Exercise Set 2

Problem 8

Task a

```
import math
import pandas as pd
from sklearn import linear_model
from sklearn.metrics import accuracy_score

spam_train = pd.read_csv("spam_train.csv")
spam_test = pd.read_csv("spam_test.csv")

clf = linear_model.LogisticRegression(penalty="l1", C=1/0.001, solver="saga", max_iter=2000)

X_train = spam_train.iloc[:,:5]
y_train = spam_train.iloc[:,5]
X_test = spam_test.iloc[:,:5]
y_test = spam_test.iloc[:,5]

clf.fit(X_train, y_train)
```

```
## LogisticRegression(C=1000.0, max_iter=2000, penalty='l1', solver='saga')
```

```
print("Intercept", clf.intercept_[0])
```

```
## Intercept -10.636602945070921
```

```
print("Coefficients:")
```

```
## Coefficients:
```

```
for c, v in zip(spam_test.columns[:5], clf.coef_[0]):
  print(c, v)
```

```
## MISSING_FROM -1.9730525766827285
## FROM_ADDR_WS -7.391615368407592
## TVD_SPACE_RATIO -1.287147718356705
## LOTS_OF_MONEY 10.07163675595645
## T_FILL_THIS_FORM_SHORT 13.245340419253964
```

```
pred_train = clf.predict(X_train)
train_acc = accuracy_score(y_train, pred_train)
pred_test = clf.predict(X_test)
test_acc = accuracy_score(y_test, pred_test)

print("Accuracy train:", train_acc)
```

```
## Accuracy train: 0.88
```

```
print("Accuracy test:", test_acc)
```

```
## Accuracy test: 0.88
```

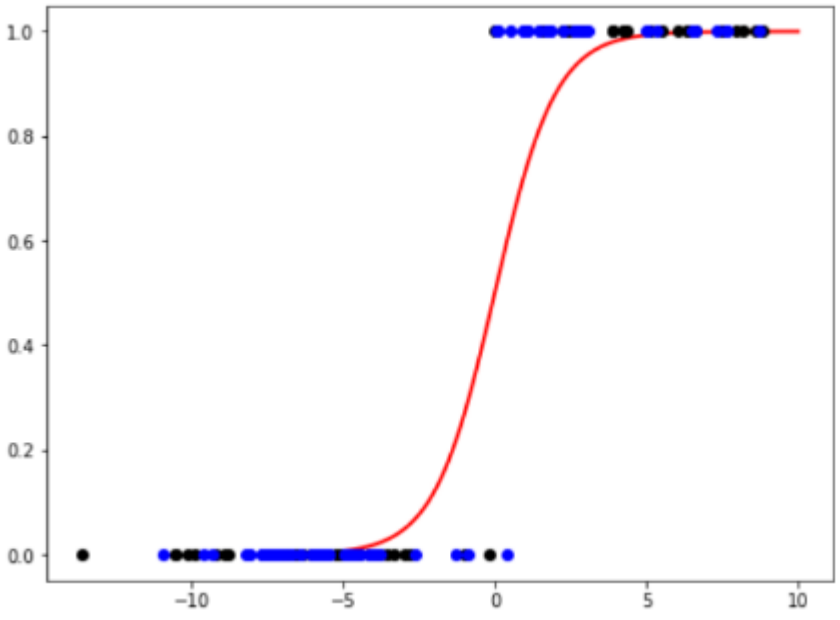
```
phat_train = clf.predict_proba(X_train)[:,1]
phat_test = clf.predict_proba(X_test)[:,1]

per_train = 0
for i in phat_train:
  per_train += math.log(i)
print("Perplexity train:", -per_train)
```

```
## Perplexity train: 405.1096623308393
```

```
per_test = 0
for i in phat_test:
  per_test += math.log(i)
print("Perplexity test:", -per_test)
```

```
## Perplexity test: 4170.543177363313
```

Convergence warning is due to hitting the iteration limit on the logistic regression model. By increasing iterations limit we can solve this warning. Plot:



Task b

```
lasso = linear_model.Lasso(alpha=0.1)

lasso.fit(X_train, y_train)
```

```
## Lasso(alpha=0.1)
```

```
for c, v in zip(spam_test.columns[:5], lasso.coef_):
  print(c, v)
```

```
## MISSING_FROM 0.0
## FROM_ADDR_WS -0.0
## TVD_SPACE_RATIO -0.0
## LOTS_OF_MONEY -0.0
## T_FILL_THIS_FORM_SHORT 0.21479500891265596
```

```
pred_train = lasso.predict(X_train)
pred_test = lasso.predict(X_test)

lassoper_train = 0
for i in pred_train:
  lassoper_train += math.log(i)
print("Perplexity train:", -lassoper_train)
```

```
## Perplexity train: 137.3988640113133
```

```
lassoper_test = 0
for i in pred_test:
  lassoper_test += math.log(i)
print("Perplexity test:", -lassoper_test)
```

```
## Perplexity test: 1326.7948999810167
```

alpha=0.1, for example, gives all but T_FILL_THIS_FORM_SHORT coef to 0/-0.

