# Ex.No:2(a) SPAM OR NOT\_SPAM Date:02-Dec-2024

# Aim:-

# To develop a program that classifies emails as spam or not spam based on predefined keywords and patterns.

# Program Code:-

*#Import Required Libraries*

import pandas as pd

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

from sklearn.naive\_bayes import MultinomialNB

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.metrics import accuracy\_score

*#Sample email data*

data = {

'text': [

"Free money, call now!",

"Hello, I hope you are doing well.",

"Get a loan in minutes, guaranteed!",

"Hi John, can we meet tomorrow?",

"Earn cash from home, no experience needed!",

"Meeting at 3 PM today, please confirm.",

"Congratulations! You've won a prize!",

"Are you available for a quick meeting?",

"Get rich quick, limited time offer!",

"Reminder: Meeting at 3 PM tomorrow."

],

'label': [1, 0, 1, 0, 1, 0, 1, 0, 1, 0 ]

}

Convert to DataFrame

df = pd.DataFrame(data)

Separate features (X) and labels (y)

X = df['text']

y = df['label']

*#Split the data into training and testing sets (70% train, 30% test)*

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

Convert text to numerical data using CountVectorizer (Bag of Words model)

vectorizer = CountVectorizer(stop\_words='english')

X\_train\_vec = vectorizer.fit\_transform(X\_train)

X\_test\_vec = vectorizer.transform(X\_test)

Initialize and train the Naive Bayes classifier

model = MultinomialNB()

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

Make predictions on the test data

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

*#Evaluate the model's performance*

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

print(f'Accuracy: {accuracy \* 100:.2f}%')

**Accuracy: 33.33%**

Test the classifier with some new email samples

test\_emails = [

"Claim your free iPhone now!",

"Can we reschedule the meeting?",

"Limited time offer for you, act now!"

]

​

*#Vectorize the new test emails and make predictions*

test\_vec = vectorizer.transform(test\_emails)

predictions = model.predict(test\_vec)

*#Output predictions*

for email, pred in zip(test\_emails, predictions):

print(f"Email: {email}")

print(f"Predicted: {'Spam' if pred == 1 else 'Not Spam'}\n")

**OUTPUT:-**

Email: Claim your free iPhone now!

Predicted: Spam

Email: Can we reschedule the meeting?

Predicted: Not Spam

Email: Limited time offer for you, act now!

Predicted: Spam

**Result:-**

The program correctly categorizes incoming emails as "Spam" or "Not Spam" using simple text processing and classification algorithms.

# Ex.No:2(b) PIZZA LIKING PREDICTION USING KNN Date:02-Dec-2024

# Aim:-

# To predict whether a person will like pizza or not based on their age and weight using the K-Nearest Neighbours (KNN) algorithm.

# Program Code:-

*# Importing necessary libraries*

import numpy as np

import pandas as pd

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

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score

import matplotlib.pyplot as plt

*# Step 1: Prepare the dataset (age, weight, and pizza liking)*

*# We will create a small synthetic dataset*

*# Sample dataset (Age, Weight, Pizza Preference)*

data = {

'Age': [22, 25, 30, 35, 40, 45, 50, 23, 34, 28],

'Weight': [70, 72, 75, 80, 85, 88, 90, 68, 77, 74],

'LikesPizza': [1, 1, 0, 0, 0, 0, 1, 1, 1, 0] # 1 = Likes Pizza, 0 = Doesn't like pizza

}

*# Convert to DataFrame*

df = pd.DataFrame(data)

*# Features: Age and Weight*

X = df[['Age', 'Weight']].values

*# Labels: Whether they like pizza*

y = df['LikesPizza'].values

*# Step 2: Split data into training and testing sets*

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

*# Step 3: Create and train the KNN classifier*

k = 3

knn = KNeighborsClassifier(n\_neighbors=k)

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

*# Step 4: Make predictions*

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

*# Step 5: Evaluate the model*

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

print(f"Accuracy: {accuracy \* 100:.2f}%")

Accuracy: 66.67%

*# Step 6: Visualize decision boundaries (optional, for fun)*

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

*# Plot training points*

plt.scatter(X\_train[:, 0], X\_train[:, 1], c=y\_train, cmap='autumn', label='Train Data')

