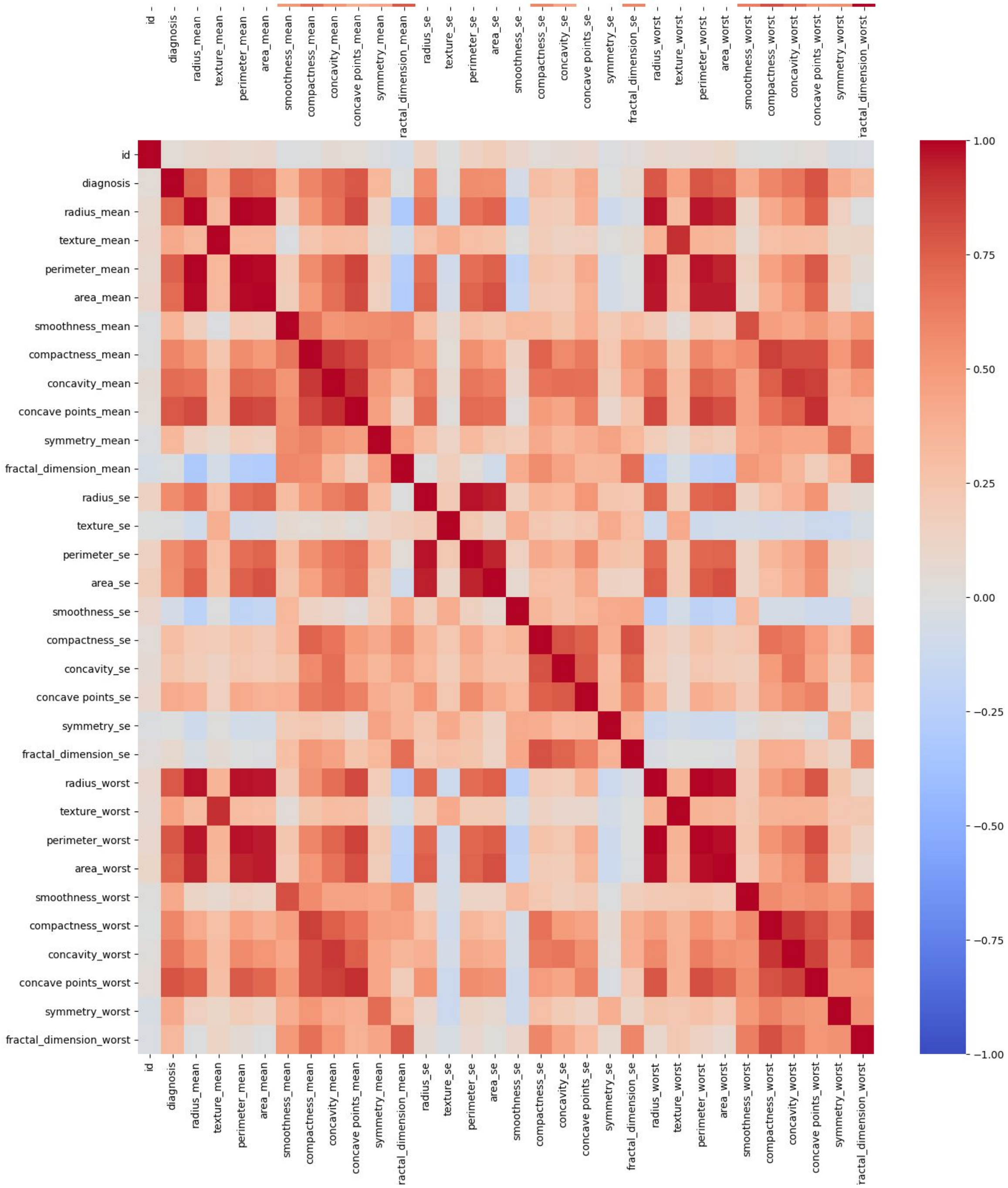
Cody Morgan
CS395
28 Oct 2020
K-Nearest Neighbor

Code Overview

Data is loaded (line186) and is processed into a dictionary using the label as the key to a list of data. Characters are converted into their asci representation. The data is then checked for missing data (line 211) and is standardized using linear standardization (line 220). The plots are generated (if option is set in settings) along with the heat map (line 227). The other plots will be attached in a folder (as there are 32 of them).



The user is also prompted if s/he wants to view any cross correlation scatter plots. The Data is then split according to the percentages set in Settings (line 282). By default, I am using a 60/20/20 split. After this K is calculated using the validation set and taking the highest K (line 288). Then, accuracy is calculated and an attempt to improve accuracy is made by ignoring various types of data. This may lead to over fitting so the user is prompted if s/he wants to keep this set or use the original set (line 338). Finally, the user can (optionally) input data for evaluation (line 375).

Accuracies

Using all of the data attributes (except “ID”) I get an accuracy of 98%, by ignoring smoothness_worst I was able to increase the accuracy to 99%. I think that ignoring too many attributes could lead to overfitting - but ignoring just this one I think is ok - so I elected to keep this change. Looking at the heatmap, we could ignore some of the highly cross correlated attributes such as "perimeter_mean", "area_mean", "radius_worst", "perimeter_worst", and "area_worst" - this gives me an accuracy of 93%. By using the same method as before we could further ignore 'texture_mean', 'smoothness_worst', and 'concave points_worst to raise the accuracy to 98% - though I think at this point we risk overfitting the data.

Improvements

I have already opted to use squared distance instead of distance for enhanced speed. I have noticed that I am actually calculating distance more times than I need to - but this doesn’t seem to have a noticeable impact on performance - I could rewrite this for better efficiency. I currently pull the training, validation, and testing information in order from the data provided. One thing that could be done to improve accuracy would be to randomize these selections - or possibly even do this several times - training and testing with different datapoints. This might help fight overfitting and also help to suss out those attributes that really aren’t as important.