**Problem Statement -Write a python program to create a data frame by uploading a csv file and carry out the basic operation of numpy such as finding the maximum value from the data set indexing and slicing of the data frame and to find the shape and dimension of the data framework .**

**Objective: To create a python program using pandas and NumPy to analyze a CSV dataset, including tasks like finding maximum values, indexing, slicing, and determining the dataset's shape and dimensions.**

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

data = pd.read\_csv('/content/Iris.csv')

# Extract only numeric columns

numeric\_data = data.drop(columns=['Id', 'Species'])

array\_data = numeric\_data.to\_numpy()

print("Shape of the array:", array\_data.shape)

print("Dimensions of the array:", array\_data.ndim)

print("Data type of the array:", array\_data.dtype)

print("First few rows of the array:")

print(array\_data[:5])

print("Value at row 0, column 1:", array\_data[0, 1])

print("Sliced array (row 0 to 2, column 0 to 2):")

print(array\_data[:3, :3])

array\_sum = np.sum(array\_data)

print("Sum of all elements in the array:", array\_sum)

array\_diff = np.diff(array\_data, axis=0)

print("Difference between consecutive elements in each row:")

print(array\_diff[:5])

array\_product = np.prod(array\_data)

print("Product of all elements in the array:", array\_product)

array\_mean = np.mean(array\_data, axis=0)

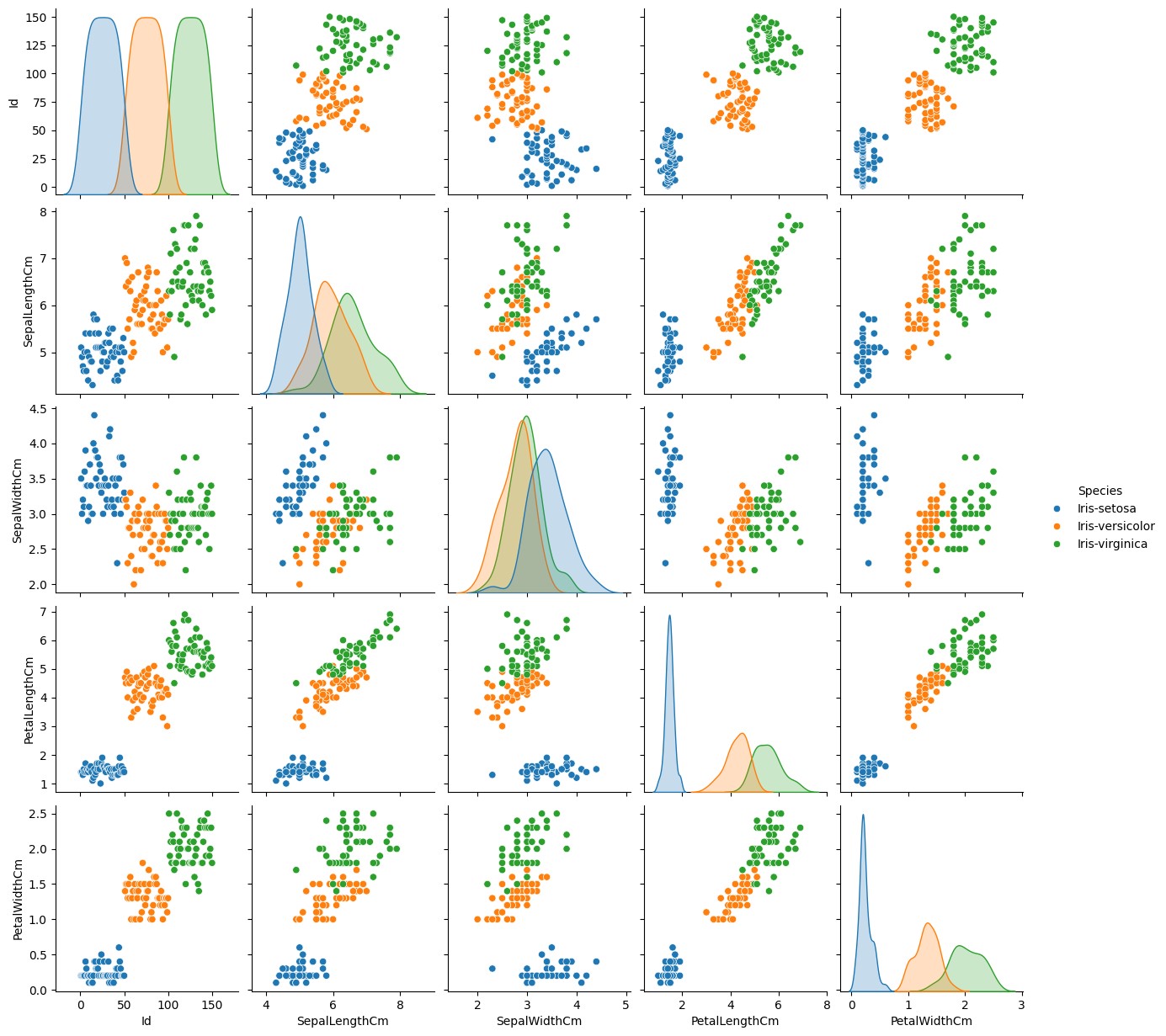
print("Mean of each column in the array:")

