Assignment 2: Classification

Innopolis University

Machine Learning Fall 2020 - Bachelors

1 General Instructions

In this assignment, you are going to solve three problems. First, you will work on improving Nearest Nighbors retrieval speed by applying specialized data structures and algorithms for fast neighbors retrieval - e.g., KD-tree, ANNOY, Navigable Small World Graph, FAISS, etc. Second, you will show your understanding of Naive Bayes by implementing its Gaussian form. And third, you will solve a problem of dimensionality reduction using PCA, and your task will be to figure out the optimal number of principal components using the so called elbow method.

Fill in the template and submit your solution via Moodle as a single **.ipynb** file (no zip archives!). Run your code before submitting, **we should see your results in the output blocks**, otherwise we will ignore your submission.

The source code should contain adequate internal documentation in the form of comments and text cells. They should explain your code as well certain choices you make. **Untidy and unreadable code will be penalized.**

Bonus points might be awarded for elegant solutions and great documentation. However, these bonus points will only be able to cancel the effect of penalties.

Plagiarism will not be tolerated, and a plagiarised solution will be heavily penalized for all parties involved. Remember that you learn nothing when you copy someone else's work, which defeats the exercise's purpose! You are allowed to collaborate on general ideas with other students as well as consult books and Internet resources. However, be sure to credit all the sources you use to make it clear what part of your solution comes from elsewhere.

2 Improving Nearest Neighbors Retrieval Speed (8 points)

Nearest Neighbors (NN) search is an important problem which has many applications: in KNN, clustering, recommendation systems, etc.

You are already familiar with KNN: during our labs you were using one of its library implementations, and most of them utilize one or another technique to speed up NN search. Now it is the time to take a look under the hood.

NN search in its simplest form is quite slow as it has to perform an exhaustive search through the dataset for every new sample. In case of an industrial-size dataset such process-

ing could be unacceptably slow. Fortunately, there exist a number of approaches to speed the processing.

Nearest neighbors methods fall into two main categories: **exact** and **approximate**.

Exact methods, such as KD-trees, Ball-tree, VP-tree, are based on *space-partitioning*. These methods work by first building an index in the form of a tree data structure which at every level splits the data according to some separating hyperplane. The speed improvement is then achieved by excluding some branches from inspection while searching for NN.

Sometimes, however, when we are processing a new sample, it may be acceptable to retrieve a "good guess" of nearest neighbours to the sample instead of true nearest neighbours. In those cases, one can use an algorithm which doesn't guarantee to return the actual nearest neighbour in every case, in return for improved speed or memory savings. Thus, with the help of such algorithms one can do a **fast approximate search** in a very large dataset. These methods include various solutions based on trees, like ANNOY, and graphs, like HNSW. Also, you can find a number of other interesting solutions available as a part of Facebook's FAISS library.

You will experiment with NN search on the Geographical dataset of around 200k places around the world (download from Moodle). Your task will be as follows:

1) Implement the simplest form of NN search (exhaustive search).

2) Choose any method you like to speed up NN search, understand it thoroughly and learn how to apply it. Explain the way it works in few sentences and in your own words.

3) To show that both methods are working, plot nearest neighbors for a given test point (details in the template). You should get two plots looking approximately like in Figure 1.

