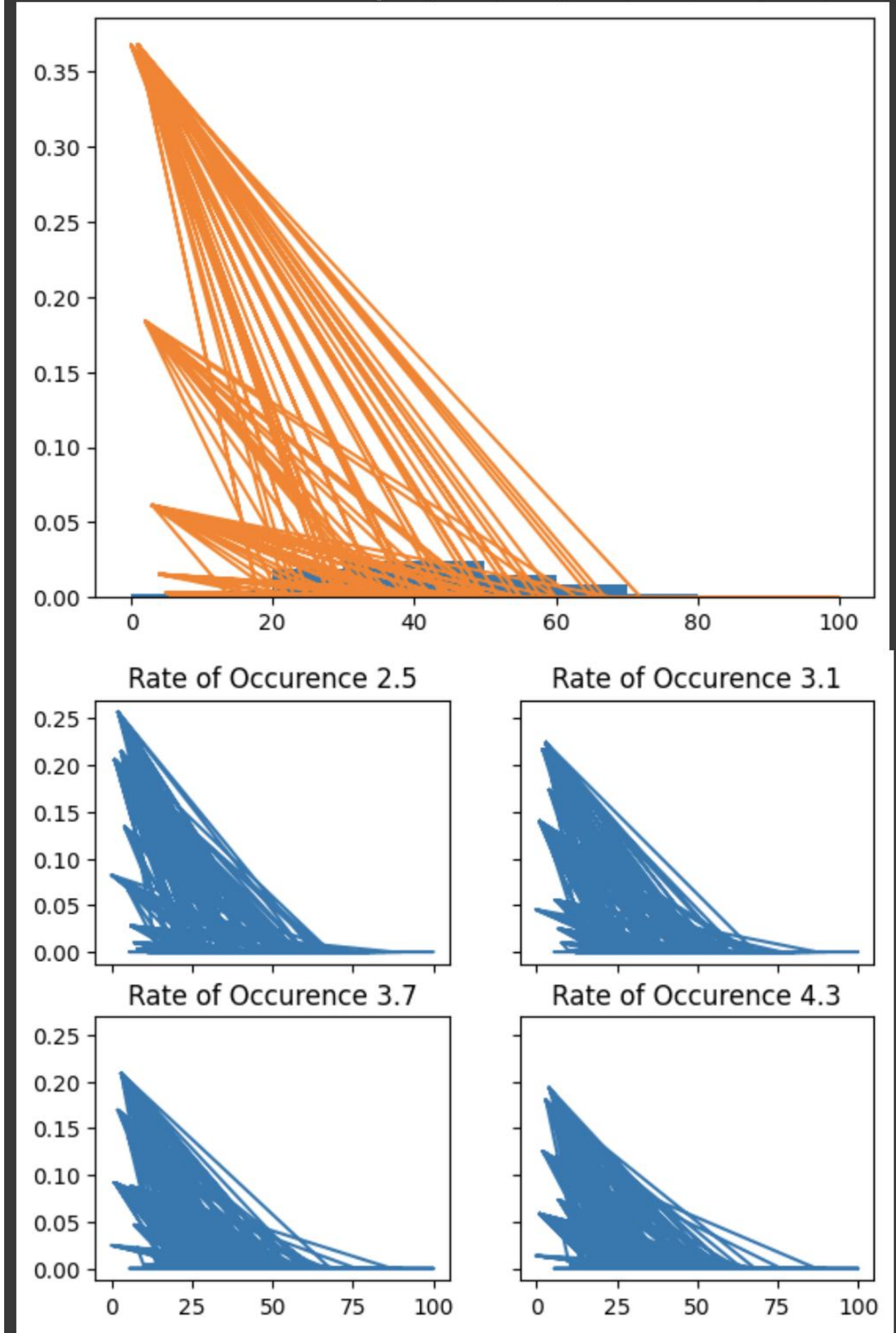
```python
import numpy as np

def problem_1a (A, B):
    return A + B

def problem_1b (A, B, C):
    return (A @ B) - C

def problem_1c (A, B, C):
    return (A * B) + C.T

def problem_1d (x, y):
    return np.dot(x,y)

def problem_1e (A, x):
    return np.linalg.solve(np.linalg.inv(A),x)

def problem_1f (A, x):
    return np.linalg.solve(A.T, x.T).T

def problem_1g (A, j):
    return np.sum(A[list(filter(lambda x: x%2==0,np.arange(A.shape[0]+1))),j])

def problem_1h (A, c, d):
    A = A[np.nonzero(A > c-1)]
    A = A[np.nonzero(A < d+1)]
    return np.mean(A)

def problem_1i (A, k):
  e_vals,e_vctr = np.linalg.eig(A)
  if k==0:
    return np.array([0.])
  elif k>A.shape[0]:
    return np.zeros(k)
  else:
    e_vals.sort()
    e_vals = e_vals[::-1][:k]
  return e_vals

def problem_1j (x, k, m, s):
    z = np.ones(x.shape[0])[:,np.newaxis]
    I = np.identity(x.shape[0])
    return np.random.multivariate_normal(mean=(x+(m*z)).flatten(),cov=(s*I),size=(x.shape[0],k))[0].T

def problem_1k (A):
    return np.random.shuffle(A)

def problem_1l (x):
    return np.divide(np.subtract(A,np.mean(A)),np.std(A))

def problem_1m (x, k):
    return np.repeat(x,k,axis=0).reshape(x.shape[0],k)
```

```python
def problem_2a()->None:
    X = np.arange(9).reshape(3,3)
    row_min = X.min(axis=1)[:,np.newaxis]
    '''
    Previously it was a row vector so it was subtracting element wise but after reshaping
    it in column vector we are able to get the desired results
    '''
    print(X-row_min)
    return None

def problem_2b()->None:
    X = np.arange(9).reshape(3,3)
    row_min = X.min(axis=1).reshape(-1,1)
    '''
    to get to this solution I didn't use chatgpt or anything else just tried by hand and then replicated it on google colab
    '''
    print(
        np.subtract(
            np.multiply(
                X,np.ones(shape=(3,3,3))
                ),
            np.vstack(
                tup=(
                    np.multiply(
                        np.array([row_min[0]]),np.ones(shape=(3,3))
                        ),
                    np.multiply(
                        np.array([row_min[1]]),np.ones(shape=(3,3))
                        ),
                    np.multiply(np.array([row_min[2]]),np.ones(shape=(3,3))),
                    )
                ).reshape(3,3,3)
                )
        )
```

```python
def linear_regression (X_tr, y_tr):
    return np.linalg.inv(X_tr.T.dot(X_tr)).dot(X_tr.T).dot(y_tr)

def train_age_regressor ()->None:
    # Load data
    X_tr = np.reshape(np.load("age_regression_Xtr.npy"), (-1, 48*48))
    y_tr = np.load("age_regression_ytr.npy")
    X_te = np.reshape(np.load("age_regression_Xte.npy"), (-1, 48*48))
    y_te = np.load("age_regression_yte.npy")

    w = linear_regression(X_tr, ytr)

    # Report fMSE cost on the training and testing data (separately)
    mse_tr = ((1 / (2 * X_tr.shape[0])) * np.sum(np.square(np.dot(X_tr,w)-y_tr)))
    print('Mean squared error for Training Set is {:.2f}'.format(mse_tr))

    mse_te = ((1 / (2 * X_te.shape[0])) * np.sum(np.square(np.dot(X_te,w)-y_te)))
    print('Mean squared error for Test Set is {:.2f}'.format(mse_te))
    '''
    Mean squared error for Training Set is 40.43
    Mean squared error for Test Set is 373.09
    '''
    return None
```

