MLA assignment 1

1 Make your own (10 points)

1. It is a reasonable assumption that the average of former grades would be a good predictor of future grades. Also, study program seems like a reasonable predictor - a student with prior exposure to e.g. mathematics and programming, would likely perform better than one without said exposure - denote the set of possible study programs, SP. The sample space would hereby be $\mathcal{X} = \mathbb{R} \times SP$.

2. The label space is the set of obtainable grades (denote this by G) in the course - $G = \{-3, 00, 02, 4, 7, 10, 12\}$. We then have $\mathcal{Y} = G$.

3. Notice that it has worse consequences when a stundent fails or passes undeservingly (re-examinations, reputation cost for the university etc.), than when a student deserving 12 receives 10 or vice versa. A reasonable loss function could therefore be a modified quadratic loss. Let $f(y', y)$ denote the indicator function of a misclassified fail/pass. We can then parametrize the loss functions in terms of $\alpha > 0$ by

$$\ell(y', y) = (y' - y)^2(1 + f(y', y)\alpha)$$

   Where α quantifies the severity of a misclassified fail/pass.

4. As we have a mix of numerical and ordinal data in our sample, a mixed distance function is most suitable. One such could be the sum of the Euclidean distance for GPA, and a scaled discrete metric for study program. That this is indeed a distance function, follows from the fact that sums of distance functions are distance functions.

5. With the notation in Seldins lecture notes, we would partition S into S_{train}, S_{val} and S_{test}. When we have trained our model on S_{train} and tuned K to K^* via S_{val} we would test the models performance by calculating $\hat{L}(h_{K^*}, S_{\text{test}}$, to get an unbiased estimate of $L(h_{K^*}) = \mathbb{E}[\ell(h(X), Y)]$

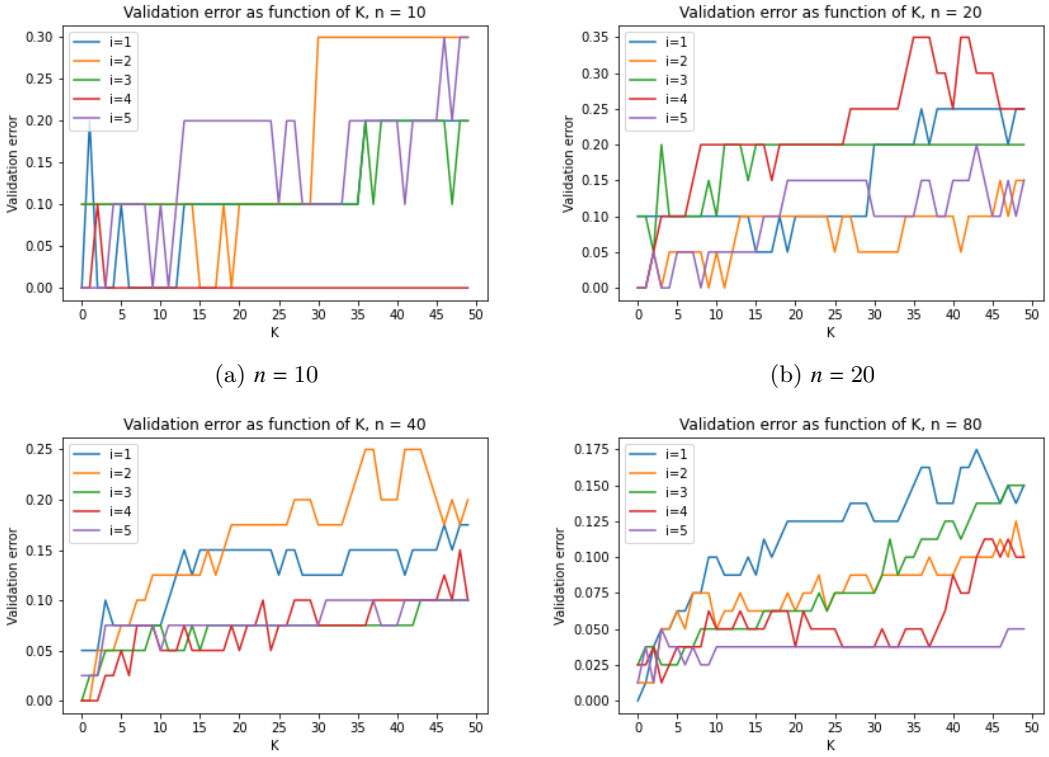
6. The major concern of the algorithm, is that it assumes i.i.d. samples. If after some time data is not drawn from the same distribution as when the model was trained, it's predictive power will worsen. A way to alleviate this, is to train the model anew on more recent data, or perhaps include time of sample into $\mathcal{X}$, and incorporate the time difference into our distance function.