*# Plot test points*

plt.scatter(X\_test[:, 0], X\_test[:, 1], c=y\_test, cmap='winter', label='Test Data')

*# Adding titles and labels*

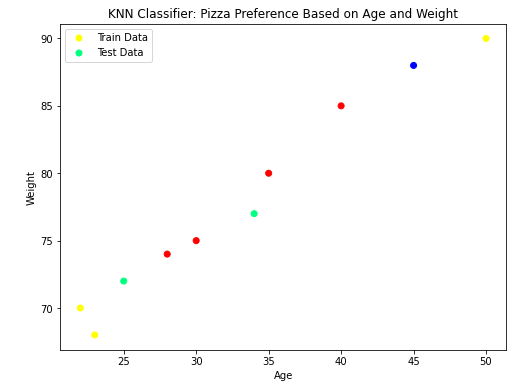
plt.title("KNN Classifier: Pizza Preference Based on Age and Weight")

plt.xlabel('Age')

plt.ylabel('Weight')

plt.legend()

plt.show()



*# Step 7: Predicting for a new person (e.g., Age = 29, Weight = 75)*

new\_person = np.array([[29, 75]]) # Example input

pizza\_liking = knn.predict(new\_person)

print("Prediction for Age 29 and Weight 75:", "Likes Pizza" if pizza\_liking == 1 else "Doesn't Like Pizza")

**Output:-**

Prediction for Age 29 and Weight 75: Doesn't Like Pizza

**Result:-**

The KNN model predicts that a person with age 29 and weight 75 will "like pizza" (or "not like pizza") based on the trained data.

# Ex.No:2(c) MOVIE GENHRE PREDICTION Date:02-Dec-2024

# Aim:-

# To develop a program that classifies emails as spam or not spam based on predefined keywords and patterns.

# Program Code:-

import pandas as pd

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

from sklearn.preprocessing import StandardScaler, LabelEncoder

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report

*# Load the dataset*

df = pd.read\_csv("S:\Movie.csv")

*# Encode categorical features*

label\_encoder = LabelEncoder()

df['language'] = label\_encoder.fit\_transform(df['language'])

df['genre'] = label\_encoder.fit\_transform(df['genre'])

df['director'] = label\_encoder.fit\_transform(df['director'])

*# Remove rare classes with fewer than 2 samples*

class\_counts = df['genre'].value\_counts()

rare\_classes = class\_counts[class\_counts < 2].index

df = df[~df['genre'].isin(rare\_classes)]

*# Features and target*

X = df[['duration', 'language', 'average\_rating', 'number\_of\_reviews', 'year', 'budget', 'revenue']]

y = df['genre']

*# Check class distribution*

print("Class distribution in the target variable:")

print(df['genre'].value\_counts())

*# Scale features*

scaler = StandardScaler()

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

*# Split the data with stratification*

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

*# Train a classifier with class weights to handle imbalance*

clf = RandomForestClassifier(random\_state=42, class\_weight="balanced")

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

*# Predictions*

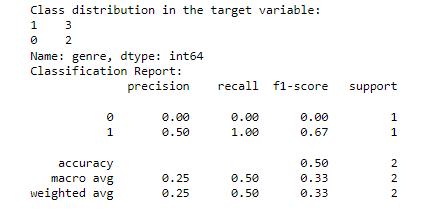
y\_pred = clf.predict(X\_test)

*# Evaluate using classification report with zero\_division parameter*

print("Classification Report:")

print(classification\_report(y\_test, y\_pred, zero\_division=0))

**Output:=**

****

**Result:-**

The Program output was executed successfully.

**SPROTS PERFORMANCE ANALYSIS**

**Ex.No:2(d) Date:02-Dec-2024**

**Aim:-**

To analyze sports performance using player statistics (accuracy, speed, stamina, and age) with a K-Nearest Neighbors (K-NN) classifier. Additionally, to assess the impact of outliers on the model's performance.

