Data Science & ML

Case Study 3

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
import pandas as pd

# Load the dataset FyntraCustomerData.csv.

from google.colab import drive
drive.mount('/content/drive')


# Load the dataset FyntraCustomerData.csv.
data = pd.read_csv('/content/drive/My Drive/Colab_Data/FyntraCustomerData.csv')
#data = pd.read_csv('FyntraCustomerData.csv')

print("Info of the dataset:")
print(data.info())
print("-"*50)
print("Check for null values in the dataset:")
print(data.isnull().sum())
print("-"*50)
print("Print the first rows from the dataset:")
print(data.head())
```

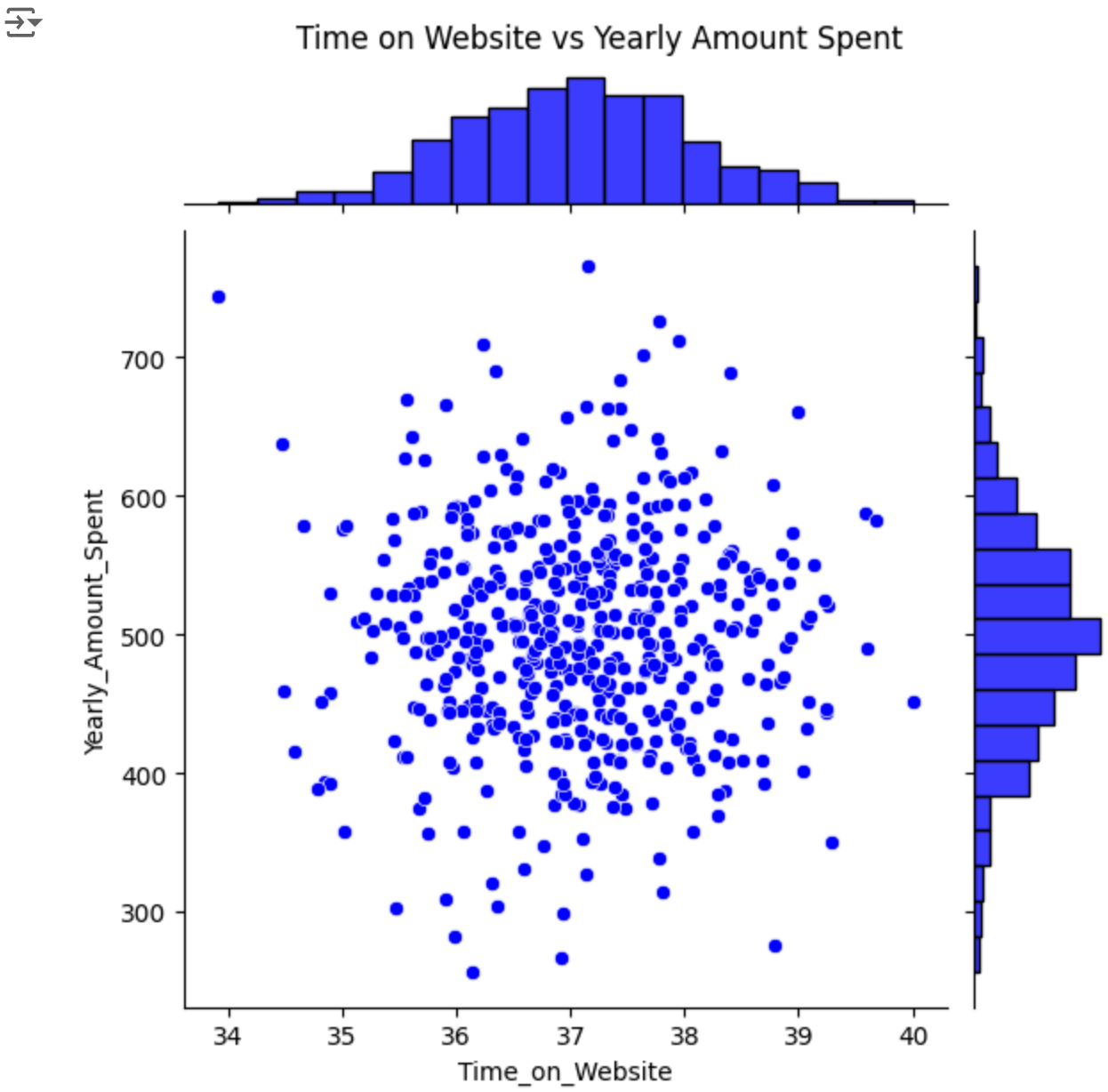
```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
Info of the dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 8 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Email                500 non-null    object 
 1   Address              500 non-null    object 
 2   Avatar               500 non-null    object 
 3   Avg_Session_Length   500 non-null    float64
 4   Time_on_App          500 non-null    float64
 5   Time_on_Website      500 non-null    float64
 6   Length_of_Membership 500 non-null    float64
 7   Yearly_Amount_Spent  500 non-null    float64
dtypes: float64(5), object(3)
memory usage: 31.4+ KB
None
--------------------------------------------------
```

