Homework Simple Machine Learning exercise.

Get your input data

Download the dataset from Kaggle.

https://www.kaggle.com/datasets/trolukovich/nutritional-values-for-common-foods-and-products

It is also included in my solution on Github:
https://github.com/SonnyBurnett/simple-machine-learning

Create a new Python project with your favorite Python IDE, or just use vi(m) if you must.

Read the input data from the csv file and put it in a Panda Dataframe.

```
import pandas as pd

df = pd.read_csv (file_name)
```

Clean the data.

You can use: 'Calories', 'Protein', 'Fat', 'Sat.Fat', 'Fiber', and 'Carbs' as predictors.

Make sure all fields contain numerical values only (float, int). Otherwise, our algorithms will crash.

Hint: use the “map” method of Pandas.

```
df[‘Calories’] = df[‘Calories’].map(fix_function)
```

Make numerical categories

Now we need to check the categories. These must be converted to numbers, because our algorithms work with numbers only.

We also need to combine a few categories in this dataset. Fruits and Vegetables are grouped in alphabetical categories, which is not what we want.

Decide which categories you want to use. You can even combine categories and create new ones.

We might not use all the rows (categories) in the dataset. Remove the rows you don't need.

Hint:

After you converted the category to a numerical value you can easily drop the category-values you don't need. In this example all categories > 4 are dropped.

```
indexCat = df[(df['Category'] > 4)].index

df.drop(indexCat, inplace=True)
```

Normalize the data.

This step is optional, but it is interesting to see the impact of this process.

Normalizing means the ranges of the data (the predictors) is the same, to avoid one range of data dominating the other ones. This is how you can do that.

```
from sklearn import preprocessing

d = preprocessing.normalize(df)

normalized_df = pd.DataFrame(d)
```

Split the data in a train-set and a test-set.

We want to use around 80% of our data for training and 20% for testing.

We need to make a random split to make sure both sets are representative for the whole collection.

You can use train_test_split but if you do that you first have to do the next 2 steps because it eats numpy instead of pandas, and it wants the predictors (X) and the outcomes (y) splitted up-front.

```
from sklearn.model_selection import train_test_split
```

Split labels, predictors and outcomes

We need to split the predictors, the outcome and the labels.

We will end up with 6 sets of data!!

1. The names (labels) of the food that we will use for training
2. The predictors for training
3. The outcome, meaning the category of that food for training
4. The names (labels) of the food that we will use for testing
5. The predictors for testing
6. The outcome, meaning the category of that food for testing

The common names for these datasets are:
X_train, = predictors train
y_train, = outcomes train
X_test, = predictors test
y_test = outcomes test
We use the labels for reporting purposes only. They are not needed for the algorithm. But without them the specific outcomes don't make much sense.

Hint: I haven't seen an easy way to use train_test_split and still have the labels available, so I just sorted the dataset and split it by myself. But, a better split can give better predictions.

Convert panda dataframe to numpy

X_train, y_train, X_test and y_test need to be converted to numpy.

Hint:

```
X_test = preprocessing.normalize(test_set[['Calories', 'Protein']].to_numpy())
```

Train your model

1. Create an empty classifier (model).
2. Train your model with the "fit" method.
3. Use your trained model to predict the test-set. (with the "predict" method).
4. Display the results (accuracy score) to see how good the model is.

Hints:

```
from sklearn.neighbors import KneighborsClassifier

from sklearn.metrics import accuracy_score

knn5 = KNeighborsClassifier(n_neighbors=5)
```

```
knn5.fit(X_train, y_train)

y_pred_5 = knn5.predict(y_test)
```

Make your own model (find the best algorithm) and try to make the best possible prediction between veggie and non-veggie food. The highest score wins.
Bonus points for nice reports.