Student Performance Analysis - Classification based Project

In [47]:
```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

In [48]:
```
# Importing the dataset
```

In [83]:
```
data = pd.read_csv("D:/HelloTech Softwares - Data Science Intern Projects/Student Performance dataset -
```

In [ ]:
```
# Display first 10 rows
```

In [19]:
```
data.head(10)
```

Out[19]:

	StudentID	Age	Gender	Ethnicity	ParentalEducation	StudyTimeWeekly	Absences	Tutoring	ParentalSupport	Extracurricular
0	1001	17	1	0	2	19.833723	7	1	2	0
1	1002	18	0	0	1	15.408756	0	0	1	0
2	1003	15	0	2	3	4.210570	26	0	2	0
3	1004	17	1	0	3	10.028829	14	0	3	1
4	1005	17	1	0	2	4.672495	17	1	3	0
5	1006	18	0	0	1	8.191219	0	0	1	1
6	1007	15	0	1	1	15.601680	10	0	3	0
7	1008	15	1	1	4	15.424496	22	1	1	1
8	1009	17	0	0	0	4.562008	1	0	2	0
9	1010	16	1	0	1	18.444466	0	0	3	1

In [ ]:
```
# Display last 10 rows
```

In [20]:
```
data.tail(10)
```

Out[20]:

	StudentID	Age	Gender	Ethnicity	ParentalEducation	StudyTimeWeekly	Absences	Tutoring	ParentalSupport	Extracurric
2382	3383	16	0	0	3	13.941823	20	0	2	
2383	3384	16	1	2	2	11.736409	18	1	4	
2384	3385	15	1	0	1	16.655581	13	1	3	
2385	3386	16	1	0	1	1.445434	20	0	3	
2386	3387	16	0	0	2	13.814021	14	0	2	
2387	3388	18	1	0	3	10.680555	2	0	4	
2388	3389	17	0	0	1	7.583217	4	1	4	
2389	3390	16	1	0	2	6.805500	20	0	2	
2390	3391	16	1	1	0	12.416653	17	0	2	
2391	3392	16	1	0	2	17.819907	13	0	2	

In [ ]:
```
# Finding the Total no. of rows and columns
```

In [21]:
```
data.shape
```

Out[21]:
```
(2392, 15)
```

Data Preprocessing

In [ ]:

```
# Checking null values
```

In [22]:

```
data.isnull().sum()
```

Out[22]:

```
StudentID            0
Age                  0
Gender               0
Ethnicity            0
ParentalEducation    0
StudyTimeWeekly      0
Absences             0
Tutoring             0
ParentalSupport      0
Extracurricular      0
Sports               0
Music                0
Volunteering         0
GPA                  0
GradeClass           0
dtype: int64
```

In [ ]:

```
# Removing empty rows
```

In [23]:

```
data.dropna(inplace=True)
```

In [24]:

```
data.shape # Size of rows and columns after removing empty rows .
```

Out[24]:

```
(2392, 15)
```

In [25]:

```
### Feature Scaling
```

In [26]:

```
numerical_features = ['Age','StudyTimeWeekly','Absences','GPA']

scaler = StandardScaler()
```

In [27]:

```
data[numerical_features] = scaler.fit_transform(data[numerical_features])
data.head(10)
```

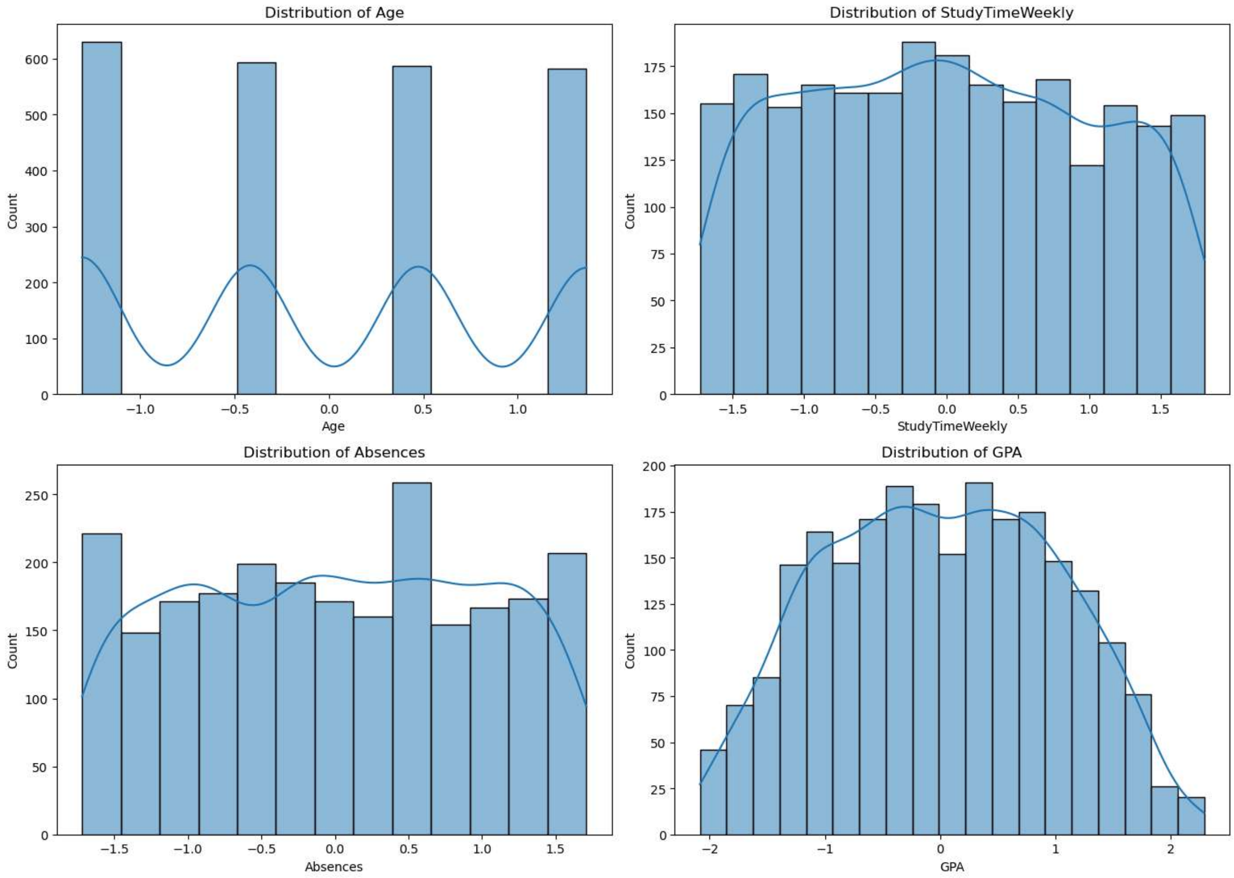
Out[27]:

	StudentID	Age	Gender	Ethnicity	ParentalEducation	StudyTimeWeekly	Absences	Tutoring	ParentalSupport	Extracurr
0	1001	0.472919	1	0	2	1.780336	-0.890822	1	2	
1	1002	1.362944	0	0	1	0.997376	-1.717694	0	1	
2	1003	-1.307132	0	2	3	-0.984045	1.353542	0	2	
3	1004	0.472919	1	0	3	0.045445	-0.063951	0	3	
4	1005	0.472919	1	0	2	-0.902311	0.290422	1	3	
5	1006	1.362944	0	0	1	-0.279704	-1.717694	0	1	
6	1007	-1.307132	0	1	1	1.031513	-0.536449	0	3	
7	1008	-1.307132	1	1	4	1.000161	0.881045	1	1	
8	1009	0.472919	0	0	0	-0.921861	-1.599569	0	2	
9	1010	-0.417106	1	0	1	1.534519	-1.717694	0	3	

Exploratory Data Analysis (EDA)

In [28]:

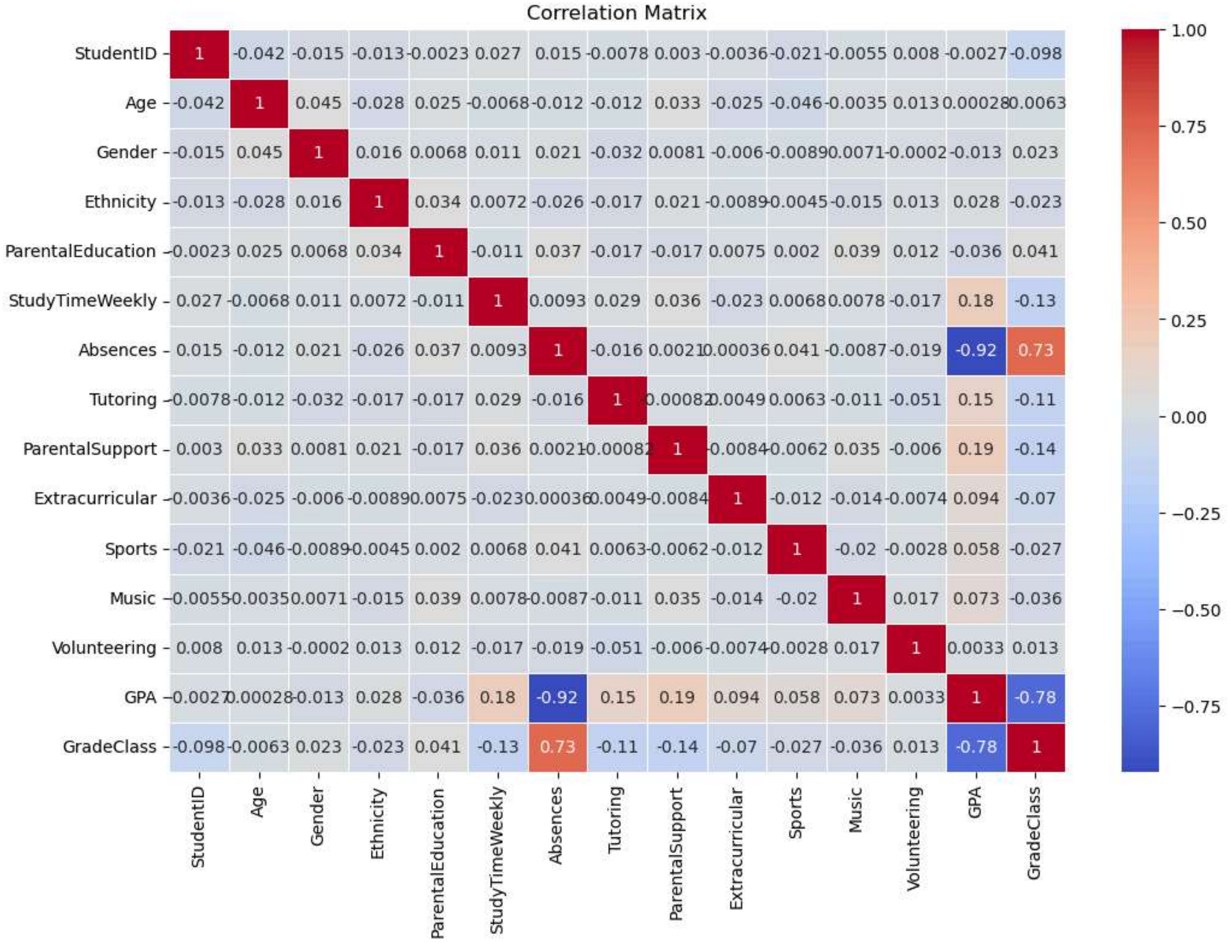
```
plt.figure(figsize=(14,10))
for i,feature in enumerate(numerical_features,1):
    plt.subplot(2,2,i)
    sns.histplot(data[feature],kde=True)
    plt.title(f'Distribution of {feature}')
plt.tight_layout()
plt.show()
```



In [29]:

```
### Correlation Matrix
```

In [30]:
```
plt.figure(figsize=(12,8))
corr_matrix = data.corr()
sns.heatmap(corr_matrix,annot = True,cmap='coolwarm',linewidths=0.5)
plt.title('Correlation Matrix')
plt.show()
```

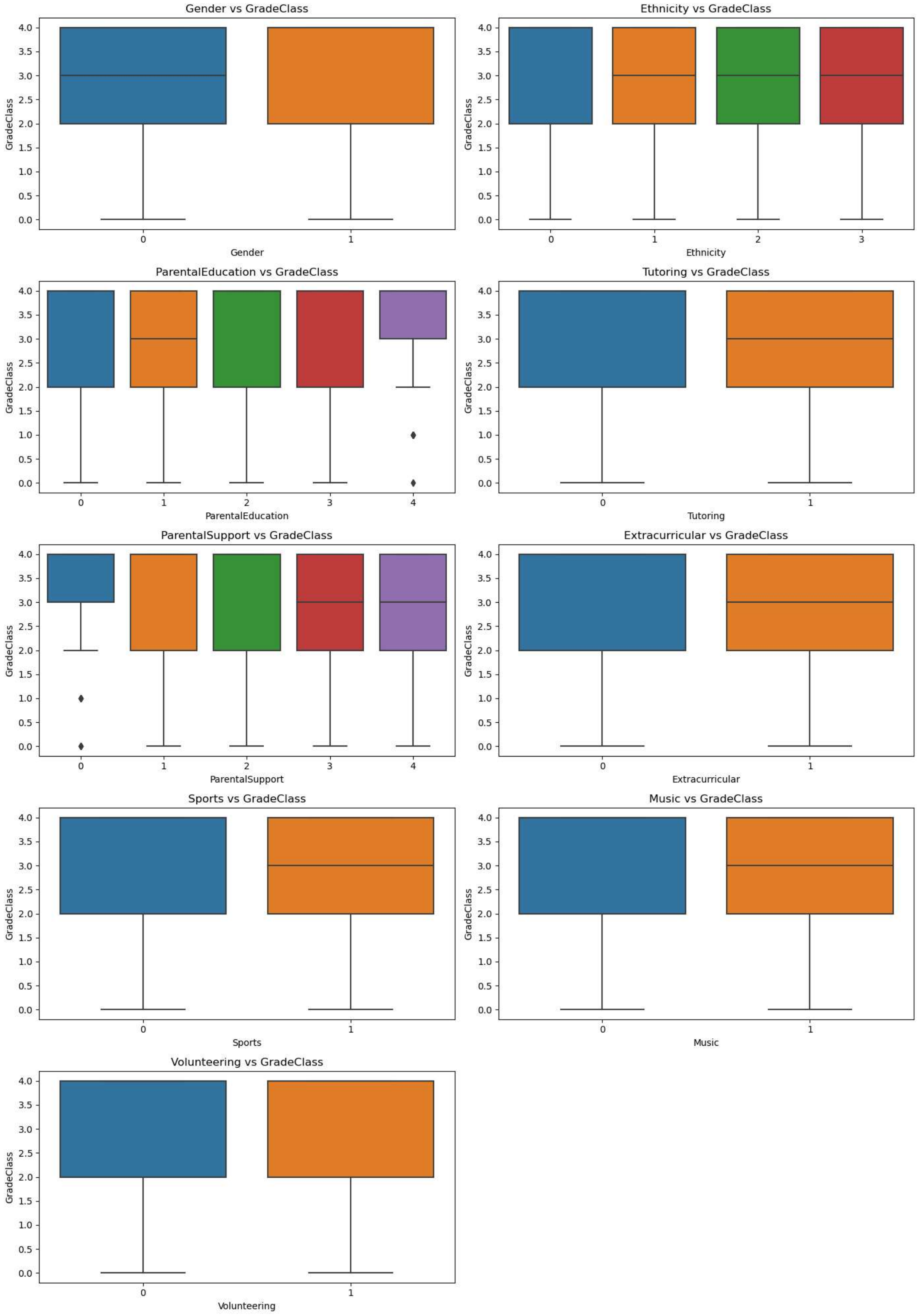


	StudentID	Age	Gender	Ethnicity	ParentalEducation	StudyTimeWeekly	Absences	Tutoring	ParentalSupport	Extracurricular	Sports	Music	Volunteering	GPA	GradeClass
StudentID	1	-0.042	-0.015	-0.013	-0.0023	0.027	0.015	-0.0078	0.003	-0.0036	-0.021	-0.0055	0.008	-0.0027	-0.098
Age	-0.042	1	0.045	-0.028	0.025	-0.0068	-0.012	-0.012	0.033	-0.025	-0.046	-0.0035	0.013	0.00028	0.0063
Gender	-0.015	0.045	1	0.016	0.0068	0.011	0.021	-0.032	0.0081	-0.006	-0.0089	0.0071	-0.0002	-0.013	0.023
Ethnicity	-0.013	-0.028	0.016	1	0.034	0.0072	-0.026	-0.017	0.021	-0.0089	-0.0045	-0.015	0.013	0.028	-0.023
ParentalEducation	-0.0023	0.025	0.0068	0.034	1	-0.011	0.037	-0.017	-0.017	0.0075	0.002	0.039	0.012	-0.036	0.041
StudyTimeWeekly	0.027	-0.0068	0.011	0.0072	-0.011	1	0.0093	0.029	0.036	-0.023	0.0068	0.0078	-0.017	0.18	-0.13
Absences	0.015	-0.012	0.021	-0.026	0.037	0.0093	1	-0.016	0.0021	0.00036	0.041	-0.0087	-0.019	-0.92	0.73
Tutoring	-0.0078	-0.012	-0.032	-0.017	-0.017	0.029	-0.016	1	0.00082	0.0049	0.0063	-0.011	-0.051	0.15	-0.11
ParentalSupport	0.003	0.033	0.0081	0.021	-0.017	0.036	0.0021	-0.00082	1	-0.0084	-0.0062	0.035	-0.006	0.19	-0.14
Extracurricular	-0.0036	-0.025	-0.006	-0.0089	0.0075	-0.023	0.00036	0.0049	-0.0084	1	-0.012	-0.014	-0.0074	0.094	-0.07
Sports	-0.021	-0.046	-0.0089	-0.0045	0.002	0.0068	0.041	0.0063	-0.0062	-0.012	1	-0.02	-0.0028	0.058	-0.027
Music	-0.0055	0.0035	0.0071	-0.015	0.039	0.0078	-0.0087	-0.011	0.035	-0.014	-0.02	1	0.017	0.073	-0.036
Volunteering	0.008	0.013	-0.0002	0.013	0.012	-0.017	-0.019	-0.051	-0.006	-0.0074	-0.0028	0.017	1	0.0033	0.013
GPA	-0.0027	0.00028	-0.013	0.028	-0.036	0.18	-0.92	0.15	0.19	0.094	0.058	0.073	0.0033	1	-0.78
GradeClass	-0.098	-0.0063	0.023	-0.023	0.041	-0.13	0.73	-0.11	-0.14	-0.07	-0.027	-0.036	0.013	-0.78	1

In [31]:
```
### Box plots for Categorical Features
```

In [33]:

```
categorical_features = ['Gender','Ethnicity','ParentalEducation','Tutoring','ParentalSupport','Extracur

plt.figure(figsize=(14,20))
for i,feature in enumerate(categorical_features,1):
    plt.subplot(5,2,i)
    sns.boxplot(x=data[feature],y=data['GradeClass'])
    plt.title(f'{feature} vs GradeClass')
plt.tight_layout()
plt.show()
```



Model Selection and Training

In [34]:

```
### Split the data
```

In [35]:
```
x=data.drop(['StudentID','GradeClass'],axis=1)
y=data['GradeClass']
```

In [36]:
```
### Splitting the data into training and testing sets
```

In [37]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2,random_state=42)
```

### Random Forest Classifier

In [50]:
```
from sklearn.ensemble import RandomForestClassifier
RFC = RandomForestClassifier(random_state=42)
RFC.fit(x_train,y_train)
```

Out[50]:
```
▾         RandomForestClassifier
RandomForestClassifier(random_state=42)
```

### Logistic Regression

In [49]:
```
from sklearn.linear_model import LogisticRegression
LR = LogisticRegression(random_state=42,max_iter=1000)  # max_iter avoids the warning
LR.fit(x_train,y_train)
```

Out[49]:
```
▾                 LogisticRegression
LogisticRegression(max_iter=1000, random_state=42)
```

### Support Vector Machine (SVM)

In [53]:
```
from sklearn.svm import SVC as SVM
SVM = SVM(random_state = 42)
SVM.fit(x_train, y_train)
```

Out[53]:
```
▾        SVC
SVC(random_state=42)
```

### K-Nearest Neigbour (KNN)

In [54]:
```
from sklearn.neighbors import KNeighborsClassifier as KNN

kNN = KNN()
kNN.fit(x_train, y_train)
```

Out[54]:
```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

### Decision Tree Classifier

In [58]:
```
from sklearn.tree import DecisionTreeClassifier

