Pridicting Heart Disease by using Principal Component Analysis (PCA)

About Dataset

The dataset contains the following features:

- age(in years)
- sex: (1 = male; 0 = female)
- cp: chest pain type
- trestbps: resting blood pressure (in mm Hg on admission to the hospital)
- chol: serum cholestoral in mg/dl
- fbs: (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
- restecg: resting electrocardiographic results
- thalach: maximum heart rate achieved
- exang: exercise induced angina (1 = yes; 0 = no)
- oldpeak: ST depression induced by exercise relative to rest
- slope: the slope of the peak exercise ST segment
- ca: number of major vessels (0-3) colored by flourosopy
- thal: 3 = normal; 6 = fixed defect; 7 = reversable defect
- target: 1 or 0

Expected Outcome from the project

1. Statistical analysis of the data
2. Univariate and bivariate analysis
3. Find the correlation
4. Perform PCA and find the variance from 2 components

In [7]:
```
# Importing all the necessary Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
from scipy import stats
import warnings
warnings.filterwarnings('ignore')
from IPython.display import display
```

In [8]:
```
# Read the dataset
df = pd.read_csv('heart.csv',encoding='ISO-8859-1')
df
```

Out[8]:

	ï»¿age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	targ
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	

303 rows × 14 columns

In [9]:
```
df.isnull().any()
```

Out[9]:
```
ï»¿age      False
sex         False
cp          False
trestbps    False
chol        False
fbs         False
restecg     False
thalach     False
exang       False
oldpeak     False
slope       False
ca          False
thal        False
target      False
dtype: bool
```