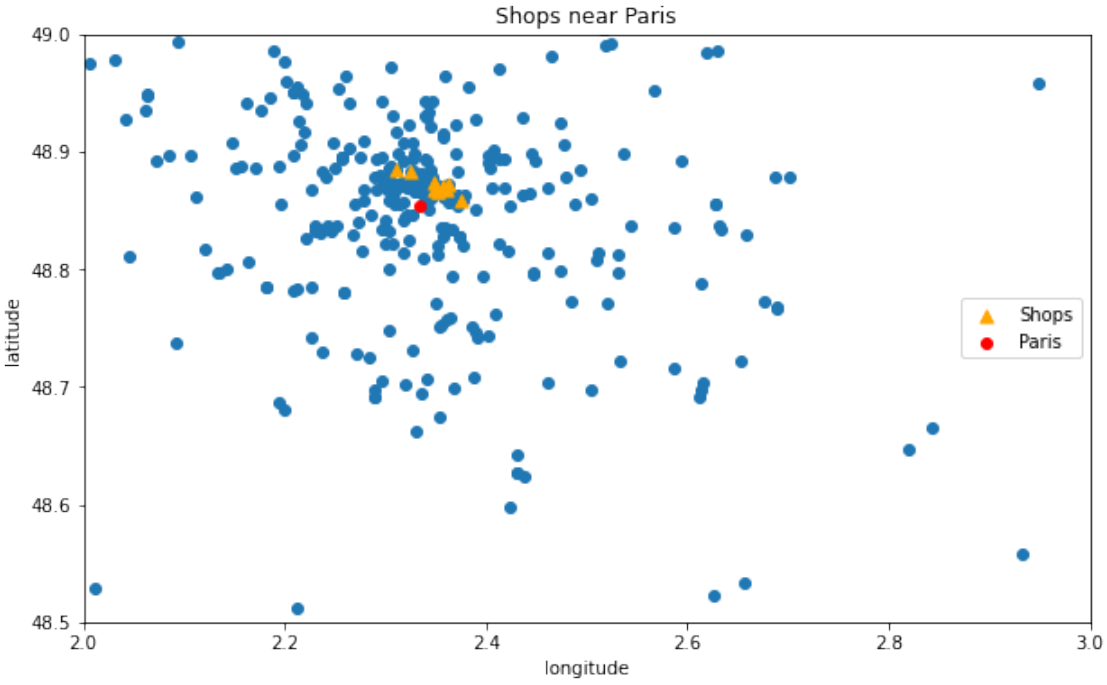

Figure 1: Sanity check.

4) For both methods, measure their retrieval times depending on the number of nearest

neighbors to be retrieved (details in the template). Plot results separately and comment on what you observe - do you see any trends, why?

5) Plot retrieval times together in the same plot. If time difference is big (as it should be), use a logarithmic scale for better visualization. You should get something similar to Figure 2. Again, analyze the results and summarise them in your notebook. If you don't achieve any time improvements, then you are doing something wrong.

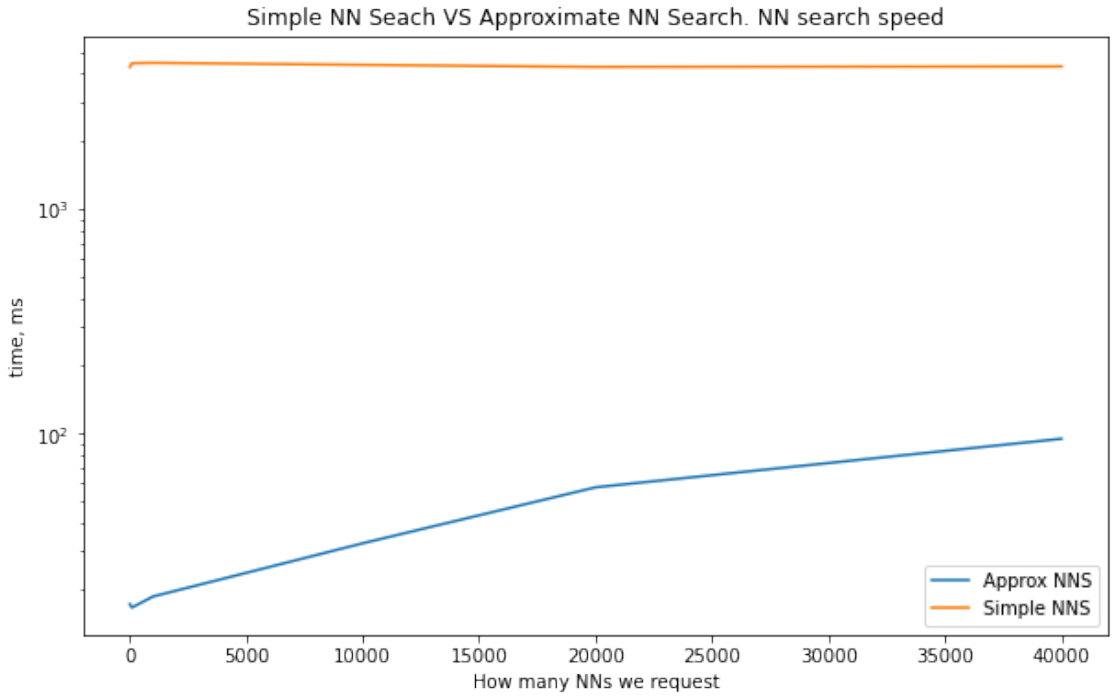

Figure 2: Measuring search speed.

