MohmmadHossein(Will) Samadi - 1193175
msamadi1@lakeheadu.ca
Fall Semester 2022

Instructor: Dr. Thiago

Assignment #1

Q1. What is the main disadvantage when applying KNN to large datasets or datasets involving a large number of features?

In KNN, there is a need to go over all the data points to make a decision. So as the dataset grows larger and larger, the volume of calculations needed for making a simple decision becomes bigger which is not ideal in super large datasets.

Q2. Condensing is a technique to reduce the complexity of the KNN algorithm by reducing which term of the O(nd) complexity?

Condensing has nothing to do with dimensionality (d) and everything to do with ignoring the data points that are less important (n).

Q3. Implement the exact solution KNN for dataset 1.

Results of the running of the code:

```
PS F:\Dev\KNN> python .\main.py
reding labels
reading ds
shuffling
printing the output
[[    394244    9289728    2093056 ...    2096896    1047552     253952]
 [     98304    2088960    4192256 ...   37715968    4063232    1048576]
 [   1048834    3932160    3932160 ...    4193280    2095104    1032192]
 ...
 [   4128768   16744448   33538048 ...   33554176    2096128    1044480]
 [   4186112    8384512   16773120 ...   16777152    4194240    2097024]
 [     16128      65472     262080 ...     258048 1610739712      61440]]
starting k-point
k = 1
accuracy for 1 is 0.01709844559585492
k = 2
accuracy for 2 is 0.018134715025906738
k = 3
accuracy for 3 is 0.01606217616580311
k = 4
accuracy for 4 is 0.22124352331606217
k = 5
accuracy for 5 is 0.018134715025906738
k = 6
accuracy for 6 is 0.021243523316062177
k = 7
accuracy for 7 is 0.023316062176165803
k = 8
accuracy for 8 is 0.01865284974093264
k = 9
accuracy for 9 is 0.021243523316062177
k = 10
accuracy for 10 is 0.02383419689119171
best k for knn is:
3
```

The best k is 4. (it says 3 because it's the index of the Ks which starts from 0). It is the best result by far (%20 accuracy which is really low but twice as good as assigning a class randomly).

Q4. Given the following loss matrix...

Radiotherapy: $R(y_1|x) = \lambda_{11}P(y_1|x) + \lambda_{12}P(y_2|x) = 20 \times 0.4 = 8$

Medication: $R(y_2|x) = \lambda_{21}P(y_1|x) + \lambda_{22}P(y_2|x) = 10 \times 0.6 = 6$

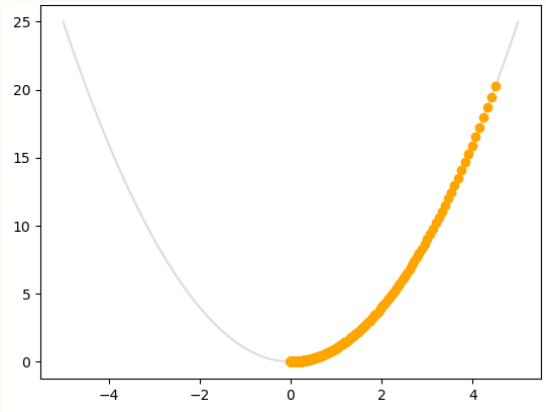
Q5. Implement a Naive Bayes classifier...

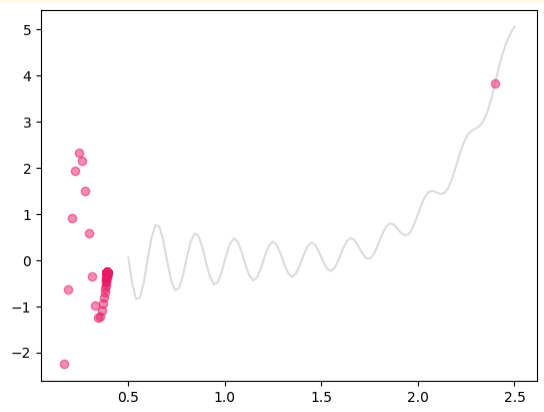
Result of running the code:

```
PS F:\Dev\KNN> python .\naivebayes\main.py
tokenizing...
tokens ready!
counting...
counts ready
testing...
tokenizing set...
counting test words...
done
0.20254777070063695
PS F:\Dev\KNN>
```

As there are 2 classes and the classifier is yielding %20 accuracy, that means I coded the accuracy part wrong (haha). Because if it's 20 percent accurate and we do the opposite of what the classifier says we'll achieve **80 percent accuracy** as there are only two classes.

Q6. GD





Q7. Why applying linear regression to a classification problem may not be adequate? What is the effect of outliers in this type of classifier?

It's a good way to summarize the data into a simple function but generally speaking, it's not ideal for classification as it doesn't really react to the differences of the data points and has a different purpose.

Outliers can drag the line to a non-ideal position if their effect is not normalized in some way.

Q8. Implement a classifier based on logistic regression...

```
reding labels
reading ds
data read, sample:
[[    394244    9289728    2093056 ...    2096896    1047552     253952]
 [     98304    2088960    4192256 ...   37715968    4063232    1048576]
 [   1048834    3932160    3932160 ...    4193280    2095104    1032192]
 ...
 [    983040    2064384    4177920 ...    4194048     524032     261632]
 [2149515264    2064384  272596992 ...    4194288    1048560     524224]
 [    786432     786504    1966080 ...    2096128    1047552     507904]]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
fitting model
making a model
F:\Dev\KNN\logistic_regression\classifier.py:36: RuntimeWarning: overflow encountered in power
  p = np.power(np.e, bTx)
F:\Dev\KNN\logistic_regression\classifier.py:37: RuntimeWarning: invalid value encountered in double_scalars
  return p / (1 + p)
F:\Dev\KNN\logistic_regression\classifier.py:41: RuntimeWarning: overflow encountered in power
  p = np.power(np.e, bTx)
fitted, testing...
accuracy:
0.5
```

I ran the optimizer for 1000 rounds and near the end (950s) it overflows. The accuracy, in the end, is 0.5 which is terrible (kinda equal to tossing a coin).