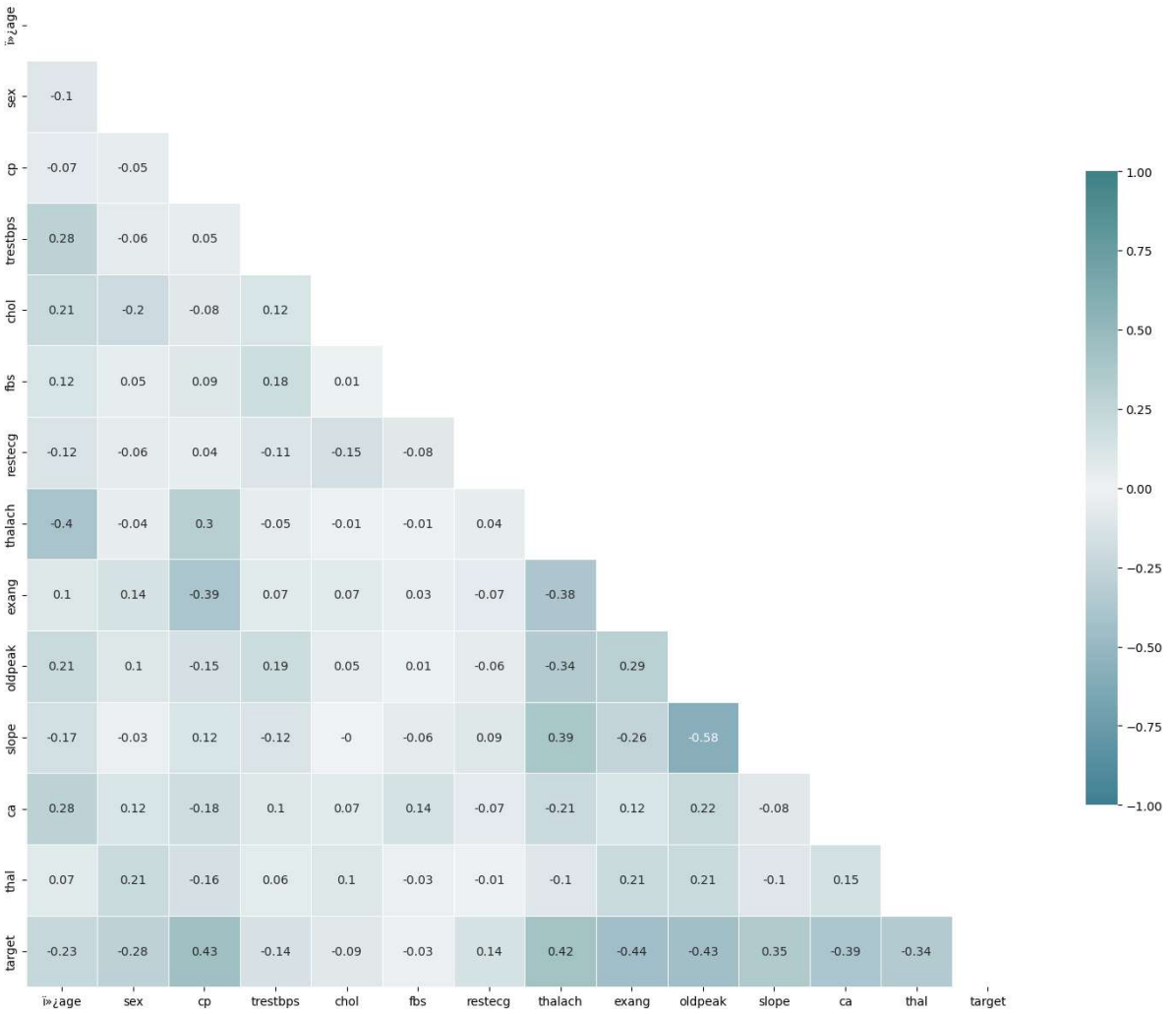
In [10]:
```
df.describe()
```

Out[10]:

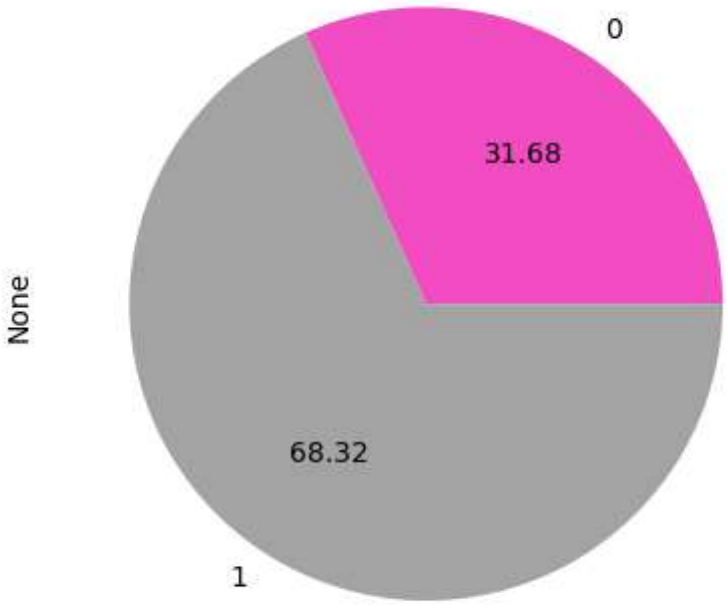
	ï»¿age	sex	cp	trestbps	chol	fbs	restecg	thalach
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000

In [11]:
```
corr = df.corr().round(2)
mask = np.zeros_like(corr, dtype=np.bool)
```

```
mask[np.triu_indices_from(mask)] = True
f1, ax1 = plt.subplots(figsize=(15, 15))
cmap = sns.diverging_palette(220, 200, as_cmap=True)
sns.heatmap(corr, mask=mask, cmap=cmap, vmin=-1, vmax=1, center=0, square=True, line
plt.tight_layout()
plt.show()
```



In [12]:
```
# Distribution of Sex
df.groupby('sex').size().plot(kind='pie', autopct='%.2f',colors=sns.color_palette('b
plt.show()
```



Statistical analysis of the data

- Mean
- Median
- Mode
- Standard Deviation

In [13]:
```
df.mean()
```

Out[13]:
```
ï»¿age       54.366337
sex          0.683168
cp           0.966997
trestbps   131.623762
chol       246.264026
fbs          0.148515
restecg      0.528053
thalach    149.646865
exang        0.326733
oldpeak      1.039604
slope        1.399340
ca           0.729373
thal         2.313531
target       0.544554
dtype: float64
```