2 Digits Classification with K nearest Neighbors (45 points)

Task 1

Results

We present our results below. For the implementation of K nearest neighbors, ties where broken with the algorithm predicting 5. Zero-one loss was used as the loss-function. Firtsly, we plot the validation error as a function of K for varying n and i. The validation sets are formed as $S_{\text{val}}(i, n) = \{100 + i \cdot n + 1, \cdots, 100 + (i + 1) \cdot n\}$.



(a) $n = 10$

(b) $n = 20$

(c) $n = 40$

(d) $n = 80$

Figure 1: Validation error as a function of K with varying n

Secondly, we plot the variance of the validation error as a function of n. Seeing as the true mean of the validation error is unknown, we lose one degree of freedom in estimating the variance, and therefor divide by $n - 1$ instead of n.

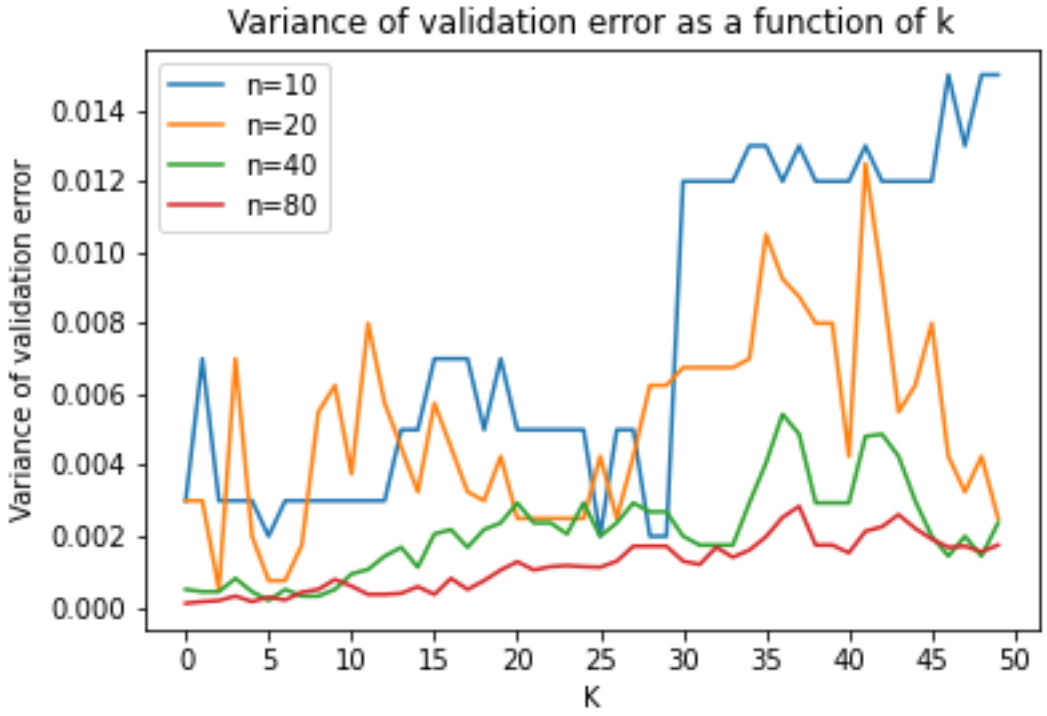


Figure 2: Variance of validation error as a function of n

Validation error as a function of n

Looking at figure 1, we notice that for $n \in \{10, 20\}$ the estimate of validation error is very sensitive to which validation set is used. Notably, when $n = 10$ and $i = 4$ the model has near constant 0 validation error for $K \in \{1, \cdots, 50\}$. In stark contrast when $n = 10$ and $i = 4$, we have that for $K > 30$ we have an estimated validation error of 0.3 - a rather large deviation from 0. The fluctuations of the validation errors for small n are also extremely jagged. In other words the estimated validation error is very sensitive to small changes in K. This is of course clear, as changing K to $K + 1$, might produce a different prediction - which would lead to a change in validation error of 0.1 and 0.05, when $n = 10$ and $n = 20$, respectively.

