EDA Theory DA

Sanjit Besthamalla 21BDS0004

Github Link : [Sanjit1806/EDA_DA-21BDS0004]

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.decomposition import PCA
from scipy.spatial.distance import pdist
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import seaborn as sns
```

```
df = pd.read_csv('opt.csv')
df.head()
```

	rownames	PID	Clinic	Group	Age	Black	White	Nat.Am	Asian	Hisp	...
0	1	100034	NY	C	25	Yes	No	No	No		...
1	2	100042	NY	C	21	Yes	No	No	No		...
2	3	100067	NY	T	25	No	Yes	No	No	Yes	...
3	4	100083	NY	C	36	Yes	No	No	No		...
4	5	100091	NY	C	21	No	Yes	No	No	Yes	...

5 rows × 172 columns

```
print("Number of rows and columns:", df.shape)
```

```
Number of rows and columns: (823, 172)
```

```
print("Header names:", df.columns.tolist())
```

```
Header names: ['rownames', 'PID', 'Clinic', 'Group', 'Age', 'Black', 'White',
```

```
print("Structure of the dataset:", df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 823 entries, 0 to 822
Columns: 172 entries, rownames to V5..S7
dtypes: float64(86), int64(13), object(73)
memory usage: 1.1+ MB
```

```
Structure of the dataset: None
```

```
print(df.dtypes)
```

```
rownames      int64
PID           int64
Clinic       object
Group        object
Age           int64
             ...
V5..TF      float64
V5..PI      float64
V5..CR      float64
V5..FN      float64
V5..S7      float64
Length: 172, dtype: object
```

```
df.tail(10)
```

	rownames	PID	Clinic	Group	Age	Black	White	Nat.Am	Asian	Hisp	...
813	814	402360	MS	T	25	Yes	No	No	No	No	...
814	815	402378	MS	C	22	Yes	No	No	No	No	...
815	816	402386	MS	T	19	Yes	No	No	No	No	...
816	817	402394	MS	T	17	Yes	No	No	No	No	...
817	818	402410	MS	C	20	Yes	No	No	No	No	...
818	819	402428	MS	T	22	Yes	No	No	No	No	...
819	820	402436	MS	C	23	Yes	No	No	No	No	...
820	821	402451	MS	C	29	Yes	No	No	No	No	...
821	822	402469	MS	T	20	Yes	No	No	No	No	...
822	823	402477	MS	C	24	Yes	No	No	No	No	...

10 rows × 172 columns

```
df.describe()
```

	rownames	PID	Age	BMI	BL.Cig.Day	BL.Drks.Day	N.
count	823.00000	823.000000	823.000000	750.000000	92.000000	13.000000	
mean	412.00000	252541.347509	25.978129	27.669333	8.717391	4.461538	
std	237.72393	106737.056578	5.565973	7.127299	6.271032	9.386297	
min	1.00000	100034.000000	16.000000	15.000000	1.000000	0.000000	
25%	206.50000	200501.000000	22.000000	23.000000	5.000000	0.000000	
50%	412.00000	202717.000000	25.000000	26.000000	7.500000	1.000000	
75%	617.50000	302208.000000	30.000000	31.000000	10.000000	5.000000	
max	823.00000	402477.000000	44.000000	68.000000	30.000000	35.000000	

8 rows × 99 columns

Data Cleaning

```
print(df.isnull().sum())
```

```
rownames      0
PID           0
Clinic        0
Group         0
Age           0
           ...
V5..TF      498
V5..PI      498
V5..CR      498
V5..FN      498
V5..S7      498
Length: 172, dtype: int64
```

```
# Fill missing values with mean
for column in df.columns:
  if df[column].dtype in ['int64', 'float64']:
    df[column].fillna(df[column].mean(), inplace=True)

print(df.isnull().sum())
```

```
rownames    0
PID         0
Clinic      0
Group       0
Age         0
           ..
V5..TF      0
V5..PI      0
V5..CR      0
V5..FN      0
V5..S7      0
```

```
Length: 172, dtype: int64
<ipython-input-122-c41cd969e4b0>:4: FutureWarning: A value is trying to be se
The behavior will change in pandas 3.0. This inplace method will never work b

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.

  df[column].fillna(df[column].mean(), inplace=True)
```

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
df['Black_enc'] = le.fit_transform(df['Black'])
df['White_enc'] = le.fit_transform(df['White'])
df['Asian_enc'] = le.fit_transform(df['Asian'])
df['Hisp_enc'] = le.fit_transform(df['Hisp'])

df.head()
```

	rownames	PID	Clinic	Group	Age	Black	White	Nat.Am	Asian	Hisp	...
0	1	100034	NY	C	25	Yes	No	No	No		...
1	2	100042	NY	C	21	Yes	No	No	No		...
2	3	100067	NY	T	25	No	Yes	No	No	Yes	...
3	4	100083	NY	C	36	Yes	No	No	No		...
4	5	100091	NY	C	21	No	Yes	No	No	Yes	...

5 rows × 176 columns

```
df.head()
```

	rownames	PID	Clinic	Group	Age	Black	White	Nat.Am	Asian	Hisp	...
0	1	100034	NY	C	25	Yes	No	No	No		...
1	2	100042	NY	C	21	Yes	No	No	No		...
2	3	100067	NY	T	25	No	Yes	No	No	Yes	...
3	4	100083	NY	C	36	Yes	No	No	No		...
4	5	100091	NY	C	21	No	Yes	No	No	Yes	...

5 rows × 176 columns

```
print("Mean of age:", df['Age'].mean())
print("Median of age:", df['Age'].median())
print("Mode of age:", df['Age'].mode()[0])
```

```
Mean of age: 25.97812879708384
Median of age: 25.0
```

```
Mode of age: 23
```

```
print("Standard deviation of age:", df['Age'].std())
print("Variance of age:", df['Age'].var())
print("Range of age:", df['Age'].max() - df['Age'].min())
```

```
Standard deviation of age: 5.5659730818927455
Variance of age: 30.98005634835463
Range of age: 28
```

```
print("Quartile ranges of age:")
print(df['Age'].quantile([0.25, 0.5, 0.75]))
iqr = df['Age'].quantile(0.75) - df['Age'].quantile(0.25)
print("IQR of age:", iqr)
```

```
Quartile ranges of age:
0.25    22.0
0.50    25.0
0.75    30.0
Name: Age, dtype: float64
IQR of age: 8.0
```

```
df.describe()
```

	rownames	PID	Age	BMI	BL.Cig.Day	BL.Drks.Day	N.
count	823.00000	823.000000	823.000000	823.000000	823.000000	823.000000	
mean	412.00000	252541.347509	25.978129	27.669333	8.717391	4.461538	
std	237.72393	106737.056578	5.565973	6.803462	2.086526	1.134094	
min	1.00000	100034.000000	16.000000	15.000000	1.000000	0.000000	
25%	206.50000	200501.000000	22.000000	23.000000	8.717391	4.461538	
50%	412.00000	202717.000000	25.000000	27.000000	8.717391	4.461538	
75%	617.50000	302208.000000	30.000000	30.000000	8.717391	4.461538	
max	823.00000	402477.000000	44.000000	68.000000	30.000000	35.000000	