DTC = DecisionTreeClassifier(random_state=42)
DTC.fit(x_train, y_train)
```

Out[58]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

Model Evaluation

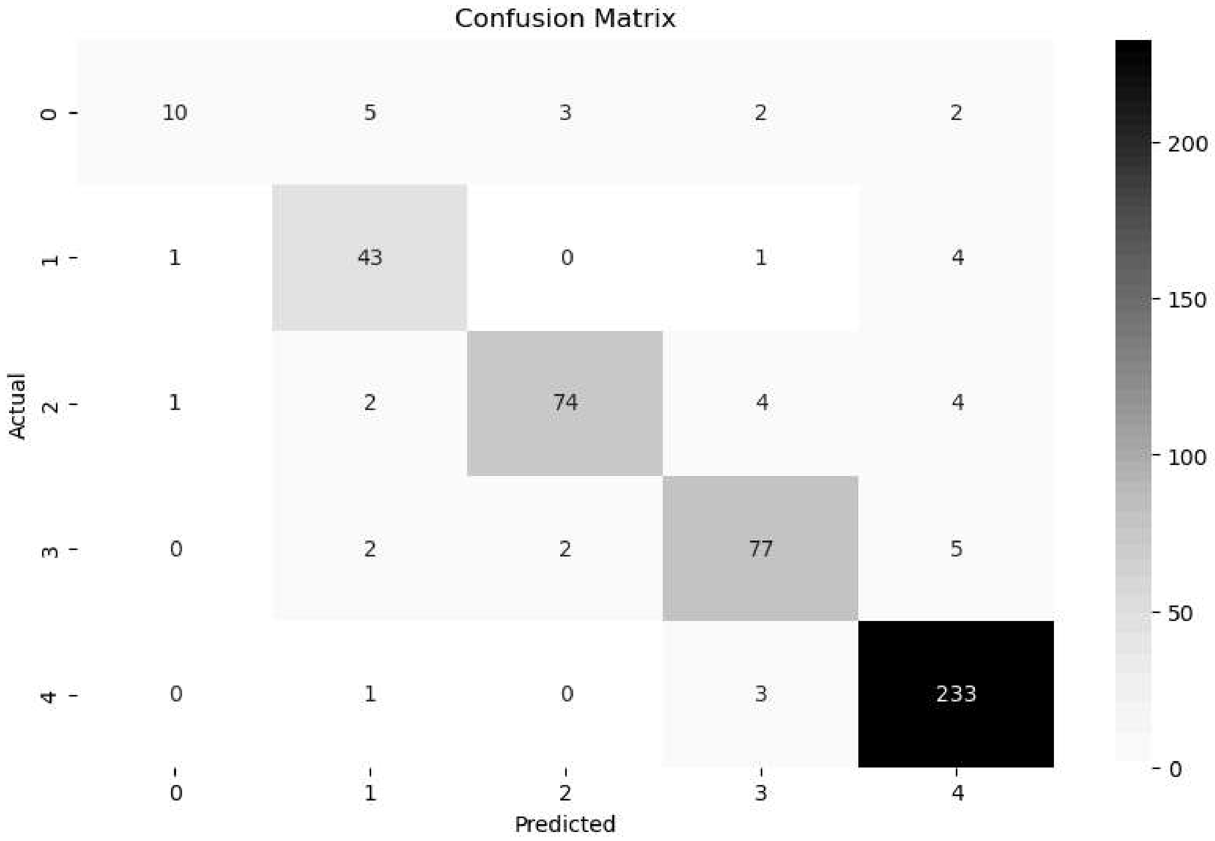
Evaluate the Model

i. Random Forest Classifier

In [75]:

```
# Make predictions
y_pred = RFC.predict(x_test)

# Accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')

# Classificatiion Report
print("Classification Report")
print(classification_report(y_test, y_pred))

# Confusion Matrix
cm = confusion_matrix(y_test,y_pred)
plt.figure(figsize=(10,6))
sns.heatmap(cm,annot=True, fmt='d', cmap='Greys')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```

```
Accuracy: 0.91
Classification Report
              precision    recall  f1-score   support

         0.0       0.83      0.45      0.59        22
         1.0       0.81      0.88      0.84        49
         2.0       0.94      0.87      0.90        85
         3.0       0.89      0.90      0.89        86
         4.0       0.94      0.98      0.96       237

    accuracy                           0.91       479
   macro avg       0.88      0.82      0.84       479
weighted avg       0.91      0.91      0.91       479
```

