Machine Learning, Fall 2019: Project 1
K-Nearest-Neighbour - due on 9/19/19 by noon on Blackboard

Anastasija Mensikova

Header: Resources used for the completion of the project:

- Python programming language
- Jupyter Notebook
- Distance measures: https://medium.com/@srishtisawla/k-nearest-neighbors-f77f6ee6b7f5
- Distance measures: https://pdfs.semanticscholar.org/a630/316f9c98839098747007753a9bb6d05f752e.pdf
- Lecture slides
- draw.io for drawing a decision tree

Using the above resources and general knowledge this project explored the K-Nearest-Neighbour algorithm on two different datasets – Pima Indians Diabetes and MNIST. The experiments, explorations, and the results are recorded both in this paper and the attached iPython notebook. All the results can be re-created by running the notebook.

The project is split up into two main sections – the programming task that implements and tests KNN on the two datasets as described above, and the "written" task that explores decision trees.

Programming Task: The implementation, as well as the step-by-step description and guide on the implementation of the programming task of the given project can be found in the iPython notebook. However, in addition to that, the results and observations made in the process of that implementation can also be found below.

Dataset details: *The Pima Dataset* contains data on the members of the Pima Indian Reservation in Arizona, specifically on their diabetes tendencies. The original dataset contains information on 768 Pima Indians and has 9 features (one of which is the 'label' that points to the presence or absence of diabetes), namely pregnancies, glucose, blood pressure, BMI, age, skin thickness, insulin, the diabetes pedigree function, and finally the outcome. A big part of implementing the KNN algorithm is pre-processing the data. The data pre-processing pipeline will be explained in the further section. However, it is integral to look at the entire dataset and visualise it in order to understand its underlying structure and any need for re-organising the data.

In order to understand the general distribution of data in *the Pima Dataset* the visualisations presented in Figures 1 through 12 were done.

Although there was practically no feature engineering needed for the MNIST dataset, it is still important to examine the visual representation of the data. The Figures 13 through 15 are just a few examples of the data points used.

