k 최근접 이웃 (kNN) 과제

1. k-NN 분류기

In [1]:

```
from typing import List
from collections import Counter
```

1.1 투표 함수

In [2]:

```
def majority_vote(labels: List[str]) -> str:
    """Assumes that labels are ordered from nearest to farthest."""
    vote_counts = Counter(labels)
    winner, winner_count = vote_counts.most_common(1)[0]
    num_winners = len([count
                       for count in vote_counts.values()
                       if count == winner_count])

    if num_winners == 1:
        return winner                     # unique winner, so return it
    else:
        return majority_vote(labels[:-1]) # try again without the farthest
```

1.2 데이터 포인트

In [3]:

```
from typing import NamedTuple
from scratch.linear_algebra import Vector, distance
```

In [4]:

```
class LabeledPoint(NamedTuple):
    point: Vector
    label: str
```

1.3 k-NN 분류기

In [5]:

```
def knn_classify(k: int,
                 labeled_points: List[LabeledPoint],
                 new_point: Vector) -> str:

    # Order the labeled points from nearest to farthest.
    by_distance = sorted(labeled_points,
                         key=lambda lp: distance(lp.point, new_point))

    # Find the labels for the k closest
    k_nearest_labels = [lp.label for lp in by_distance[:k]]

    # and let them vote.
    return majority_vote(k_nearest_labels)
```

2. 유방 암 데이터 분류

In [6]:

```
import matplotlib.pyplot as plt
import os
from typing import Dict
import csv
from collections import defaultdict
```

2.1 데이터셋 다운로드

위스콘신 유방암 진단 데이터셋 (Wisconsin Breast Cancer Diagnostic dataset)
https://www.kaggle.com/uciml/breast-cancer-wisconsin-data (https://www.kaggle.com/uciml/breast-cancer-wisconsin-data)

In [7]:

```
import requests

data = requests.get("https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wiscons
dataset_path = os.path.join('data', 'wdbc.data')

with open(dataset_path, "w") as f:
  f.write(data.text)
```

2.2 데이터 파싱 (Q)

In [8]:

```
def parse_cancer_row(row: List[str]) -> LabeledPoint:
    measurements = [float(value) for value in row[2:]]
    label = row[1]
    return LabeledPoint(measurements, label)
```

2.3 데이터 읽기

In [9]:

```
with open(dataset_path) as f:
    reader = csv.reader(f)
    cancer_data = [parse_cancer_row(row) for row in reader if row]
```

데이터 탐색 단계의 시각화를 위해 데이터 행렬 생성

In [10]:

```
columns = [
    "radius_mean", "texture_mean", "perimeter_mean", "area_mean", "smoothness_mean",
    "compactness_mean", "concavity_mean", "points_mean", "symmetry_mean", "dimension_mean",
    "radius_se", "texture_se", "perimeter_se", "area_se", "smoothness_se",
    "compactness_se", "concavity_se", "points_se", "symmetry_se", "dimension_se",
    "radius_worst", "texture_worst", "perimeter_worst", "area_worst", "smoothness_worst",
    "compactness_worst", "concavity_worst", "points_worst", "symmetry_worst", "dimension_worst",
    ]
```

In [11]:

```
from scratch.linear_algebra import get_column, shape
def make_matrix(dataset):
    matrix = []
    for datapoint in dataset:
        matrix.append(datapoint.point)
    return matrix
```

In [12]:

```
cancer_matrix = make_matrix(cancer_data)
print(shape(cancer_matrix))
```

```
(569, 30)
```