8 rows × 103 columns

```
correlation_matrix = df[['Age', 'BMI']].corr()  # Replace 'feature1' and 'feature
print(correlation_matrix)
```

```
          Age       BMI
Age  1.000000  0.123381
BMI  0.123381  1.000000
```

```
corr = df['Age'].corr(df['BL.Cig.Day'])
print(f"Correlation between Age and Cig per day: {corr}")
```

```
Correlation between Age and Cig per day: 0.017079166666948153
```

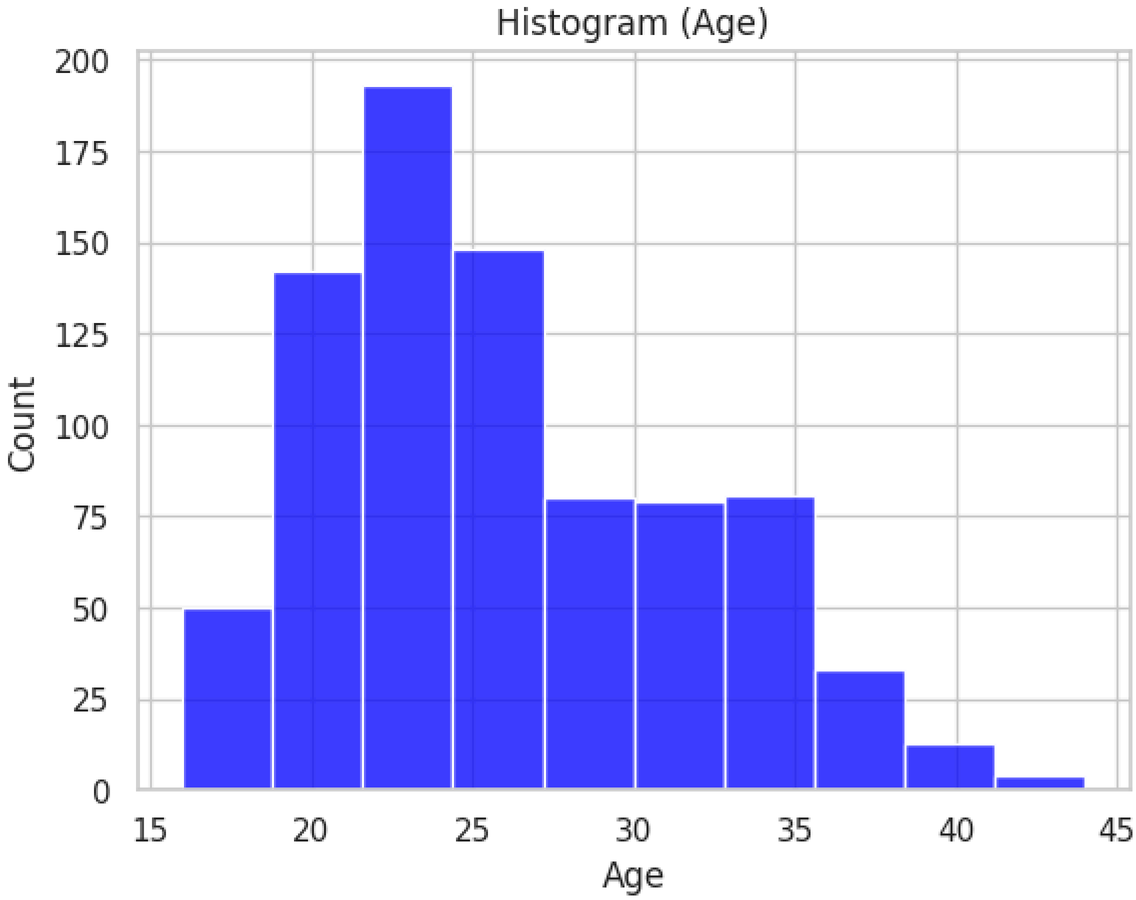
```
# rom google.colab import files

# df.to_csv('modOpt.csv', encoding = 'utf-8-sig')
# files.download('modOpt.csv')
```

```
selected_columns = ['Age', 'BL..FN', 'BL..S7', 'V5..AA', 'V5..PG', 'V5..TD']

plt.figure(figsize=(20, 25))
sns.set(style="whitegrid")
```

```
<Figure size 2000x2500 with 0 Axes>
```

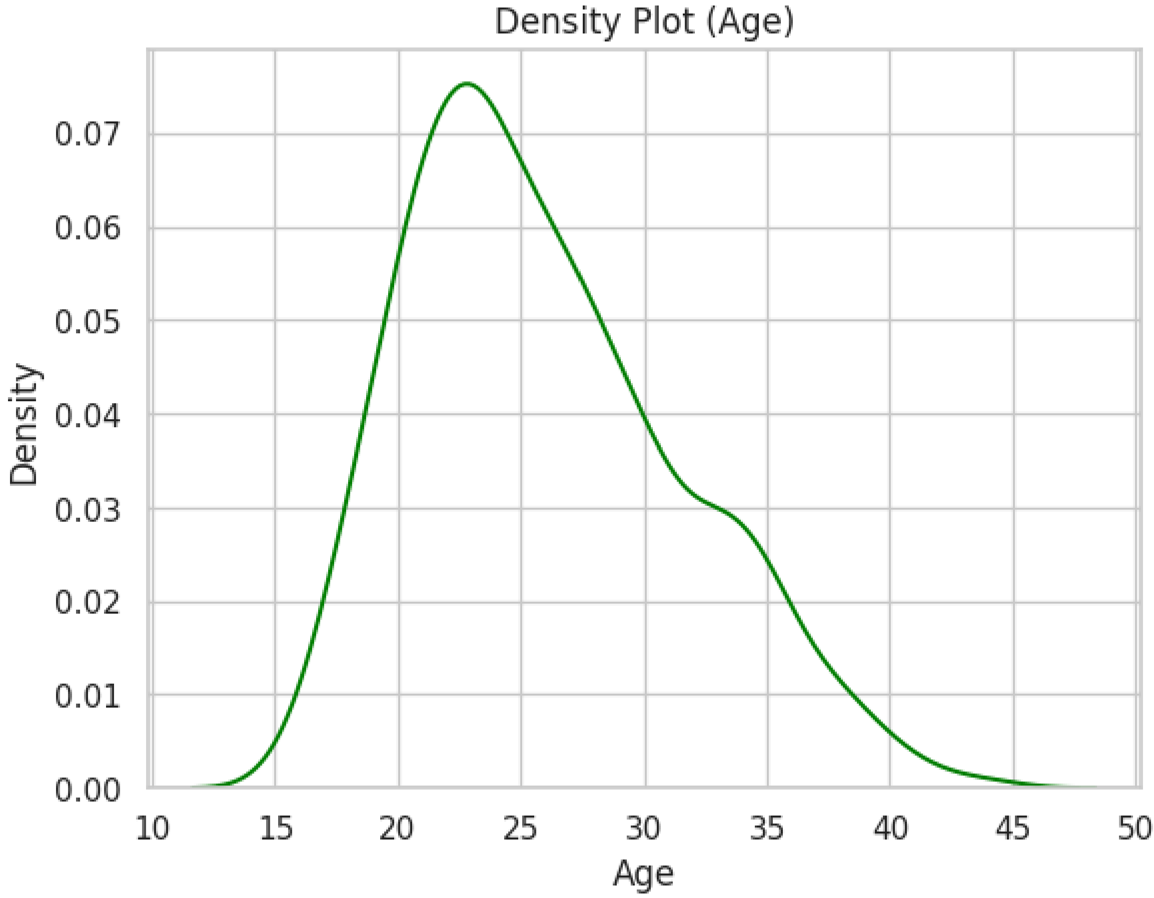
```
sns.histplot(df['Age'], kde=False, color='blue', bins=10)
plt.title('Histogram (Age)')
```

```
Text(0.5, 1.0, 'Histogram (Age)')
```