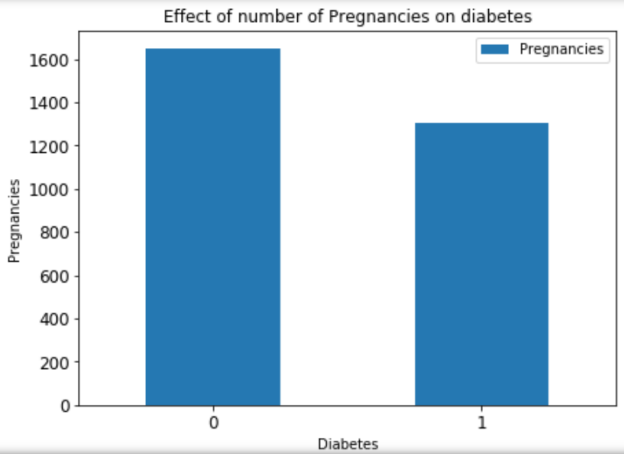

Figure 1: Pregnancies

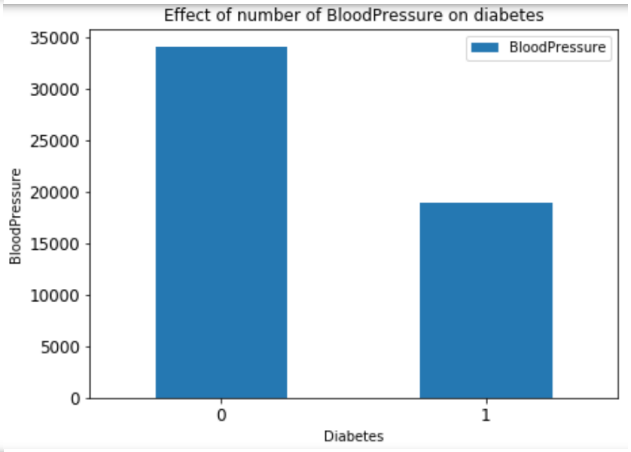

Figure 2: Blood Pressure

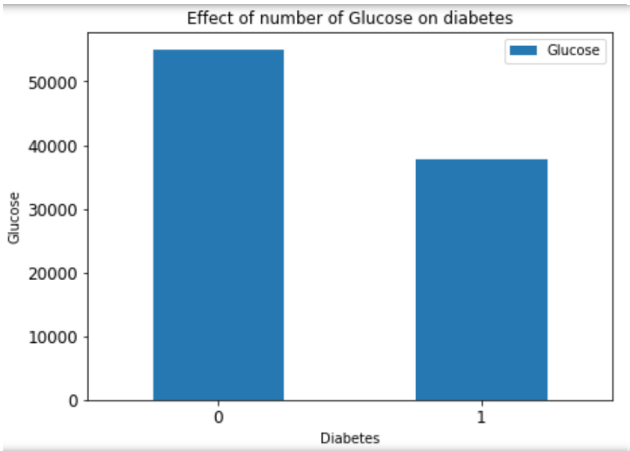

Figure 3: Glucose

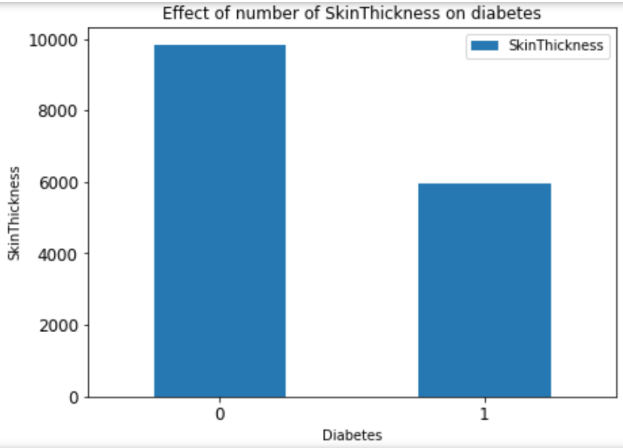

Figure 4: Skin Thickness

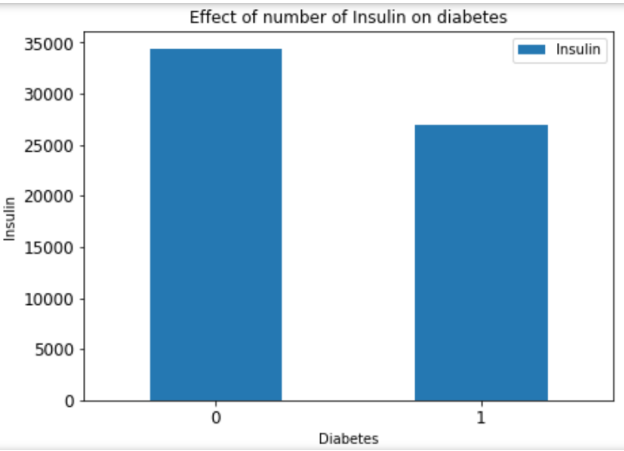

Figure 5: Insulin

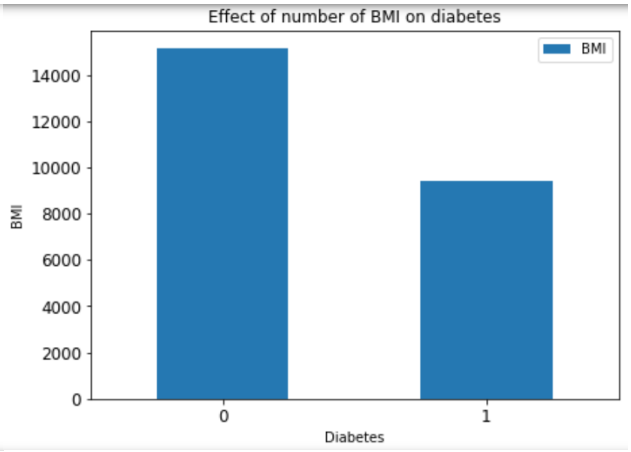

Figure 6: BMI

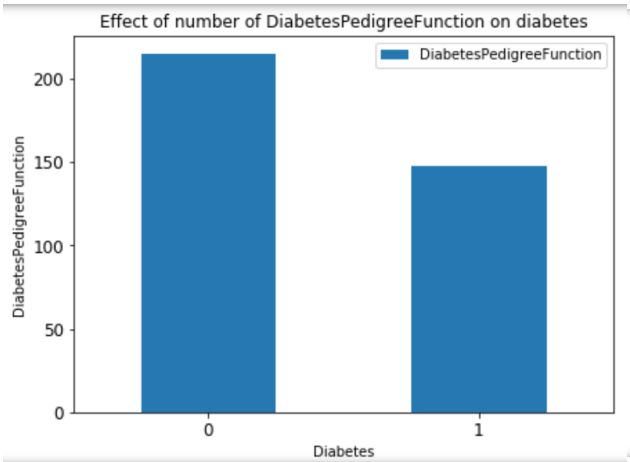

Figure 7: DPF

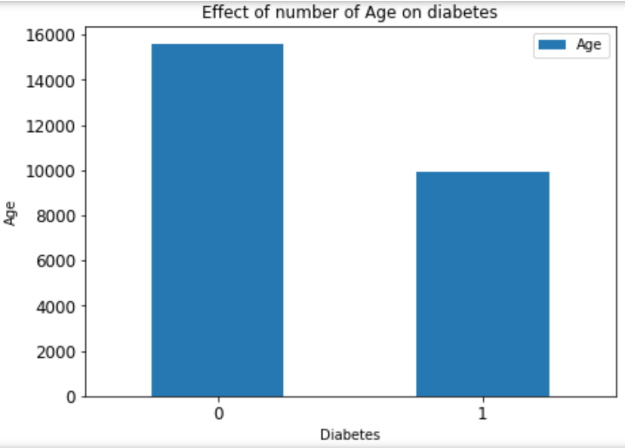

Figure 8: Age

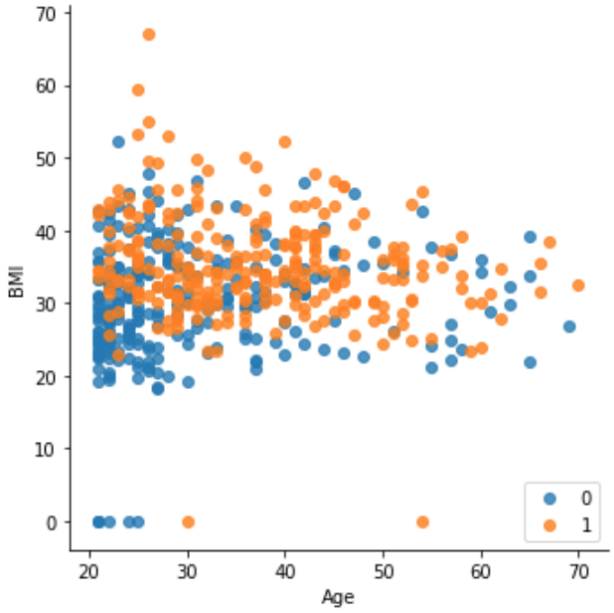

Figure 9: BMI + Age

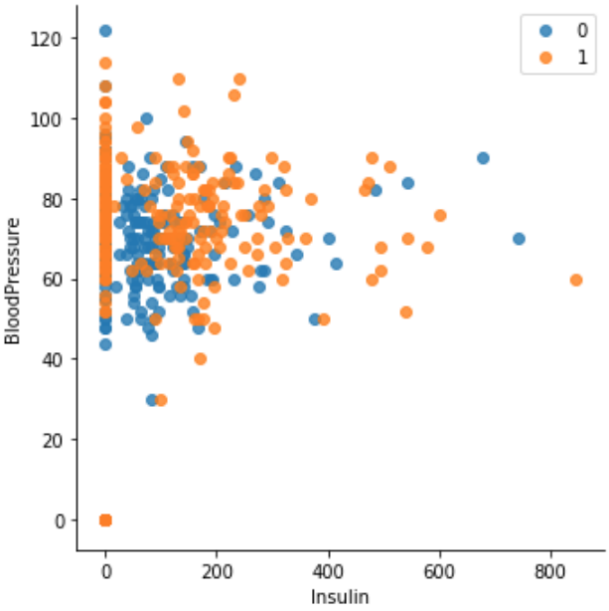

Figure 10: Blood Pressure + Insulin

For both of the datasets 80% of the data was used for training purposes and 20% for testing. After performing numerous experiments it was found that the 80:20 distribution seems to be the most optimal one for both datasets as it leaves enough context for the model to learn while having enough data points for efficient testing.

Algorithm Description: The general idea of KNN in this implementation remained the same as it is explained in any scholarly resource – given the number k and a distance measure, k closest neighbours to the point examined are found, and by the "majority rules" a class is given to this point. This implementation made use of Numpy arrays to speed up the process.

Out of the two datasets, *the Pima Dataset* is certainly more complex due to the variety of features it encompasses. The first step that was undertaken before performing any sort of feature engineering was ensuring the balanced distribution of data.