For $n \in \{40, 80\}$ the change in validation error as a function of K is much smoother. The distances between lines are also smaller. Already for $n = 40$ we see the lines following each other much more closely, and we do not have as many large discrepencies of validation errors between i's. When $n = 80$, their is not much sensistivity to which i we are looking at - execept for $i = 5$ being a bit of an extreme case. For the most part the validation errors follow each other within a 0.05 bound.

To quantify these observations, we can look at figure 2. We immediately notice that the variance of validation error is much larger for $n \in \{10, 20\}$ compared to $n \in \{40, 80\}$ for almost all K. This corresponds very nicely with the 2 above paragraphs. Furthermore, the variance of the validation error also becomes much more sensitive to K, when n is small. This is of course not desirable, as the validation error essentially decides which K in $K - NN$ is optimal. When the validation error varies largely across validation sets, our choice of K does not generalise very well.

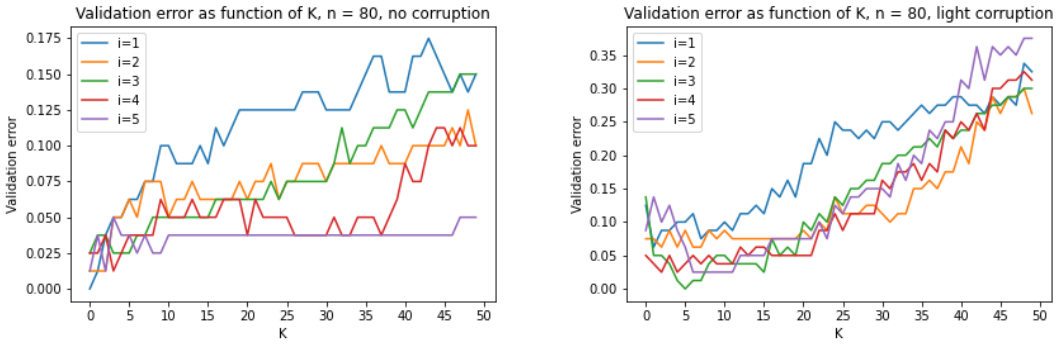
Prediction accuracy as a function of K

Per the above section, it is most reasonable to say something about prediction accuracy as a function of K, when $n = 80$. In this case, the prediction accuracy of $K - NN$ seems to be decreasing fairly monotonically as a function of K, with an eyeballed optimal value of around 2-3, for almost every validation set. It seems looking at more than a couple of the nearest neighbors introduces a lot of unneccesary uncertainty in the prediction.

Task 2

Results

We plot the validation error as a function of K for the orignial, the lightly corrupted, the moderately corrupted and the heavily corrupted dataset.



(a) No corruption

(b) Light corruption

(c) Moderate corruption

(d) Heavy corruption

Figure 3: Validation error as a function of K for varrying levels of corruption

Discussion

From figure 3 we see that, as one perhaps would expect, that the more corrupted the images are, the higher the validation error is and thereby the lower the prediction accuracy is. It is worth noticing, that the more corrupted the images are, the worse KNN performs for small K. Espicially for the moderately and heavily corrupted images, the validation error for $K \leq 5$ is very poor, which is interesting as these are the K's favored on the lightly and non-corrupted set. Furthermore, the degree of corruption also seems to introduce more instabillity of prediction error around the optimal value

of K. In the none-corrupted and lightly corrupted datasets K's around the optimal K's still produce prediction errors not too far from the prediction errors of the optimal K's. Compare this to the moderately and heavily corrupted datasets, where lowering or raising K by, say 5, completely destroys the accuracy of KNN.

