Logical regression:

Accuracy is 0.91

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

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

from sklearn.linear\_model import LogisticRegression

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

# Load your existing dataset

existing\_data = pd.read\_excel(r"C:\Users\Hi\OneDrive\Desktop\rnsit\8th sem\final year project\dry eyes dataset\augmented\_dataset1222.xlsx")

# Prepare features (X) and target variable (y)

X = existing\_data[[ 'age', 'q1', 'q2', 'q3', 'q4', 'q5', 'q6', 'q7', 'q8', 'q9', 'q10', 'q11', 'q12','Score','Score\_code']]

y = existing\_data['Score\_code']

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Train Logistic Regression model

log\_reg\_model = LogisticRegression()

log\_reg\_model.fit(X\_train, y\_train)

# Predict on the testing set

y\_pred = log\_reg\_model.predict(X\_test)

# Calculate accuracy

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

print("Accuracy:", accuracy)

# Plot a line graph for accuracy

plt.plot([0, 1], [accuracy, accuracy], linestyle='--', marker='o')

plt.title('Accuracy of Logistic Regression Model')

plt.xlabel('Iterations')

plt.ylabel('Accuracy')

plt.xticks([0, 1], ['Iteration 1', 'Iteration 2'])

plt.ylim(0, 1) # Set y-axis limits

plt.grid(True)

plt.savefig('accuracy\_line\_graph\_lr.png') # Save the line graph as an image

plt.show()

# Calculate confusion matrix

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

# Plot confusion matrix

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

sns.heatmap(conf\_matrix, annot=True, cmap='Blues', fmt='g')

plt.xlabel('Predicted Label')

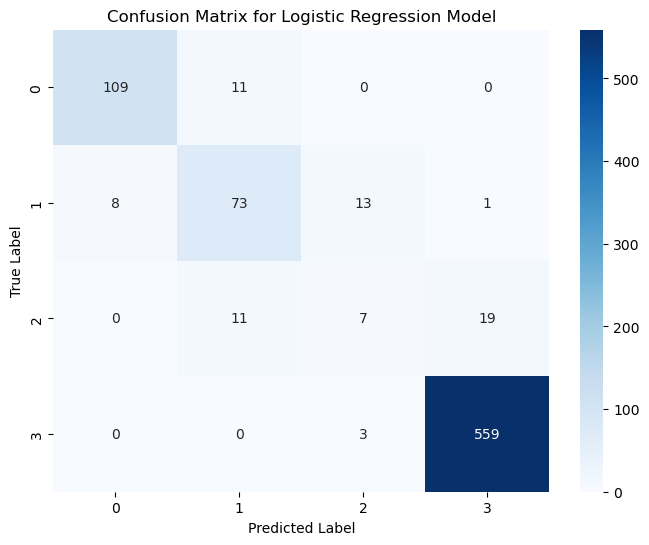
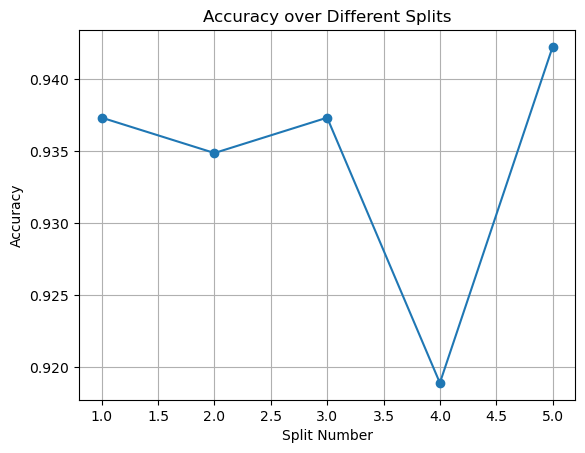
plt.ylabel('True Label')

plt.title('Confusion Matrix for Logistic Regression Model')

plt.savefig('confusion\_matrix\_lr.png') # Save the confusion matrix as an image

plt.show()

line graph and confusion matrix:



KNN algorithm:

Accuracy is 0.95

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

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

from sklearn.neighbors import KNeighborsClassifier

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

# Load your existing dataset

existing\_data = pd.read\_excel(r"C:\Users\Hi\OneDrive\Desktop\rnsit\8th sem\final year project\dry eyes dataset\augmented\_dataset1222.xlsx")

# Take a smaller sample of the data

sample\_size = 100 # You can adjust this as needed

existing\_data\_sample = existing\_data.sample(n=sample\_size, random\_state=42)

# Prepare features (X) and target variable (y)

X = existing\_data\_sample[['age', 'q1', 'q2', 'q3', 'q4', 'q5', 'q6', 'q7', 'q8', 'q9', 'q10', 'q11', 'q12', 'Score']]

y = existing\_data\_sample['Score\_code']