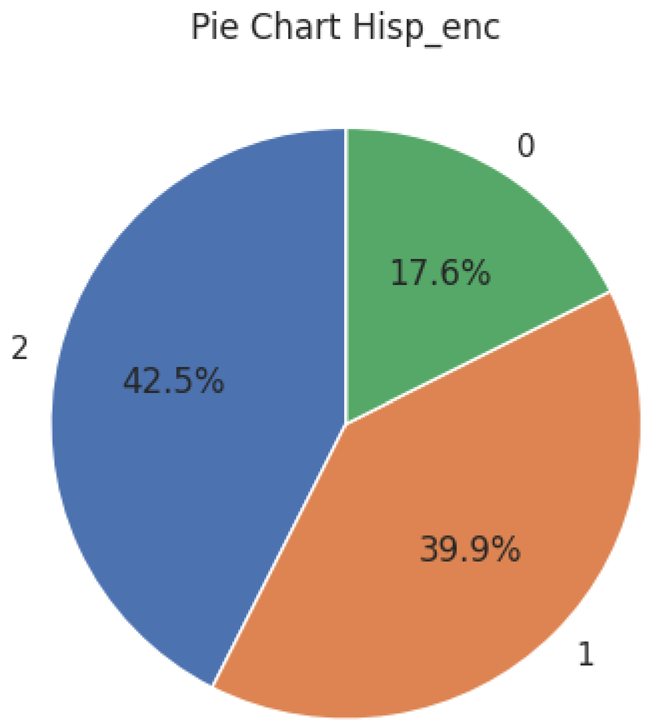



```
sns.kdeplot(df['Age'], color='green')
plt.title('Density Plot (Age)')
```

```
Text(0.5, 1.0, 'Density Plot (Age)')
```

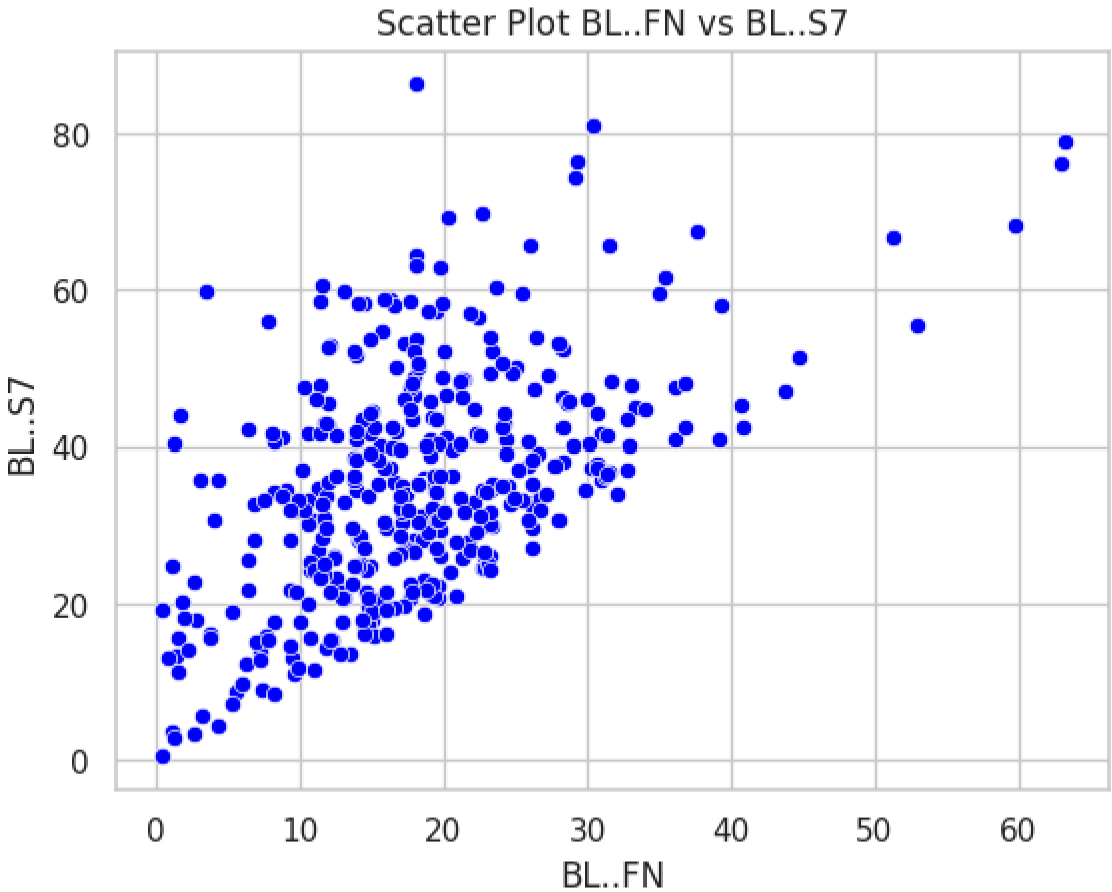



```
hisp_enc_counts = df['Hisp_enc'].value_counts()
plt.pie(hisp_enc_counts, labels=hisp_enc_counts.index, autopct='%1.1f%%', startan
plt.title('Pie Chart Hisp_enc')
plt.show()
```



