PROGRAMMING ASSIGNMENT 5

PERSONALITY CLASSIFICATION WITH K-NEAREST NEIGHBOR ALGORITHM

Teaching Assistant: Burcu YALÇINER

Due Date: 16.01.2023(23:00)

1. INTRODUCTION

The aim of this experiment is to make you get familiar machine learning problem and Python data science libraries such as Pandas, NumPy and Matplotlib. In this experiment, you are expected to implement K-Nearest Neighbor Algorithm to solve a real word classification problem with a data.

2. BACKGROUND

2.1. K-Nearest Neighbor Algorithm

The k-nearest neighbor algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. While it can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another.

For **classification problems**, a class label is assigned on the basis of a majority vote—i.e. the label that is most frequently represented around a given data point is used. While this is technically considered "plurality voting", the term, "majority vote" is more commonly used in literature. The distinction between these terminologies is that "majority voting" technically requires a majority of greater than 50%, which primarily works when there are only two categories. When you have multiple classes—e.g. four categories, you don't necessarily need 50% of the vote to make a conclusion about a class; you could assign a class label with a vote of greater than 25%.

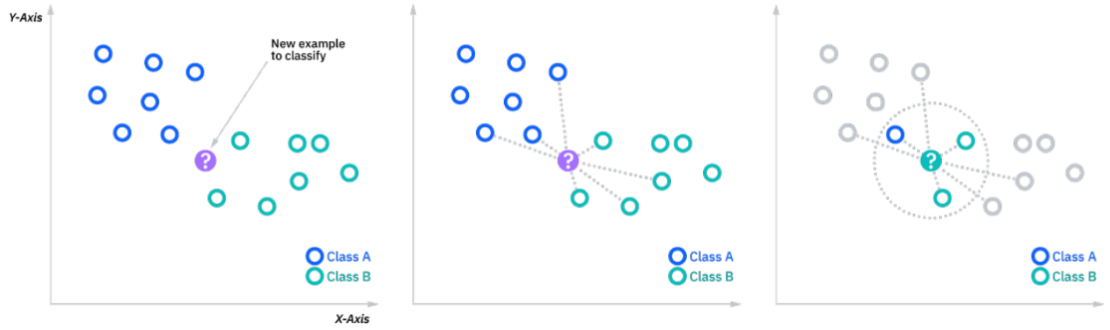


Figure 1. KNN Classification Diagram

2.2. Compute K-Nearest Neighbor: Distance Metric

The goal of the k-nearest neighbor algorithm is to identify the nearest neighbors of a given query point, so that we can assign a class label to that point. In order to do this, KNN has a few requirements:

2.2.1. Determine your distance metrics

In order to determine which data points are closest to a given query point, the distance between the query point and the other data points will need to be calculated. These distance metrics help to form decision boundaries, which partitions query points into different regions.

While there are several distance measures that we can choose from, you should use Euclidean distance in this assignment.

Euclidean distance (p=2): This is the most commonly used distance measure, and it is limited to real-valued vectors. Using the below formula, it measures a straight line between the query point and the other point being measured.

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2}$$

2.2.2. Compute KNN: defining k

The k value in the k-NN algorithm defines how many neighbors will be checked to determine the classification of a specific query point. For example, if k=1, the instance will be assigned to the same class as its single nearest neighbor. Defining k can be a balancing act as different values can lead to overfitting or underfitting. Lower values of k can have high variance, but low bias, and larger values of k may lead to high bias and lower variance. The choice of k will largely depend on the input data as data with more outliers or noise will likely perform better with higher values of k. Overall, it is recommended to have an odd number for k to avoid ties in classification, and cross-validation tactics can help you choose the optimal k for your dataset.