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Initialize KNN classifier with k=3 (you can adjust k as needed)

knn = KNeighborsClassifier(n\_neighbors=3)

# Train the KNN classifier

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

# Predict on the testing set

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

# Calculate accuracy

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

print("KNN Accuracy:", accuracy)

# Plot line graph

plt.figure(figsize=(10, 6))

plt.plot(y\_test.values, label='Actual')

plt.plot(y\_pred, label='Predicted')

plt.title('Actual vs Predicted Scores')

plt.xlabel('Index')

plt.ylabel('Score')

plt.legend()

plt.show()

# Plot confusion matrix

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

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

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues', cbar=False)

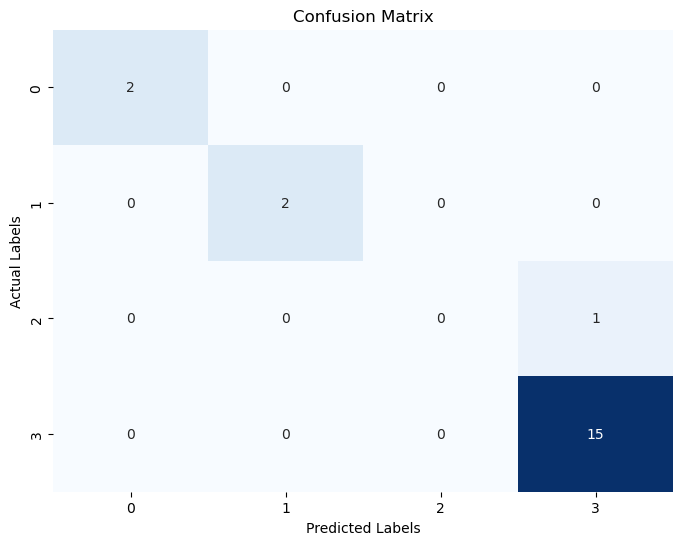
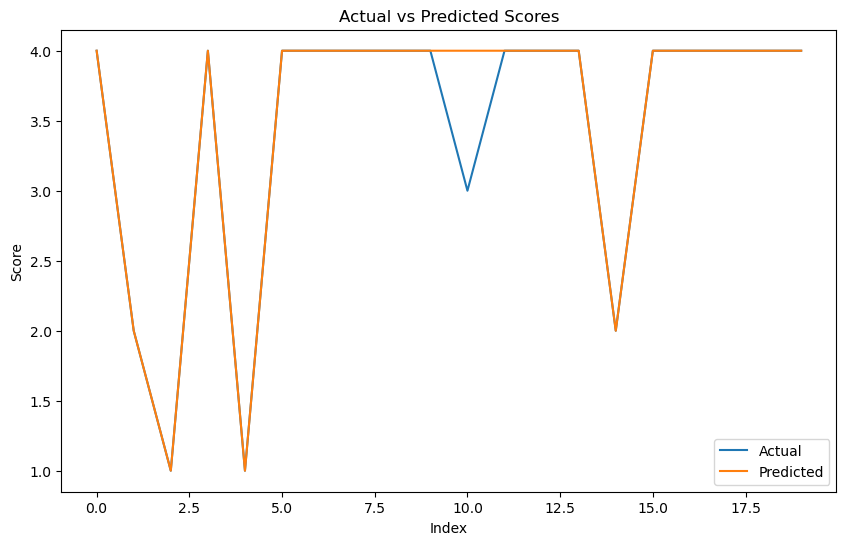
plt.title('Confusion Matrix')

plt.xlabel('Predicted Labels')

plt.ylabel('Actual Labels')

plt.show()

Line graph and confusion matrix:



Random forest:

Accuracy is 0.95.

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

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

from sklearn.ensemble import RandomForestClassifier

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

# Load your existing dataset

existing\_data = pd.read\_excel(r"C:\Users\Hi\OneDrive\Desktop\rnsit\8th sem\final year project\dry eyes dataset\augmented\_dataset1222.xlsx")

# Take a smaller sample of the data

sample\_size = 100 # You can adjust this as needed

existing\_data\_sample = existing\_data.sample(n=sample\_size, random\_state=42)

# Prepare features (X) and target variable (y)

X = existing\_data\_sample[['age', 'q1', 'q2', 'q3', 'q4', 'q5', 'q6', 'q7', 'q8', 'q9', 'q10', 'q11', 'q12', 'Score']]

y = existing\_data\_sample['Score\_code']

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Initialize Random Forest classifier

rf\_classifier = RandomForestClassifier(n\_estimators=100, random\_state=42)

# Train the Random Forest classifier

rf\_classifier.fit(X\_train, y\_train)

# Predict on the testing set

y\_pred = rf\_classifier.predict(X\_test)

# Calculate accuracy

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

print("Random Forest Accuracy:", accuracy)

