ACT_4

October 21, 2023

1 Team Number: 7

2 Team Captain: Jacob Silva

3 Team Members: Joel Hurtado

4 Juliana Steele

5 1. Read the data into your software system

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import norm
import statsmodels.api as sm
import pylab as py
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix
from scipy import stats
```

[ ]:
```
data = pd.read_csv("HeartDisease.csv")

# chd is target variable
data.head()
```

[ ]:
```
   names  sbp  tobacco   ldl  adiposity  famhist  typea  obesity  alcohol  \
0      1  160    12.00  5.73      23.11  Present     49    25.30    97.20
1      2  144     0.01  4.41      28.61   Absent     55    28.87     2.06
2      3  118     0.08  3.48      32.28  Present     52    29.14     3.81
3      4  170     7.50  6.41      38.03  Present     51    31.99    24.26
4      5  134    13.60  3.50      27.78  Present     60    25.99    57.34

   age  chd
0   52    1
1   63    1
2   46    0
3   58    1
```

```
4    49     1
```

6 2. Examine univariate statistics for the following variables: sbp, tobacco, ldl, adiposity, typea, obesity, alcohol, and age

[ ]:
```
df = data[['sbp','tobacco','ldl','adiposity','typea','obesity','alcohol','age']]

stats = df.describe()
stats = stats.transpose()

print(stats)
```

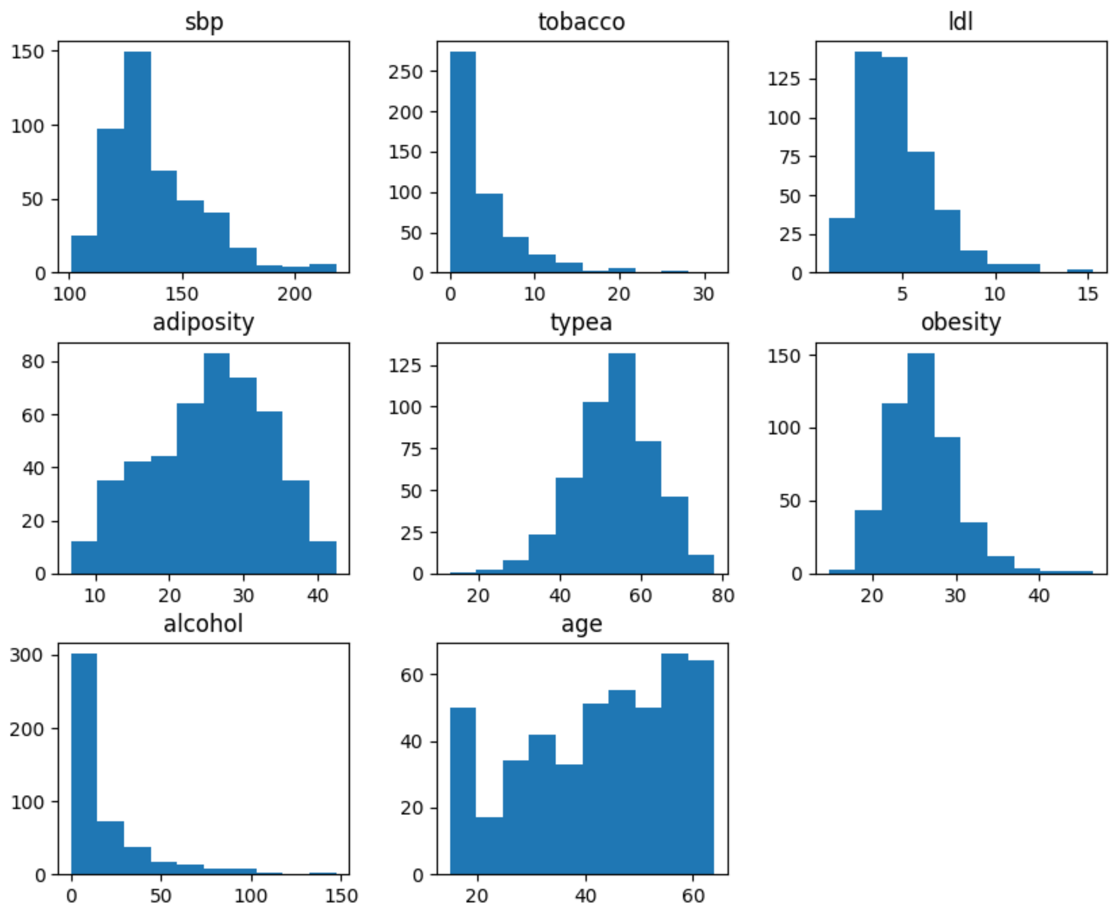
```
           count        mean        std     min       25%      50%       75%  \
sbp        462.0  138.326840  20.496317  101.00  124.0000  134.000  148.0000
tobacco    462.0    3.635649   4.593024    0.00    0.0525    2.000    5.5000
ldl        462.0    4.740325   2.070909    0.98    3.2825    4.340    5.7900
adiposity  462.0   25.406732   7.780699    6.74   19.7750   26.115   31.2275
typea      462.0   53.103896   9.817534   13.00   47.0000   53.000   60.0000
obesity    462.0   26.044113   4.213680   14.70   22.9850   25.805   28.4975
alcohol    462.0   17.044394  24.481059    0.00    0.5100    7.510   23.8925
age        462.0   42.816017  14.608956   15.00   31.0000   45.000   55.0000

