Machine Learning

Pen and Paper 2

Task 1.

Subtask 1.

After plotting the features and the target values, it was obvious to me that the features follow some type of absolute function, with the first having an offset of 1, and the second having the offset 2. After I added two additional features by having a absolute value of coordidinate and than subtracting the offset, I already had really small error, but to make it under the threshold, I kinda experimented with different things that made sense to me. I added the absolute value of difference between two features, the absolute value of their multiplication and their norms. Using those features, I got an error around 0.007.

As you can see from the next plot, the optimal number of features I calculated was 6, which gives me an error of around 0.1.

Subtask 2.

I again plotted the features and target values, and got kinda similar data map as in the first subtask, only that target values were in some way discrete. It also had a shape of an absolute function so I added the absolute value of difference between two features, an absolute value of the exponential, the norm and the cosine after some experimenting to achieve the accuracy of 95%. I also scaled the features with Standard scaler. After calculating the confusion matrix we can see that every class was fairly successful except the 4th one where it correctly predicted 4 and incorectly labeled 12 data points,

Task 2.

Subtask 1.

When I calculated the MSE on training and test data with non-polynomial features, I got quite similar MSE-s being around 0.5, while with degree 2 polynomial features I got a 0.3 MSE on my training data while getting MSE of 20 on the test data. That is the classic example of overfitting that we discussed in class.

Subtask 2.

By using Monte Carlo cross valdation, I got a average MSE of around 4, with the variance being around 68, which was weird for me at first, but when I plotted the MSE's, you can see that this model is heavily dependant on the part of data you are testing it on, so the variance will naturaly be big over large

number of iterations.

On the other hand, with K-fold cross validation I got much better results, getting an average MSE of around 1.35 for 10-fold cross validation and an MSE around 1.33 for 20-fold cross validation. The variances were also really low, first being under 0.0003 and the second one being even smaller so the method gives us consistent results. I got much lower MSE's than in the last task, but I haven't noticed much difference between two fold sizes I was using.

Task 3.

The first model gave me an optimal degree of 5 which is logical due to testing only on training data, so the model gets used to it and preforms well on it. The error I get on the training set is around 0.18, while the error on the test set is over 930157590284.

The second model gave me an optimal degree of 1, with the error growing rapidly in proportions to the degree. Using this degree on the validation set, I get an error of around 0.52.

With 10-fold and 20-fold cross validation I also get a degree of one, which is the final optimal one which you can see from the last plot of my code.

The optimal degree didn't vary that much, only for the first model so I would never use it because of overfitting, but if I had to choose I would always choose k-fold.