# Plot line graph

plt.figure(figsize=(10, 6))

plt.plot(y\_test.values, label='Actual')

plt.plot(y\_pred, label='Predicted')

plt.title('Actual vs Predicted Scores')

plt.xlabel('Index')

plt.ylabel('Score')

plt.legend()

plt.show()

# Plot confusion matrix

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

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

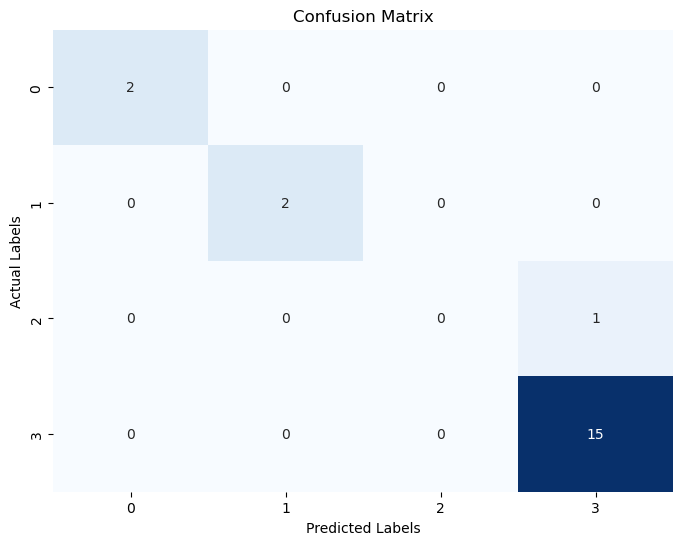
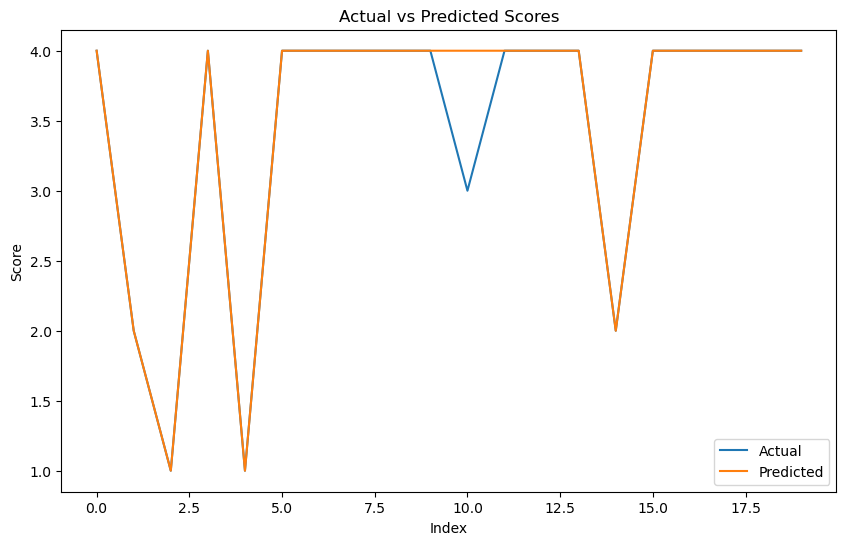
sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues', cbar=False)

plt.title('Confusion Matrix')

plt.xlabel('Predicted Labels')

plt.ylabel('Actual Labels')

plt.show()

line graph and confusion matrix:

svm:

accuracy is 0.95.

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

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

from sklearn.svm import SVC

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

# Load your existing dataset

existing\_data = pd.read\_excel(r"C:\Users\Hi\OneDrive\Desktop\rnsit\8th sem\final year project\dry eyes dataset\augmented\_dataset1222.xlsx")

# Take a smaller sample of the data

sample\_size = 100 # You can adjust this as needed

existing\_data\_sample = existing\_data.sample(n=sample\_size, random\_state=42)

# Prepare features (X) and target variable (y)

X = existing\_data\_sample[['age', 'q1', 'q2', 'q3', 'q4', 'q5', 'q6', 'q7', 'q8', 'q9', 'q10', 'q11', 'q12', 'Score']]

y = existing\_data\_sample['Score\_code']

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Initialize SVM classifier

svm\_classifier = SVC(kernel='linear', random\_state=42)

# Train the SVM classifier

svm\_classifier.fit(X\_train, y\_train)

# Predict on the testing set

y\_pred = svm\_classifier.predict(X\_test)

# Calculate accuracy

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

print("SVM Accuracy:", accuracy)

# Plot line graph

plt.figure(figsize=(10, 6))

plt.plot(y\_test.values, label='Actual')

plt.plot(y\_pred, label='Predicted')

plt.title('Actual vs Predicted Scores')

plt.xlabel('Index')

plt.ylabel('Score')

plt.legend()

plt.show()

# Plot confusion matrix

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

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

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues', cbar=False)