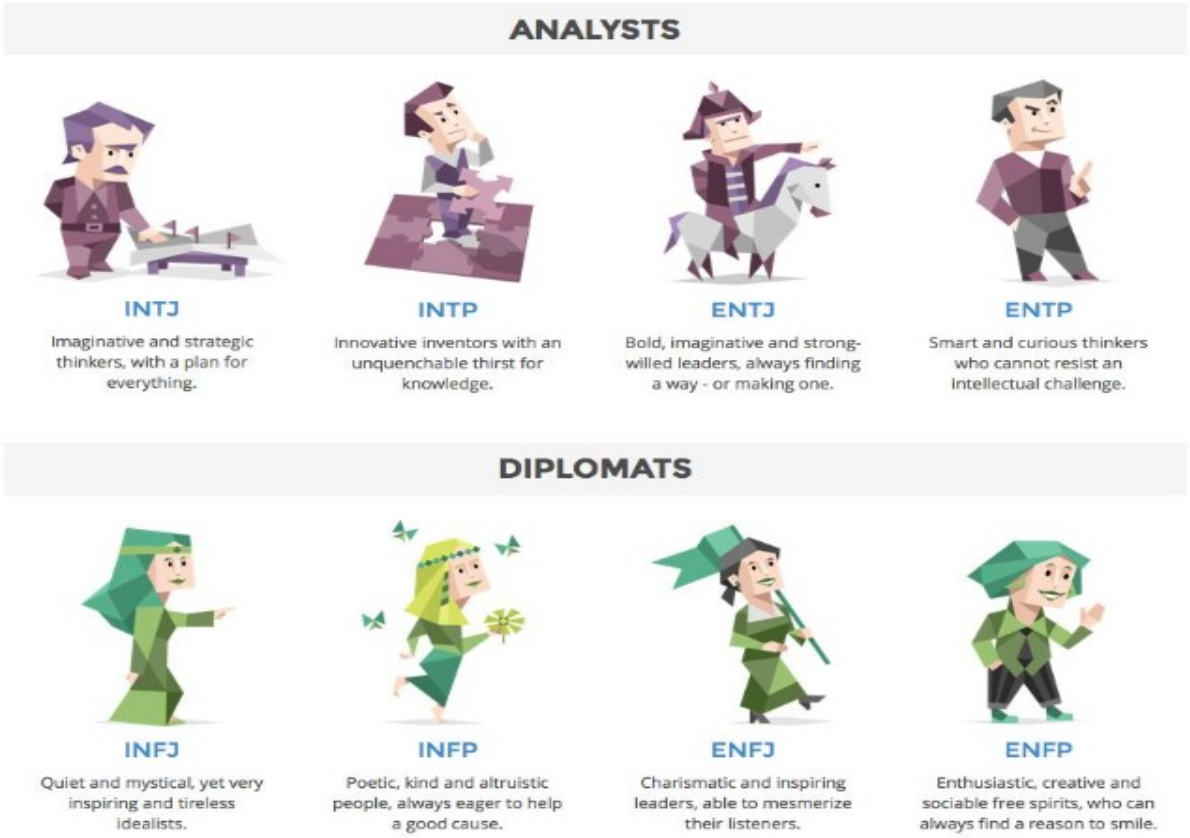
3. ASSIGNMENT

3.1. Packages

- **pandas** is the fundamental package for data manipulation and analysis. You will use it to analyze and manipulate tabular data in DataFrame(s).
- **numpy** is the fundamental package for scientific computing with Python. You will use it to perform a wide variety of mathematical operations on data/dataset.
- **math** is a library to make basic mathematical and statistical operations on DataFrame(s).
- **random** is a library used just for creating random integer in a given range.

In this assignment, you will implement a K-Nearest Neighbor Algorithm to classify different personality types (16 personalities) of people. Furthermore, you will extend your implementation for the same problem by implementing weighted K-Nearest Neighbor Algorithm.

