K-Nearest Neighbor (KNN)

Parameter:

N\_neighbor: the number of neighbors used to calculate the class

How to find the n\_neighbor value:

1. The most used equation for calculating a k value is the sqrt(number of data points). If it comes out of even you can add 1 or subtract 1.
2. Using gridsearchcv to find the best k value to optimize a certain metric of the model

Weights: if the distance between neighbors and the point is weighted

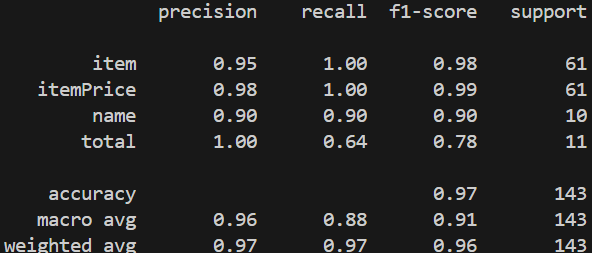
When the distance parameter is activated then point closer to the unknow point are valued more then points further away. This help minority class be chosen even when you use high k-value to avoid over fitting.

Algorithms: the tree used to calculate the nearest neighbor

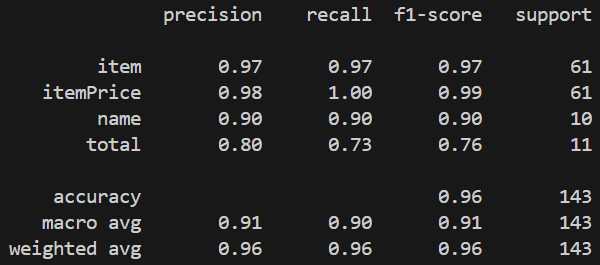
P: the power of the root for the Minkowski equation

The basic equation for the Minkowski is \_\_\_ the p value impacts the strength of the root. If the p=1 then the distance equation is Manhattan distance, p=2 is the eculidain distance and p>3 is just minkowski equation.

When running knn with (n\_nieghbers = 25,weight=’uniform’, algorithm=’auto’, p=2) you get the classification report shown below.

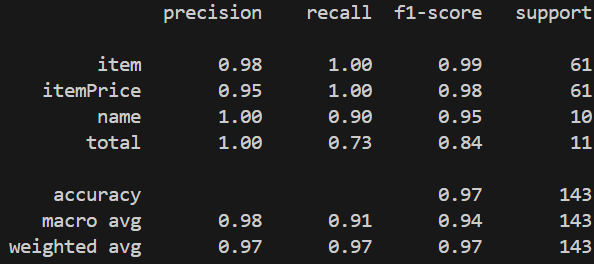


The f1 score indications that the model has a easier time predicting item and itemprices rather than minority class like name and total. The reason being that out dataset is imbalance since there higher amount of item and itemPrice than name and total. There two ways to even out your data to improve your model is either oversampling or undersampling the data. Oversampling is the process of removing data points from majority class to even out the data while undersampling adds data points to minority class to even out data. One undersampling method is smote that synthinic add data points by creating lines between minority labels and add points. With smote you can either have the data point be complete even or increased at a ratio. After running smote we get the classification report below that shows an overall 1% decrease in accuracy but that wanted increase in racall for total but with the sacrifice of total precision. The data correlates with how smote works as the increase in data points allows the machine learning model to have a more exhaust criteria for the label total so it predicts total less time unless it certain it total represent the increase of total. However since the synthetic points aren’t real totals the machine learning model has problems predicting actually total labels leading to the decrease of precision.

7

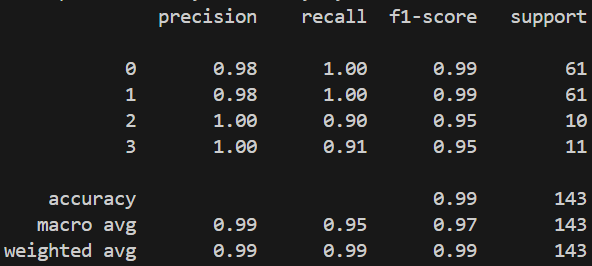
Knn hyper-tuning

Another method is hyper-tuning the parameters. You can manually hyper-tune each parameter. Here is the graph created from hyper-tuning all 4 parameters. Using a stafiedkfold I was able to simutate running grid searchcv on each parameter by itself with the other at the default values. We can then graph each parameter to th recall score produced and analyze the graph to find the most optimal parameters. Looking at each graph the k-value with the highest recall\_score was at k = 7, the p-value with highest recall\_score is p=3, the wiegths had the same recall value so we pick the default. If you run a model with these parameters you get the classifaction report below. There a overall increase in the f1 macro score and accuracy. Compare to smote the model increase the recall score of total the same from default however didn’t sacrificing the precision of each label. Giving the overall higher scores since the model had a better job create a criteria for total since they wasn’t systenic data add to improve the imbalancing of the data.



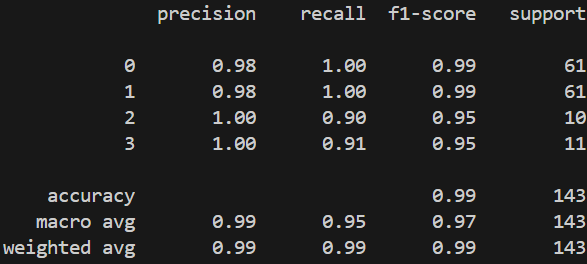
GridscearchCV

The problem with manual optimizing the parameters is that they aren’t independent from each other. That where gridsearchcv comes into play its as it check every combination of the parameters if you computer Is strong enough to handle the computation. Running gridsearchcv on our model you get the below classifaction report.



knn onevone

The onevone classifier allows binary classification models excucate on multiclass datasets. It achieves this conversion by having multiple models run with two class at a time. Knn by default is multiclass classification model. However, knn model works better when there only binary outputs. The problem with onevone is that it takes a long time since it has to run each combination of models get the score and compare to out its most accury model. After running a onevone classifer on knn the f1 score is at it highest since it onevone.



Overall with hyperparameter tunning your able to get a very high accuracy and f1 score model. However, I believe that our model is highly overfitted as it prod

naïve baye

parameter:

var\_smoothing:

different equation:

Grassuian:

hyper-tuning

testing on ocr data