1a)

```python
def process_data(data, labels):
    """
    Preprocess a dataset of strings into vector representations.

    Parameters
    ----------
    data: numpy array
        An array of N strings.
    labels: numpy array
        An array of N integer labels.

    Returns
    -------
    train_X: numpy array
        Array with shape (N, D) of N inputs.
    train_Y:
        Array with shape (N,) of N labels.
    val_X:
        Array with shape (M, D) of M inputs.
    val_Y:
        Array with shape (M,) of M labels.
    test_X:
        Array with shape (M, D) of M inputs.
    test_Y:
        Array with shape (M,) of M labels.
    """

    # Split the dataset of string into train, validation, and test
    # Use a 70/15/15 split
    # train_test_split shuffles the data before splitting it
    # Stratify keeps the proportion of labels the same in each split

    # -- WRITE THE SPLITTING CODE HERE --
    # Preprocess each dataset of strings into a dataset of feature vectors
    # using the CountVectorizer function.
    # Note, fit the Vectorizer using the training set only, and then
    # transform the validation and test sets.

    train_X, t_X, train_Y, t_Y = train_test_split(data, labels, test_size= 0.3, stratify= labels)

    val_X, test_X, val_Y, test_Y = train_test_split(t_X, t_Y, test_size= 0.5, stratify= t_Y)

    # -- WRITE THE PROCESSING CODE HERE --

    # Return the training, validation, and test set inputs and labels

    vectorizer = CountVectorizer()

    train_X = vectorizer.fit_transform(train_X)

    train_X.toarray()

    val_X = vectorizer.transform(val_X)

    val_X.toarray()

    test_X = vectorizer.transform(test_X)

    test_X.toarray()


    # -- RETURN THE ARRAYS HERE --

    return train_X, train_Y, val_X, val_Y, test_X, test_Y
```

1 b)

```python
def select_knn_model(train_X, val_X, train_Y, val_Y):
    """
    Test k in {1, ..., 20} and return the a k-NN model
    fitted to the training set with the best validation loss.

    Parameters
    ----------
    train_X: numpy array
        Array with shape (N, D) of N inputs.
    val_X: numpy array
        Array with shape (M, D) of M inputs.
    train_Y: numpy array
        Array with shape (N,) of N labels.
    val_Y: numpy array
        Array with shape (M,) of M labels.

    Returns
    -------
    best_model : KNeighborsClassifier
        The best k-NN classifier fit on the training data
    and selected according to validation loss.
    best_k : int
        The best k value according to validation loss.
    """
    acc = 0
    model = None
    expected_k = 0
    for k in range(20):
        if k != 0:
            neigh = KNeighborsClassifier(n_neighbors=k)
            neigh.fit(train_X, train_Y)
            s = neigh.score(val_X, val_Y)
            if s > acc:
                acc = s
                model = neigh
                expected_k = k
    return model, expected_k
```

Output:

```
Selected K: 9
Test Acc: 0.6591836734693878
```

1 c)

The output when changing to metric to cosine is:

```
Selected K: 19
Test Acc: 0.746938775510204
```

The test accuracy when using cosine metric is around 0.75, which is greater than the Euclidean metric. Euclidean metric considers the distance between each vector (in terms of headline, it considers the difference of how many times each word appears). For cosine metric, it considers angle between each vector (in terms of headline, it considers whether certain word appear in the headline or not. To be specific, it considers whether certain word appear once or never occur). One of the important evidence to distinguish real news from fake news is the occurrence of some certain word (not necessarily how many times each word appears). As a result, a cosine metric fits more when we want to distinguish real news from fake news.

2 a) $E[R] = E[\sum_{i=1}^{d} z_i]$

$= \sum_{i=1}^{d} E[z_i]$

$= d\,E[z]$

$Var[R] = Var[\sum_{i=1}^{d} z_i]$

$= \sum_{i=1}^{d} Var[z_i] + \sum_{i=1}^{d} \sum_{j \neq i}^{d} Cov[z_i, z_j]$

$= \sum_{i=1}^{d} Var[z_i]$ (by independence of $z_1, z_2, \dots, z_d$

$Cov[z_i, z_j] = 0$ for $i \neq j$, $i, j \in \{1, 2, \dots d\}$)