As mentioned before, *the Pima Dataset* is binary, that is, it contains data with either one of the two

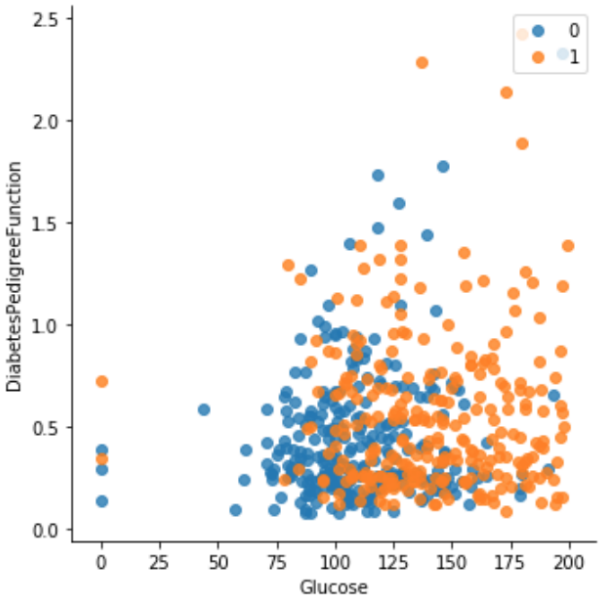

Figure 11: DPF + Glucose

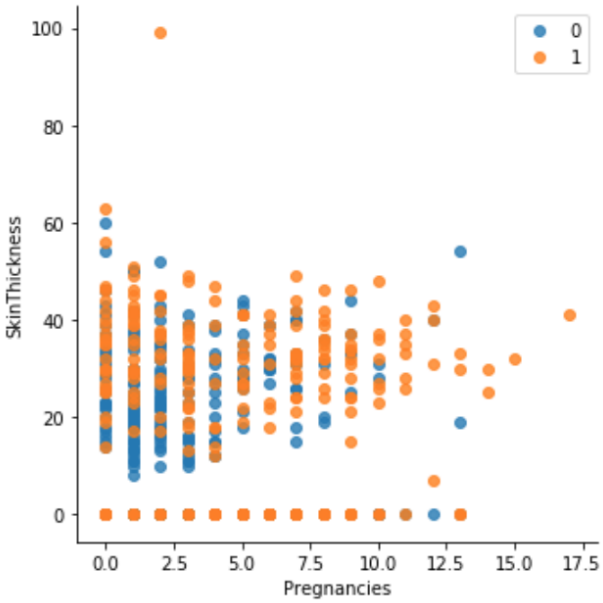

Figure 12: SKin Thickness + Pregnancies

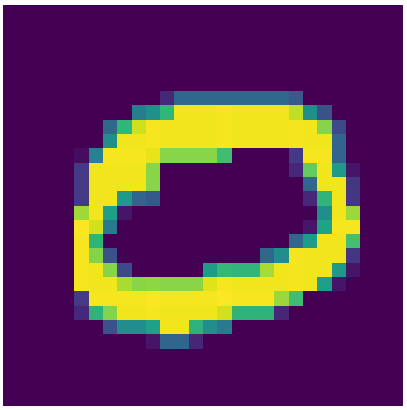

Figure 13: Zero

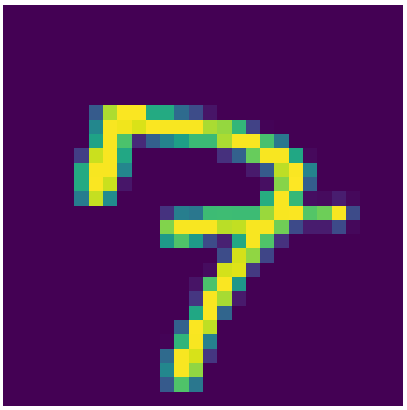

Figure 14: Seven

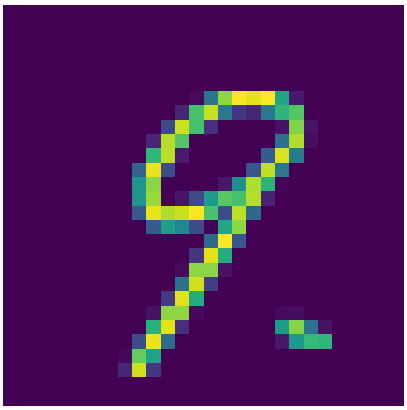

Figure 15: Nine

labels (0 or 1). It was therefore important to ensure that the amount of samples per label was even. Upon examination, it turned out that there are 500 negative samples (with label 0, no diabetes) and 268 positive samples (with label 1, diabetes present). The excess 232 negative samples were therefore eliminated.