2.4 데이터 탐색

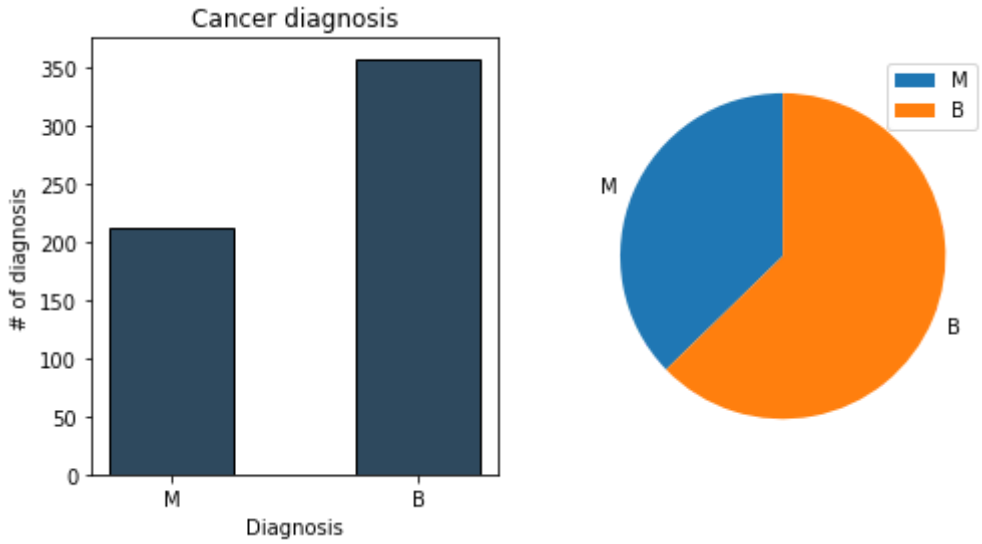
2.3.1 클래스 비율 확인

In [13]:

```
label_type = defaultdict(int)
for cancer in cancer_data:
    label_type[cancer.label] += 1
```

In [14]:

```
plt.figure(figsize=(8,4))
plt.subplot(1, 2, 1)
plt.bar(label_type.keys(),
        label_type.values(),
        0.5,
        facecolor="#2E495E",
        edgecolor=(0, 0, 0))                # Black edges for each bar

plt.xlabel("Diagnosis")
plt.ylabel("# of diagnosis")
plt.title("Cancer diagnosis")

plt.subplot(1, 2, 2)
pies = plt.pie(label_type.values(),
               labels=label_type.keys(),
               startangle=90)
plt.legend()
plt.show()
```