$= d\, Var[z]$

3. let y be a random variable which represents the result of $h(x)$

Since T is fixed, so $y-T$ is also a random variable

Since $y_1, y_2, \dots, y_m$ are samples of y

so $(y_1-T), (y_2-T), \dots, (y_m-T)$ are samples of $y-T$

$$\text{so } \hat{E}((y-T)) = 1/m \sum_{i=1}^{m} (y_i - T)$$

$$\hat{E}((y-T)^2) = 1/m \sum_{i=1}^{m} (y_i - T)^2$$

$$\hat{E}(y-T)^2 = (1/m \sum_{i=1}^{m} (y_i - T))^2$$

$$= (1/m(\sum_{i=1}^{m} y_i - mT))^2$$

$$= (1/m \sum_{i=1}^{m} y_i - T)^2$$

$$Var(y-T) = \hat{E}((y-T)^2) - \hat{E}(y-T)^2 \geq 0$$

$$\text{so } 1/m \sum_{i=1}^{m} (y_i - T)^2 - (1/m \sum_{i=1}^{m} y_i - T)^2 \geq 0$$

$$\text{so } 1/2m \sum_{i=1}^{m} (y_i - T)^2 - 1/2 (1/m \sum_{i=1}^{m} y_i - T)^2 \geq 0$$

$$\text{so } 1/m \sum_{i=1}^{m} 1/2 (y_i - T)^2 - 1/2 (1/m \sum_{i=1}^{m} y_i - T)^2 \geq 0$$

$$\Rightarrow 1/m \sum_{i=1}^{m} 1/2 (h_i(x) - T)^2 - 1/2 (1/m \sum_{i=1}^{m} h_i(x) - T)^2 \geq 0$$

$$\Rightarrow 1/m \sum_{i=1}^{m} L(h_i(x), T) - L(\bar{h}(x), T) \geq 0$$

$$\text{so } L(\bar{h}(x), T) \leq 1/m \sum_{i=1}^{m} L(h_i(x), T)$$

4a) $R[y] = \sum_{Y \in \{0,1\}} \sum_{X \in \{0,1\}} L_{0-1}(y(x), Y)\, P(X = x, Y = Y)$

$= L_{0-1}(y(0), 0)\, P(0|0)\, P(X=0) + L_{0-1}(y(0), 1)\, P(1|0)\, P(X=0)$

$+ L_{0-1}(y(1), 0)\, P(0|1)\, P(X=1) + L_{0-1}(y(1), 1)\, P(1|1)\, P(X=1)$

$= (L_{0-1}(y(0), 0)\, P(0|0) + L_{0-1}(y(0), 1)\, P(1|0))\, P(X=0)$

$+ (L_{0-1}(y(1), 0)\, P(0|1) + L_{0-1}(y(1), 1)\, P(1|1))\, P(X=1)$

$= (\frac{1}{2} L_{0-1}(y(0), 0) + \frac{1}{2} L_{0-1}(y(0), 1))\, P(X=0)$

$+ (\frac{1}{2} L_{0-1}(y(1), 0) + \frac{1}{2} L_{0-1}(y(1), 1))\, P(X=1)$

By definition:

$y(0) = a_1 \Rightarrow y(0) \neq 1 - a_1$, $y(1) = b_1 \Rightarrow y(1) \neq 1 - b_1$ for $a_1, b_1 \in \{0,1\}$

So $L_{0-1}(y(0), 0) + L_{0-1}(y(0), 1) = 1$

$L_{0-1}(y(1), 0) + L_{0-1}(y(1), 1) = 1$

So $R[y] = \frac{1}{2} P(X=0) + \frac{1}{2} P(X=1)$

$= \frac{1}{2}(P(X=0) + P(X=1))$

Since $X \in \{0, 1\}$

So $P(X=0) + P(X=1) = 1$

So $R[y] = \frac{1}{2}$

4b) $y^*(x) = \underset{\tau \in \{0,1\}}{\arg\max}\ p(\tau | x)$

$y^*(x) = 1$ when $p(1|x) > p(0|x)$ $\forall x \in \{0,1\}$

$y^*(x) = 0$ when $p(0|x) > p(1|x)$ $\forall x \in \{0,1\}$

$$\begin{aligned} R[y^*] = & L_{0\text{-}1}[y^*(0), 0]\, p(0|0) \cdot p(x=0) \\ & + L_{0\text{-}1}[y^*(0), 1]\, p(1|0) \cdot p(x=0) \\ & + L_{0\text{-}1}[y^*(1), 0]\, p(0|1) \cdot p(x=1) \\ & + L_{0\text{-}1}[y^*(1), 1]\, p(1|1) \cdot p(x=1) \end{aligned}$$

so $y^*(0) = a_1$, where $a_1 \in \{0,1\}$ and

$y^*(1) = a_2$, where $a_2 \in \{0,1\}$

Take $b_1 = 1 - a_1$ so $p(a_1 | 0) > p(b_1 | 0)$

Take $b_2 = 1 - a_2$ so $p(a_2 | 1) > p(b_2 | 1)$

so rewrite
$$\begin{aligned} R[y^*] = & L_{0\text{-}1}[y^*(0), a_1]\, p(a_1 | 0)\, p(x=0) + \\ & L_{0\text{-}1}[y^*(0), b_1]\, p(b_1 | 0)\, p(x=0) + \\ & L_{0\text{-}1}[y^*(1), a_2]\, p(a_2 | 1)\, p(x=1) + \\ & L_{0\text{-}1}[y^*(1), b_2]\, p(b_2 | 1)\, p(x=1) \end{aligned}$$

since $y^*(0) = a_1$, $y^*(1) = a_2$

so $L_{0\text{-}1}[y^*(0), a_1] = 0$ $\quad L_{0\text{-}1}[y^*(1), a_2] = 0$

$L_{0\text{-}1}[y^*(0), b_1] = 1$ $\quad L_{0\text{-}1}[y^*(1), b_2] = 1$

so $R[y^*] = p(b_1 | 0)\, p(x=0) + p(b_2 | 1)\, p(x=1)$

Let y be a predictor and $y(x) \neq y^*(x)$ for some $x \in \{0,1\}$

There are 3 cases:

Case 1: $y(0) = b_1$, $y(1) = a_2$

$L_{0-1}[y(0), a_1] = 1$, $L_{0-1}[y(1), b_2] = 1$

$R[y] = P(a_1|0) P(x=0) + P(b_2|1) P(x=1)$

Since $y^*(0) = a_1$ so $P(a_1|0) > P(b_1|0)$, and $P(x=0) > 0$

so $R[y] - R[y^*] = (P(a_1|0) - P(b_1|0)) P(x=0) > 0$

so $R[y^*] < R[y]$

Case 2: $y(0) = a_1$, $y(1) = b_2$

$L_{0-1}[y(0), b_1] = 1$, $L_{0-1}[y(1), a_2] = 1$

$R[y] = P(b_1|0) P(x=0) + P(a_2|1) P(x=1)$

Since $y^*(1) = a_2$, so $P(a_2|1) > P(b_2|1)$, $P(x=1) > 0$

so $R[y] - R[y^*] = (P(a_2|1) - P(b_2|1)) P(x=1) > 0$

so $R[y^*] < R[y]$

Case 3: $y(0) = b_1$, $y(1) = b_2$

$L_{0-1}[y(0), a_1] = 1$, $L_{0-1}[y(1), a_2] = 1$

$R[y] = P(a_1|0) P(x=0) + P(a_2|1) P(x=1)$

$$\text{so } R[y] - R[y^*] = (P(a_1|0) - P(b_1|0)) P(x=0) + (P(a_2|1) - P(b_2|1)) P(x=1) > 0$$

so $R[y^*] < R[y]$

so $\forall y$ s.t. $y(x) \neq y^*(x)$ for some $x \in \{0, 1\}$

$R[y^*] < R[y]$

so $R[y^*] = R[y] = P(b_1|0) P(x=0) + P(b_2|1) P(x=1)$

only if y have the same prediction with y^*

so $R[y^*] \leq R[y]$ with equality only if $y^*(x) = y(x)$ for all $x \in \{0, 1\}$

4c) $P(X=0, Y=0) = 1/8$, $P(X=0, Y=1) = 3/8$, $P(X=1, Y=0) = 1/4$, $P(X=1, Y=1) = 1/4$

There are 4 different predictors y_1, y_2, y_3, y_4

$y_1(0) = 1$, $y_1(1) = 0$ $\Longrightarrow L_{0\text{-}1}[y_1(0), 0] = 1$, $L_{0\text{-}1}[y_1(0), 1] = 0$

$L_{0\text{-}1}[y_1(1), 0] = 0$, $L_{0\text{-}1}[y_1(1), 1] = 1$

$y_2(0) = 1$, $y_2(1) = 1$ $\Longrightarrow L_{0\text{-}1}[y_2(0), 0] = 1$, $L_{0\text{-}1}[y_2(0), 1] = 0$

$L_{0\text{-}1}[y_2(1), 0] = 1$, $L_{0\text{-}1}[y_2(1), 1] = 0$

$y_3(0) = 0$, $y_3(1) = 0$ $\Longrightarrow L_{0\text{-}1}[y_3(0), 0] = 0$, $L_{0\text{-}1}[y_3(0), 1] = 1$

$L_{0\text{-}1}[y_3(1), 0] = 0$, $L_{0\text{-}1}[y_3(1), 1] = 1$

$y_4(0) = 0$, $y_4(1) = 1$ $\Longrightarrow L_{0\text{-}1}[y_4(0), 0] = 0$, $L_{0\text{-}1}[y_4(0), 1] = 1$

$L_{0\text{-}1}[y_4(1), 0] = 1$, $L_{0\text{-}1}[y_4(1), 1] = 0$

$R[y_1] = \sum_{Y\in\{0,1\}} \sum_{X\in\{0,1\}} L_{0\text{-}1}(y_1(x), y)\, P(X=x, Y=y)$

$= P(X=0, Y=0) + P(X=1, Y=1) = 1/8 + 1/4 = 3/8$

$R[y_2] = P(X=0, Y=0) + P(X=1, Y=0) = 3/8$

$R[y_3] = P(X=0, Y=1) + P(X=1, Y=1) = 5/8$

$R[y_4] = P(X=0, Y=1) + P(X=1, Y=0) = 5/8$

So in this case, there are exactly two optimal predictors. y_1 and y_2