print(array\_mean)

sns.pairplot(data, hue='Species')

plt.show()

**Output:**



**Problem Statement - Write a python program to carry out a visualization for each feature separately.**

**Objective: python program to carry out a visualization for each feature separately.**

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.datasets import load\_iris

iris = load\_iris()

X = iris.data

y = iris.target

feature\_names = iris.feature\_names

target\_names = iris.target\_names

plt.figure(figsize=(12, 6))

for i in range(X.shape[1]):

plt.subplot(2, 2, i+1)

sns.histplot(X[:, i], kde=True, color='skyblue')

plt.title(feature\_names[i])

plt.tight\_layout()

plt.show()

sns.pairplot(sns.load\_dataset('iris'), hue='2es')

plt.show()

from sklearn.decomposition import PCA

pca = PCA(n\_components=2)

X\_pca = pca.fit\_transform(X)

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

sns.scatterplot(x=X\_pca[:, 0], y=X\_pca[:, 1], hue=target\_names[y], palette='viridis')

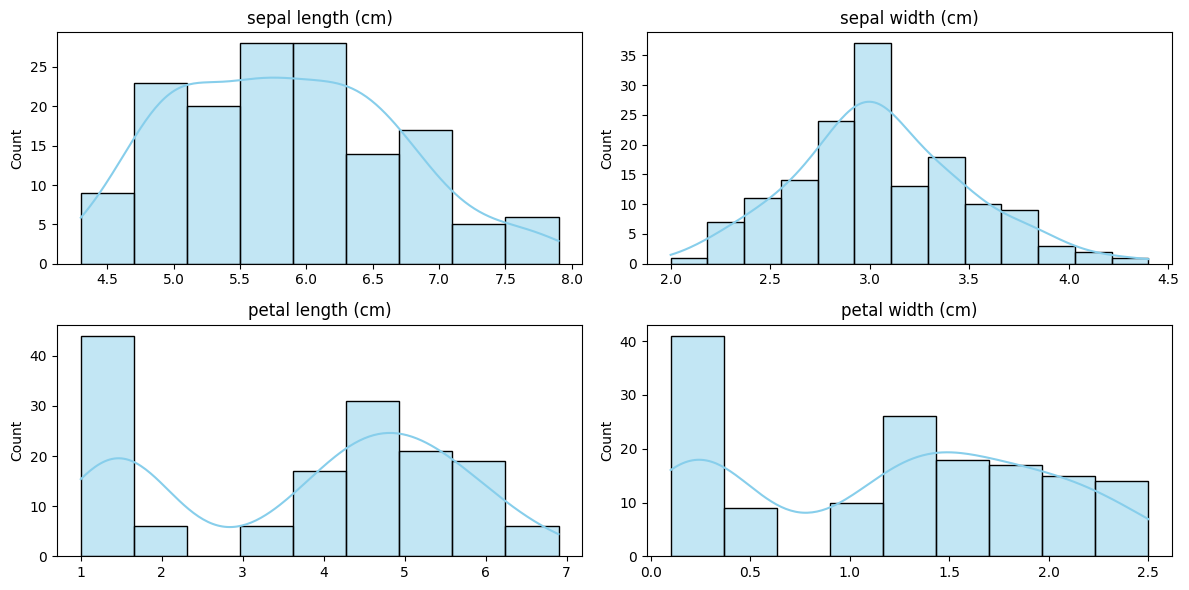
plt.title('PCA Visualization')

plt.xlabel('Principal Component 1')

plt.ylabel('Principal Component 2')

plt.show()