              max
sbp        218.00
tobacco     31.20
ldl         15.33
adiposity   42.49
typea       78.00
obesity     46.58
alcohol    147.19
age         64.00
```

7 3. Produce histogram of each of the following variables with imposing normal curve: sbp, tobacco, ldl, adiposity, typea, obesity, alcohol, and age.

[ ]:
```
axes = df.hist(grid=False, figsize=(10, 8), layout=(3, 3))

axes
```

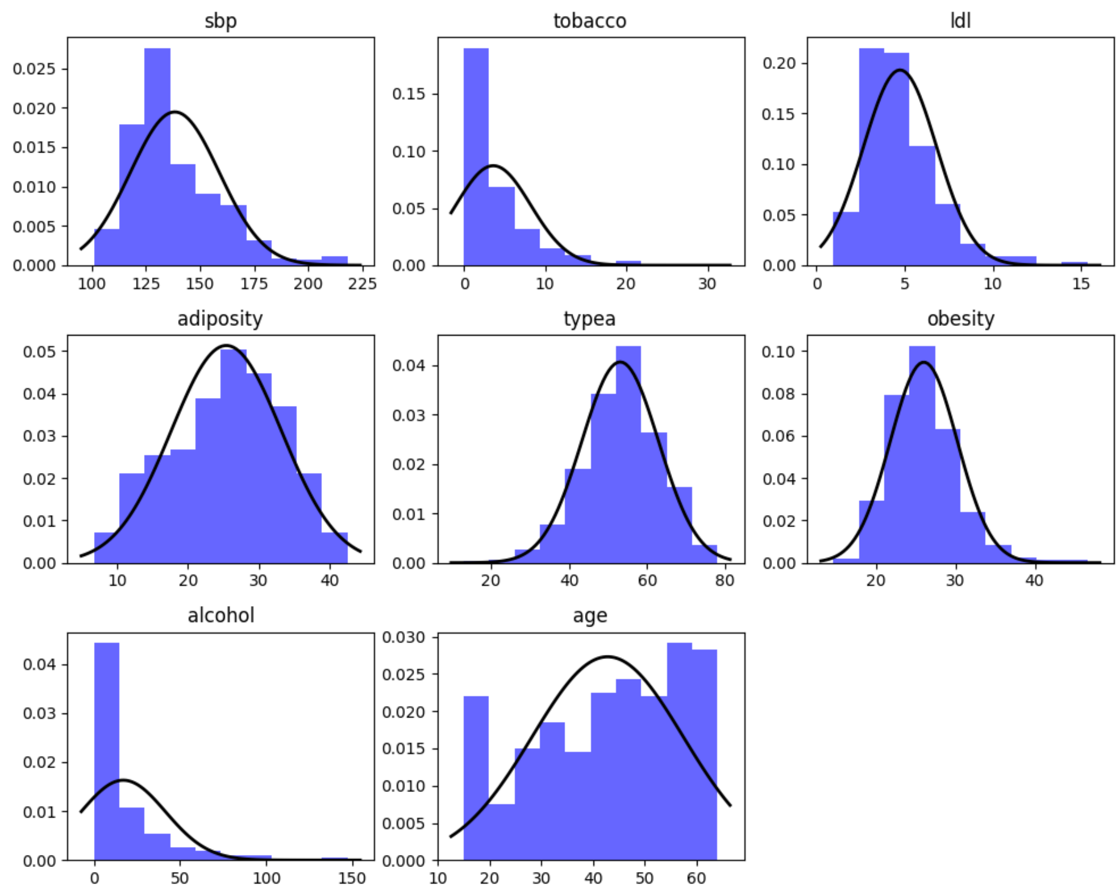
[ ]:
```
array([[<Axes: title={'center': 'sbp'}>,
        <Axes: title={'center': 'tobacco'}>,
        <Axes: title={'center': 'ldl'}>],
       [<Axes: title={'center': 'adiposity'}>,
        <Axes: title={'center': 'typea'}>,
```