In [14]:
```
df.median()
```

Out[14]:
```
ï»¿age      55.0
sex          1.0
cp           1.0
trestbps   130.0
chol       240.0
fbs          0.0
restecg      1.0
thalach    153.0
exang        0.0
oldpeak      0.8
slope        1.0
ca           0.0
thal         2.0
target       1.0
dtype: float64
```

In [15]:
```
df.mode()
```

Out[15]:

	ï»¿age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	58.0	1.0	0.0	120.0	197	0.0	1.0	162.0	0.0	0.0	2.0	0.0	2.0
1	NaN	NaN	NaN	NaN	204	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	234	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [16]:
```
df.min()
```

Out[16]:
```
ï»¿age      29.0
sex          0.0
cp           0.0
trestbps    94.0
chol       126.0
fbs          0.0
restecg      0.0
thalach     71.0
exang        0.0
oldpeak      0.0
slope        0.0
ca           0.0
thal         0.0
target       0.0
dtype: float64
```

In [17]:
```
print("Sample variance: ", stats.tstd(df)**2)
```

```
Sample variance:  [8.24845584e+01 2.17166087e-01 1.06513234e+00 3.07586453e+02
 2.68642675e+03 1.26876926e-01 2.76528315e-01 5.24646406e+02
 2.20706839e-01 1.34809521e+00 3.79734662e-01 1.04572378e+00
 3.74882521e-01 2.48836142e-01]
```

In [18]:
```
print("Sample standard deviation: ", stats.tstd(df))
```

```
Sample standard deviation:  [ 9.08210099  0.46601082  1.03205249 17.53814281 51.8307
5099  0.35619787
  0.5258596  22.90516111  0.46979446  1.16107502  0.61622615  1.02260636
  0.61227651  0.49883478]
```

Standardization and Normalization

In [19]:
```
from sklearn.preprocessing import StandardScaler

ss = StandardScaler()
df.iloc[:,1:-1] = ss.fit_transform(df.iloc[:,1:-1])
df
```

Out[19]:

	ï»¿age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	o
0	63	0.681005	1.973123	0.763956	-0.256334	2.394438	-1.005832	0.015443	-0.696631	1.
1	37	0.681005	1.002577	-0.092738	0.072199	-0.417635	0.898962	1.633471	-0.696631	2.
2	41	-1.468418	0.032031	-0.092738	-0.816773	-0.417635	-1.005832	0.977514	-0.696631	0.
3	56	0.681005	0.032031	-0.663867	-0.198357	-0.417635	0.898962	1.239897	-0.696631	-0.
4	57	-1.468418	-0.938515	-0.663867	2.082050	-0.417635	0.898962	0.583939	1.435481	-0.
...	...	...	...	...	...	...	...	...	...	...
298	57	-1.468418	-0.938515	0.478391	-0.101730	-0.417635	0.898962	-1.165281	1.435481	-0.
299	45	0.681005	1.973123	-1.234996	0.342756	-0.417635	0.898962	-0.771706	-0.696631	0.
300	68	0.681005	-0.938515	0.706843	-1.029353	2.394438	0.898962	-0.378132	-0.696631	2.
301	57	0.681005	-0.938515	-0.092738	-2.227533	-0.417635	0.898962	-1.515125	1.435481	0.
302	57	-1.468418	0.032031	-0.092738	-0.198357	-0.417635	-1.005832	1.064975	-0.696631	-0.

303 rows × 14 columns

In [20]:
```
from sklearn.preprocessing import Normalizer

norm = Normalizer()
df.iloc[:,1:-1] = norm.fit_transform(df.iloc[:,1:-1])
df
```