Figures 1 through 12 above were all used to examine the makeup of the data and decide on what steps could be undertaken to make improvements to the algorithm. The following observations were made about Figures 1 through 12:

- Generally, we can see that the larger amount of pregnancies per lifetime correlates to the presence of diabetes, but it's not a very obvious correlation so we will leave it out.
- Glucose levels also correlate to a higher likelihood of having diabetes (which is pretty intuitive).
- There isn't much correlation between blood pressure and diabetes.
- Although thicker skin is more likely to correlate to the presence of diabetes, that correlation is not very strong.

- Higher levels of insulin seem to have a strong correlation to diabetes.
- Higher BMI is more prevalent in those with diabetes.
- Higher diabetes pedigree function seems to correlate with higher likelihood of diabetes, but it's not very obvious.
- Lower age correlates to lower likelihood of diabetes, but it is not very strong.

After such observations the following features were selected for the training and testing process as they had a tendency to influence diabetes more clearly:

- BMI
- Glucose
- Insulin

The next step undertaken in pre-processing this dataset was re-scaling. Since all three selected features are measured on completely different scales, it is important to re-scale them to ensure a better distance measure between points. Upon examination, Min-Max re-scaling appeared to be one of the best options for this purpose, so all three features were subsequently re-scaled.

One of the benefits of the KNN algorithm is its flexibility in terms of the distance metrics allowed. Although Euclidean Distance is certainly one of the most commonly used distance metrics in KNN, Manhattan Distance and Chebyshev Distance are another two that were explored for the purpose of this project, specifically for the *Pima Dataset*. All three distances are quite similar, and their exact implementation can be seen in the iPython notebook attached. However, some differences in accuracy were observed for all three of them for this dataset.

As mentioned earlier, the *MNIST dataset* did not require any pre-processing as all of its data points represent pixels in an image, which automatically ensures that all of them are within one scale and all bear more-or-less same importance. It is, however, important to note that Euclidean Distance was the main distance measure for this dataset. Some experiments were performed with Manhattan Distance, and it was clear that it dramatically dropped the accuracy of the algorithm, so the rest of the analysis was performed using only the Euclidean Distance.

Algorithm Results: As mentioned above, for the sake of testing KNN on the *Pima Dataset* three distance measures were used. However, as per the task, k remained to be 3. The following will show the results of using different distance measures.

Table 1: Pima Performance Measures

Performance measure for k=3				
Distance Measure	Accuracy	Precision	Recall	Wall Time
Euclidean	70.09%	0.65	0.80	90.9 ms
Manhattan	70.09%	0.65	0.80	20.5 ms
Chebyshev	73.83%	0.68	0.82	19.0 ms

The following tables show the confusion matrices for the *Pima dataset* given three distance measures.

Table 2: Pima Confusion Matrices

Eucl	0	1
0	35	22
1	10	40

Manh	0	1
0	35	22
1	10	40

Cheb	0	1
0	38	19
1	9	41

From the above tables it is obvious that both Euclidean and Manhattan distances seem to produce the same results. However, Euclidean distance does take significantly more amount of time. Chebyshev distance,

on the other hand, produces slightly better results in the least amount of time. Although the difference is not dramatic, the Chebyshev distance measures seem to be more fitting for this type of binary data.

With regards to the *MNIST dataset*, on the other hand, the main parameter that was varying throughout the tests was the number of k. Below are the performance measures of running KNN on *MNIST* and confusion matrices for various k values.

Table 3: MNIST Performance Measures

k	Accuracy	Wall Time
1	96.74%	1h 3min 31s
3	96.95%	1h 1m 12s
5	96.90%	47m 20s
15	96.04%	50m 10s
91	93.44%	55m 11s

From the Table 3 it is clear that k number 3 is the most optimal number of neighbours for this dataset most likely due to its medium complexity. However, it is surprising that increasing number k decreases the wall time of the algorithm on this dataset.

