AI Lab 2

Karl-Johan Djervbrant

January 3, 2020

Task 1

1.1 a)

Classifiers $k = 5$	Algorithm	Accuracy (%)	Correct	Time (s)
Own KNN_Classifier_fast	euclidean	0.8525	341/400	4.76143
Own KNN_Classifier	euclidean	0.8525	341/400	54.7676
Sklearn	euclidean	0.8525	341/400	0.0145493

Table 1: Performance of own KNN algorithm vs sklearn implementation.

In this task we had to implement our own version of a KNN_Classifier. The Algorithm works by calculating the distance from one data point to another, and assigning the class to the new data point which the K-Nerest-Neighbors has.

The difference between `KNN_Classifier` and `KNN_Classifier_fast` is that the faster version only keep track of the K closest distances currently calculated, this means that you don't have to sort the whole list of calculated distances, (which increases every time you calculate a distance). With the faster algorithm, you only need to sort a list with K elements in it. This also saves on memory allocation.

As we can see in Table 1.1, both version of the own implementation perform equally good compared to the KNN-algorithm implemented by scikit-learn. The only difference is the execution time, where the scikit-learn implementation is much faster.

1.2 b)

Classifiers	Score (R^2)	Time (s)
SVC, $kernel = linear,\ C = 0.025$	0.9475	0.230703
Decision Tree, $max\_depth = 5$	0.8800	0.005497
Random Forest, $max\_depth = 5, n\_esti = 10, max\_feat = 1$	0.9100	0.013979

Table 2: Performance between three classification algorithms, higher score is better.

In this task we got to implement and compare three different algorithms from the scikit-learn library. I choose to test the performance on Scalar-Vector-Classifier (SVC), Decision Tree Classifier and Random Forest Classifier. From the data in Table 1.2 we can see that they perform good on different measures, the SVC algorithm does the best prediction but is also the slowest. The Decision Tree algorithm is the worst one, but also the fastest.

1.3 c)

To change the distance algorithm used, you initialize the constructor with the parameter p set to a specific value. $p = 1$ is for Manhattan-distance and $p = 2$ is for Euclidean-distance. If you set p to another value you can calculate other types of distances, the algorithm used is the Minkowski-distance algorithm [**?**]. In Table 1.3 we can se the difference in performance (using cross-validation) depending on distance algorithm and amount of neighbors. On this data set

Classifier	Neighbors	Distance Measure	Score (Mean/Std)
KNN	3	Manhattan	0.73 (+/-0.45)
KNN	5	Manhattan	0.73 (+/-0.45)
KNN	7	Manhattan	0.74 (+/-0.45)
KNN	9	Manhattan	0.71 (+/-0.47)
KNN	11	Manhattan	0.73 (+/-0.48)
KNN	3	Euclidean	0.70 (+/-0.45)
KNN	5	Euclidean	0.71 (+/-0.48)
KNN	7	Euclidean	0.70 (+/-0.50)
KNN	9	Euclidean	0.68 (+/-0.48)
KNN	11	Euclidean	0.67 (+/-0.48)
KNN	3	Minkowski p=$2^{1.5}$	0.68 (+/-0.47)
KNN	5	Minkowski p=$2^{1.5}$	0.70 (+/-0.49)
KNN	7	Minkowski p=$2^{1.5}$	0.70 (+/-0.47)
KNN	9	Minkowski p=$2^{1.5}$	0.67 (+/-0.48)
KNN	11	Minkowski p=$2^{1.5}$	0.67 (+/-0.47)

Table 3: Cross-Validation score of KNN-Classifier with different amount of neighbors and distance algorithms

the Manhattan-distance perform slightly better than the Euclidean-distance, the best one is with $K = 7$.

1.4 e)

Regressors $k = 5$	Algorithm	Score (R^2)	Time (s)
Own KNN_Regressor_fast	euclidean	0.395646	9.06785
Own KNN_Regressor	euclidean	0.395646	78.4536
Sklearn	euclidean	0.395646	0.00223327

Table 4: Performance of own KNN-Regression algorithm vs sklearn on estimating Fuel Consumption.

A regression algorithm is used to estimate a value for a specific feature, in this case we use it to estimate the fuel consumption. The only modification done to the algorithm compared to the classification version is that it now returns the mean of the K nearest classes. As we can see in the Table 1.4, KNN-Regression isn't working too well on estimating the fuel consumption on this data set. This is partly due to that KNN is sort of a improved look-up table and is relatively simple, which is why it doesn't perform great in all occasions.

1.5 f)

Regressors	Score (R^2)	Time (s)
SVR, $kernel = linear,\ C = 0.025$	0.768634	3.850483
Decision Tree Regressor, $max\_depth = 5$	0.754809	0.007603
Random Forest Regressor, $max\_depth = 5, n\_esti = 10, max\_feat = 1$	0.608807	0.023281

Table 5: Performance of three different regression algorithms from the sklearn library.

With the use of other regression algorithms we can get much better prediction results compared to KNN-Regression, as we can see in Table 2.1 SVR and Decision Tree Regressor perform almost twice as good as KNN-Regression. Out the tree algorithms tested, the SVR (Scalar Vector Regressor) performed best, bust was also much slower compared to Decision Tree Regressor. The DTR algorithm did almost score as high as the SVR algorithm but were much faster to calculate.