```
Check for null values in the dataset:
Email                   0
Address                 0
Avatar                  0
Avg_Session_Length      0
Time_on_App             0
Time_on_Website         0
Length_of_Membership    0
Yearly_Amount_Spent     0
dtype: int64
--------------------------------------------------
Print the first rows from the dataset:
                          Email  \
0      mstephenson@fernandez.com
1              hduke@hotmail.com
2               pallen@yahoo.com
3         riverarebecca@gmail.com
4  mstephens@davidson-herman.com

                                        Address            Avatar  \
0     835 Frank Tunnel\r\nWrightmouth, MI 82180-9605            Violet
1   4547 Archer Common\r\nDiazchester, CA 06566-8576         DarkGreen
2  24645 Valerie Unions Suite 582\r\nCobbborough,...            Bisque
3  1414 David Throughway\r\nPort Jason, OH 22070-...       SaddleBrown
4  14023 Rodriguez Passage\r\nPort Jacobville, PR...  MediumAquaMarine

   Avg_Session_Length  Time_on_App  Time_on_Website  Length_of_Membership  \
0           34.497268    12.655651        39.577668              4.082621
1           31.926272    11.109461        37.268959              2.664034
2           33.000915    11.330278        37.110597              4.104543
3           34.305557    13.717514        36.721283              3.120179
4           33.330673    12.795189        37.536653              4.446308

   Yearly_Amount_Spent
0           587.951054
1           392.204933
2           487.547505
3           581.852344
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

