Exercise 1

We can define the *edit distance* $d_{\text{edit}}(w, w') : \Sigma^2 \to \mathbb{R}$ as follows. (Let $w = w_1 \dots w_n$ and $w' = w'_1 \dots w'_m$)

$$d_{\text{edit}}(w, w') \mapsto \begin{cases} |w| & \text{if } |w'| = 0 \\ |w'| & \text{if } |w| = 0 \\ d_{\text{edit}}(w_2 \dots w_n, w'_2 \dots w'_m) & \text{if } w_1 = w'_1 \\ 1 + \min \begin{cases} d_{\text{edit}}(w_2 \dots w_n, w') \\ d_{\text{edit}}(w, w'_2 \dots w'_m) \\ d_{\text{edit}}(w_2 \dots w_n, w'_2 \dots w'_m) \end{cases} & \text{otherwise} \end{cases}$$

As this definition of d_{edit} works by removing at most the first character of each word, we can proof by induction the length of $x, y, z \in \Sigma$, that d_{edit} is a metric on Σ:

- Let $|x| = |y| = |z| = 0$. Therefore, also $x = y = z$.
  Then $0 \leq d_{\text{edit}} = 0$. Thus, Nonnegativity is given.
  Since $x = y$, also $d_{\text{edit}}(x, y) = d_{\text{edit}}(y, x)$. Thus, Symmetry is given.
  Since $x = y = z$, the Triangle Inequality $d_{\text{edit}}(x, z) \leq d_{\text{edit}}(x, y) + (d_{\text{edit}})(y, z) \Leftrightarrow 0 \leq 0 + 0$ is given.

- Let $x = x_1 \dots x_n$, $y = y_1 \dots y_m$, and $z = z_1 \dots z_o$, $n, m, o \geq 1$. For $x' = x_2 \dots x_n$, $y' = y_2 \dots y_m$, and $z' = z_2 \dots z_o$ Nonnegativity, Symmetry, and the Triangle Inequality of d_{edit} is given.

- Since $n, m \geq 1$, the second rule of Nonnegativity, namely $d_{\text{edit}}(x, y) \Leftrightarrow x = y$ does not apply here.
  Since all $d_{\text{edit}}(x', y'), d_{\text{edit}}(x', y), d_{\text{edit}}(x, y')$ are non-negative, by definition of d_{edit}, $d_{\text{edit}}(x, y)$ must be non-negative as well. Therefore, the Nonnegativity of d_{edit} is proven.

- If $x_1 = y_1$, then $d_{\text{edit}}(x, y) = d_{\text{edit}}(x', y') = d_{\text{edit}}(y', x') = d_{\text{edit}}(y, x)$
  If $x_1 \neq y_1$, then

- Triangle Inequality

Exercise 2

Result (see Appendix for code):

Classification: k=2 Manhattan Distance
Test (1, -2, 0): Prediction 1
Test (4, -0.5, 2): Prediction -1
Test (1, 1.5, -2.5): Prediction 0
Test (-2, -1, -2): Prediction 0
Test (-4, -1, -1): Prediction 0

Classification: k=3 Manhattan Distance
Test (1, -2, 0): Prediction 1
Test (4, -0.5, 2): Prediction -1
Test (1, 1.5, -2.5): Prediction 1
Test (-2, -1, -2): Prediction -1
Test (-4, -1, -1): Prediction 1

Classification: k=2 Euclidean Distance
Test (1, -2, 0): Prediction 1
Test (4, -0.5, 2): Prediction -1
Test (1, 1.5, -2.5): Prediction 1
Test (-2, -1, -2): Prediction 0
Test (-4, -1, -1): Prediction 0

Classification: k=3 Euclidean Distance
Test (1, -2, 0): Prediction 1
Test (4, -0.5, 2): Prediction -1
Test (1, 1.5, -2.5): Prediction 1
Test (-2, -1, -2): Prediction 1
Test (-4, -1, -1): Prediction -1

Appendix

Code for Exercise 2

```python
from math import sqrt
#from statistics import mode

training_set = [
    ((-4, -2.1, -1), -1),
    ((-3.6, -1.4, 0.2), 1),
    ((1, -0.2, -0.3), 1),
    ((0.3, -0.5, -0.5), 1),
    ((-2, -3.5, -1), -1),
    ((-4.2, -4, 0.2), 1),
    ((-1.3, -0.1, -3), 1),
    ((-0.7, 0.9, -0.7), 1),
    ((1, 2, 1.4), 1),
    ((2.6, -1.5, 0.2), 1),
    ((2, 4.3, -0.7), -1),
    ((0.6, 0.4, 0.2), -1),
    ((2.9, -1.7, 3.6), -1),
    ((3.6, 0.4, -2.5), -1),
    ((1.2, 4, 1.2), -1),
    ((-1, 0.5, 0.5), -1),
    ((3, 2.7, 2.3), -1),
    ((4, -3, 2.2), -1),
    ((0.1, 0.1, 3.5), -1),
    ((2.8, 1.2, 2.4), -1)
]

test_set = [
    (1, -2, 0),
    (4, -0.5, 2),
    (1, 1.5, -2.5),
    (-2, -1, -2),
    (-4, -1, -1)
]


def manhattan_distance(x, y):
    assert len(x) == len(y)
    sum = 0
    for i in range(len(x)):
        sum += abs(x[i] - y[i])
    return sum


def euclidean_distance(x, y):
    assert len(x) == len(y)
    sum = 0
    for i in range(len(x)):
        sum += pow(x[i] - y[i], 2)
    sum = sqrt(sum)
    return sum


def get_k_nearest_neighbors(training_set, test, k, distance_func):
    assert callable(distance_func)
    distances = list()
    for data in training_set:
        distance = distance_func(data[0], test)
        distances.append((data, distance))
    distances.sort(key=lambda x: x[1]) # sort by distance
    neighbors = list()
    for i in range(k): # get data of k nearest neighbors
        neighbors.append(distances[i][0])
    return neighbors


def predict_classificataion(training_set, test, k, distance_func):
    assert callable(distance_func)
    neighbors = get_k_nearest_neighbors(training_set, test, k, distance_func)
    classifications = [neighbor[1] for neighbor in neighbors] # list of all classifications
    #prediction = mode(classifications) # get classification most often in k nearest neighbors
    #return prediction
    count_neg = classifications.count(-1)
    count_pos = classifications.count(1)
    assert count_neg + count_pos == k
    if count_pos > count_neg:
        return 1
    if count_pos < count_neg:
        return -1
    if count_pos == count_neg:
        return 0


if __name__ == ’__main__’:
    print(’Classification: k=2 Manhattan Distance’)
    for test in test_set:
        prediction = predict_classificataion(
            training_set, test, 2, manhattan_distance)
        print(f’Test {test}: Prediction {prediction}’)
    print(’\n’)
    print(’Classification: k=3 Manhattan Distance’)
    for test in test_set:
        prediction = predict_classificataion(
            training_set, test, 3, manhattan_distance)
        print(f’Test {test}: Prediction {prediction}’)
    print(’\n’)
    print(’Classification: k=2 Euclidean Distance’)
    for test in test_set:
        prediction = predict_classificataion(
            training_set, test, 2, euclidean_distance)
        print(f’Test {test}: Prediction {prediction}’)
    print(’\n’)
    print(’Classification: k=3 Euclidean Distance’)
    for test in test_set:
        prediction = predict_classificataion(
            training_set, test, 3, euclidean_distance)
        print(f’Test {test}: Prediction {prediction}’)
```