Exp3 data preprocessing

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

dataset = pd.read\_csv("Data.csv")

x= dataset.iloc[:,:-1].values

y= dataset.iloc[:,-1].values

print(y)

print(x)

print(dataset)

from sklearn.impute import SimpleImputer

imputer = SimpleImputer(missing\_values=np.nan, strategy='mean')

imputer.fit(x[:,1:3])

x[:,1:3]=imputer.transform(x[:,1:3])

print(x)

from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import OneHotEncoder

ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0])], rem)ainder='passthrough'

x = np.array(ct.fit\_transform(x))

print(X)

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

y = np.array(le.fit\_transform(y))

print(y)

Exp 4 DATA EXPLORATION

import pandas as pd

import seaborn as sb

import matplotlib.pyplot as plt

import numpy as np

iris\_d = sb.load\_dataset("iris")

iris\_d.head()

iris\_d.tail()

iris\_d.shape

iris\_d.info()

iris\_d['sepal\_length'].describe()

iris\_d.describe()

iris\_d.isnull().sum()

plt.scatter(iris\_d['sepal\_length'],iris\_d['sepal\_width'], color='red')

plt.title("scatter plot")

plt.xlabel("Sepal length")

plt.ylabel("Sepal width")

plt.show()

plt.hist(iris\_d['sepal\_width'], bins=40)

plt.title("Histogram")

plt.xlabel("Sepal width")

plt.ylabel("Frequency")

plt.show()

plt.hist(iris\_d['sepal\_width'], bins=15)

plt.title("Histogram")

plt.xlabel("Sepal width")

plt.ylabel("Frequency")

plt.show()

plt.hist(iris\_d['sepal\_width'], bins='auto')

plt.title("Histogram")

plt.xlabel("Sepal width")

plt.ylabel("Frequency")

plt.show()

plt.hist(iris\_d['petal\_width'], bins=40)

plt.title("Histogram")

plt.xlabel("Petal width")

plt.ylabel("Frequency")

plt.show()

plt.hist(iris\_d['petal\_width'], bins=5)

plt.title("Histogram")

plt.xlabel("Petal width")

plt.ylabel("Frequency")

plt.show()

sb.boxplot(x="sepal\_width", data=iris\_d)

plt.title("Box Plot")

sb.boxplot(x="sepal\_length", data=iris\_d)

plt.title("Box Plot")

import scipy.stats as stats

stats.probplot(iris\_d['petal\_length'], dist="norm", plot=plt)

plt.title("Q-Q Plot of Sepal Width (Normal Distribution)")

plt.grid(True)

plt.show()

import scipy.stats as stats

stats.probplot(iris\_d['sepal\_length'], dist="norm", plot=plt)

plt.title("Q-Q Plot of Sepal Length (Normal Distribution)")

plt.grid(True)

plt.show()

import scipy.stats as stats

stats.probplot(iris\_d['petal\_length'], dist="uniform", plot=plt)

plt.title("Q-Q Plot of Sepal Width (Uniform Distribution)")

plt.grid(True)

plt.show()

Exp5 decision tree

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('Social\_Network\_Ads.csv')

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, -1].values

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 0)

print(X\_train)

print(y\_train)

print(X\_test)

print(y\_test)

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

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

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

print(X\_train)

print(X\_test)

from sklearn.tree import DecisionTreeClassifier

classifier = DecisionTreeClassifier(criterion = 'entropy', random\_state = 0)

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

print(classifier.predict(sc.transform([[20,87000]])))

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

print(y\_pred)

print(y\_test)

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

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

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

EXp6 kmeanss

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('Mall\_Customers.csv')

X = dataset.iloc[:, [3, 4]].values

X

from sklearn.cluster import KMeans

wcss = []

for i in range(1, 11):

kmeans = KMeans(n\_clusters = i, init = 'k-means++', random\_state = 42)

kmeans.fit(X)

wcss.append(kmeans.inertia\_)

plt.plot(range(1, 11), wcss)

plt.title('The Elbow Method')

plt.xlabel('Number of clusters')

plt.ylabel('WCSS')

plt.show()

kmeans = KMeans(n\_clusters = 5, init = 'k-means++', random\_state = 42)

y\_kmeans = kmeans.fit\_predict(X)

plt.scatter(X[y\_kmeans == 0, 0], X[y\_kmeans == 0, 1], s = 100, c = 'red', label = 'Cluster 1')

plt.scatter(X[y\_kmeans == 1, 0], X[y\_kmeans == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')

plt.scatter(X[y\_kmeans == 2, 0], X[y\_kmeans == 2, 1], s = 100, c = 'green', label = 'Cluster 3')

plt.scatter(X[y\_kmeans == 3, 0], X[y\_kmeans == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')

plt.scatter(X[y\_kmeans == 4, 0], X[y\_kmeans == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')

plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], s = 300, c = 'yellow', label = 'Centroids')

plt.title('Clusters of customers')

plt.xlabel('Annual Income (k$)')

plt.ylabel('Spending Score (1-100)')

plt.legend()

plt.show()

Exp7 apriori

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('Market\_Basket\_Optimisation.csv', header = None)

transactions = []

for i in range(0, 7501):

transactions.append([str(dataset.values[i,j]) for j in range(0, 20)])

dataset

transactions

!pip install apyori

from apyori import apriori

rules = apriori(transactions = transactions, min\_support = 0.003, min\_confidence = 0.2)

results = list(rules)

results

EXP10 page rank

import numpy as np

links = np.array([

[0, 1, 1, 0], # A -> B, C

[0, 0, 1, 0], # B -> C

[1, 0, 0, 1], # C -> A, D

[0, 0, 1, 0] # D -> C

])

n = len(links)

damping\_factor = 0.85

tolerance = 0.0001

max\_iter = 100

outgoing\_links = links.sum(axis=1)

for i in range(n):

if outgoing\_links[i] != 0:

links[i] = links[i] / outgoing\_links[i]

PR = np.ones(n) / n

for \_ in range(max\_iter):

new\_PR = (1 - damping\_factor) / n + damping\_factor \* np.dot(links.T, PR)

if np.linalg.norm(new\_PR - PR, ord=1) < tolerance:

Break

PR = new\_PR

pages = ['A', 'B', 'C', 'D']

print("Final Page Rank Values:\n")

for i in range(n):

print(f"Page {pages[i]}: {PR[i]:.4f}")