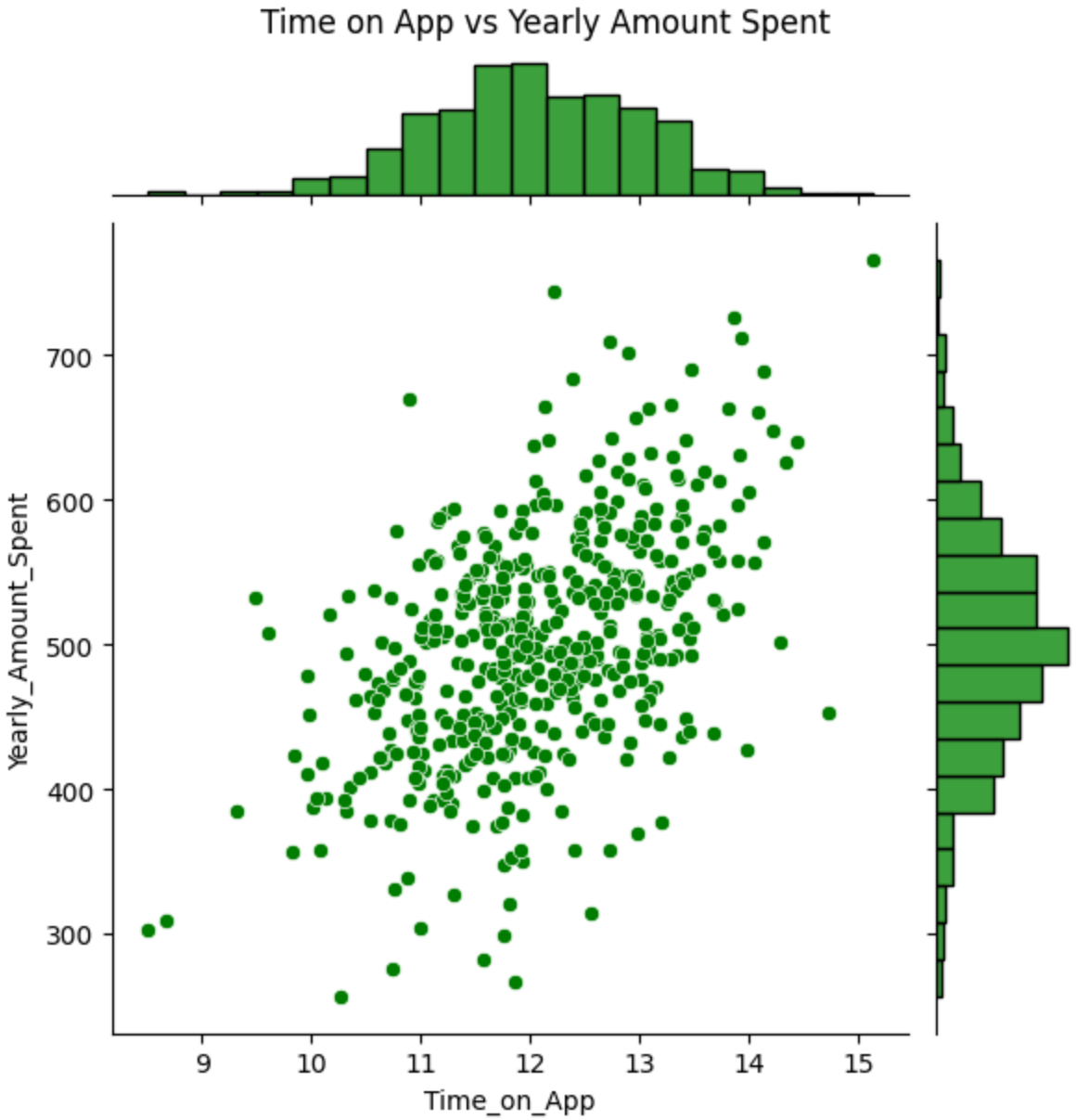
```
# 1. Compare Time on Website and Yearly Amount Spent
import seaborn as sns
import matplotlib.pyplot as plt

# Jointplot: Time on Website vs Yearly Amount Spent
sns.jointplot(
    x="Time_on_Website",
    y="Yearly_Amount_Spent",
    data=data,
    kind="scatter",
    color="blue"
)
```

```
plt.suptitle('Time on Website vs Yearly Amount Spent', y=1.02)
plt.show()
```




```
# 2. Compare Time on App and Yearly Amount Spent

# Jointplot: Time on App vs Yearly Amount Spent
sns.jointplot(
    x="Time_on_App",
    y="Yearly_Amount_Spent",
    data=data,
    kind="scatter",
    color="green"
)
plt.suptitle('Time on App vs Yearly Amount Spent', y=1.02)
```

```
plt.show()
```

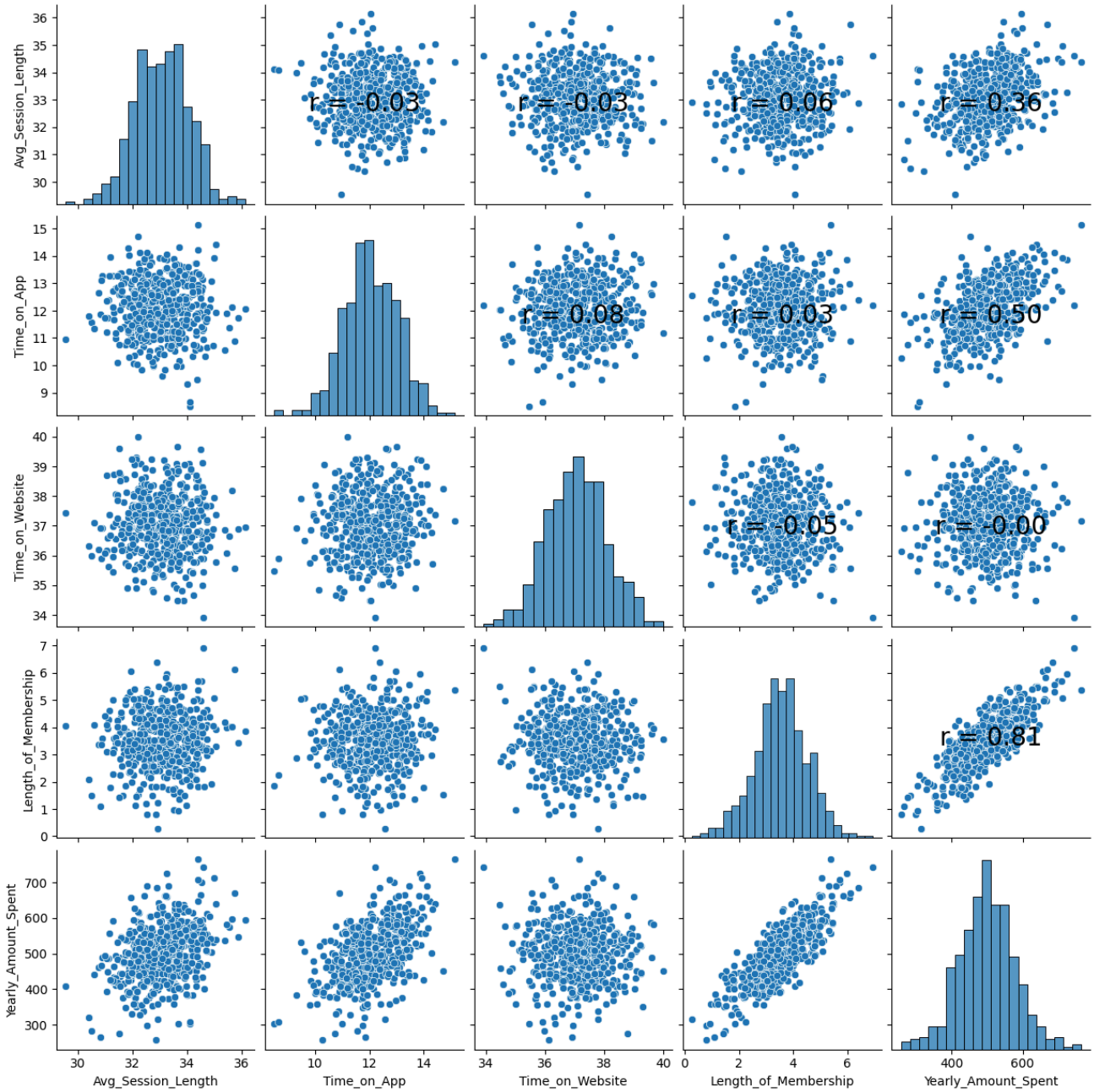



```
# Comparison of Correlations

website_corr = data['Time_on_Website'].corr(data['Yearly_Amount_Spent'])
app_corr = data['Time_on_App'].corr(data['Yearly_Amount_Spent'])

print(f"Correlation between Time on Website and Yearly Amount Spent: {website_corr:.2f}")
print(f"Correlation between Time on App and Yearly Amount Spent: {app_corr:.2f}")
```