**Output:**



**Problem Statement -Write a python program to implement logistic regression on California\_housing dataset.**

**Objective: python program to implement logistic regression on California\_housing dataset.**

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import accuracy\_score, confusion\_matrix

# Load the dataset

df = pd.read\_csv('/content/sample\_data/california\_housing\_train.csv')

# Data preprocessing by dropping any rows with missing values

df.dropna(inplace=True)

# Splitting the dataset into X and y variable

X = df.drop('median\_house\_value', axis=1) # Replace 'target\_column\_name' with the actual target column name

y = df['median\_house\_value'] # Replace 'target\_column\_name' with the actual target column name

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Feature scaling

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# Logistic Regression model

model = LogisticRegression()

# Training the model

model.fit(X\_train\_scaled, y\_train)

# Predictions on the testing set

y\_pred = model.predict(X\_test\_scaled)

accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy:", accuracy)

**Output:**

Accuracy: 0.05058823529411765

/usr/local/lib/python3.10/dist-packages/sklearn/linear\_model/\_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):

STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max\_iter) or scale the data as shown in:

https://scikit-learn.org/stable/modules/preprocessing.html

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear\_model.html#logistic-regression

n\_iter\_i = \_check\_optimize\_result(

**Problem Statement -Write a python program to implement ID3 algorithm using entropy in decision tree.**

**Objective: python program to implement ID3 algorithm using entropy in decision tree.**

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import accuracy\_score, confusion\_matrix

# Load the dataset

df = pd.read\_csv('/content/sample\_data/california\_housing\_train.csv')

# Data preprocessing

# Dropping any rows with missing values

df.dropna(inplace=True)

# Splitting the dataset into features and target variable

X = df.drop('median\_house\_value', axis=1) # Replace 'target\_column\_name' with the actual target column name

y = df['median\_house\_value'] # Replace 'target\_column\_name' with the actual target column name

# Splitting the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Feature scaling

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# Decision Tree model

model = DecisionTreeClassifier(criterion='entropy') # Using ID3 (Entropy) criterion

# Training the model

model.fit(X\_train\_scaled, y\_train)

# Predictions on the testing set

y\_pred = model.predict(X\_test\_scaled)

# Model evaluation

accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy:", accuracy)

print("Confusion Matrix:")

**Output:**

Accuracy: 0.025

Confusion Matrix:

**Problem Statement -Write a python program to implement CART algo for decision tree.**

**Objective: python program to implement CART algo for decision tree.**

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import accuracy\_score, confusion\_matrix

# Load the dataset

df = pd.read\_csv('/content/sample\_data/california\_housing\_train.csv')

# Data preprocessing

# Dropping any rows with missing values

df.dropna(inplace=True)

# Splitting the dataset into features and target variable

X = df.drop('median\_house\_value', axis=1) # Replace 'target\_column\_name' with the actual target column name

y = df['median\_house\_value'] # Replace 'target\_column\_name' with the actual target column name

# Splitting the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Feature scaling

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# CART (Decision Tree) model

model = DecisionTreeClassifier(criterion='gini') # Using CART (Gini) criterion

# Training the model

model.fit(X\_train\_scaled, y\_train)

# Predictions on the testing set

y\_pred = model.predict(X\_test\_scaled)

# Model evaluation

accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy:", accuracy)

**Output**

Accuracy: 0.025294117647058825

**Problem Statement -Write a python program to implement SVM using linear kernel on iris.csv.**

**Objective: python program to implement SVN.**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVC

from sklearn.metrics import classification\_report, accuracy\_score

url = "

https://archive.ics.uci.edu/ml/machine-learning-da

tabases/iris/iris.data"

column\_names = ['sepal\_length', 'sepal\_width', 'petal\_length', 'petal\_width', 'species']

iris = pd.read\_csv(url, header=None, names=column\_names)

print(iris.head())

X = iris.iloc[:, :-1].values # all columns except the last one

y = iris.iloc[:, -1].values # the last column

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

svm = SVC(kernel='linear', random\_state=42)

svm.fit(X\_train, y\_train)

y\_pred = svm.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Accuracy: {accuracy:.2f}")

print(classification\_report(y\_test, y\_pred))