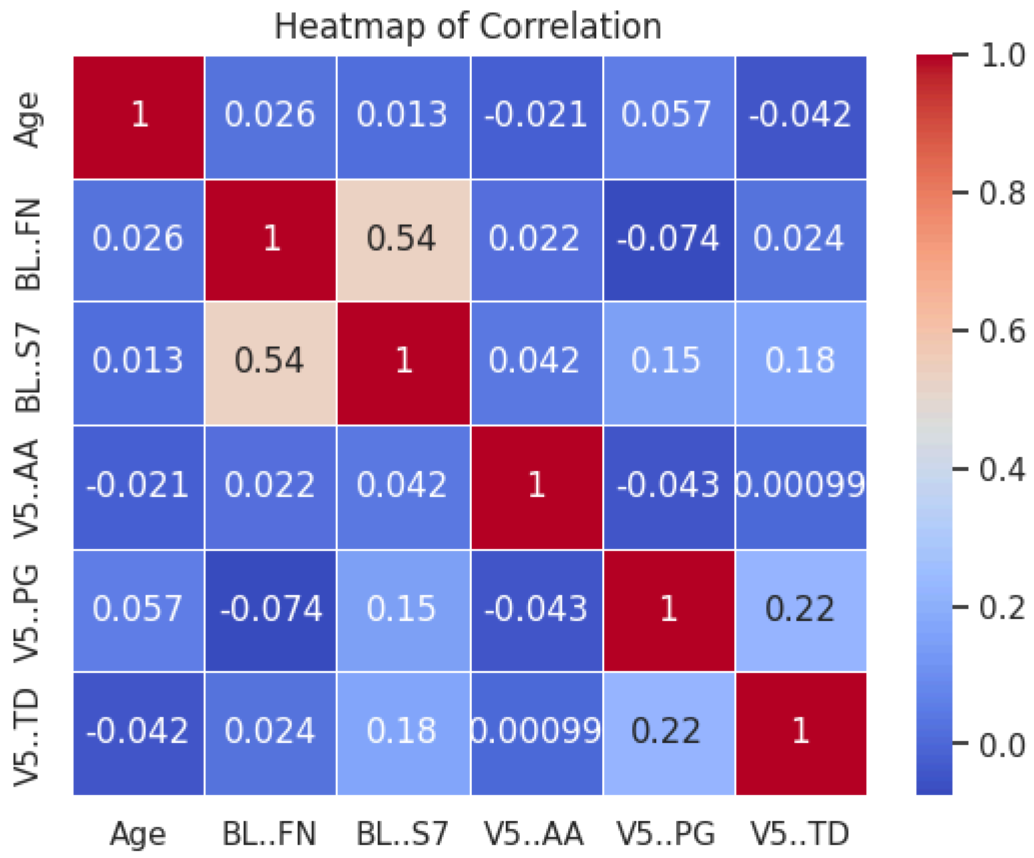
```
sns.scatterplot(x='BL..FN', y='BL..S7', data=df, color='blue')
plt.title('Scatter Plot BL..FN vs BL..S7')
```

```
Text(0.5, 1.0, 'Scatter Plot BL..FN vs BL..S7')
```



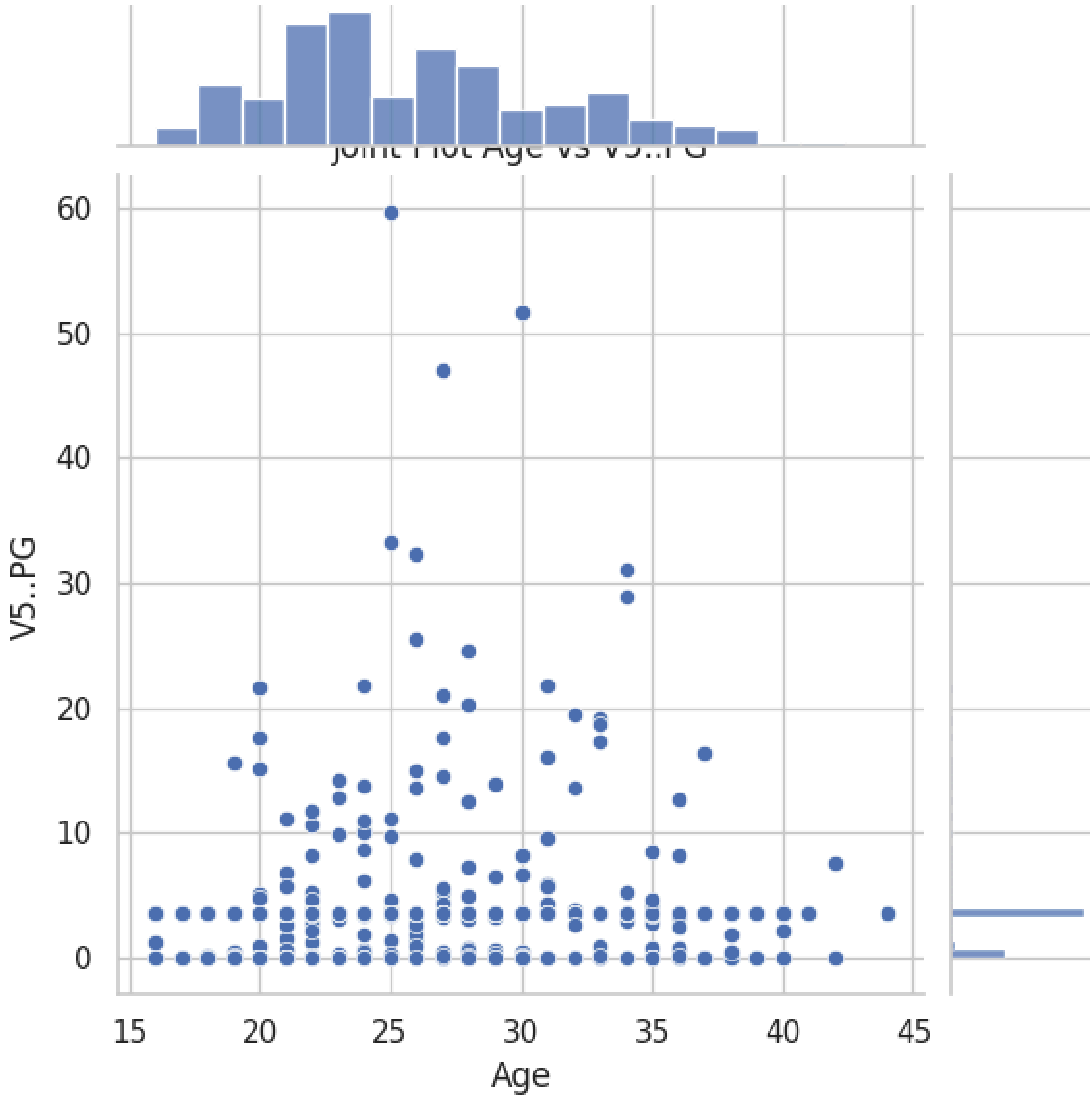

```
corr_matrix = df[['Age', 'BL..FN', 'BL..S7', 'V5..AA', 'V5..PG', 'V5..TD']].corr(
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Heatmap of Correlation')
```

```
Text(0.5, 1.0, 'Heatmap of Correlation')
```