Problem 11

Task a

The authors say that the difference between discriminative and generative learning model isn't always clear even though many seem to think that discriminative is obviously better. The larger the dataset the larger asymptotic error the generative model has, over the discriminative model, but if the generative model has already reached its asymptotic error it usually performs better.

Task b

h_{Gen}: a generative model chosen by optimizing the joint likelihood of the inputs and the labels. h_{Dis}: a discriminative model chosen or by optimizing the conditional likelihood or minimizing the 0-1 training error.

Task c

In these graphs the x-axis is "m"; random samples from train/test splits, and y-axis which is the error of the Naive Bayes and Logistic Regression models. Results of these graphs are very mixed. Notable ones are `liver disorders` and `lenses`, where Logistic Regression starts performing better from a certain m, which is what we discussed in Task a.

Problem 12

Task a

For Naive Bayes assumption the data would have to be conditionally independent, given class. In this dataset, each datapoint is created independently from the normal distribution and hence the Naive Bayes assumption is fulfilled.

Task b

```
import numpy as np
import pandas as pd
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.dummy import DummyClassifier
from sklearn.neighbors import KNeighborsClassifier

ns = [2**i for i in range(3,13)]
data = [pd.read_csv(f'toy_train_{n}.csv') for n in ns]
data_test = pd.read_csv("toy_test.csv")
y_test = data_test["y"]

phat = lambda m: [
  m.fit(d.iloc[:, :2], d["y"]).predict_proba(data_test.iloc[:, :2])[:, 1]
  for d in data
]

accuracy = lambda p: (y_test * np.round(p) + (1 - y_test) * (1 - np.round(p))).mean()
perplexity = lambda p: np.exp(-np.mean(np.log(y_test*p + (1 - y_test) * (1 - p))))

m_NB = GaussianNB()
m_LR = LogisticRegression(penalty="none", solver="newton-cg")
m_LRx = LogisticRegression(penalty="l2", solver="newton-cg")
m_NBx = GaussianNB(priors=[0.4, 0.6])
m_D = DummyClassifier()
m_kNN = KNeighborsClassifier()

# Store results in a pandas dataframe
res_acc = pd.DataFrame(index=ns)
res_perp = pd.DataFrame(index=ns)
res_acc["NB"] = [accuracy(p) for p in phat(m_NB)]
res_perp["NB"] = [perplexity(p) for p in phat(m_NB)]
```

```
## C:\Users\isopo\AppData\Local\R-MINI~1\envs\R-RETI~1\lib\site-packages\pandas\core\arraylike.py:402: R
##   result = getattr(ufunc, method)(*inputs, **kwargs)
```

```
res_acc["LR"] = [accuracy(p) for p in phat(m_LR)]
res_perp["LR"] = [perplexity(p) for p in phat(m_LR)]
```

```
## C:\Users\isopo\AppData\Local\R-MINI~1\envs\R-RETI~1\lib\site-packages\pandas\core\arraylike.py:402: R
##   result = getattr(ufunc, method)(*inputs, **kwargs)
```

```
res_acc["LR inter"] = [accuracy(p) for p in phat(m_LRx)]
res_perp["LR inter"] = [perplexity(p) for p in phat(m_LRx)]
res_acc["NB prior"] = [accuracy(p) for p in phat(m_NBx)]
res_perp["NB prior"] = [perplexity(p) for p in phat(m_NBx)]
```

```
## C:\Users\isopo\AppData\Local\R-MINI~1\envs\R-RETI~1\lib\site-packages\pandas\core\arraylike.py:402: R
##   result = getattr(ufunc, method)(*inputs, **kwargs)
```

```
res_acc["Dummy"] = [accuracy(p) for p in phat(m_D)]
res_perp["Dummy"] = [perplexity(p) for p in phat(m_D)]
res_acc["kNN"] = [accuracy(p) for p in phat(m_kNN)]
res_perp["kNN"] = [perplexity(p) for p in phat(m_kNN)]
```

```
## C:\Users\isopo\AppData\Local\R-MINI~1\envs\R-RETI~1\lib\site-packages\pandas\core\arraylike.py:402: R
##   result = getattr(ufunc, method)(*inputs, **kwargs)
```

```
res_acc
```

```
##           NB      LR  LR inter  NB prior   Dummy     kNN
## 8     0.7537  0.7575    0.7601    0.7533  0.5452  0.6820
## 16    0.6875  0.6489    0.6295    0.6966  0.5452  0.4369
## 32    0.7681  0.7797    0.7839    0.7755  0.5452  0.6858
## 64    0.7302  0.7413    0.7396    0.7475  0.4548  0.7650
## 128   0.7768  0.7812    0.7810    0.7757  0.5452  0.7608
## 256   0.7780  0.7843    0.7841    0.7710  0.5452  0.7367
## 512   0.7822  0.7856    0.7857    0.7775  0.5452  0.7587
## 1024  0.7879  0.7880    0.7877    0.7859  0.5452  0.7638
## 2048  0.7845  0.7867    0.7867    0.7792  0.5452  0.7634
## 4096  0.7866  0.7877    0.7878    0.7813  0.5452  0.7662
```

```
res_perp
```

```
##             NB        LR  LR inter  NB prior     Dummy  kNN
## 8          inf       inf  1.635358       inf  2.018444  inf
## 16    1.921385  1.837043  1.848718  1.902800  2.018444  inf
## 32    1.683344  1.618673  1.645152  1.679266  1.993042  inf
## 64    1.746827  1.686106  1.672229  1.683012  2.015287  inf
## 128   1.596874  1.584198  1.586543  1.597164  1.996237  inf
## 256   1.595571  1.583013  1.585419  1.603049  1.991984  inf
## 512   1.587207  1.581450  1.580217  1.590528  1.994884  inf
## 1024  1.584385  1.576836  1.576088  1.585281  1.992196  inf
## 2048  1.581807  1.576362  1.576067  1.586784  1.994096  inf
## 4096  1.582219  1.574807  1.574726  1.586927  1.992274  inf
```