Linear Regression (45 points)

Task 1

We implement the linear model (with an exponential transformation back to the original label space for the fitted values) as a class. The code is documented below. Due to the ambiguity of the problem statement - source code being deliverable for this exercise, while on the front page it says not to include code in the report, I have included the code that defines the Linmod_exp_trans class. The rest of the calculations in this exercise follows from this class and are not include in the report, but in the .zip-file.

```
import numpy as np
#Notice that the class expects the labels to be log-transformed
class Linmod_exp_trans:
    def __init__(self,x,y):
        X = np.c_[np.ones(len(x)),x] #Model matrix expansion to add 1's for the intercept
        coeff = np.linalg.inv(X.T@X)@(X.T)@y #calculate the coefficients for the ols regression
        self.coeff = coeff #assign coeff to self
        self.fitted = np.exp((self.coeff[0] + self.coeff[1]*x)) #Fit values.
        # Notice the exponential function particular for this exercise
        self.mse = np.mean((np.exp(y)-self.fitted)**2) #Calculate mse over original labels
        self.R2 = 1-self.mse/np.mean((np.exp(y)-np.mean( np.exp(y) ) )**2)
        #Calculate R^2 value over original labels
```

Task 2

We load in the data, log-transform the PCB variable, and fit the model $h'(x) = ax + b$ on $S' = \{(x_1, \ln y_1) \cdots (x_N, \ln y_N)\}$.

This yields the parameters, $a = 0.2591$ and $b = 0.0315$. Additionally we have that the mean squared error of $h = exp(h'(x))$ over S is 34.8356.

Task 3

We plot the data and the model output, as the logarithm of the PCB concentration versus the age.

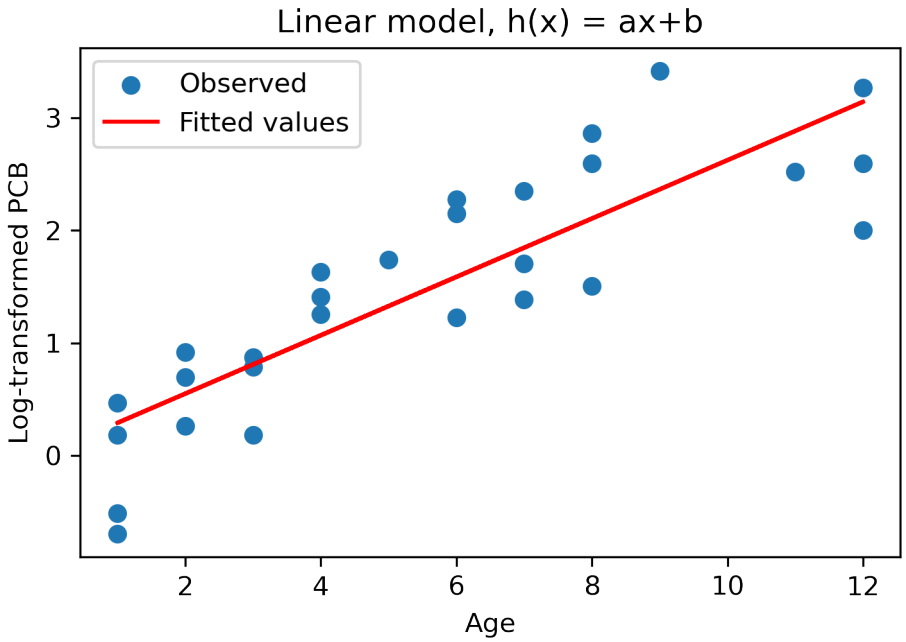


Figure 4: log(PCB) versus age for $h(x) = \exp(ax + b)$

Task 4

Notice that we can rewrite

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - h(x_i))^2/N}{\sum_{i=1}^{N}(y_i - \bar{y})^2/N}$$

We recognize the second term as the mean squared error of the model, divided by the mean squared error of the model that always predicts the mean of the label space in the training, data given new data. R^2 Therefore measures how well a prediction rule h fits data compared to using the mean as the prediction rule, measured in mean squared error. A high value of R^2 indicates that the model fits data well compared to the mean of the label space. We have that

$$R^2 = 1 \Leftrightarrow \sum_{i=1}^{N}(y_i - h(x_i))^2 = 0 \Leftrightarrow y_i = h(x_i), \forall i$$