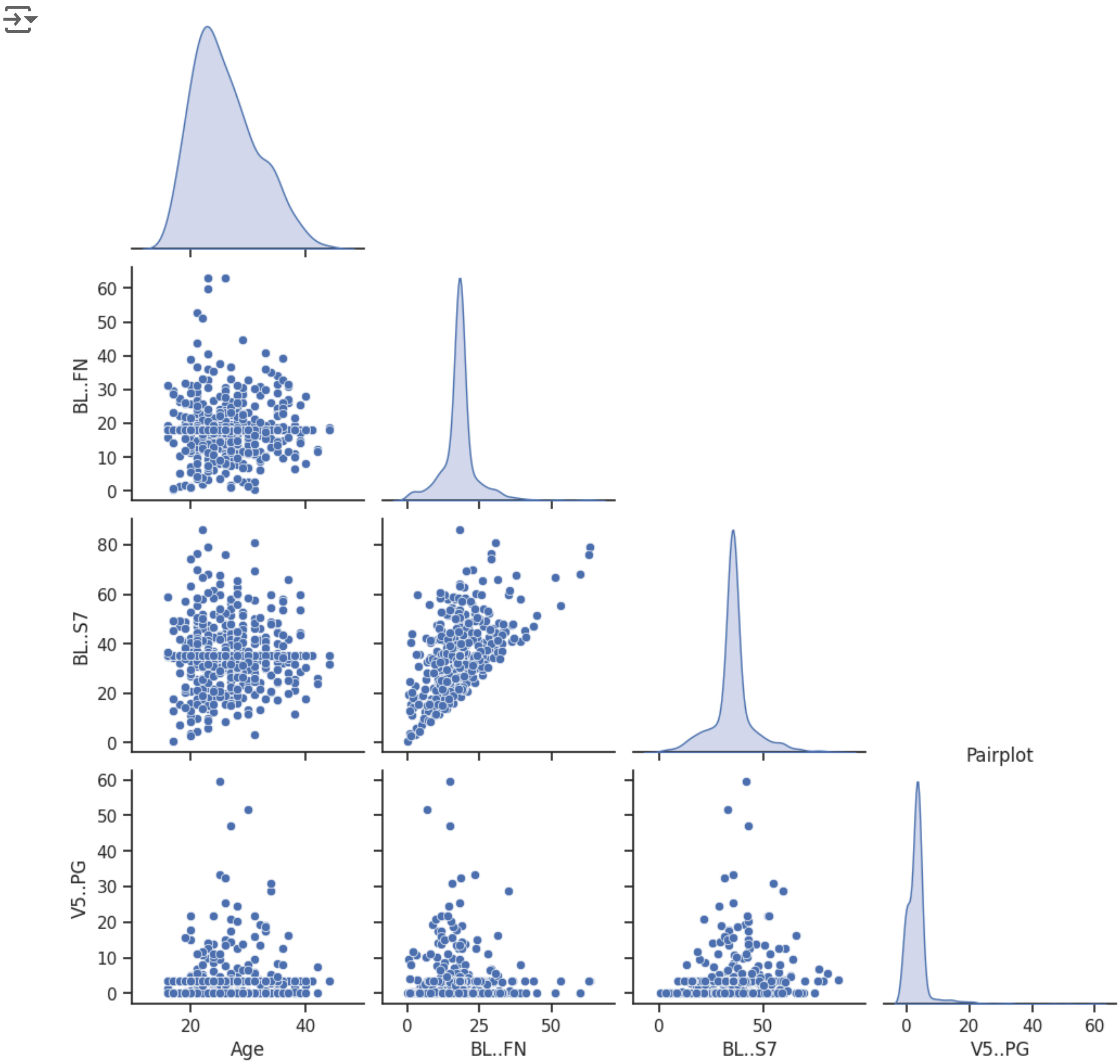

```
sns.jointplot(x='Age', y='V5..PG', data=df, kind='scatter', height=6)
plt.title('Joint Plot Age vs V5..PG')
```

```
Text(0.5, 1.0, 'Joint Plot Age vs V5..PG')
```

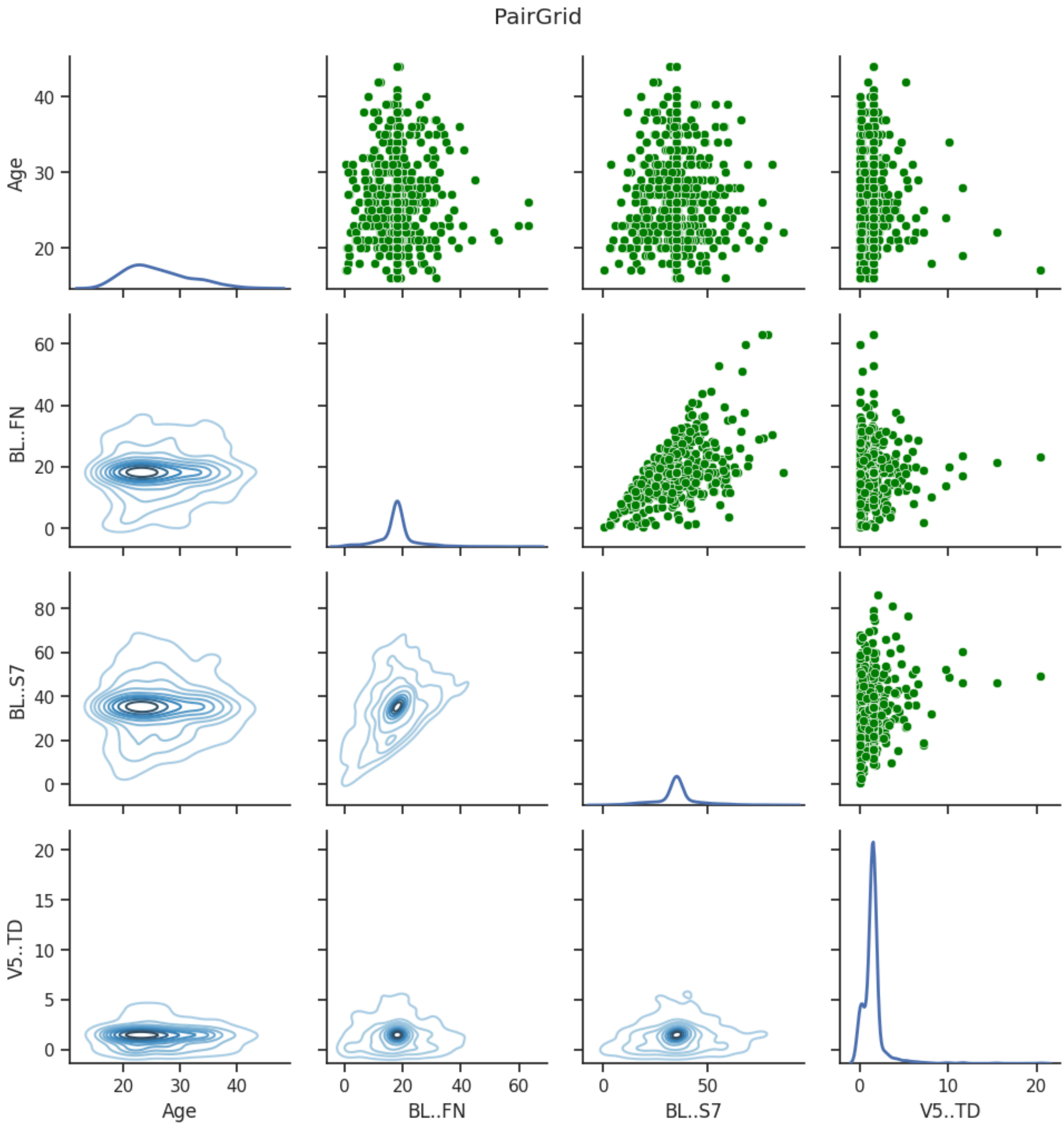



```
sns.set(style="ticks")
```

```
sns.pairplot(df[['Age', 'BL..FN', 'BL..S7', 'V5..PG']], diag_kind='kde', corner=T
plt.title('Pairplot')
plt.show()
```