Task c

- The logistic regression with interaction terms didn't differ much from the LR with interaction terms.
- Probabilistic and generative models were the NB models, where as kNN and LRs were discriminative models.
- The perplexities had more interesting outcomes and differences. The higher the accuracy the lower the perplexity.
- The accuracies were so similar that no difference as in problem 11 were seen.
- Interaction term did not perform majorly different.
- Dummy classifier outperformed the kNN in toy_train_16 dataset. Otherwise the accuracy was very far off. Perplexity was however closer.
- Naive Bayes and Logistic Regression go hand in hand in the accuracy and perplexity.

Problem 13

Task a

I selected classification error rate, equation 8.5.

Task b

Classification error rate is specified as $E = 1 - max_k(\hat{p}_{mk})$. From the starting figure we can calculate the impurity as follows: $E_0 = 1 - max(\frac{15}{23}, \frac{8}{23}) = \frac{8}{23}$

Step 1: Split where x1 = 1, impurity: $E_1 = 1 - max(\frac{8}{15}, \frac{7}{15}) = \frac{7}{15}$

Step 2: Split where x2 = 2, impurity: $E_2 = 1 - max(\frac{7}{8}, \frac{1}{8}) = \frac{1}{8}$

Step 3: Split where x2 = 0.5, impurity: $E_3 = 1 - max(1, 0) = 0$

Problem 14

Task a

Classification boundaries for 1-NN are split in to six: $i_1 \in 1, 2, 3$, $i_2 \in 4$, $i_3 \in 5, 6, 7$, $i_4 \in 8, 9, 10$, $i_5 \in 11$, $i_6 \in 12, 13, 14$. Classification error is 0.

For 3-NN: $i_1 \in 1, 2, 3, 4, 5, 6, 7$, $i_2 \in 8, 9, 10, 11, 12, 13, 14$. There are 2 missclassified datapoints, so the error is $\frac{2}{14} = 0.14$

Task b

The higher the k the simpler the classification boundary. Extreme cases are the overfitting in 1-NN (every point and their neighbourhood matters) and with high k's where underfitting becomes the problem (classification boundaries are too simple and cannot classify correctly anymore).