Table 4: MNIST Confusion Matrix for k = 1

k=1	0	1	2	3	4	5	6	7	8	9
0	846	0	0	0	0	1	5	0	0	0
1	0	943	1	1	1	0	1	3	0	0
2	8	2	768	3	1	2	1	13	2	1
3	1	4	6	846	0	10	0	3	9	3
4	0	6	0	0	779	0	4	1	0	21
5	1	2	0	11	1	720	7	0	1	4
6	4	3	0	0	1	4	819	0	0	0
7	0	8	3	0	2	0	0	864	0	7
8	1	16	2	10	3	7	3	2	747	11
9	6	2	1	4	14	2	0	15	2	794

Table 5: MNIST Confusion Matrix for k = 3

k=1	0	1	2	3	4	5	6	7	8	9
0	848	1	0	0	0	1	2	0	0	0
1	0	947	1	1	1	0	0	0	0	0
2	8	7	769	3	0	2	0	9	2	1
3	0	3	6	857	0	6	0	3	4	3
4	0	8	0	0	781	0	5	0	0	17
5	3	0	0	16	0	715	7	0	2	4
6	5	2	0	0	0	1	823	0	0	0
7	0	9	3	0	1	0	0	864	0	7
8	3	19	5	12	5	10	1	2	738	7
9	6	2	2	8	7	2	0	11	0	802

Table 6: MNIST Confusion Matrix for k = 5

k=1	0	1	2	3	4	5	6	7	8	9
0	849	0	0	0	0	1	2	0	0	0
1	0	949	1	0	0	0	0	0	0	0
2	9	7	767	2	0	3	0	12	0	1
3	2	3	8	854	0	6	0	2	3	4
4	0	8	0	0	784	0	3	0	1	15
5	2	1	0	14	0	715	9	0	2	4
6	3	2	0	0	1	1	824	0	0	0
7	0	10	3	0	1	0	0	861	0	9
8	3	20	2	8	4	12	1	1	745	6
9	6	4	2	5	8	2	0	19	2	792

Table 7: MNIST Confusion Matrix for k = 15

k=1	0	1	2	3	4	5	6	7	8	9
0	843	1	0	0	0	2	6	0	0	0
1	0	947	1	0	1	0	1	0	0	0
2	15	17	742	2	0	4	2	15	1	3
3	2	9	3	847	0	9	0	4	4	4
4	0	10	0	0	772	0	3	0	1	25
5	3	6	0	10	3	709	12	0	1	3
6	4	2	0	0	1	0	824	0	0	0
7	0	13	3	0	0	0	0	858	0	10
8	3	20	0	11	6	9	7	1	736	9
9	7	6	2	9	6	1	0	20	0	789

If we are to choose number 7 as the number to be examined for comparing different k values, Table 8 contains the accuracy depicting such.

Table 8: MNIST Number 7 Performance

k	Accuracy
1	97.74%
3	97.74%
5	97.40%
15	97.06%
91	95.14%

Once again, similar to the total accuracy table, smaller size of k seems to guarantee higher accuracy, with k = 1 and k = 3 having the highest accuracy. Overall, given the two observations, k = 3 seems to be the most optimal for the given problem.

Runtime: The wall clock time per each test run was recorded in the tables above. Clearly, the *Pima dataset* took very little time due to the small size of the dataset and the small number of features. Euclidean distance seems to be taking the longest, however higher size of neighbours for the *MNIST dataset* seems to slightly decrease the runtime.

Upon close examination, in the worst case KNN takes

$$O(F * n^2 * log(n))$$

time, where F is the number of features and n is the number of data points. This can be explained by the need to go through every element of the training set per each element being examined, taking into consideration every selected feature. A portion of KNN involves sorting all the neighbours by distance, which takes n*log(n) time, hence why log(n) is present in the runtime.

$$O(F*(n*(n+n*log(n)))) = O(F*(n^2+n^2*log(n))) = O(F*n^2*log(n))$$

Decision Trees: The following contains the tasks, as well as the solutions to the "written" portion of the project.
Consider the following set of training examples for the unknown target function $< X_1, X_2 > \rightarrow Y$.

