Analyzing KAGGLE Petrol Dataset.

Importing the needed librarys

In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, r2_score
from sklearn.preprocessing import LabelEncoder

from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

import pickle
```

Loading The Dataset

In [2]:
```python
# Load the dataset into a pandas dataframe
dataset = pd.read_csv('Exam Results April.csv')
```

Analyzing The Dataset

In [3]:
```python
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 148 entries, 0 to 147
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Exam_Date   148 non-null    object
 1   Section     148 non-null    object
 2   Student_ID  148 non-null    int64
 3   Class       144 non-null    object
 4   Score       134 non-null    float64
dtypes: float64(1), int64(1), object(3)
memory usage: 5.9+ KB
```

In [4]:
```python
dataset.shape
```

Out[4]: (148, 5)

In [5]:
```python
dataset.columns
```

Out[5]: Index(['Exam_Date', 'Section', 'Student_ID', 'Class', 'Score'], dtype='object')

In [6]:
```python
dataset.describe().T
```

Out[6]:

	count	mean	std	min	25%	50%	75%	max
Student_ID	148.0	1074.500000	42.868014	1001.0	1037.75	1074.5	1111.25	1148.0
Score	134.0	85.201493	8.260587	70.0	78.00	84.5	92.00	100.0

Dropping Unwanted Data

In [7]:
```python
# Drop the "Exam_Date" column
dataset = dataset.drop("Exam_Date", axis=1)
```

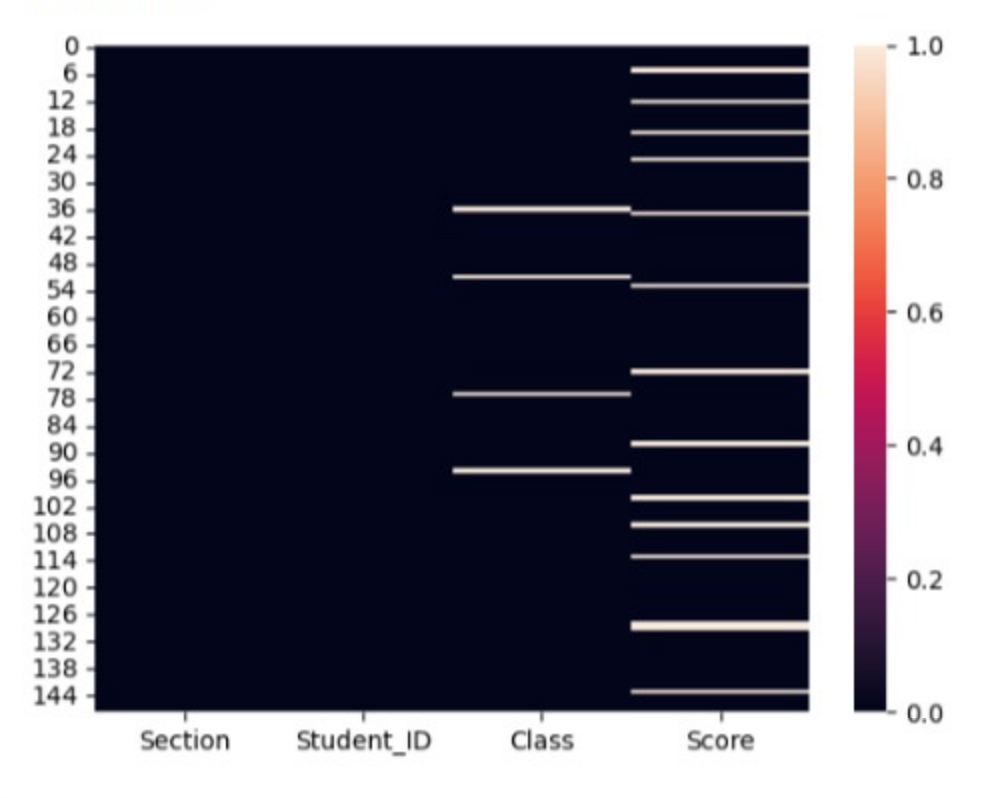
Cleaning The Data

In [8]:
```python
dataset.isnull().sum()
```

Out[8]:
```
Section        0
Student_ID     0
Class          4
Score         14
dtype: int64
```

In [9]:
```python
sns.heatmap(dataset.isnull())
```

Out[9]: <AxesSubplot:>



