**import** pandas **as** pd

**import** numpy **as** np

**def** create\_dataset():

# np.random.seed(1) # aynı random sayıları üretmek için

sample = 1000

user\_id = np.arange(0, sample)

age = np.random.randint(18, 70, size=sample)

purchase\_frequency = np.random.randint(1, 20, size=sample) # aylık alışveriş sıklığı

total\_spent = np.random.randint(30, 1500, size=sample) # ylık harcama

high\_spender = np.zeros(sample) # herkes düşük harcama olarak başlar

# aşağıdaki kontrolllerle güncellenir değeri

high\_spender[(age < 30) & (purchase\_frequency > 10) & (total\_spent > 500)] = 1

high\_spender[(age >= 60) & (purchase\_frequency < 8) & (total\_spent > 1000)] = 1

high\_spender[(age >= 30) & (age < 60) & (purchase\_frequency > 12) & (total\_spent > 800)] = 1

data = {

'User\_ID': user\_id,

'Age': age,

'Purchase\_Frequency': purchase\_frequency,

'Total\_Spent': total\_spent,

'High\_Spender': high\_spender # Sınıflandırma hedefi: 1 = yüksek harcama, 0 = düşük harcama

}

df = pd.DataFrame(data)

df.to\_csv('C:\\Users\\Administrator\\Desktop\\Dosyalar\\Kodlar\\python\\data-mining\\sample\_data.csv', index=**False**)

print("Dataset succesfully saved.")

**return** df

create\_dataset()

**import** pandas **as** pd

**import** numpy **as** np

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

**from** sklearn.preprocessing **import** StandardScaler

**from** sklearn.neighbors **import** KNeighborsClassifier

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

data\_path = 'data-mining/sample\_data.csv'

data = pd.read\_csv(data\_path)

features = ['Age', 'Purchase\_Frequency', 'Total\_Spent']

target = 'High\_Spender'

**def** run\_knn(data, n\_neighbors=5, test\_size=0.4, random\_state=1):

features = ['Age', 'Purchase\_Frequency', 'Total\_Spent']

x = data[features]

y = data[target]

scaler = StandardScaler()

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

x\_train, x\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=test\_size, random\_state=random\_state)

knn = KNeighborsClassifier(n\_neighbors=n\_neighbors)

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

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

test\_results = pd.DataFrame(x\_test, columns=features)

test\_results['Actual'] = y\_test.values

test\_results['Predicted'] = y\_pred

print("\nPREDICTED VALUES")

print(test\_results)

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

print("Modelin doğruluğu:", accuracy)

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

print("\nCONFUSION MATRIX")

print(cm)

print("\nORIGINAL DATAS KNN")

run\_knn(data)

data.iloc[0:5, [0,1,2]] = np.nan

data.iloc[20:45, [1,2]] = np.nan

data.iloc[17, [0,1,2]] = np.nan

print("\nAFTER REPLACEMENT WITH NAN")

print(data)

data\_filled = data.copy()

data\_filled[features[0]] = data\_filled[features[0]].fillna(data\_filled[features[0]].mean().astype(int))

data\_filled[features[1]] = data\_filled[features[1]].fillna(data\_filled[features[1]].mean().astype(int))

data\_filled[features[2]] = data\_filled[features[2]].fillna(data\_filled[features[2]].mean().astype(int))

print("\nAFTER FILLING WITH COLUMN'S AVERAGE")

print(data\_filled)

print("\nFILLED DATAS SVM")

run\_knn(data\_filled)

**import** pandas **as** pd

**import** numpy **as** np

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

**from** sklearn.preprocessing **import** StandardScaler

**from** sklearn.svm **import** SVC

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

data\_path = 'data-mining/sample\_data.csv'

data = pd.read\_csv(data\_path)

features = ['Age', 'Purchase\_Frequency', 'Total\_Spent']

target = 'High\_Spender'

**def** run\_svm(data, target\_column, test\_size = 0.4, random\_state=1):

x = data[features]

y = data[target\_column]

scaler = StandardScaler()

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

x\_train, x\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=test\_size, random\_state=random\_state)

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

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

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

test\_results = pd.DataFrame(x\_test, columns=features)

test\_results['Actual'] = y\_test.values

test\_results['Predicted'] = y\_pred

print("\nPREDICTED VALUES")

print(test\_results)

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

print("Modelin doğruluğu:", accuracy)

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

print("\nCONFUSION MATRIX")

print(cm)

print("\nORIGINAL DATAS SVM")

run\_svm(data, target)

data.iloc[0:5, [0,1,2]] = np.nan

data.iloc[20:45, [1,2]] = np.nan

data.iloc[17, [0,1,2]] = np.nan

print("\nAFTER REPLACEMENT WITH NAN")

print(data)

data\_filled = data.copy()

data\_filled[features[0]] = data\_filled[features[0]].fillna(data\_filled[features[0]].mean().astype(int))

data\_filled[features[1]] = data\_filled[features[1]].fillna(data\_filled[features[1]].mean().astype(int))

data\_filled[features[2]] = data\_filled[features[2]].fillna(data\_filled[features[2]].mean().astype(int))

print("\nAFTER FILLING WITH COLUMN'S AVERAGE")

print(data\_filled)

print("\nFILLED DATAS SVM")

run\_svm(data\_filled, target)