Bahria University,

Karachi Campus

A logo with text on it

Description automatically generated

LAB EXPERIMENT NO.

08

LIST OF TASKS

|  |  |
| --- | --- |
| TASK NO | OBJECTIVE |
| **01** | Implement KNN for Iris Flower Classification in KNIME. |
| **02** | Implement Using Python KNN for Iris Flower Classification. |
| 03 | Implement a Movie Recommendation System Using K-Nearest Neighbors (KNN) in KNIME. |

Submitted On:

25-04-2024

(Date: DD/MM/YYYY)

**Task No. 01:** Implement KNN for Iris Flower Classification in KNIME.

**Description**: You are tasked with implementing the K-Nearest Neighbors (KNN) algorithm using KNIME Analytics Platform to classify Iris flowers based on their features. The Iris dataset contains four features: sepal length, sepal width, petal length, and petal width. There are three classes of Iris flowers in the dataset: Setosa, Versicolor, and Virginica.

**Solution:**

A diagram of a machine

Description automatically generated

A screenshot of a computer

Description automatically generated

**Output:**

**Task No. 02:** Implement KNN for Iris Flower Classification.

**Description**: You are tasked with implementing the K-Nearest Neighbors (KNN) algorithm to classify Iris flowers based on their features. The Iris dataset contains four features: sepal length, sepal width, petal length, and petal width. There are three classes of Iris flowers in the dataset: Setosa, Versicolor, and Virginica.

**Solution:**

from sklearn.datasets import load\_iris

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

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score

iris = load\_iris()

X = iris.data # Features

y = iris.target # Target labels

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=4)

k = 3

knn\_classifier\_euclidean = KNeighborsClassifier(n\_neighbors=k, metric='euclidean')

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

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

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

print("Accuracy with Euclidean distance metric:", accuracy\_euclidean)

knn\_classifier\_minkowski = KNeighborsClassifier(n\_neighbors=k, metric='minkowski')

plt.show()

print("Accuracy with Minkowski distance metric:", accuracy\_minkowski)

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

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

sns.heatmap(cm\_minkowski, annot=True, fmt="d", cmap="Greens")

plt.title("Confusion Matrix - Minkowski Distance Metric")

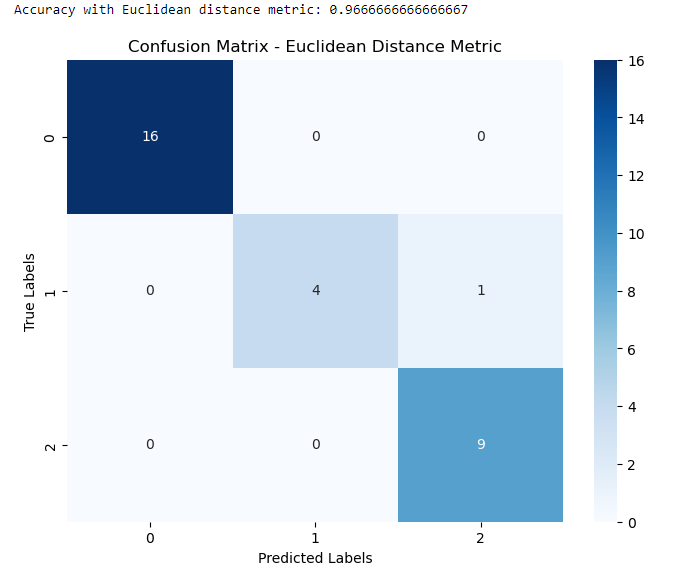
plt.xlabel("Predicted Labels")

plt.ylabel("True Labels")

plt.show()

**Output:**

A green squares with white text

Description automatically generated

**Task No. 03:** Implement a Movie Recommendation System Using K-Nearest Neighbors (KNN) in KNIME.

**Description**: You are tasked with building a movie recommendation system using the K-Nearest Neighbors (KNN) algorithm in KNIME Analytics Platform. The goal of the recommendation system is to suggest movies to users based on their similarity to other users'

preferences.

