Model Development

```
import random

# Sample data with 77 entries
age = [47, 36, 43, 25, 52, 23, 38, 22, 57, 36,
       32, 51, 21, 26, 21, 20, 51, 31, 60, 53,
       42, 58, 33, 64, 53, 38, 31, 43, 38, 45,
       43, 55, 41, 51, 48, 42, 45, 41, 50, 61,
       43, 58, 47, 37, 53, 48, 35, 44, 37, 40,
       32, 25, 32, 52, 19, 34, 60, 44, 52, 24,
       48, 44, 66, 64, 20, 42, 32, 39, 41, 34,
       45, 29, 27, 30, 28, 31, 33]

iq = [81, 104, 108, 106, 102, 104, 100, 98, 112, 71,
      102, 93, 87, 100, 100, 110, 110, 101, 102, 108,
      108, 98, 104, 114, 106, 87, 106, 109, 108, 85,
      104, 102, 112, 100, 106, 100, 95, 89, 108, 83,
      106, 104, 114, 98, 93, 93, 100, 110, 104, 85,
      102, 102, 106, 106, 98, 116, 97, 104, 89, 110,
      102, 93, 98, 97, 105, 110, 95, 99, 96, 103,
      100, 104, 101, 105, 99, 107, None]

group = ['HC', 'AVH-', 'AVH+', 'HC', 'AVH-', 'AVH+',
         'AVH-', 'HC', 'AVH+', 'HC', 'AVH-', 'AVH-',
         'HC', 'AVH+', 'AVH-', 'AVH-', 'AVH+', 'HC',
         'AVH-', 'AVH-', 'HC', 'AVH-', 'AVH+', 'HC',
         'AVH-', 'AVH+', 'HC', 'AVH-', 'HC', 'AVH+',
         'AVH-', 'AVH+', 'HC', 'AVH-', 'AVH+', 'AVH+',
         'HC', 'AVH-', 'HC', 'AVH+', 'HC', 'AVH-',
         'AVH+', 'AVH+', 'HC', 'AVH-', 'HC', 'AVH+',
         'AVH+', 'AVH+', 'HC', 'HC', 'AVH+', 'AVH+',
         'AVH-', 'HC', 'AVH-', 'HC', 'AVH+', 'AVH-',
         'HC', 'AVH+', 'AVH+', 'AVH+', 'AVH-', 'AVH-',
         'AVH-', 'HC', 'AVH+', 'AVH-', 'HC', 'HC', 'AVH-', 'AVH+', 'HC', 'AVH-', 'AVH+']

# Ensure all lists have the same length
if not (len(age) == len(iq) == len(group)):
    raise ValueError("The lists 'age', 'iq', and 'group' must have the same length.")

# Creating a DataFrame-like structure
```

```
df = []
for i in range(len(age)):
    df.append({
        'age': age[i],
        'iq': iq[i],
        'group': group[i]
    })

# Creating target variable
for row in df:
    row['target'] = 1 if row['group'] == 'AVH+' else 0  # Binary classification

# Handling missing IQ values by replacing them with the mean
iq_values = [row['iq'] for row in df if row['iq'] is not None]
iq_mean = sum(iq_values) / len(iq_values)

for row in df:
    if row['iq'] is None:
        row['iq'] = iq_mean

# Splitting the data into training and test sets
random.seed(42)
random.shuffle(df)

split_index = int(0.7 * len(df))  # 70% for training
train_data = df[:split_index]
test_data = df[split_index:]

# Manual implementation of a basic decision rule
def predict(row):
    # Simple rule: if age > 40 and iq > 100, predict AVH+
    return 1 if (row['age'] > 40 and row['iq'] > 100) else 0

# Making predictions
y_pred = []
y_true = [row['target'] for row in test_data]

for row in test_data:
    y_pred.append(predict(row))
```

```
# Evaluating accuracy
accuracy = sum(1 for i in range(len(y_true)) if y_true[i] == y_pred[i]) / len(y_true)

# Confusion Matrix calculation
confusion_matrix = {
    'TP': sum(1 for i in range(len(y_true)) if y_true[i] == 1 and y_pred[i] == 1),
    'TN': sum(1 for i in range(len(y_true)) if y_true[i] == 0 and y_pred[i] == 0),
    'FP': sum(1 for i in range(len(y_true)) if y_true[i] == 0 and y_pred[i] == 1),
    'FN': sum(1 for i in range(len(y_true)) if y_true[i] == 1 and y_pred[i] == 0),
}

# Model evaluation output
print(f'Accuracy: {accuracy * 100:.2f}%')
print(f'Confusion Matrix:\nTP: {confusion_matrix["TP"]}, TN: {confusion_matrix["TN"]},
FP: {confusion_matrix["FP"]}, FN: {confusion_matrix["FN"]}')
```