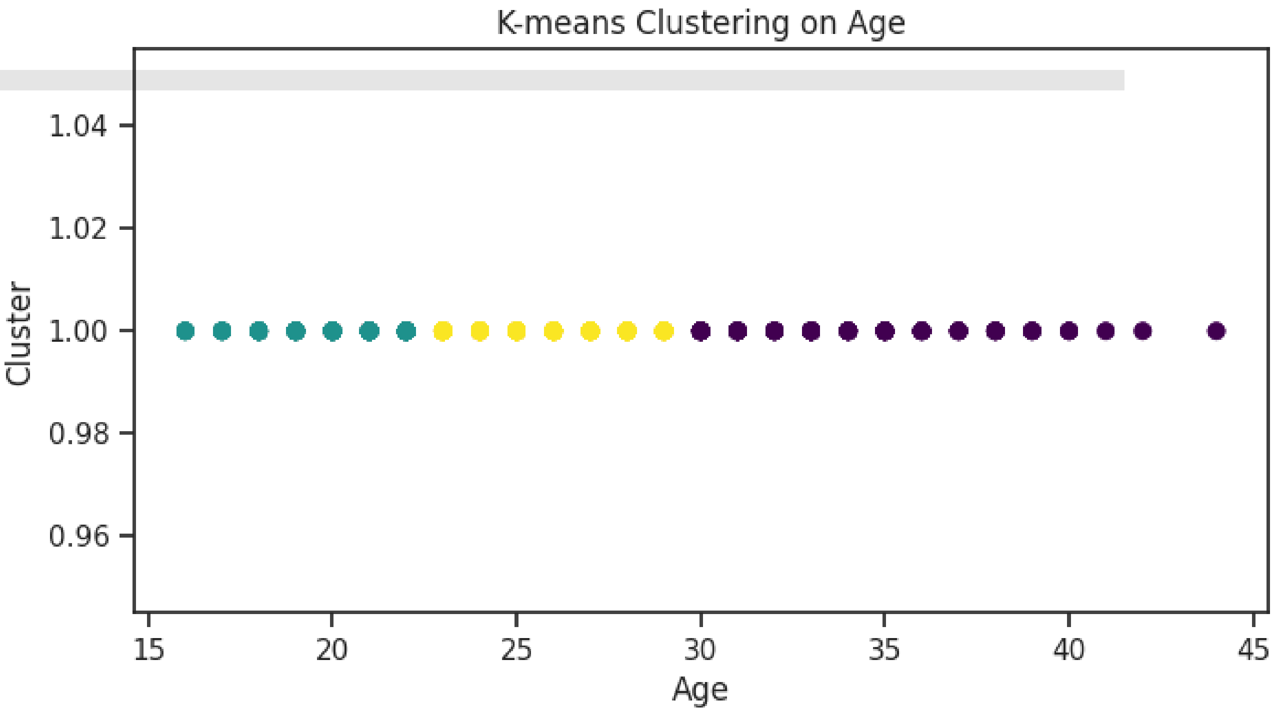

```
g = sns.PairGrid(df[['Age', 'BL..FN', 'BL..S7', 'V5..TD']])
g.map_upper(sns.scatterplot, color="green")
g.map_lower(sns.kdeplot, cmap="Blues_d")
g.map_diag(sns.kdeplot, lw=2)

plt.suptitle('PairGrid', y=1.02)
plt.show()
```



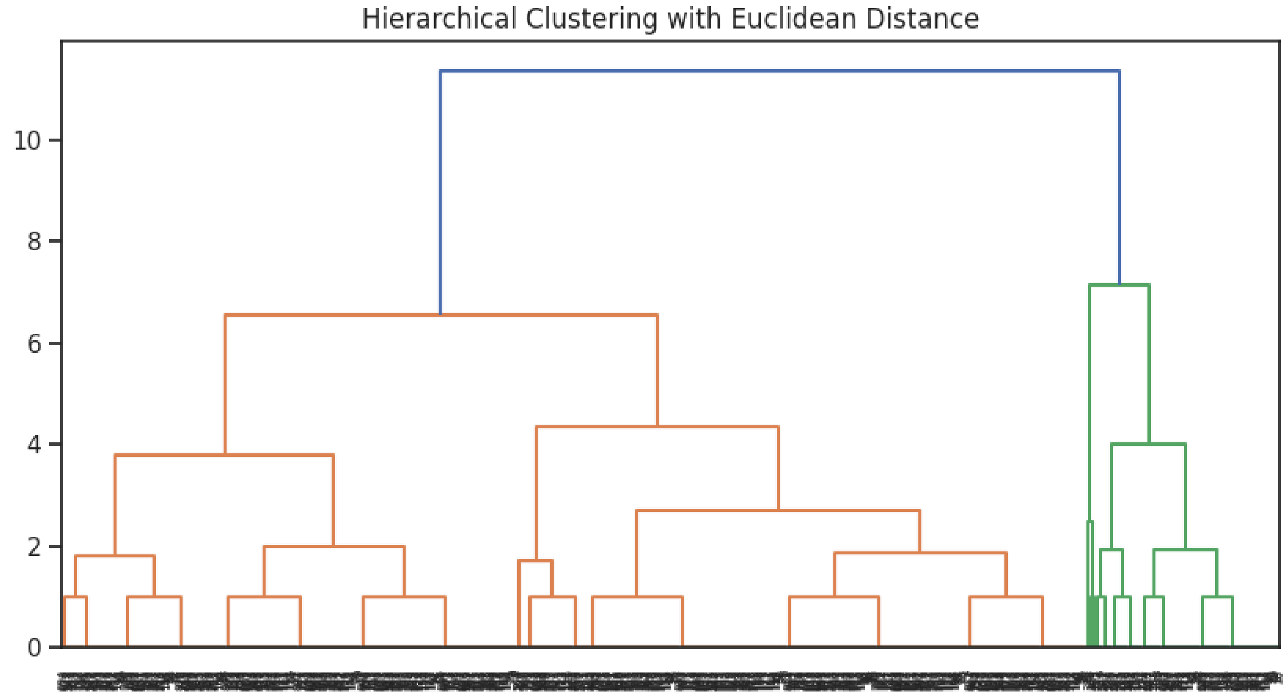

```
# K-means Clustering on "Age"
age_data = df[['Age']].dropna()  # Select 'Age' column and drop NA values

kmeans = KMeans(n_clusters=3, random_state=123)
age_data['Cluster'] = kmeans.fit_predict(age_data)
```

```
plt.figure(figsize=(8, 4))
plt.scatter(age_data['Age'], np.ones(len(age_data)), c=age_data['Cluster'], cmap=
plt.title("K-means Clustering on Age")
```

