```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.decomposition import PCA
from sklearn.metrics import classification_report, confusion_matrix
import plotly.express as px

# Step 1: Extract - Load Data
data = pd.read_csv('athlete_performance.csv')

# Step 2: Transform - Data Cleaning
data.drop_duplicates(inplace=True) # Remove duplicates
numeric_columns = data.select_dtypes(include=[np.number]).columns
data[numeric_columns] = data[numeric_columns].fillna(data[numeric_columns].mean())
non_numeric_columns = data.select_dtypes(exclude=[np.number]).columns
data[non_numeric_columns] = data[non_numeric_columns].fillna('Unknown')

# Step 3: Data Transformation - Create derived attributes
data['Average_Score'] = data.groupby('Athlete')['Score'].transform('mean')
data['Score_Per_Hour'] = data['Score'] / data['Training_Hours']
data['Ranking'] = data['Average_Score'].rank(ascending=False)

# Display average score for each athlete
average_scores = data[['Athlete', 'Average_Score']].drop_duplicates().sort_values(by='Average_Score',
ascending=False)
print("Average Score of Each Athlete:")
print(average_scores)

# Step 4: Data Retrieval - Querying for top athletes
top_athletes = data.nlargest(10, 'Average_Score')

# Step 5: Descriptive Analysis - Summary Statistics
print(data.describe())

# Step 6: Data Visualisation
plt.figure(figsize=(12, 6))
sns.barplot(data=top_athletes, x='Athlete', y='Average_Score')
plt.title('Top 10 Athletes by Average Score')
plt.xticks(rotation=45)
plt.show()

# Step 6 Alternative: Bar plot for a random selection of 7 athletes with different colors
random_7_athletes = average_scores.sample(n=7, random_state=42)

# Create a color palette for different bar colors
```

```
colors = sns.color_palette('husl', len(random_7_athletes))

# Plotting the bar plot with different colors for each bar
plt.figure(figsize=(12, 6))
sns.barplot(data=random_7_athletes, x='Athlete', y='Average_Score', palette=colors)
plt.title('Average Score of 7 Randomly Selected Athletes (Different Bar Colors)')
plt.xticks(rotation=90)
plt.show()

# Step 7: Clustering - Group athletes based on performance
X = data[['Average_Score', 'Training_Hours', 'Score_Per_Hour']]
kmeans = KMeans(n_clusters=3)
data['Cluster'] = kmeans.fit_predict(X)

# Visualizing Clusters
plt.figure(figsize=(10, 6))
sns.scatterplot(data=data, x='Average_Score', y='Training_Hours', hue='Cluster', palette='viridis')
plt.title('Athlete Performance Clusters')
plt.show()

# Advanced Clustering using PCA
pca = PCA(n_components=2)
data_pca = pca.fit_transform(X)
data['PCA1'] = data_pca[:, 0]
data['PCA2'] = data_pca[:, 1]

# Visualizing PCA Clusters
plt.figure(figsize=(10, 6))
sns.scatterplot(data=data, x='PCA1', y='PCA2', hue='Cluster', palette='viridis')
plt.title('Athlete Performance Clusters (PCA)')
plt.show()

# Step 8: Regression Analysis - Predict future performance
X = data[['Training_Hours']]
y = data['Score']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Using Gradient Boosting Regressor
model = GradientBoostingRegressor()
model.fit(X_train, y_train)



# Pivot table for heatmap
heatmap_data = top_athletes.pivot_table(index="Athlete", columns="'Training_Hours",
values="Average_Score")

# Set the figure size for the heatmap
plt.figure(figsize=(12, 6))

# Create the heatmap with 'coolwarm' color palette
```

```
sns.heatmap(heatmap_data, annot=True, cmap='coolwarm', linewidths=.5)

# Add title and labels
plt.title('Heatmap of Top 10 Athletes by Average Score and Training Hours')
plt.xlabel('Training Hours')
plt.ylabel('Athlete')

# Display the heatmap
plt.show()

# Evaluating the model
scores = cross_val_score(model, X, y, cv=5)
print("Cross-validated scores:", scores)

# Predicting future score
future_training_hours = pd.DataFrame({'Training_Hours': [10, 15, 20]})
predicted_scores = model.predict(future_training_hours)
print("Predicted scores for future training hours:", predicted_scores)

# Step 9: Time Series Analysis (if applicable)
data['Date'] = pd.to_datetime(data['Date']) # Convert the Date column to datetime format
data.set_index('Date', inplace=True) # Set Date column as index
numeric_columns = data.select_dtypes(include=[np.number])

# Perform the resampling and mean on only numeric columns
numeric_columns.resample('MS').mean()['Score'].plot(figsize=(12, 6))
plt.title('Average Monthly Score Over Time')
plt.ylabel('Average Score')
plt.show()
```

