median value

col = iris['petal\_length']

col

col.describe()

#calculating quartiles and Inter-Quartile Range

q1 = np.percentile(col,25)

q3 = np.percentile(col,75)

iqr = q3-q1

#calculating Lower\_bound and Upper\_Bound

lower\_bound = q1 - (1.5\*iqr)

upper\_bound = q3 + (1.5\*iqr)

print(lower\_bound)

print(upper\_bound)

median = np.median(col)

median

#checking outliers in the dataset

outliers = [x for x in col if x < lower\_bound or x > upper\_bound]

print(outliers)

if(len(outliers) == 0):

    print("Hurrrayyy!!! No Outliers")

else:

    print("Saddddd!!!! Outlierssss")

#replacing outliers with median value

for i in range(len(col)):

    if col[i] in outliers:

        col[i] = median

print("Data with Outliers Replaced by Median:\n", col)

#checking Outliers present after replacing

q1 = np.percentile(col,25)

q3 = np.percentile(col,75)

iqr = q3-q1

lower\_bound = q1 - (1.5\*iqr)

upper\_bound = q3 + (1.5\*iqr)

# print(lower\_bound)

# print(upper\_bound)

median = np.median(col)

median

outliers = [x for x in col if x < lower\_bound or x > upper\_bound]

if(len(outliers) == 0):

    print("Hurrrayyy!!! No Outliers")

else:

    print("Saddddd!!!! Outlierssss")

import numpy as np

from scipy.stats import kstest, norm

# Generate a sample of data that you want to test

np.random.seed(0)  # Setting a seed for reproducibility

Sample\_data = np.random.normal(loc=0, scale=1, size=1000)  # Sample data from a normal distribution

# Perform a KS test to check if the sample\_data follows a normal distribution

ks\_statistic, p\_value = kstest(col, 'norm')

# Define the significance level (alpha)

alpha = 0.05

# Check the result of the KS test

if p\_value < alpha:

    print(f"The sample does NOT follow a normal distribution (p-value = {p\_value})")

else:

    print(f"The sample follows a normal distribution (p-value = {p\_value})")