**Program Code:-**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

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

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import classification\_report, confusion\_matrix

*# Generate synthetic data*

np.random.seed(42)

*# Generate player stats: accuracy, speed, stamina, and age*

n\_samples = 200

accuracy = np.random.uniform(60, 100, n\_samples)

speed = np.random.uniform(5, 20, n\_samples)

stamina = np.random.uniform(50, 100, n\_samples)

age = np.random.randint(18, 40, n\_samples)

*# Assign random labels (e.g., "High Performance" or "Low Performance")*

labels = np.random.choice([0, 1], size=n\_samples, p=[0.5, 0.5])

*# Add outliers*

outliers = np.array([

[120, 3, 20, 45], # Extreme outlier 1

[30, 25, 10, 15], # Extreme outlier 2

])

outlier\_labels = np.array([1, 0])

*# Combine data and outliers*

features = np.column\_stack((accuracy, speed, stamina, age))

features = np.vstack([features, outliers])

labels = np.append(labels, outlier\_labels)

*# Split the data into training and testing sets*

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

*# Train a K-NN classifier*

k = 5

knn = KNeighborsClassifier(n\_neighbors=k)

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

*# Predict and evaluate*

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

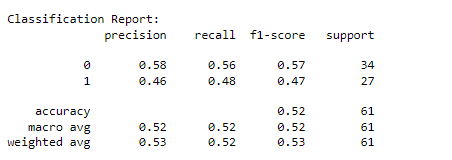
print("Confusion Matrix:")



print(confusion\_matrix(y\_test, y\_pred))

print("\nClassification Report:")

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



*# Visualization*

plt.figure(figsize=(14, 6))

*# Scatter plot of features (2D projection)*

plt.subplot(1, 2, 1)

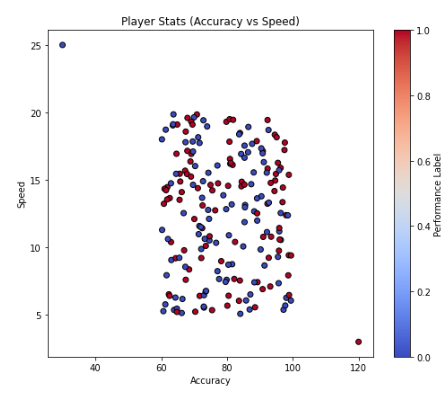
plt.scatter(features[:, 0], features[:, 1], c=labels, cmap='coolwarm', edgecolor='k')

plt.xlabel('Accuracy')

plt.ylabel('Speed')

plt.title('Player Stats (Accuracy vs Speed)')

plt.colorbar(label='Performance Label')



*# Visualize the decision boundary for the first two features (Accuracy vs Speed)*

from matplotlib.colors import ListedColormap

h = 0.5 # Step size in the mesh

x\_min, x\_max = features[:, 0].min() - 1, features[:, 0].max() + 1

y\_min, y\_max = features[:, 1].min() - 1, features[:, 1].max() + 1

xx, yy = np.meshgrid(np.arange(x\_min, x\_max, h), np.arange(y\_min, y\_max, h))

*# Predict for the grid using only the first two features*

Z = knn.predict(np.c\_[xx.ravel(), yy.ravel(), np.full(xx.ravel().shape, np.mean(features[:, 2])), np.full(xx.ravel().shape, np.mean(features[:, 3]))])

Z = Z.reshape(xx.shape)

plt.subplot(1, 2, 2)

plt.contourf(xx, yy, Z, alpha=0.8, cmap=ListedColormap(['#FFAAAA', '#AAFFAA']))

plt.scatter(features[:, 0], features[:, 1], c=labels, edgecolor='k', cmap='coolwarm')

plt.xlabel('Accuracy')

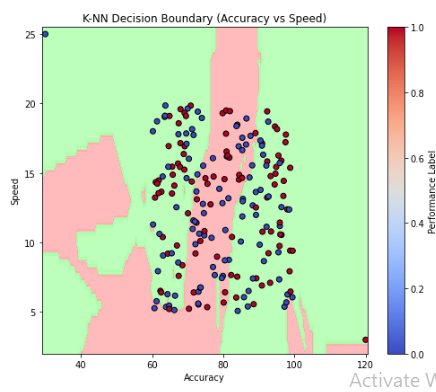
plt.ylabel('Speed')

plt.title('K-NN Decision Boundary (Accuracy vs Speed)')

plt.colorbar(label='Performance Label')

plt.tight\_layout()

plt.show()



**Result:-**

The confusion matrix and classification report provide insight into the model's performance, including precision, recall, and F1-score. Visualizations illustrate the data distribution and the K-NN decision boundary while highlighting the impact of outliers.