1.6 g)

Classifier	Neighbors	Distance Measure	Score (Mean/Std)
KNN	3	Manhattan	-0.31 (+/-2.87)
KNN	5	Manhattan	-0.27 (+/-2.94)
KNN	7	Manhattan	-0.17 (+/-2.61)
KNN	9	Manhattan	-0.15 (+/-2.46)
KNN	11	Manhattan	-0.11 (+/-2.38)
KNN	3	Euclidean	-0.46 (+/-3.36)
KNN	5	Euclidean	-0.35 (+/-3.02)
KNN	7	Euclidean	-0.25 (+/-2.65)
KNN	9	Euclidean	-0.23 (+/-2.61)
KNN	11	Euclidean	-0.22 (+/-2.55)
KNN	3	Minkowski p=$2^{1.5}$	-0.50 (+/-3.33)
KNN	5	Minkowski p=$2^{1.5}$	-0.35 (+/-3.07)
KNN	7	Minkowski p=$2^{1.5}$	-0.26 (+/-2.77)
KNN	9	Minkowski p=$2^{1.5}$	-0.25 (+/-2.70)
KNN	11	Minkowski p=$2^{1.5}$	-0.26 (+/-2.74)

Table 6: Cross-Validation score of KNN-Regression with different amount of neighbors and distance algorithms

As previously discussed in Task 1.3 1c), the distance algorithm used is changed by changing the value of p. In Table 1.6 we see that KNN-Regression perform quite bad on estimating the Fuel Consumption, with a low score and high standard deviation, but the best performing one out of the tested values were with Manhattan Distance as distance measure and $K = 11$.

Task 2

2.1 a)

To use the data in `Lab4PokerData.csv` we first need to translate the data to something a classification algorithm can understand, the text in the data were translated as followed:

Values	Ranks	Actions	Empty Cell
2: 2	0: highcard	0: Fold	-1: NO_DATA
3: 3	1: onepair	1: Call	
4: 4	2: twopair	2: Raise	
5: 5	3: 3ofakind	3: Allin	
6: 6	4: straight		
7: 7	5: flush		
8: 8	6: fullhouse		
9: 9	7: 4ofakind		
T: 10	8: straightflush		
J: 11			
Q: 12			
K: 13			
A: 14			

Table 7: Translation table for data in Lab4PokerData.csv

To make it easier to understand the data, every column were assigned with a name. Following table shows an example of the labels used.

p1_hand_category	p1_hand_rank	p1_hand_value	p1_action_n
p1_raise_n	p1_money_left	p1_spent_per_hand_value	p1_final_action
p2_hand_category	p2_hand_rank	p2_hand_value	p2_action_n
p2_raise_n	p2_money_left	p2_spent_per_hand_value	p2_final_action

Table 8: Labels of each column in Lab4PokerData.csv, n is for each round.

The added attributes are *pX_money_left, pX_spent_per_hand_value & pX_final_action*. "*pX_money_left*" is calculated by summing how much the player bet each round and subtract it from 400. "*pX_spent_per_hand_value*" is derived from dividing how much a player spent on that round with the given hand value.

2.2 b)

Classifier	Score (R^2)	Time (s)
KNN, $k = 5$	0.75	0.000464
SVC, $kernel = linear,\ C = 0.025$	0.95	0.017687
Decision Tree, $max\_depth = 5$	1.00	0.000918

Table 9: Performance of three different classification algorithms on the Poker Data set.

Here we get some really interesting results, the Decision Tree Classifier manages to score 1.00 which means it got every prediction correct. The SVC algorithm did also perform really good with a score of 0.95. The KNN-Classifier did not perform as good as the other two algorithms with a score of only 0.75, but it was the fastest among the three.

2.3 c)

Classifier	Neighbors	Distance Measure	Score (Mean/Std)
KNN	3	Manhattan	0.75 (+/-0.22)
KNN	5	Manhattan	0.80 (+/-0.25)
KNN	7	Manhattan	0.70 (+/-0.34)
KNN	9	Manhattan	0.60 (+/-0.29)
KNN	11	Manhattan	0.45 (+/-0.12)
KNN	3	Euclidean	0.75 (+/-0.22)
KNN	5	Euclidean	0.70 (+/-0.30)
KNN	7	Euclidean	0.57 (+/-0.25)
KNN	9	Euclidean	0.40 (+/-0.19)
KNN	11	Euclidean	0.42 (+/-0.12)
KNN	3	Minkowski p=$2^{1.5}$	0.70 (+/-0.20)
KNN	5	Minkowski p=$2^{1.5}$	0.65 (+/-0.33)
KNN	7	Minkowski p=$2^{1.5}$	0.60 (+/-0.19)
KNN	9	Minkowski p=$2^{1.5}$	0.42 (+/-0.25)
KNN	11	Minkowski p=$2^{1.5}$	0.42 (+/-0.12)

Table 10: Cross-Validation score of KNN-Classifier with different amount of neighbors and distance algorithms

When trying different parameters for the KNN-Classifier we can tell by the results in Table 2.3 that the Manhattan Distance as distance measure and with $k = 5$ perform best. We can also tell that the score drops rapidly with $k > 5$ for all the different distance measures tested.