```
 <Axes: title={'center': 'obesity'}>],
[<Axes: title={'center': 'alcohol'}>,
 <Axes: title={'center': 'age'}>, <Axes: >]], dtype=object)
```



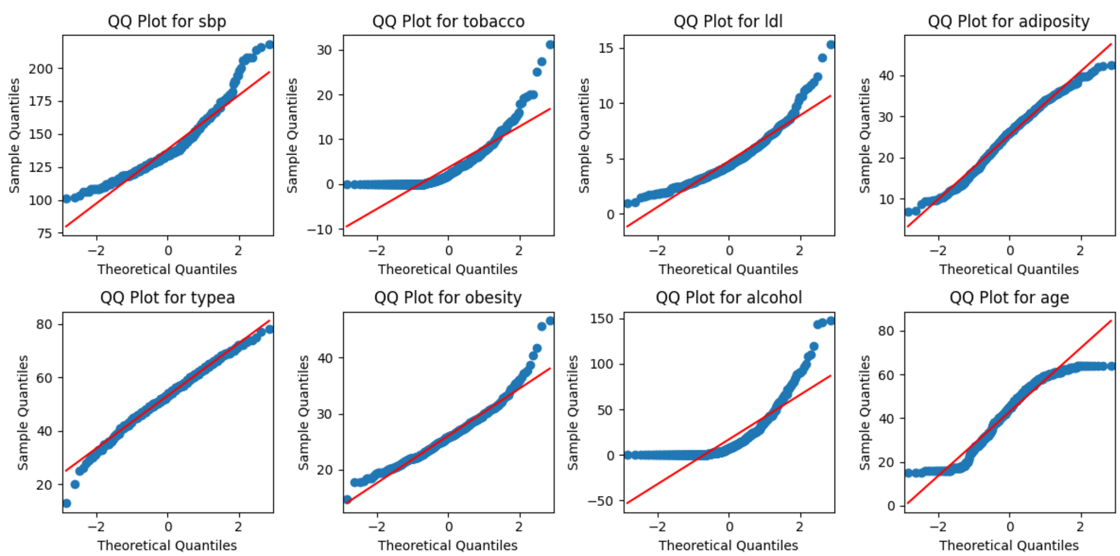
[ ]:
```python
num_cols = len(df.columns)

fig, axes = plt.subplots(3, 3, figsize=(10, 8))

axes = axes.ravel()

for i in range(min(num_cols, 9)):
    column_data = df.iloc[:, i]

    axes[i].hist(column_data, bins=10, density=True, alpha=0.6, color='b')

    mu, std = column_data.mean(), column_data.std()
    xmin, xmax = axes[i].get_xlim()
    x = np.linspace(xmin, xmax, 100)
```

```
    p = norm.pdf(x, mu, std)

    axes[i].plot(x, p, 'k', linewidth=2)

    axes[i].set_title(df.columns[i])

for i in range(num_cols, 9):
    axes[i].axis('off')

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



8 4. Produce quantile plot of each of the following variables: sbp, tobacco, ldl, adiposity, typea, obesity, alcohol, and age

[ ]:
```
fig, axes = plt.subplots(2, 4, figsize=(12, 6))
axes = axes.ravel()

for i, column in enumerate(df.columns):
    ax = axes[i]
    sm.qqplot(df[column], line='s', ax=ax)
    ax.set_title(f'QQ Plot for {column}')

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



9 5. Build a logistic regression model with all predictors

[ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
X = df
y = data['chd']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)


# in LogisticRegression x=10.0 changes the regularization strength
```

```
model = LogisticRegression(solver='liblinear', random_state=0).fit(X,y)
model.fit(X_train, y_train)


y_pred = model.predict(X_test)

confusion_matrix(y_test, y_pred)
```

[ ]: array([[49, 10],
       [19, 15]])

[ ]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.72      0.83      0.77        59
           1       0.60      0.44      0.51        34

    accuracy                           0.69        93
   macro avg       0.66      0.64      0.64        93
weighted avg       0.68      0.69      0.68        93
```

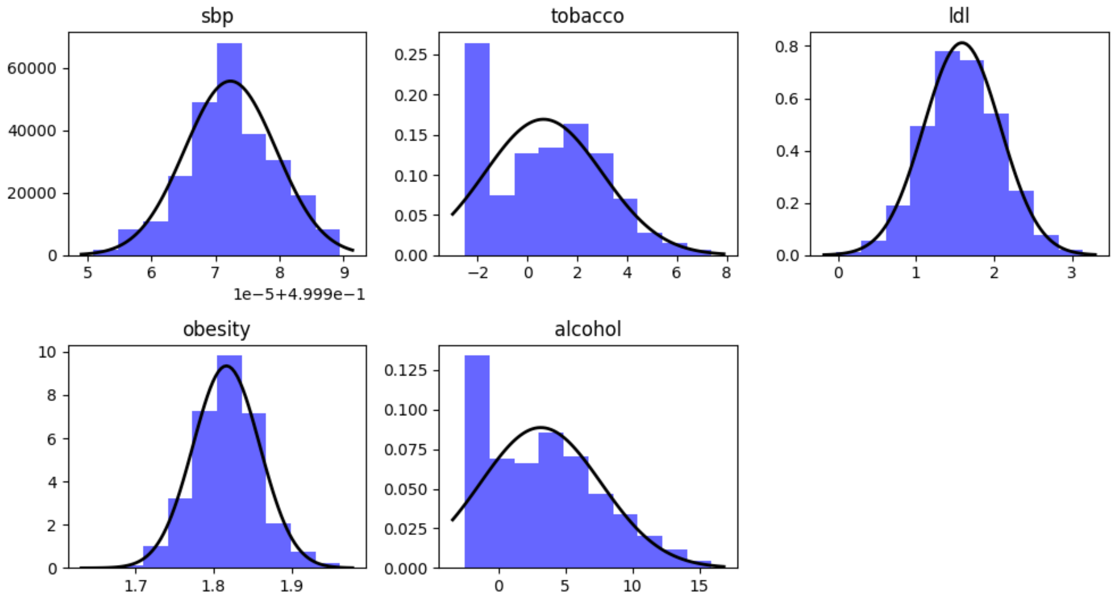
10 6. Perform power transformation on the following variables: sbp (power = -2), tobacco (power = 0.4), ldl (power = 0.1), obesity (power = -0.4), and alcohol (power = 0.4)

[ ]:
```
from scipy import stats
df = data[['sbp','tobacco','ldl','adiposity','typea','obesity','alcohol','age']]

lambda_values = [-2.0, 0.4, 0.1, 1.0, 1.0, -0.4, 0.4, 1.0]

df_power = df.copy()

for column, lam in zip(df, lambda_values):
    df_power[column] = stats.boxcox(df[column], lmbda=lam)
```

11 7. Produce histogram of each of the following transformed variables with imposing normal curve: sbp, tobacco, ldl, obesity, and alcohol.

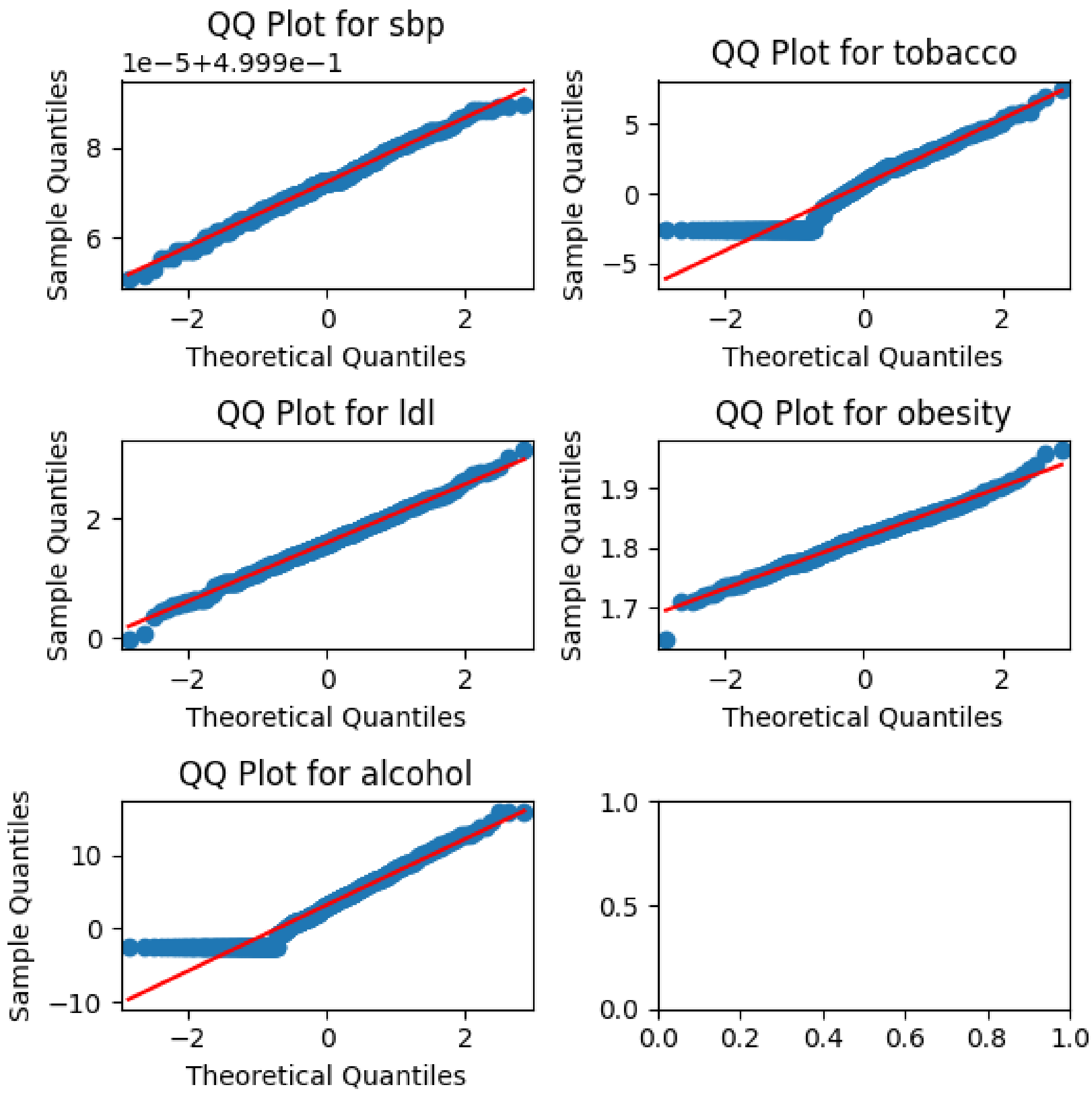
[ ]:
```
df_power_red = df_power[['sbp','tobacco','ldl','obesity','alcohol']]

num_cols = len(df_power_red.columns)
```

```
fig, axes = plt.subplots(3, 3, figsize=(10, 8))

axes = axes.ravel()

for i in range(min(num_cols, 9)):
    column_data = df_power_red.iloc[:, i]

    axes[i].hist(column_data, bins=10, density=True, alpha=0.6, color='b')

    mu, std = column_data.mean(), column_data.std()
    xmin, xmax = axes[i].get_xlim()
    x = np.linspace(xmin, xmax, 100)
    p = norm.pdf(x, mu, std)

    axes[i].plot(x, p, 'k', linewidth=2)

    axes[i].set_title(df_power_red.columns[i])

for i in range(num_cols, 9):
    axes[i].axis('off')

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



12 8. Produce quantile plot of each of the following transformed variables: sbp, tobacco, ldl, obesity, and alcohol

[ ]:
```
fig, axes = plt.subplots(3, 2, figsize=(6, 6))
axes = axes.ravel()

for i, column in enumerate(df_power_red.columns):
    ax = axes[i]
    sm.qqplot(df_power_red[column], line='s', ax=ax)
    ax.set_title(f'QQ Plot for {column}')

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



13 9. Build a logistic regression model with all predictors (transformed and three remaining original variables do not perform any transformation)

[ ]:
```
X_1 = df_power
y_1 = data['chd']

X_train_1, X_test_1, y_train_1, y_test_1 = train_test_split(X_1, y_1,␣
↪test_size=0.2, random_state=42)
```

```
# in LogisticRegression x=10.0 changes the regularization strength
model_1 = LogisticRegression(solver='liblinear', random_state=0).fit(X_1,y_1)
model_1.fit(X_train_1, y_train_1)


y_pred_1 = model_1.predict(X_test_1)

confusion_matrix(y_test_1, y_pred_1)
```

[ ]:
```
array([[48, 11],
       [20, 14]])
```

[ ]:
```
print(classification_report(y_test_1, y_pred_1))
```

```
              precision    recall  f1-score   support

           0       0.71      0.81      0.76        59
           1       0.56      0.41      0.47        34

    accuracy                           0.67        93
   macro avg       0.63      0.61      0.62        93
weighted avg       0.65      0.67      0.65        93
```

14 10. Build another logistic regression model with all predictors as in Part 9 except using significant predictors only.

[ ]:
```
X_2 = df_power_red
y_2 = data['chd']

X_train_2, X_test_2, y_train_2, y_test_2 = train_test_split(X_2, y_2,␣
 ↪test_size=0.2, random_state=42)


# in LogisticRegression x=10.0 changes the regularization strength
model_2 = LogisticRegression(solver='liblinear', random_state=0).fit(X_2,y_2)
model_2.fit(X_train_2, y_train_2)


y_pred_2 = model_2.predict(X_test_2)

confusion_matrix(y_test_2, y_pred_2)
```

[ ]:
```
array([[50,  9],
       [21, 13]])
```

[ ]:
```
print(classification_report(y_test_2, y_pred_2))
```

```
              precision    recall  f1-score   support

           0       0.70      0.85      0.77        59
           1       0.59      0.38      0.46        34

    accuracy                           0.68        93
   macro avg       0.65      0.61      0.62        93
weighted avg       0.66      0.68      0.66        93
```

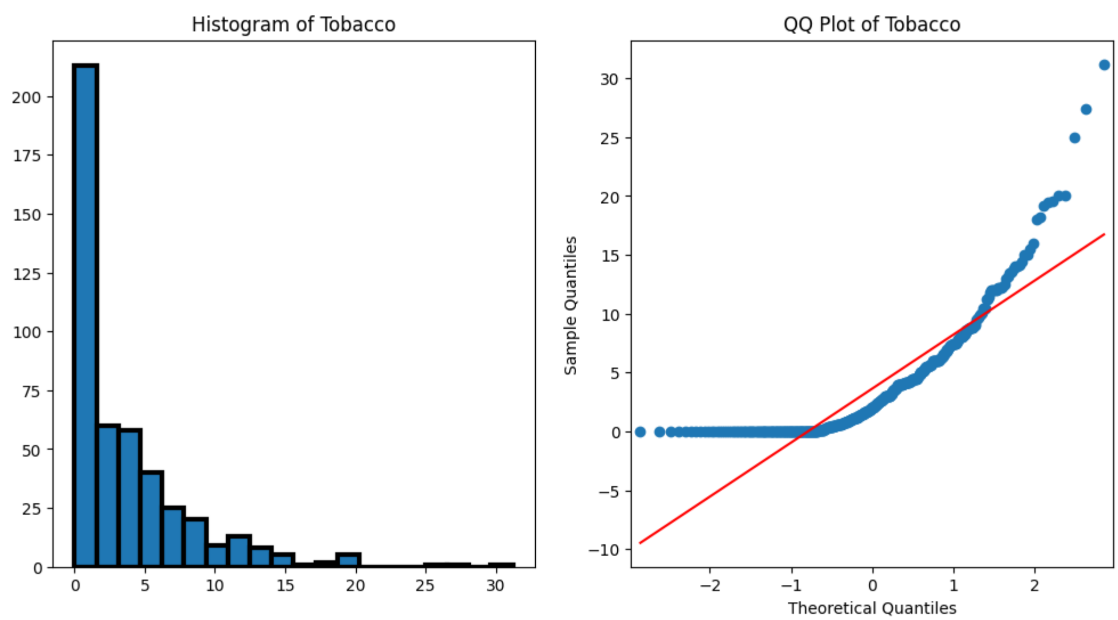
PART II Reporting (10 Points) 1. After completion of this activity, complete the following table

[ ]:
```
df_select_columns = data[['names', 'sbp', 'tobacco', 'ldl', 'adiposity',␣
 ↪'typea', 'obesity', 'alcohol', 'age']]



column_means = df_select_columns.mean()
column_median = df_select_columns.median()
column_skewness = df_select_columns.skew()

data_tbl = {
    'Variables': ['names', 'sbp', 'tobacco', 'ldl', 'adiposity', 'typea',␣
 ↪'obesity', 'alcohol', 'age'],
    'Mean':␣
 ↪[column_means['names'],column_means['sbp'],column_means['tobacco'],column_means['ldl'],colu
    'Median':␣
 ↪[column_median['names'],column_median['sbp'],column_median['tobacco'],column_median['ldl'],
    'Skewness':␣
 ↪[column_skewness['names'],column_skewness['sbp'],column_skewness['tobacco'],column_skewness
}

Table = pd.DataFrame(data_tbl)

style_dict = {
    'text-align': 'center',
    'border': '1px solid black'
}

styled_Table = Table.style.set_properties(**style_dict).
 ↪set_table_styles([{'selector': '', 'props': [('border', '1px solid␣
 ↪black')]}])

styled_Table
```

[ ]:
```
<pandas.io.formats.style.Styler at 0x7dfa7045a560>
```

2. Display the histogram and quantile plot of “tobacco”.

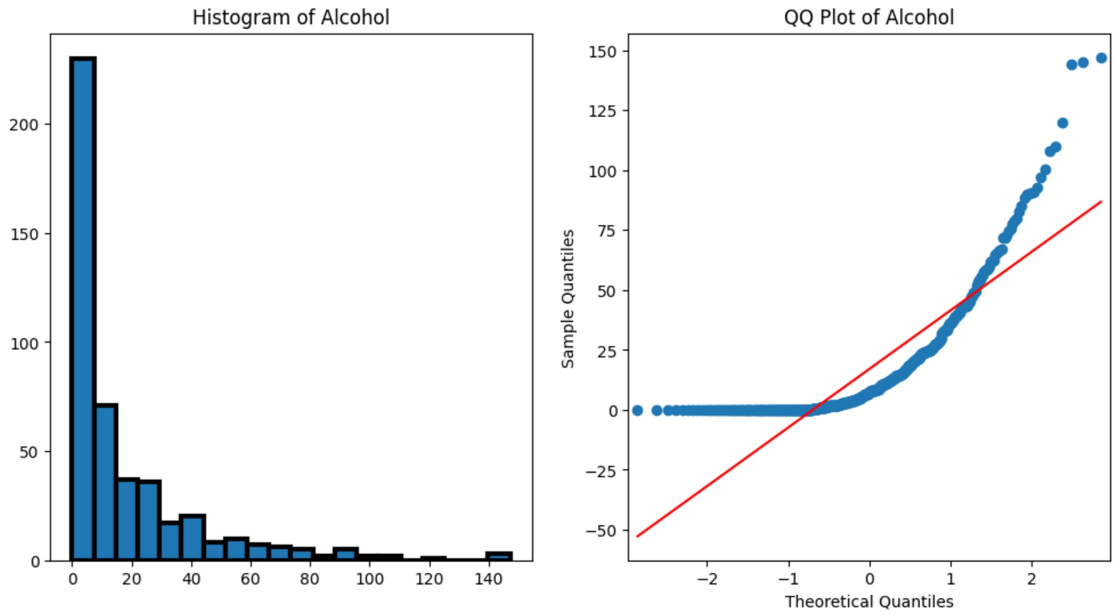
[ ]:
```
tobacco_data = df_select_columns["tobacco"]


fig, axes = plt.subplots(1, 2, figsize=(12, 6))


axes[0].hist(tobacco_data, bins=20, edgecolor="k",linewidth=3)

axes[0].set_title("Histogram of Tobacco")


sm.qqplot(tobacco_data, line='s', ax=axes[1])
axes[1].set_title("QQ Plot of Tobacco")

plt.show()
```



3. Display the histogram and quantile plot of "alcohol"

[ ]:
```
alcohol_data = df_select_columns["alcohol"]


fig, axes = plt.subplots(1, 2, figsize=(12, 6))


axes[0].hist(alcohol_data, bins=20, edgecolor="k",linewidth=3)

axes[0].set_title("Histogram of Alcohol")
```

```python
sm.qqplot(alcohol_data, line='s', ax=axes[1])
axes[1].set_title("QQ Plot of Alcohol")

plt.show()
```



4. Complete the following table

[ ]:
```python
for column in df_power.columns:
    data = df_power[column]
    mean = np.mean(data)
    median = np.median(data)
    skewness = stats.skew(data)

    print(f"Column: {column}")
    print(f"Mean: {mean}")
    print(f"Median: {median}")
    print(f"Skewness: {skewness}")
    print()
```

```
Column: sbp
Mean: 0.4999723389752801
Median: 0.4999721541546001
Skewness: -0.06309804169275851

Column: tobacco
Mean: 0.6524242122443431
```

```
Median: 0.7987697769322355
Skewness: 0.13429048073980526

Column: ldl
Mean: 1.5908671072190528
Median: 1.5810776346869808
Skewness: 0.02865204784191533

Column: adiposity
Mean: 24.4067316017316
Median: 25.115000000000002
Skewness: -0.21394839672197175

Column: typea
Mean: 52.103896103896105
Median: 52.00000000000001
Skewness: -0.34531194082632766

Column: obesity
Mean: 1.8166002789842062
Median: 1.8188253647320982
Skewness: 0.00027225561502506 47

Column: alcohol
Mean: 3.158923226305479
Median: 3.099983437191029
Skewness: 0.39964197601136525

Column: age
Mean: 41.816017316017316
Median: 43.99999999999999
Skewness: -0.3804937421038407
```

7. 95% confidence interval for tobacco

[ ]:
```python
tobacco_data = df['tobacco']

# Calculate the confidence interval
results = sm.stats.DescrStatsW(tobacco_data)
confidence_interval = results.tconfint_mean(alpha=0.05)

print("95% Confidence Interval for 'tobacco':", confidence_interval)
```

```
95% Confidence Interval for 'tobacco': (3.215728421332208, 4.055570279966494)
```

8. 95% confidence interval for alcohol

[ ]:
```
alcohol_data = df['alcohol']

# Calculate the confidence interval
results = sm.stats.DescrStatsW(alcohol_data)
confidence_interval = results.tconfint_mean(alpha=0.05)

print("95% Confidence Interval for 'tobacco':", confidence_interval)
```

```
95% Confidence Interval for 'tobacco': (14.806193411597095, 19.28259446719079)
```

9. Which model performed better based on C-statistic

Model 1

[ ]:
```
from sklearn.metrics import roc_auc_score, roc_curve, auc
logreg = model
logreg.fit(X_train, y_train)

# Predict probabilities for the positive class (class 1) on the test set
y_pred_proba = logreg.predict_proba(X_test_1)[:, 1]

# Compute ROC curve
fpr, tpr, _ = roc_curve(y_test, y_pred_proba)

# Calculate the AUC (Area Under the Curve)
roc_auc = auc(fpr, tpr)

print(f'AUC-ROC (C-Statistic): {roc_auc:.2f}')
```

```
AUC-ROC (C-Statistic): 0.73
```

Model 2

[ ]:
```
from sklearn.metrics import roc_auc_score, roc_curve, auc
logreg_1 = model_1
logreg_1.fit(X_train_1, y_train_1)

# Predict probabilities for the positive class (class 1) on the test set
y_pred_proba_1 = logreg_1.predict_proba(X_test_1)[:, 1]

# Compute ROC curve
fpr, tpr, _ = roc_curve(y_test_1, y_pred_proba_1)

# Calculate the AUC (Area Under the Curve)
roc_auc = auc(fpr, tpr)

print(f'AUC-ROC (C-Statistic): {roc_auc:.2f}')
```

```
AUC-ROC (C-Statistic): 0.78
```

Based on the C-statistic model 2 is the better model