Y	X_1	X_2	Count
+	T	T	3
+	T	F	4
+	F	T	4
+	F	F	1
-	T	T	0
-	T	F	1
-	F	T	3
-	F	F	5

1. **(10 points)** What is the sample entropy $H(Y)$ for this training data (with logarithms base 2)?

$$H(Y) = -\sum_j P(j|Y) log P(j|Y)$$

$$P(Y=+) = \frac{12}{21}$$

$$P(Y=-) = \frac{9}{21}$$

$$H(Y) = -\frac{12}{21} log_2 \frac{12}{21} - \frac{9}{21} log_2 \frac{9}{21} = 0.985 \, to \, 3 \, d.p.$$

2. **(10 points)** What are the information gains $IG(X_1) \equiv H(Y) - H(Y|X_1)$ and $IG(X_2) \equiv H(Y) - H(Y|X_2)$ for this sample of training data?

$$IG(X_1) \equiv H(Y) - H(Y|X_1)$$

$$P(X_1 = T) = \frac{8}{21}$$

$$P(X_1 = F) = \frac{13}{21}$$

$$P(Y=+|X_1=T) = \frac{7}{8}$$

$$P(Y=+|X_1=F) = \frac{5}{13}$$

$$P(Y=-|X_1=T) = \frac{1}{8}$$

$$P(Y=-|X_1=F) = \frac{8}{13}$$

$$H(Y|X_1) = -\sum_j P(j|X_1) \sum_i P(Y=i|X_1=j) log_2 P(Y=i|X_1=j) = 0.802 \, to \, 3 \, d.p.$$

$$P(X_2 = T) = \frac{10}{21}$$

$$P(X_2 = F) = \frac{11}{21}$$

$$P(Y = +|X_2 = T) = \frac{1}{10}$$

$$P(Y = +|X_2 = F) = \frac{5}{11}$$

$$P(Y = -|X_2 = T) = \frac{3}{10}$$

$$P(Y = -|X_2 = F) = \frac{6}{11}$$

$$H(Y|X_2) = -\sum_j P(j|X_2) \sum_i P(Y = i|X_2 = j) log_2 P(Y = i|X_2 = j) = 0.940 \, to \, 3 \, d.p.$$

$$IG(X_1) = 0.985 - 0.802 = 0.183 \, to \, 3 \, d.p.$$

$$IG(X_2) = 0.985 - 0.940 = 0.045 \, to \, 3 \, d.p.$$

$$\boldsymbol{IG(X_2) < IG(X_1)}$$

3. **(5 points)** Draw the decision tree that would be learned by ID3 (without postpruning) from this sample of training data.

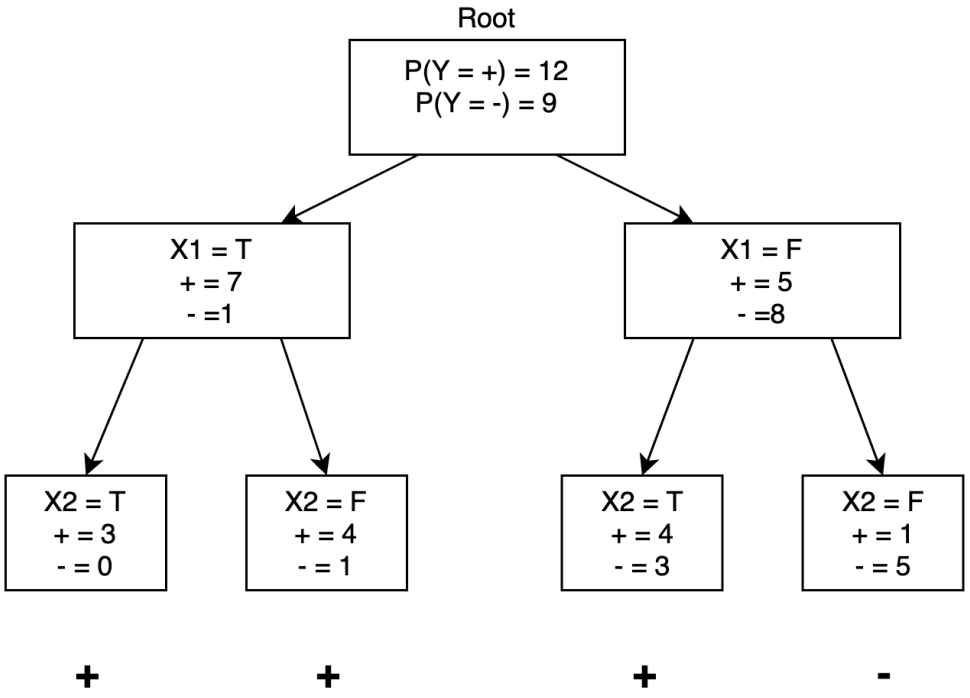

Figure 16: Decision Tree for Q.3