3.2. Personality Classification Data



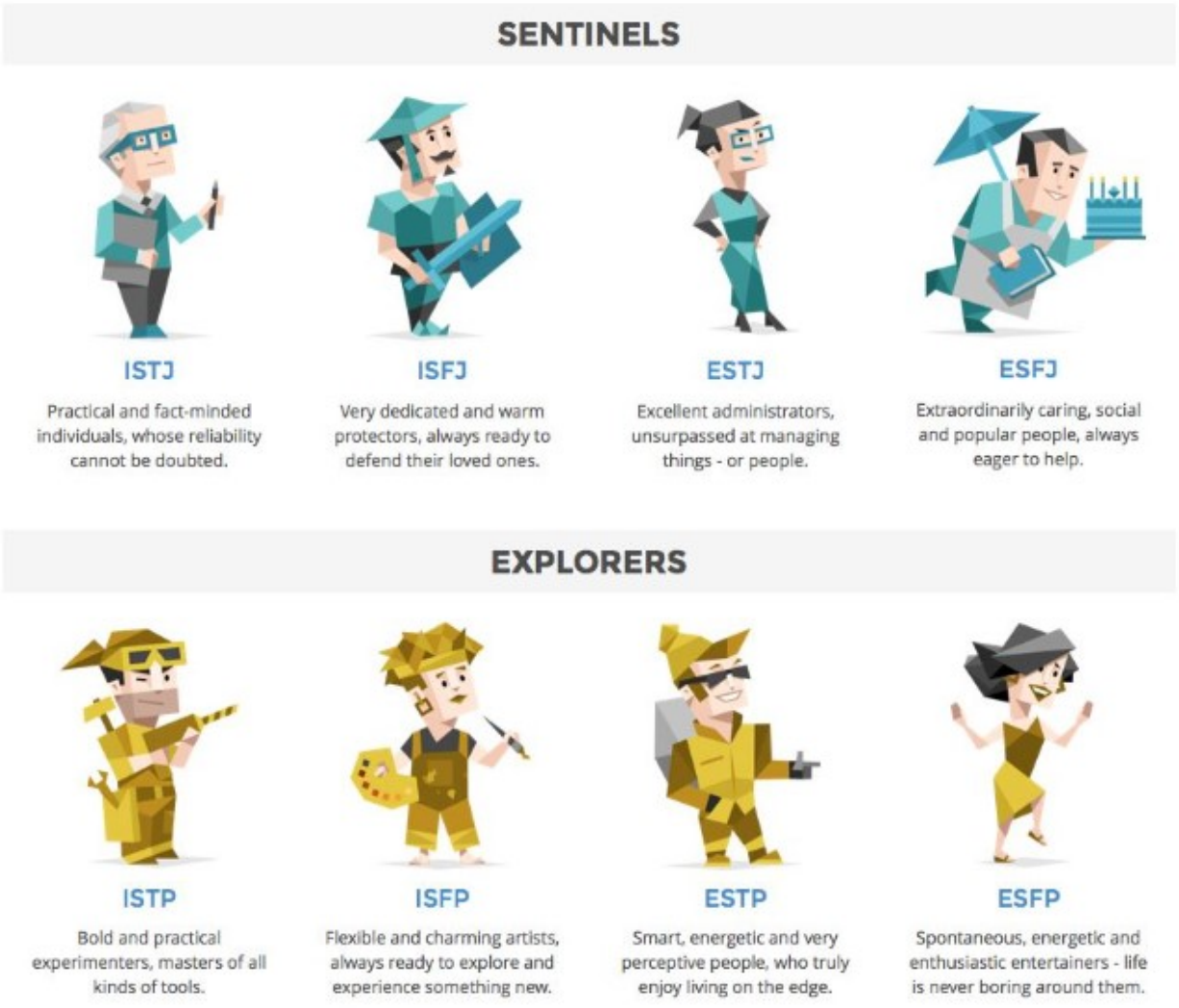


Figure 2. Personalities

The personality classification data will be downloaded from the given link.

This data contains the questions from the 16 Personality (given above) tests and the answers from people. The answers are numerically encoded as below.

ANSWERS	**NUMERIC ENCODE**
Fully Agree	3
Partially Agree	2
Slightly Agree	1
Neutral	0
Slightly Disagree	-1
Partially Disagree	-2
Fully Disagree	-3

The personality classification data consists of 60.000 samples (rows) with discrete 16 ("Personality" attribute) ground- truth class types.

Each sample in the personality classification data contains 62 variables/attributes/features. 61 of them are independent variables and 1 dependent variable. These independent variables except "Response Id" represent the quantitative response with respect to their counterpart questions. You have to remove/drop "Response Id" from the data. The dependent variable represents "Personality" types.

3.3. Classification Performance Evaluation Metric

In order to measure the performance success of your K-Nearest Neighbor classification model, you will compute "***Accuracy***", "***Precision***" and "***Recall***" of your implemented K-Nearest Neighbor classification model.

Accuracy: Accuracy is an performance evaluation metric for evaluating classification models. Informally, it is the fraction of predictions the model got right.

Accuracy can be calculated in terms of positives and negatives as follows:

$$\boldsymbol{Accuracy} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

where *TP* = True Positives, *TN* = True Negatives, *FP* = False Positives, and *FN* = False Negatives.

Example: Assume that we have a model that classified 100 tumors as **malignant** (the positive class) or **benign** (the negative class). Let's try calculating the accuracy of this model:

True Positive (TP): • Reality: Malignant • ML model predicted: Malignant • Number of TP results: 1	False Positive (FP): • Reality: Benign • ML model predicted: Malignant • Number of FP results: 1
False Negative (FN): • Reality: Malignant • ML model predicted: Benign • Number of FN results: 8	True Negative (TN): • Reality: Benign • ML model predicted: Benign • Number of TN results: 90

$$\boldsymbol{Accuracy} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} = \frac{1 + 90}{1 + 90 + 1 + 8} = \frac{91}{100} = 0.91$$

This model has an accuracy of 0.91 (91%). It means that this model has 91% correct predictions out of 100 total samples.

Precision: Precision is a performance evaluation metric which is measured in order to answer the following question:

"What proportion of positive identifications was actually correct?"

Precision is defined as follows:

$$\boldsymbol{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

Example: Let's try calculating the precision of classification model which analyzed/classified 100 tumors

$$\boldsymbol{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} = \frac{1}{1+1} = \frac{1}{2} = 0.5$$

This model has a precision of 0.5. In other words, when it predicts a tumor is malignant, it is correct 50% of the time.