**Solution:**

import numpy as np # linear algebra

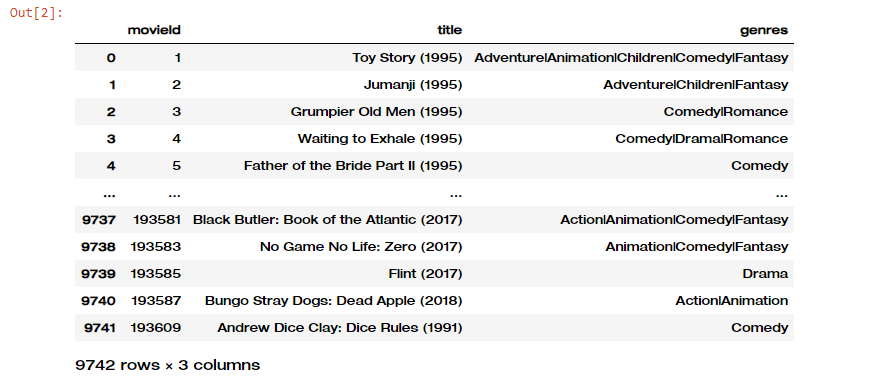
import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)

import matplotlib.pyplot as plt

import seaborn as sns

movie = pd.read\_csv("movies.csv")

movie



rating = pd.read\_csv("ratings.csv")

rating

A screenshot of a computer

Description automatically generated

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

tags

A screenshot of a computer

Description automatically generated

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

links

A screenshot of a computer

Description automatically generated

data = pd.merge(movie,rating)

data

A screenshot of a computer

Description automatically generated

data.drop(["genres","timestamp"], axis = 1,inplace = True)

data

A screenshot of a movie

Description automatically generated

data\_pivot = data.pivot\_table(index='userId',columns='title',values='rating')

data\_pivot

A table with numbers and letters

Description automatically generated

data\_pivot2 = pd.pivot(index = "movieId", columns = "userId", data = rating, values = "rating")

data\_pivot2

Movie\_voted = pd.DataFrame(rating.groupby("movieId")["rating"].agg("count"))

Movie\_voted.reset\_index(level = 0, inplace = True)

Movie\_voted

A table with numbers and letters

Description automatically generated

User\_Voted = pd.DataFrame(rating.groupby("userId")["rating"].agg("count"))

User\_Voted.reset\_index(level = 0, inplace = True)

User\_Voted

A white rectangular box with black numbers

Description automatically generated

data\_pivot2.fillna(0, inplace = True)

data\_pivot2

A table with numbers and symbols

Description automatically generated

dataLast = data\_pivot2.loc[Movie\_voted[Movie\_voted["rating"] > 10]["movieId"],:]

dataLast = dataLast.loc[:, User\_Voted[User\_Voted["rating"] > 60]["userId"]]

dataLast

A screenshot of a table

Description automatically generated

**Recommendation Systems**

from scipy.sparse import csr\_matrix

csr\_data = csr\_matrix(dataLast.values)

dataLast.reset\_index(inplace=True)

csr\_data

from sklearn.neighbors import NearestNeighbors

knn = NearestNeighbors(metric='cosine', algorithm='brute', n\_neighbors=20, n\_jobs=-1)

knn.fit(csr\_data)

def movie\_recommendation(movie\_name):

for val in rec\_movie\_indices:

movie\_idx = dataLast.iloc[val[0]]['movieId']

idx = movie[movie['movieId'] == movie\_idx].index

recommend\_frame.append({'Title':movie.iloc[idx]['title'].values[0],'Distance':val[1]})

df = pd.DataFrame(recommend\_frame,index=range(1,n\_movies\_to\_reccomend+1))

return df

else:

return "No movies found. Please check your input"

# result of Recommendation System # result of Recommendation System

movie\_recommendation('Iron Man')

movie\_recommendation('Fight Club')

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated