DETAILED PROJECT REPORT

Mouse movements reflect personality traits and task attentiveness in online experiments

Table of Contents

- Insights from Article
  - Why We Are Using Mouse Movement Features to Predict Individuals' Personality?
  - ii. How Was the Experiment Conducted?
  - iii. Data Analysis Methodologies Used
  - iv. Insights Derived from the Analysis

- General Analysis of Personality Traits with Respect to Age & Gender
- Scientific Analysis of the Data Using Different Machine Learning Models and Power BI Reports
  - Linear Regressor Model Predicting Personality Traits from 11 Mouse-Related Features
  - PLS Analysis for Predicting Big Five Personality Traits from 11 Mouse-Related Features & AUC
  - PLS Analysis for Big Five with Demographics ~ Mouse Movements & AUC
  - PLS Analysis for Big Five ~ Mouse Movements
  - Random Forest Regression Analysis
  - Random Forest Regression Analysis for AUC ~ Mouse Movements
  - Effect Sizes for PLS Analyses
- Innovative Ideas for Enhanced Analysis

Why we are using Mouse movement features to predict individuals' personality?

i. In various previously conducted studies relevant information was found on the relationship b/w mouse movement features and an individual's behaviors.
ii. It has been found that one's online behaviors reflects a person's offline behaviors (as per Orchard & Fullwood, 2010).
iii. Extroverts have higher levels of motor activity (mouse clicking) at a higher frequency in a given task (Brebner 1983 khan et al. ,2008).
iv. Mouse hover time and movement patterns can predict individual's self-efficacy and risk perception (Tzafilkou et al. (2004).
v. Several speed & distance-based mouse movements can show how interesting an article is (Arapakis et al.,2014).
vi. Usefulness & ease of use of a Web based tool can also be predicted by Mouse Movements (Tzafilkou & Nicolaos, 2018)
vii. Average number of mouse clicks was positively correlated with a subtrait of Conscientiousness (r = 0.52) and negatively correlated with a subtrait of Neuroticism (r = -0.40).

How the experiment was conducted?

i. A total of 791 participants , including 483 male , 303 Female & 5 others.
ii. Demographic information such as Age & Gender was also used. Mean age 38.8years, standard deviation 10.8 years.
iii. An Image-rating task was conducted, which showed 12 images of streets.
iv. They need to select 4 images based on a given attribute like 4 most liked images.
v. Attention checks were also conducted randomly as inattentiveness could affect the final result.
vi. They were also asked to drag and put the corrupted image in trash can

Which Data Analysis Methodologies were used ?

i. Pearson correlation or product moment correlation or correlation coefficient & OLS Regression Analysis were performed using 11 mouse related features predicting Abs_Area_Under_Curve (measure of attentiveness = Area_under_curve – 0.5).

ii. AUC close to 1 , if individual's choice of image is highly predictable of the selection of rest of the group.
iii. AUC close to 0, if individual's choice of image is extremely different from choice of rest of the group.
iv. AUC close to 0.5 means inattentiveness or random responding.
v. OLS regression was used to predict Big Five Personality Traits using all 11 mouse related features.
vi. 3 Partial Least Square(PLS) Analysis were done b/w
    a) Big Five Personality Traits & Mouse Movement Features & AUC.
    b) Big Five Personality Traits with Age & Gender and Mouse Movement Features & AUC.
    c) Big Five Personality Traits & Mouse Movement Features.

What all insights were derived from the Analysis ?

i. Outliers were found in the 11 mouse related features avg_click_att , reclick_percent_att ,avg_click_norm, reclick_percent_norm ,avg_euc_dist, avg_euc_speed, avg_completion_time ,total_pause_cnt ,avg_fixation_dur , avg_agg_fixation_dur & avg_fixation_cnt.
ii. total_pause_cnt ,avg_fixation_dur , avg_agg_fixation_dur & avg_fixation_cnt are right skewed , most of the data points are located on the left side.
iii. avg_euc_dist , avg_euc_speed , avg_completion_time are right skewed , most of the data points are located on the left side.
iv. avg_click_att , reclick_percent_att , avg_click_norm & reclick_percent_norm are right skewed , most of the data points are located on the left side.
v. avg_click_att is positively correlated with reclick_percent_att.
vi. If clicks are more , attentiveness is less or inattentiveness is high because avg_click_att is negatively correlated with Abs_area_under_curve.
vii. If reclicks are high again attention is less as reclick_percent_att is negatively correlated with Abs_area_under_curve.
viii. If fixations count increases attentiveness also increases.
ix. If avg_euc_dist increases avg_click_norm , reclick_percent_norm and avg_fixation_cnt also increases.
x. If avg_euc_speed increases, time taken for completing the task decreases, there are less number of pauses taken by the individual, fixation duration and

count also decreases. Avg_euc_speed and attentiveness have weak negative correlation.

xi. If an individual is taking more time to complete the trials means person is taking more pauses, more clicks and reclicks, fixation duration and fixation counts are also high.

xii. If attentiveness decreases, extraversion & neuroticism increases i.e an individual is more stressful and has anxiety and agreeableness, conscientiousness and openness also decreases.

xiii. If avg_click_att increases, a person is more friendly, sociable & positive (Extraversion increases). However, agreeableness, conscientiousness decreases i.e a person is less cooperative, polite, kind and less responsible & organized.

xiv. If reclick is increasing, a person can be more friendly and sociable (Extraversion increases) but openness or frankness decreases.

xv. If avg_click_norm increases, a person is less cooperative, polite , kind and less responsible (agreeableness and conscientiousness decreases).

xvi. Euc_speed is positively correlated with Neuroticism, if speed is high a person has more anxiety , negative emotions, stress but if speed is less agreeableness and conscientiousness is high.

xvii. If avg_agg_fixation_dur increases mean a person is more organized & responsible and more frank.

xviii. If avg_fixation_cnt increases, a person has less anxiety or stress and is more open, social, frank, responsible and organized as Agreeableness, conscientiousness and openness are positively correlated.

xix. More clicking can be related to impatience and inattentiveness.

xx. Multivariate approach used in the analysis doesn't treat the Big Five traits as independent but establishes intercorrelations b/w these factors.

xxi. Agreeableness & Conscientiousness tends to be positively correlated & shows negative correlation with Neuroticism.

xxii. Age created stronger relation with time-related features and lessened role of Agreeableness & Neuroticism.

xxiii. Agreeableness, Consientiousness increases with age and Neuroticism decreases with Age.

Insights from study12pilots_personality_id_cleaned CSV File:

Skewness of Personality Traits:

- Agreeableness, Conscientiousness, and Openness exhibit left skewness, indicating that the majority of data points are located towards the higher end of the scale.
- Neuroticism displays right skewness, implying that most data points are concentrated towards the lower end of the scale.

Outlier Detection:

- Outliers were identified using Boxplot & IQR Method.
- Few outliers were observed in the Openness trait; however, they were retained considering their potential importance.

Correlation Analysis:

- Scatter plot analysis between Age and Big Five Inventory (BFI) traits (Agreeableness, Conscientiousness, Extraversion, Neuroticism, and Openness) revealed a neutral relationship.
- No significant correlation was found between Age and any of the personality traits, suggesting no clear trend of trait changes with age.

Correlation Among Personality Traits:

- Pairplot and Heatmap analysis indicated positive intermediate correlations between Openness and Extraversion, Agreeableness, and Conscientiousness.
- Openness exhibited a negative weak correlation with Neuroticism, suggesting that as openness increases, neuroticism tends to decrease.

Trait Correlations with Neuroticism:

- Neuroticism showed negative intermediate correlations with Extraversion, Agreeableness, and Conscientiousness, and a negative weak correlation with Openness.

Descriptive Statistics:

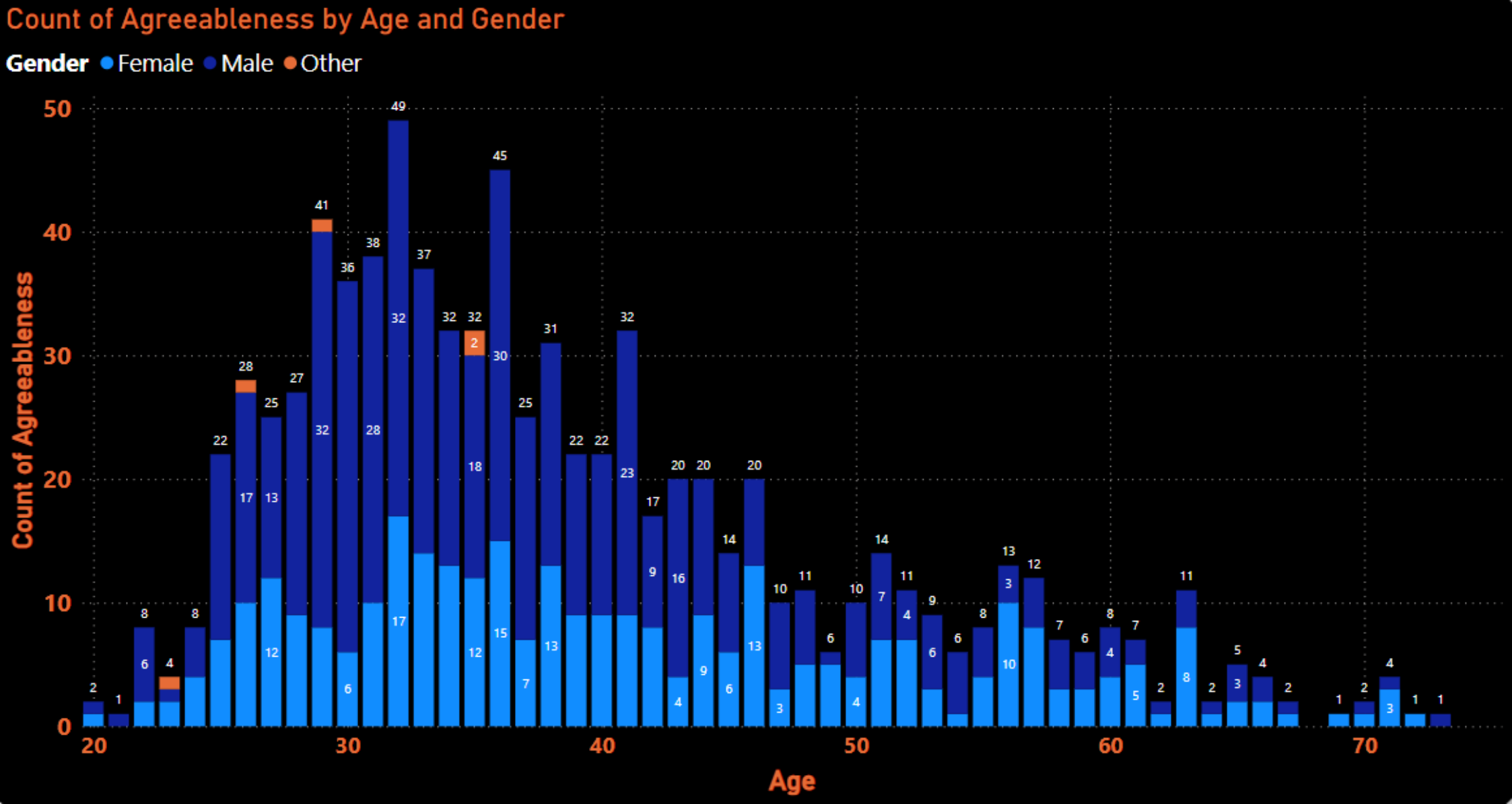
- Utilized the Describe function to obtain statistics such as Mean, Median, Quartiles, Minimum, and Maximum scores for each trait.

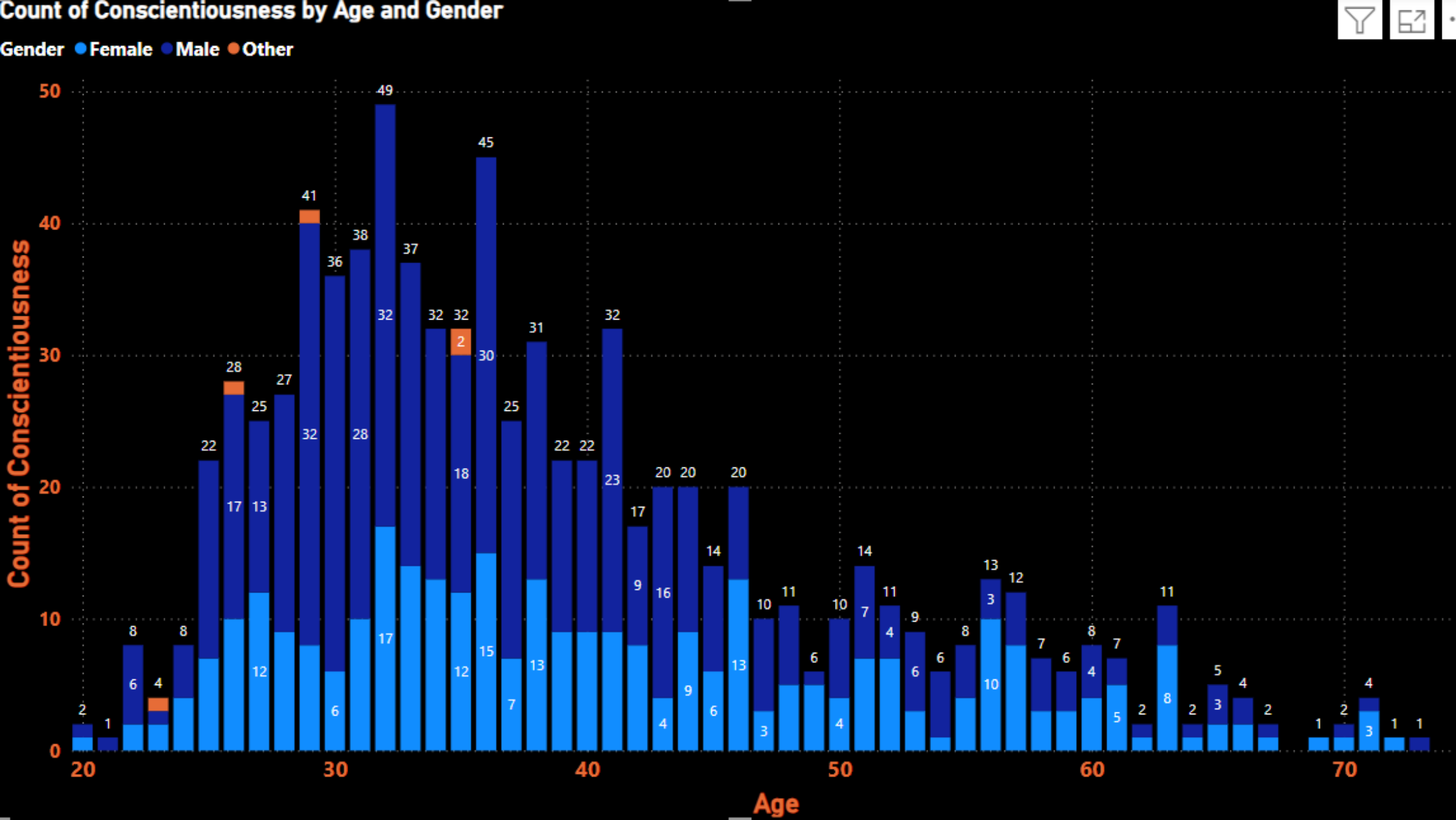
Distribution Analysis:

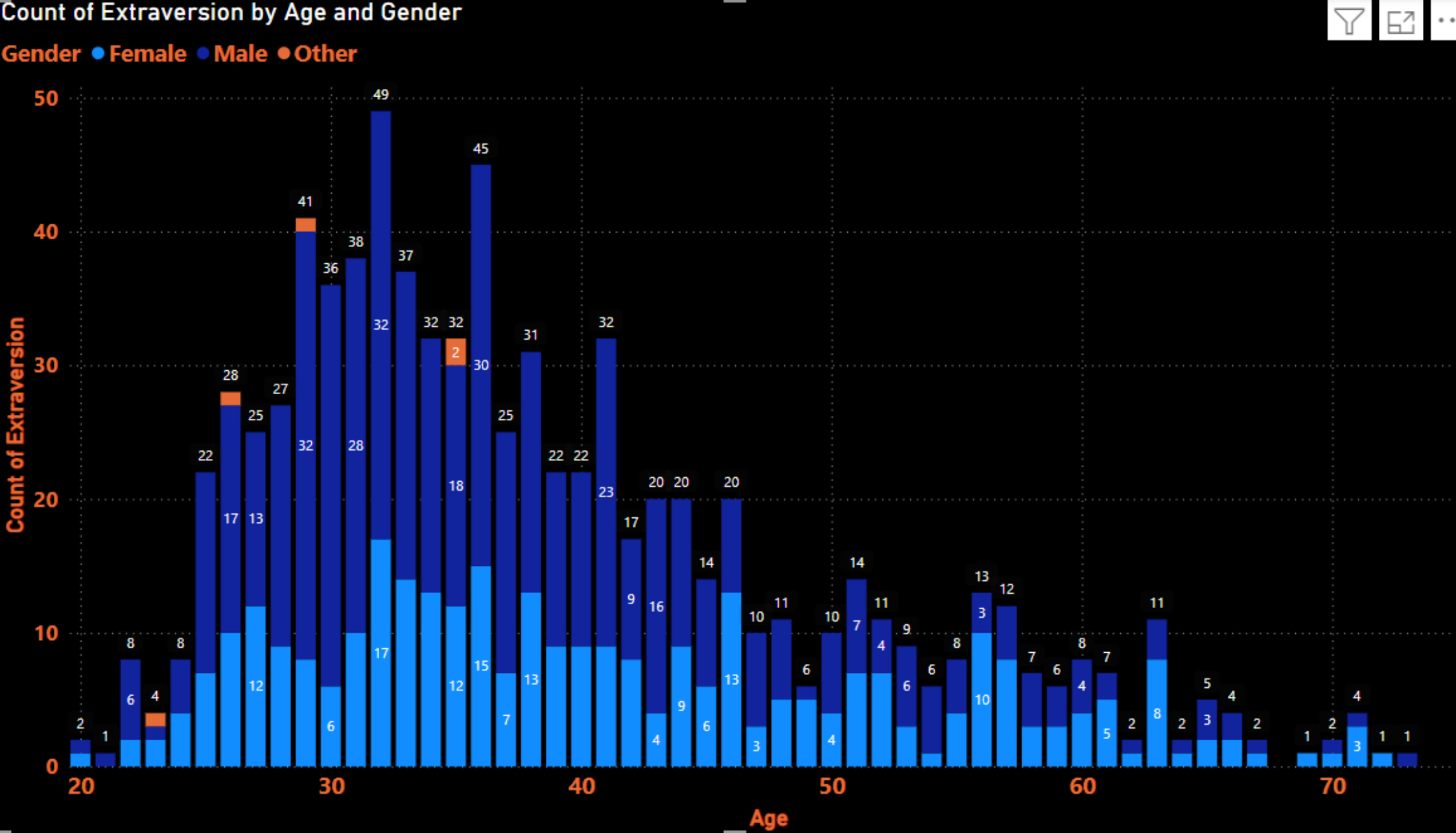
- Histograms were used to analyze the distribution of personality traits.

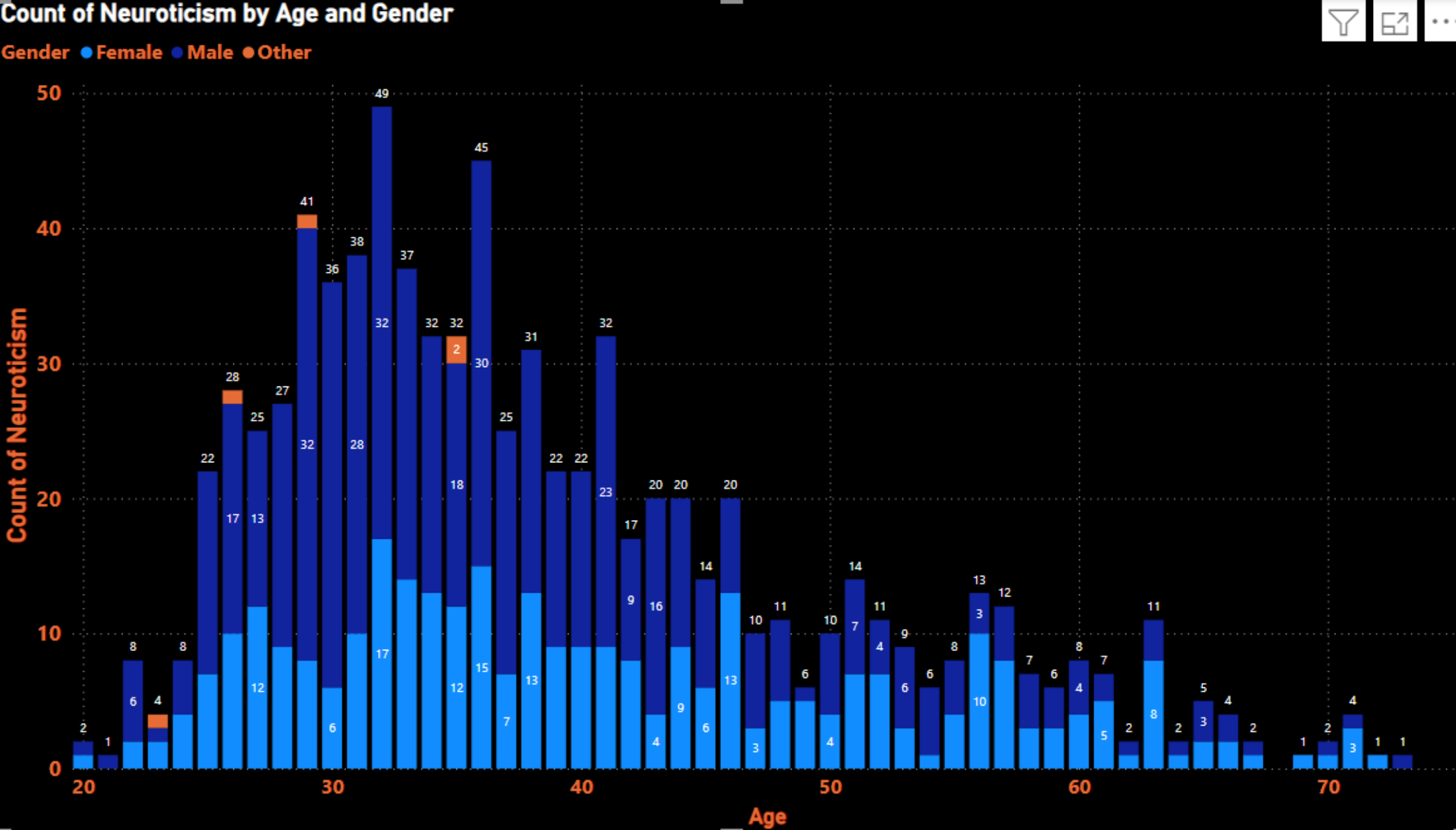
Gender-Based Normal Distribution:

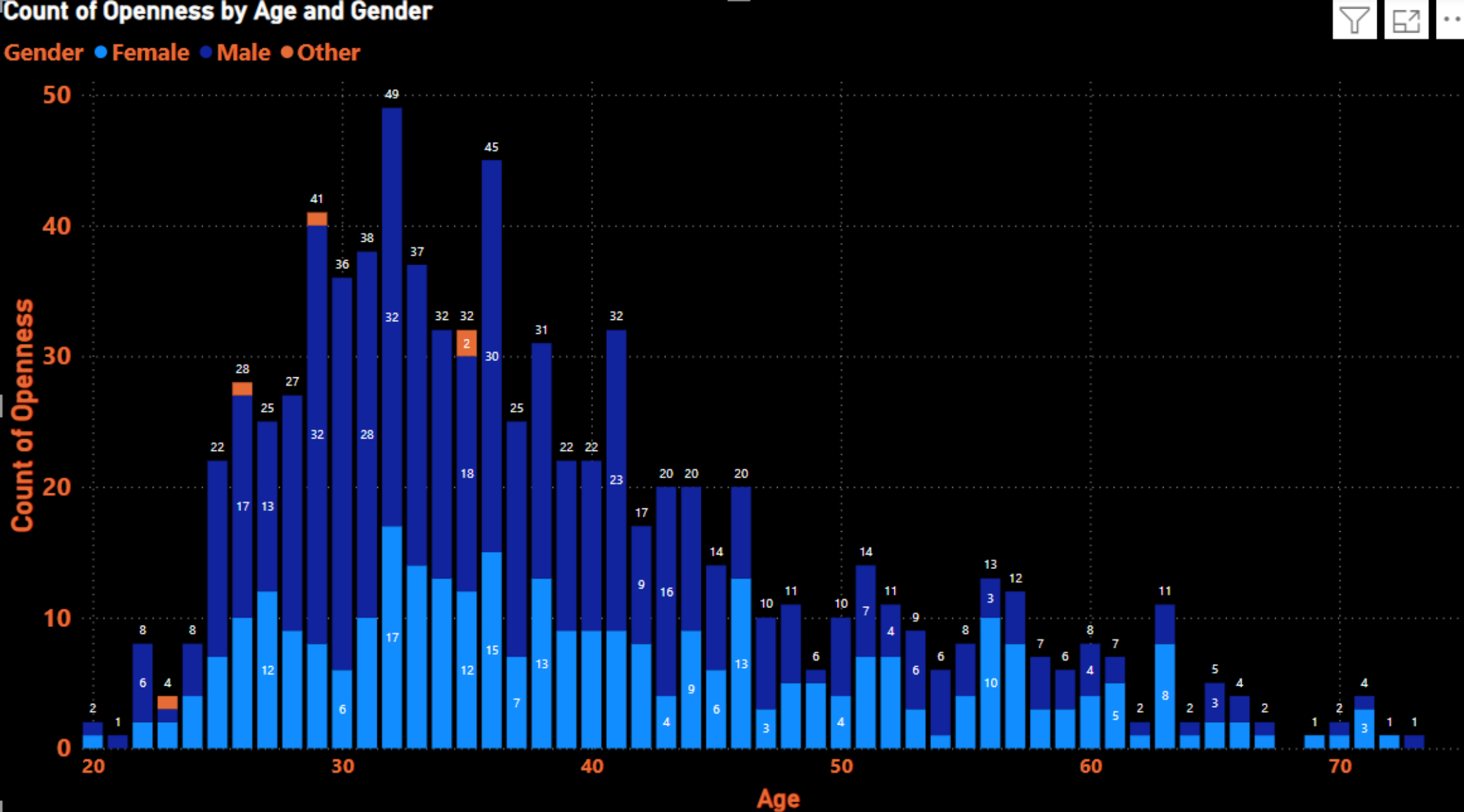
- Normal distribution graphs were plotted for each trait by gender.
- Male and Female mean scores for Openness were comparable, indicating a similar level of frankness.
- Male mean score for Extraversion was higher than Female, suggesting greater sociability and liveliness in males.
- Female mean score for Conscientiousness exceeded Male, indicating a higher level of responsibility and diligence in females.
- Female mean score for Neuroticism surpassed Male, suggesting higher susceptibility to anxiety, depression, and anger in females.
- Female mean score for Agreeableness exceeded Male, indicating a higher level of cooperativeness, politeness, trust, and friendliness in females.


Count of Agreeableness by Age and Gender
Gender Female Male Other
Count of Agreeableness
Age
0
10
20
30
40
50
20
30
40
50
60
70


Count of Conscientiousness by Age and Gender
Gender Female Male Other
Count of Conscientiousness
Age
0
10
20
30
40
50
20
30
40
50
60
70


Count of Extraversion by Age and Gender
Gender Female Male Other
Count of Extraversion
Age
0
10
20
30
40
50
20
30
40
50
60
70


Count of Neuroticism by Age and Gender
Gender Female Male Other
Count of Neuroticism
Age
0
10
20
30
40
50
20
30
40
50
60
70


Count of Openness by Age and Gender
Gender Female Male Other
Count of Openness
Age
0
10
20
30
40
50
20
30
40
50
60
70

Importing Modules

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

Importing the csv file

In [2]:
```
dataset = pd.read_csv('study12pilots_personality_id_cleaned.csv')
dataset.head()
```

Out[2]:

	Extraversion	Agreeableness	Conscientiousness	Neuroticism	Openness	BirthYear	Gender	Study	Age
0	37	39	35	8	33	1987	Male	pilot1_1	33
1	34	39	35	11	31	1960	Male	pilot1_1	60
2	30	37	32	10	27	1968	Female	pilot1_1	52
3	8	33	35	15	11	1977	Female	pilot1_1	43
4	19	27	27	22	18	1988	Female	pilot1_1	32

Dropping ID Column

In [3]:
```
dataset = dataset.drop(['pseudo_workerID'], axis = 1)
```

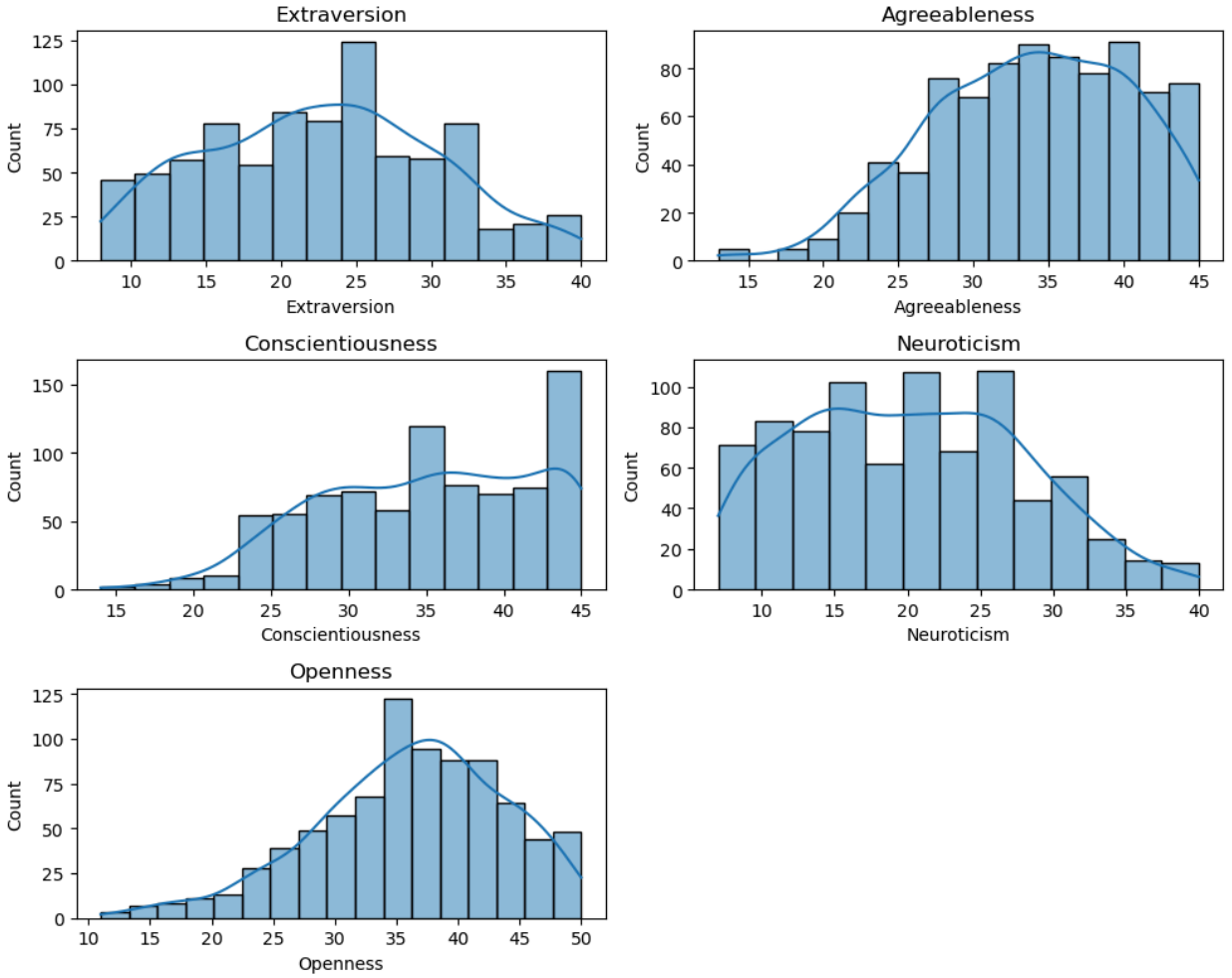
In [4]:
```
dataset.head()
```

Out[4]:

	Extraversion	Agreeableness	Conscientiousness	Neuroticism	Openness	BirthYear	Gender	Study	Age
0	37	39	35	8	33	1987	Male	pilot1_1	33
1	34	39	35	11	31	1960	Male	pilot1_1	60
2	30	37	32	10	27	1968	Female	pilot1_1	52
3	8	33	35	15	11	1977	Female	pilot1_1	43
4	19	27	27	22	18	1988	Female	pilot1_1	32

Analysing Normal Distribution for each traits , checking for skewness

In [5]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Set the size of each subplot
plt.figure(figsize=(10, 8))

# Define the layout of subplots
rows = 3
cols = 2

# List of variable names
variables = ['Extraversion', 'Agreeableness','Conscientiousness','Neuroticism','Openness
for i, var in enumerate(variables, 1):
    plt.subplot(rows, cols, i)
    sns.histplot(dataset[var], kde=True)
    plt.title(var)

plt.tight_layout()
plt.show()
```

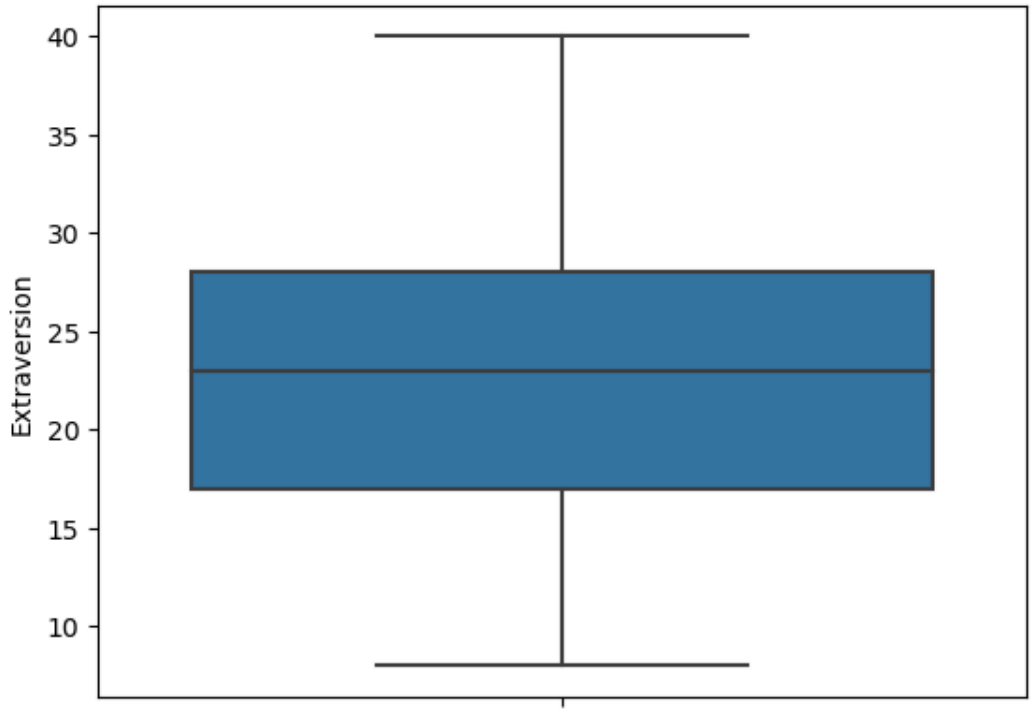


In [6]:
```
#'Agreeableness','Conscientiousness','Openness' are left skewed means most of the data p
#located on the right side
#Neuroticism is right skewed it means most of the data points are located in the left s
```

Utilizing Boxplot & IQR Method for each Personality Trait to identity Outliers

In [7]:
```
sns.boxplot(y = 'Extraversion', data = dataset)
```

Out[7]: <Axes: ylabel='Extraversion'>

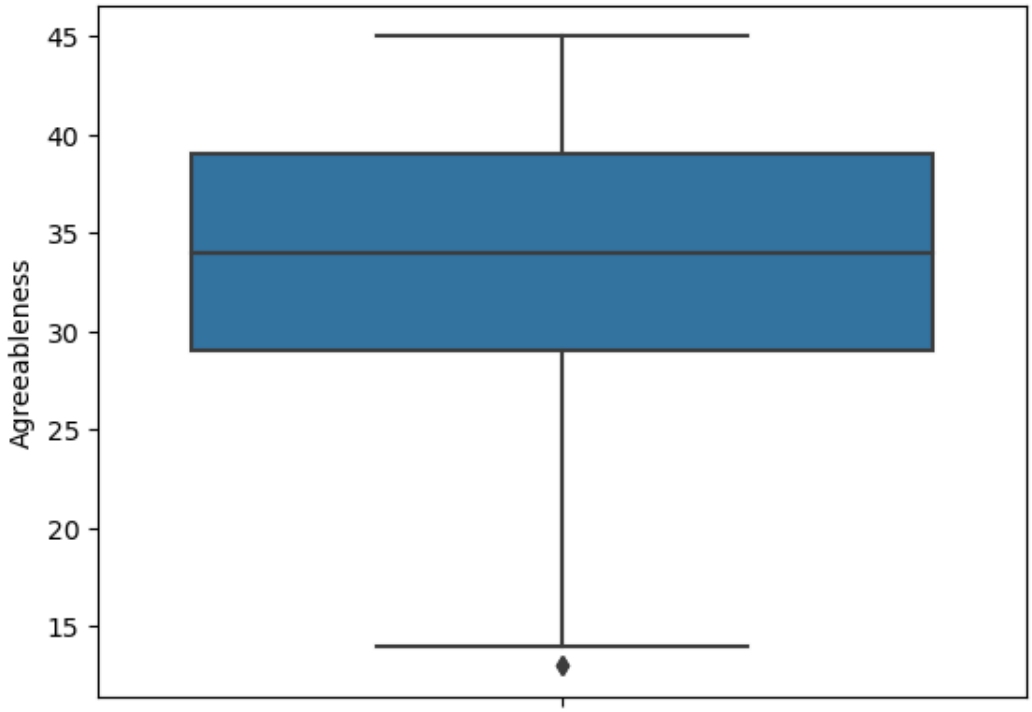


In [8]:
```
Q1 = dataset['Extraversion'].quantile(0.25)
Q3 = dataset['Extraversion'].quantile(0.75)
IQR = Q3-Q1
IQR
upper_limit = Q3 + (1.5*IQR)
lower_limit = Q1 - (1.5*IQR)
print(upper_limit)
print(lower_limit)
```

```
44.5
0.5
```

In [9]:
```
sns.boxplot(y = 'Agreeableness', data = dataset)
```

Out[9]: <Axes: ylabel='Agreeableness'>

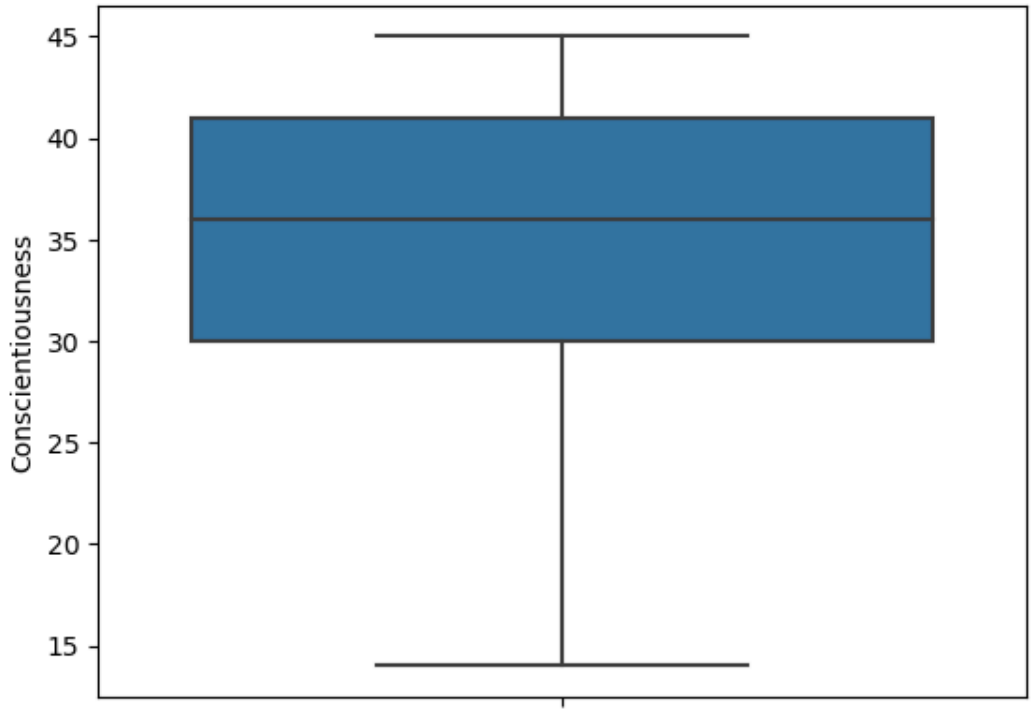


In [10]:
```
Q1 = dataset['Agreeableness'].quantile(0.25)
Q3 = dataset['Agreeableness'].quantile(0.75)
IQR = Q3-Q1
IQR
upper_limit = Q3 + (1.5*IQR)
lower_limit = Q1 - (1.5*IQR)
print(upper_limit)
print(lower_limit)
```

```
54.0
14.0
```

In [11]:
```
sns.boxplot(y = 'Conscientiousness', data = dataset)
```

Out[11]: <Axes: ylabel='Conscientiousness'>

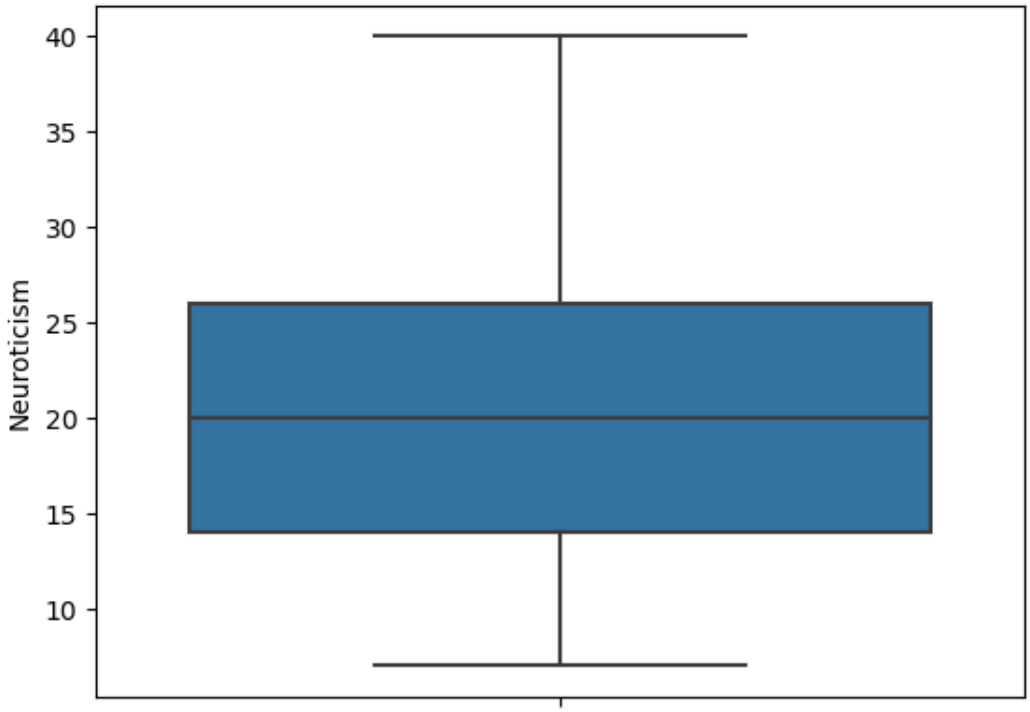


In [12]:
```
Q1 = dataset['Conscientiousness'].quantile(0.25)
Q3 = dataset['Conscientiousness'].quantile(0.75)
IQR = Q3-Q1
IQR
upper_limit = Q3 + (1.5*IQR)
lower_limit = Q1 - (1.5*IQR)
print(upper_limit)
print(lower_limit)
```

```
57.5
13.5
```

In [13]:
```
sns.boxplot(y = 'Neuroticism', data = dataset)
```

Out[13]: <Axes: ylabel='Neuroticism'>

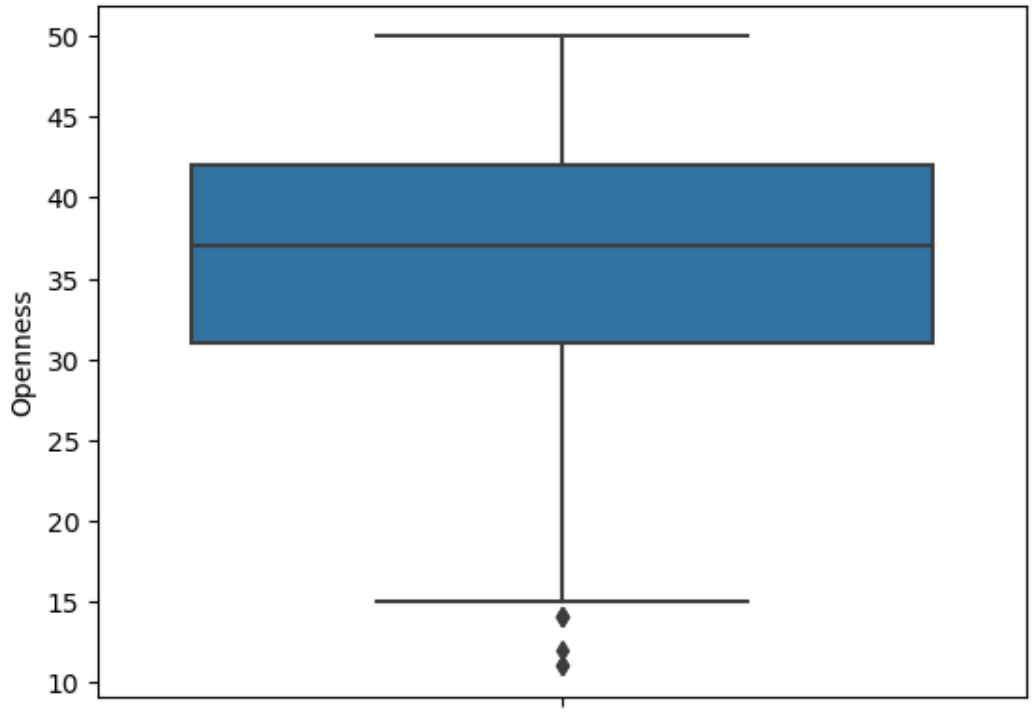


In [14]:
```
Q1 = dataset['Neuroticism'].quantile(0.25)
Q3 = dataset['Neuroticism'].quantile(0.75)
IQR = Q3-Q1
IQR
upper_limit = Q3 + (1.5*IQR)
lower_limit = Q1 - (1.5*IQR)
print(upper_limit)
print(lower_limit)
```

```
44.0
-4.0
```

In [15]:
```
sns.boxplot(y = 'Openness', data = dataset)
```

Out[15]: <Axes: ylabel='Openness'>

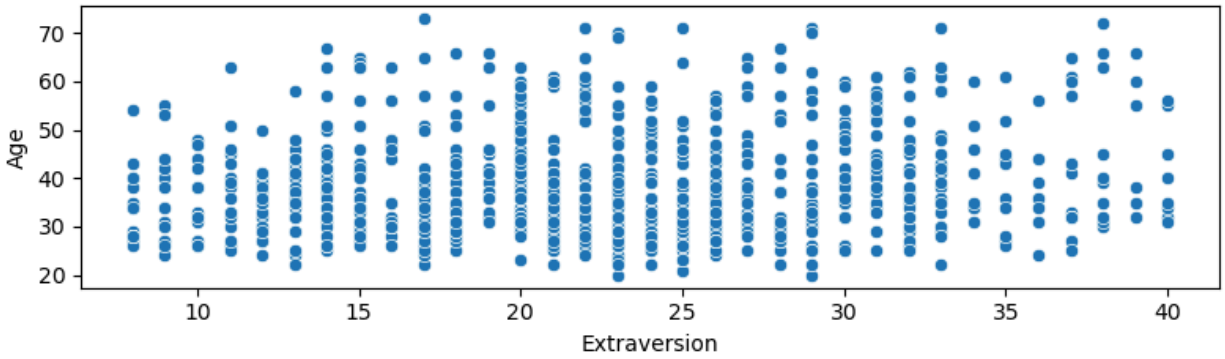


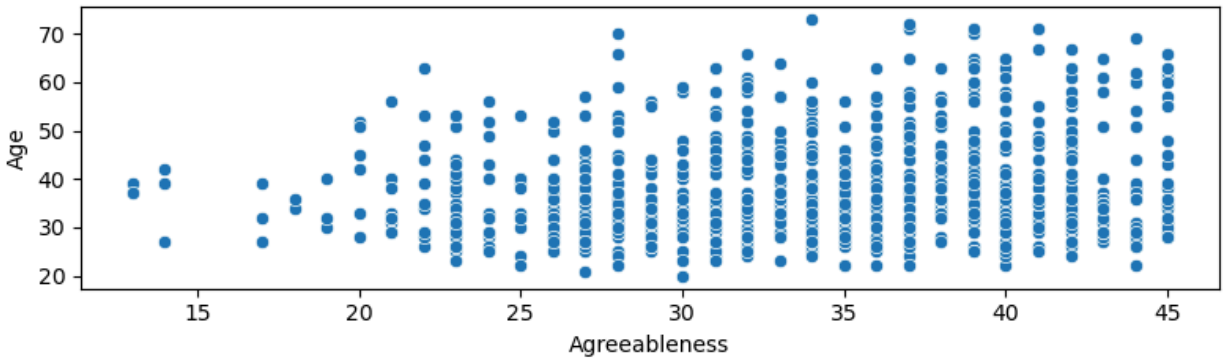
In [16]:
```
Q1 = dataset['Openness'].quantile(0.25)
Q3 = dataset['Openness'].quantile(0.75)
IQR = Q3-Q1
IQR
upper_limit = Q3 + (1.5*IQR)
lower_limit = Q1 - (1.5*IQR)
print(upper_limit)
print(lower_limit)
#Openness has outliers
```

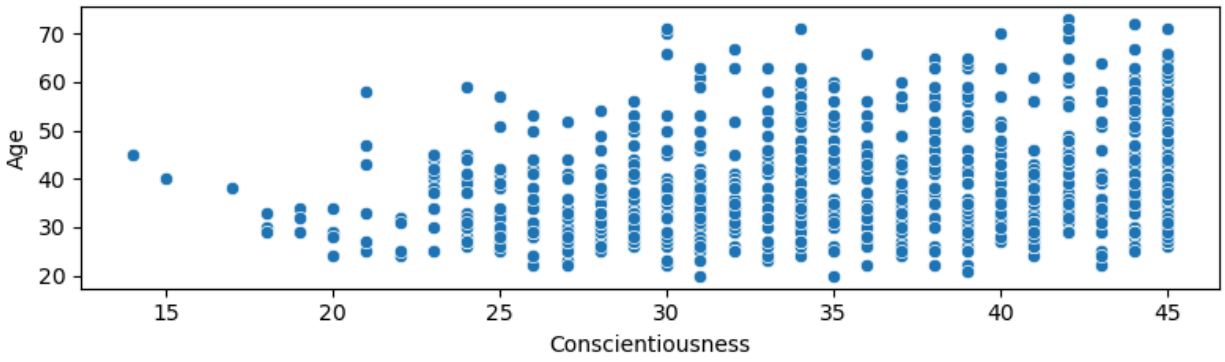
```
58.5
14.5
```

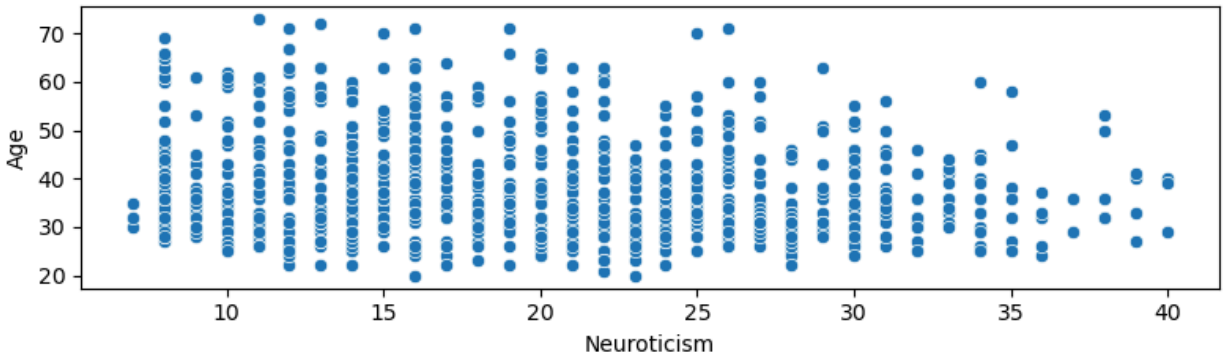
Checking Correlation b/w Age vs BFIs (Using Scatter Plot)

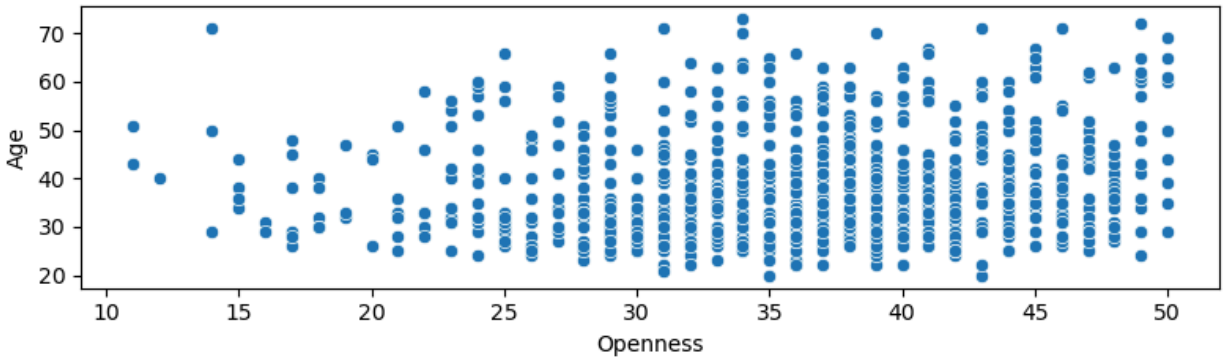
In [17]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming 'dataset' contains your data and you want to plot columns 'A', 'B', 'C', 'D',
variables = ['Extraversion', 'Agreeableness','Conscientiousness','Neuroticism','Openness

# Create scatter plots
fig, axes = plt.subplots(5, 1, figsize=(8, 12))  # Adjust figsize as needed

for i, var in enumerate(variables):
    sns.scatterplot(x=var, y='Age', data=dataset, ax=axes[i])  # Replace 'target_variab

plt.tight_layout()  # Adjust spacing between subplots
plt.show()
```


Age
Extraversion


Age
Agreeableness


Age
Conscientiousness

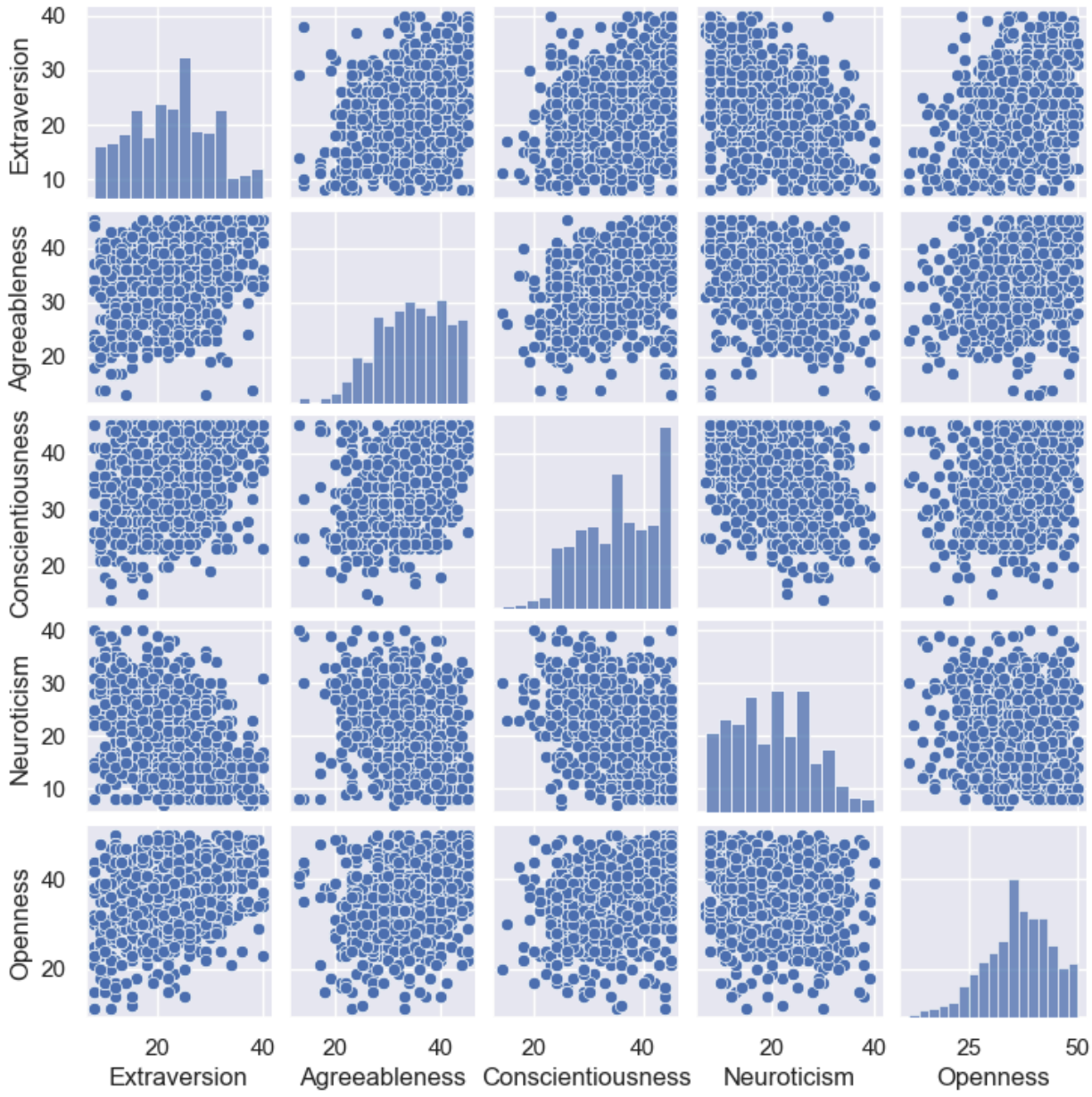

Age
Neuroticism


Age
Openness

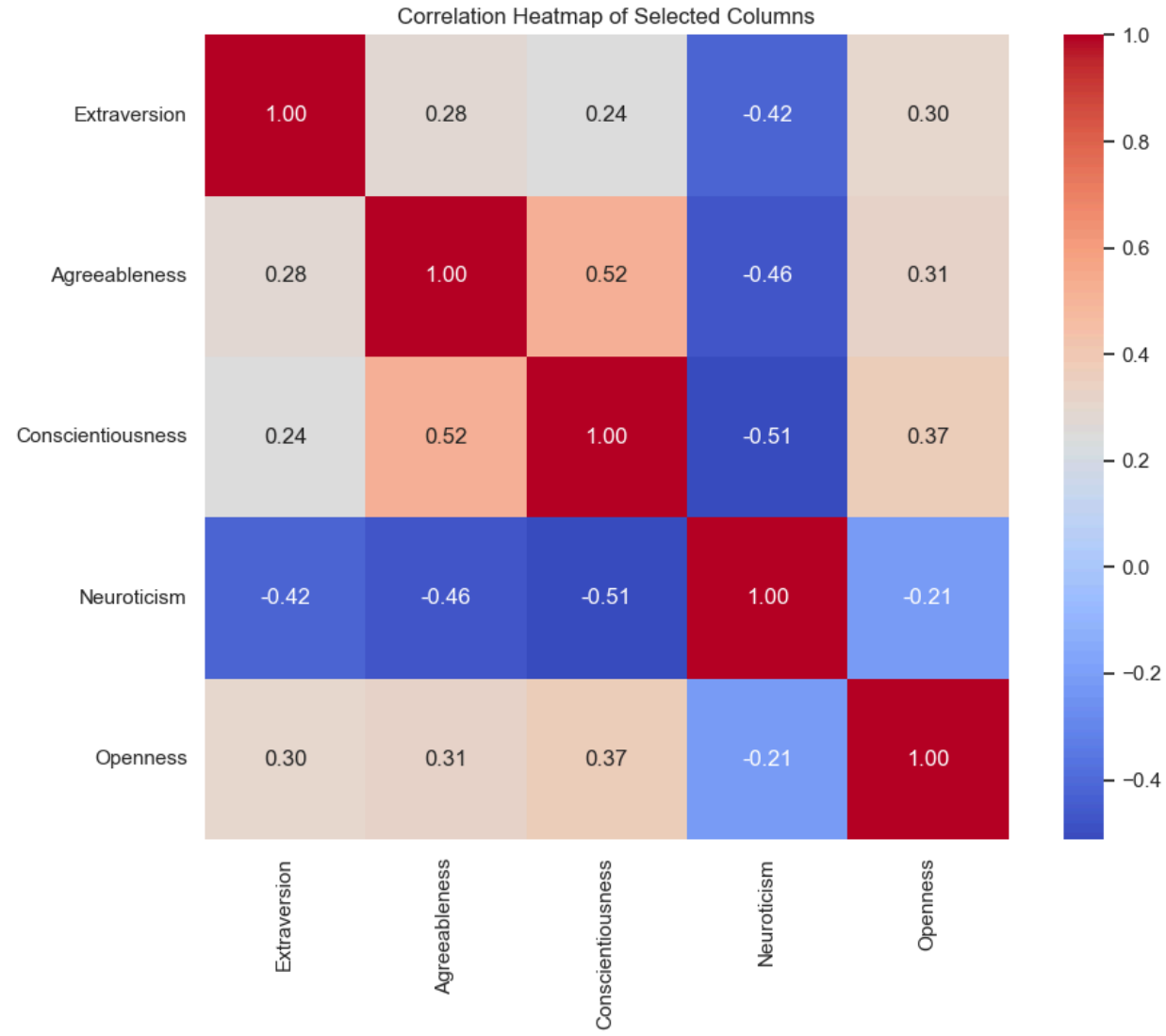
In [18]:
```
#No correlation b/w Age and 'Extraversion', 'Agreeableness','Conscientiousness','Neurot
```

Coorelation check for all the personality traits using Pairplot & Heatmap

In [19]:
```
sns.set()
cols = ['Extraversion', 'Agreeableness','Conscientiousness','Neuroticism','Openness']
sns.pairplot(dataset[cols], size = 1.5)
plt.show();
```



In [20]:
```
import seaborn as sns
import matplotlib.pyplot as plt

selected_columns = ['Extraversion', 'Agreeableness','Conscientiousness','Neuroticism','(
selected_data = dataset[selected_columns]
correlation_matrix = selected_data.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", square=True)
plt.title('Correlation Heatmap of Selected Columns')
plt.yticks(rotation=0)
plt.xticks(rotation=90)
plt.tight_layout()

# Show plot
plt.show()
```



In [21]:
```
#Openness means Frankness , Extraversion is a trait in which a person is more sociable,
#positive attitude,Agreeableness is a personality trait that can be described as coopera
#polite, kind, and friendly. People high in agreeableness are more trusting, affectiona
#Conscientiousness being responsible, careful & diligent
#Neuroticism -negative affects, including anger, anxiety, self-consciousness, irritabil
'''
--------------------------------------------------------------------------------
Openness & (Extraversion,Agreeableness,Conscientiousness) & vice versa= +ve intermediate
Openness & Neuroticism = -ve weak correlation

It means If openness increases the other traints like Extraversion, Agreeableness,Consci
increases
If frankness increases in a personality trait Neuroticism decreases
--------------------------------------------------------------------------------
Neuroticism has -ve intermediate correlation with Extraversion,Agreeableness & Conscient
& -ve weak correlation with openness
--------------------------------------------------------------------------------
'''
```

Out[21]:
```
'\n--------------------------------------------------------------------------------
--\nOpenness & (Extraversion,Agreeableness,Conscientiousness) & vice versa= +ve interm
ediate correlation\nOpenness & Neuroticism = -ve weak correlation\n\nIt means If openn
ess increases the other traints like Extraversion, Agreeableness,Conscientiousness als
o \nincreases\nIf frankness increases in a personality trait Neuroticism decreases\n--
------------------------------------------------------------------------------\nNeuro
ticism has -ve intermediate correlation with Extraversion,Agreeableness & Conscientiou
sness\n& -ve weak correlation with openness\n----------------------------------------
----------------------------------------\n'
```

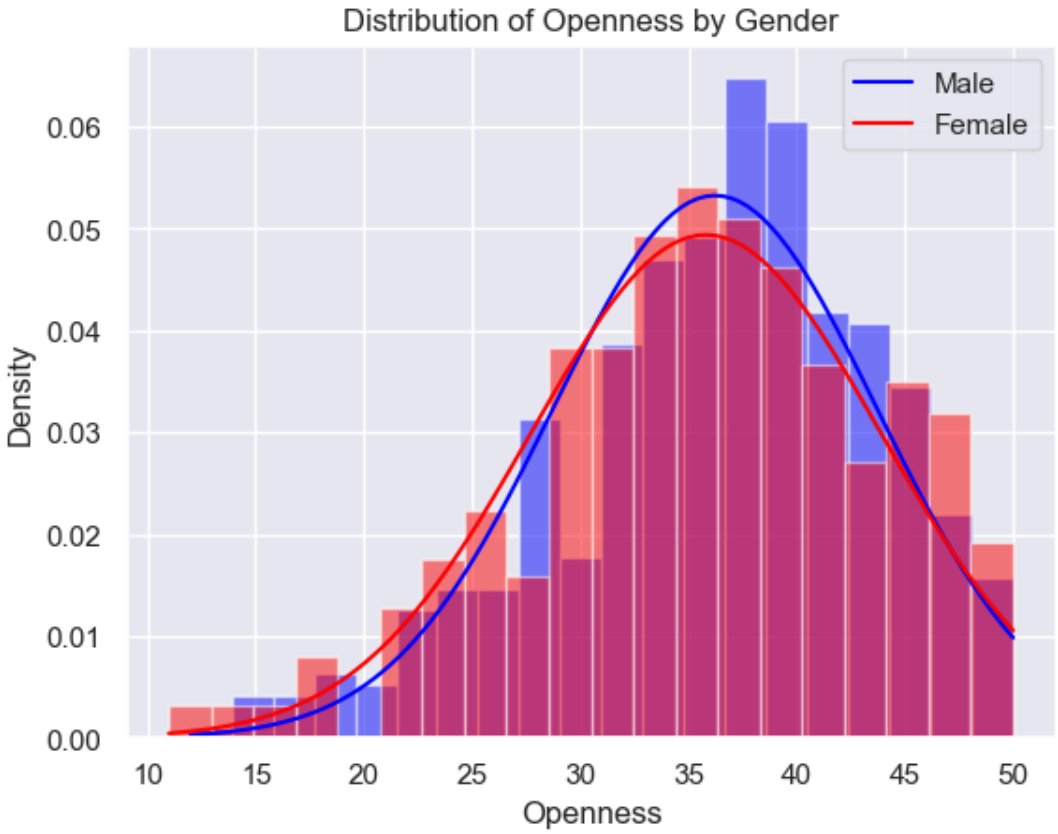
In [24]:
```
dataset.describe()
```

Out[24]:

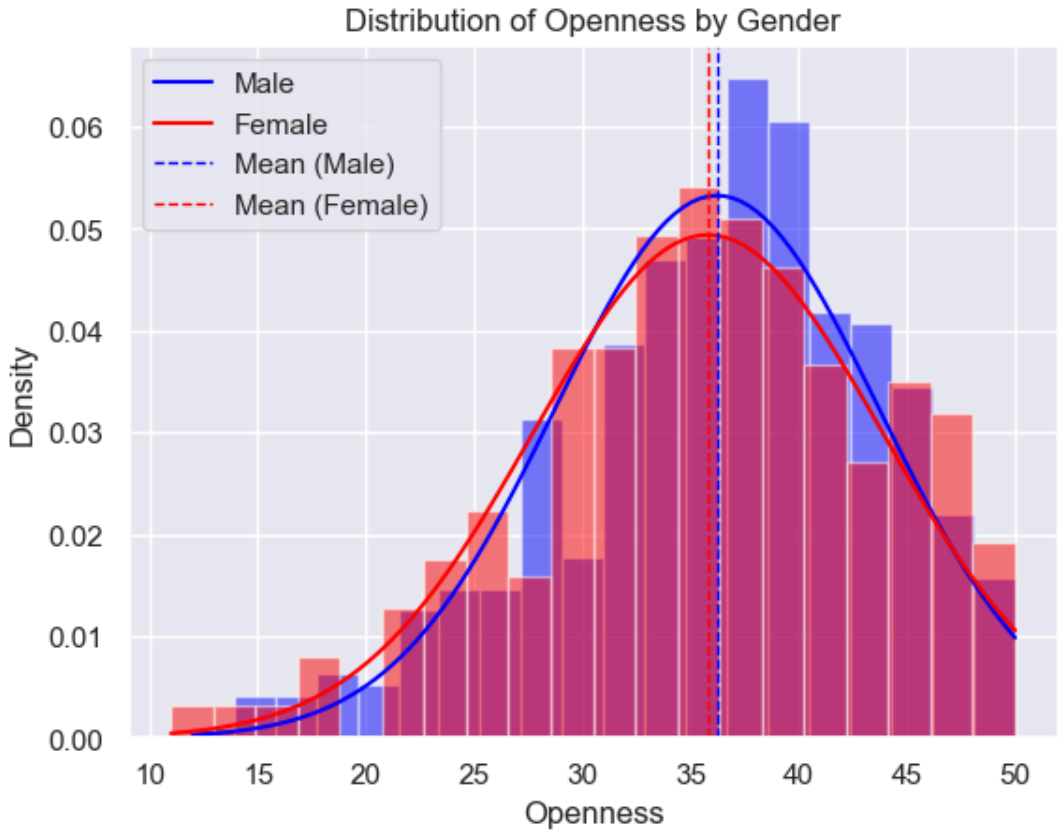
	Extraversion	Agreeableness	Conscientiousness	Neuroticism	Openness	BirthYear	Age
count	831.00000	831.000000	831.000000	831.000000	831.000000	831.000000	831.000000
mean	22.65704	33.790614	35.148014	20.216606	36.084236	1981.305656	38.694344
std	7.71786	6.626668	6.938872	7.676832	7.720892	10.821017	10.821017
min	8.00000	13.000000	14.000000	7.000000	11.000000	1947.000000	20.000000
25%	17.00000	29.000000	30.000000	14.000000	31.000000	1976.000000	31.000000
50%	23.00000	34.000000	36.000000	20.000000	37.000000	1984.000000	36.000000
75%	28.00000	39.000000	41.000000	26.000000	42.000000	1989.000000	44.000000
max	40.00000	45.000000	45.000000	40.000000	50.000000	2000.000000	73.000000

Building Histogram to observe Normal Distribution for each Personality Trait w.r.t to Gender

In [34]:
```
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import norm

# Assuming 'dataset' is your DataFrame containing 'Openness' and 'Gender' columns
# Subset the data by gender
openness_male = dataset.loc[dataset['Gender'] == 'Male', 'Openness']
openness_female = dataset.loc[dataset['Gender'] == 'Female', 'Openness']

# Calculate mean and standard deviation for each gender group
mean_male = np.mean(openness_male)
std_male = np.std(openness_male)

mean_female = np.mean(openness_female)
std_female = np.std(openness_female)

# Plot normal distribution curves
x_male = np.linspace(openness_male.min(), openness_male.max(), 100)
pdf_male = norm.pdf(x_male, mean_male, std_male)
plt.plot(x_male, pdf_male, label='Male', color='blue')

x_female = np.linspace(openness_female.min(), openness_female.max(), 100)
pdf_female = norm.pdf(x_female, mean_female, std_female)
plt.plot(x_female, pdf_female, label='Female', color='red')

# Overlay histograms
plt.hist(openness_male, bins=20, density=True, alpha=0.5, color='blue')
plt.hist(openness_female, bins=20, density=True, alpha=0.5, color='red')

plt.title('Distribution of Openness by Gender')
plt.xlabel('Openness')
plt.ylabel('Density')
plt.legend()
plt.grid(True)

plt.show()
```

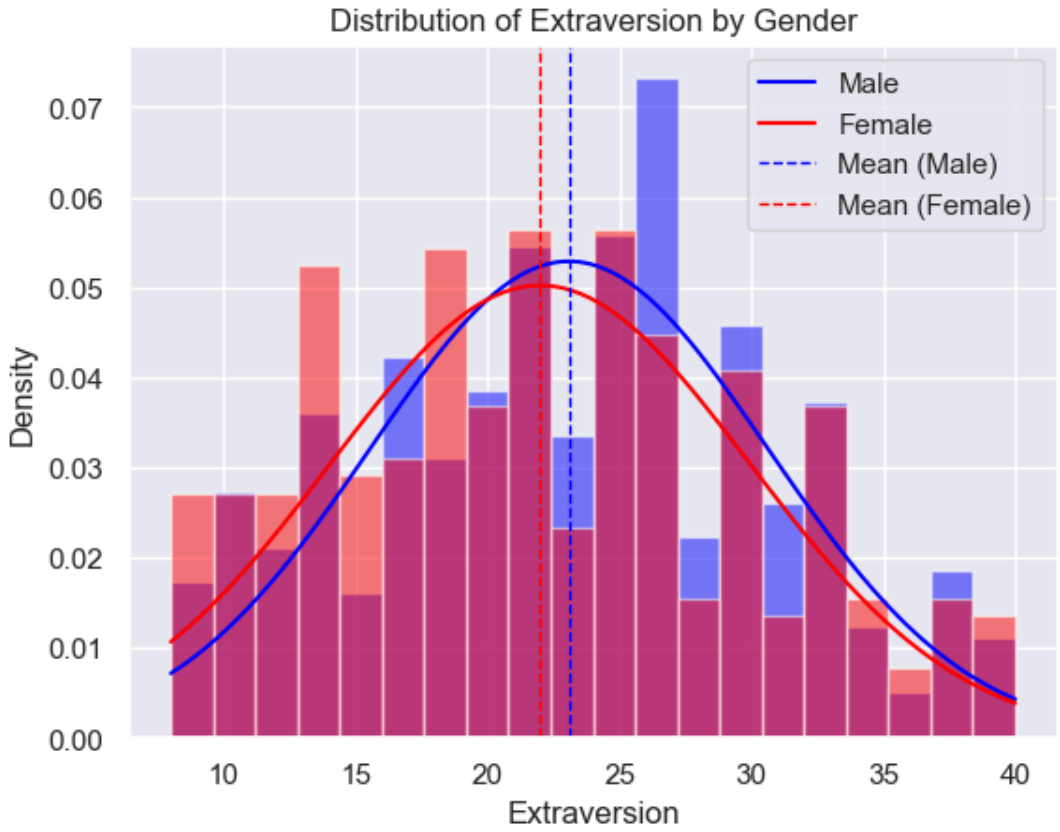

Distribution of Openness by Gender
Male
Female
Density
0.06
0.05
0.04
0.03
0.02
0.01
0.00
10
15
20
25
30
35
40
45
50
Openness

In [35]:
```
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import norm

# Assuming 'dataset' is your DataFrame containing 'Openness' and 'Gender' columns
# Subset the data by gender
openness_male = dataset.loc[dataset['Gender'] == 'Male', 'Openness']
openness_female = dataset.loc[dataset['Gender'] == 'Female', 'Openness']

# Calculate mean and standard deviation for each gender group
mean_male = np.mean(openness_male)
std_male = np.std(openness_male)

mean_female = np.mean(openness_female)
std_female = np.std(openness_female)

# Plot normal distribution curves
x_male = np.linspace(openness_male.min(), openness_male.max(), 100)
pdf_male = norm.pdf(x_male, mean_male, std_male)
plt.plot(x_male, pdf_male, label='Male', color='blue')

x_female = np.linspace(openness_female.min(), openness_female.max(), 100)
pdf_female = norm.pdf(x_female, mean_female, std_female)
plt.plot(x_female, pdf_female, label='Female', color='red')

# Overlay histograms
plt.hist(openness_male, bins=20, density=True, alpha=0.5, color='blue')
plt.hist(openness_female, bins=20, density=True, alpha=0.5, color='red')

# Plot mean lines
plt.axvline(mean_male, color='blue', linestyle='dashed', linewidth=1, label='Mean (Male
plt.axvline(mean_female, color='red', linestyle='dashed', linewidth=1, label='Mean (Fem

plt.title('Distribution of Openness by Gender')
plt.xlabel('Openness')
plt.ylabel('Density')
plt.legend()
plt.grid(True)

plt.show()

# Calculate z-scores for each data point
z_scores_male = (openness_male - mean_male) / std_male
z_scores_female = (openness_female - mean_female) / std_female

print("Z-scores (Male):", z_scores_male)
print("Z-scores (Female):", z_scores_female)
```




```
Z-scores (Male): 0     -0.431195
1      -0.698176
5      -1.499119
6      -1.365629
7      -0.965157
          ...
818     1.170690
821     0.236257
822     1.037200
827    -0.431195
830     0.236257
Name: Openness, Length: 504, dtype: float64
Z-scores (Female): 2     -1.090399
3      -3.071264
4      -2.204635
9      -1.833223
11     -1.585615
          ...
823     1.138075
824     0.395250
826     1.385683
828     0.766663
829     0.642858
Name: Openness, Length: 322, dtype: float64
```

In [42]:
```
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import norm

# Assuming 'dataset' is your DataFrame containing 'Openness' and 'Gender' columns
# Subset the data by gender
Extraversion_male = dataset.loc[dataset['Gender'] == 'Male', 'Extraversion']
Extraversion_female = dataset.loc[dataset['Gender'] == 'Female', 'Extraversion']

# Calculate mean and standard deviation for each gender group
mean_male = np.mean(Extraversion_male)
std_male = np.std(Extraversion_male)

mean_female = np.mean(Extraversion_female)
std_female = np.std(Extraversion_female)

# Plot normal distribution curves
x_male = np.linspace(Extraversion_male.min(), Extraversion_male.max(), 100)
pdf_male = norm.pdf(x_male, mean_male, std_male)
plt.plot(x_male, pdf_male, label='Male', color='blue')

x_female = np.linspace(Extraversion_female.min(), Extraversion_female.max(), 100)
pdf_female = norm.pdf(x_female, mean_female, std_female)
plt.plot(x_female, pdf_female, label='Female', color='red')

# Overlay histograms
plt.hist(Extraversion_male, bins=20, density=True, alpha=0.5, color='blue')
plt.hist(Extraversion_female, bins=20, density=True, alpha=0.5, color='red')

# Plot mean lines
plt.axvline(mean_male, color='blue', linestyle='dashed', linewidth=1, label='Mean (Male
plt.axvline(mean_female, color='red', linestyle='dashed', linewidth=1, label='Mean (Fem

plt.title('Distribution of Extraversion by Gender')
plt.xlabel('Extraversion')
plt.ylabel('Density')
plt.legend()
plt.grid(True)

plt.show()

# Calculate z-scores for each data point
z_scores_male = (Extraversion_male - mean_male) / std_male
z_scores_female = (Extraversion_female - mean_female) / std_female

print("Z-scores (Male):", z_scores_male)
print("Z-scores (Female):", z_scores_female)
```

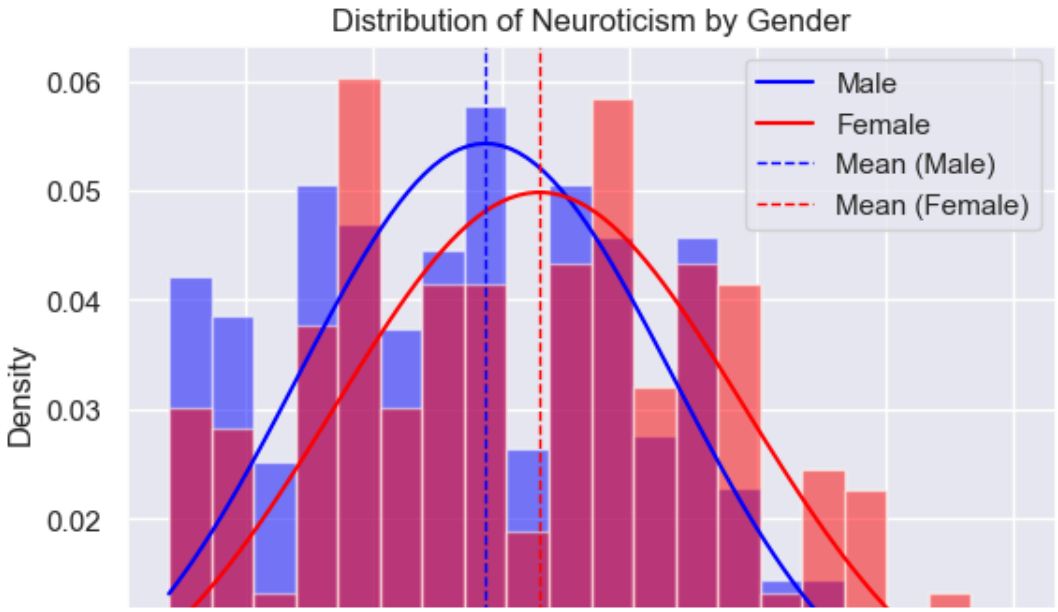



```
Z-scores (Male): 0      1.844029
1       1.446343
5      -2.000262
6      -0.939768
7      -0.011836
          ...
818     0.783535
821    -0.276959
822     1.313782
827    -0.011836
830    -0.542083
Name: Extraversion, Length: 504, dtype: float64
Z-scores (Female): 2      1.007546
3      -1.761057
4      -0.376755
9      -0.502601
11      0.000782
          ...
823     1.259238
824    -0.376755
826    -1.005983
828     0.252473
829    -0.502601
Name: Extraversion, Length: 322, dtype: float64
```

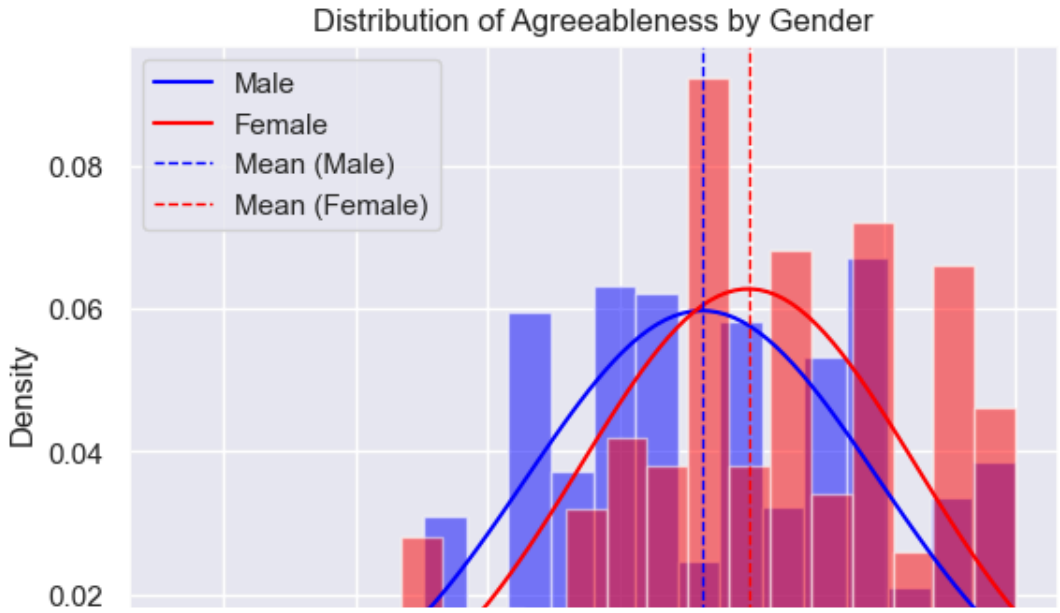
In [41]:
```
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import norm

# Assuming 'dataset' is your DataFrame containing 'Openness' and 'Gender' columns
# Subset the data by gender
Conscientiousness_male = dataset.loc[dataset['Gender'] == 'Male', 'Conscientiousness']
Conscientiousness_female = dataset.loc[dataset['Gender'] == 'Female', 'Conscientiousnes

# Calculate mean and standard deviation for each gender group
mean_male = np.mean(Conscientiousness_male)
std_male = np.std(Conscientiousness_male)

mean_female = np.mean(Conscientiousness_female)
std_female = np.std(Conscientiousness_female)

# Plot normal distribution curves
x_male = np.linspace(Conscientiousness_male.min(), Conscientiousness_male.max(), 100)
pdf_male = norm.pdf(x_male, mean_male, std_male)
plt.plot(x_male, pdf_male, label='Male', color='blue')

x_female = np.linspace(Conscientiousness_female.min(), Conscientiousness_female.max(),
pdf_female = norm.pdf(x_female, mean_female, std_female)
plt.plot(x_female, pdf_female, label='Female', color='red')

# Overlay histograms
plt.hist(Conscientiousness_male, bins=20, density=True, alpha=0.5, color='blue')
plt.hist(Conscientiousness_female, bins=20, density=True, alpha=0.5, color='red')

# Plot mean lines
plt.axvline(mean_male, color='blue', linestyle='dashed', linewidth=1, label='Mean (Male
plt.axvline(mean_female, color='red', linestyle='dashed', linewidth=1, label='Mean (Fem

plt.title('Distribution of Conscientiousness by Gender')
plt.xlabel('Conscientiousness')
plt.ylabel('Density')
plt.legend()
plt.grid(True)

plt.show()

# Calculate z-scores for each data point
z_scores_male = (Conscientiousness_male - mean_male) / std_male
z_scores_female = (Conscientiousness_female - mean_female) / std_female

print("Z-scores (Male):", z_scores_male)
print("Z-scores (Female):", z_scores_female)
```

```
7      0.067912
         ...
818    0.211124
821    0.640761
822    0.640761
827    1.356823
830    1.500035
Name: Conscientiousness, Length: 504, dtype: float64
Z-scores (Female): 2     -0.619107
3     -0.171794
4     -1.364630
9     -1.961047
11    -2.259256
         ...
823    1.319250
824   -0.768212
826    0.126415
828   -1.215525
829    0.126415
Name: Conscientiousness, Length: 322, dtype: float64
```

In [43]:
```
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import norm

# Assuming 'dataset' is your DataFrame containing 'Openness' and 'Gender' columns
# Subset the data by gender
Neuroticism_male = dataset.loc[dataset['Gender'] == 'Male', 'Neuroticism']
Neuroticism_female = dataset.loc[dataset['Gender'] == 'Female', 'Neuroticism']

# Calculate mean and standard deviation for each gender group
mean_male = np.mean(Neuroticism_male)
std_male = np.std(Neuroticism_male)

mean_female = np.mean(Neuroticism_female)
std_female = np.std(Neuroticism_female)

# Plot normal distribution curves
x_male = np.linspace(Neuroticism_male.min(), Neuroticism_male.max(), 100)
pdf_male = norm.pdf(x_male, mean_male, std_male)
plt.plot(x_male, pdf_male, label='Male', color='blue')

x_female = np.linspace(Neuroticism_female.min(), Neuroticism_female.max(), 100)
pdf_female = norm.pdf(x_female, mean_female, std_female)
plt.plot(x_female, pdf_female, label='Female', color='red')

# Overlay histograms
plt.hist(Neuroticism_male, bins=20, density=True, alpha=0.5, color='blue')
plt.hist(Neuroticism_female, bins=20, density=True, alpha=0.5, color='red')

# Plot mean lines
plt.axvline(mean_male, color='blue', linestyle='dashed', linewidth=1, label='Mean (Male
plt.axvline(mean_female, color='red', linestyle='dashed', linewidth=1, label='Mean (Fem

plt.title('Distribution of Neuroticism by Gender')
plt.xlabel('Neuroticism')
plt.ylabel('Density')
plt.legend()
plt.grid(True)

plt.show()

# Calculate z-scores for each data point
z_scores_male = (Neuroticism_male - mean_male) / std_male
z_scores_female = (Neuroticism_female - mean_female) / std_female

print("Z-scores (Male):", z_scores_male)
print("Z-scores (Female):", z_scores_female)
```

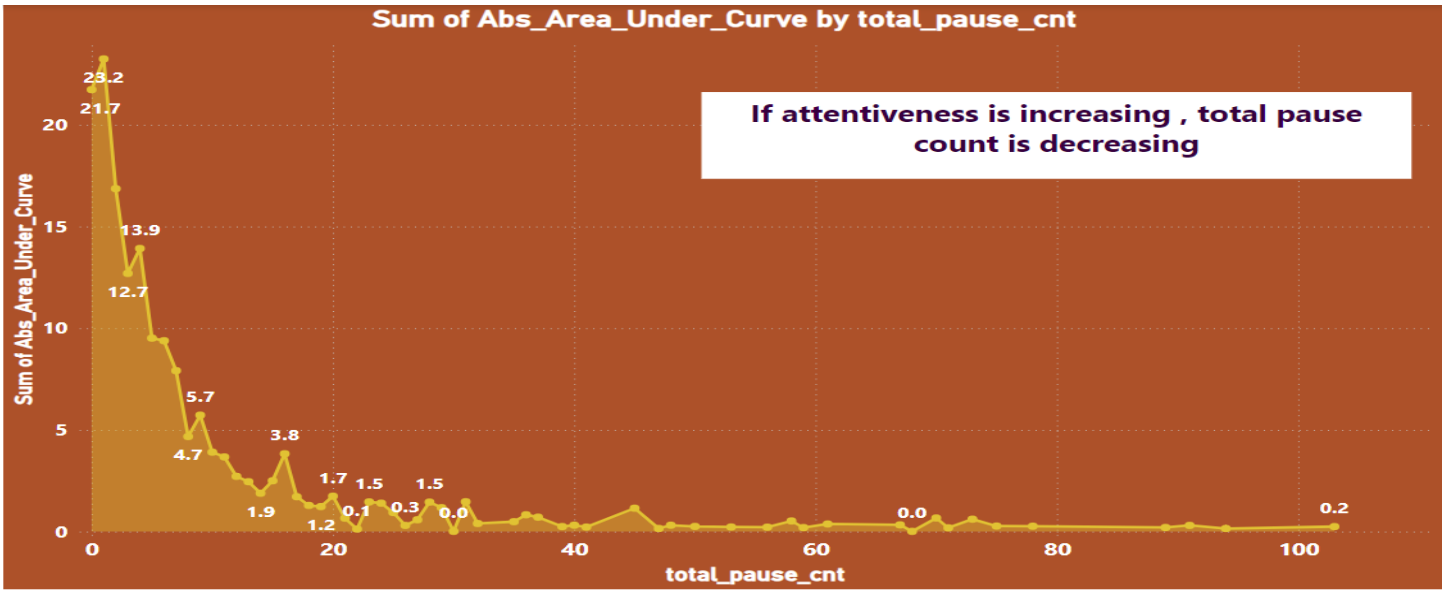

Distribution of Neuroticism by Gender
Male
Female
Mean (Male)
Mean (Female)
Density
0.06
0.05
0.04
0.03
0.02

In [44]:
```
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import norm

# Assuming 'dataset' is your DataFrame containing 'Openness' and 'Gender' columns
# Subset the data by gender
Agreeableness_male = dataset.loc[dataset['Gender'] == 'Male', 'Agreeableness']
Agreeableness_female = dataset.loc[dataset['Gender'] == 'Female', 'Agreeableness']

# Calculate mean and standard deviation for each gender group
mean_male = np.mean(Agreeableness_male)
std_male = np.std(Agreeableness_male)

mean_female = np.mean(Agreeableness_female)
std_female = np.std(Agreeableness_female)

# Plot normal distribution curves
x_male = np.linspace(Agreeableness_male.min(), Agreeableness_male.max(), 100)
pdf_male = norm.pdf(x_male, mean_male, std_male)
plt.plot(x_male, pdf_male, label='Male', color='blue')

x_female = np.linspace(Agreeableness_female.min(), Agreeableness_female.max(), 100)
pdf_female = norm.pdf(x_female, mean_female, std_female)
plt.plot(x_female, pdf_female, label='Female', color='red')

# Overlay histograms
plt.hist(Agreeableness_male, bins=20, density=True, alpha=0.5, color='blue')
plt.hist(Agreeableness_female, bins=20, density=True, alpha=0.5, color='red')

# Plot mean lines
plt.axvline(mean_male, color='blue', linestyle='dashed', linewidth=1, label='Mean (Male
plt.axvline(mean_female, color='red', linestyle='dashed', linewidth=1, label='Mean (Fem

plt.title('Distribution of Agreeableness by Gender')
plt.xlabel('Agreeableness')
plt.ylabel('Density')
plt.legend()
plt.grid(True)

plt.show()

# Calculate z-scores for each data point
z_scores_male = (Neuroticism_male - mean_male) / std_male
z_scores_female = (Neuroticism_female - mean_female) / std_female

print("Z-scores (Male):", z_scores_male)
print("Z-scores (Female):", z_scores_female)
```

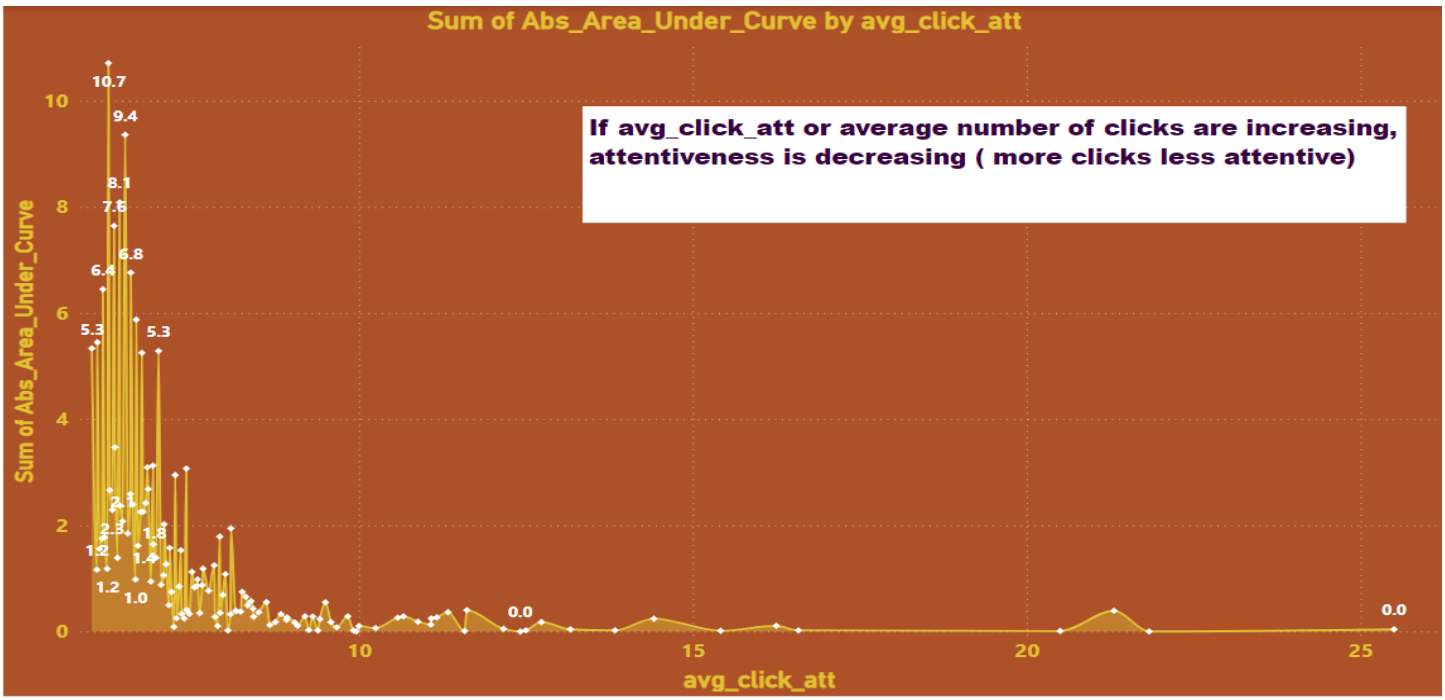

Distribution of Agreeableness by Gender
Male
Female
Mean (Male)
Mean (Female)
Density
0.08
0.06
0.04
0.02

Scientific Analysis of the Data Using Different Machine Learning Models and Power BI Reports

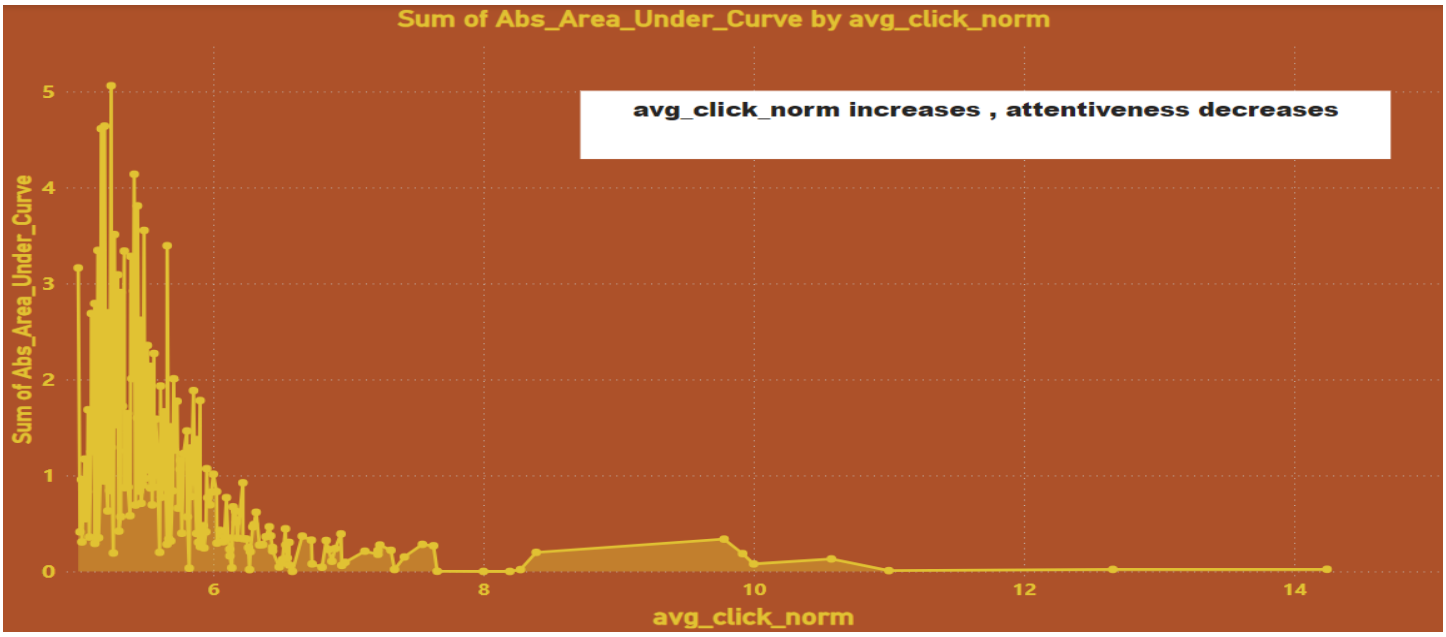
i. If attentiveness is high, pause count is less.



ii. If avg_click_att or average number of clicks are increasing, attentiveness decreasing.



iii. Attentiveness decreases with increase in average click norm



iv. Avg_fixation_cnt was positively correlated with Abs_area_under_curve (r = 0.11).

v. **Linear Regressor Model predicting personality traits from 11 Mouse related features**

R-Squared Value :
Extraversion : 0.028
Agreeableness : 0.026
Conscietiousness : 0.050
Neuroticism : 0.019
Openness : 0.041
R-Squared value indicates the amount of variation in the outcome variable (dependent variable) that can be explained by the predictor(independent variable).

vi. **PLS Analysis for predicting Big Five Personality Traits from 11 mouse related features & AUC.**

a) Coefficient of Determination **R2 – 0.03216**

b) ***Analysis showed Openness, Conscientiousness & Agreeableness are higher and Neuroticism is low*** when there is ***more attentiveness*** (Abs_Area_Under_Curve high) ,***lower values of total pause count , click related features like average number of clicks ,reclicks & lower euc_speed*** & ***higher values of average fixation count, avg_agg_fixation_dur.***

vii. **PLS Analysis for Big Five with Demographics ~ Mouse movements & AUC**

a) Coefficient of Determination **R2 – 0.0438**

b) ***Increase in Age*** showed ***increase in Openness, Conscientiousness & Agreeableness*** associated with ***less unnecessary clicks & reclicks, higher fixation duration & count & higher attentiveness.***

viii. **PLS Analysis for Big Five ~ Mouse movements ,Demographics & AUC**

a) Coefficient of Determination **R2 – 0.05522**

b) ***Agreeableness & Conscientiousness is increasing with increase in Age & Neuroticism decreases with increase in Age***

ix. **PLS Analysis for Big Five ~ Mouse movements**

a) Coefficient of Determination **R2 – 0.0198**

b) Analysis is similar to the PLS Analysis for Big Five from 11 Mouse related features & AUC.

c) ***Agreeableness, Conscientiousness and openness is increasing, Extraversion & Neuroticism is decreasing*** with ***lower click related activities, higher aggregate average Fixation duration & Fixation count & lower pause count.***

x. **Random Forest Regression Analysis for Big Five ~ Mouse Movements, Demographics & AUC**

a) Coefficient of Determination **R2 – 0.0049**

b) ***Top 5 features : avg_click_norm, avg_euc_speed, avg_fixation_cnt, Abs_Area_Under_Curve, avg_completion_time.***

xi. **Random Forest Regression Analysis for AUC ~ Mouse Movements**

a) Coefficient of Determination **R2 – 0.16**

b) Mean squared error **(MSE) : -0.011**

c) Random Forest is prediction AUC in an efficient way.

xii. **Effect sizes for PLS Analyses:**

a) Big Five ~ Mouse Movements and AUC:

U Scores representing Big Five Personality Traits.

V Scores representing Mouse Movements & AUC.

1. r = 0.29 indicates a ***moderate positive correlation*** b/w 2 sets of variables.

2. Pseudo R2 value for model predicting BF Personality traits from mouse movements & AUC is 0.89. ***Approx 89% of the variability in BF Personality traits is explained by mouse movements & AUC***.
3. Pseudo R2 value for the model prediction mouse movements & AUC from BF personality traits is 0.72. ***72% of variability in mouse movements & AUC is explained by BF personality traits***.

b) Big Five with demographics ~ Mouse movements & AUC

U Scores representing BF personality traits & demographics variables.

V Scores representing Mouse movements & AUC

1. r = 0.38 indicates ***moderate positive correlation*** b/w 2 sets of variables.
2. ***2.*** Pseudo R2 values for model predicting Big 5 personality traits with demographics from mouse movements & AUC is 0.83. ***83% of variability in combines BF personality traits with demographics is explained by mouse movements & AUC.***
3. ***3.*** Pseudo R2 value for model predicting mouse movements & AUC from BF personality traits with demographics is 0.70. ***70% of variability in mouse movements & AUC is explained by BF with demographics.***

c) Big Five ~ Mouse movements

U Scores represents BF personality traits.

V Scores represents Mouse movements.

1. r = 0.22 indicating **moderate positive correlation** b/w 2 sets of variables.
2. Pseudo R2 value for model predicting BF personality traits from Mouse movements is 0.92. ***Approx 92% of variability in BF personality traits is explained by Mouse movements.***
3. ***3.*** Pseudo R2 value for model predicting Mouse movements from BF personality traits is 0.64. ***64% of variability in mouse movements is explained by BF.***

Enhancing Analysis: Innovative strategies for Inclusivity & Improvements

i. **Time-Based Analysis:** Incorporate the time of day to understand how mouse movement relates to personality traits at different times.

ii. **Long-Term Study:** Extend analysis over days, weeks, or months to capture trends and changes in behavior and personality over time.

iii. **Multimodal Fusion:** Combine mouse data with keyboard usage, eye-tracking, or biometric signals for a comprehensive view of behavior.

iv. **Other Personality Traits:** Other then Agreeableness, Openness, Extraversion, Neuroticism & Conscientiousness we can use different personality traits.

v. **Different tasks or combination of Tasks:** picking an activity other than image clicking

vi. **Different Cursor Devices:** This can lead to variability in raw data collected.

vii. **Cross-Cultural Comparison:** Explore how personality traits vary across different cultures, enhancing cultural understanding.

viii. **Real-Time Feedback:** Develop a system to analyze mouse behavior in real-time, providing instant personalized feedback.

These streamlined ideas offer innovative ways to enhance analysis inclusivity and effectiveness while maintaining clarity and conciseness.

Importing Libraries & Modules

In [4]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

Pdf to Text

In [40]:
```
!pip install PyPDF2
```

```
Requirement already satisfied: PyPDF2 in g:\data science -notes\anaconda\lib
\site-packages (3.0.1)
```

In [42]:
```
import PyPDF2

# Open the PDF file in read-binary mode
with open(r'C:\Users\hp\Downloads\Journal of Personality - 2022 - Meidenbauer
    # Create a PDF reader object
    pdf_reader = PyPDF2.PdfReader(file)

    # Extract text from each page
    text = ''
    for page_num in range(len(pdf_reader.pages)):
        page = pdf_reader.pages[page_num]
        text += page.extract_text()

# Print the extracted text
print(text)
```

```
online behaviors and personality traits. More specifically,
the research question of interest is “Are mouse movement
patterns exhibited in a choice- making task reflective of a
person's internal states and traits?”
Mouse cursor movements are a cost-  effective mea-
surement of individuals' behaviors. Mouse trajectories
have been used in previous research to track attention
in computer interactions (Rodden et al.,  2008) and mea-
sure website engagement (Arapakis & Leiva,  2016). For
example, in a study by Arapakis et al. (2014), several
speed-  and distance- based mouse movement features
showed significant correlations (coefficients between
0.37 and 0.4, N = 22) with participant ratings of how in-
teresting they found a news article that they were read-
ing. It has also been shown that mouse movements are
significantly correlated with individuals' attitudes. In
the context of evaluating implicit bias, researchers have
examined the time and trajectory from one location on
screen to another to evaluate hesitancy or “corrections”
to initial decisions (Freeman,  2018; Hehman et al., 2015;
```

Importing the csv file

In [43]:
```
dataset = pd.read_csv('personality_mouse_final_table (2).csv')
dataset.head()
```

Out[43]:

	Age	BirthYear	Gender	Extraversion	Agreeableness	Conscientiousness	Neuroticism	Open
0	45	1975	Female	20	33	40	34	
1	29	1991	Male	31	44	43	10	
2	58	1962	Male	13	31	21	35	
3	30	1990	Male	33	19	25	15	
4	46	1974	Male	14	38	33	21	

5 rows × 32 columns

Dropping ID columns

In [44]:
```
click_data = dataset.drop(['pseudo_workerID','Mturk_ID'], axis = 1)
```

In [45]:
```
click_data.head()
```

Out[45]:

	Age	BirthYear	Gender	Extraversion	Agreeableness	Conscientiousness	Neuroticism	Open
0	45	1975	Female	20	33	40	34	
1	29	1991	Male	31	44	43	10	
2	58	1962	Male	13	31	21	35	
3	30	1990	Male	33	19	25	15	
4	46	1974	Male	14	38	33	21	

5 rows × 30 columns

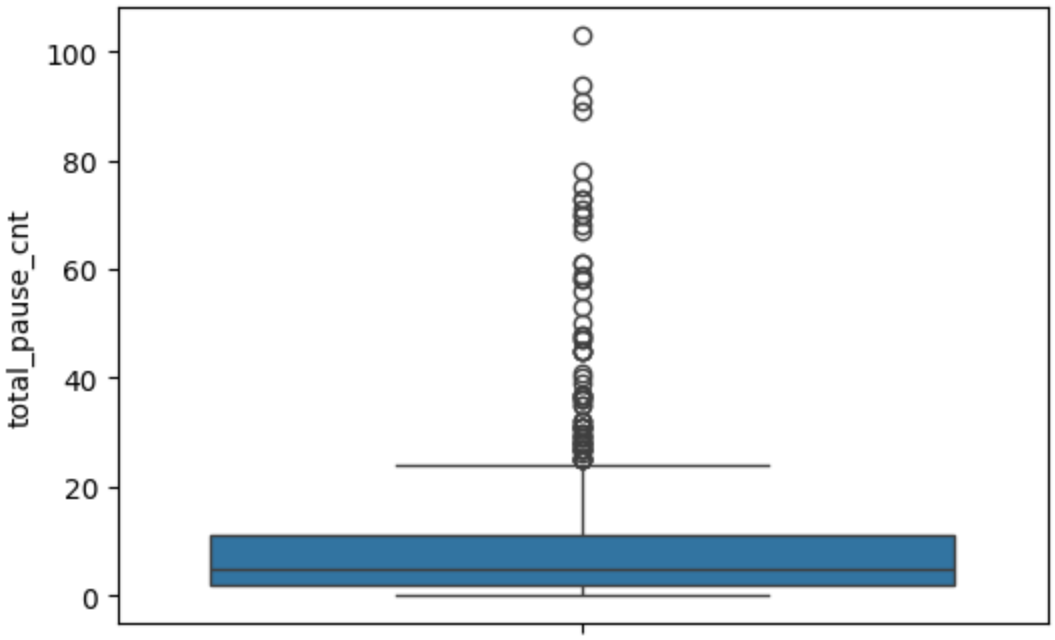
Finding null values

In [46]:
```
click_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 791 entries, 0 to 790
Data columns (total 30 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Age                   791 non-null    int64  
 1   BirthYear             791 non-null    int64  
 2   Gender                791 non-null    object 
 3   Extraversion          791 non-null    int64  
 4   Agreeableness         791 non-null    int64  
 5   Conscientiousness     791 non-null    int64  
 6   Neuroticism           791 non-null    int64  
 7   Openness              791 non-null    int64  
 8   Study                 791 non-null    object 
 9   avg_click_att         791 non-null    float64
 10  reclick_percent_att   791 non-null    float64
 11  avg_click_norm        791 non-null    float64
 12  reclick_percent_norm  791 non-null    float64
 13  avg_x_dist            791 non-null    float64
 14  avg_euc_dist          791 non-null    float64
 15  avg_euc_speed         791 non-null    float64
 16  avg_completion_time   791 non-null    float64
 17  total_pause_cnt       791 non-null    int64  
 18  description           791 non-null    object 
 19  Extraversion_N        791 non-null    float64
 20  Agreeableness_N       791 non-null    float64
 21  Conscientiousness_N   791 non-null    float64
 22  Neuroticism_N         791 non-null    float64
 23  Openness_N            791 non-null    float64
 24  avg_fixation_dur      791 non-null    float64
 25  avg_agg_fixation_dur  791 non-null    float64
 26  avg_fixation_cnt      791 non-null    float64
 27  Area_Under_Curve      791 non-null    float64
 28  Task_duration         791 non-null    float64
 29  Abs_Area_Under_Curve  791 non-null    float64
dtypes: float64(19), int64(8), object(3)
memory usage: 185.5+ KB
```

Outliers detection using Box Plot and IQR method

In [47]:
```
total_pause_cnt_data = dataset['total_pause_cnt']
plt.figure(figsize=(6, 4))
sns.boxplot(y=total_pause_cnt_data)
plt.show()
```

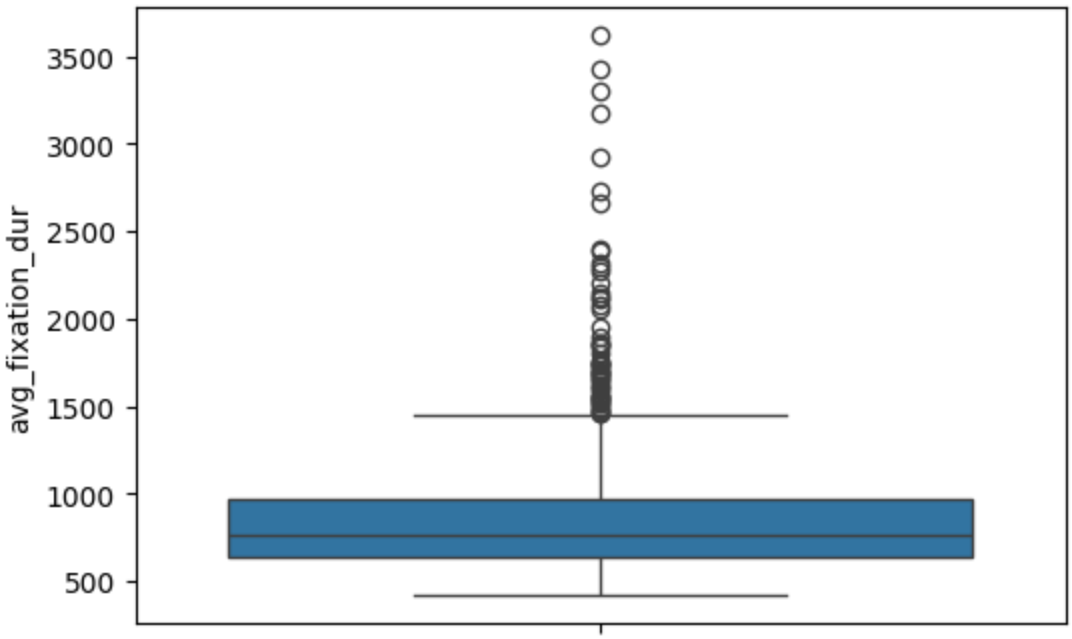


In [48]:
```
Q1 = dataset['total_pause_cnt'].quantile(0.25)
Q3 = dataset['total_pause_cnt'].quantile(0.75)
IQR = Q3-Q1
IQR
upper_limit = Q3 + (1.5*IQR)
lower_limit = Q1 - (1.5*IQR)
print(upper_limit)
print(lower_limit)
```

```
24.5
-11.5
```

In [ ]:
```
#Outliers present in total_pause_cnt
```

In [49]:
```
plt.figure(figsize=(6, 4))
sns.boxplot(dataset['avg_fixation_dur'])
plt.show()
```

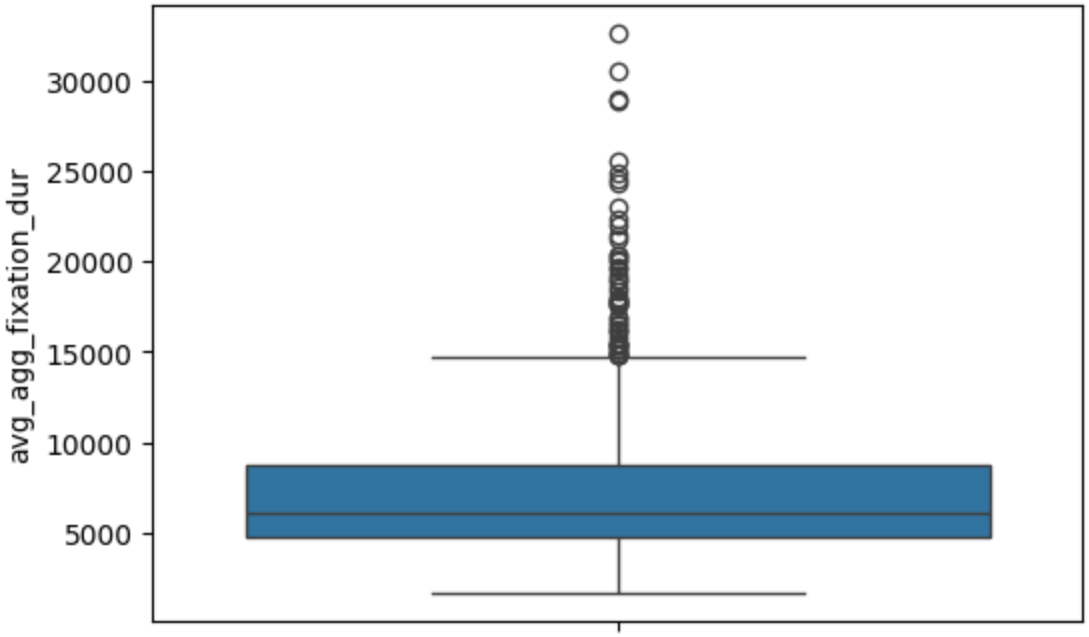


In [50]:
```
Q1 = dataset['avg_fixation_dur'].quantile(0.25)
Q3 = dataset['avg_fixation_dur'].quantile(0.75)
IQR = Q3-Q1
IQR
upper_limit = Q3 + (1.5*IQR)
lower_limit = Q1 - (1.5*IQR)
print(upper_limit)
print(lower_limit)
```

```
1457.7009471740776
147.44533477379173
```

In [ ]:
```
#Outliers present in avg_fixation_dur
```

In [51]:
```
plt.figure(figsize=(6, 4))
sns.boxplot(dataset['avg_agg_fixation_dur'])
plt.show()
```

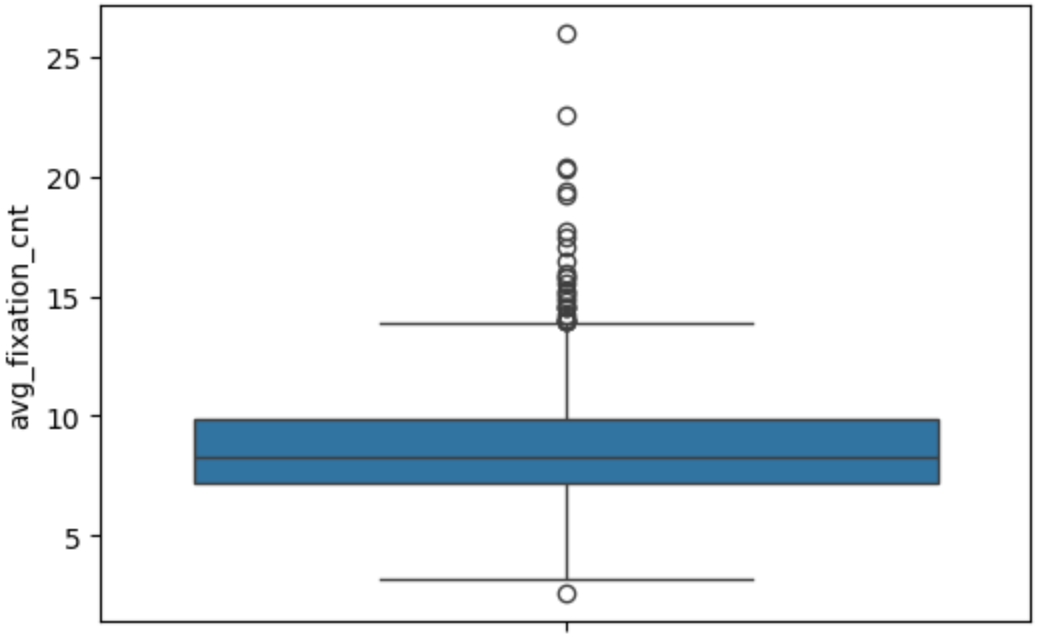


In [52]:
```
Q1 = dataset['avg_agg_fixation_dur'].quantile(0.25)
Q3 = dataset['avg_agg_fixation_dur'].quantile(0.75)
IQR = Q3-Q1
IQR
upper_limit = Q3 + (1.5*IQR)
lower_limit = Q1 - (1.5*IQR)
print(upper_limit)
print(lower_limit)
```

```
14705.787307032591
-1129.3436249285323
```

In [ ]:
```
#Outliers present in avg_agg_fixation_dur
```

In [53]:
```
plt.figure(figsize=(6, 4))
sns.boxplot(dataset['avg_fixation_cnt'])
plt.show()
```

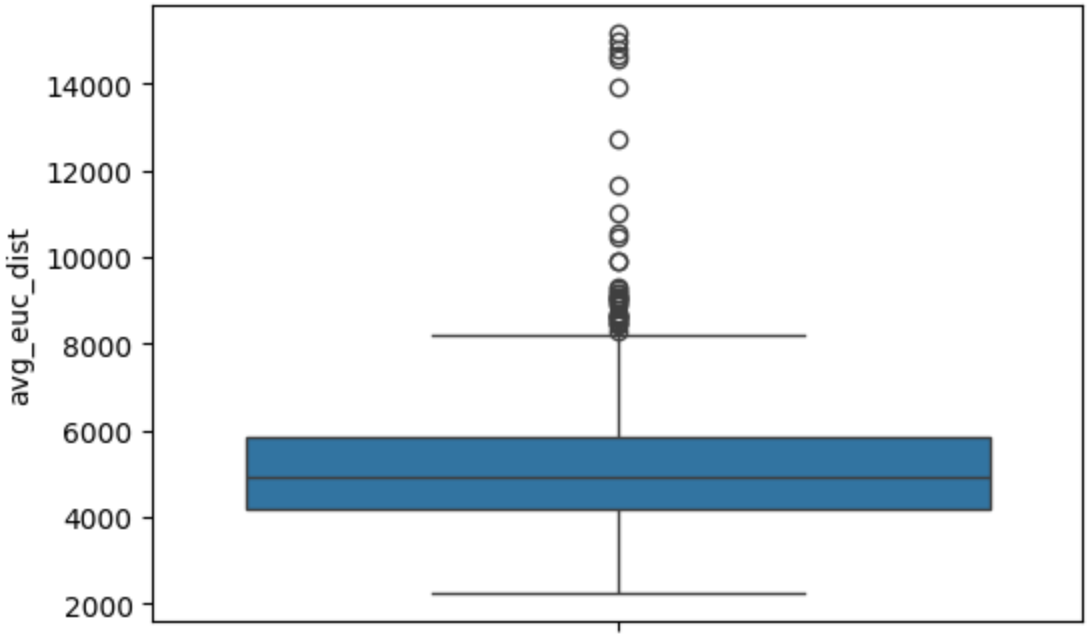


In [54]:
```
Q1 = dataset['avg_fixation_cnt'].quantile(0.25)
Q3 = dataset['avg_fixation_cnt'].quantile(0.75)
IQR = Q3-Q1
IQR
upper_limit = Q3 + (1.5*IQR)
lower_limit = Q1 - (1.5*IQR)
print(upper_limit)
print(lower_limit)
```

```
13.934819897084049
3.104250047646274
```

In [ ]:
```
#Outliers present in avg_fixation_cnt
```

In [55]:
```
plt.figure(figsize=(6, 4))
sns.boxplot(dataset['avg_euc_dist'])
plt.show()
```

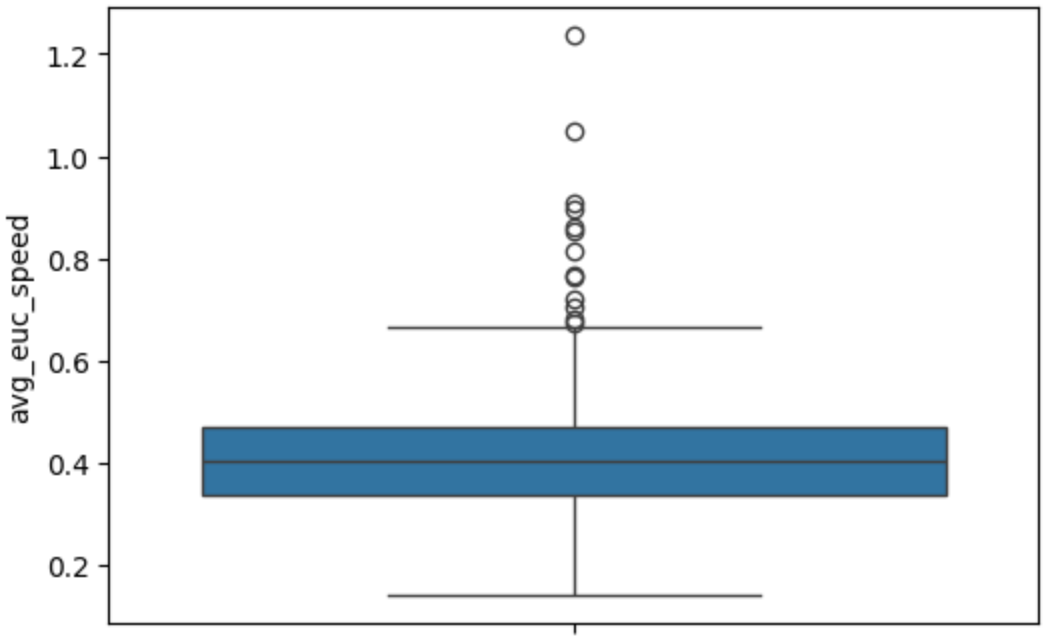


In [56]:
```
Q1 = dataset['avg_euc_dist'].quantile(0.25)
Q3 = dataset['avg_euc_dist'].quantile(0.75)
IQR = Q3-Q1
IQR
upper_limit = Q3 + (1.5*IQR)
lower_limit = Q1 - (1.5*IQR)
print(upper_limit)
print(lower_limit)
```

```
8271.182572932525
1761.1652065041244
```

In [ ]:
```
#Outliers present in avg_euc_dist
```

In [57]:
```
plt.figure(figsize=(6, 4))
sns.boxplot(dataset['avg_euc_speed'])
plt.show()
```

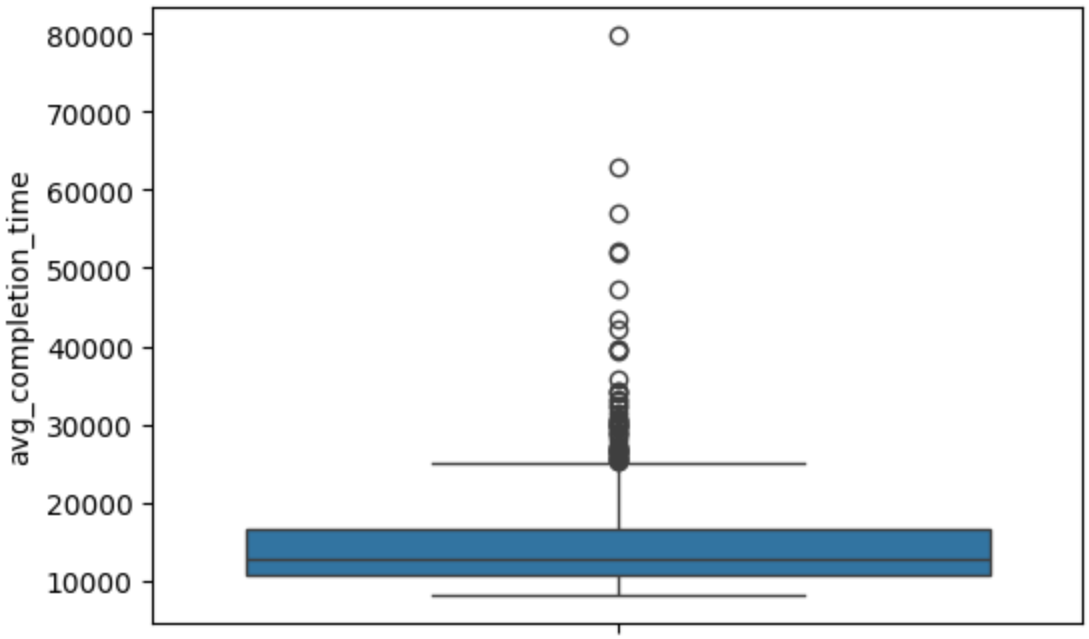


In [58]:
```
Q1 = dataset['avg_euc_speed'].quantile(0.25)
Q3 = dataset['avg_euc_speed'].quantile(0.75)
IQR = Q3-Q1
IQR
upper_limit = Q3 + (1.5*IQR)
lower_limit = Q1 - (1.5*IQR)
print(upper_limit)
print(lower_limit)
```

```
0.6707795711820126
0.13450174685352434
```

In [ ]:
```
#Outliers present in avg_euc_speed
```

In [59]:
```
plt.figure(figsize=(6, 4))
sns.boxplot(dataset['avg_completion_time'])
plt.show()
```

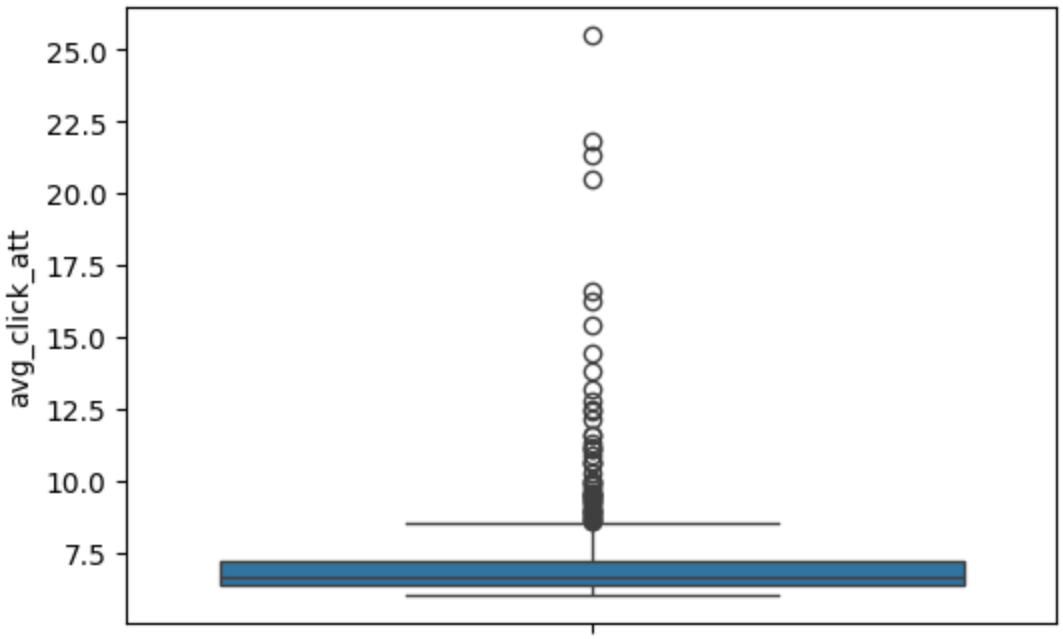


In [60]:
```
Q1 = dataset['avg_completion_time'].quantile(0.25)
Q3 = dataset['avg_completion_time'].quantile(0.75)
IQR = Q3-Q1
IQR
upper_limit = Q3 + (1.5*IQR)
lower_limit = Q1 - (1.5*IQR)
print(upper_limit)
print(lower_limit)
```

```
25151.811463693542
2137.4966171145406
```

In [ ]:
```
#Outliers present in avg_completion_time
```

In [61]:
```
plt.figure(figsize=(6, 4))
sns.boxplot(dataset['avg_click_att'])
plt.show()
```

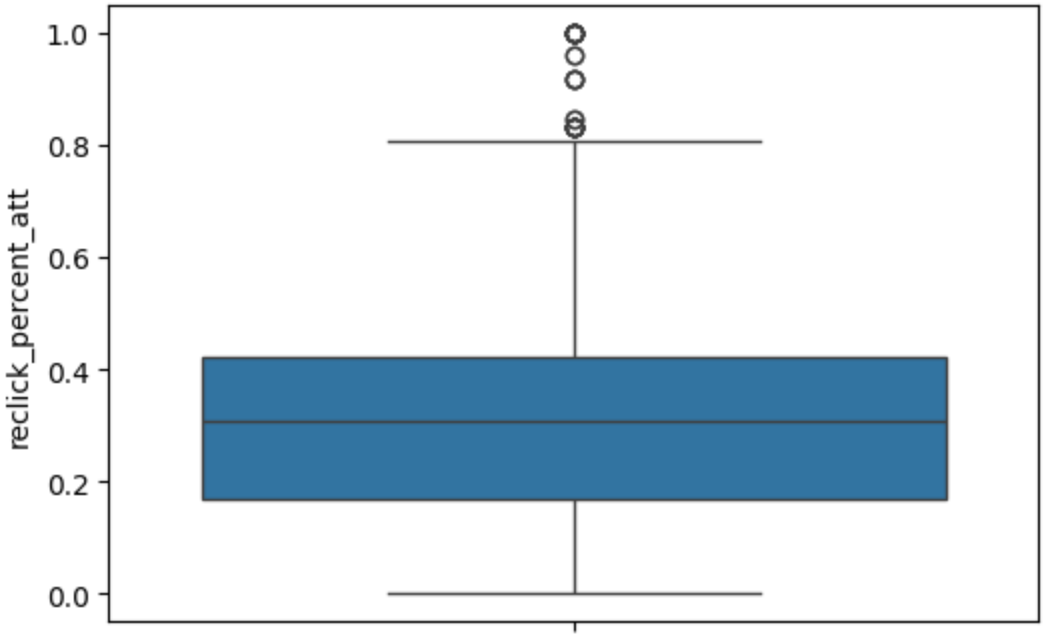


In [62]:
```
Q1 = dataset['avg_click_att'].quantile(0.25)
Q3 = dataset['avg_click_att'].quantile(0.75)
IQR = Q3-Q1
IQR
upper_limit = Q3 + (1.5*IQR)
lower_limit = Q1 - (1.5*IQR)
print(upper_limit)
print(lower_limit)
```

```
8.528846153846159
5.016025641025637
```

In [ ]:
```
#Outliers present in avg_click_att
```

In [63]:
```
plt.figure(figsize=(6, 4))
sns.boxplot(dataset['reclick_percent_att'])
plt.show()
```

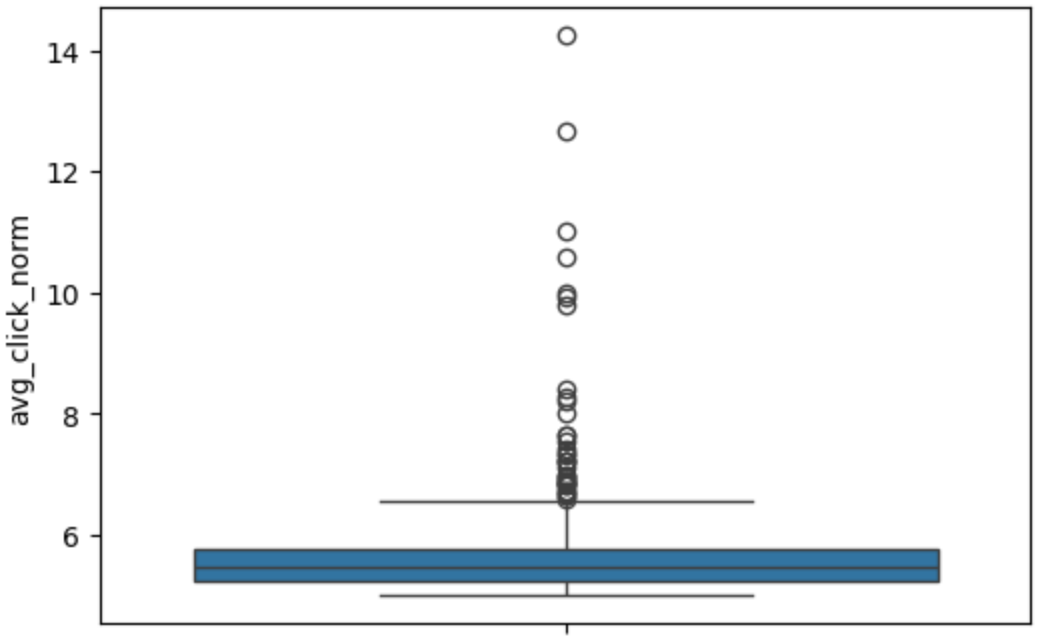


In [64]:
```
Q1 = dataset['reclick_percent_att'].quantile(0.25)
Q3 = dataset['reclick_percent_att'].quantile(0.75)
IQR = Q3-Q1
IQR
upper_limit = Q3 + (1.5*IQR)
lower_limit = Q1 - (1.5*IQR)
print(upper_limit)
print(lower_limit)
```

```
0.8076923076923078
-0.21794871794871815
```

In [ ]:
```
#Outliers present in reclick_percent_att
```

In [65]:
```
plt.figure(figsize=(6, 4))
sns.boxplot(dataset['avg_click_norm'])
plt.show()
```

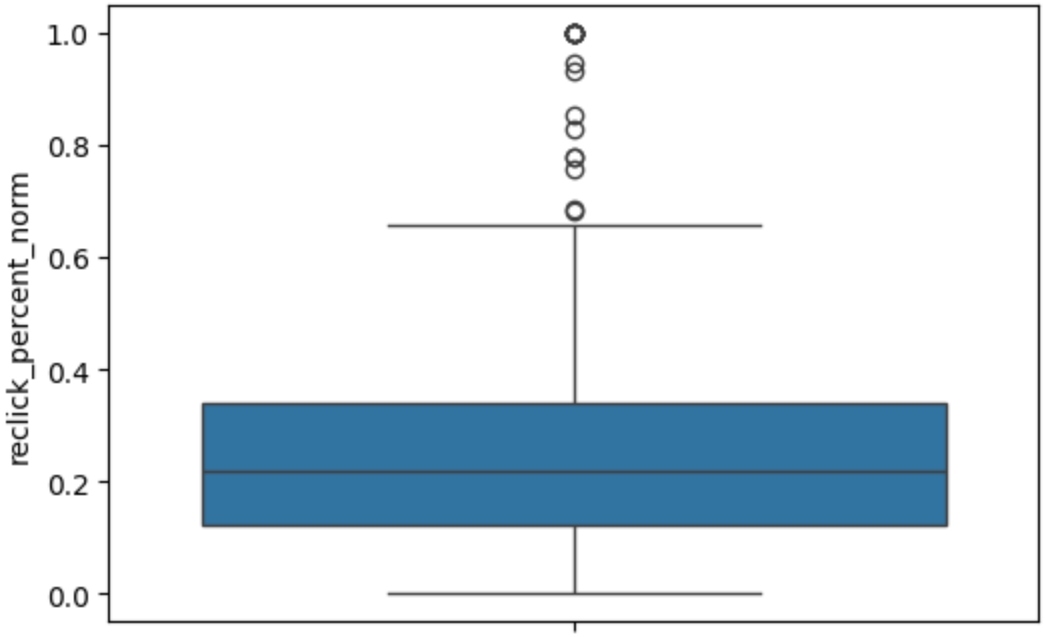


In [66]:
```
Q1 = dataset['avg_click_norm'].quantile(0.25)
Q3 = dataset['avg_click_norm'].quantile(0.75)
IQR = Q3-Q1
IQR
upper_limit = Q3 + (1.5*IQR)
lower_limit = Q1 - (1.5*IQR)
print(upper_limit)
print(lower_limit)
```

```
6.568660207150016
4.449047778149017
```

In [ ]:
```
#Outliers present in avg_click_norm
```

In [67]:
```
plt.figure(figsize=(6, 4))
sns.boxplot(dataset['reclick_percent_norm'])
plt.show()
```



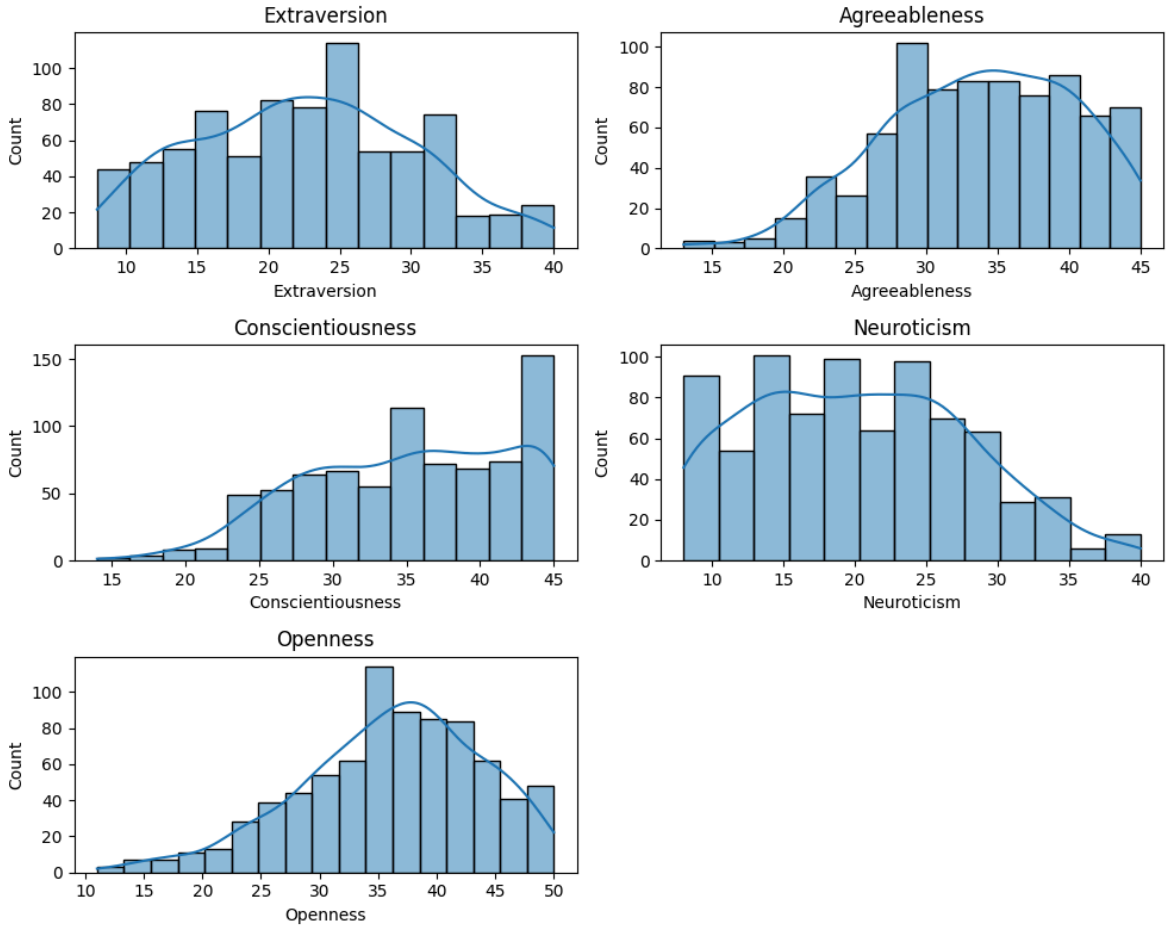
In [68]:
```
Q1 = dataset['reclick_percent_norm'].quantile(0.25)
Q3 = dataset['reclick_percent_norm'].quantile(0.75)
IQR = Q3-Q1
IQR
upper_limit = Q3 + (1.5*IQR)
lower_limit = Q1 - (1.5*IQR)
print(upper_limit)
print(lower_limit)
```

```
0.6707317073170733
-0.20731707317073184
```

In [ ]:
```
#Outliers present in reclick_percent_norm
```

Normal Distribution graph for Big Five Personality Traits

In [69]:
```
plt.figure(figsize=(10, 8))

# Define the layout of subplots
rows = 3
cols = 2

# List of variable names
variables = ['Extraversion', 'Agreeableness','Conscientiousness','Neuroticism
for i, var in enumerate(variables, 1):
    plt.subplot(rows, cols, i)
    sns.histplot(dataset[var], kde=True)
    plt.title(var)

plt.tight_layout()
plt.show()
```



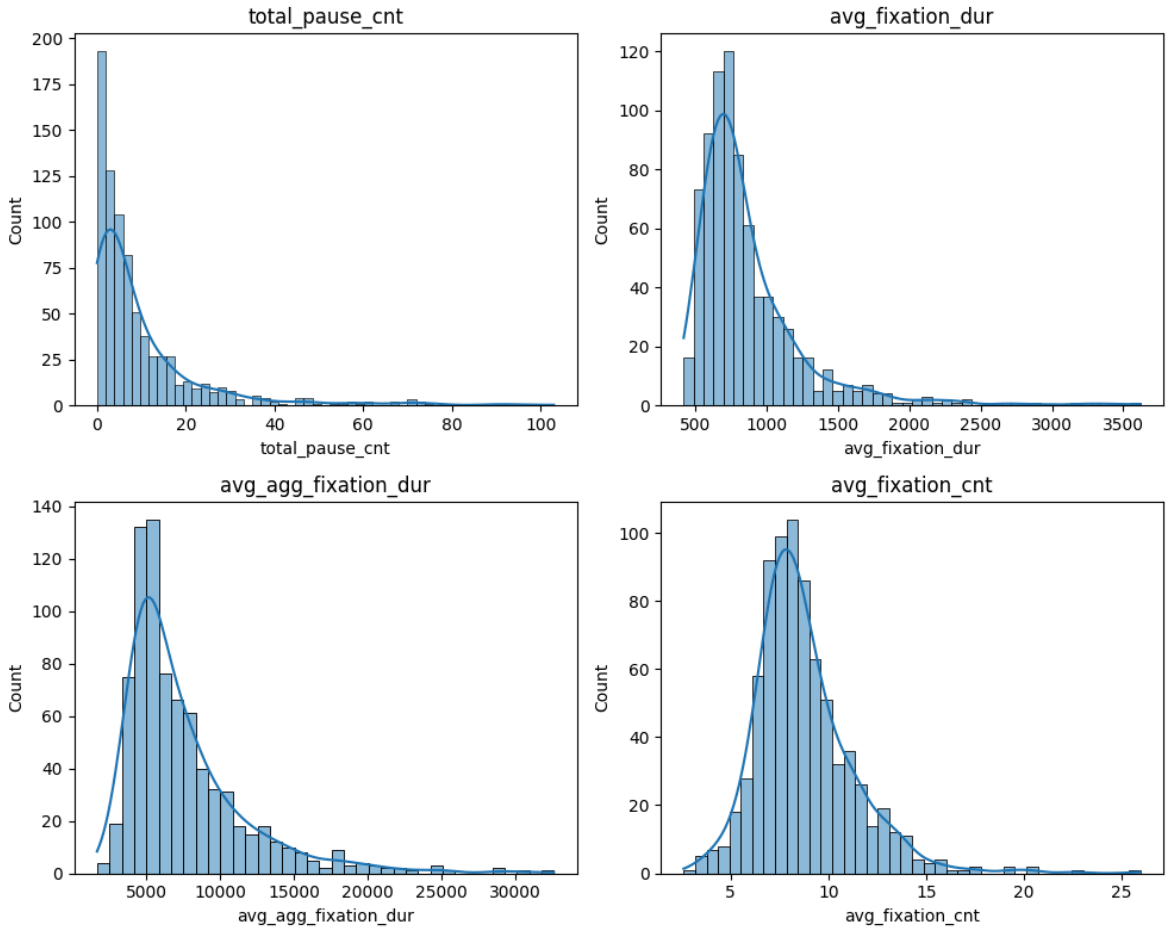
Symmetrical or approximately Normal Distribution graph was observed for Extraversion. Agreeableness, Consientiousness and Openness are left skewed & Neuroticism is right skewed

In [29]:
```
'avg_click_att','reclick_percent_att','avg_click_norm','reclick_percent_norm'
'avg_euc_speed','avg_completion_time','total_pause_cnt','avg_fixation_dur','av
'avg_fixation_cnt'
```

Out[29]:
```
'avg_fixation_cnt'
```

Normal Distribution graph for Time related Mouse features

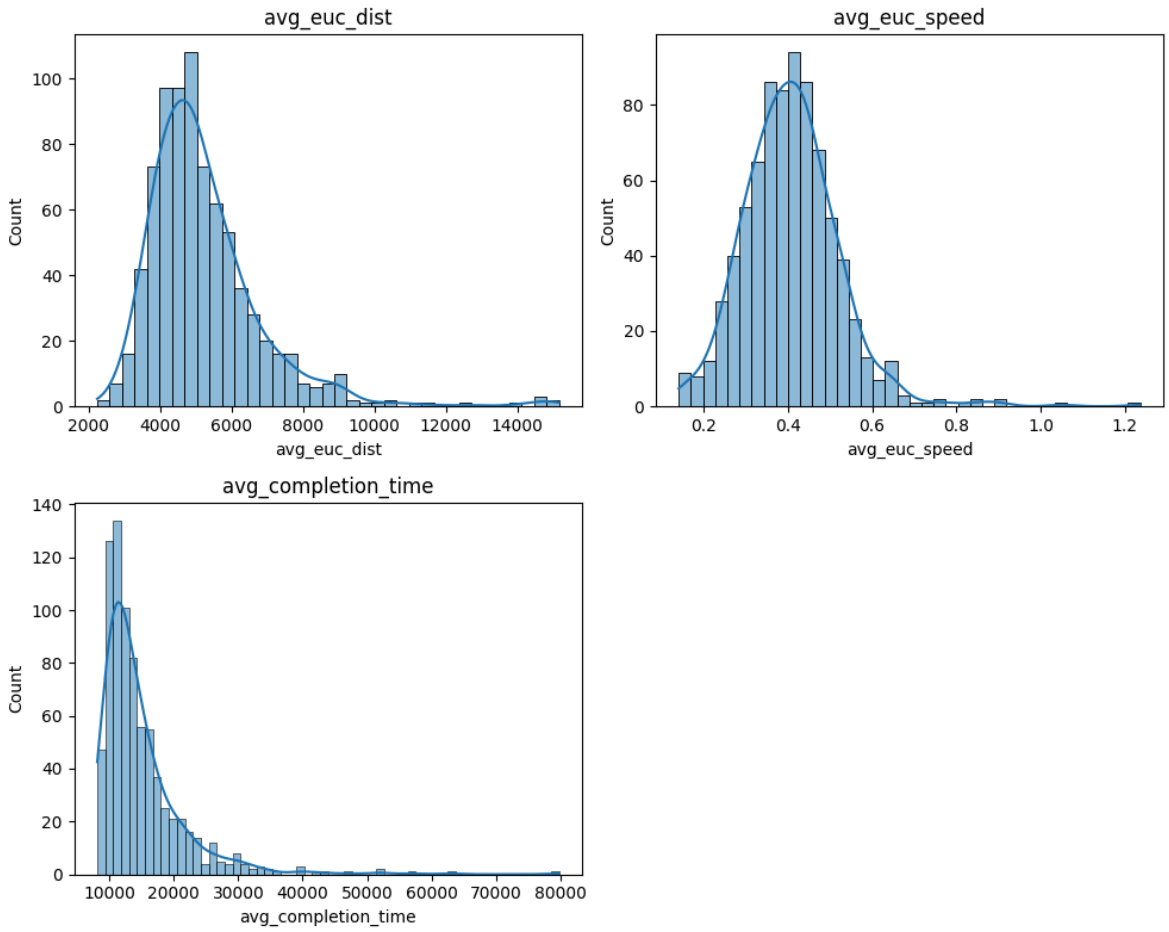
In [70]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Set the size of each subplot
plt.figure(figsize=(10, 8))

# Define the layout of subplots
rows = 2
cols = 2

# List of variable names
variables = ['total_pause_cnt','avg_fixation_dur','avg_agg_fixation_dur','avg_
for i, var in enumerate(variables, 1):
    plt.subplot(rows, cols, i)
    sns.histplot(dataset[var], kde=True)
    plt.title(var)

plt.tight_layout()
plt.show()
```



'total_pause_cnt','avg_fixation_dur','avg_agg_fixation_dur','avg_fixation_cnt'-right skewed , most of the data points are located on the left side

Normal Distribution graph for Activity-related Mouse Features

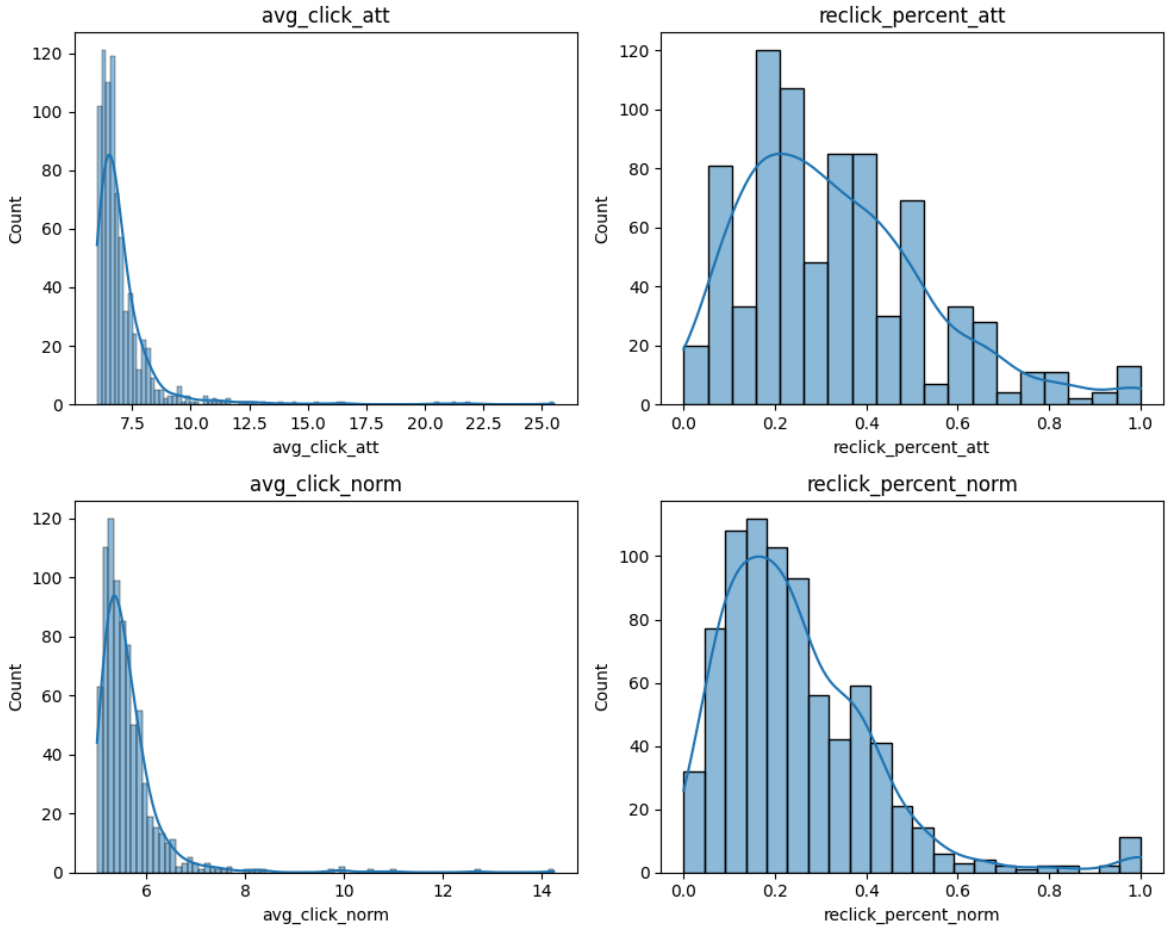
In [71]:
```
plt.figure(figsize=(10, 8))

# Define the layout of subplots
rows = 2
cols = 2

# List of variable names
variables = ['avg_euc_dist','avg_euc_speed','avg_completion_time']
for i, var in enumerate(variables, 1):
    plt.subplot(rows, cols, i)
    sns.histplot(dataset[var], kde=True)
    plt.title(var)

plt.tight_layout()
plt.show()
```



'avg_euc_dist','avg_euc_speed','avg_completion_time'-right skewed , most of the data points are located on the left side.

Normal Distribution graph for Click-related Mouse features

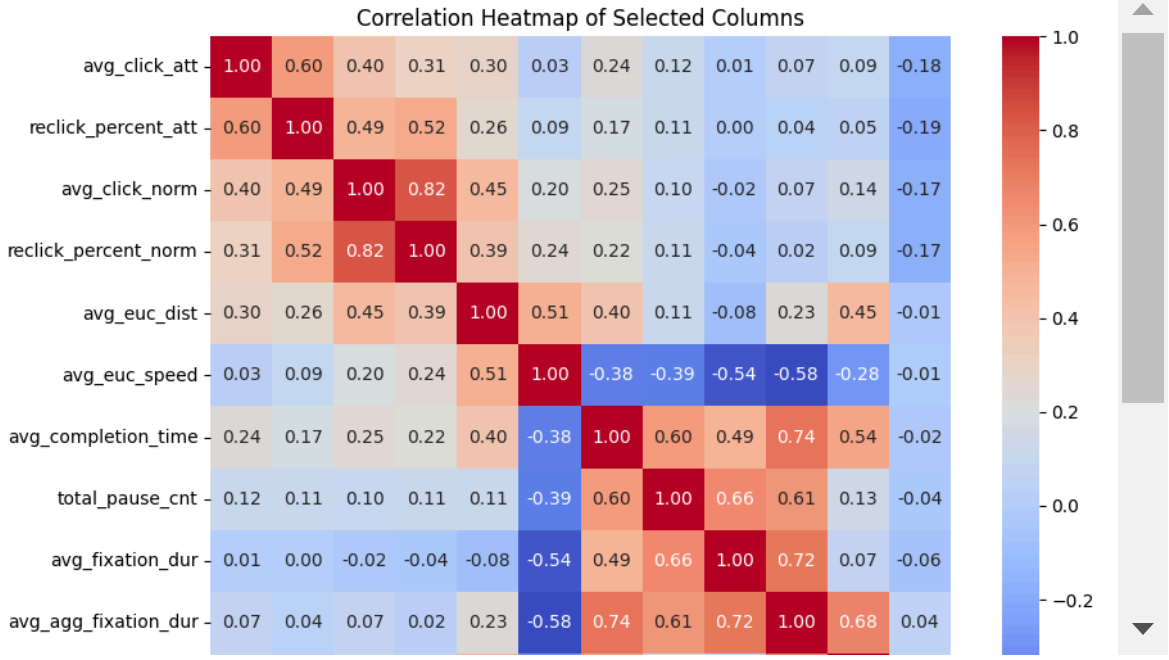
In [72]:
```
plt.figure(figsize=(10, 8))

# Define the layout of subplots
rows = 2
cols = 2

# List of variable names
variables = ['avg_click_att','reclick_percent_att','avg_click_norm','reclick_
for i, var in enumerate(variables, 1):
    plt.subplot(rows, cols, i)
    sns.histplot(dataset[var], kde=True)
    plt.title(var)

plt.tight_layout()
plt.show()
```



'avg_click_att','reclick_percent_att','avg_click_norm','reclick_percent_norm'-right skewed , most of the data points are located on the left side.

Heatmap for 11 Mouse features & Abs_Area_Under_Curve

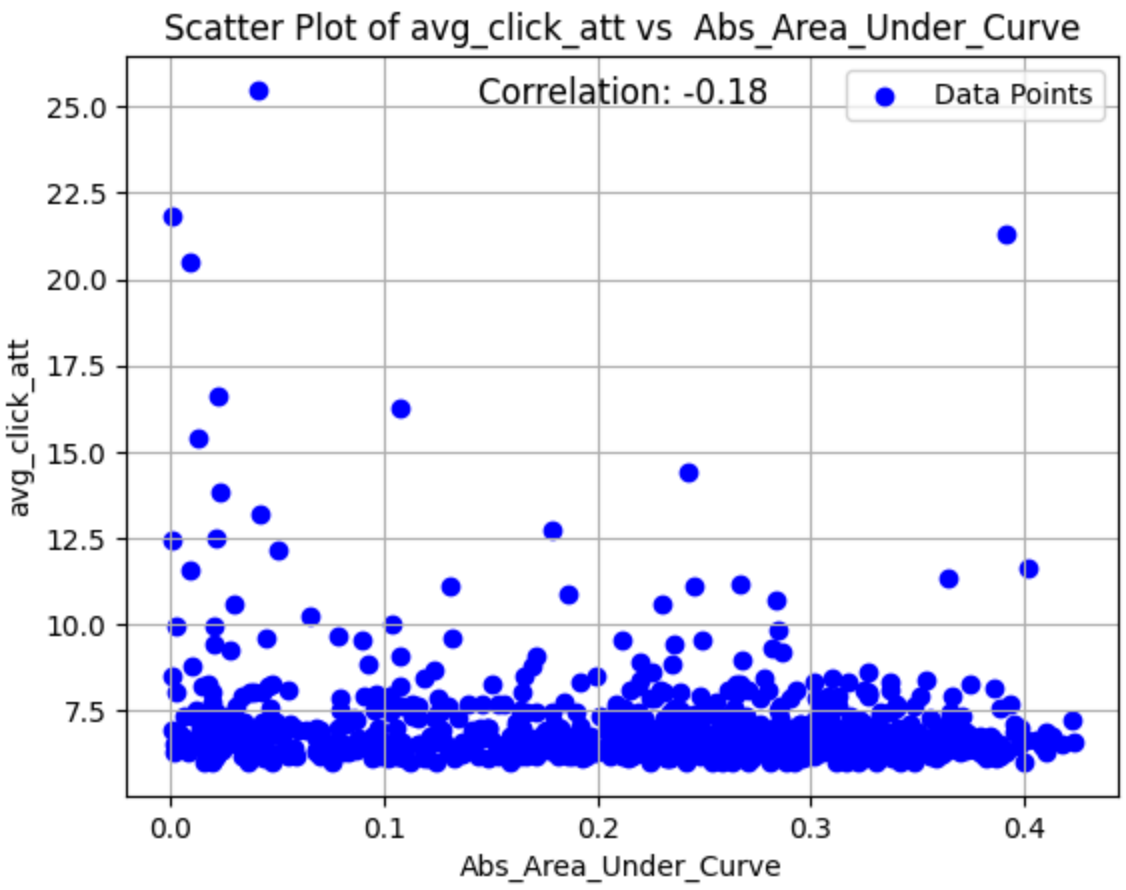
In [73]:
```
import seaborn as sns
import matplotlib.pyplot as plt

selected_columns = ['avg_click_att','reclick_percent_att','avg_click_norm','re
'avg_euc_speed','avg_completion_time','total_pause_cnt','avg_fixation_dur','av
'avg_fixation_cnt','Abs_Area_Under_Curve']
selected_data = dataset[selected_columns]
correlation_matrix = selected_data.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", square
plt.title('Correlation Heatmap of Selected Columns')
plt.yticks(rotation=0)
plt.xticks(rotation=90)
plt.tight_layout()

# Show plot
plt.show()
```



Scatter Plot b/w Abs_Area_Under_Curve and 11 Mouse features

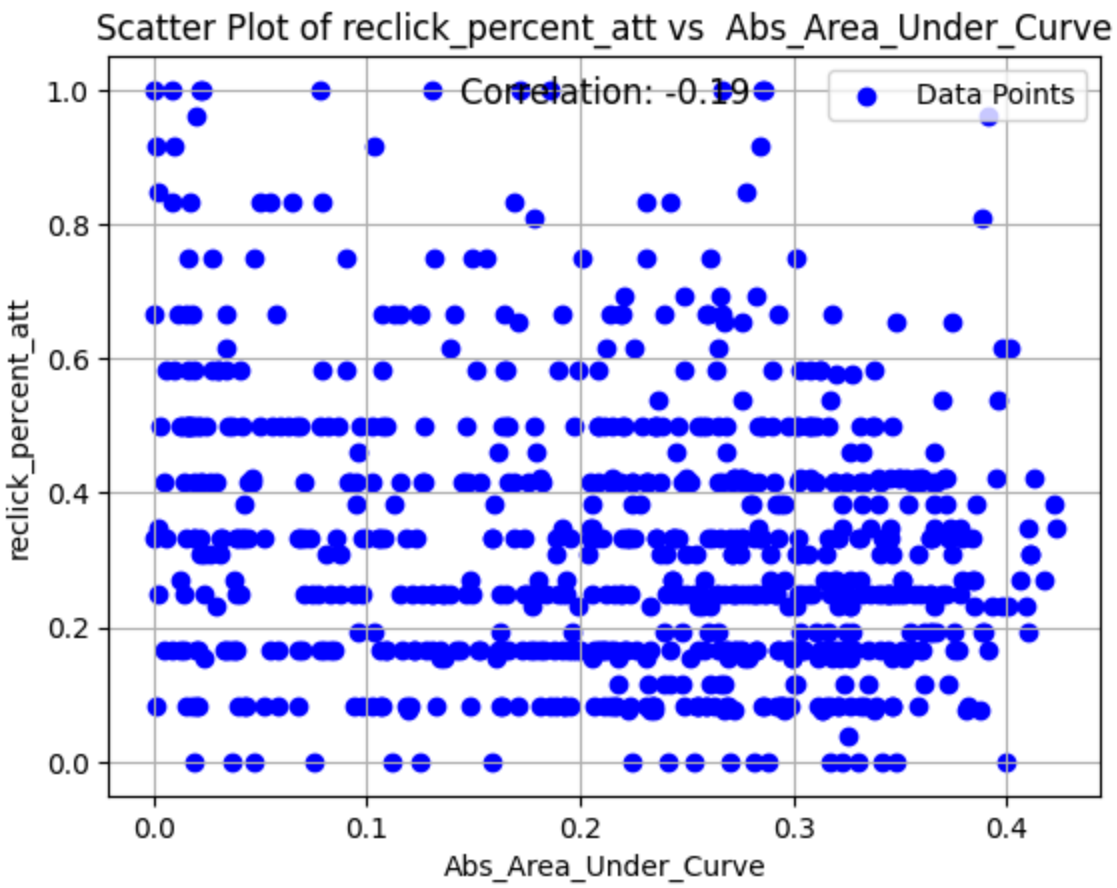
In [15]:
```
x_values = click_data['Abs_Area_Under_Curve']
y_values = click_data['avg_click_att']
```

In [17]:
```
import numpy as np
import matplotlib.pyplot as plt

# Calculate the correlation coefficient
correlation = np.corrcoef(x_values, y_values)[0, 1]

# Plot the scatter plot
plt.scatter(x_values, y_values, color='blue', label='Data Points')

# Add the correlation coefficient as text to the plot
plt.text(0.5, 0.95, f'Correlation: {correlation:.2f}', horizontalalignment='c

# Add labels and title
plt.xlabel('Abs_Area_Under_Curve')
plt.ylabel('avg_click_att')
plt.title('Scatter Plot of avg_click_att vs  Abs_Area_Under_Curve')

# Add a legend
plt.legend()

# Show the plot
plt.grid(True)
plt.show()
```



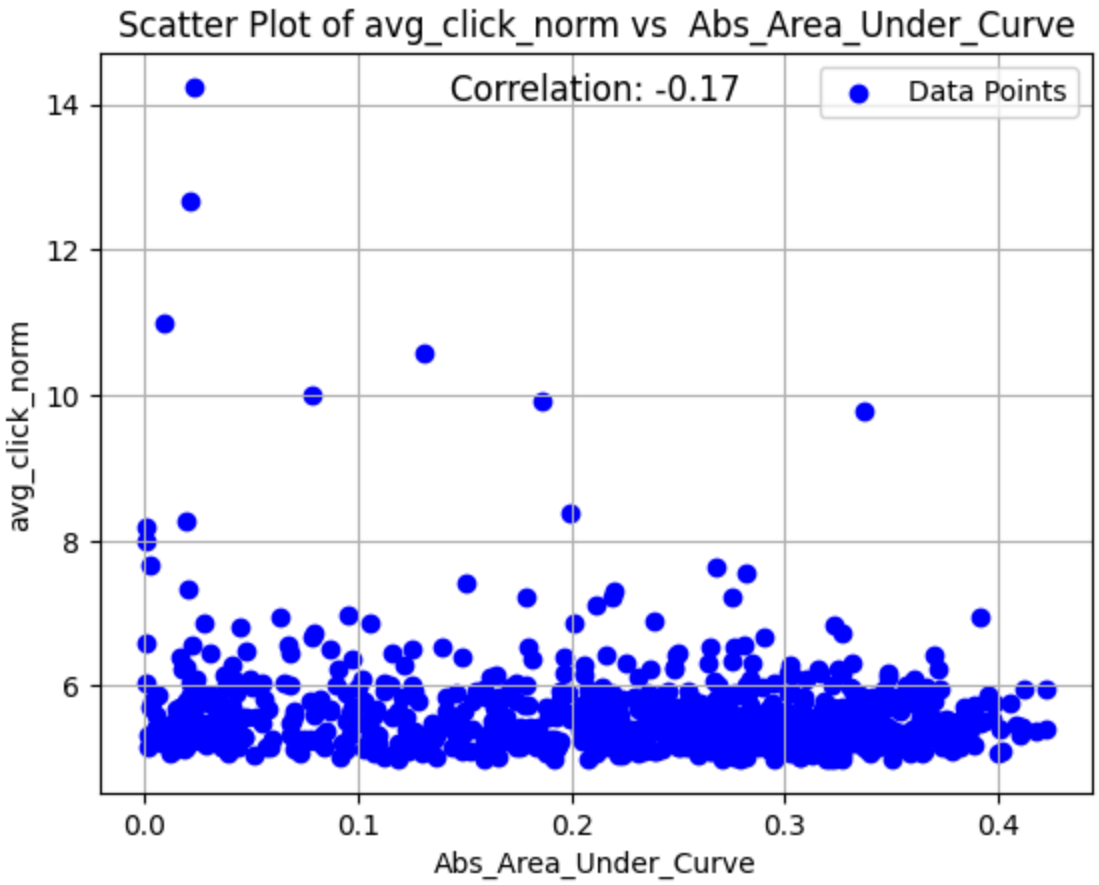
In [18]:
```
x_values = click_data['Abs_Area_Under_Curve']
y_values = click_data['reclick_percent_att']
```

In [19]:
```
import numpy as np
import matplotlib.pyplot as plt

# Calculate the correlation coefficient
correlation = np.corrcoef(x_values, y_values)[0, 1]

# Plot the scatter plot
plt.scatter(x_values, y_values, color='blue', label='Data Points')

# Add the correlation coefficient as text to the plot
plt.text(0.5, 0.95, f'Correlation: {correlation:.2f}', horizontalalignment='c

# Add labels and title
plt.xlabel('Abs_Area_Under_Curve')
plt.ylabel('reclick_percent_att')
plt.title('Scatter Plot of reclick_percent_att vs  Abs_Area_Under_Curve')

# Add a legend
plt.legend()

# Show the plot
plt.grid(True)
plt.show()
```



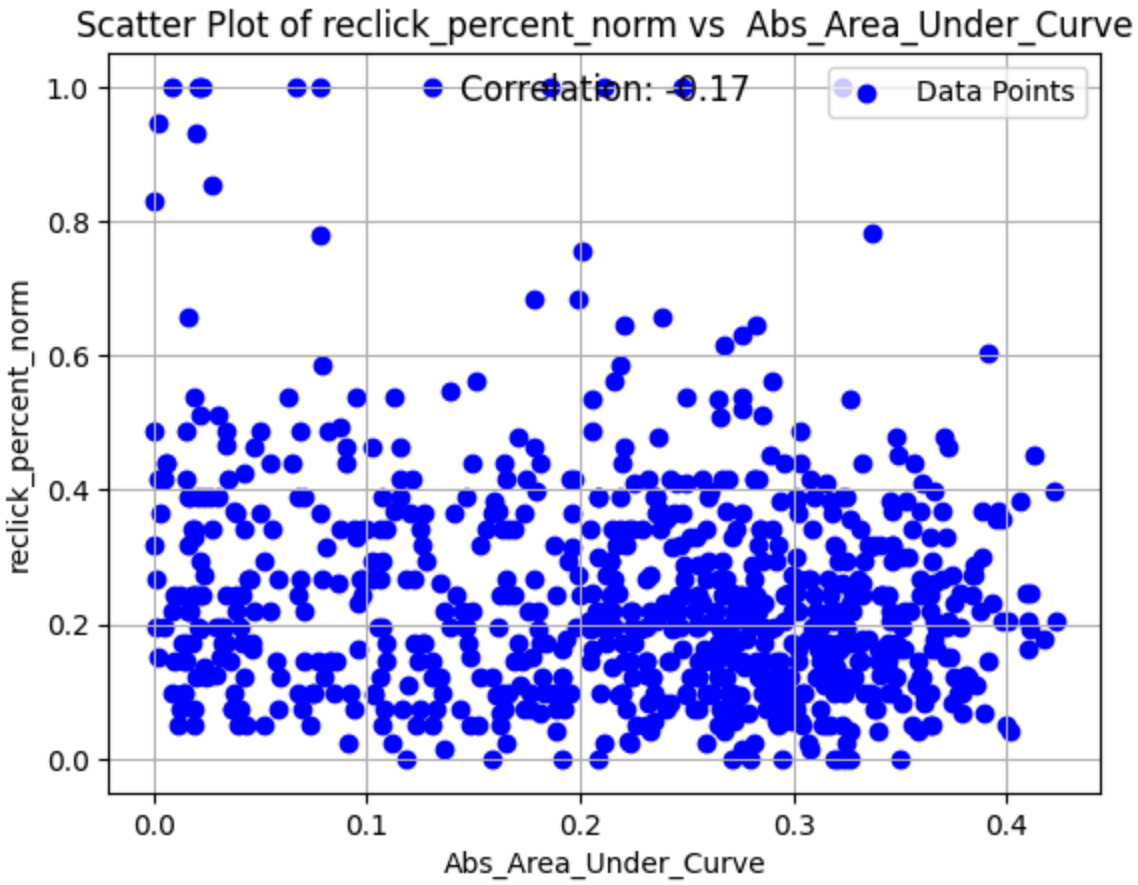
In [20]:
```
x_values = click_data['Abs_Area_Under_Curve']
y_values = click_data['avg_click_norm']
```

In [21]:
```
import numpy as np
import matplotlib.pyplot as plt

# Calculate the correlation coefficient
correlation = np.corrcoef(x_values, y_values)[0, 1]

# Plot the scatter plot
plt.scatter(x_values, y_values, color='blue', label='Data Points')

# Add the correlation coefficient as text to the plot
plt.text(0.5, 0.95, f'Correlation: {correlation:.2f}', horizontalalignment='c

# Add labels and title
plt.xlabel('Abs_Area_Under_Curve')
plt.ylabel('avg_click_norm')
plt.title('Scatter Plot of avg_click_norm vs  Abs_Area_Under_Curve')

# Add a legend
plt.legend()

# Show the plot
plt.grid(True)
plt.show()
```



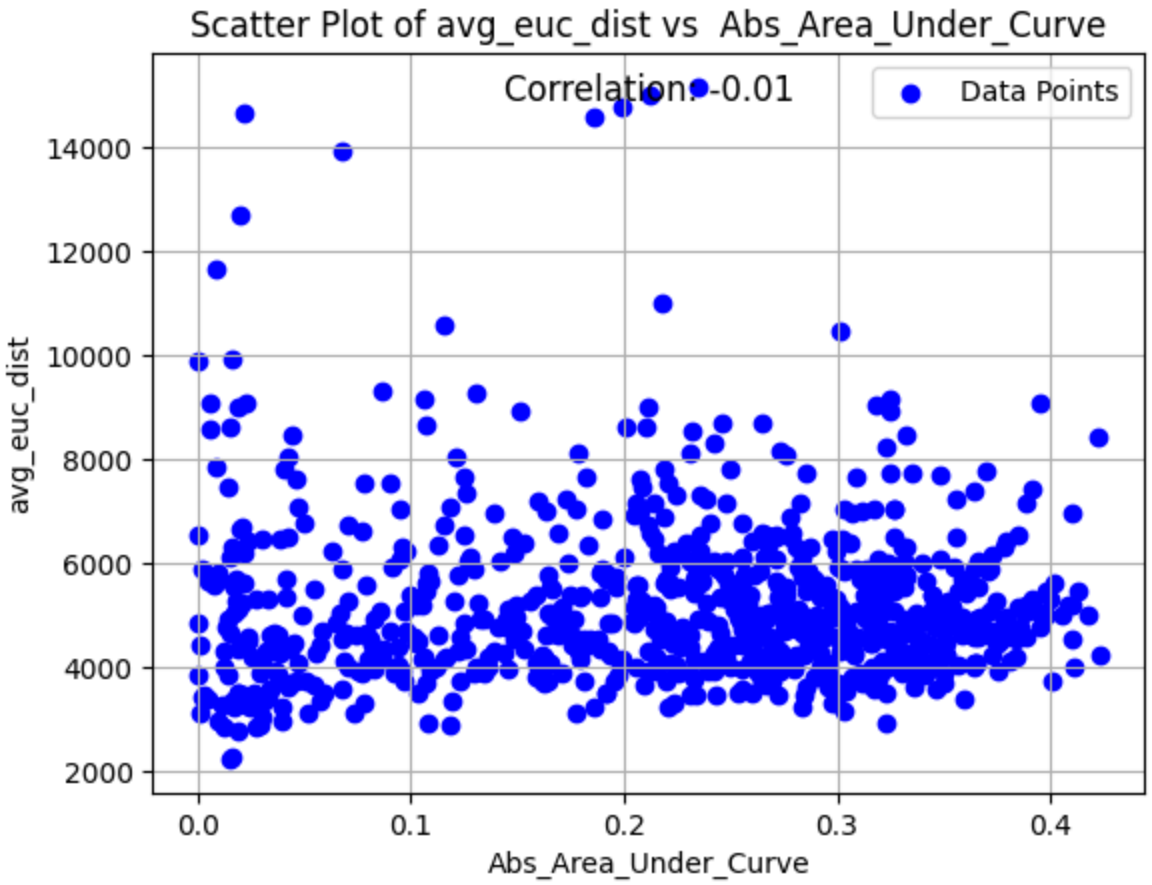
In [22]:
```
x_values = click_data['Abs_Area_Under_Curve']
y_values = click_data['reclick_percent_norm']
```

In [23]:
```
import numpy as np
import matplotlib.pyplot as plt

# Calculate the correlation coefficient
correlation = np.corrcoef(x_values, y_values)[0, 1]

# Plot the scatter plot
plt.scatter(x_values, y_values, color='blue', label='Data Points')

# Add the correlation coefficient as text to the plot
plt.text(0.5, 0.95, f'Correlation: {correlation:.2f}', horizontalalignment='c

# Add labels and title
plt.xlabel('Abs_Area_Under_Curve')
plt.ylabel('reclick_percent_norm')
plt.title('Scatter Plot of reclick_percent_norm vs  Abs_Area_Under_Curve')

# Add a legend
plt.legend()

# Show the plot
plt.grid(True)
plt.show()
```



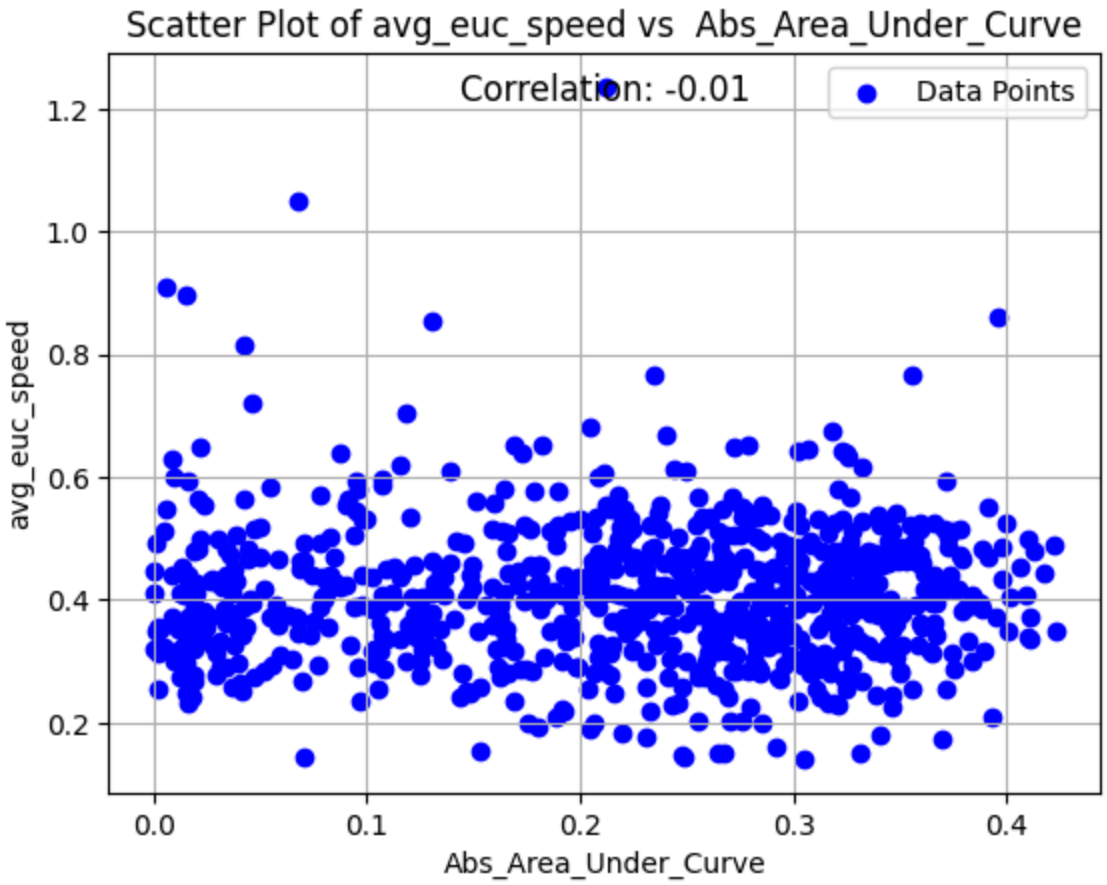
In [24]:
```
x_values = click_data['Abs_Area_Under_Curve']
y_values = click_data['avg_euc_dist']
```

In [25]:
```
import numpy as np
import matplotlib.pyplot as plt

# Calculate the correlation coefficient
correlation = np.corrcoef(x_values, y_values)[0, 1]

# Plot the scatter plot
plt.scatter(x_values, y_values, color='blue', label='Data Points')

# Add the correlation coefficient as text to the plot
plt.text(0.5, 0.95, f'Correlation: {correlation:.2f}', horizontalalignment='c

# Add labels and title
plt.xlabel('Abs_Area_Under_Curve')
plt.ylabel('avg_euc_dist')
plt.title('Scatter Plot of avg_euc_dist vs  Abs_Area_Under_Curve')

# Add a legend
plt.legend()

# Show the plot
plt.grid(True)
plt.show()
```



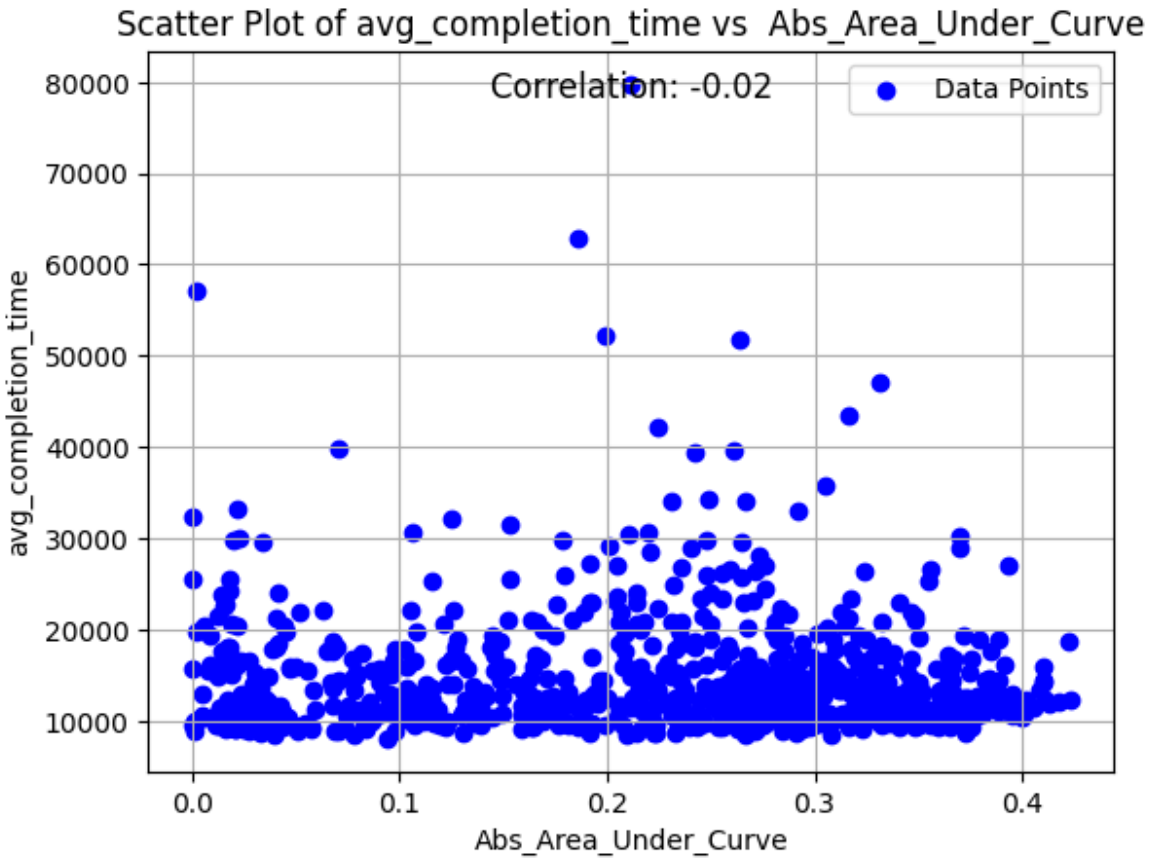
In [26]:
```
x_values = click_data['Abs_Area_Under_Curve']
y_values = click_data['avg_euc_speed']
```

In [27]:
```
import numpy as np
import matplotlib.pyplot as plt

# Calculate the correlation coefficient
correlation = np.corrcoef(x_values, y_values)[0, 1]

# Plot the scatter plot
plt.scatter(x_values, y_values, color='blue', label='Data Points')

# Add the correlation coefficient as text to the plot
plt.text(0.5, 0.95, f'Correlation: {correlation:.2f}', horizontalalignment='c

# Add labels and title
plt.xlabel('Abs_Area_Under_Curve')
plt.ylabel('avg_euc_speed')
plt.title('Scatter Plot of avg_euc_speed vs  Abs_Area_Under_Curve')

# Add a legend
plt.legend()

# Show the plot
plt.grid(True)
plt.show()
```



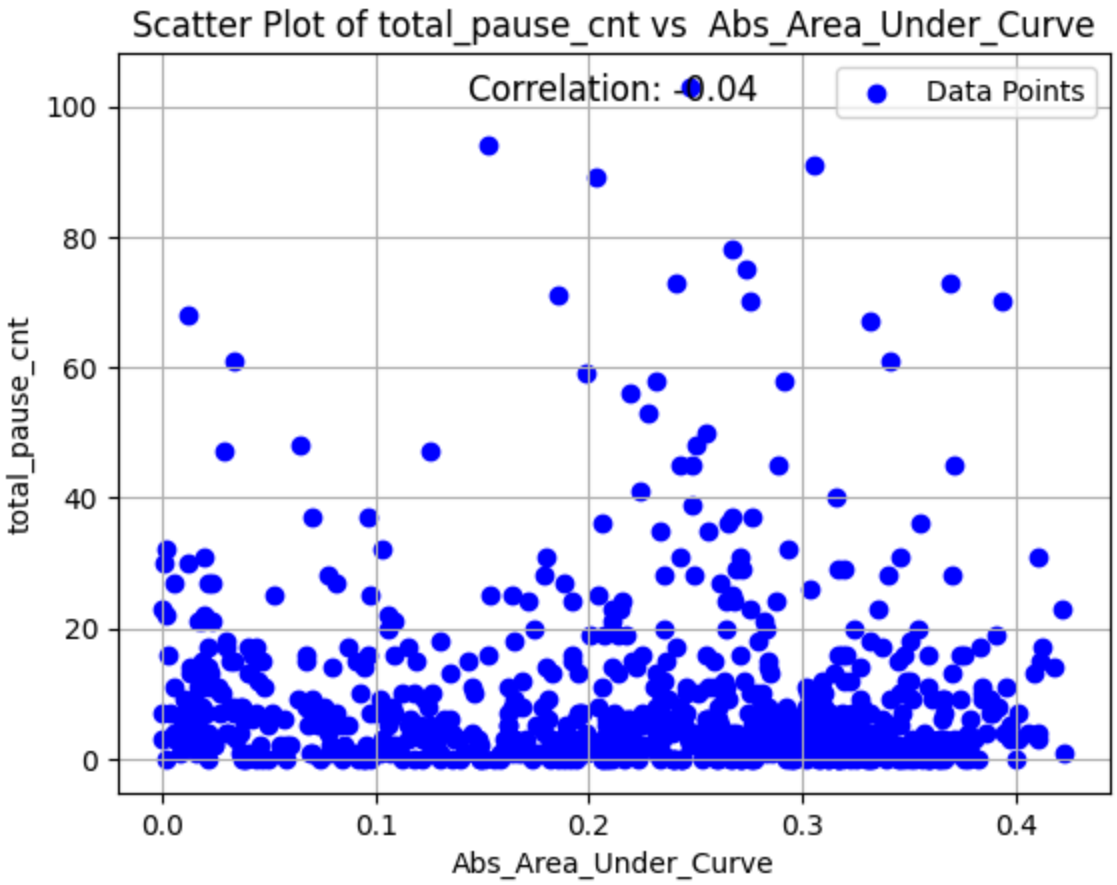
In [28]:
```
x_values = click_data['Abs_Area_Under_Curve']
y_values = click_data['avg_completion_time']
```

In [29]:
```
import numpy as np
import matplotlib.pyplot as plt

# Calculate the correlation coefficient
correlation = np.corrcoef(x_values, y_values)[0, 1]

# Plot the scatter plot
plt.scatter(x_values, y_values, color='blue', label='Data Points')

# Add the correlation coefficient as text to the plot
plt.text(0.5, 0.95, f'Correlation: {correlation:.2f}', horizontalalignment='c

# Add labels and title
plt.xlabel('Abs_Area_Under_Curve')
plt.ylabel('avg_completion_time')
plt.title('Scatter Plot of avg_completion_time vs  Abs_Area_Under_Curve')

# Add a legend
plt.legend()

# Show the plot
plt.grid(True)
plt.show()
```



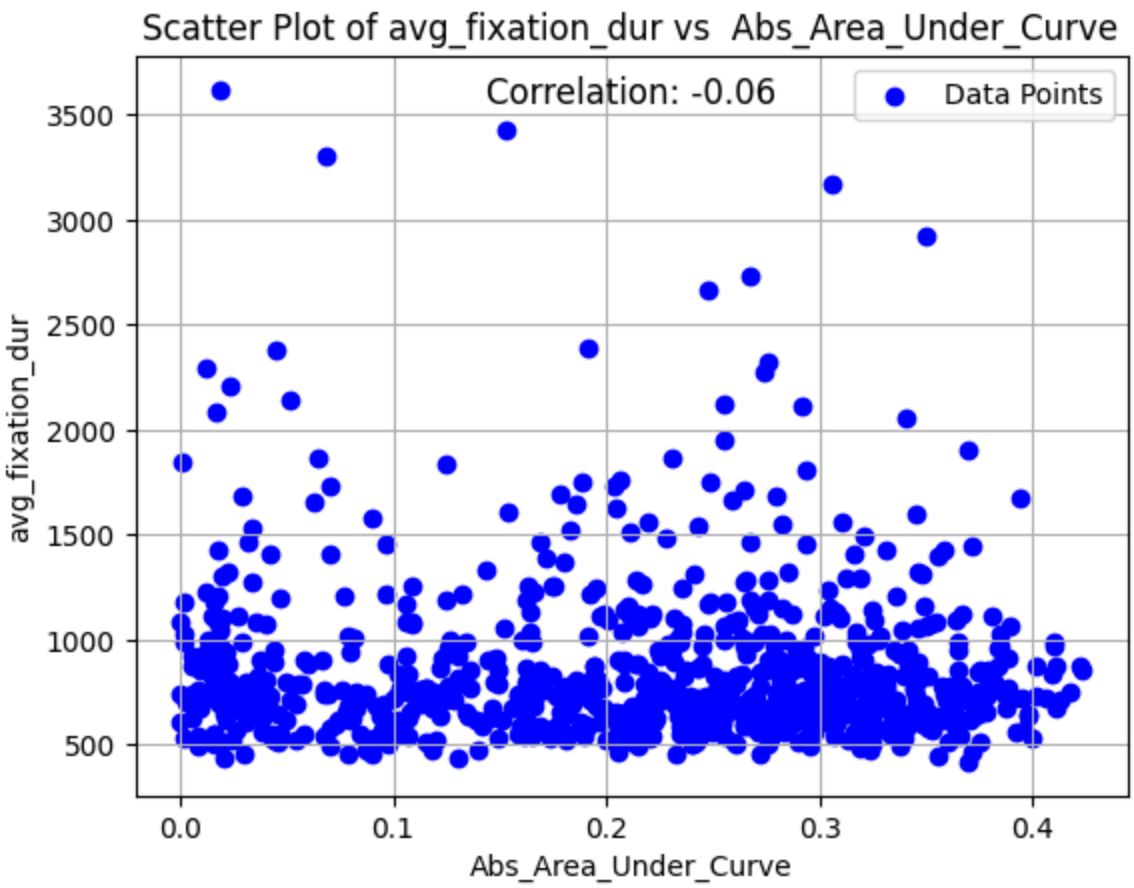
In [30]:
```
x_values = click_data['Abs_Area_Under_Curve']
y_values = click_data['total_pause_cnt']
```

In [31]:
```
import numpy as np
import matplotlib.pyplot as plt

# Calculate the correlation coefficient
correlation = np.corrcoef(x_values, y_values)[0, 1]

# Plot the scatter plot
plt.scatter(x_values, y_values, color='blue', label='Data Points')

# Add the correlation coefficient as text to the plot
plt.text(0.5, 0.95, f'Correlation: {correlation:.2f}', horizontalalignment='c

# Add labels and title
plt.xlabel('Abs_Area_Under_Curve')
plt.ylabel('total_pause_cnt')
plt.title('Scatter Plot of total_pause_cnt vs  Abs_Area_Under_Curve')

# Add a legend
plt.legend()

# Show the plot
plt.grid(True)
plt.show()
```



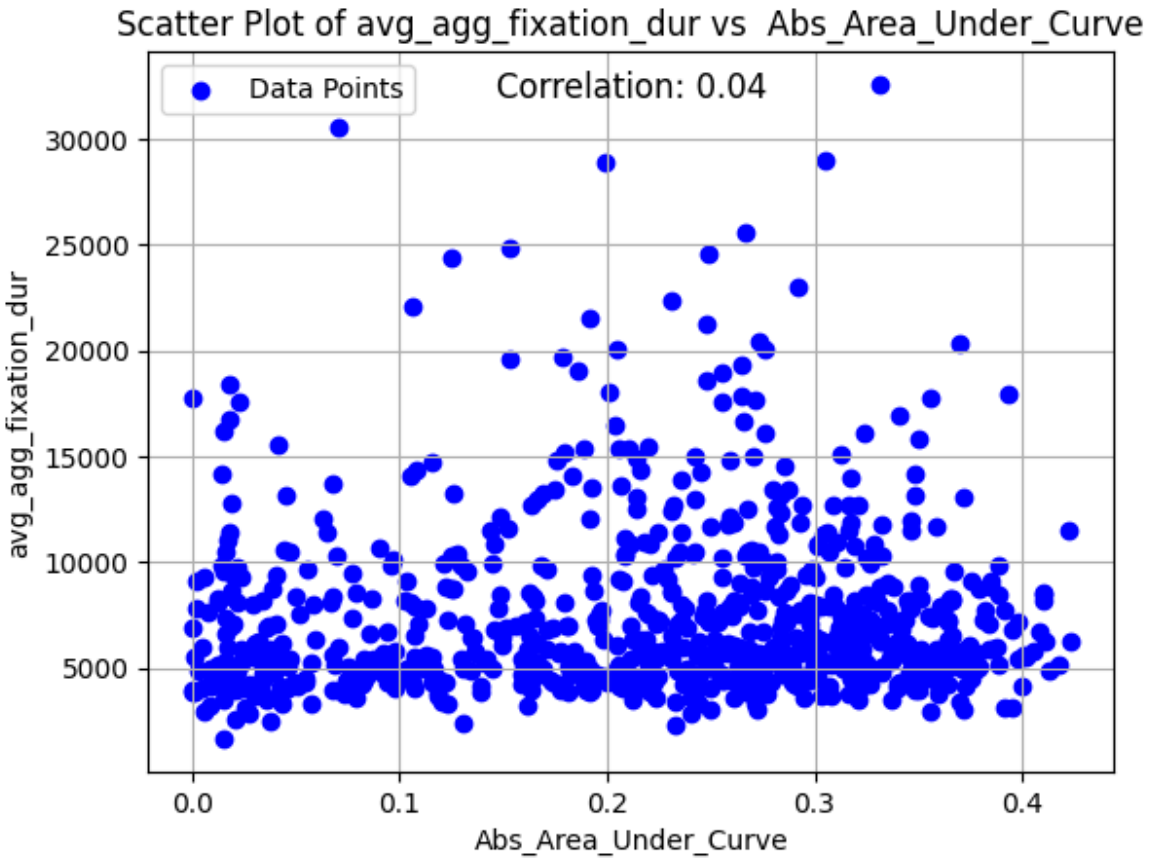
In [32]:
```
x_values = click_data['Abs_Area_Under_Curve']
y_values = click_data['avg_fixation_dur']
```

In [33]:
```
import numpy as np
import matplotlib.pyplot as plt

# Calculate the correlation coefficient
correlation = np.corrcoef(x_values, y_values)[0, 1]

# Plot the scatter plot
plt.scatter(x_values, y_values, color='blue', label='Data Points')

# Add the correlation coefficient as text to the plot
plt.text(0.5, 0.95, f'Correlation: {correlation:.2f}', horizontalalignment='c

# Add labels and title
plt.xlabel('Abs_Area_Under_Curve')
plt.ylabel('avg_fixation_dur')
plt.title('Scatter Plot of avg_fixation_dur vs  Abs_Area_Under_Curve')

# Add a legend
plt.legend()

# Show the plot
plt.grid(True)
plt.show()
```



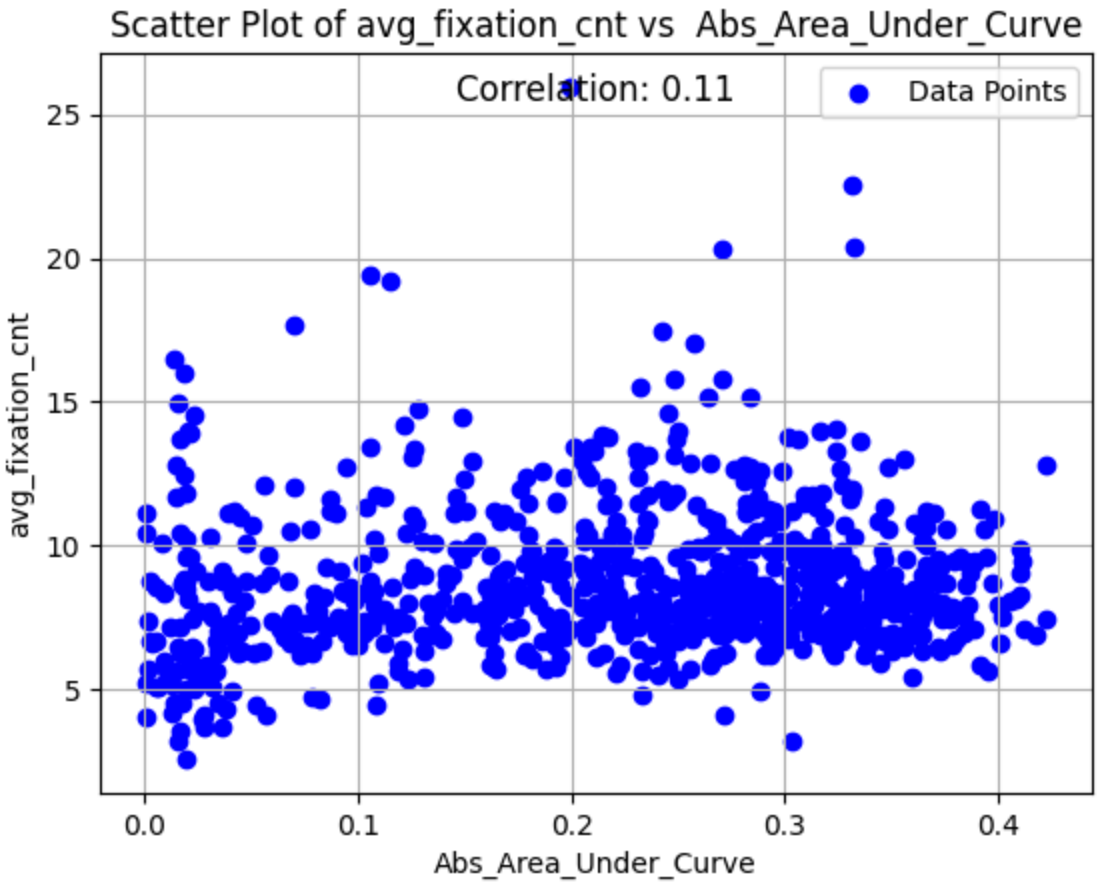
In [34]:
```
x_values = click_data['Abs_Area_Under_Curve']
y_values = click_data['avg_agg_fixation_dur']
```

In [35]:
```
import numpy as np
import matplotlib.pyplot as plt

# Calculate the correlation coefficient
correlation = np.corrcoef(x_values, y_values)[0, 1]

# Plot the scatter plot
plt.scatter(x_values, y_values, color='blue', label='Data Points')

# Add the correlation coefficient as text to the plot
plt.text(0.5, 0.95, f'Correlation: {correlation:.2f}', horizontalalignment='c

# Add labels and title
plt.xlabel('Abs_Area_Under_Curve')
plt.ylabel('avg_agg_fixation_dur')
plt.title('Scatter Plot of avg_agg_fixation_dur vs  Abs_Area_Under_Curve')

# Add a legend
plt.legend()

# Show the plot
plt.grid(True)
plt.show()
```



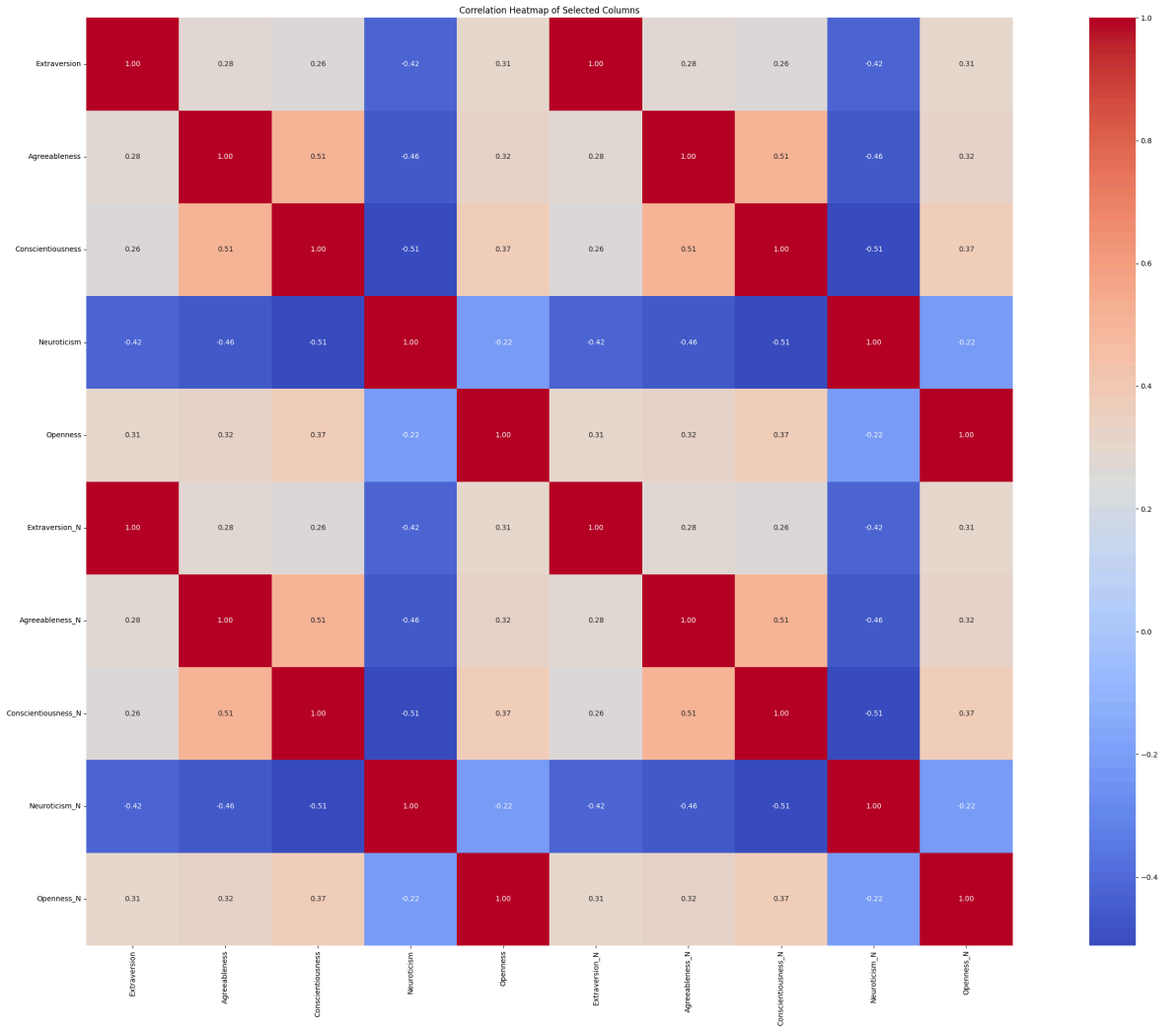
In [36]:
```
x_values = click_data['Abs_Area_Under_Curve']
y_values = click_data['avg_fixation_cnt']
```

In [37]:
```
import numpy as np
import matplotlib.pyplot as plt

# Calculate the correlation coefficient
correlation = np.corrcoef(x_values, y_values)[0, 1]

# Plot the scatter plot
plt.scatter(x_values, y_values, color='blue', label='Data Points')

# Add the correlation coefficient as text to the plot
plt.text(0.5, 0.95, f'Correlation: {correlation:.2f}', horizontalalignment='c

# Add labels and title
plt.xlabel('Abs_Area_Under_Curve')
plt.ylabel('avg_fixation_cnt')
plt.title('Scatter Plot of avg_fixation_cnt vs  Abs_Area_Under_Curve')

# Add a legend
plt.legend()

# Show the plot
plt.grid(True)
plt.show()
```



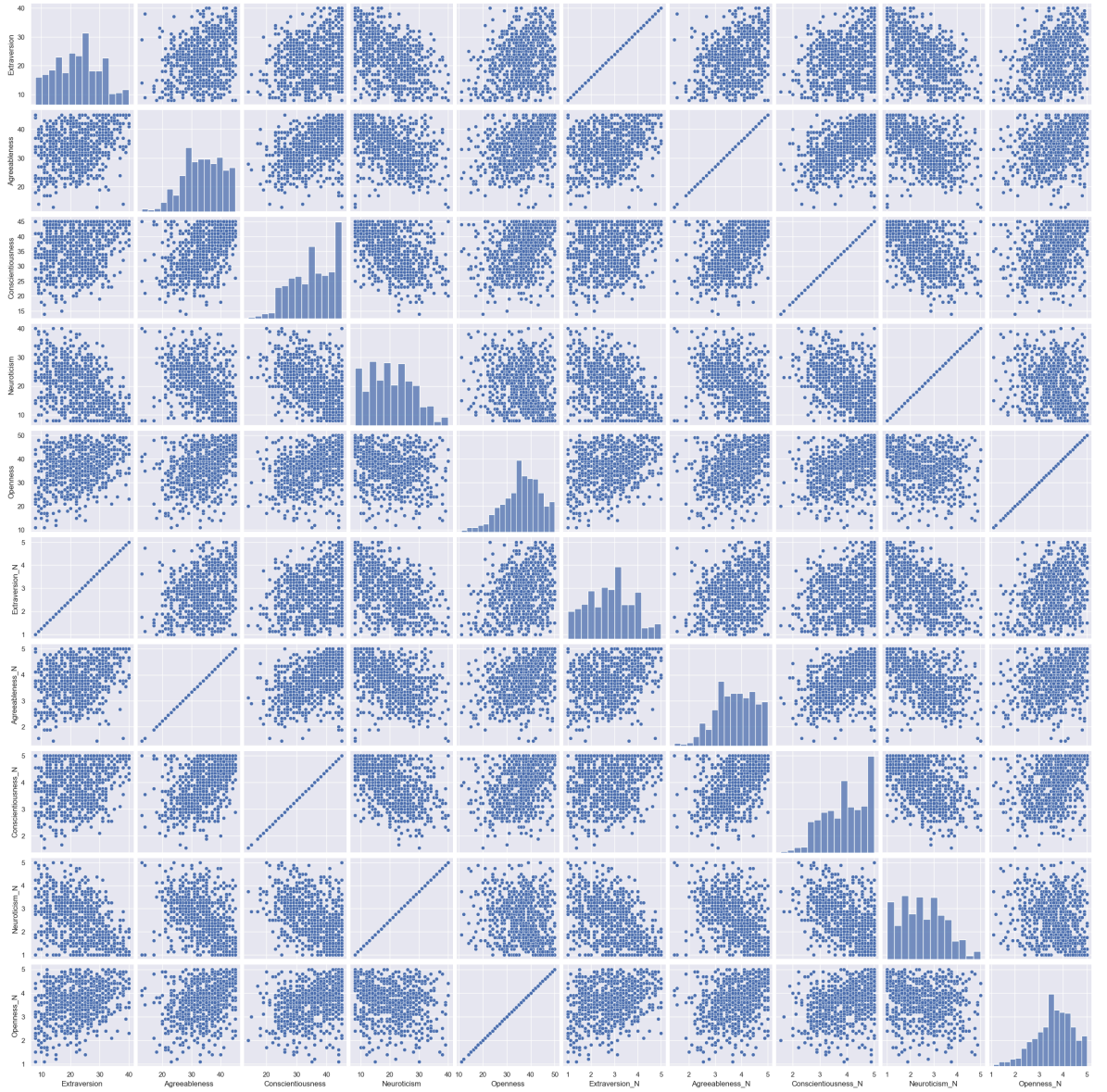
Correlation check b/w the columns to determine which should be actually taken

In [74]:
```
selected_columns = ['Extraversion','Agreeableness','Conscientiousness','Neurot
                    'Extraversion_N','Agreeableness_N','Conscientiousness_N',
                    'Openness_N']
selected_data = dataset[selected_columns]
correlation_matrix = selected_data.corr()
plt.figure(figsize=(30, 20))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", squar
plt.title('Correlation Heatmap of Selected Columns')
plt.yticks(rotation=0)
plt.xticks(rotation=90)
plt.tight_layout()

# Show plot
plt.show()

'''Perfect correlation b/w Extraversion & Extraversion_N ,Agreeableness & Agre
& Conscientiousness_N, Neuroticism & Neuroticism_N, Openness & Openness_N;

We can take the Normalized columns for analyses, the one ending with _N ,this
no particular variable is given more weightage'''
```



Out[74]:
```
'Perfect correlation b/w Extraversion & Extraversion_N ,Agreeableness & Agre
eableness_N,Conscientiousness\n& Conscientiousness_N, Neuroticism & Neurotic
ism_N, Openness & Openness_N;\n\nWe can take the Normalized columns for anal
yses, the one ending with _N ,this has been done so that\nno particular vari
able is given more weightage'
```

In [75]:
```
sns.set()
cols = ['Extraversion','Agreeableness','Conscientiousness','Neuroticism','Oper
        'Agreeableness_N','Conscientiousness_N','Neuroticism_N','Openness_N']
sns.pairplot(click_data[cols], size = 2.5)
plt.show();
```



Linear Regression Model predicting Extraversion from 11 Mouse related features

In [76]:
```
import statsmodels.api as sm
x_cols = ['avg_click_att', 'reclick_percent_att', 'avg_click_norm', 'reclick_
          'avg_euc_speed', 'avg_completion_time', 'total_pause_cnt', 'avg_fix
          'avg_fixation_cnt']

y_cols = ['Extraversion_N']

x = click_data[x_cols]
y = click_data[y_cols]
```

In [77]:
```
x.head()
```

Out[77]:

	avg_click_att	reclick_percent_att	avg_click_norm	reclick_percent_norm	avg_euc_dist	avg_e
0	6.307692	0.115385	5.520548	0.205479	6144.443128	
1	6.846154	0.384615	5.808219	0.342466	7322.577197	
2	6.346154	0.269231	5.493151	0.191781	3939.286111	
3	6.307692	0.230769	5.424658	0.273973	5434.640400	
4	6.692308	0.423077	5.205479	0.068493	3723.353789	

In [78]:
```
y.head()
```

Out[78]:

	Extraversion_N
0	2.500
1	3.875
2	1.625
3	4.125
4	1.750

In [79]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x ,y, train_size = 0.7, t
```

In [80]:
```
x_train
```

Out[80]:

	avg_click_att	reclick_percent_att	avg_click_norm	reclick_percent_norm	avg_euc_dist	avg
155	7.269231	0.692308	5.698630	0.287671	5330.330911	
722	6.000000	0.000000	5.146341	0.073171	4860.645327	
678	6.250000	0.083333	5.390244	0.195122	6206.910479	
492	6.916667	0.416667	5.512195	0.219512	4793.090106	
786	6.666667	0.250000	5.682927	0.317073	4647.898088	
...	...	...	...	...	...	
73	6.346154	0.153846	5.315068	0.164384	5565.231164	
400	7.916667	0.416667	6.097561	0.292683	5191.897515	
118	6.692308	0.384615	5.575342	0.287671	4225.796672	
701	6.500000	0.250000	6.000000	0.414634	5927.018920	
206	6.961538	0.615385	5.767123	0.356164	5460.168478	

553 rows × 11 columns

In [81]:
```
y_train
```

Out[81]:

	Extraversion_N
155	1.750
722	1.750
678	2.500
492	2.875
786	1.000
...	...
73	3.750
400	3.125
118	1.500
701	4.000
206	3.250

553 rows × 1 columns

In [82]:
```
x_test
```

Out[82]:

	avg_click_att	reclick_percent_att	avg_click_norm	reclick_percent_norm	avg_euc_dist	avg
470	7.000000	0.416667	5.560976	0.292683	7720.484362	
788	6.333333	0.250000	5.341463	0.219512	4961.114325	
283	6.250000	0.166667	5.317073	0.195122	5997.524694	
151	6.538462	0.230769	5.561644	0.205479	4541.612765	
762	9.666667	1.000000	10.000000	1.000000	7523.975701	
...	...	...	...	...	...	
747	6.750000	0.333333	5.609756	0.268293	4716.845952	
433	6.916667	0.333333	6.560976	0.414634	5119.802216	
620	6.083333	0.083333	5.341463	0.146341	4972.706311	
419	8.333333	0.666667	5.219512	0.121951	5690.435700	
232	6.461538	0.230769	5.465753	0.246575	5191.469672	

238 rows × 11 columns

In [83]:
```
y_test
```

Out[83]:

	Extraversion_N
470	3.250
788	2.500
283	2.875
151	3.750
762	3.250
...	...
747	1.500
433	4.375
620	1.375
419	2.875
232	1.750

238 rows × 1 columns

In [84]:
```
'''In Linear Regression model it is required to add a constant'''
#Adding a constant to get an intercept
x_train_sm = sm.add_constant(x_train)
```

In [85]:
```
'''Fitting the regression model using OLS(of sm)'''
lr  = sm.OLS(y_train, x_train_sm).fit()
```

In [86]:
```
lr.params
```

Out[86]:
```
const                  3.522395
avg_click_att          0.029748
reclick_percent_att    0.318623
avg_click_norm        -0.125596
reclick_percent_norm   0.155747
avg_euc_dist           0.000111
avg_euc_speed         -1.187895
avg_completion_time    0.000002
total_pause_cnt        0.000292
avg_fixation_dur       0.000082
avg_agg_fixation_dur  -0.000022
avg_fixation_cnt      -0.047829
dtype: float64
```

In [87]:
```
lr.summary()
```

Out[87]:

OLS Regression Results

Dep. Variable:	Extraversion_N	**R-squared:**	0.028
Model:	OLS	**Adj. R-squared:**	0.008
Method:	Least Squares	**F-statistic:**	1.419
Date:	Sun, 24 Mar 2024	**Prob (F-statistic):**	0.160
Time:	02:34:19	**Log-Likelihood:**	-760.86
No. Observations:	553	**AIC:**	1546.
Df Residuals:	541	**BIC:**	1598.
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	3.5224	0.732	4.811	0.000	2.084	4.961
avg_click_att	0.0297	0.031	0.965	0.335	-0.031	0.090
reclick_percent_att	0.3186	0.278	1.147	0.252	-0.227	0.864
avg_click_norm	-0.1256	0.106	-1.190	0.234	-0.333	0.082
reclick_percent_norm	0.1557	0.448	0.348	0.728	-0.723	1.035
avg_euc_dist	0.0001	6.39e-05	1.737	0.083	-1.45e-05	0.000
avg_euc_speed	-1.1879	0.875	-1.358	0.175	-2.906	0.530
avg_completion_time	1.524e-06	1.13e-05	0.135	0.893	-2.07e-05	2.37e-05
total_pause_cnt	0.0003	0.005	0.061	0.952	-0.009	0.010
avg_fixation_dur	8.164e-05	0.000	0.260	0.795	-0.001	0.001
avg_agg_fixation_dur	-2.181e-05	4.15e-05	-0.525	0.600	-0.000	5.98e-05
avg_fixation_cnt	-0.0478	0.046	-1.041	0.298	-0.138	0.042

Omnibus:	19.287	**Durbin-Watson:**	1.999
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	11.793
Skew:	0.206	**Prob(JB):**	0.00275
Kurtosis:	2.414	**Cond. No.**	4.86e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.86e+05. This might indicate that there are strong multicollinearity or other numerical problems.

Extraversion_N | R-squared: 0.028

Linear Regression Model predicting Agreeableness from 11 Mouse related features

In [88]:
```
y_cols_a = ['Agreeableness_N']
```

In [89]:
```
y_a = click_data[y_cols_a]
```

In [90]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x ,y_a, train_size = 0.7,
```

In [91]:
```
y_train
```

Out[91]:

	Agreeableness_N
155	4.222222
722	3.000000
678	4.111111
492	4.888889
786	3.777778
...	...
73	4.000000
400	4.111111
118	4.555556
701	5.000000
206	4.444444

553 rows × 1 columns

In [92]:
```
lr  = sm.OLS(y_train, x_train_sm).fit()
```

In [93]:
```
lr.params
```

Out[93]:
```
const                  4.174651
avg_click_att         -0.038417
reclick_percent_att    0.083698
avg_click_norm        -0.073027
reclick_percent_norm  -0.010543
avg_euc_dist           0.000002
avg_euc_speed          0.007358
avg_completion_time   -0.000004
total_pause_cnt        0.003602
avg_fixation_dur      -0.000135
avg_agg_fixation_dur   0.000010
avg_fixation_cnt       0.029593
dtype: float64
```

In [94]:
```
lr.summary()
```

Out[94]:

OLS Regression Results

Dep. Variable:	Agreeableness_N	**R-squared:**	0.026
Model:	OLS	**Adj. R-squared:**	0.006
Method:	Least Squares	**F-statistic:**	1.323
Date:	Sun, 24 Mar 2024	**Prob (F-statistic):**	0.208
Time:	02:34:31	**Log-Likelihood:**	-622.12
No. Observations:	553	**AIC:**	1268.
Df Residuals:	541	**BIC:**	1320.
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	4.1747	0.570	7.328	0.000	3.056	5.294
avg_click_att	-0.0384	0.024	-1.601	0.110	-0.086	0.009
reclick_percent_att	0.0837	0.216	0.387	0.699	-0.341	0.508
avg_click_norm	-0.0730	0.082	-0.890	0.374	-0.234	0.088
reclick_percent_norm	-0.0105	0.348	-0.030	0.976	-0.695	0.673
avg_euc_dist	2.103e-06	4.97e-05	0.042	0.966	-9.55e-05	9.97e-05
avg_euc_speed	0.0074	0.680	0.011	0.991	-1.329	1.344
avg_completion_time	-3.873e-06	8.8e-06	-0.440	0.660	-2.12e-05	1.34e-05
total_pause_cnt	0.0036	0.004	0.965	0.335	-0.004	0.011
avg_fixation_dur	-0.0001	0.000	-0.552	0.581	-0.001	0.000
avg_agg_fixation_dur	1.023e-05	3.23e-05	0.317	0.752	-5.33e-05	7.37e-05
avg_fixation_cnt	0.0296	0.036	0.828	0.408	-0.041	0.100

Omnibus:	14.092	**Durbin-Watson:**	2.102
Prob(Omnibus):	0.001	**Jarque-Bera (JB):**	13.496
Skew:	-0.340	**Prob(JB):**	0.00117
Kurtosis:	2.648	**Cond. No.**	4.86e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.86e+05. This might indicate that there are strong multicollinearity or other numerical problems.

Agreeableness_N | R-squared: 0.026

Linear Regression Model predicting Conscientiousness from 11 Mouse related features

In [95]:
```
y_cons = ['Conscientiousness_N']
y_c = click_data[y_cons]
```

In [96]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x ,y_c, train_size = 0.7,
```

In [97]:
```
y_train.head()
```

Out[97]:

	Conscientiousness_N
155	3.555556
722	3.666667
678	4.111111
492	4.555556
786	3.888889

In [98]:
```
lr  = sm.OLS(y_train, x_train_sm).fit()
```

In [99]:
```
lr.params
```

Out[99]:
```
const                  4.939827e+00
avg_click_att         -2.451373e-02
reclick_percent_att    3.173402e-02
avg_click_norm        -9.896562e-02
reclick_percent_norm   4.100689e-02
avg_euc_dist           2.221315e-06
avg_euc_speed         -5.306181e-02
avg_completion_time    9.945278e-07
total_pause_cnt       -5.726513e-03
avg_fixation_dur      -6.999461e-04
avg_agg_fixation_dur   9.852319e-05
avg_fixation_cnt      -4.744887e-02
dtype: float64
```

In [100]:
```
lr.summary()
```

Out[100]:

OLS Regression Results

Dep. Variable:	Conscientiousness_N	**R-squared:**	0.050
Model:	OLS	**Adj. R-squared:**	0.031
Method:	Least Squares	**F-statistic:**	2.581
Date:	Sun, 24 Mar 2024	**Prob (F-statistic):**	0.00336
Time:	02:34:42	**Log-Likelihood:**	-634.91
No. Observations:	553	**AIC:**	1294.
Df Residuals:	541	**BIC:**	1346.
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	4.9398	0.583	8.473	0.000	3.795	6.085
avg_click_att	-0.0245	0.025	-0.998	0.319	-0.073	0.024
reclick_percent_att	0.0317	0.221	0.143	0.886	-0.403	0.466
avg_click_norm	-0.0990	0.084	-1.178	0.239	-0.264	0.066
reclick_percent_norm	0.0410	0.356	0.115	0.908	-0.659	0.741
avg_euc_dist	2.221e-06	5.09e-05	0.044	0.965	-9.77e-05	0.000
avg_euc_speed	-0.0531	0.696	-0.076	0.939	-1.421	1.315
avg_completion_time	9.945e-07	9.01e-06	0.110	0.912	-1.67e-05	1.87e-05
total_pause_cnt	-0.0057	0.004	-1.499	0.135	-0.013	0.002
avg_fixation_dur	-0.0007	0.000	-2.800	0.005	-0.001	-0.000
avg_agg_fixation_dur	9.852e-05	3.31e-05	2.978	0.003	3.35e-05	0.000
avg_fixation_cnt	-0.0474	0.037	-1.297	0.195	-0.119	0.024

Omnibus:	45.596	**Durbin-Watson:**	2.087
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	21.845
Skew:	-0.298	**Prob(JB):**	1.80e-05
Kurtosis:	2.231	**Cond. No.**	4.86e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.86e+05. This might indicate that there are strong multicollinearity or other numerical problems.

Conscientiousness_N | R-squared: 0.050

Linear Regression Model predicting Neuroticism from 11 Mouse related features

In [101]:
```
y_neuro = ['Neuroticism_N']
y_n = click_data[y_neuro]
```

In [102]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x ,y_n, train_size = 0.7,
```

In [103]:
```
y_train.head()
```

Out[103]:

	Neuroticism_N
155	3.625
722	4.000
678	1.625
492	1.625
786	3.125

In [104]:
```
lr  = sm.OLS(y_train, x_train_sm).fit()
```

In [105]:
```
lr.params
```

Out[105]:
```
const                  1.311031
avg_click_att          0.057058
reclick_percent_att   -0.271114
avg_click_norm         0.119666
reclick_percent_norm  -0.310447
avg_euc_dist          -0.000045
avg_euc_speed          0.867579
avg_completion_time   -0.000005
total_pause_cnt        0.005534
avg_fixation_dur       0.000204
avg_agg_fixation_dur  -0.000034
avg_fixation_cnt       0.032272
dtype: float64
```

In [106]:
```
lr.summary()
```

Out[106]:

OLS Regression Results

Dep. Variable:	Neuroticism_N	**R-squared:**	0.019
Model:	OLS	**Adj. R-squared:**	-0.001
Method:	Least Squares	**F-statistic:**	0.9499
Date:	Sun, 24 Mar 2024	**Prob (F-statistic):**	0.492
Time:	02:34:55	**Log-Likelihood:**	-754.36
No. Observations:	553	**AIC:**	1533.
Df Residuals:	541	**BIC:**	1585.
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	1.3110	0.724	1.812	0.071	-0.110	2.732
avg_click_att	0.0571	0.030	1.872	0.062	-0.003	0.117
reclick_percent_att	-0.2711	0.275	-0.988	0.324	-0.810	0.268
avg_click_norm	0.1197	0.104	1.148	0.252	-0.085	0.324
reclick_percent_norm	-0.3104	0.442	-0.702	0.483	-1.179	0.558
avg_euc_dist	-4.539e-05	6.31e-05	-0.719	0.472	-0.000	7.86e-05
avg_euc_speed	0.8676	0.864	1.004	0.316	-0.830	2.565
avg_completion_time	-4.669e-06	1.12e-05	-0.418	0.676	-2.66e-05	1.73e-05
total_pause_cnt	0.0055	0.005	1.167	0.244	-0.004	0.015
avg_fixation_dur	0.0002	0.000	0.657	0.511	-0.000	0.001
avg_agg_fixation_dur	-3.41e-05	4.11e-05	-0.831	0.407	-0.000	4.66e-05
avg_fixation_cnt	0.0323	0.045	0.711	0.477	-0.057	0.121

Omnibus:	26.422	**Durbin-Watson:**	1.890
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	14.936
Skew:	0.238	**Prob(JB):**	0.000571
Kurtosis:	2.350	**Cond. No.**	4.86e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.86e+05. This might indicate that there are strong multicollinearity or other numerical problems.

Neuroticism_N | R-squared: 0.019

Linear Regression Model predicting Openness from 11 Mouse related features

In [107]:
```
y_open = ['Openness_N']
y_o = click_data[y_open]
```

In [108]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x ,y_o, train_size = 0.7,
```

In [109]:
```
y_train.head()
```

Out[109]:

	Openness_N
155	3.3
722	4.0
678	4.1
492	4.4
786	3.1

In [110]:
```
lr  = sm.OLS(y_train, x_train_sm).fit()
```

In [111]:
```
lr.params
```

Out[111]:
```
const                  3.878423
avg_click_att          0.014319
reclick_percent_att   -0.499561
avg_click_norm         0.002225
reclick_percent_norm  -0.244403
avg_euc_dist           0.000037
avg_euc_speed         -0.378896
avg_completion_time   -0.000005
total_pause_cnt        0.000418
avg_fixation_dur      -0.000416
avg_agg_fixation_dur   0.000041
avg_fixation_cnt      -0.005638
dtype: float64
```

In [112]:
```
lr.summary()
```

Out[112]:

OLS Regression Results

Dep. Variable:	Openness_N	**R-squared:**	0.041
Model:	OLS	**Adj. R-squared:**	0.022
Method:	Least Squares	**F-statistic:**	2.110
Date:	Sun, 24 Mar 2024	**Prob (F-statistic):**	0.0182
Time:	02:35:00	**Log-Likelihood:**	-624.87
No. Observations:	553	**AIC:**	1274.
Df Residuals:	541	**BIC:**	1326.
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	3.8784	0.573	6.774	0.000	2.754	5.003
avg_click_att	0.0143	0.024	0.594	0.553	-0.033	0.062
reclick_percent_att	-0.4996	0.217	-2.300	0.022	-0.926	-0.073
avg_click_norm	0.0022	0.083	0.027	0.978	-0.160	0.164
reclick_percent_norm	-0.2444	0.350	-0.698	0.485	-0.932	0.443
avg_euc_dist	3.655e-05	5e-05	0.732	0.465	-6.16e-05	0.000
avg_euc_speed	-0.3789	0.684	-0.554	0.580	-1.722	0.964
avg_completion_time	-5.337e-06	8.85e-06	-0.603	0.547	-2.27e-05	1.2e-05
total_pause_cnt	0.0004	0.004	0.111	0.911	-0.007	0.008
avg_fixation_dur	-0.0004	0.000	-1.696	0.090	-0.001	6.58e-05
avg_agg_fixation_dur	4.085e-05	3.25e-05	1.257	0.209	-2.3e-05	0.000
avg_fixation_cnt	-0.0056	0.036	-0.157	0.875	-0.076	0.065

Omnibus:	21.739	**Durbin-Watson:**	1.997
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	23.406
Skew:	-0.502	**Prob(JB):**	8.27e-06
Kurtosis:	3.094	**Cond. No.**	4.86e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.86e+05. This might indicate that there are strong multicollinearity or other numerical problems.

Openness_N | R-squared: 0.041

R-Squared Value : Extraversion : 0.028 Agreeableness : 0.026 Conscietiousness : 0.050 Neuroticism : 0.019 Openness : 0.041

PLS ANALYSIS FOR BIG 5 TRAITS ~ MOUSE MOVEMENTS & AUC

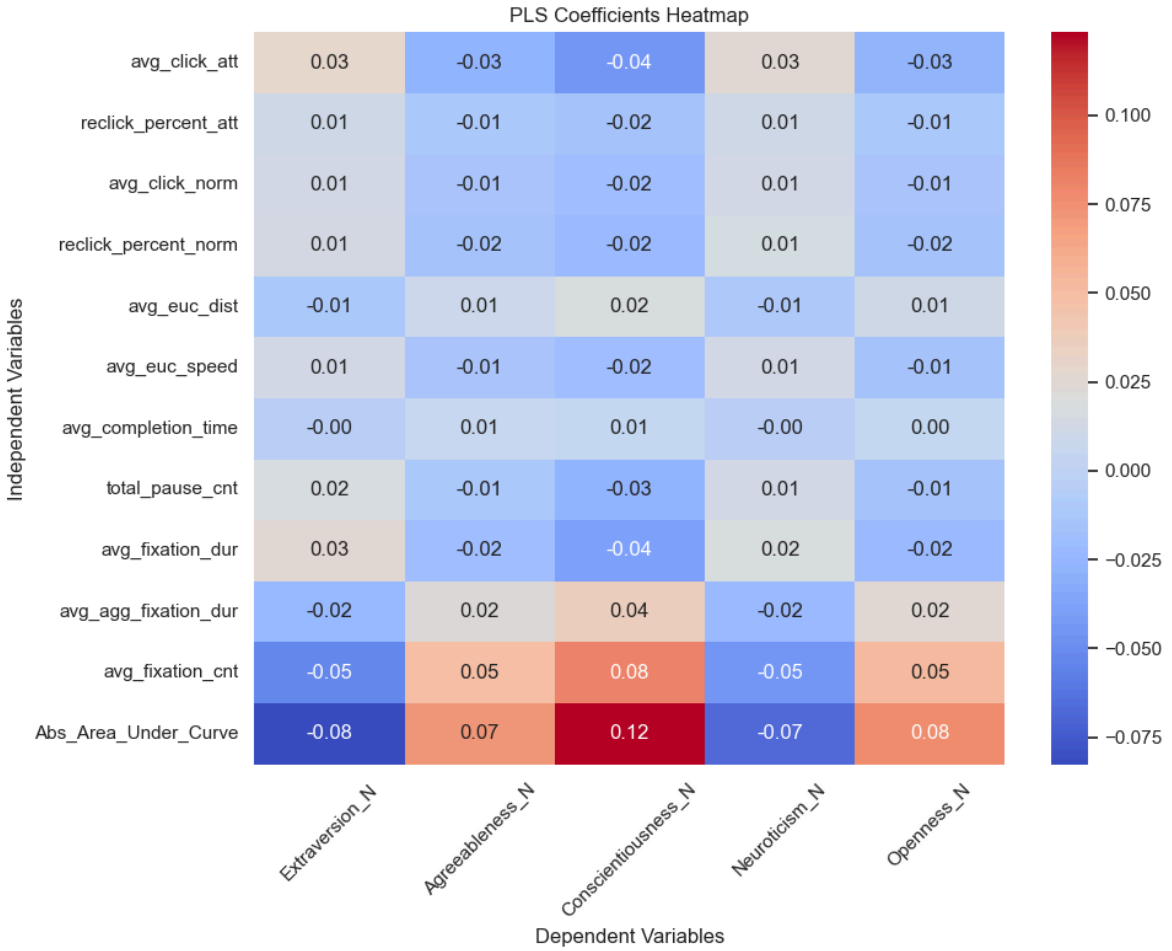
In [113]:
```
x_cols = ['avg_click_att', 'reclick_percent_att', 'avg_click_norm', 'reclick_
          'avg_euc_speed', 'avg_completion_time', 'total_pause_cnt', 'avg_fix
          'avg_fixation_cnt','Abs_Area_Under_Curve']

y_cols = ['Extraversion_N','Agreeableness_N','Conscientiousness_N','Neuroticis

x = click_data[x_cols]
y = click_data[y_cols]
```

In [114]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x ,y, train_size = 0.7, t
```

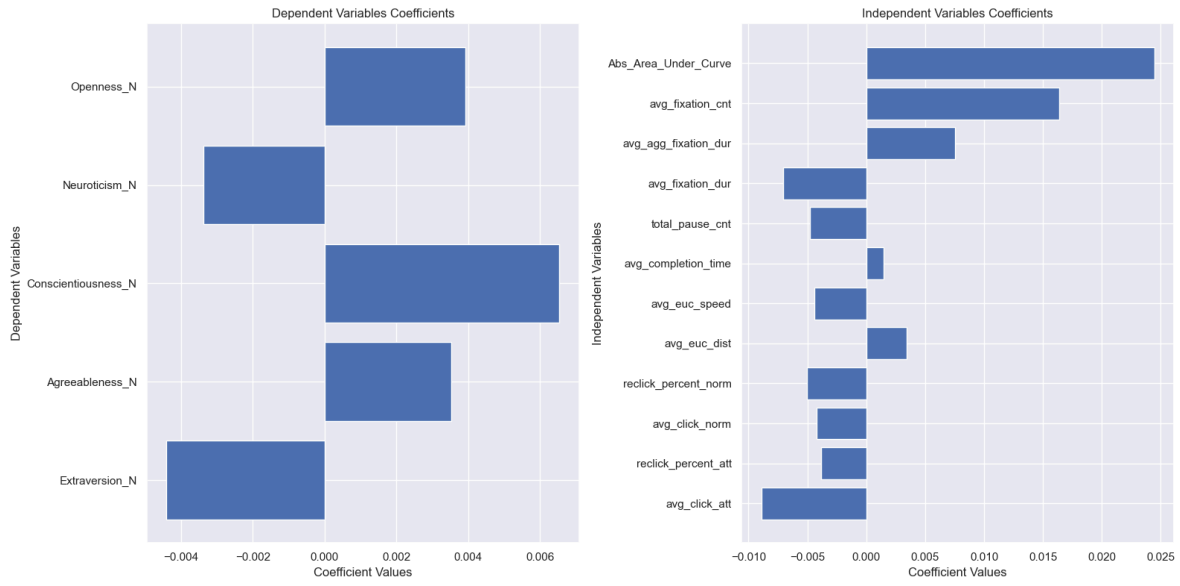
In [115]:
```
from sklearn.cross_decomposition import PLSRegression

# Assuming X contains 11 independent variables and Y contains 5 dependent var
# Replace X_data and Y_data with your actual data
#X = X_data  # Shape: (n_samples, 11)
#Y = Y_data  # Shape: (n_samples, 5)

# Instantiate PLS model with the desired number of components
n_components = 2  # You can adjust this value
pls = PLSRegression(n_components=n_components)

# Fit the model to the data
pls.fit(x, y)

# Get the regression coefficients
coefficients = pls.coef_

# Optionally, evaluate the model performance
# For example, you can calculate R^2
R2 = pls.score(x, y)

# Print the regression coefficients
print("Regression Coefficients:")
print(coefficients)

# Print the R^2 score
print("R^2 Score:", R2)
```

```
Regression Coefficients:
[[ 0.02882428 -0.02675828 -0.04331724  0.02510108 -0.02846811]
 [ 0.01056019 -0.012552   -0.0162606   0.01151271 -0.012512  ]
 [ 0.01200566 -0.0136266  -0.01839486  0.01254622 -0.01373714]
 [ 0.01467476 -0.01597266 -0.02238727  0.01475953 -0.01627347]
 [-0.01246636  0.00942702  0.01842949 -0.00904766  0.01068675]
 [ 0.0125727  -0.01408019 -0.01923666  0.01297866 -0.01424202]
 [-0.0031744   0.00516228  0.00508539 -0.00463146  0.00481346]
 [ 0.01790848 -0.01295264 -0.02639097  0.01250038 -0.01490527]
 [ 0.02632405 -0.01913963 -0.03880688  0.01845907 -0.02198553]
 [-0.0231516   0.02320821  0.03503622 -0.02160736  0.02416549]
 [-0.05436086  0.04834636  0.08139264 -0.045554    0.05208452]
 [-0.08261698  0.07134156  0.12339614 -0.06743337  0.07754031]]
R^2 Score: 0.03216926928227559
```

In [116]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Create a DataFrame for the coefficients
coefficients_df = pd.DataFrame(coefficients, columns=y_cols,
                               index=x_cols)

# Create a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(coefficients_df, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('PLS Coefficients Heatmap')
plt.xlabel('Dependent Variables')
plt.ylabel('Independent Variables')

# Customize x-axis and y-axis labels
plt.xticks(ticks=np.arange(len(y_cols)) + 0.5, labels=y_cols, rotation=45)
plt.yticks(ticks=np.arange(len(x_cols)) + 0.5, labels=x_cols, rotation=0)

plt.tight_layout()
plt.show()
```



In [117]:
```
import matplotlib.pyplot as plt
import numpy as np

# Create subplots with 1 row and 2 columns
fig, axs = plt.subplots(1, 2, figsize=(16, 8))

# Plot for dependent variables
axs[0].barh(y_cols, coefficients.mean(axis=0))
axs[0].set_xlabel('Coefficient Values')
axs[0].set_ylabel('Dependent Variables')
axs[0].set_title('Dependent Variables Coefficients')

# Plot for independent variables
axs[1].barh(x_cols, coefficients.mean(axis=1))
axs[1].set_xlabel('Coefficient Values')
axs[1].set_ylabel('Independent Variables')
axs[1].set_title('Independent Variables Coefficients')

# Adjust layout
plt.tight_layout()

# Show plots
plt.show()
```



PLS ANALYSIS FOR BIG FIVE WITH DEMOGRAPHICS ~ MOUSE MOVEMENTS & AUC

In [118]:
```
from sklearn import preprocessing
```

In [119]:
```
data_age = click_data
```

In [120]:
```
def preprocessor(df):
    res_df = df.copy()
    le = preprocessing.LabelEncoder()
 #LabelEncoder is encoding the columns into numerical values
    res_df['Gender']=le.fit_transform(res_df['Gender'])
    return res_df
```

In [121]:
```
encoded_age = preprocessor(data_age)
```

In [122]:
```
encoded_age.head()
```

Out[122]:

	Age	BirthYear	Gender	Extraversion	Agreeableness	Conscientiousness	Neuroticism	Open
0	45	1975	0	20	33	40	34	
1	29	1991	1	31	44	43	10	
2	58	1962	1	13	31	21	35	
3	30	1990	1	33	19	25	15	
4	46	1974	1	14	38	33	21	

5 rows × 30 columns

In [123]:
```
x_cols_ag = ['avg_click_att', 'reclick_percent_att', 'avg_click_norm', 'reclic
          'avg_euc_speed', 'avg_completion_time', 'total_pause_cnt', 'avg_fixa
          'avg_fixation_cnt','Abs_Area_Under_Curve']

y_cols_ag = ['Extraversion_N','Agreeableness_N','Conscientiousness_N','Neurot

x_ag = encoded_age[x_cols_ag]
y_ag = encoded_age[y_cols_ag]
```

In [124]:
```
x_ag.head()
```

Out[124]:

	avg_click_att	reclick_percent_att	avg_click_norm	reclick_percent_norm	avg_euc_dist	avg_e
0	6.307692	0.115385	5.520548	0.205479	6144.443128	
1	6.846154	0.384615	5.808219	0.342466	7322.577197	
2	6.346154	0.269231	5.493151	0.191781	3939.286111	
3	6.307692	0.230769	5.424658	0.273973	5434.640400	
4	6.692308	0.423077	5.205479	0.068493	3723.353789	

In [125]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_ag ,y_ag, train_size = (
```

In [126]:
```
x_train.head()
```

Out[126]:

	avg_click_att	reclick_percent_att	avg_click_norm	reclick_percent_norm	avg_euc_dist	avg
155	7.269231	0.692308	5.698630	0.287671	5330.330911	
722	6.000000	0.000000	5.146341	0.073171	4860.645327	
678	6.250000	0.083333	5.390244	0.195122	6206.910479	
492	6.916667	0.416667	5.512195	0.219512	4793.090106	
786	6.666667	0.250000	5.682927	0.317073	4647.898088	

In [127]:
```
y_train.head()
```

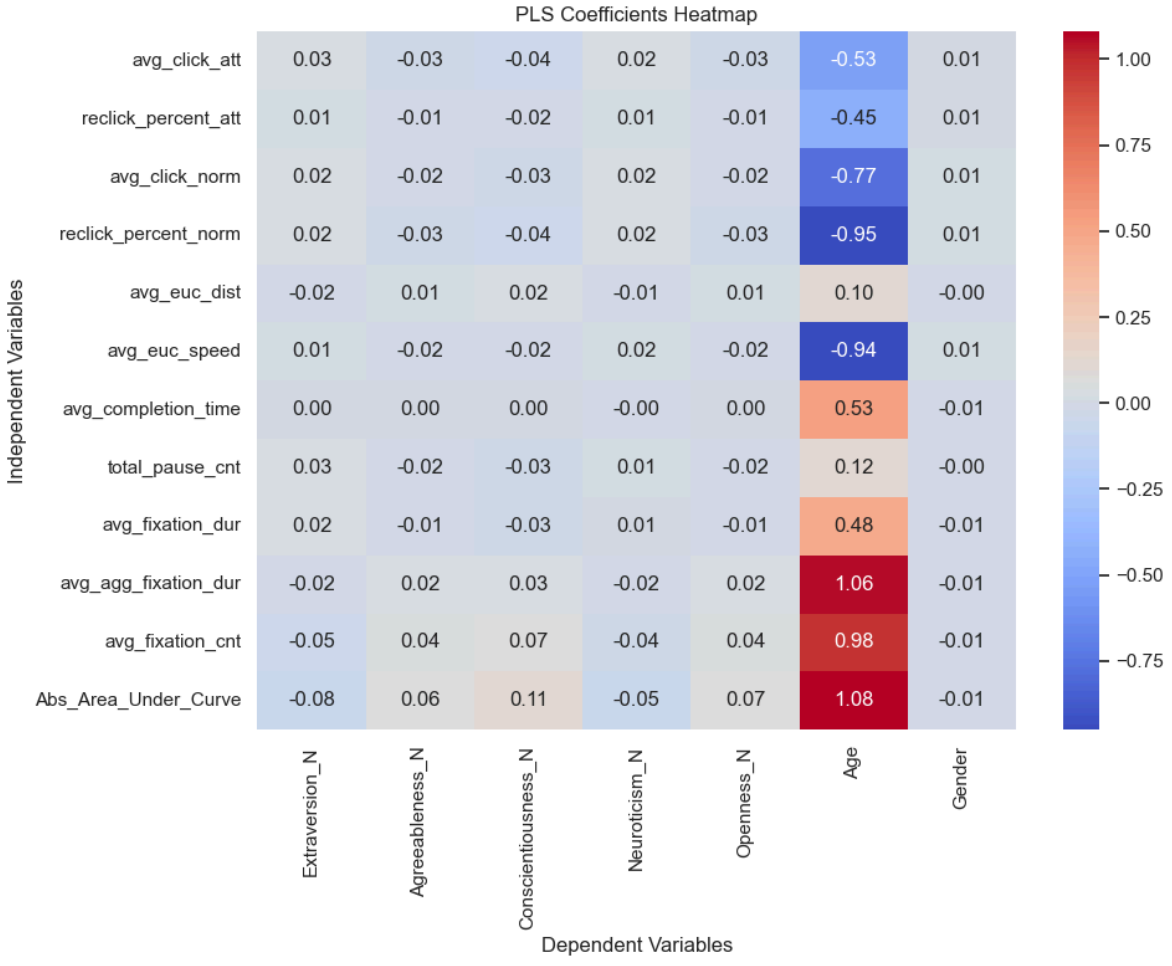
Out[127]:

	Extraversion_N	Agreeableness_N	Conscientiousness_N	Neuroticism_N	Openness_N	Age
155	1.750	4.222222	3.555556	3.625	3.3	38
722	1.750	3.000000	3.666667	4.000	4.0	36
678	2.500	4.111111	4.111111	1.625	4.1	31
492	2.875	4.888889	4.555556	1.625	4.4	30
786	1.000	3.777778	3.888889	3.125	3.1	40

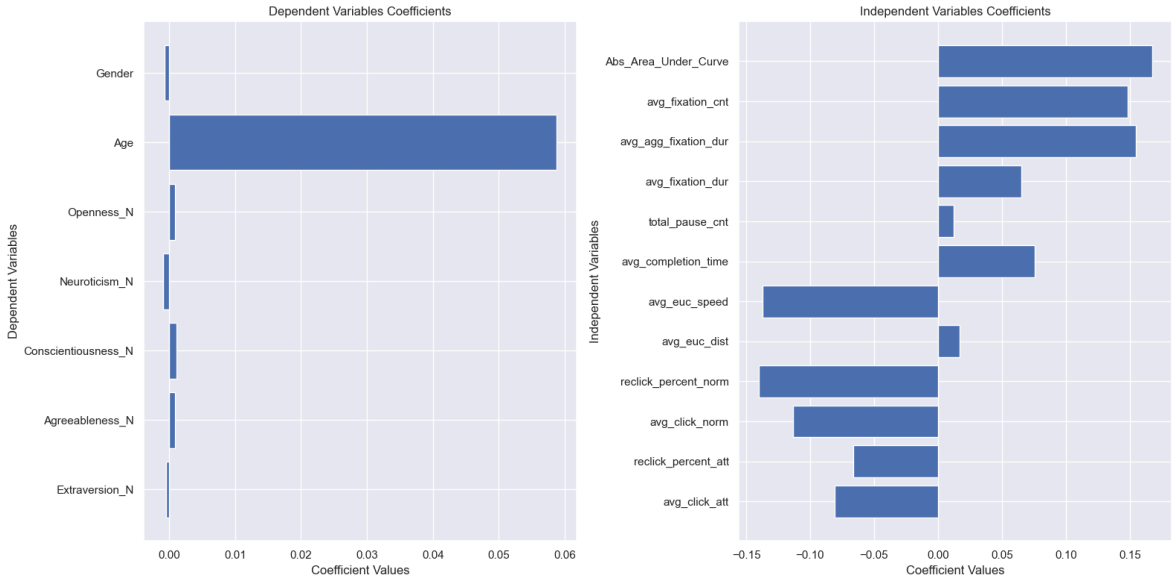
In [128]:
```
from sklearn.cross_decomposition import PLSRegression

# Assuming X contains 11 independent variables and Y contains 5 dependent var
# Replace X_data and Y_data with your actual data
#X = X_data  # Shape: (n_samples, 11)
#Y = Y_data  # Shape: (n_samples, 5)

# Instantiate PLS model with the desired number of components
n_components = 2 #ou can adjust this value
pls = PLSRegression(n_components=n_components)

# Fit the model to the data
pls.fit(x_ag, y_ag)

# Get the regression coefficients
coefficients = pls.coef_

# Optionally, evaluate the model performance
# For example, you can calculate R^2
R2 = pls.score(x_ag, y_ag)

# Print the regression coefficients
print("Regression Coefficients:")
print(coefficients)

# Print the R^2 score
print("R^2 Score:", R2)
```

```
Regression Coefficients:
[[ 0.0295027  -0.0254493  -0.04187167  0.02236138 -0.02711481 -0.53041266
   0.0069359 ]
 [ 0.01372203 -0.01366848 -0.02157112  0.0125911  -0.01438486 -0.44663153
   0.00575466]
 [ 0.01886431 -0.02022026 -0.03129074  0.0190192  -0.02115964 -0.77004073
   0.00988471]
 [ 0.02475499 -0.02598085 -0.04042837  0.02429631 -0.02723117 -0.95008357
   0.01220725]
 [-0.01579995  0.01195732  0.02051086 -0.00997608  0.01290245  0.10157687
  -0.00140647]
 [ 0.00962373 -0.01537225 -0.02174985  0.01575017 -0.01569063 -0.94478053
   0.01202359]
 [ 0.00245791  0.00313703  0.00252774 -0.00442421  0.00283105  0.52963224
  -0.00667977]
 [ 0.02756563 -0.01816524 -0.03269911  0.01418043 -0.01989994  0.11707757
  -0.00126838]
 [ 0.02463236 -0.01277865 -0.02526744  0.00854129 -0.01443862  0.48157669
  -0.00590111]
 [-0.01755761  0.02194054  0.03269469 -0.02143407  0.02271557  1.0573431
  -0.01350839]
 [-0.04618236  0.04121363  0.06711924 -0.03664956  0.04377698  0.98050884
  -0.01275716]
 [-0.07677762  0.06347416  0.10581396 -0.0548985   0.06789611  1.07964685
  -0.01424678]]
R^2 Score: 0.04386497705152367
```

In [129]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Create a DataFrame for the coefficients
coefficients_df = pd.DataFrame(coefficients, columns=y_cols_ag,
                               index=x_cols_ag)

# Create a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(coefficients_df, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('PLS Coefficients Heatmap')
plt.xlabel('Dependent Variables')
plt.ylabel('Independent Variables')

# Customize x-axis and y-axis labels
plt.tight_layout()
plt.show()
```



In [131]:
```
import matplotlib.pyplot as plt
import numpy as np

# Create subplots with 1 row and 2 columns
fig, axs = plt.subplots(1, 2, figsize=(16, 8))

# Plot for dependent variables
axs[0].barh(y_cols_ag, coefficients.mean(axis=0))
axs[0].set_xlabel('Coefficient Values')
axs[0].set_ylabel('Dependent Variables')
axs[0].set_title('Dependent Variables Coefficients')

# Plot for independent variables
axs[1].barh(x_cols_ag, coefficients.mean(axis=1))
axs[1].set_xlabel('Coefficient Values')
axs[1].set_ylabel('Independent Variables')
axs[1].set_title('Independent Variables Coefficients')

# Adjust layout
plt.tight_layout()

# Show plots
plt.show()
```



PLS ANALYSIS FOR BIG FIVE ~ MOUSE MOVEMENTS, DEMOGRAPHICS & AUC

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
dataset = pd.read_csv('personality_mouse_final_table (2).csv')
dataset.head()
```

Out[2]:

	Age	BirthYear	Gender	Extraversion	Agreeableness	Conscientiousness	Neuroticism	Open
0	45	1975	Female	20	33	40	34	
1	29	1991	Male	31	44	43	10	
2	58	1962	Male	13	31	21	35	
3	30	1990	Male	33	19	25	15	
4	46	1974	Male	14	38	33	21	

5 rows × 32 columns

In [3]:
```
click_data = dataset.drop(['pseudo_workerID','Mturk_ID'], axis = 1)
```

In [4]:
```
from sklearn import preprocessing
```

In [5]:
```
data_age = click_data
```

In [6]:
```
def preprocessor(df):
    res_df = df.copy()
    le = preprocessing.LabelEncoder()
 #LabelEncoder is encoding the columns into numerical values
    res_df['Gender']=le.fit_transform(res_df['Gender'])
    return res_df
```

In [7]:
```
encoded_age = preprocessor(data_age)
```

In [8]:
```
encoded_age.head()
```

Out[8]:

	Age	BirthYear	Gender	Extraversion	Agreeableness	Conscientiousness	Neuroticism	Open
0	45	1975	0	20	33	40	34	
1	29	1991	1	31	44	43	10	
2	58	1962	1	13	31	21	35	
3	30	1990	1	33	19	25	15	
4	46	1974	1	14	38	33	21	

5 rows × 30 columns

In [9]:
```
x_cols_ag = ['avg_click_att', 'reclick_percent_att', 'avg_click_norm', 'recli
          'avg_euc_speed', 'avg_completion_time', 'total_pause_cnt', 'avg_fix
          'avg_fixation_cnt','Abs_Area_Under_Curve','Age','Gender']

y_cols_ag = ['Extraversion_N','Agreeableness_N','Conscientiousness_N','Neurot

x_ag = encoded_age[x_cols_ag]
y_ag = encoded_age[y_cols_ag]
```

In [10]:
```
x_ag.head()
```

Out[10]:

	avg_click_att	reclick_percent_att	avg_click_norm	reclick_percent_norm	avg_euc_dist	avg_e
0	6.307692	0.115385	5.520548	0.205479	6144.443128	
1	6.846154	0.384615	5.808219	0.342466	7322.577197	
2	6.346154	0.269231	5.493151	0.191781	3939.286111	
3	6.307692	0.230769	5.424658	0.273973	5434.640400	
4	6.692308	0.423077	5.205479	0.068493	3723.353789	

In [11]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_ag ,y_ag, train_size = 
```

In [12]:
```
x_train.head()
```

Out[12]:

	avg_click_att	reclick_percent_att	avg_click_norm	reclick_percent_norm	avg_euc_dist	avg
155	7.269231	0.692308	5.698630	0.287671	5330.330911	
722	6.000000	0.000000	5.146341	0.073171	4860.645327	
678	6.250000	0.083333	5.390244	0.195122	6206.910479	
492	6.916667	0.416667	5.512195	0.219512	4793.090106	
786	6.666667	0.250000	5.682927	0.317073	4647.898088	

In [13]:
```
y_train.head()
```

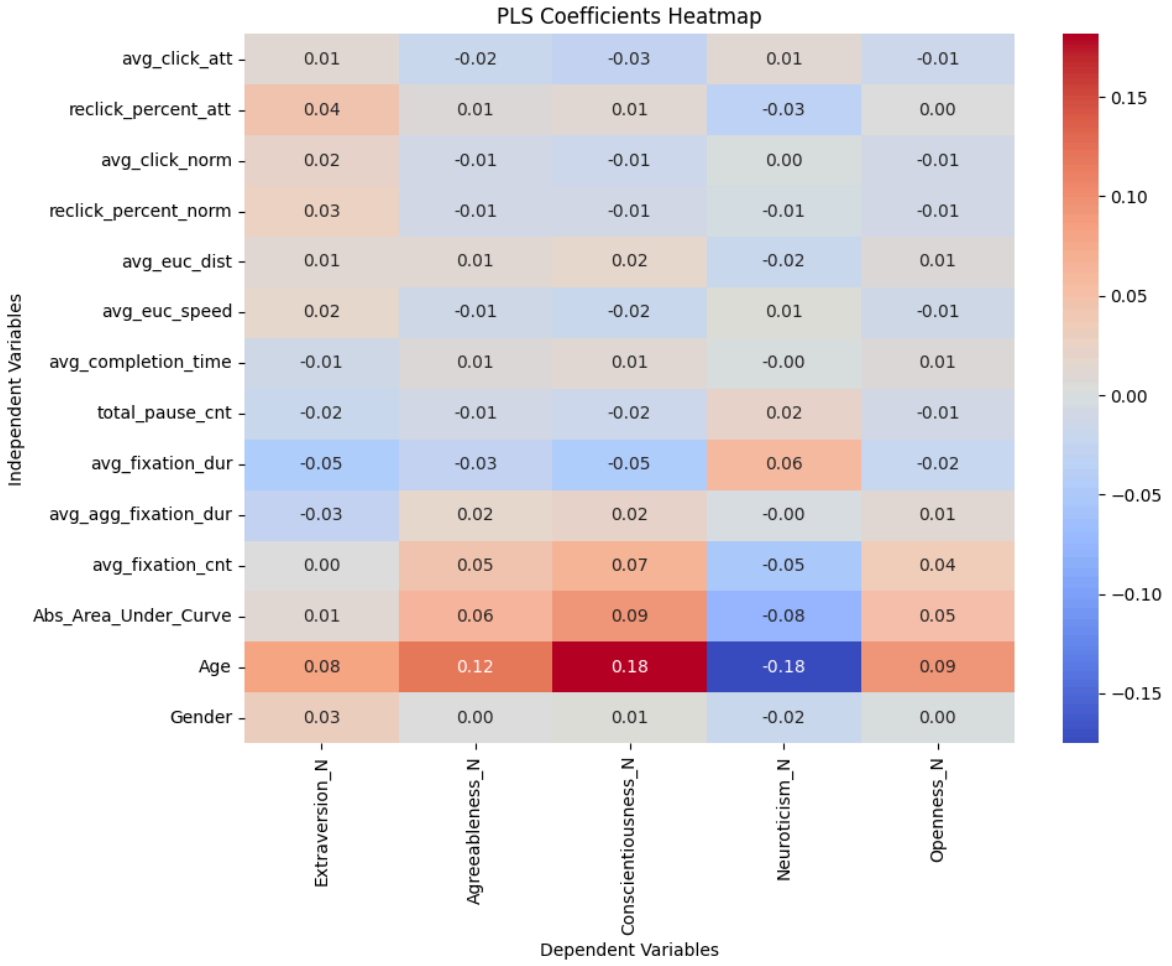
Out[13]:

	Extraversion_N	Agreeableness_N	Conscientiousness_N	Neuroticism_N	Openness_N
155	1.750	4.222222	3.555556	3.625	3.3
722	1.750	3.000000	3.666667	4.000	4.0
678	2.500	4.111111	4.111111	1.625	4.1
492	2.875	4.888889	4.555556	1.625	4.4
786	1.000	3.777778	3.888889	3.125	3.1

In [14]:
```
from sklearn.cross_decomposition import PLSRegression

# Assuming X contains 11 independent variables and Y contains 5 dependent var
# Replace X_data and Y_data with your actual data
#X = X_data  # Shape: (n_samples, 11)
#Y = Y_data  # Shape: (n_samples, 5)

# Instantiate PLS model with the desired number of components
n_components = 2 #ou can adjust this value
pls = PLSRegression(n_components=n_components)

# Fit the model to the data
pls.fit(x_ag, y_ag)

# Get the regression coefficients
coefficients = pls.coef_

# Optionally, evaluate the model performance
# For example, you can calculate R^2
R2 = pls.score(x_ag, y_ag)

# Print the regression coefficients
print("Regression Coefficients:")
print(coefficients)

# Print the R^2 score
print("R^2 Score:", R2)
```

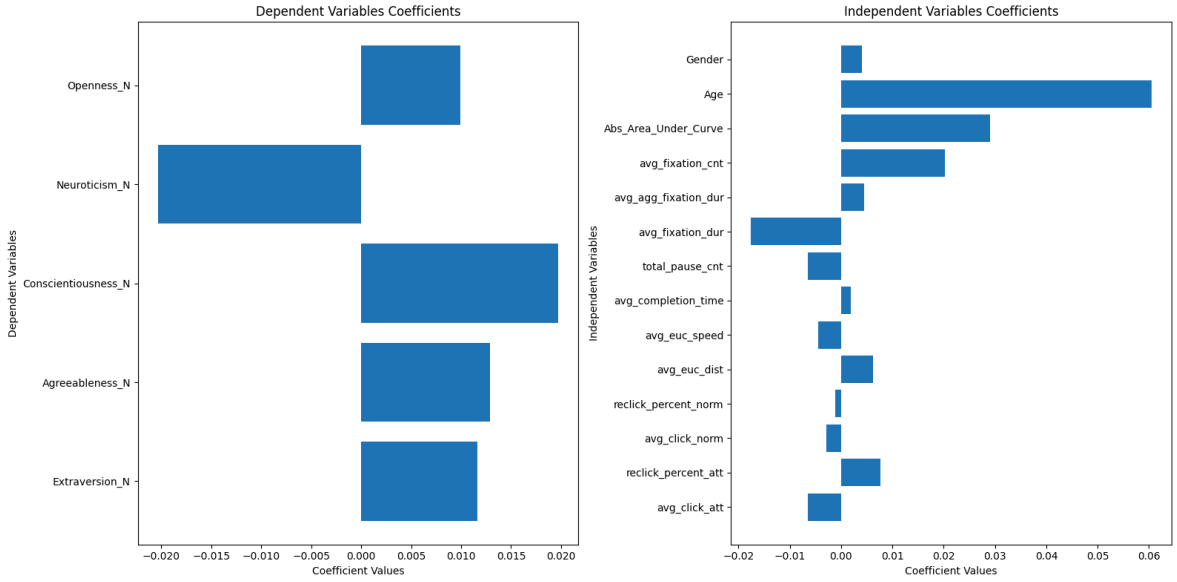
```
Regression Coefficients:
[[ 0.01401379 -0.01832171 -0.02542689  0.01203103 -0.01496909]
 [ 0.04428169  0.00721079  0.01433583 -0.03227104  0.00459159]
 [ 0.02077804 -0.01146973 -0.01487414  0.00089322 -0.00968427]
 [ 0.02767987 -0.00908508 -0.01080621 -0.00549774 -0.00796372]
 [ 0.01340511  0.01115283  0.01738471 -0.0194532   0.00853949]
 [ 0.01791506 -0.01457164 -0.01963408  0.00582518 -0.01208192]
 [-0.01278302  0.00723947  0.0094171  -0.00074718  0.00610386]
 [-0.01829431 -0.01063327 -0.01705412  0.02159584 -0.00799777]
 [-0.04680836 -0.02944691 -0.04689323  0.0576744  -0.02224896]
 [-0.0258443   0.01627538  0.02142255 -0.00327994  0.01364685]
 [ 0.00232197  0.0461432   0.06730741 -0.05110032  0.03671786]
 [ 0.01290095  0.06372077  0.09378989 -0.07592694  0.05045196]
 [ 0.08244954  0.12001469  0.18170317 -0.17516003  0.09350582]
 [ 0.03068142  0.00207956  0.00569131 -0.01921087  0.00085671]]
R^2 Score: 0.05522253502264374
```

In [15]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Create a DataFrame for the coefficients
coefficients_df = pd.DataFrame(coefficients, columns=y_cols_ag,
                               index=x_cols_ag)

# Create a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(coefficients_df, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('PLS Coefficients Heatmap')
plt.xlabel('Dependent Variables')
plt.ylabel('Independent Variables')

# Customize x-axis and y-axis labels
plt.tight_layout()
plt.show()
```



Agreeablenss & Conscientiousness is increasing with increase in Age & Neuroticism decreases with increase in Age

In [16]:
```
import matplotlib.pyplot as plt
import numpy as np

# Create subplots with 1 row and 2 columns
fig, axs = plt.subplots(1, 2, figsize=(16, 8))

# Plot for dependent variables
axs[0].barh(y_cols_ag, coefficients.mean(axis=0))
axs[0].set_xlabel('Coefficient Values')
axs[0].set_ylabel('Dependent Variables')
axs[0].set_title('Dependent Variables Coefficients')

# Plot for independent variables
axs[1].barh(x_cols_ag, coefficients.mean(axis=1))
axs[1].set_xlabel('Coefficient Values')
axs[1].set_ylabel('Independent Variables')
axs[1].set_title('Independent Variables Coefficients')

# Adjust layout
plt.tight_layout()

# Show plots
plt.show()
```



PLS ANALYSIS FOR BIG FIVE ~ MOUSE MOVEMENTS

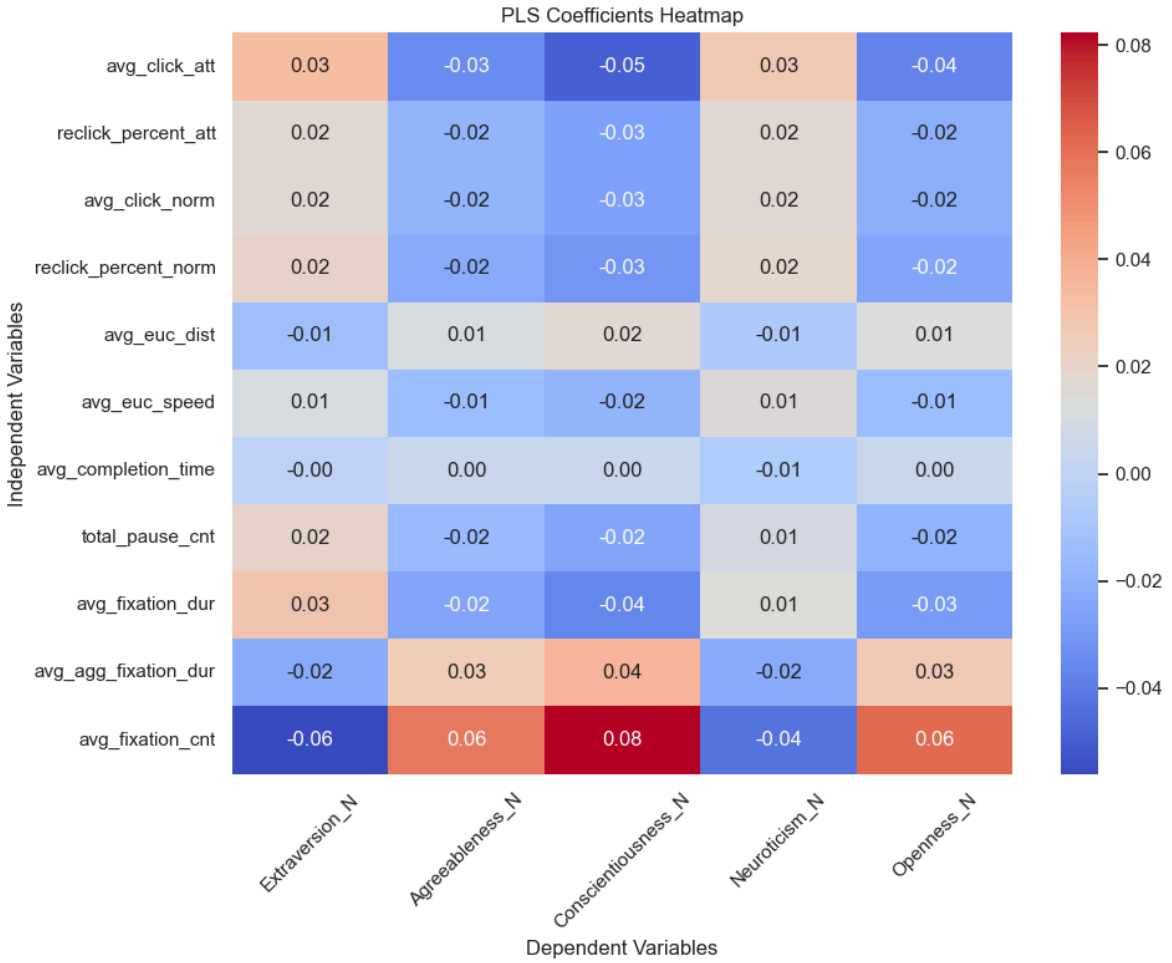
In [132]:
```
x_cols_only = ['avg_click_att', 'reclick_percent_att', 'avg_click_norm', 'rec
          'avg_euc_speed', 'avg_completion_time', 'total_pause_cnt', 'avg_fix
          'avg_fixation_cnt']

y_cols_only = ['Extraversion_N','Agreeableness_N','Conscientiousness_N','Neur

x_only = click_data[x_cols_only]
y_only = click_data[y_cols_only]
```

In [133]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_only ,y_only, train_siz
```

In [134]:
```
from sklearn.cross_decomposition import PLSRegression

# Assuming X contains 11 independent variables and Y contains 5 dependent var
# Replace X_data and Y_data with your actual data
#X = X_data  # Shape: (n_samples, 11)
#Y = Y_data  # Shape: (n_samples, 5)

# Instantiate PLS model with the desired number of components
n_components = 2  # You can adjust this value
pls = PLSRegression(n_components=n_components)

# Fit the model to the data
pls.fit(x_only, y_only)

# Get the regression coefficients
coefficients = pls.coef_

# Optionally, evaluate the model performance
# For example, you can calculate R^2
R2 = pls.score(x_only, y_only)

# Print the regression coefficients
print("Regression Coefficients:")
print(coefficients)

# Print the R^2 score
print("R^2 Score:", R2)
```

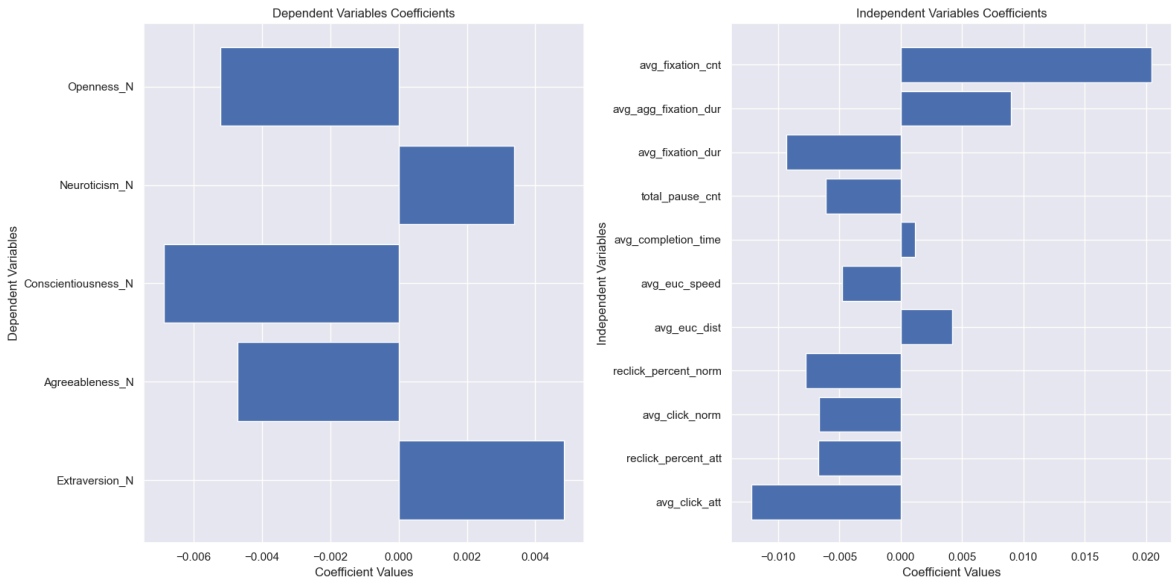
```
Regression Coefficients:
[[ 0.03318312 -0.03412174 -0.04914411  0.02631556 -0.03718739]
 [ 0.01710801 -0.01945049 -0.0273607   0.01665255 -0.02064384]
 [ 0.01668049 -0.01929621 -0.02703824  0.01678717 -0.02039062]
 [ 0.0197197  -0.02231023 -0.03141826  0.01901287 -0.02370858]
 [-0.01255574  0.01109153  0.01661392 -0.00693708  0.01263053]
 [ 0.01066087 -0.01451453 -0.01965659  0.01435099 -0.01475944]
 [-0.00106318  0.0041626   0.00491677 -0.00593826  0.00362139]
 [ 0.0191161  -0.0159654  -0.02429132  0.0090321  -0.01850049]
 [ 0.02892621 -0.02441866 -0.03704045  0.01409892 -0.02820055]
 [-0.02247486  0.02600788  0.03644007 -0.02263299  0.02748066]
 [-0.05602306  0.05700235  0.08231081 -0.04342363  0.06230422]]
R^2 Score: 0.01980242800175618
```

In [135]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Create a DataFrame for the coefficients
coefficients_df = pd.DataFrame(coefficients, columns=y_cols_only,
                               index=x_cols_only)

# Create a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(coefficients_df, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('PLS Coefficients Heatmap')
plt.xlabel('Dependent Variables')
plt.ylabel('Independent Variables')

# Customize x-axis and y-axis labels
plt.xticks(ticks=np.arange(len(y_cols_only)) + 0.5, labels=y_cols_only, rotat
plt.yticks(ticks=np.arange(len(x_cols_only)) + 0.5, labels=x_cols_only, rotat

plt.tight_layout()
plt.show()
```



Agreeableness, Conscientiousness and Openness are on positive end and Extraversion and Neuroticism is at negative side

In [136]:
```
import matplotlib.pyplot as plt
import numpy as np

# Create subplots with 1 row and 2 columns
fig, axs = plt.subplots(1, 2, figsize=(16, 8))

# Plot for dependent variables
axs[0].barh(y_cols_only, coefficients.mean(axis=0))
axs[0].set_xlabel('Coefficient Values')
axs[0].set_ylabel('Dependent Variables')
axs[0].set_title('Dependent Variables Coefficients')

# Plot for independent variables
axs[1].barh(x_cols_only, coefficients.mean(axis=1))
axs[1].set_xlabel('Coefficient Values')
axs[1].set_ylabel('Independent Variables')
axs[1].set_title('Independent Variables Coefficients')

# Adjust layout
plt.tight_layout()

# Show plots
plt.show()
```



Random Forest Regression Analysis

In [137]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
```

In [139]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_ag ,y_ag, train_size = (
```

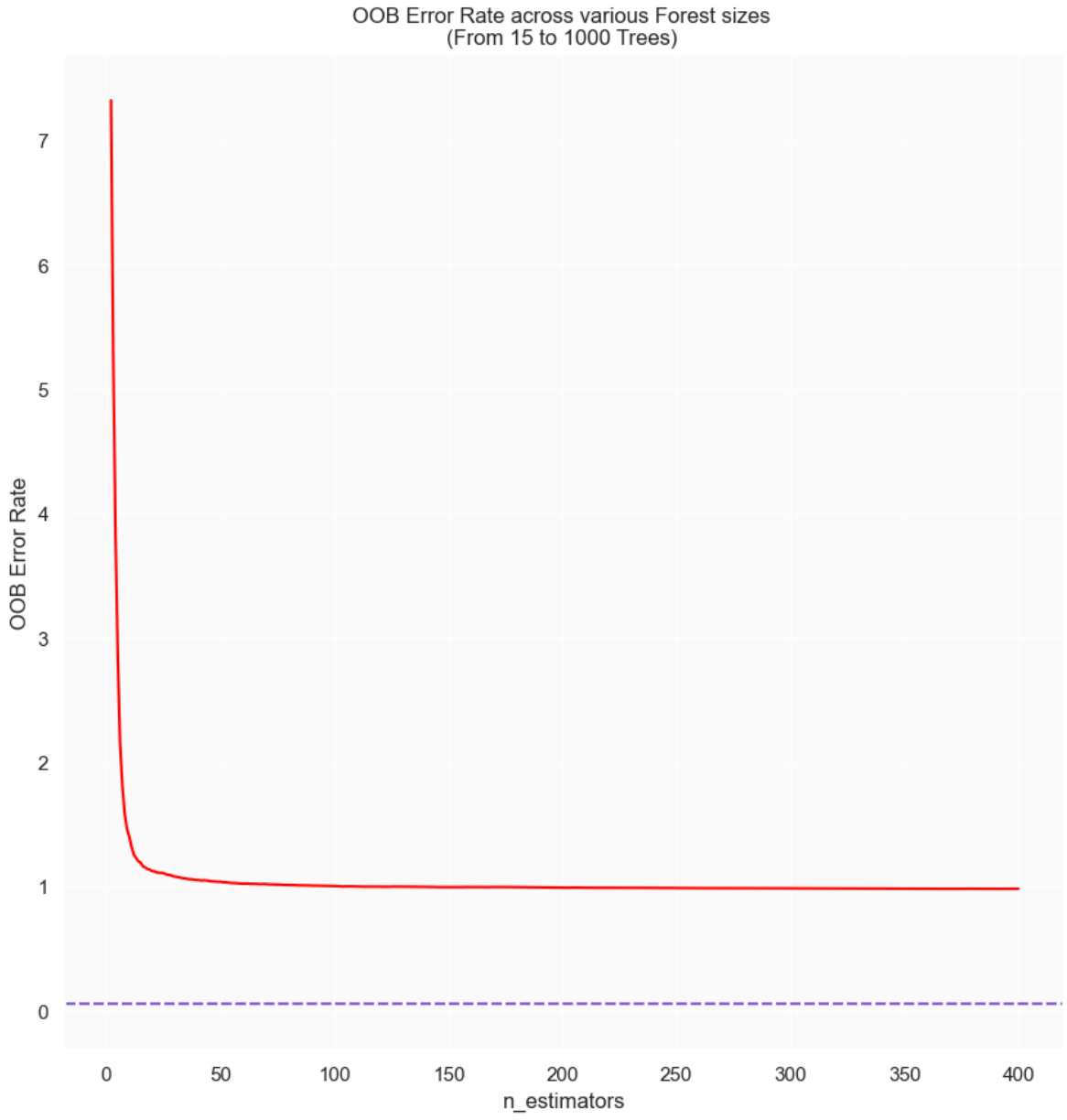
In [140]:
```
fit_rf = RandomForestRegressor(random_state = 21)
```

In [141]:
```
'''OOB Rate'''
fit_rf.set_params(warm_start = True, oob_score = True)

min_estimators = 2
max_estimators = 400

error_rate = {}

for i in range(min_estimators,max_estimators +1):
    fit_rf.set_params(n_estimators = i)
    fit_rf.fit(x_train, y_train)#fitting into x_test & y_test

    oob_error = 1 - fit_rf.oob_score_
    error_rate[i] = oob_error
```

In [142]:
```
oob_series = pd.Series(error_rate)
```

In [143]:
```
fig,ax = plt.subplots(figsize = (10,10))

ax.set_facecolor('#fafafa')

oob_series.plot(kind = 'line', color = 'red')
plt.axhline(0.055, color = '#875FDB',linestyle = '--')
plt.axhline(0.05, color = '#875FDB',linestyle = '--')
plt.xlabel('n_estimators')
plt.ylabel('OOB Error Rate')
plt.title('OOB Error Rate across various Forest sizes \n(From 15 to 1000 Trees
```

Out[143]:
```
Text(0.5, 1.0, 'OOB Error Rate across various Forest sizes \n(From 15 to 100
0 Trees) ')
```



In [144]:
```
rf_regressor = RandomForestRegressor(n_estimators=50, random_state=21)

rf_regressor.fit(x_train, y_train)

y_pred = rf_regressor.predict(x_test)
```

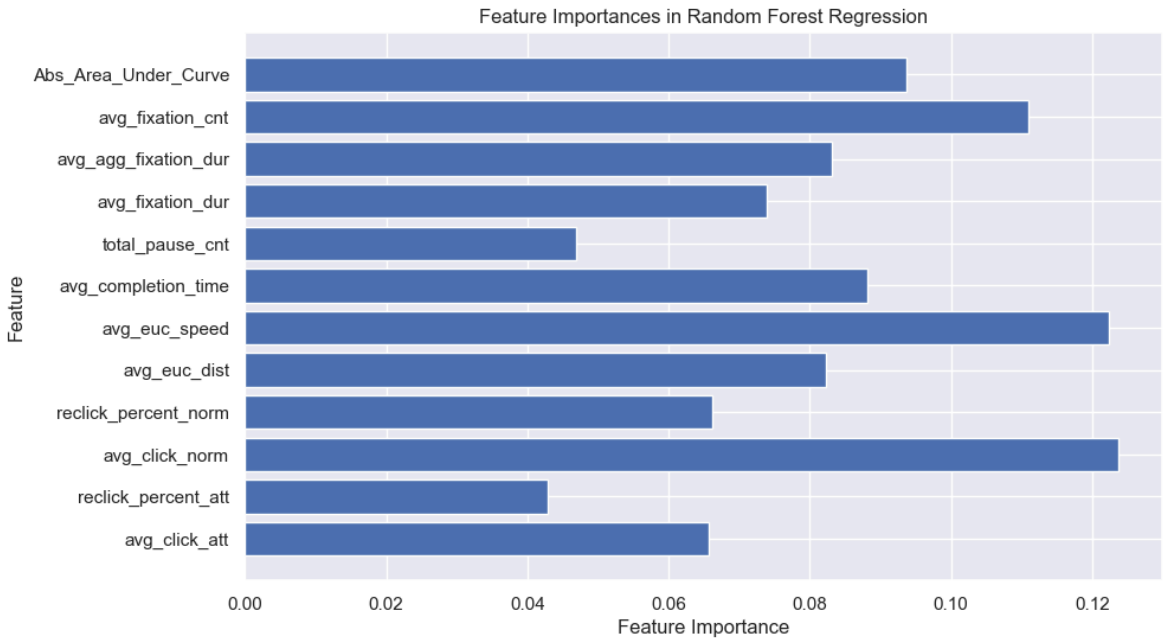
In [145]:
```
from sklearn.metrics import mean_squared_error, r2_score

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("R-squared:", r2)
```

```
Mean Squared Error: 15.334023022331158
R-squared: 0.004900066154089349
```

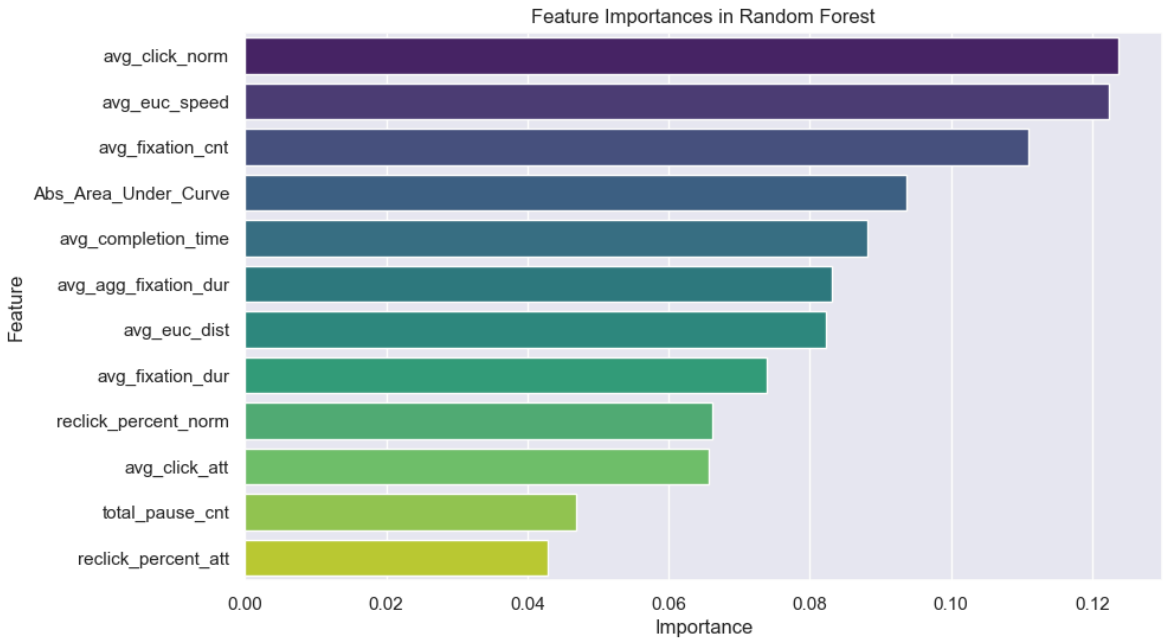
In [152]:
```
feature_importances = rf_regressor.feature_importances_

# Print feature importances
for i, feature_name in enumerate(x_ag.columns):
    print(f"{feature_name}: {feature_importances[i]}")
```

```
avg_click_att: 0.06568876546315759
reclick_percent_att: 0.0428065004457914
avg_click_norm: 0.12378119810369703
reclick_percent_norm: 0.06624725306747922
avg_euc_dist: 0.08223442475908807
avg_euc_speed: 0.12236079672437501
avg_completion_time: 0.08825033145823301
total_pause_cnt: 0.04697903543325511
avg_fixation_dur: 0.07384909911569341
avg_agg_fixation_dur: 0.08306289272066608
avg_fixation_cnt: 0.1109720376546877
Abs_Area_Under_Curve: 0.09376766505387654
```

In [153]:
```
import matplotlib.pyplot as plt

# Assuming feature_importances contains your feature importances
# Plot feature importances
plt.figure(figsize=(10, 6))
plt.barh(x_ag.columns, feature_importances)
plt.xlabel('Feature Importance')
plt.ylabel('Feature')
plt.title('Feature Importances in Random Forest Regression')
plt.show()
```



In [156]:
```
# Define the column names you want to use as features
selected_columns = ['avg_click_att', 'reclick_percent_att', 'avg_click_norm',
          'avg_euc_speed', 'avg_completion_time', 'total_pause_cnt', 'avg_fixa
          'avg_fixation_cnt','Abs_Area_Under_Curve']  # Replace with your actu

# Create a DataFrame with selected column names and importances
importance_df = pd.DataFrame({'Feature': selected_columns, 'Importance': featu

# Sort the DataFrame by importance in descending order
importance_df = importance_df.sort_values(by='Importance', ascending=False)

# Plot the bar graph
plt.figure(figsize=(10, 6))
sns.barplot(data=importance_df, x='Importance', y='Feature', palette='viridis
plt.title('Feature Importances in Random Forest')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.show()
```



Random Forest Regression Analysis for predicting Abs_area_under_curve

In [1]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
```

In [5]:
```
click_data = pd.read_csv('personality_mouse_final_table (2).csv')
```

In [6]:
```
x_cols_only = ['avg_click_att', 'reclick_percent_att', 'avg_click_norm', 'rec
          'avg_euc_speed', 'avg_completion_time', 'total_pause_cnt', 'avg_fix
          'avg_fixation_cnt']

y_cols_only = ['Abs_Area_Under_Curve']

x_only = click_data[x_cols_only]
y_only = click_data[y_cols_only]
```

In [7]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_only ,y_only, train_siz
```

In [8]:
```
fit_rf = RandomForestRegressor(random_state = 21)
```

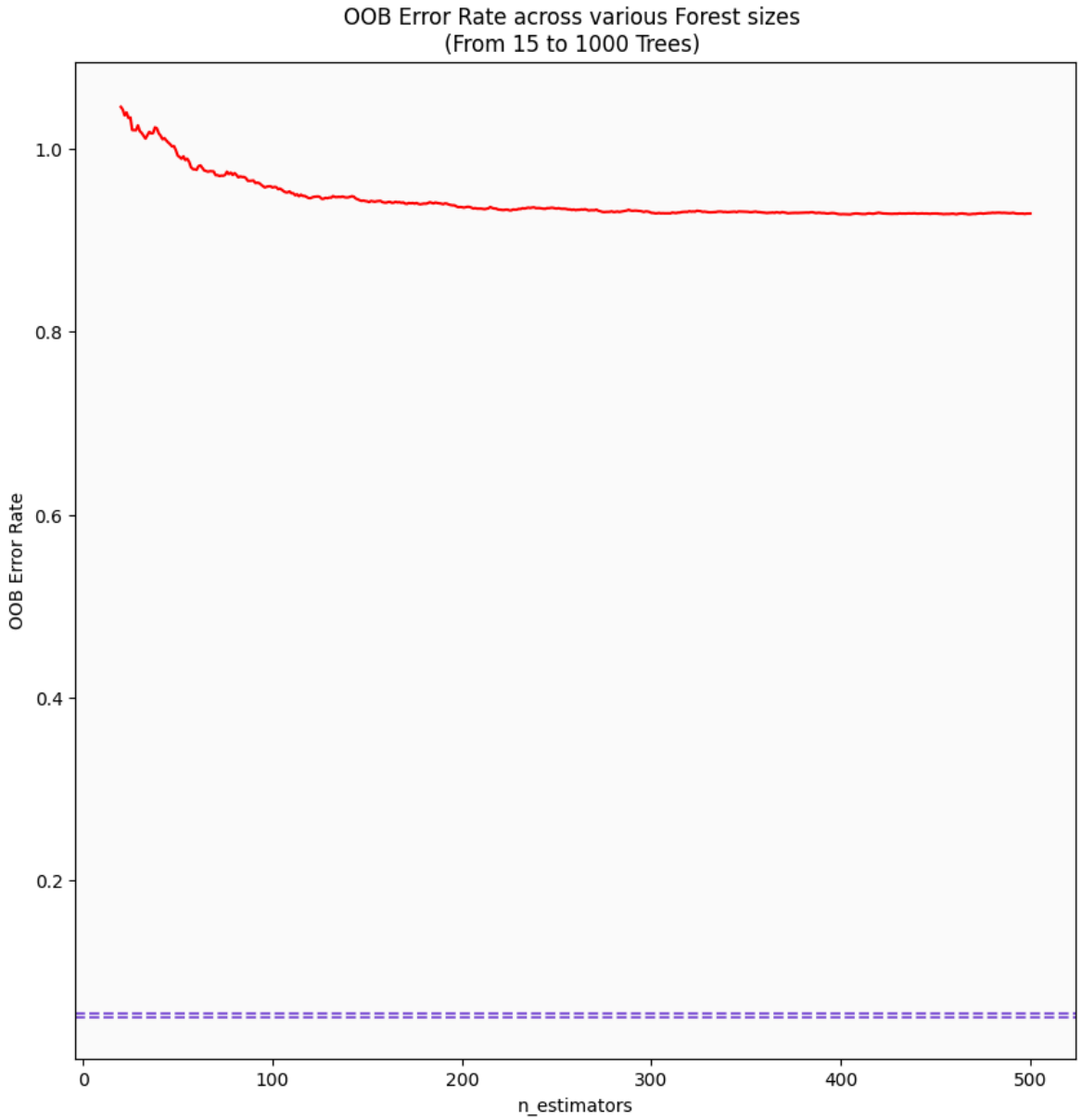
In [9]:
```
y_train = y_train.values.ravel()
y_test = y_test.values.ravel()
```

In [10]:
```
'''OOB Rate'''
fit_rf.set_params(warm_start = True, oob_score = True)

min_estimators = 20
max_estimators = 500

error_rate = {}

for i in range(min_estimators,max_estimators +1):
    fit_rf.set_params(n_estimators = i)
    fit_rf.fit(x_train, y_train)#fitting into x_test & y_test

    oob_error = 1 - fit_rf.oob_score_
    error_rate[i] = oob_error
```

In [11]:
```
oob_series = pd.Series(error_rate)
```

In [12]:
```
fig,ax = plt.subplots(figsize = (10,10))

ax.set_facecolor('#fafafa')

oob_series.plot(kind = 'line', color = 'red')
plt.axhline(0.055, color = '#875FDB',linestyle = '--')
plt.axhline(0.05, color = '#875FDB',linestyle = '--')
plt.xlabel('n_estimators')
plt.ylabel('OOB Error Rate')
plt.title('OOB Error Rate across various Forest sizes \n(From 15 to 1000 Trees
```

Out[12]:
```
Text(0.5, 1.0, 'OOB Error Rate across various Forest sizes \n(From 15 to 100
0 Trees) ')
```



In [13]:
```
rf_regressor = RandomForestRegressor(n_estimators=300, random_state=21)

rf_regressor.fit(x_train, y_train)

y_pred = rf_regressor.predict(x_test)
```

In [14]:
```
from sklearn.metrics import mean_squared_error, r2_score

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("R-squared:", r2)
```

```
Mean Squared Error: 0.011316415895130251
R-squared: 0.16562424034018985
```

Random Forest Regression Analysis predicting attentiveness is giving R2 = 0.16 and Mean Squared Error - 0.011