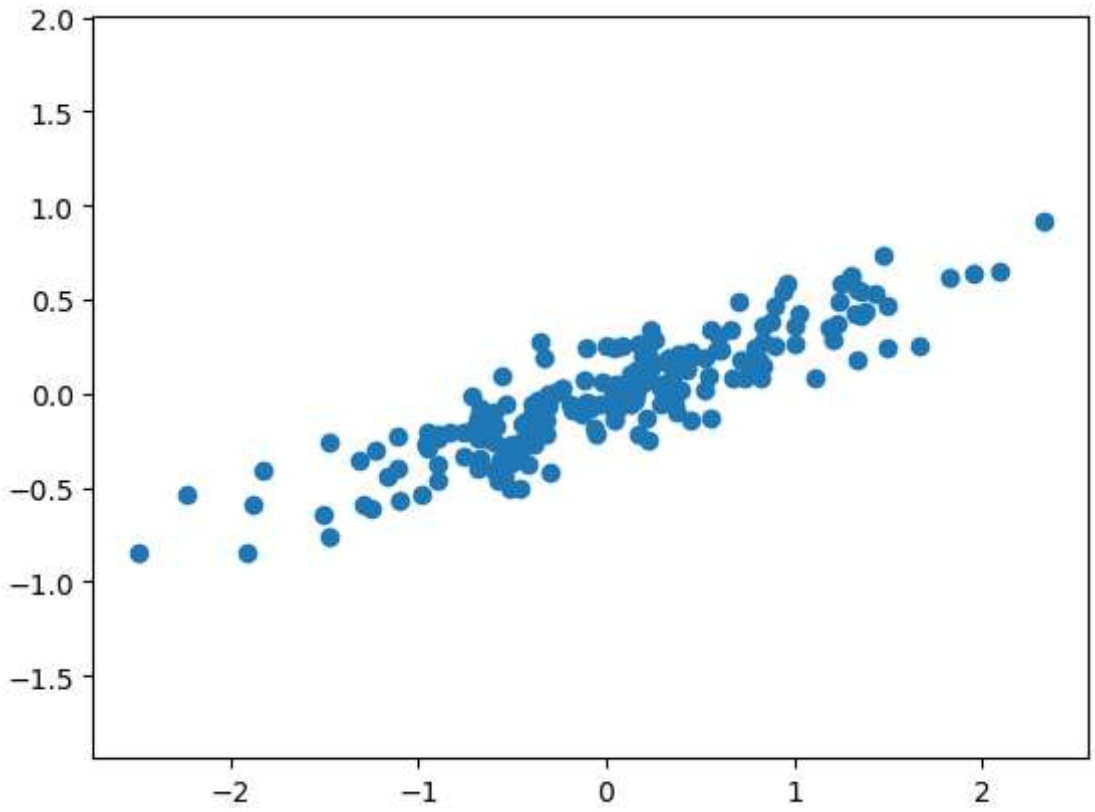
Out[20]:

	ï»¿age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	o
0	63	0.139825	0.405124	0.156856	-0.052631	0.491629	-0.206519	0.003171	-0.143033	0.
1	37	0.169873	0.250087	-0.023133	0.018010	-0.104177	0.224241	0.407461	-0.173771	0.
2	41	-0.544106	0.011869	-0.034363	-0.302646	-0.154750	-0.372700	0.362207	-0.258129	0.
3	56	0.284754	0.013393	-0.277588	-0.082941	-0.174629	0.375890	0.518448	-0.291288	-0.
4	57	-0.406213	-0.259625	-0.183648	0.575965	-0.115532	0.248683	0.161537	0.397102	-0.
...	...	...	...	...	...	...	...	...	...	...
298	57	-0.455449	-0.291092	0.148379	-0.031553	-0.129535	0.278825	-0.361427	0.445233	-0.
299	45	0.212450	0.615545	-0.385275	0.106928	-0.130287	0.280445	-0.240745	-0.217324	0.
300	68	0.163017	-0.224659	0.169202	-0.246403	0.573173	0.215191	-0.090516	-0.166757	0.
301	57	0.185858	-0.256136	-0.025310	-0.607931	-0.113979	0.245342	-0.413503	0.391767	0.
302	57	-0.572890	0.012497	-0.036181	-0.077387	-0.162936	-0.392416	0.415490	-0.271784	-0.

303 rows × 14 columns

Principal Component Analysis

In [21]:
```
rng = np.random.RandomState(1)
X = np.dot(rng.rand(2, 2), rng.randn(2, 200)).T
plt.scatter(X[:, 0], X[:, 1])
plt.axis('equal');
```



It is clear that x and y variable is highly co-related to each other as showen in above plot.

Decomposition

Now lets do the decomposition of 2 components

In [22]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca.fit(X)
```

Out[22]:
```
PCA(n_components=2)
```

In [23]:
```
print(pca.components_)
```

```
[[-0.94446029 -0.32862557]
 [-0.32862557  0.94446029]]
```

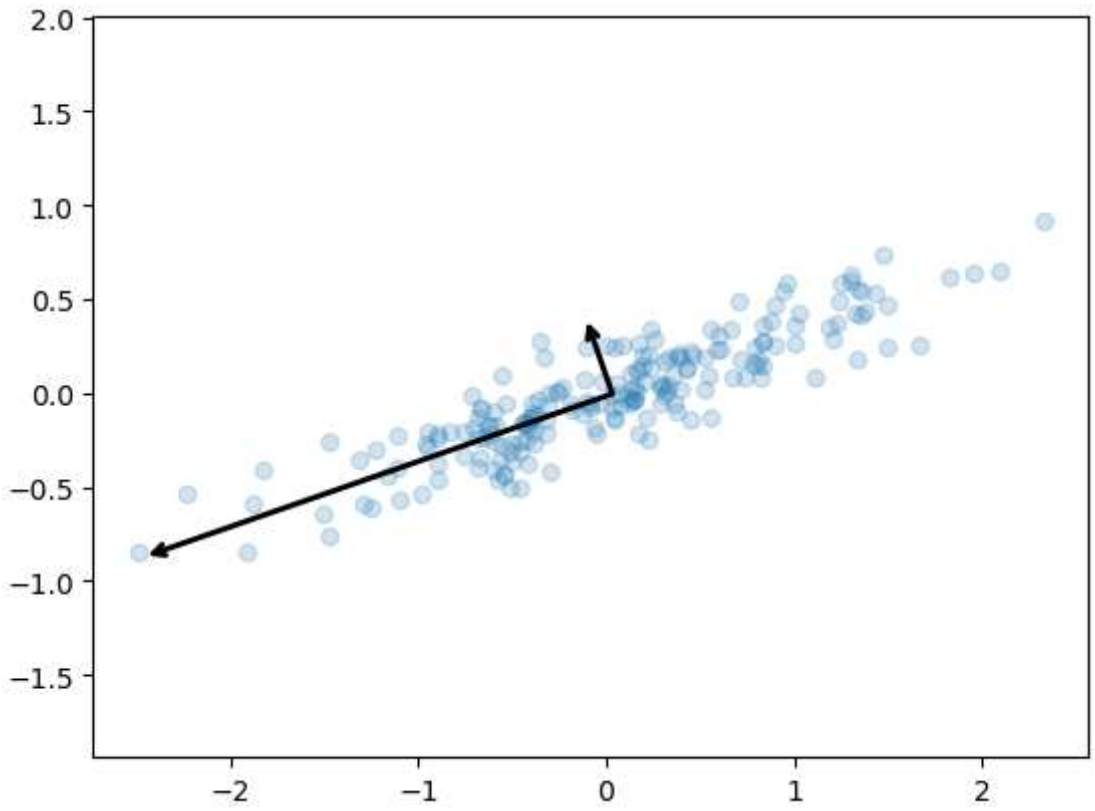
In [24]:
```
print(pca.explained_variance_)
```

```
[0.7625315 0.0184779]
```

In [25]:
```
def draw_vector(v0, v1, ax=None):
    ax = ax or plt.gca()
    arrowprops=dict(arrowstyle='->',
                    linewidth=2,
```

```
                    shrinkA=0, shrinkB=0)
    ax.annotate('', v1, v0, arrowprops=arrowprops, color='black')

# plot data
plt.scatter(X[:, 0], X[:, 1], alpha=0.2)
for length, vector in zip(pca.explained_variance_, pca.components_):
    v = vector * 3 * np.sqrt(length)
    draw_vector(pca.mean_, pca.mean_ + v)
plt.axis('equal');
```

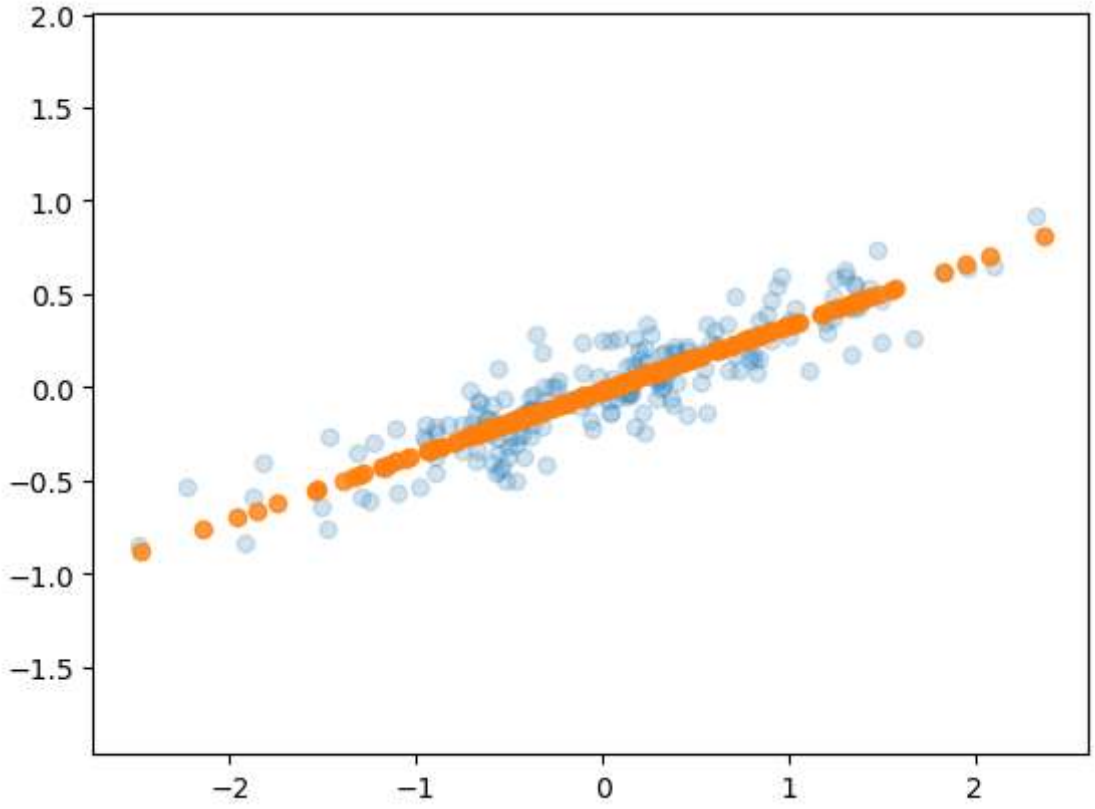


Principal components model fitting

In [26]:
```
pca = PCA(n_components=1)
pca.fit(X)
X_pca = pca.transform(X)
print("original shape:   ", X.shape)
print("transformed shape:", X_pca.shape)
```

```
original shape:    (200, 2)
transformed shape: (200, 1)
```

In [27]:
```
X_new = pca.inverse_transform(X_pca)
plt.scatter(X[:, 0], X[:, 1], alpha=0.2)
plt.scatter(X_new[:, 0], X_new[:, 1], alpha=0.8)
plt.axis('equal');
```

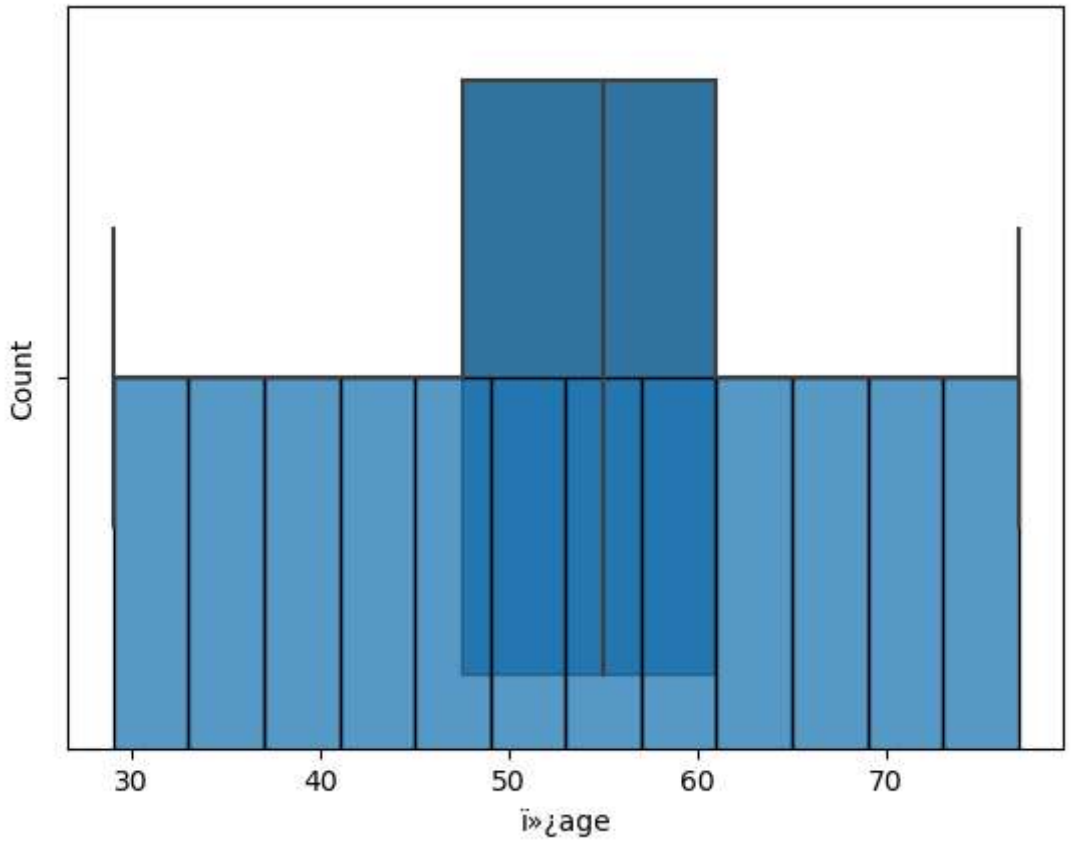


scikit-learn's `PCA` estimator and create the principal components

Univariate and bivariate analysis

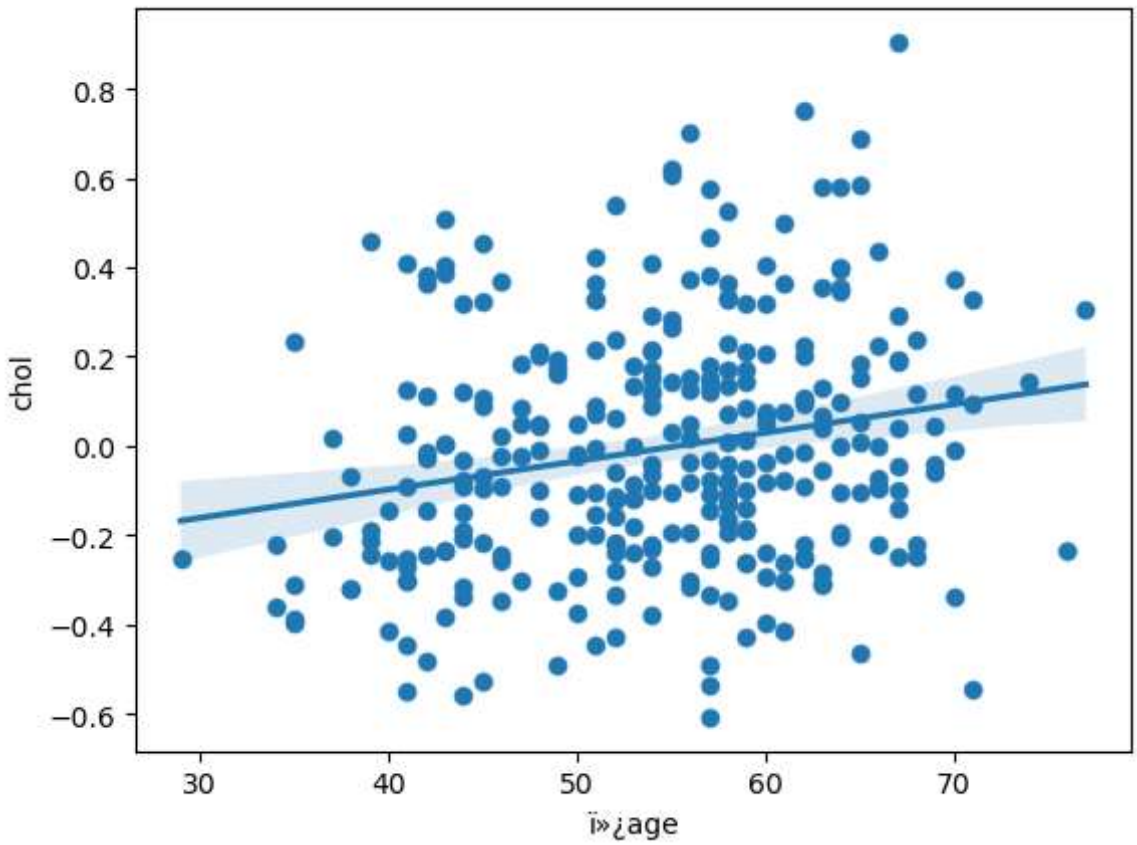
In [48]:
```
# Perform univariate analysis on a specific variable (e.g., 'age')
sns.histplot(df["ï»¿age"])
sns.boxplot(df["ï»¿age"])
# Add more visualizations or summary statistics as needed
```

Out[48]:
```
<AxesSubplot:xlabel='ï»¿age', ylabel='Count'>
```



In [49]:
```
# Perform bivariate analysis on two variables (e.g., 'age' and 'chol')
sns.scatterplot(data=df, x="ï»¿age", y="chol")
sns.regplot(data=df, x="ï»¿age", y="chol")
# Add more visualizations or statistical tests as needed
```

Out[49]:
```
<AxesSubplot:xlabel='ï»¿age', ylabel='chol'>
```



Selecting 2 important features Age and Cholestoral

In [37]:
```
features = ["ï»¿age", "chol"]

X = df.copy()
y = X.pop('target')
X = X.loc[:, features]

# Standardize
X_scaled = (X - X.mean(axis=0)) / X.std(axis=0)
```

In [38]:
```
from sklearn.decomposition import PCA

# Create principal components
pca = PCA()
X_pca = pca.fit_transform(X_scaled)

# Convert to dataframe
component_names = [f"PC{i+1}" for i in range(X_pca.shape[1])]
X_pca = pd.DataFrame(X_pca, columns=component_names)

X_pca.head()
```

Out[38]:

	PC1	PC2
0	0.552785	0.791600
1	-1.288584	-1.415604
2	-1.807464	-0.273867
3	-0.070700	0.325084
4	1.713331	-1.303232

In [39]:
```
df1 = pd.DataFrame(
    pca.components_.T,  # transpose the matrix of loadings
    columns=component_names,  # so the columns are the principal components
    index=X.columns,  # and the rows are the original features
)
df1
```

Out[39]:

	PC1	PC2
ï»¿age	0.707107	0.707107
chol	0.707107	-0.707107

Conclusion

- The coefficients in PC1 and PC2 represent the proportion of variance explained by each principal component. Since the coefficients for age and chol are the same in magnitude (0.707107), it suggests that both variables contribute equally to the variance explained by PC1 and PC2.
- Overall, based on the given coefficients, PC1 captures a common pattern or trend between age and chol, while PC2 represents an inverse relationship between age and chol. The orthogonality between PC1 and PC2 indicates that they capture different

sources of variation in the data. The equal coefficients suggest that both age and chol contribute equally to the variance explained by PC1 and PC2.

Thank You