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import scipy

problem_4a = np.load("age_regression_ytr.npy")

plt.hist(x=problem_4a,density=True)

print(
  np.mean(problem_4a),
  np.median(problem_4a),
  np.min(problem_4a),
  np.max(problem_4a),
  sorted(problem_4a)[40:60],
)

plt.plot(problem_4a,scipy.stats.poisson.pmf(problem_4a,1))
plt.show()

rate_of_occurence_mu = [2.5,3.1,3.7,4.3]

figure, axis = plt.subplots(nrows=2,ncols=2,sharex=True,sharey=True,)

axis[0, 0].plot(problem_4a,scipy.stats.poisson.pmf(problem_4a,rate_of_occurence_mu[0]))
axis[0, 0].set_title(f"Rate of Occurence {rate_of_occurence_mu[0]}")

axis[0, 1].plot(problem_4a,scipy.stats.poisson.pmf(problem_4a,rate_of_occurence_mu[1]))
axis[0, 1].set_title(f"Rate of Occurence {rate_of_occurence_mu[1]}")

axis[1, 0].plot(problem_4a,scipy.stats.poisson.pmf(problem_4a,rate_of_occurence_mu[2]))
axis[1, 0].set_title(f"Rate of Occurence {rate_of_occurence_mu[2]}")

axis[1, 1].plot(problem_4a,scipy.stats.poisson.pmf(problem_4a,rate_of_occurence_mu[3]))
axis[1, 1].set_title(f"Rate of Occurence {rate_of_occurence_mu[3]}")

plt.show()
```

```
39.57944735 40.0 0.0 100.0 [3.0, 3.0, 3.0, 3.0, 4.0, 4.0, 4.0, 4.0, 4.0, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0, 5.5, 6.0, 6.0, 6.0, 6.0]
```