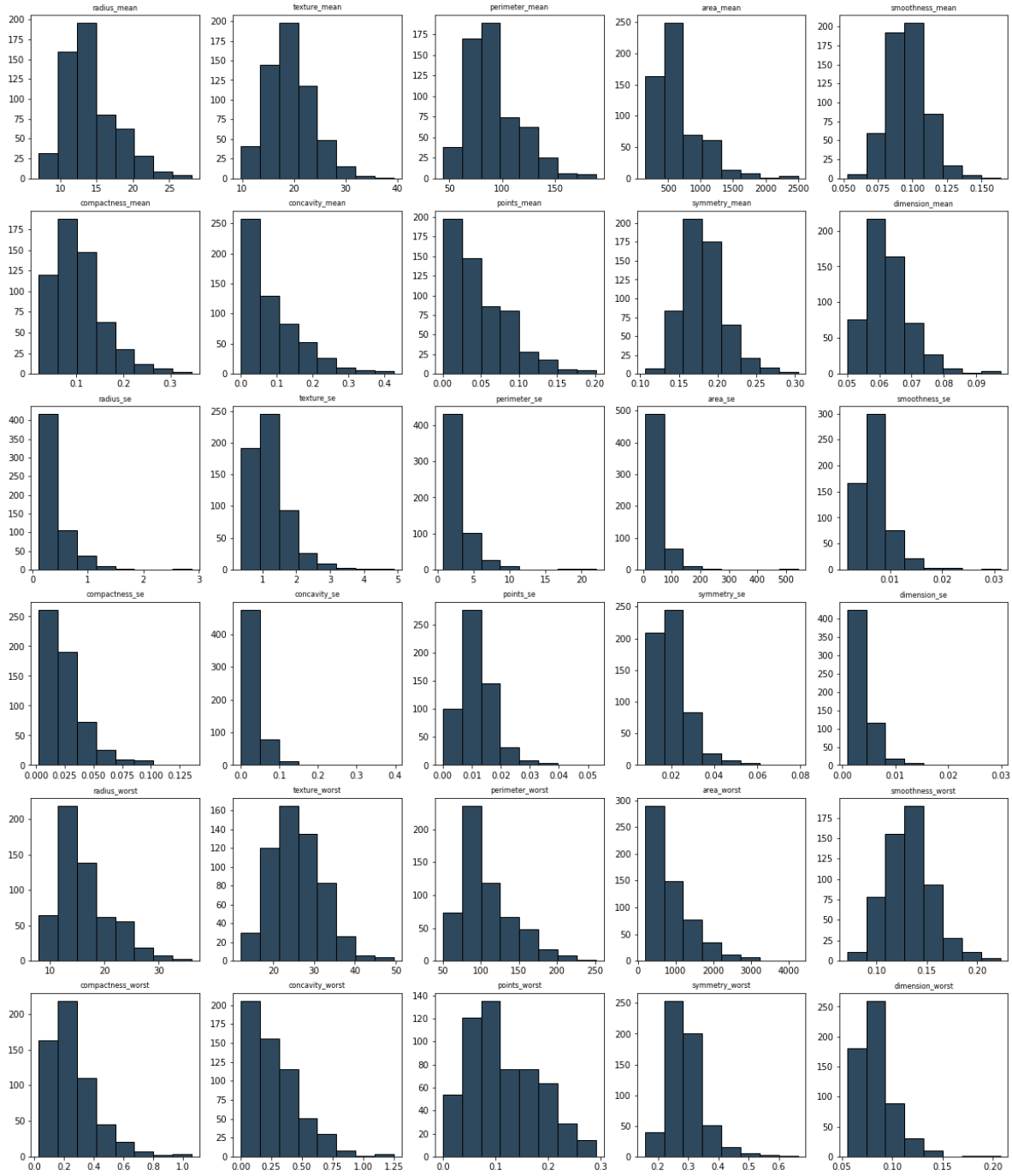
2.3.2 특징 별 히스토그램

In [15]:

```
def histogram(ax, col : int):

    n, bins, patches = ax.hist(get_column(cancer_matrix, col),
                               8,
                               facecolor="#2E495E",
                               edgecolor=(0, 0, 0))

    ax.set_title(columns[col], fontsize=8)
```

In [16]:

```
from matplotlib import pyplot as plt
num_rows = 6
num_cols = 5

fig, ax = plt.subplots(num_rows, num_cols, figsize=(num_cols*4, num_rows*4))
for row in range(num_rows):
    for col in range(num_cols):
        histogram(ax[row][col], num_cols * row + col)
plt.show()
```



2.3.3 특징 쌍 별 산포도

In [17]:

```
points_by_diagnosis: Dict[str, List[Vector]] = defaultdict(list)
for cancer in cancer_data:
    points_by_diagnosis[cancer.label].append(cancer.point)
```

In [18]:

```
start = 0
end = start + 10
pairs = [(i, j) for i in range(start, end) for j in range(i+1, end) if i < j]
print(pairs)
marks = ['+', '.']
```

```
[(0, 1), (0, 2), (0, 3), (0, 4), (0, 5), (0, 6), (0, 7), (0, 8), (0, 9), (1, 2), (1,
3), (1, 4), (1, 5), (1, 6), (1, 7), (1, 8), (1, 9), (2, 3), (2, 4), (2, 5), (2, 6),
(2, 7), (2, 8), (2, 9), (3, 4), (3, 5), (3, 6), (3, 7), (3, 8), (3, 9), (4, 5), (4,
6), (4, 7), (4, 8), (4, 9), (5, 6), (5, 7), (5, 8), (5, 9), (6, 7), (6, 8), (6, 9),
(7, 8), (7, 9), (8, 9)]
```

In [19]:

```
from matplotlib import pyplot as plt
num_rows = 9
num_cols = 5

fig, ax = plt.subplots(num_rows, num_cols, figsize=(num_cols*3, num_rows*3))

for row in range(num_rows):
    for col in range(num_cols):
        i, j = pairs[num_cols * row + col]
        ax[row][col].set_title(f"{columns[i]} vs {columns[j]}", fontsize=8)
        ax[row][col].set_xticks([])
        ax[row][col].set_yticks([])

        for mark, (diagnosis, points) in zip(marks, points_by_diagnosis.items()):
            xs = [point[i] for point in points]
            ys = [point[j] for point in points]
            ax[row][col].scatter(xs, ys, marker=mark, label=diagnosis)

ax[-1][-1].legend(loc='lower right', prop={'size': 6})
plt.show()
```