**Output:**

sepal\_length sepal\_width petal\_length petal\_width species

0 5.1 3.5 1.4 0.2 Iris-setosa

1 4.9 3.0 1.4 0.2 Iris-setosa

2 4.7 3.2 1.3 0.2 Iris-setosa

3 4.6 3.1 1.5 0.2 Iris-setosa

4 5.0 3.6 1.4 0.2 Iris-setosa

Accuracy: 0.98

precision recall f1-score support

Iris-setosa 1.00 1.00 1.00 19

Iris-versicolor 1.00 0.92 0.96 13

Iris-virginica 0.93 1.00 0.96 13

accuracy 0.98 45

macro avg 0.98 0.97 0.97 45

weighted avg 0.98 0.98 0.98 45

**Problem Statement - Write a python program to implement SVM using sigmoid kernel on iris.csv and write a python program to implement k-NN on iris.csv with k=3**

**Objective: python program to implement KNeighbors.**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVC

from sklearn.metrics import classification\_report, accuracy\_score

url = "

https://archive.ics.uci.edu/ml/machine-learning-da

tabases/iris/iris.data"

column\_names = ['sepal\_length', 'sepal\_width', 'petal\_length', 'petal\_width', 'species']

iris = pd.read\_csv(url, header=None, names=column\_names)

print(iris.head())

X = iris.iloc[:, :-1].values # all columns except the last one

y = iris.iloc[:, -1].values # the last column

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

1/3

https://colab.research.google.com/drive/1WV1jY8oMkT9ZCczN8AQLxsdgc2JLueoH#scrollTo=BM6zEw0Nn58n&printMode=true

5/16/24, 9:54 AM

Untitled127.ipynb - Colab

svm = SVC(kernel='sigmoid', random\_state=42)

svm.fit(X\_train, y\_train)

y\_pred = svm.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Accuracy: {accuracy:.2f}")

print(classification\_report(y\_test, y\_pred))

**Output:**

sepal\_length sepal\_width petal\_length petal\_width species

0 5.1 3.5 1.4 0.2 Iris-setosa

1 4.9 3.0 1.4 0.2 Iris-setosa

2 4.7 3.2 1.3 0.2 Iris-setosa

3 4.6 3.1 1.5 0.2 Iris-setosa

4 5.0 3.6 1.4 0.2 Iris-setosa

Accuracy: 0.89

precision recall f1-score support

Iris-setosa 1.00 1.00 1.00 19

Iris-versicolor 0.90 0.69 0.78 13

Iris-virginica 0.75 0.92 0.83 13

accuracy 0.89 45

macro avg 0.88 0.87 0.87 45

weighted avg 0.90 0.89 0.89 45

# Import necessary libraries

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score, classification\_report

url = "https://archive.ics.uci.edu/ml/machine-learning-da

tabases/iris/iris.data"

column\_names = ['sepal\_length', 'sepal\_width', 'petal\_length', 'petal\_width', 'species']

data = pd.read\_csv(url, header=None, names=column\_names)

print(data.head())

# The Iris dataset typically has columns: SepalLength, SepalWidth, PetalLength, PetalWidth, Species

X = data.drop(columns=['species'])

y = data['species']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

knn = KNeighborsClassifier(n\_neighbors=3)

knn.fit(X\_train, y\_train)

y\_pred = knn.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print(f'Accuracy: {accuracy:.2f}')

print(classification\_report(y\_test, y\_pred))

**Output:**

sepal\_length sepal\_width petal\_length petal\_width species

0 5.1 3.5 1.4 0.2 Iris-setosa

1 4.9 3.0 1.4 0.2 Iris-setosa

2 4.7 3.2 1.3 0.2 Iris-setosa

3 4.6 3.1 1.5 0.2 Iris-setosa

4 5.0 3.6 1.4 0.2 Iris-setosa

Accuracy: 1.00

precision recall f1-score support

Iris-setosa 1.00 1.00 1.00 10

Iris-versicolor 1.00 1.00 1.00 9

Iris-virginica 1.00 1.00 1.00 11

accuracy 1.00 30

2/3

https://colab.research.google.com/drive/1WV1jY8oMkT9ZCczN8AQLxsdgc2JLueoH#scrollTo=BM6zEw0Nn58n&printMode=true

5/16/24, 9:54 AM

Untitled127.ipynb - Colab

macro avg 1.00 1.00 1.00 30

weighted avg 1.00 1.00 1.00 30

3/3

https://colab.research.google.com/drive/1WV1jY8oMkT9ZCczN8AQLxsdgc2JLueoH#scrollTo=BM6zEw0Nn58n&printMode=true