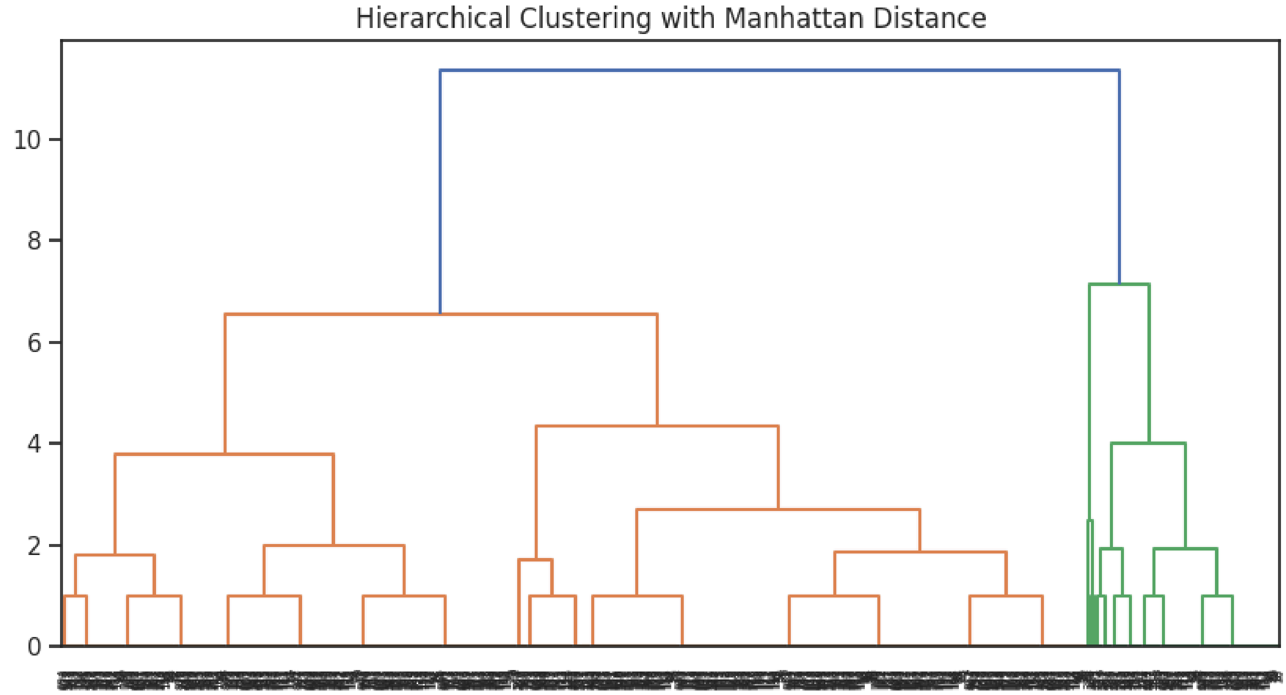
```
plt.xlabel("Age")
plt.ylabel("Cluster")
plt.show()
```



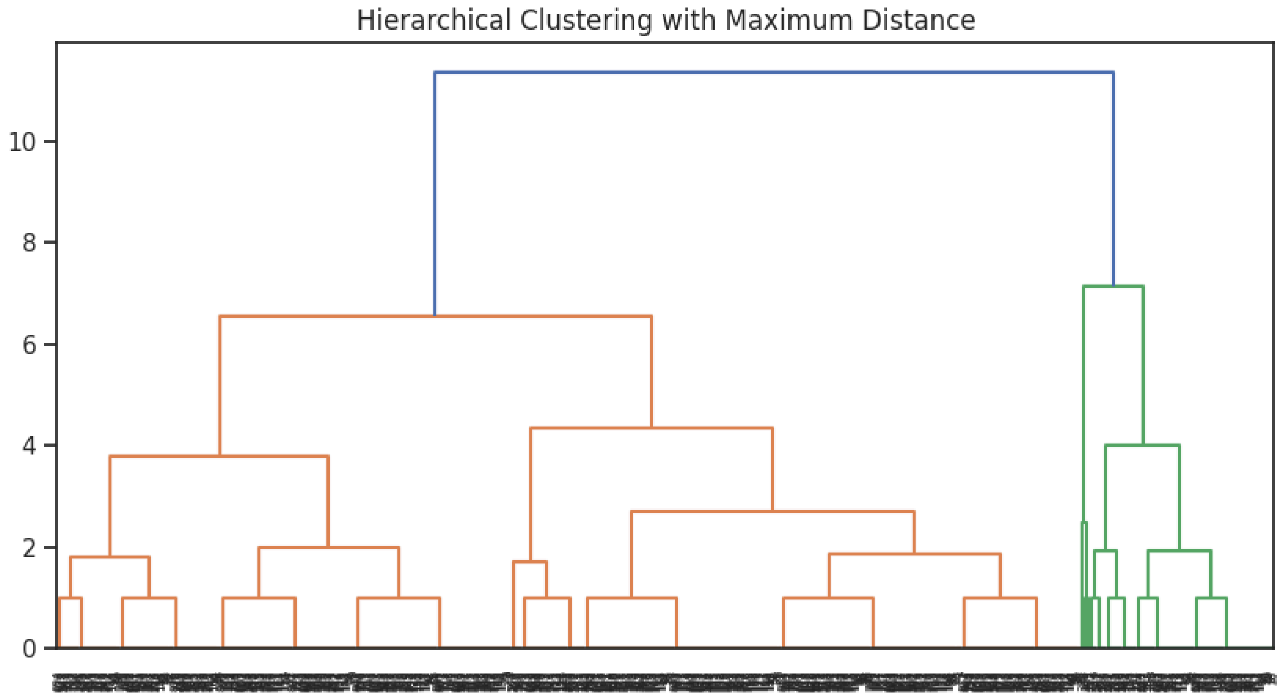

```
# Hierarchical Clustering
# Euclidean distance
euclidean_linkage = linkage(age_data[['Age']], method='average', metric='euclidea
plt.figure(figsize=(10, 5))
dendrogram(euclidean_linkage)
plt.title("Hierarchical Clustering with Euclidean Distance")
plt.show()
```




```
# Manhattan distance
manhattan_linkage = linkage(pdist(age_data[['Age']], metric='cityblock'), method=
plt.figure(figsize=(10, 5))
dendrogram(manhattan_linkage)
plt.title("Hierarchical Clustering with Manhattan Distance")
plt.show()
```



