1.

a) The model is linear. There is no under-modeling. $\beta_0 = 1,\ \beta_1 = 2,\ \beta_2 = 0.$

b) The model is nonlinear. There is no under-modeling. $a_0 = 3,\ a_1 = 3,\ b_0 = 2,\ b_1 = 3.$

c) The model is linear. There is under-modeling.

2.

```python
def generate_U(u, d):
    n = len(u)
    Xdly = np.zeros((n, d))
    row = Xdly.shape[0]
    Xdly_row = np.zeros((d, 1))
    for i in range(row):
        for j in range(d):
            Xdly_row[j, :] = np.exp(-j * u[i, :] / d)
        Xdly[i, :] = Xdly_row.reshape(1, -1)
    return Xdly

dmax = 11
utr, uts, ytr, yts = train_test_split(u, y, test_size=0.5)
dtest = np.arange(1, dmax)
nd = len(dtest)
mses = np.zeros(nd)

for it, d in enumerate(dtest):
    Utr = generate_U(utr, d)
    Uts = generate_U(uts, d)

    regr = LinearRegression().fit(Utr, ytr)
    yhat = regr.predict(Uts)
    mses[it] = np.mean((yhat - yts) ** 2)
optimal_arg = np.argmin(mses)
optimal_d = dtest[optimal_arg]
print(optimal_d)
```

Code:

```python
def generate_U(u, d):
    n = len(u)
    Xdly = np.zeros((n, d))
    row = Xdly.shape[0]
    Xdly_row = np.zeros((d, 1))
    for i in range(row):
        for j in range(d):
            Xdly_row[j, :] = np.exp(-j * u[i, :] / d)
        Xdly[i, :] = Xdly_row.reshape(1, -1)
    return Xdly
```

```
dmax = 11
utr, uts, ytr, yts = train_test_split(u, y, test_size=0.5)
dtest = np.arange(1, dmax)
nd = len(dtest)
mses = np.zeros(nd)

for it, d in enumerate(dtest):
    Utr = generate_U(utr, d)
    Uts = generate_U(uts, d)

    regr = LinearRegression().fit(Utr, ytr)
    yhat = regr.predict(Uts)
    mses[it] = np.mean((yhat - yts) ** 2)
optimal_arg = np.argmin(mses)
optimal_d = dtest[optimal_arg]
print(optimal_d)
```

3.

a) $Bias(x) = E\left(f\left(x,\hat{\beta}\right)\right) - f(x,\beta_0) = f(x,\beta_0) - f(x,\beta_0) = 0$

b) $Bias(x) = E\left(f\left(x,\hat{\beta}\right)\right) - f(x,\beta_0) = f(x,\beta_0) + \epsilon - f(x,\beta_0) = \epsilon$

c) $Bias(x) = E\left(f\left(x,\hat{\beta}\right)\right) - f(x,\beta_0) = f(x+\epsilon,\beta_0) - f(x,\beta_0)$

$= \beta_0(x+\epsilon)^2 + \beta_0 x^2 = \beta_0(\epsilon^2 + 2\epsilon x)$

4.

a) Assume $A = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}$, $y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$, $\hat{\beta} = \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix}$

$\hat{\beta} = (A^T A)^{-1} A^T y$

b) Assume $y = \begin{bmatrix} \beta_{00} + \beta_{01}x_1 + \beta_{02}x_1^2 \\ \beta_{00} + \beta_{01}x_2 + \beta_{02}x_2^2 \\ \vdots \\ \beta_{00} + \beta_{01}x_n + \beta_{02}x_n^2 \end{bmatrix}$