```python
def get_mu_sigma(x:float)->tuple:
    return np.power(x,2), (2 - (1 / (1 + np.exp(-np.power(x,2)))))
mu_val,sigma_val = get_mu_sigma(x=1)
random_value_score = 0
z_score_val = ((random_value_score - mu_val) / sigma_val)
prob = scipy.stats.norm.cdf(z_score_val)
comute_percent = (1-prob) * 100 # more than random_value_score
print('Proba : {:.2f}%'.format(comute_percent))
```

```
Proba : 78.47%
```

```python
# for problem 4b
# 1. when value of x is large in magnitute the value of y tends to be larger
# 2. when value of x is small in magnitute the uncertainty in the corresponding value of y tend to be larger
# 3. The probability that a r.v. Y sampled from that distribution is positive - Proba : 78.47%
```

DATE / / /

5(a) Two column vectors x & a.

$x \rightarrow \begin{bmatrix} x_1 \\ x_2 \\ \vdots \end{bmatrix} \quad a \rightarrow \begin{bmatrix} a_1 \\ a_2 \\ \vdots \end{bmatrix}$

$x^T a = [x_1, x_2 \ldots x_n] \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}$

$= x_1 a_1 + x_2 a_2 + \ldots x_n a_n \rightarrow (1)$

$a^T x = [a_1, a_2 \ldots a_n] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$

$= a_1 x_1 + a_2 x_2 \ldots a_n x_n \rightarrow (2)$

$\nabla_x (x^T a) = \sum_{i=1}^{n} a_i = a$ (vector)

Similarly $\nabla_x (a^T x) = \sum_{i=1}^{n} a_i = a$ (vector)

$\boxed{\therefore \nabla_x (x^T a) = \nabla_x (a^T x) = a}$

5(b) To prove $\nabla_x (x^T A x) = (A + A^T) x$

$x \in R^n$ & $A \rightarrow$ any $(n \times n)$ matrix

$x \rightarrow$ column vector

$x \rightarrow \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad A \rightarrow \begin{bmatrix} a \ldots & \\ & a_{n \times n} \end{bmatrix}$

$x^T . A = [x_1 \ldots x_n] \begin{bmatrix} a \ldots & \\ \vdots & a_{n \times n} \end{bmatrix}$

$= [x_1 \times a_{00} + x_2 \times a_{10} + x_3 \times a_{20} \ldots] \ldots (1)$

(will give a vector)

DATE / /

$\therefore \nabla_x (X^T A X) = (X^T A)_{vector} \cdot (X)_{vector}$

$\Rightarrow \nabla_x (X^T A X) = (X^T A) + (X^T \cdot A)^T$

$\Rightarrow$ To simplify $(X^T A)^T = (A^T X)$

$\therefore \nabla_x (X^T A X) = (X^T A)^T + (A^T \cdot X)$

$= (X A^T) + (A^T \cdot X)$

$= X (A^T + A) \quad \ldots \rightarrow \quad (2)$

$$\boxed{\therefore \nabla_x (X^T A X) = (A + A^T) X}$$

5 (c) To prove $\nabla_x (X^T A X) = 2AX$

$X \rightarrow$ column vector

$A \rightarrow$ any symmetric $n \times n$ matrix

since we know that

$\nabla_x (X^T A X) = (A + A^T) X$

(from eq (2))

Also, since A is symmetric matrix

$A^T = A \quad \therefore A + A^T = 2A$

$$\boxed{\therefore \nabla_x (X^T A X) = 2AX}$$

DATE / / /

(2) to prove $\nabla_x \left[(Ax+b)^T (Ax+b) \right]$
$= 2A^T (Ax+b)$

$x \rightarrow$ column vector
$A \rightarrow$ symmetric $n \times n$ matrix (ie $A^T = A$)
$b \rightarrow$ constant column vector.

$$(Ax+b)^T \cdot (Ax+b) = \left[(Ax^T + b^T) \cdot (Ax+b) \right]$$

$$\frac{d}{dx}(Ax)^T = A \quad \& \quad \frac{d}{dx} b^T = 0 \text{ (constant)}$$

$$\therefore \frac{d}{dx}\left[(Ax+b)^T \cdot (Ax+b) \right]$$

$$= (Ax+b) \cdot 2A^T$$

$$\boxed{\therefore \nabla_x \left[(Ax+b)^T (Ax+b) \right] = 2A^T (Ax+b)}$$