Bivariate Distribution: Study Hours vs Exam Performance

1. Importing necessary libraries

First, we need to import the libraries required for our analysis. We will use `numpy` for numerical operations and data generation, `pandas` for data manipulation, and `matplotlib` and `seaborn` for data visualisation.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

2. Generating sample data

We generate a synthetic bivariate dataset that simulates the relationship between hours studied and exam scores. We use a normal distribution to create a random sample of hours studied, with a mean of 5 hours and a standard deviation of 2, plus some added noise to reflect real-world variability.

```
np.random.seed(42)
hours_studied = np.random.normal(5, 2, 100)
exam_scores = hours_studied * 10 + np.random.normal(0, 10, 100)  # Linear relationship with noise

data = pd.DataFrame({
    'Hours Studied': hours_studied,
    'Exam Scores': exam_scores
})

# Displaying the first few rows of the dataset
data.head()
```

```
   Hours Studied  Exam Scores
0       5.993428    45.780576
1       4.723471    43.028261
2       6.295377    59.526626
3       8.046060    72.437824
4       4.531693    43.704075
```

Next steps: Generate code with data View recommended plots New interactive sheet

3. Calculating statistical measures

In this section, we compute key statistical measures for both variables, such as the mean, variance, and standard deviation. These calculations help us understand the distribution of the data and interpret the scatterplot in the subsequent section.

```
mean_hours = data['Hours Studied'].mean()
mean_scores = data['Exam Scores'].mean()
variance_hours = data['Hours Studied'].var()
variance_scores = data['Exam Scores'].var()
std_dev_hours = data['Hours Studied'].std()
std_dev_scores = data['Exam Scores'].std()
```

4. Visualising the data

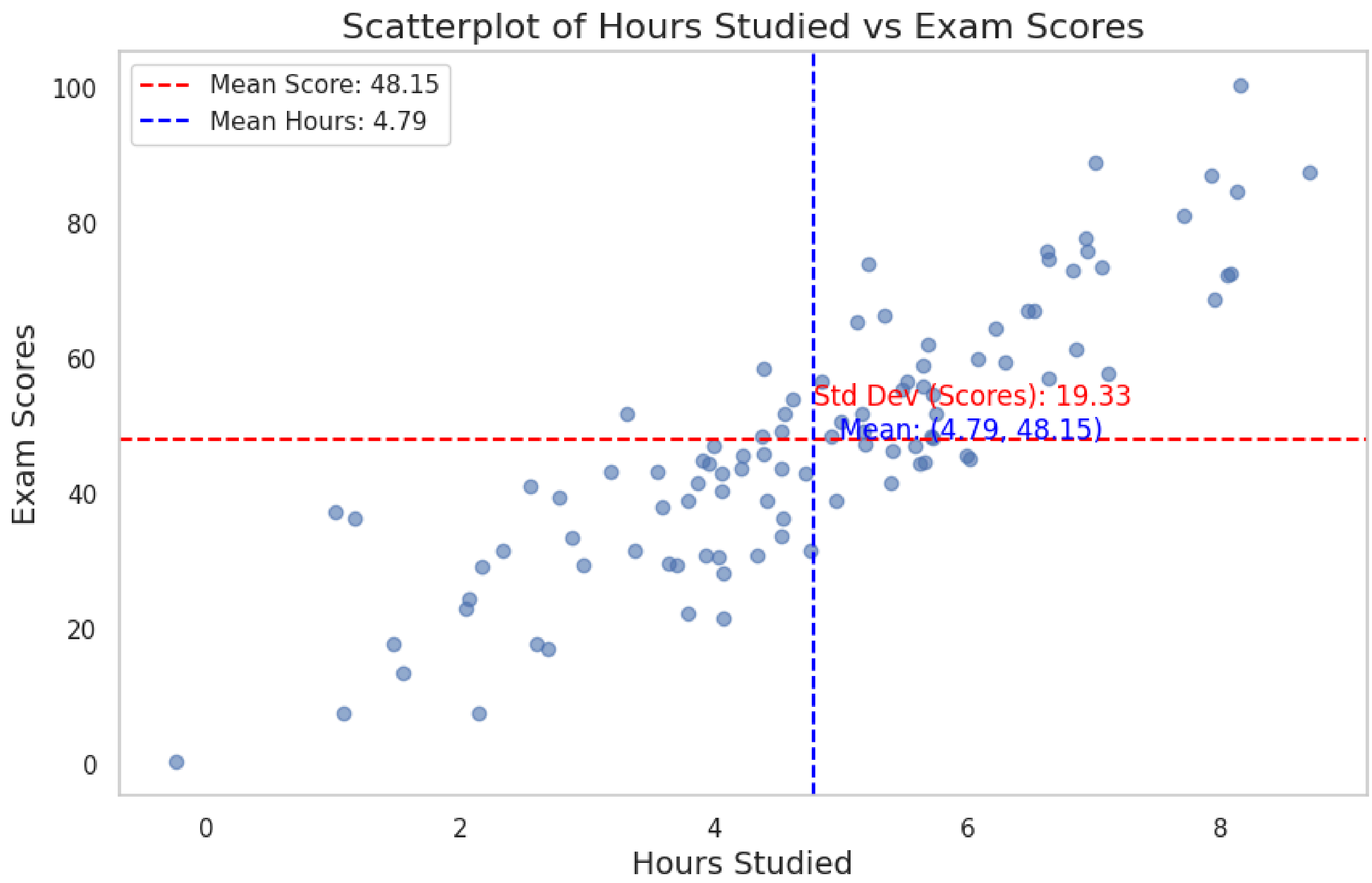
Finally, we focus on visualising the relationship between hours studied and exam scores using a scatterplot. The scatterplot provides a clear graphical representation of the data points, allowing us to observe any trends or patterns. Labels are also added for the various statistical measures.

```
sns.set(style="whitegrid")

plt.figure(figsize=(10, 6))
plt.scatter(data['Hours Studied'], data['Exam Scores'], alpha=0.6)
plt.title('Scatterplot of Hours Studied vs Exam Scores', fontsize=16)
plt.xlabel('Hours Studied', fontsize=14)
plt.ylabel('Exam Scores', fontsize=14)

plt.axhline(mean_scores, color='red', linestyle='--', label=f'Mean Score: {mean_scores:.2f}')
plt.axvline(mean_hours, color='blue', linestyle='--', label=f'Mean Hours: {mean_hours:.2f}')
plt.text(mean_hours + 0.2, mean_scores, f'Mean: ({mean_hours:.2f}, {mean_scores:.2f})', color='blue')
plt.text(mean_hours, mean_scores + 5, f'Std Dev (Scores): {std_dev_scores:.2f}', color='red')

plt.legend()
plt.grid()
plt.show()
```


Scatterplot of Hours Studied vs Exam Scores
Mean Score: 48.15
Mean Hours: 4.79
Std Dev (Scores): 19.33
Mean: (4.79, 48.15)
Exam Scores
100
80
60
40
20
0
0
2
4
6
8
Hours Studied