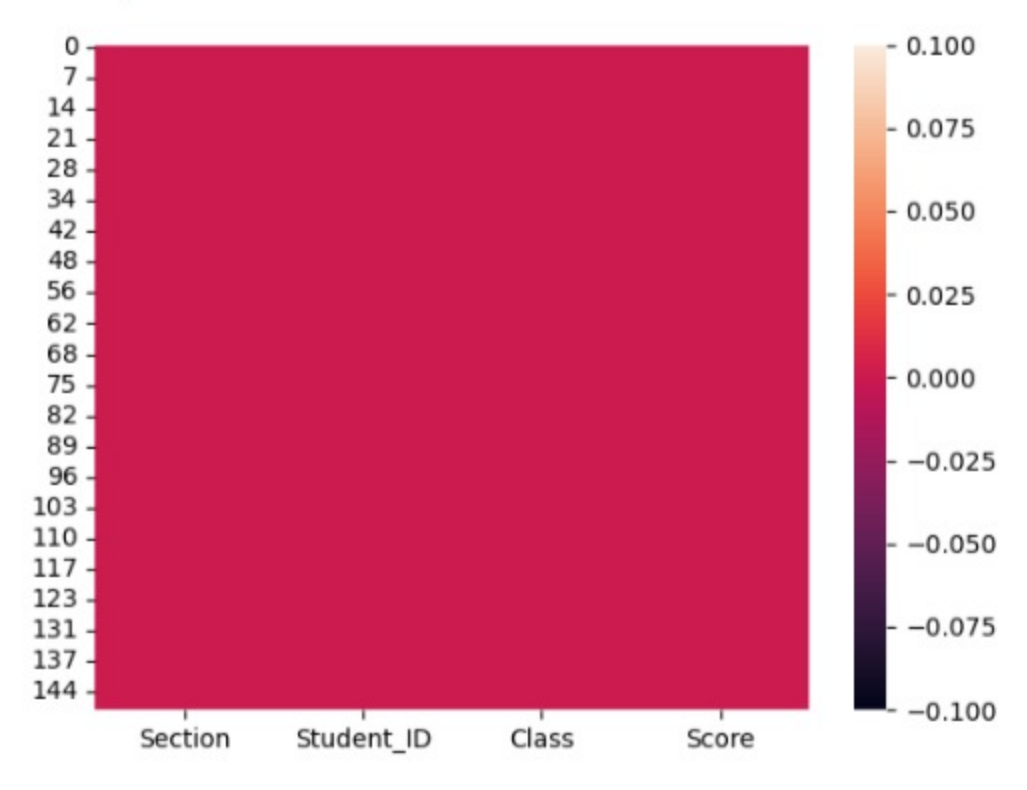
In [10]:
```python
dataset = dataset.dropna()
```

In [11]:
```python
dataset.isnull().sum()
```

Out[11]:
```
Section       0
Student_ID    0
Class         0
Score         0
dtype: int64
```

In [12]:
```python
sns.heatmap(dataset.isnull())
```

Out[12]: <AxesSubplot:>



Finding The Average

In [13]:
```python
average_scores = dataset.groupby('Class')['Score'].mean()
average_scores
```

Out[13]:
```
Class
Arts        84.884615
Computer    86.920000
History     83.884615
Math        87.000000
Science     82.925926
Name: Score, dtype: float64
```

Finding Top 10 Student

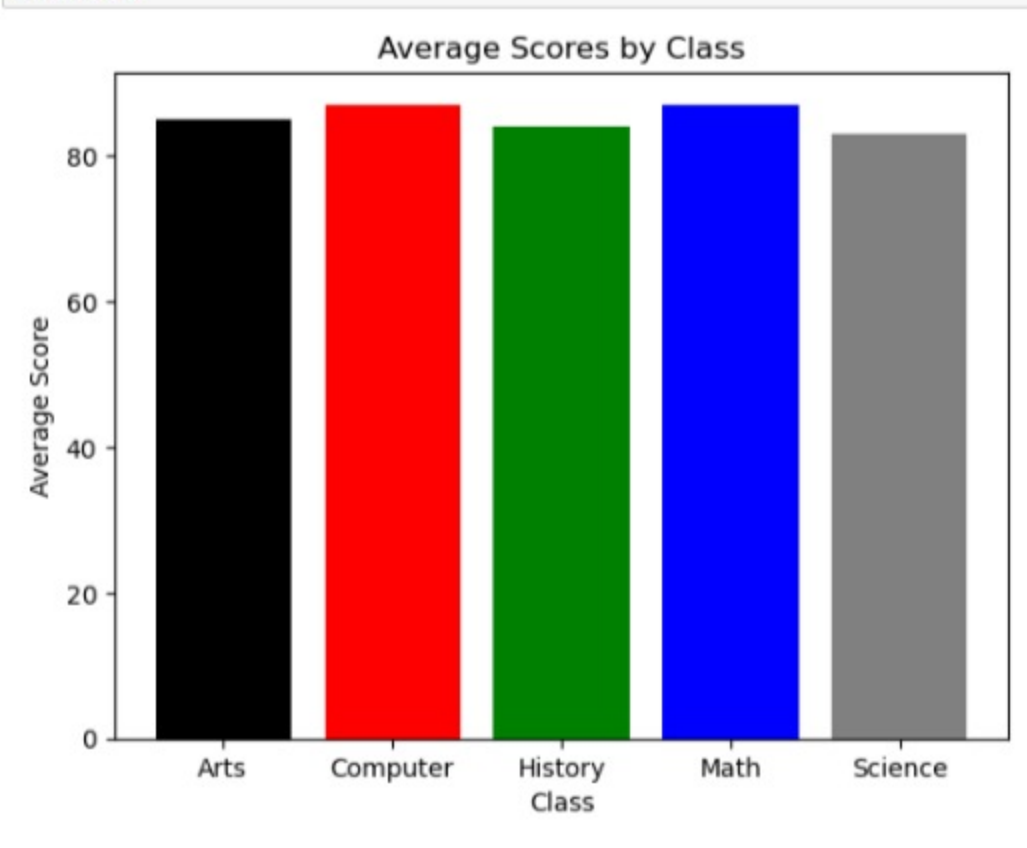
In [14]:
```python
dataset.sort_values(by='Score', ascending=False, inplace=True)
top_students = dataset.head(10)
top_students
```

Out[14]:

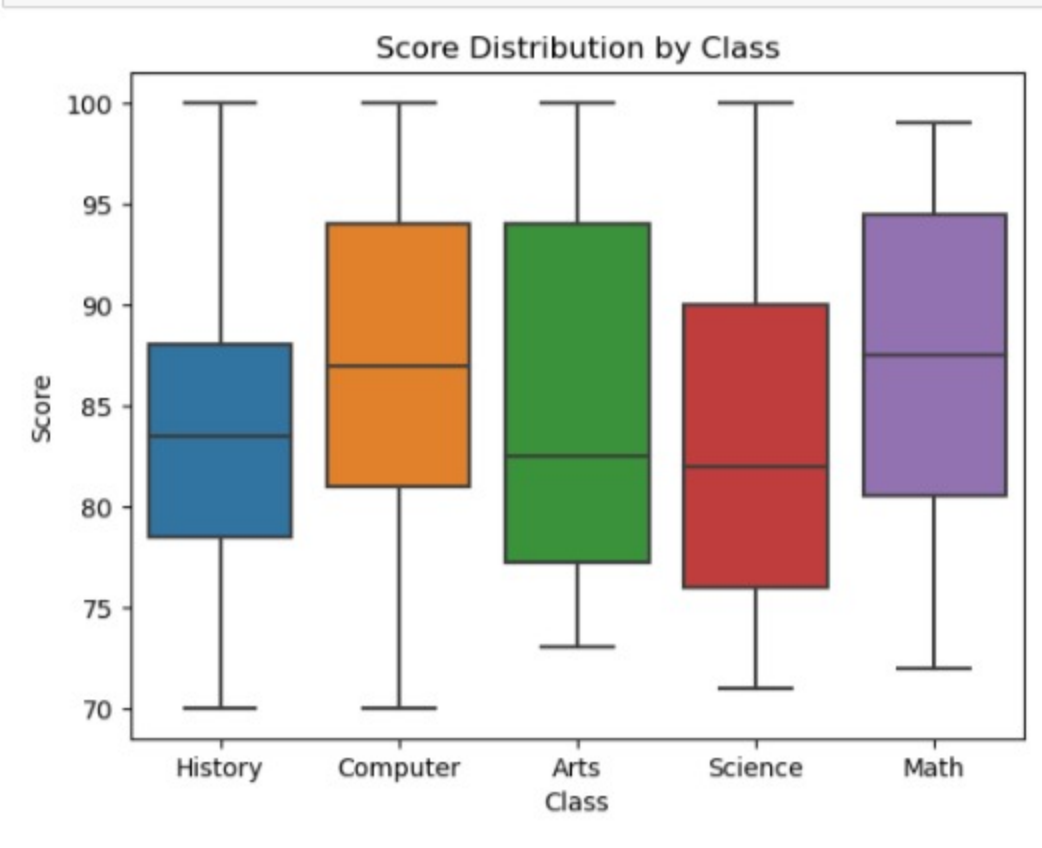