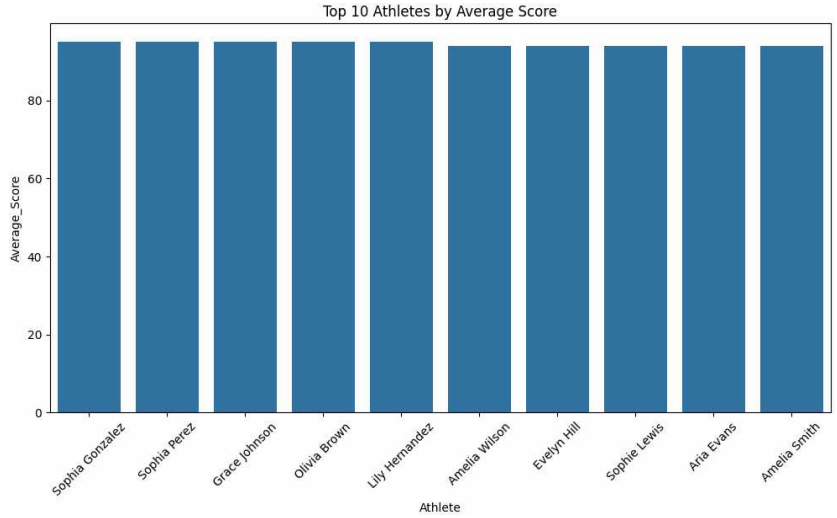
Average Score of Each Athlete:

	Athlete	Average_Score
63	LilyHernandez	95.0
53	OliviaBrown	95.0
7	SophiaGonzalez	95.0
43	GraceJohnson	95.0
27	SophiaPerez	95.0
13	EmilyBrown	86.8
1	AvaMartinez	85.0
24	MichaelCarter	85.8
33	AveryDavis	85.0
9	CharlotteSmith	84.0

[61rows x 2 columns]

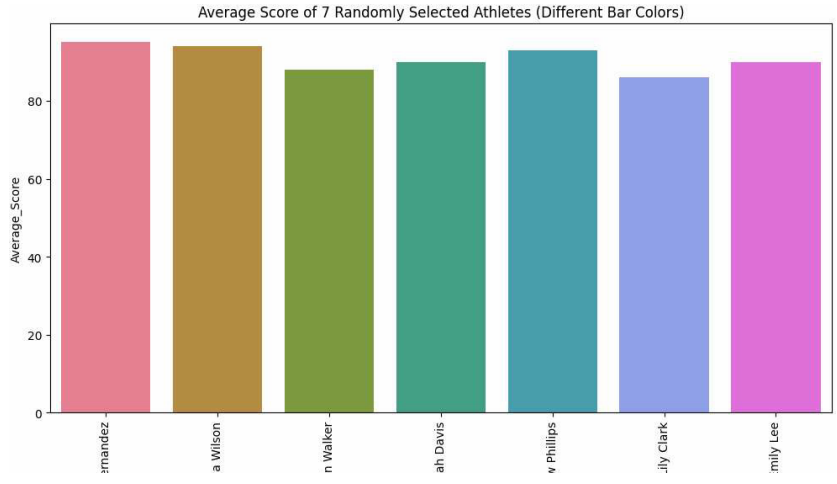
	Score	Training_Hours	Age	Average_Score	Score_Per_Hour	\
count	64.000000	64.000000	64.000000	64.000000	64.000000	
mean	90.218750	10.500000	24.406250	90.218750	8.800898	
std	2.858953	1.763834	1.610666	2.818413	1.286915	
min	84.000000	7.000000	22.000000	84.000000	6.785714	
25%	88.000000	9.000880	23.000000	88.000000	7.812500	
50%	90.000000	11.000000	24.000000	90.000000	8.363636	
75%	92.000000	12.000000	26.000000	92.000000	9.777778	
max	95.000000	14.000000	28.000000	95.000000	12.142857	

	Ranking
count	64.00000
mean	32.50000
std	18.52947
min	3.00000
25%	19.00000
50%	32.50000
75%	50.50000
max	64.00000