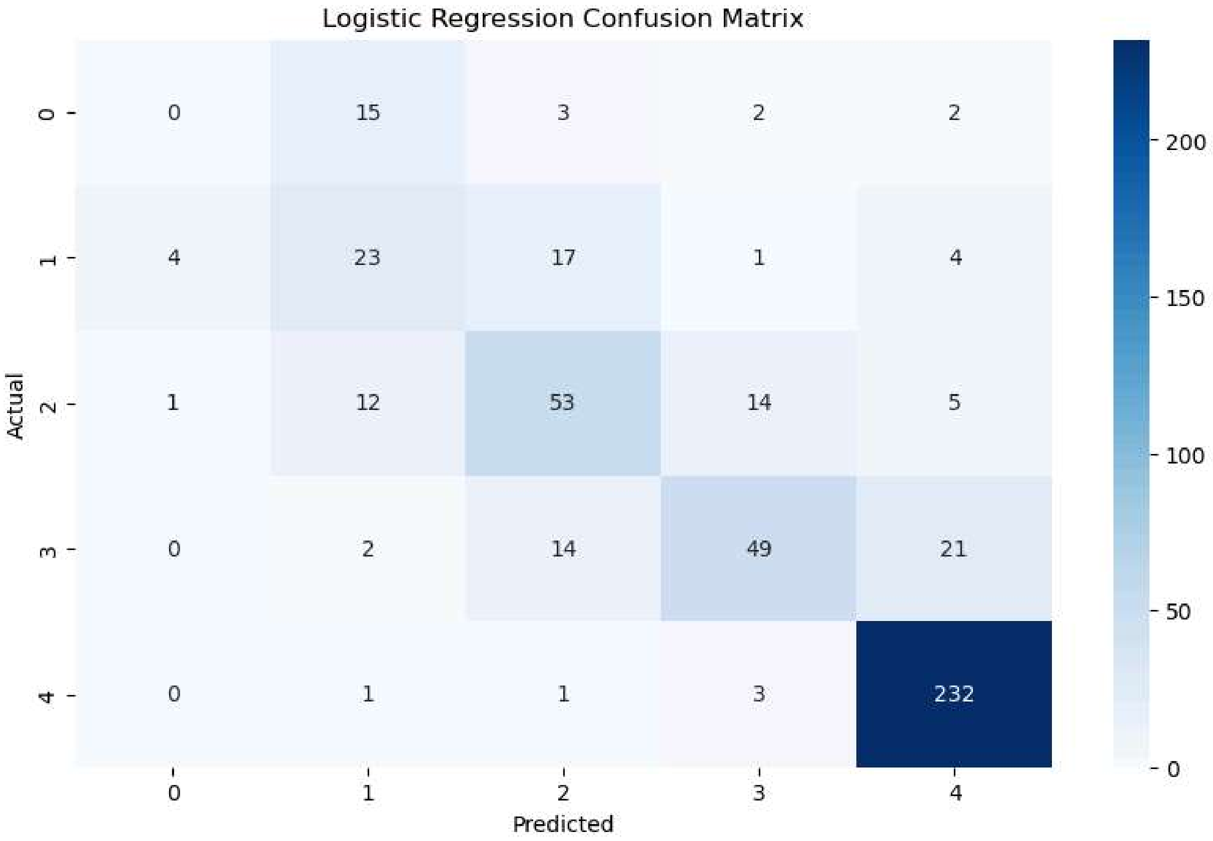


In [ ]:

```
ii. Logistic Regression
```

In [74]:
```
# Make prediction
y_pred=LR.predict(x_test)

# Accuracy
accuracy = accuracy_score(y_test,y_pred)
print(f'Logistic Regression Accuracy: {accuracy:.2f}')

# Classification Report
print("Logistic Regression Classification Report")
print(classification_report(y_test,y_pred))

# Confusion Matrix
cm = confusion_matrix(y_test,y_pred)
plt.figure(figsize=(10,6))
sns.heatmap(cm,annot=True, fmt='d', cmap='Blues')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Logistic Regression Confusion Matrix")
plt.show()
```

```
Logistic Regression Accuracy: 0.75
Logistic Regression Classification Report
              precision    recall  f1-score   support

         0.0       0.00      0.00      0.00        22
         1.0       0.43      0.47      0.45        49
         2.0       0.60      0.62      0.61        85
         3.0       0.71      0.57      0.63        86
         4.0       0.88      0.98      0.93       237

    accuracy                           0.75       479
   macro avg       0.53      0.53      0.52       479
weighted avg       0.71      0.75      0.73       479
```



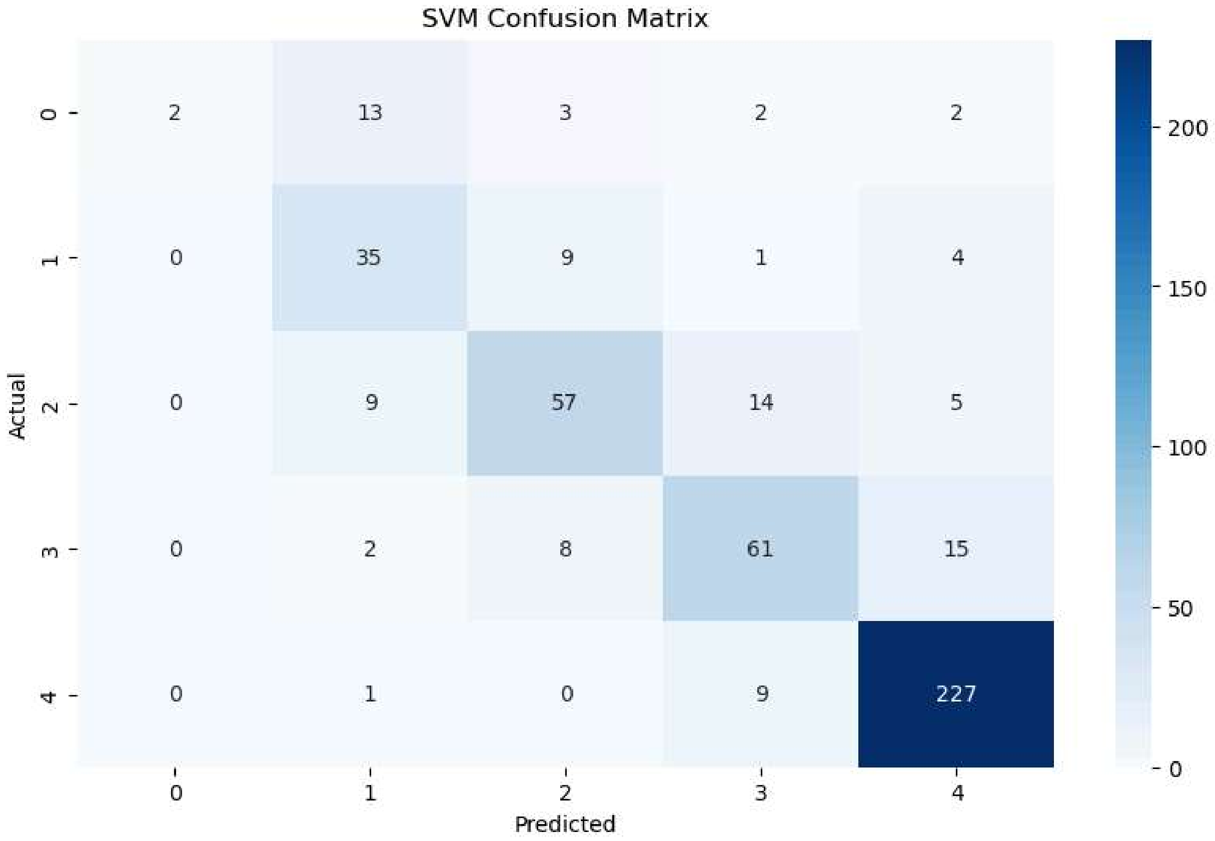
In [ ]:

iii. Support Vector Machine (SVM)

In [81]:

```
# Make prediction
y_pred=SVM.predict(x_test)

# Accuracy
accuracy = accuracy_score(y_test,y_pred)
print(f'SVM Accuracy: {accuracy:.2f}')

# Classification Report
print("SVM Classification Report")
print(classification_report(y_test,y_pred))

# Confusion Matrix
cm = confusion_matrix(y_test,y_pred)
plt.figure(figsize=(10,6))
sns.heatmap(cm,annot=True, fmt='d', cmap='Blues')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("SVM Confusion Matrix")
plt.show()
```

```
SVM Accuracy: 0.80
SVM Classification Report
              precision    recall  f1-score   support

         0.0       1.00      0.09      0.17        22
         1.0       0.58      0.71      0.64        49
         2.0       0.74      0.67      0.70        85
         3.0       0.70      0.71      0.71        86
         4.0       0.90      0.96      0.93       237

    accuracy                           0.80       479
   macro avg       0.78      0.63      0.63       479
weighted avg       0.81      0.80      0.78       479
```



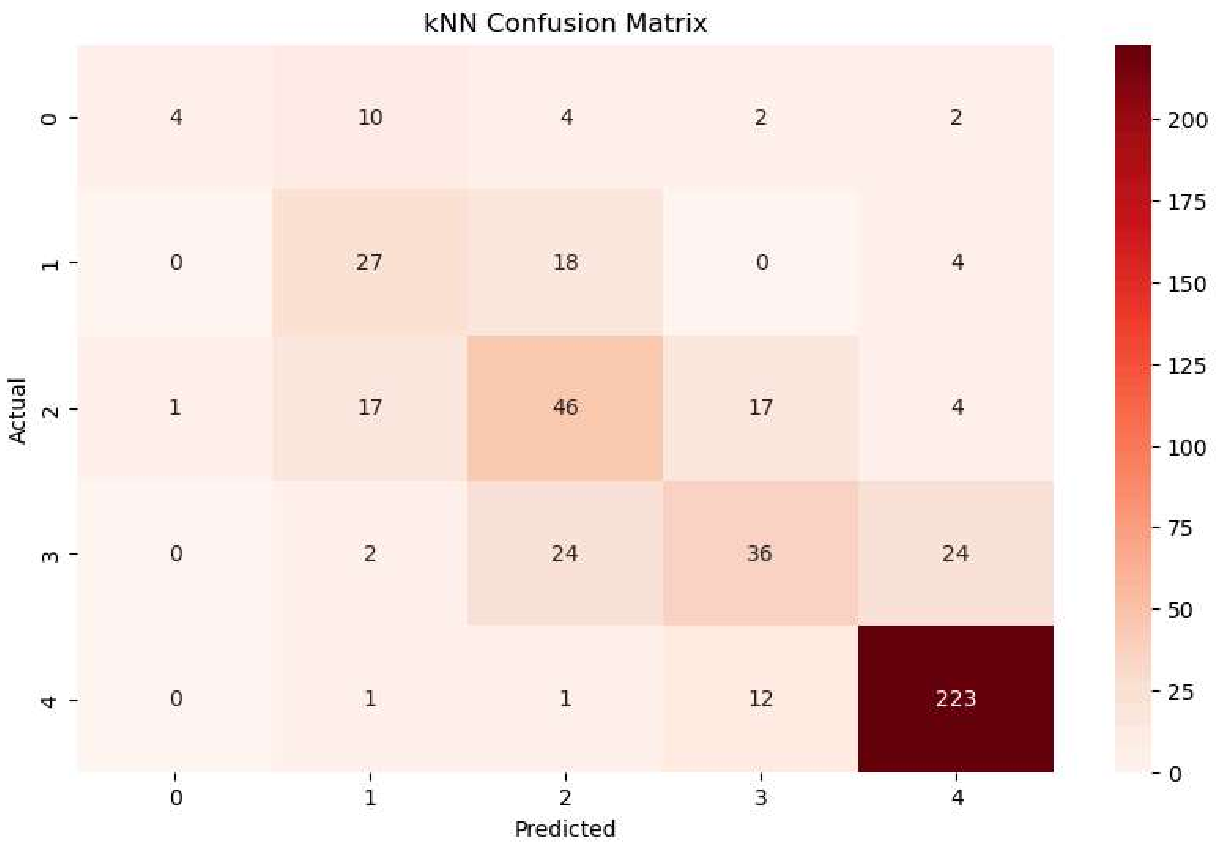
In [ ]:

iv. k - Nearest Neighbors

In [66]:

```
# Make prediction
y_pred=kNN.predict(x_test)

# Accuracy
accuracy = accuracy_score(y_test,y_pred)
print(f'kNN Accuracy: {accuracy:.2f}')

# Classification Report
print("kNN Classification Report")
print(classification_report(y_test,y_pred))

# Confusion Matrix
cm = confusion_matrix(y_test,y_pred)
plt.figure(figsize=(10,6))
sns.heatmap(cm,annot=True, fmt='d', cmap='Reds')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("kNN Confusion Matrix")
plt.show()
```

```
kNN Accuracy: 0.70
kNN Classification Report
              precision    recall  f1-score   support

         0.0       0.80      0.18      0.30        22
         1.0       0.47      0.55      0.51        49
         2.0       0.49      0.54      0.52        85
         3.0       0.54      0.42      0.47        86
         4.0       0.87      0.94      0.90       237

    accuracy                           0.70       479
   macro avg       0.63      0.53      0.54       479
weighted avg       0.70      0.70      0.69       479
```



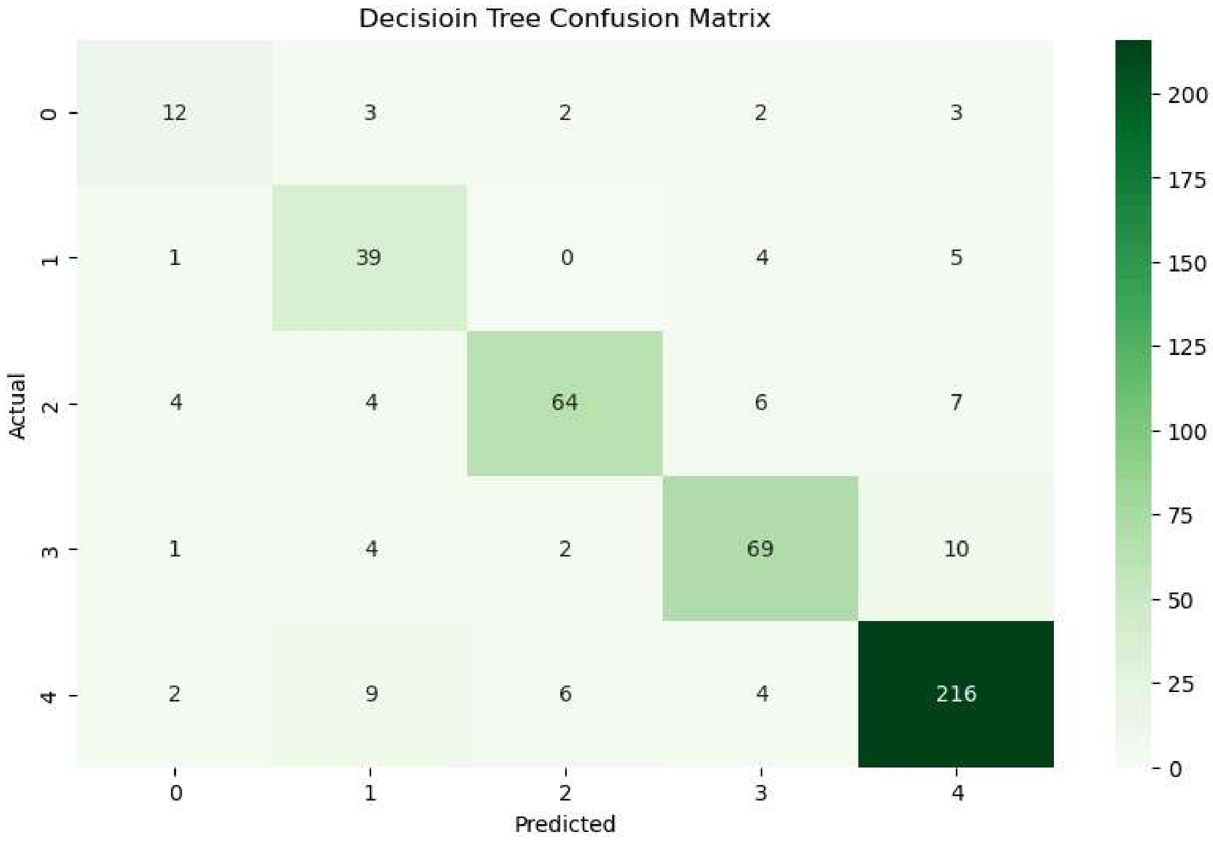
In [ ]:

v. Decision Tree Classifier

In [67]:

```
# Make prediction
y_pred=DTC.predict(x_test)

# Accuracy
accuracy = accuracy_score(y_test,y_pred)
print(f'Decison Tree Accuracy: {accuracy:.2f}')

# Classification Report
print("Decision Tree Classification Report")
print(classification_report(y_test,y_pred))

# Confusion Matrix
cm = confusion_matrix(y_test,y_pred)
plt.figure(figsize=(10,6))
sns.heatmap(cm,annot=True, fmt='d', cmap='Greens')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Decisioin Tree Confusion Matrix")
plt.show()
```

```
Decison Tree Accuracy: 0.84
Decision Tree Classification Report
              precision    recall  f1-score   support

         0.0       0.60      0.55      0.57        22
         1.0       0.66      0.80      0.72        49
         2.0       0.86      0.75      0.81        85
         3.0       0.81      0.80      0.81        86
         4.0       0.90      0.91      0.90       237

    accuracy                           0.84       479
   macro avg       0.77      0.76      0.76       479
weighted avg       0.84      0.84      0.84       479
```



In [ ]: