


This is a copy of a conversation between ChatGPT & Anonymous.

[Report conversation](#)

 Uploaded a file

help me understand task A and modeling the task

Got it  — your **Task A** from the document is about **modeling how blood pressure changes with age** before using those models to explore dataset bias in Task B.
Here's a breakdown of **what Task A asks** and **how to model it step by step**:

Task A: Modeling Blood Pressure as a Function of Age

Goal

Capture how **systolic (SBP)** and **diastolic (DBP)** blood pressures vary with age, using data from the referenced preprint.

Step 1: Choose model types

You must propose or compare models that describe these age-related trends.

Two main options are suggested:

1. Polynomial regression:

$$SBP(a) = c_1a^2 + c_2a + c_3, \quad DBP(a) = d_1a^2 + d_2a + d_3$$

- Captures smooth, curved trends.
- The sign of c_1, d_1 indicates curvature (e.g., increasing then flattening).

2. Sigmoidal or Gaussian models:

- For SBP (monotonic increasing):

$$SBP(a) = \frac{S_{\max}}{1 + e^{-k(a-a_0)}}$$

where S_{\max} is the upper plateau, k is growth rate, a_0 is the inflection (half-max) age.

- For DBP (rises then falls):

$$DBP(a) = D_{\max} \cdot e^{-\frac{(a-a_{\text{peak}})^2}{2\sigma^2}}$$

where D_{\max} is the max DBP, a_{peak} is the age of the peak, and σ controls spread.

Step 2: Fit parameters to data

Use the numerical mean SBP/DBP vs. age data (from the preprint) and fit model parameters.

In Python, for example:

python

 Copy code

```
import numpy as np
from scipy.optimize import curve_fit
```

```

# Example data
age = np.array([...]) # from preprint table
sbp_mean = np.array([...])
dbp_mean = np.array([...])

# Polynomial fit
coeff_sbp = np.polyfit(age, sbp_mean, 2)
coeff_dbp = np.polyfit(age, dbp_mean, 2)

# Sigmoid fit
def sigmoid(a, Smax, k, a0):
    return Smax / (1 + np.exp(-k * (a - a0)))

popt_sbp, _ = curve_fit(sigmoid, age, sbp_mean)

# Gaussian fit
def gaussian(a, Dmax, apeak, sigma):
    return Dmax * np.exp(-((a - apeak)**2) / (2 * sigma**2))

popt_dbp, _ = curve_fit(gaussian, age, dbp_mean)

```

Step 3: Evaluate fits

For each model:

- Compute **MSE** (mean squared error).
- Compute **R²** (goodness of fit).
- Plot both fitted curves against the data.

Example:

```
from sklearn.metrics import mean_squared_error, r2_score

pred_poly = np.polyval(coeff_sbp, age)
mse_poly = mean_squared_error(sbp_mean, pred_poly)
r2_poly = r2_score(sbp_mean, pred_poly)
```

Step 4: Interpret parameters

- **Polynomial:**

$c_1, d_1 \rightarrow$ curvature (positive = convex upward, negative = downward).

Units: since SBP is in mmHg and a in years, c_1 has units mmHg/year².

- **Sigmoid/ Gaussian:**

- S_{\max} : plateau (max BP)
 - a_0 : half-max age
 - D_{\max} : max DBP
 - a_{peak} : age where DBP peaks
 - σ : width/spread of DBP curve
-

Step 5: Discussion

You'll compare:

- Which model fits better (lower MSE, higher R²)?
- Do the parameters make physiological sense?
 - SBP steadily rises then flattens (sigmoid often fits better).

- DBP peaks midlife (Gaussian fits better).
 - Limitations: can't capture demographic nuances like sex, ethnicity, etc.
-

Would you like me to **show a full Python modeling example** (with synthetic age–BP data and both model types fitted/compared)? That would make the structure of Task A completely clear.

continue for task b

Excellent — let's now move to **Task B**, which builds directly on Task A's modeling foundation.

Task A modeled *how* blood pressure behaves with age;

Task B now uses **synthetic blood pressure data** to explore **bias in machine-learning classification**.

Task B: Model-Based Bias Removal Using Synthetic Blood Pressure Data

Objective

You'll generate synthetic blood pressure (SBP, DBP) data for males and females, train a binary classifier to predict sex, then analyze how **data imbalance** affects performance and bias.

Step 1: Generate Synthetic Data

Each sex follows a **bivariate normal distribution** for SBP and DBP with given parameters.

Parameter	Male	Female
μ_{SBP}	133.0	125.8
σ_{SBP}	18.6	19.0
μ_{DBP}	78.8	74.8
σ_{DBP}	12.6	12.4
ρ (correlation)	0.45	0.50

Formula

For a bivariate normal distribution:

$$\Sigma = \begin{bmatrix} \sigma_{SBP}^2 & \rho \sigma_{SBP} \sigma_{DBP} \\ \rho \sigma_{SBP} \sigma_{DBP} & \sigma_{DBP}^2 \end{bmatrix}$$

Then draw samples using:

$$\begin{bmatrix} SBP \\ DBP \end{bmatrix} \sim \mathcal{N} \left(\begin{bmatrix} \mu_{SBP} \\ \mu_{DBP} \end{bmatrix}, \Sigma \right)$$

Example (Python)

python

 Copy code

```
import numpy as np
import pandas as pd

n_total = 100_000
ratio_male = 0.7          # try 0.5, 0.7, 0.9 later to test bias
n_male = int(n_total * ratio_male)
n_female = n_total - n_male

# parameters
mu_m = [133.0, 78.8]
sigma_m = [18.6, 12.6]
rho_m = 0.45

mu_f = [125.8, 74.8]
sigma_f = [19.0, 12.4]
rho_f = 0.5

# covariance matrices
cov_m = [[sigma_m[0]**2, rho_m*sigma_m[0]*sigma_m[1]],
         [rho_m*sigma_m[0]*sigma_m[1], sigma_m[1]**2]]
cov_f = [[sigma_f[0]**2, rho_f*sigma_f[0]*sigma_f[1]],
         [rho_f*sigma_f[0]*sigma_f[1], sigma_f[1]**2]]

# sample data
male_data = np.random.multivariate_normal(mu_m, cov_m, n_male)
female_data = np.random.multivariate_normal(mu_f, cov_f, n_female)
```

```
# combine
df = pd.DataFrame(np.vstack([male_data, female_data]), columns=["SBP", "DBP", "Sex"])
df["Sex"] = [1]*n_male + [0]*n_female
```

This gives you a **synthetic population** with controllable imbalance.

Step 2: Build a Binary Classifier

Train a model to predict sex (1 = male, 0 = female) using only SBP and DBP.

Typical classifiers:

- Logistic Regression  (interpretable, baseline)
- SVM or Decision Tree (optional comparison)

Example:

python

 Copy code

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, auc, f1_score, accuracy_score

X = df[["SBP", "DBP"]]
y = df["Sex"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, s

clf = LogisticRegression()
clf.fit(X_train, y_train)
```

```
y_pred = clf.predict(X_test)
y_prob = clf.predict_proba(X_test)[:,1]

# metrics
f1 = f1_score(y_test, y_pred)
acc = accuracy_score(y_test, y_pred)
print(f"F1: {f1:.3f}, Accuracy: {acc:.3f}")
```

Step 3: Evaluate Bias

Repeat the experiment with different **male : female ratios**:

- 50:50 (balanced)
- 70:30 (moderate imbalance)
- 90:10 (severe imbalance)

For each setting:

- Compute ROC AUC, F1, and accuracy.
- Plot ROC curves.

Bias will appear as:

- Model accuracy skewed toward the **majority class**.
 - F1 score dropping for the **minority**.
 - AUC possibly stable, but precision/recall deteriorating asymmetrically.
-

Step 4: Discussion of Bias

Reflect on:

- Why imbalanced data make classifiers biased
→ the model learns to optimize accuracy, so it favors the dominant class.
- Real-world analogy: fewer female data → model poorer for women → diagnostic bias.
- Importance of balanced datasets in healthcare models.

Step 5: Bias Mitigation

You're asked to **"modify the training strategy"** to reduce sex bias.

Several standard techniques apply:

Approach	Implementation	Idea
Resampling	Oversample minority (e.g., SMOTE) or undersample majority	Balance class frequencies
Class weighting	In logistic regression: class_weight='balanced'	Penalize errors on minority more
Fair-loss modification	Custom loss to equalize performance (e.g., equal opportunity)	Adjust gradients to counter bias
Synthetic data augmentation	Generate more synthetic samples for minority	Expand training coverage

Example using **class weights**:

python

 Copy code

```
clf = LogisticRegression(class_weight='balanced')
clf.fit(X_train, y_train)
```

Step 6: Reflection

In your report/discussion, answer:

1. How performance changes with different ratios?
 2. What's the evidence of bias?
 3. Which mitigation technique helped most?
 4. What challenges exist when real data are unbalanced (e.g., demographic skew, missing data, sampling limits)?
-

Would you like me to extend this by **showing full Python code** that:

- generates imbalanced datasets,
- trains and evaluates classifiers for several ratios,
- and visualizes ROC curves and F1 trends?

That would complete Task B's implementation and analysis.