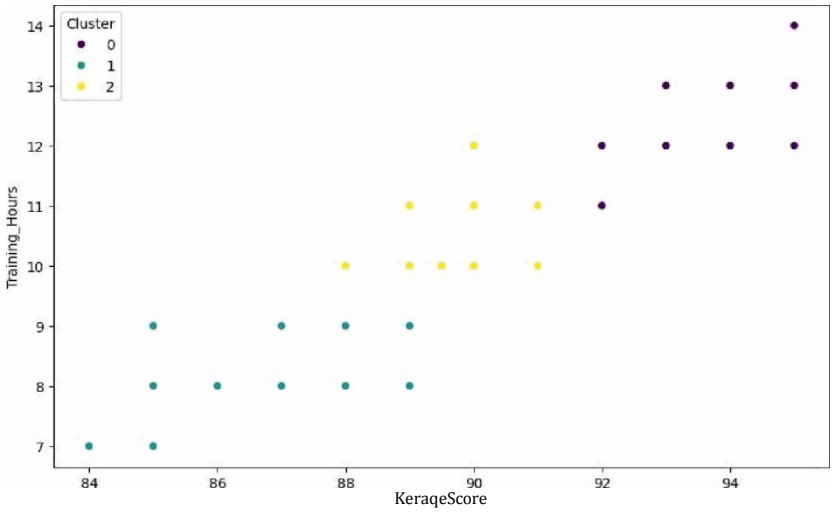
‹ipython-input-12-0883ccl43775›:55:FutureWarniug:

Passing’palette’withoutassigning’hue’isdeprecatedandwillberemovedinv0.

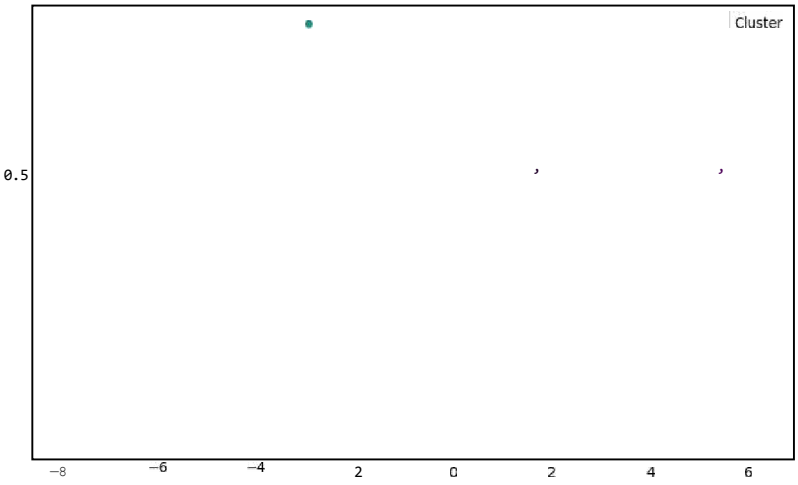


Benjami

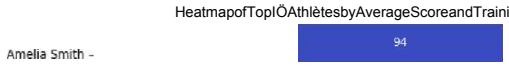
AthĬetePerformanceClusters



AthlètePerformanceÉlusLers(PCAj



HeatmapofTopIÖAthlètesbyAverageScoreandTrainingHours



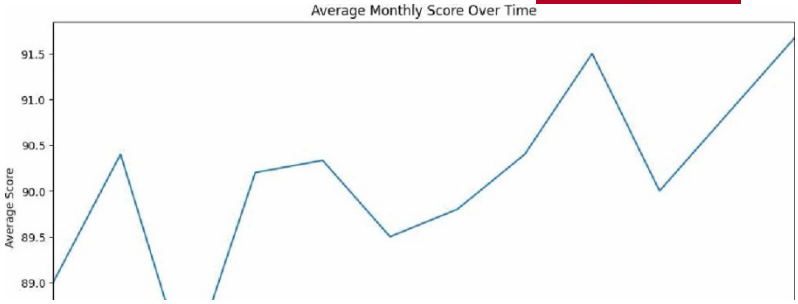
AriaEvans

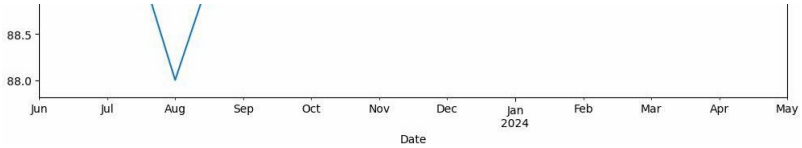
EveĭyriHill

Grazelohnso

Cross-validatedscores:[0.764819880.879722430.84893110.745363310.84840715]

Predictedscoresforfuturetrainginghours:[9B.3998336594.9998169594.99981695]




88.5
88.0
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Jan
2024
Feb
Mar
Apr
May
Date