This simply says, that every predicted label is equal to every observed label in the training data. On the other hand we have that

$$R^2 = 0 \Leftrightarrow \sum_{i=1}^{N}(y_i - h(x_i))^2/N = \sum_{i=1}^{N}(y_i - \bar{y})^2/N$$

This states that the mean squared error of h on the training data is equal to the mean squared error of the prediction rule that always predicts $\bar{y}$. In other words when $R^2 = 0$, $h(x_i)$ fits the data precisely as well as the mean of the labels.

Whether R^2 is negative depends entirely on the prediction rule. Consider observing $(x_1, y_1) = (1, 1)$ and $(x_2, y_2) = (3, 3)$. We have that $\bar{y} = 2$ and that $(y_1 - \bar{y})^2 + (y_2 - \bar{y})^2 = 2$. Now consider a prediction

rule that constantly returns 0, that is, the prediction rule $h(\cdot) = 0$. We have that $(1-0)^2+(3-0)^2) = 10$. We can calculate

$$R^2 = 1 - \frac{10}{2} = -4$$

Obviously, $h(\cdot)$ is a terrible prediction rule, but in the definition of R^2 there is nothing that gurantees it's positivity. If we however consider linear regression done through minimizing the sum of squares, where we allow a constant term (i.e. $h(x) = ax + b, a, b \in \mathbb{R}$), it is easy to see that $R^2 \geq 0$, as we can simply set $a = 0$ and $b = \bar{y}$.

For this model we have that $R^2 = 0.357$.

Task 5

We build the model $h(x) = \exp(a\sqrt{x} + b)$ by applying $x \mapsto \sqrt{x}$ to the input data before performing the regression. We get the following plot of the observed data versus the model

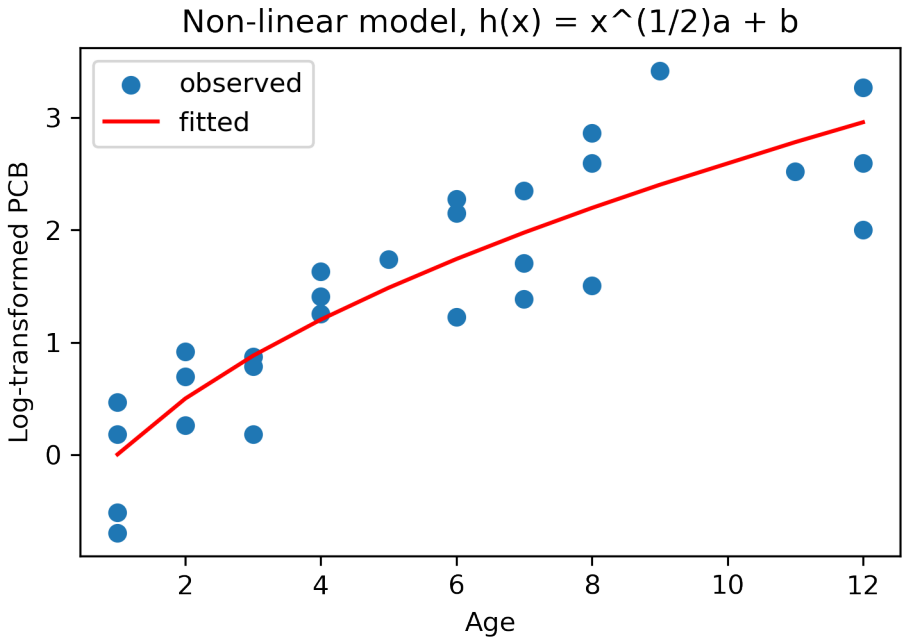


Figure 5: log(PCB) versus age for $h(x) = \exp(a\sqrt{x} + b)$

For this model we have that the mean squared erros is 28.0844 and that $R^2 = 0.4816$.

We notice that the R^2 value for the non-linear model is higher than that of the linear model. In this regard the non-linear model fits the data better than the linear model.