radius_mean vs texture_mean
radius_mean vs perimeter_mean
radius_mean vs area_mean
radius_mean vs smoothness_mean
radius_mean vs compactness_mean
radius_mean vs concavity_mean
radius_mean vs points_mean
radius_mean vs symmetry_mean
radius_mean vs dimension_mean
texture_mean vs perimeter_mean
texture_mean vs area_mean
texture_mean vs smoothness_mean
texture_mean vs compactness_mean
texture_mean vs concavity_mean
texture_mean vs points_mean
texture_mean vs symmetry_mean
texture_mean vs dimension_mean
perimeter_mean vs area_mean
perimeter_mean vs smoothness_mean
perimeter_mean vs compactness_mean
perimeter_mean vs concavity_mean
perimeter_mean vs points_mean
perimeter_mean vs symmetry_mean
perimeter_mean vs dimension_mean
area_mean vs smoothness_mean
area_mean vs compactness_mean
area_mean vs concavity_mean
area_mean vs points_mean
area_mean vs symmetry_mean
area_mean vs dimension_mean
smoothness_mean vs compactness_mean
smoothness_mean vs concavity_mean
smoothness_mean vs points_mean
smoothness_mean vs symmetry_mean
smoothness_mean vs dimension_mean
compactness_mean vs concavity_mean
compactness_mean vs points_mean
compactness_mean vs symmetry_mean
compactness_mean vs dimension_mean
concavity_mean vs points_mean
concavity_mean vs symmetry_mean
concavity_mean vs dimension_mean
points_mean vs symmetry_mean
points_mean vs dimension_mean
symmetry_mean vs dimension_mean
M
B

2.5 데이터셋 분리 (Q)

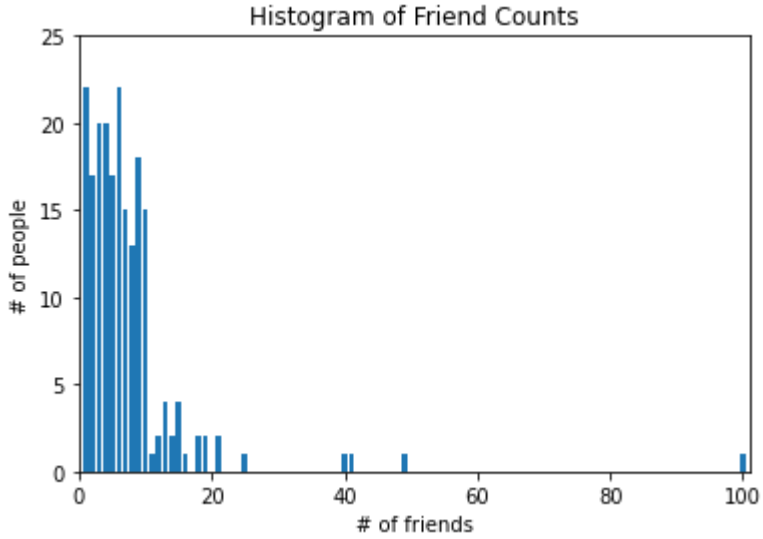
In [20]:

```
import random
from scratch.machine_learning import split_data

random.seed(12)
cancer_train, cancer_test = split_data(cancer_data, 0.70)
```

2.6 데이터 표준화 (Standardization)

In [21]:

```
from scratch.working_with_data import scale, rescale

def normalization(dataset):
    return rescale(make_matrix(dataset))
```



In [22]:

```
cancer_train_matrix = normalization(cancer_train)
cancer_test_matrix = normalization(cancer_test)
```

2.7 예측 (Q)

In [23]:

```
from typing import Tuple

def prediction(k : int) -> Tuple[float, Dict[Tuple[str, str], int]]:
    num_correct = 0
    confusion_matrix = defaultdict(int)

    for cancer in cancer_test:
        predicted = knn_classify(k, cancer_train, cancer.point)
        actual = cancer.label

        if predicted == actual:
            num_correct += 1

        confusion_matrix[(predicted, actual)] += 1
    pct_correct = num_correct / len(cancer_test)

    return pct_correct, confusion_matrix
```

2.8 엘보 방법 (Elbow method)으로 k 선정 (Q)

In [24]:

```
k_candidate = (k for k in range(1, 30))
optimal_k = 0

acc_list : List[float] = []
for k in k_candidate:
    accuracy, confusion_matrix = prediction(k)
    acc_list.append(accuracy)
    if k == 1: early_acc, optimal_k = acc_list[0], 1
    else:
        if early_acc < acc_list[k-1]: early_acc, optimal_k = acc_list[k-1], k

print("")
print("Optimal k = ", optimal_k)
plt.plot(acc_list)
plt.show()
```

```
Optimal k =  9
```