Recall: Recall is a performance evaluation metric which is measured in order to answer the following question:

What proportion of actual positives was identified correctly?

Recall is defined as follows:

$$\boldsymbol{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

Example: Let's try calculating the recall of classification model which analyzed/classified 100 tumors

$$\boldsymbol{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} = \frac{1}{1+8} = \frac{1}{9} = 0.11$$

This model has a recall of 0.11. In other words, it correctly identifies 11% of all malignant tumors.

3.4. Feature Normalization

In order to re-scale each independent variables (60 feature/attribute columns on the data) of your samples between (0-1) range, you will **use min-max normalization** on them.

Min-max normalization (in other words **Feature Scaling)** is used to perform a linear transformation on the original data. This technique gets all the scaled data in the range (0, 1).

The formula to achieve this is the following:

$$n_i = \frac{f_i - \min(f)}{\max(f) - \min(f)}$$

3.5. Error Analysis for Classification

In this part of the assignment, you are expected to

- find a few misclassified samples and explain why you think these misclassified samples were hard to classify.
- make general performance analysis for the concepts Effect of Neighbor Number (k), Effect of Normalization, Effect of Algorithm, Effect of K-fold, Accuracy, Precision and Recall.

3.6. Steps to Follow for Classification

1. Download personality classification data

2. Read personality classification data. For this, you should use a Pandas DataFrame which is a 2-dimensional data structure like a 2-dimensional array, or a table with rows and columns.

3. Since the "Response Id" column in the DataFrame has no effect on classification, you should drop it.

4. The most important column in the DataFrame is "Personality" column which you will try to predict. This column includes string values. String values are not acceptable for Machine Learning. Therefore, you will encode them with integers given below.

PERSONALITY	**ENCODE**
ESTJ	0
ENTJ	1
ESFJ	2
ENFJ	3
ISTJ	4
ISFJ	5
INTJ	6
INFJ	7
ESTP	8
ESFP	9
ENTP	10
ENFP	11
ISTP	12
ISFP	13
INTP	14
INFP	15

5. Transform your DataFrame to the NumPy array collection.

6. Split your data into two sets which are predictor (independent variables) and target (dependent variable)

7. For the test samples,

- predict their classes using k-NN. (with feature normalization and without feature normalization)
- predict their classes using weighted k-NN. (with feature normalization and without feature normalization) **(BONUS)**

8. Compute and report your accuracy, precision and recall of your different k-NN and weighted k-NN models with different k parameters (You must experiment with this k parameters: (1,3,5,7,9)) on 5-fold cross validation. (https://machinelearningmastery.com/k-fold-cross-validation/)

9. Report your findings in" Error Analysis for Classification" section.

4. GRADING POLICY

TASK	**POINT**
Data Preparation	5
Data Scaling	10
KNN Implementation	20
Performance Evaluation Metrics Implementation	15
Execution of KNN and Performance Evaluation Metrics with standard KNN algorithm	20
Execution of KNN and Performance Evaluation Metrics with weighted KNN algorithm **(BONUS)**	10
Cross Validation	5
Coding Standard/Clean Code	10
Error Analysis for Classification	15
TOTAL (with bonus)	110

5. IMPORTANT NOTES

- The assignment must be original, individual work. Downloaded, Modified, Duplicate or very similar source codes are all going to be considered as cheating.
- We will be using a software similarity program to identify the cases of possible plagiarism. Our department takes the act of plagiarism very seriously. You cannot share algorithms or source code. All work must be individual! Those caught plagiarizing (both originators and copiers) will be sanctioned.
- You can ask your questions via Piazza and you are supposed to be aware of everything discussed on Piazza. General discussion of the problem is allowed, but DO NOT SHARE answers, algorithms, source codes, and reports.

- In the “Coding Standard/Clean Code” task of the grading policy section, it will be taken into account whether unnecessary code fragments are used whether the abstraction level is regular, and whether your code is understandable.
- In the “Coding Standard/Clean Code” task of the grading policy section, it will be taken into account whether variable names or function names are used in the correct format.
- Do not miss the submission deadline.
- Don’t forget to write comments on your codes when necessary.
- Save all your work until the assignment is graded.
- You must submit your work with the file as stated below:

assignment5.ipynb

- URL for deadline: https://classroom.github.com/a/h8LH2W17
- URL for the first late day: https://classroom.github.com/a/iGog0OrA
- URL for the second late day: https://classroom.github.com/a/jIdj_2Nr
- URL for the third late day: https://classroom.github.com/a/3uZ86hxg