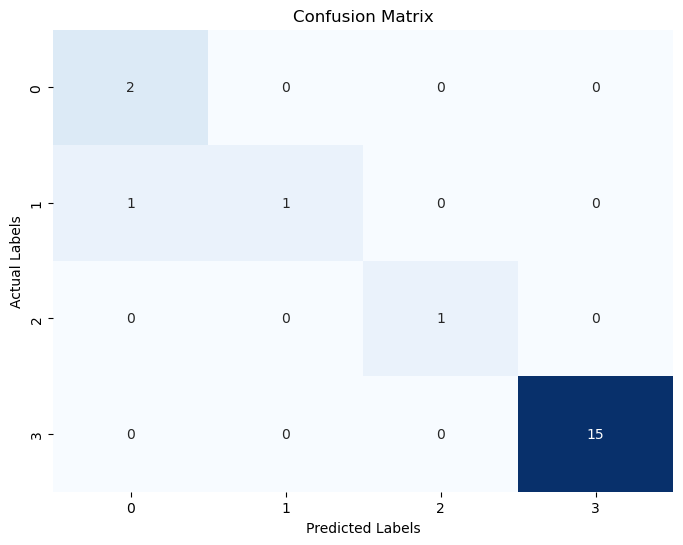
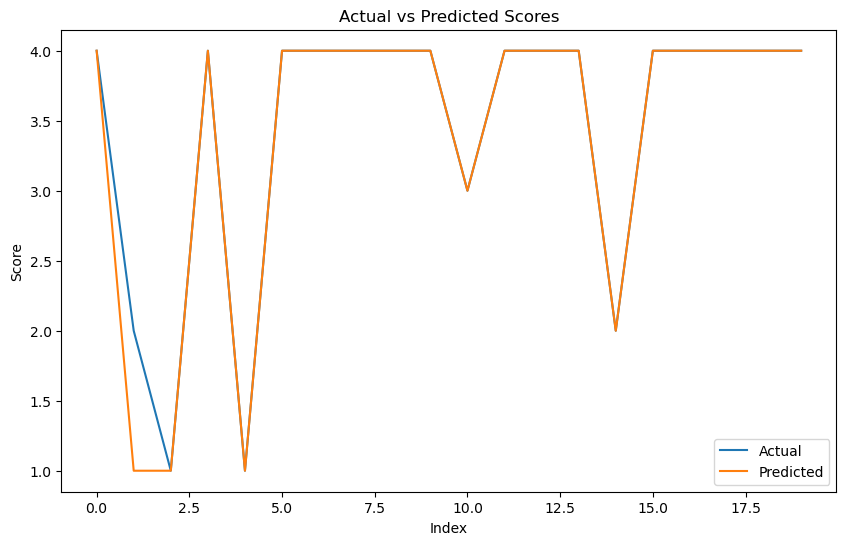
plt.title('Confusion Matrix')

plt.xlabel('Predicted Labels')

plt.ylabel('Actual Labels')

plt.show()

line graph and confusion matrix:



Ensemble model:

Accuracy is 0.95.

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

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

from sklearn.ensemble import VotingClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.svm import SVC

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

# Load your existing dataset

existing\_data = pd.read\_excel(r"C:\Users\Hi\OneDrive\Desktop\rnsit\8th sem\final year project\dry eyes\dry eyes dataset\augmented\_dataset1222.xlsx")

# Take a smaller sample of the data

sample\_size = 1000 # You can adjust this as needed

existing\_data\_sample = existing\_data.sample(n=sample\_size, random\_state=42)

# Prepare features (X) and target variable (y)

X = existing\_data\_sample[['age', 'q1', 'q2', 'q3', 'q4', 'q5', 'q6', 'q7', 'q8', 'q9', 'q10', 'q11', 'q12', 'Score']]

y = existing\_data\_sample['Score\_code']

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Initialize individual classifiers

log\_reg\_model = LogisticRegression()

knn\_classifier = KNeighborsClassifier()

rf\_classifier = RandomForestClassifier()

svm\_classifier = SVC(kernel='linear')

# Create an ensemble model using VotingClassifier

ensemble\_model = VotingClassifier(estimators=[

('logistic', log\_reg\_model),

('knn', knn\_classifier),

('random\_forest', rf\_classifier),

('svm', svm\_classifier)

], voting='hard')

# Train the ensemble model

ensemble\_model.fit(X\_train, y\_train)

# Predict on the testing set

y\_pred = ensemble\_model.predict(X\_test)

# Calculate accuracy

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

print("Ensemble Model Accuracy:", accuracy)

# Plot confusion matrix

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

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

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues', cbar=False)

plt.title('Confusion Matrix')

plt.xlabel('Predicted Labels')

plt.ylabel('Actual Labels')

plt.show()

line graph and confusion matrix