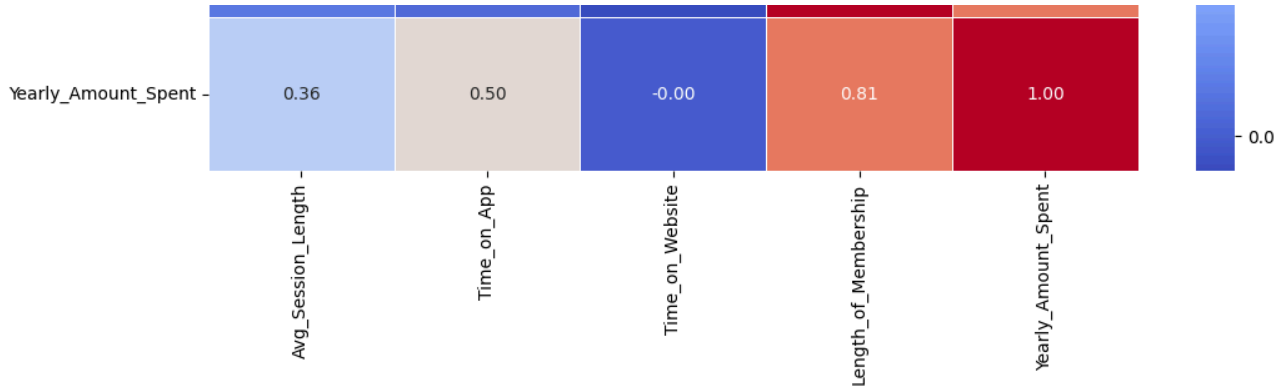
```
Correlation between Time on Website and Yearly Amount Spent: -0.00
Correlation between Time on App and Yearly Amount Spent: 0.50
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
# 3. Pairplot to Explore Relationships
# Objective: Visualize relationships across the dataset to identify the feature most correla

import numpy as np

# Define a function to annotate correlation values
def annotate_corr(x, y, **kwargs):
    """Annotate the correlation coefficient in the upper triangle of pairplot."""
    r = np.corrcoef(x, y)[0, 1]  # Compute correlation coefficient
    ax = plt.gca()  # Get the current axis
    ax.annotate(f"r = {r:.2f}", xy=(0.5, 0.5), xycoords=ax.transAxes,
                ha='center', va='center', fontsize=20, color='black')

# Create the pairplot and apply the custom correlation annotation
pairplot = sns.pairplot(data, kind="scatter")

# Map the correlation annotation function to the upper triangle
pairplot.map_upper(annotate_corr)

# Create a pairplot
# Notice that: sns.pairplot automatically identifies columns with numeric data types (e.g.,
# sns.pairplot(data)
plt.show()

# Compute the correlation matrix to get a better idea
correlation_matrix = data.select_dtypes(include=['float64']).corr()

# Visualize the correlation matrix as a heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)
plt.title("Correlation Heatmap of Independent Variables")
plt.show()
```


Avg_Session_Length
Time_on_App
Time_on_Website
Length_of_Membership
Yearly_Amount_Spent
r = -0.03
r = -0.03
r = 0.06
r = 0.36
r = 0.08
r = 0.03
r = 0.50
r = -0.05
r = -0.00
r = 0.81


Correlation Heatmap of Independent Variables
Avg_Session_Length
Time_on_App
Time_on_Website
Length_of_Membership
1.00 -0.03 -0.03 0.06 0.36
-0.03 1.00 0.08 0.03 0.50
-0.03 0.08 1.00 -0.05 -0.00
0.06 0.03 -0.05 1.00 0.81


Yearly_Amount_Spent
0.36
0.50
-0.00
0.81
1.00
0.0
Avg_Session_Length
Time_on_App
Time_on_Website
Length_of_Membership
Yearly_Amount_Spent

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
# 4. Linear Model Plot
# Objective: Examine the relationship between Length_of_Membership and Yearly_Amount_Spent u

# Linear regression plot
sns.lmplot(x='Length_of_Membership', y='Yearly_Amount_Spent', data=data)
plt.title('Length of Membership vs Yearly Amount Spent')
plt.show()
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
# 5.  Train-Test Split and Random State
# Objective: Split the data into training and testing sets and explain the purpose of random

from sklearn.model_selection import train_test_split

# Define predictors (X) and target variable (y)
```

```
X = data[['Avg_Session_Length', 'Time_on_App', 'Time_on_Website', 'Length_of_Membership']]
y = data['Yearly_Amount_Spent']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=85)

print("Training Data Shape:", X_train.shape)
print("Test Data Shape:", X_test.shape)
```

```
Training Data Shape: (375, 4)
Test Data Shape: (125, 4)
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
#6. Predict and Scatter Plot
# Objective: Train a linear regression model, predict on the test set, and compare actual vs


from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Train the model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Scatter plot: Actual vs Predicted
plt.scatter(y_test, y_pred, alpha=0.7)
plt.xlabel('Actual Yearly Amount Spent')
plt.ylabel('Predicted Yearly Amount Spent')
plt.title('Actual vs Predicted Values')
plt.show()

print(model.coef_)
```