3 Gaussian Naive Bayes (4 points)

Your next task will be to show your understanding of Naive Bayes algorithm by implementing its basic parts from scratch. While lecture slides gave you an example of running NB on categorical data, in this case, you will work with numeric Iris dataset.

In order to apply NB to numeric features you need to assume some distribution - in this case it will be a Gaussian (Normal) distribution. For each class you will calculate its prior probability; for each class and feature you will estimate distribution parameters: mean and standard deviation. Then, using the calculated values, for any new sample you will be able to compute the probability score of this sample belonging to this or another class. Refer to lecture slides and any free resources as needed.

You should get the accuracy score not less than 90%.

4 PCA (3 points)

You last task will be to figure out the optimal number of Principal Components (PCs) for the Digits dataset, which initially consists of 64 numeric features. You need to apply the so called *elbow method*. Its essence is as follows. You plot the special graph called "scree plot", which shows the proportion of explained variance for each principal component (monotonically decreasing), as in Figure 3. In this plot it is clear that starting from a certain point, resembling human elbow (3-4 here), adding new principal components doesn't add much in terms of explained variance. Or, in other words, diminishing returns are no longer worth the additional cost. Hence, this is the point to stop - we take only 4 first PCs, and ignore the rest of them. This is a simple intuitive heuristic which helps to identify the appropriate number of PCs when there is a trade-off between the number of features and the amount of variance we want to preserve.

Your task will be to apply PCA, produce the scree plot for the given data and decide the optimal number of PCs to keep. Calculate and print their total explained variance. Thoroughly justify your choice - this is the main part of the work for this task.

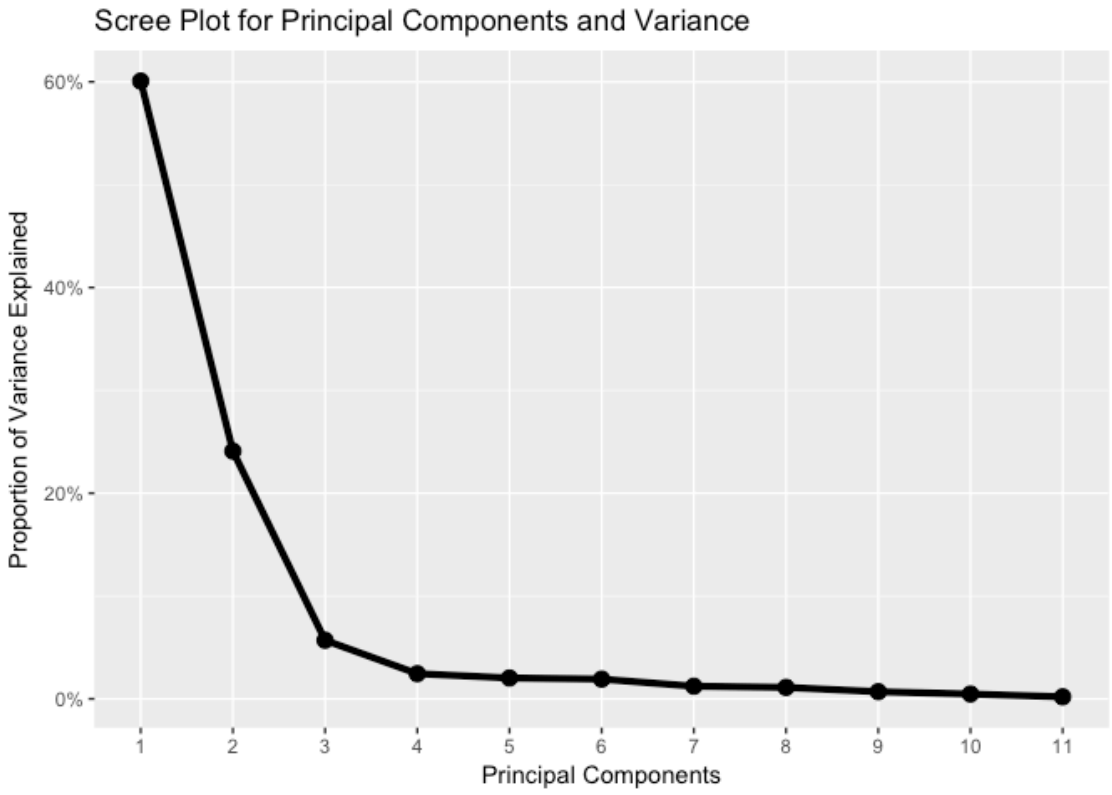

Figure 3: Scree plot.