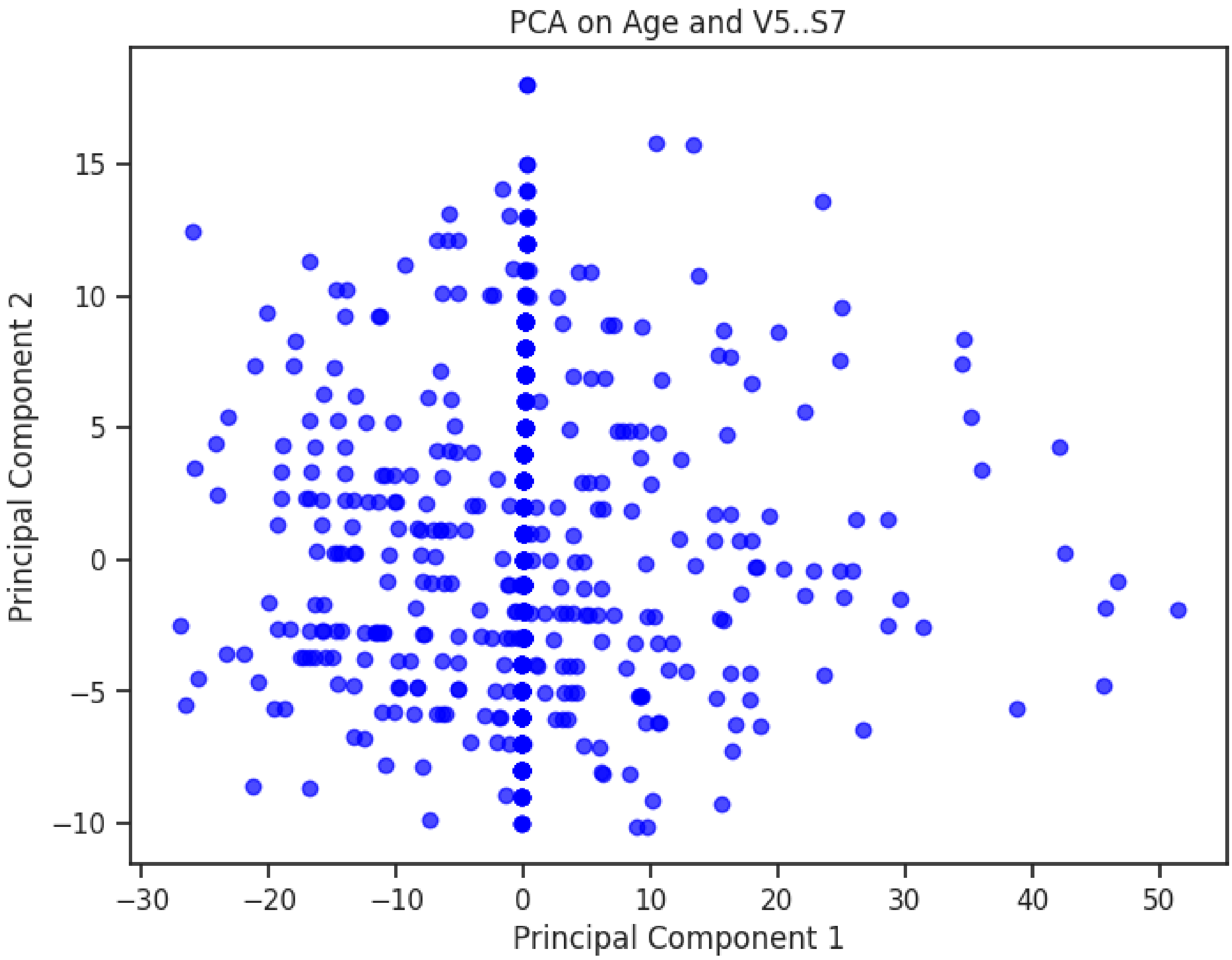
```
# Maximum distance
maximum_linkage = linkage(pdist(age_data[['Age']], metric='chebyshev'), method='a
plt.figure(figsize=(10, 5))
dendrogram(maximum_linkage)
plt.title("Hierarchical Clustering with Maximum Distance")
plt.show()
```




```
# Principal Component Analysis (PCA) on "Age" and "V5..S7"
pca_data = df[['Age', 'V5..S7']].dropna()

pca = PCA(n_components=2)
pca_result = pca.fit_transform(pca_data)

print("Explained variance by each component:", pca.explained_variance_ratio_)
```

```
Explained variance by each component: [0.73236915 0.26763085]
```

```
plt.figure(figsize=(8, 6))
plt.scatter(pca_result[:, 0], pca_result[:, 1], color='blue', alpha=0.7)
plt.title("PCA on Age and V5..S7")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.show()
```




```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
```

```
X = df[['Age']]
y = df['V5..S7']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_s
```

```
model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
```

```
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: 82.36517283690578
R-squared: -0.012183390170729558
```

```
plt.figure(figsize=(8, 6))
plt.scatter(X_test, y_test, color='blue', label='Actual data')
plt.plot(X_test, y_pred, color='red', linewidth=2, label='Regression line')
plt.title("Linear Regression of V5..S7 on Age")
plt.xlabel("Age")
plt.ylabel("V5..S7")
plt.legend()
plt.show()
```


```
X2 = df[['Age']]
y2 = df['V5..AA']

X2_train, X2_test, y2_train, y2_test = train_test_split(X2, y2, test_size=0.2, ra
```

```
model = LinearRegression()
model.fit(X2_train, y2_train)

y2_pred = model.predict(X2_test)
```

```
mse2 = mean_squared_error(y2_test, y2_pred)
r2_2 = r2_score(y2_test, y2_pred)
```

```
print("Model 2 - Mean Squared Error:", mse2
print("Model 2 - R-squared:", r2_2)
```

```
Model 2 - Mean Squared Error: 1.9287966187709857
Model 2 - R-squared: -0.006013851818570393
```

```
plt.figure(figsize=(8, 6))
plt.scatter(X2_test, y2_test, color='blue', label='Actual data')
plt.plot(X2_test, y2_pred, color='red', linewidth=2, label='Regression line')
plt.title("Linear Regression of V5..AA on Age")
plt.xlabel("Age")
plt.ylabel("V5..AA")
plt.legend()
plt.show()
```