	Section	Student_ID	Class	Score
20	B	1021	History	100.0
122	A	1123	Computer	100.0
118	B	1119	Arts	100.0
63	A	1064	Science	100.0
40	A	1041	Math	99.0
34	A	1035	Math	99.0
68	A	1069	Science	99.0
15	B	1016	History	99.0
145	B	1146	Computer	98.0
44	A	1045	Math	98.0

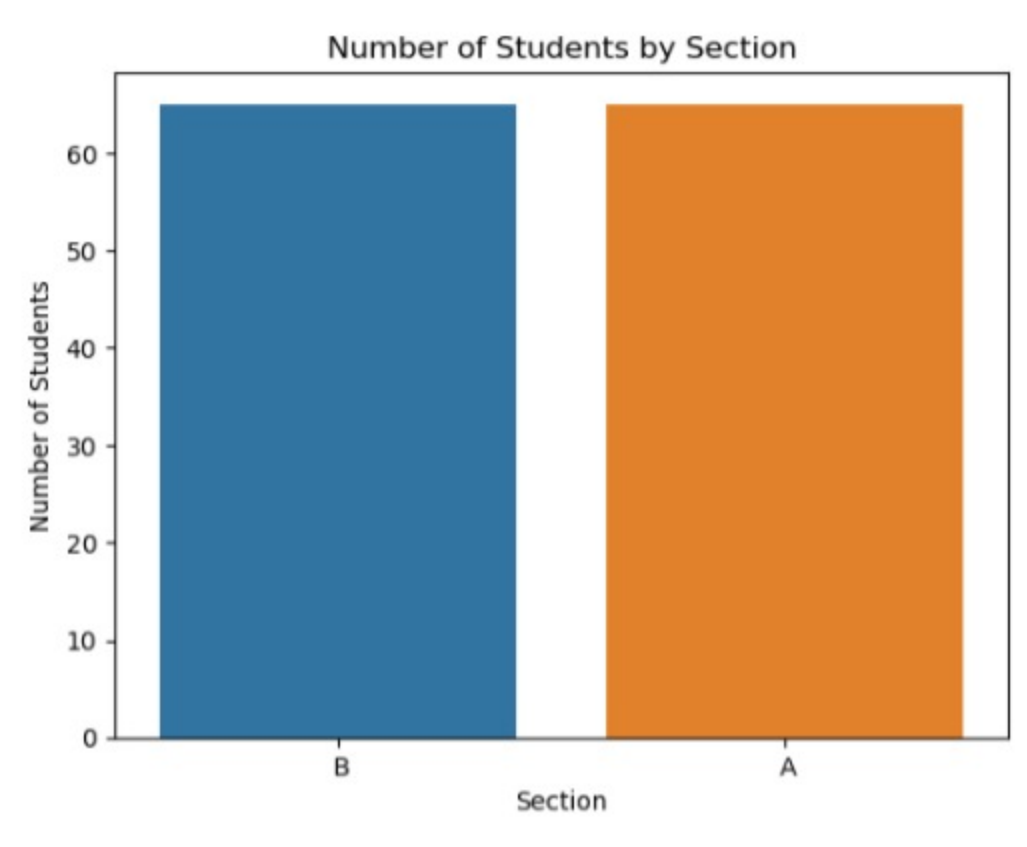
Visualizing the Data

In [15]:
```python
plt.bar(average_scores.index, average_scores.values,color=['black', 'red', 'green', 'blue', 'gray'])
plt.xlabel('Class')
plt.ylabel('Average Score')
plt.title('Average Scores by Class')
plt.show()
```



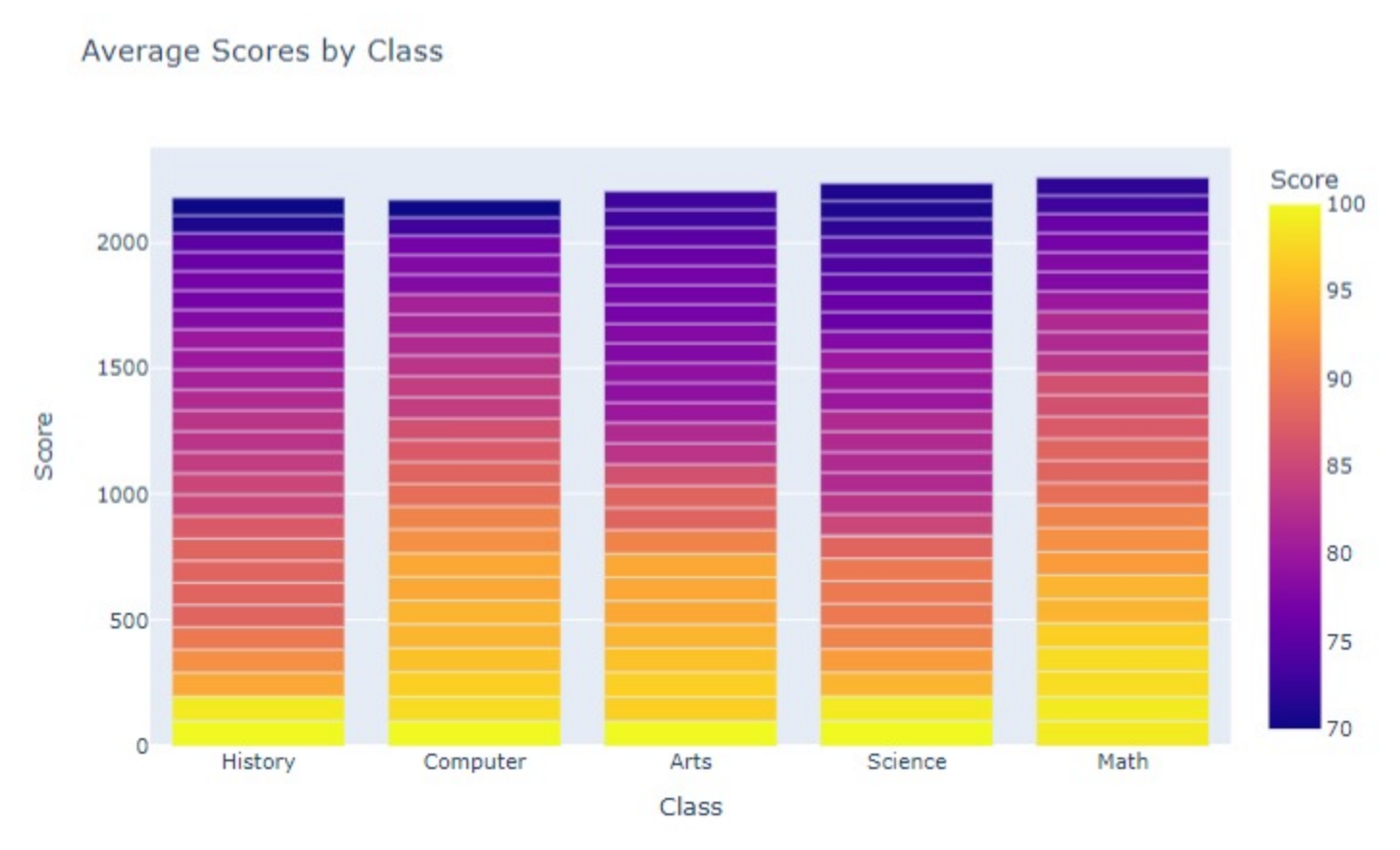
In [16]:
```python
# Create a box plot showing the distribution of scores for each class
sns.boxplot(x='Class', y='Score', data=dataset)
plt.xlabel('Class')
plt.ylabel('Score')
plt.title('Score Distribution by Class')
plt.show()

# Create a bar plot showing the number of students in each section
sns.countplot(x='Section', data=dataset)
plt.xlabel('Section')
plt.ylabel('Number of Students')
plt.title('Number of Students by Section')
plt.show()
```



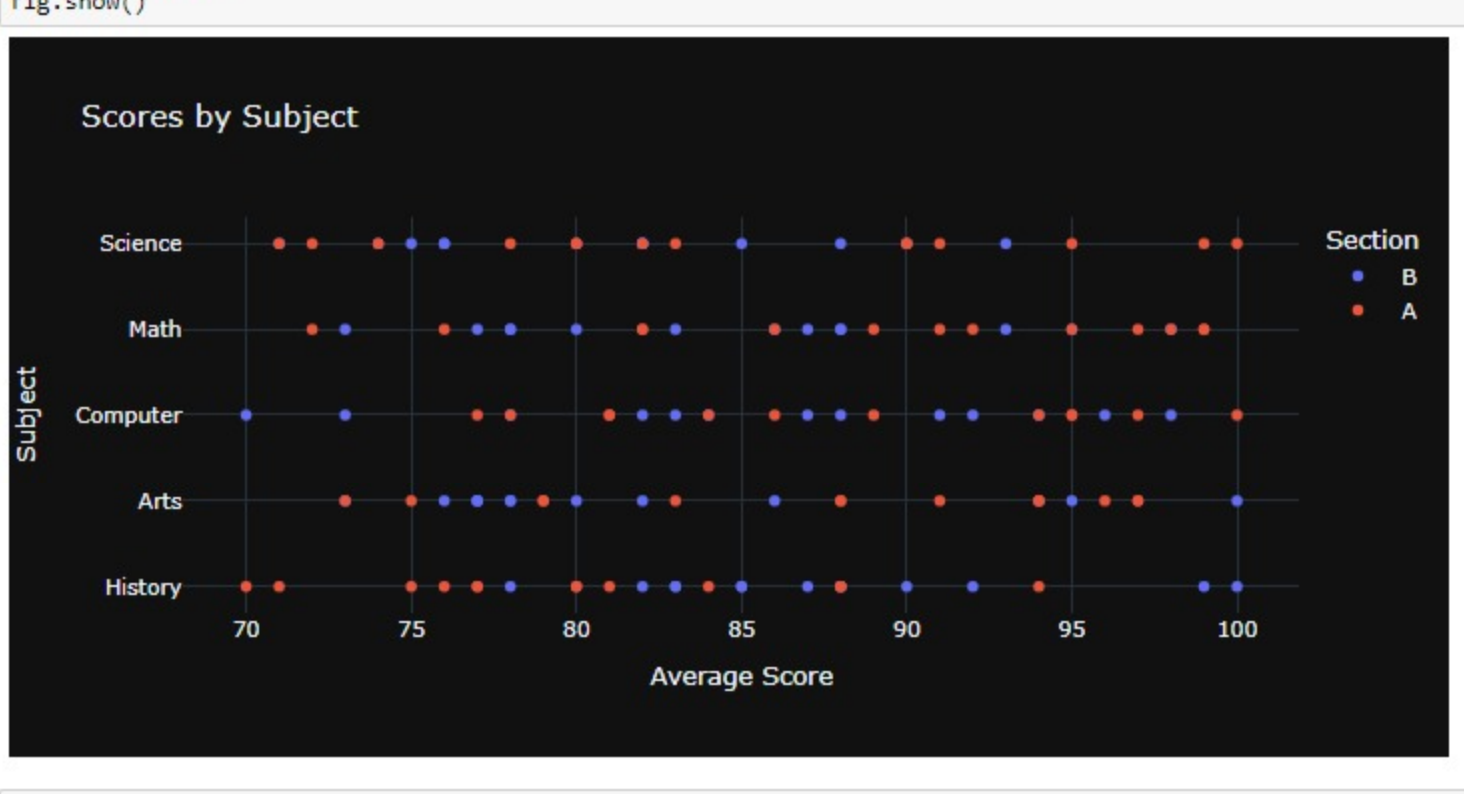


Using plotly.express

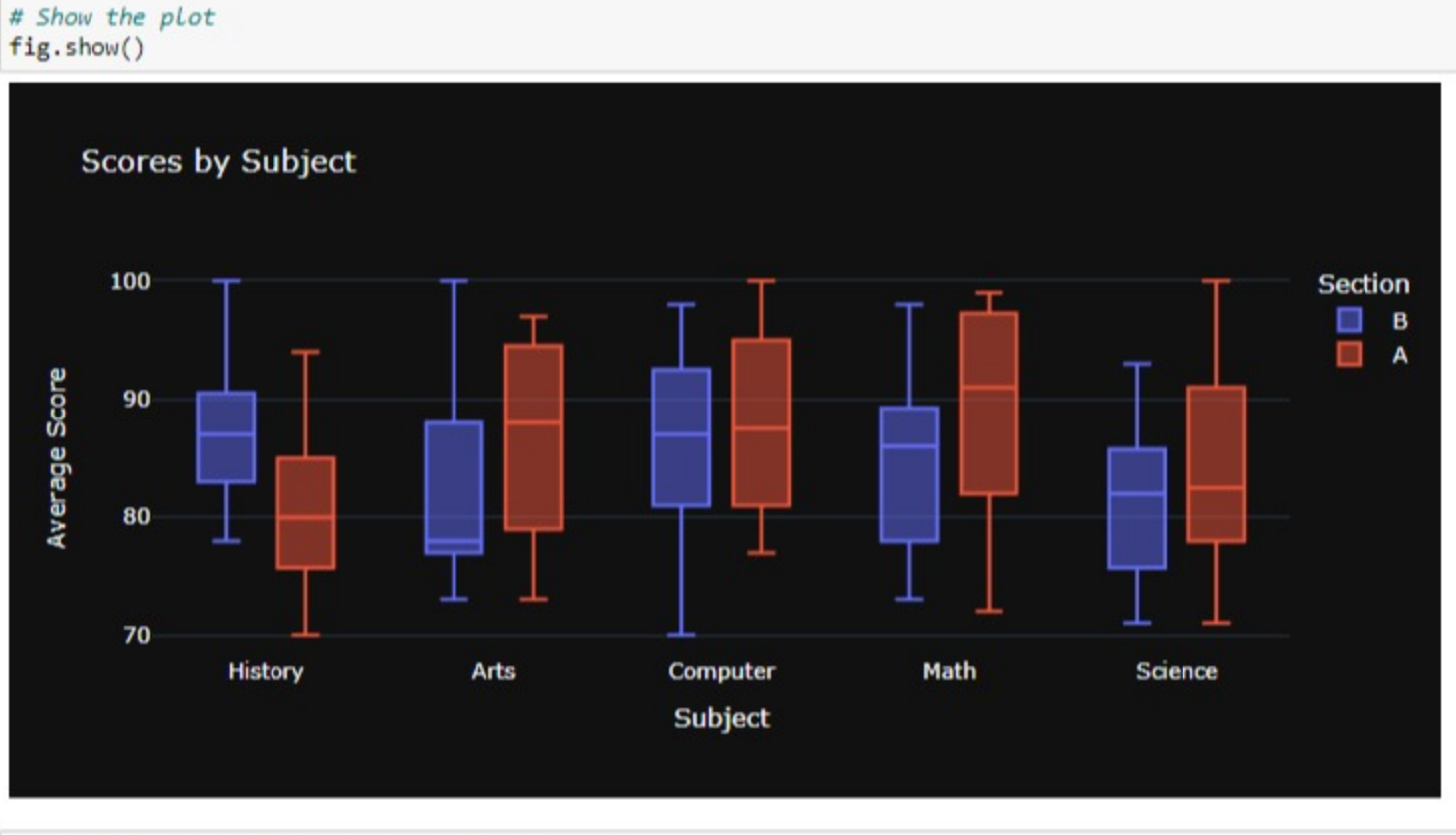
In [17]:
```python
fig = px.bar(dataset, x="Class", y="Score", title="Average Scores by Class", color="Score", width=800)
fig.show()
```



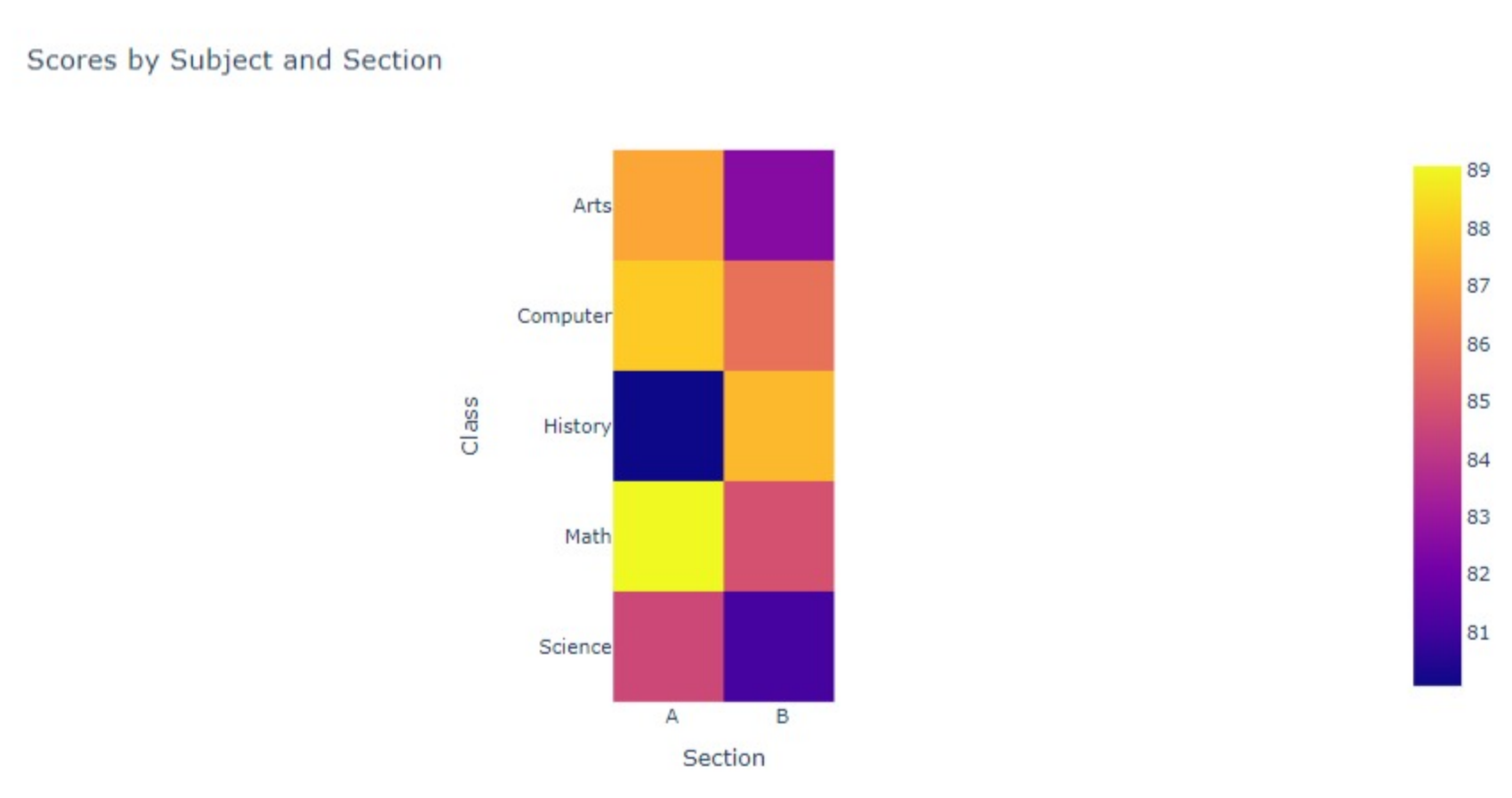
In [18]:
```python
fig = px.scatter(dataset, x="Score", y="Class", title="Scores by Subject", color="Section",
                 hover_name="Student_ID", labels={"Score": "Average Score", "Class": "Subject"},
                 template="plotly_dark", width=800, height=400)

# Show the plot
fig.show()
```



In [19]:
```python
# Plot the data using a box plot
fig = px.box(dataset, x="Class", y="Score", title="Scores by Subject", color="Section",
             hover_name="Student_ID", labels={"Class": "Subject", "Score": "Average Score"},
             template="plotly_dark", width=800, height=400)

# Show the plot
fig.show()
```



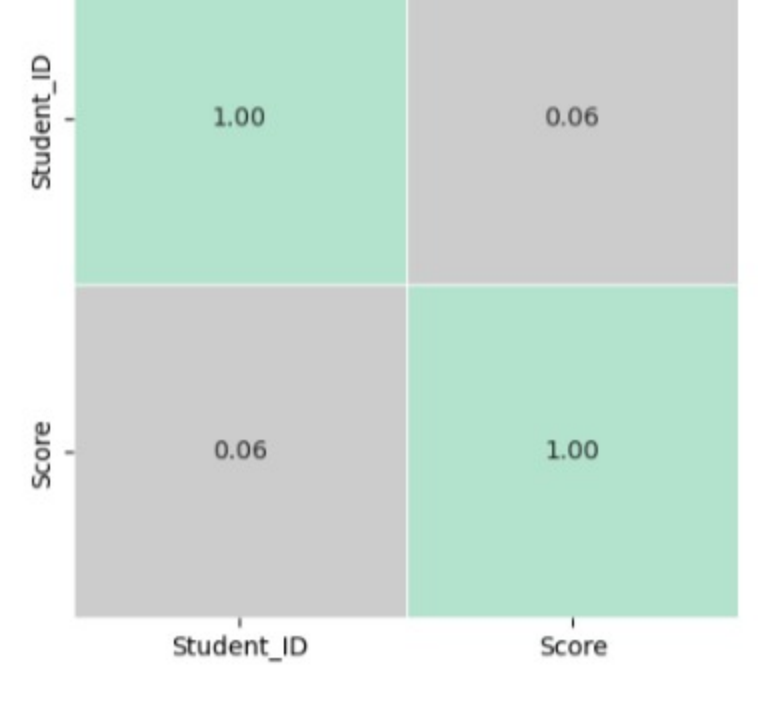
In [20]:
```python
# Create a pivot table to summarize the data
pivot = dataset.pivot_table(index="Class", columns="Section", values="Score")

