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}")
```