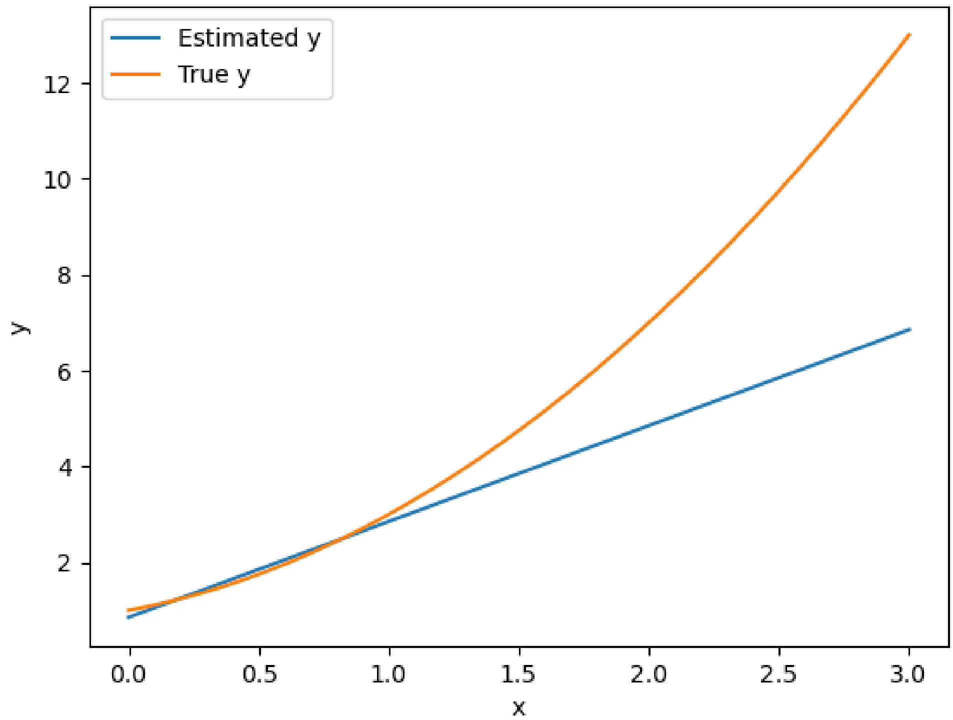
$\hat{\beta} = (A^T A)^{-1} A^T y$

c)

```python
n = 10
beta = np.array([1, 2, -1])
x = np.linspace(0, 1, n).reshape(-1, 1)

A = np.zeros((n, 2))
A[:, 0] += 1
A[:, 1] += x[:, 0]

y = 1 + x + x ** 2
estimated_beta = np.linalg.inv(A.T @ A) @ A.T @ y
print(estimated_beta)
x = np.linspace(0, 3, 100)
estimated_y = estimated_beta[0] + estimated_beta[1] * x
true_y = 1 + x + x ** 2
plt.plot(x, estimated_y)
plt.plot(x, true_y)
plt.xlabel('x')
plt.ylabel('y')
plt.legend(['Estimated y', 'True y'])
plt.show()
```



d) When $x = 3$, bias is the largest

5.

a) Let cancer volume to be x_1, age to be x_2, type I to be x_3 and type II to be x_4.

Model 1: $y = \beta_0 + \beta_1 x_1$
Model 2: $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$
Model 3: $y = \beta_0 + (\beta_3 x_3 + \beta_4 x_4)x_1 + \beta_2 x_2$

b) There are two parameters in model 1, three parameters in model 2 and four parameters in model 3. Model 3 is the most complex.

c) Model 1: $A = \begin{bmatrix} 1 & 0.7 \\ 1 & 1.3 \\ 1 & 1.6 \end{bmatrix}$

Model 2: $A = \begin{bmatrix} 1 & 0.7 & 55 \\ 1 & 1.3 & 65 \\ 1 & 1.6 & 70 \end{bmatrix}$

Model 3: $A = \begin{bmatrix} 1 & 0.7 & 55 & 1 & 0 \\ 1 & 1.3 & 65 & 0 & 1 \\ 1 & 1.6 & 70 & 0 & 1 \end{bmatrix}$

d) The smallest mean RSS is 0.7.

$$Target\ score = 0.7 + \frac{0.05}{3} = 0.7167$$

Only model 3 satisfy $mean\ RSS < Target\ score$. Thus, we should select model 3.