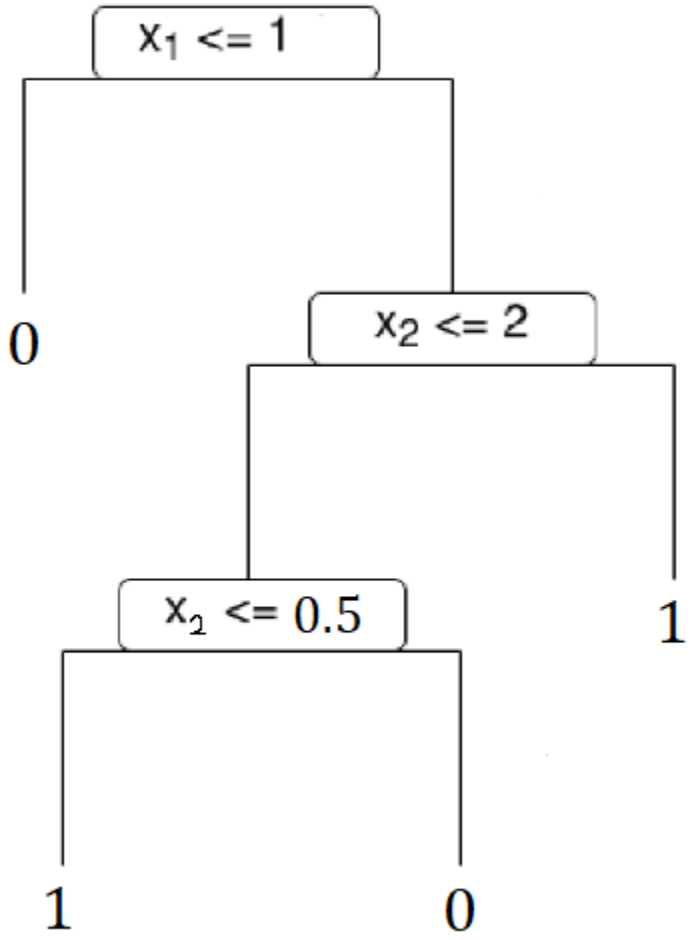
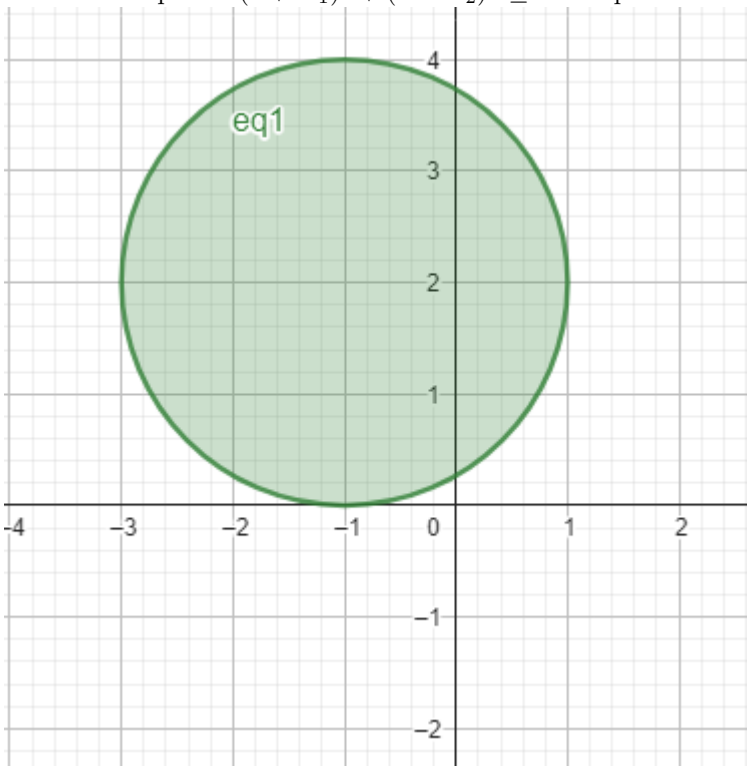


Figure 1: Decision tree

Problem 15

Task a

The curve is a circle. Points inside the circle are points $(1 + X_1)^2 + (2 - X_2)^2 \leq 4$ and points outside the circle are $(1 + X_1)^2 + (2 - X_2)^2 > 4$



Task b

(0,0): blue (-1, 1): red (2, 2): blue (3, 8): blue

Problem 16

I learned about the classification models and how to utilize them. Following the lectures, I was especially surprised how effective the naive Bayes was. I will most likely utilize it more in the future. Exercises were more challenging than sets 0 and 1, but in a good way. I really like the premade code with things that you need to add to get the full results. It is a way of visual learning and helps me understand the scope of the exercise.