# Plot the data using a heatmap
fig = px.imshow(pivot, title="Scores by Subject and Section")

# Show the plot
fig.show()
```



Finding the Correlation

In [21]:
```python
# Calculate the correlation matrix
correlation = dataset.corr()

# Visualize the correlation matrix using a heatmap
sns.heatmap(correlation, annot=True, cmap="Pastel2_r", linewidths=0.5,
            fmt=".2f", cbar=False, square=True, xticklabels=correlation.columns, yticklabels=correlation.columns)
```

Out[21]: <AxesSubplot:>



Machine Learning Over The Data

In [22]:
```python
# Encode the "Class" column to numerical values
le = LabelEncoder()
dataset["Class"] = le.fit_transform(dataset["Class"])
dataset["Section"] = le.fit_transform(dataset["Section"])
```

In [23]:
```python
# Select the features and target
X = dataset[["Class", "Section"]]
y = dataset["Score"]
```

In [24]:
```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [25]:
```python
# Create a random forest regressor
model = RandomForestRegressor(n_estimators=100, random_state=42)
```

In [26]:
```python
model.fit(X_train, y_train)
```

Out[26]: RandomForestRegressor(random_state=42)

Save the Model

In [28]:
```python
# Save the model to a file
with open("model.pkl", "wb") as f:
    pickle.dump(model, f)
```

Load the Model

In [29]:
```python
# Load the saved model
with open("model.pkl", "rb") as f:
    saved_model = pickle.load(f)
```

Test the Model

In [30]:
```python
# Make predictions on the test data using the saved model
y_pred = saved_model.predict(X_test)
y_pred
```

Out[30]:
```
array([86.1228307 , 80.80349802, 86.67291162, 87.49012413, 80.80349802,
       83.33959284, 83.33959284, 81.17452961, 84.6665701 , 87.22775266,
       82.33411513, 86.67291162, 81.17452961, 87.49012413, 86.67291162,
       86.1228307 , 87.22775266, 88.06059459, 87.22775266, 81.17452961,
       82.33411513, 83.33959284, 82.33411513, 86.67291162, 81.17452961,
       88.06059459])
```