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

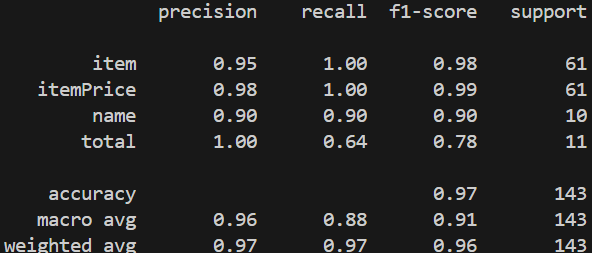
When the distance parameter is activated then point closer to the unknow point are valued more then points further away. This help minority class be chosen even when you use high k-value to avoid over fitting.

Algorithms: There three algorithms used by KNN to find the closest point. This doesn’t actually impact the performance of the KNN model but the speed of the model. The first one is brute which is usually the slowest where the KNN loops through all the points and finds the closest point. The other two algorithms are little smarter they use trees to help optimize the speed of the algorithms.

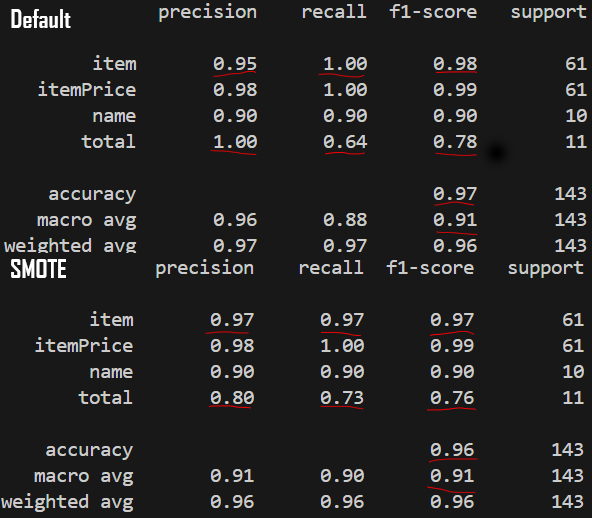
P: the power of the root for the Minkowski equation

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 the parameters (n\_nieghbers = 25,weight=’uniform’, algorithm=’auto’, p=2)the classification report below is produced



The figure indicts that the model is better at predicting item and itemprice as there f1 scores are higher than total and name. There is an easy explanation for this occurred as the issue lies within our dataset as there a larger number of items and itemprices than names and totals causes our dataset to be imbalance. As pervious stated some parameters in KNN help with imbalance data, mainly weight, however there two methods commonly used to mitigate the issue of imbalanced data: oversampling and undersampling: Undersampling is the process of removing data points from the majority class to even out the data while oversampling adds data points to minority class to even out data. Between oversampling and undersampling oversampling is preferred since undersampling would increase the likelihood of the model overfitting the data since the dataset is already smaller than a typical dataset. One oversampling method is smote that syntheticly adds data points to your dataset for minority classes. Smote creates the new data points by creating slope lines between two datapoints and add points within that line. After running our training set through smote and then running the new dataset through KNN with the pervious parameters which outputs figure 12. Looking at figure 12 there’s an overall 1% decrease in accuracy with no change in f1 macro score but with an increase in the f1 score of total where recall increased but precision decrease. The data correlates with how smote works as the decrease in precision comes from the fact that the increase data points allows the knn model to predict more total but that leads to an increase in false positives. Additionally, the increase in recall comes from the fact that less items that are total are be classed as other class decrease the false negatives. Choosing to use smote depends on if the objective is to over predict the minority class or underpredict the minority class.



Knn hyper-tuning

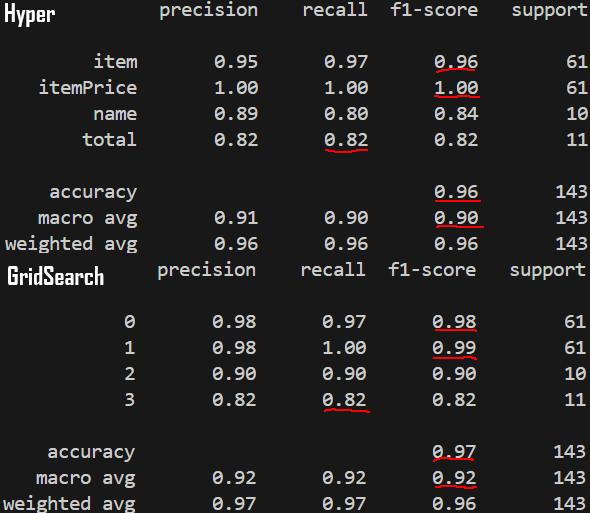
As mentioned in the pervious section the KNN parameters themselves can help with imbalance datasets to find the best parameter for our dataset we can hypertune the model. Using stafiedkfold and some for loops the best parameters for KNN can be manually obtained. When hypertunnig one metric is chosen to help focus the hypertuning. During the rest of this section we will use recall\_score since we want to over predict the minority class. Since each parameter manually loop through, a graph can be created, plotting the parameter value against it recall\_score, so that their a visualization of how each parameter impacts the model. Looking at figure 13 as the k value increases the recall\_score decreases. The relationship is connected to our imbalance dataset because total is such a small amount of the dataset if the k-value is larger the majority class are more likely to be chosen simply because there more of them increase false negatives since total are wrong classed decrease recall. That leaves the most optimal K-value as 3. Looking at figure 14 the p-values isn’t as linear as the k-value as its more impacted by the dimensionity of the data. Since the dataset has 8 feaetures we have 8 dimension so the usually distance formula would limit the distance found by the knn model making the most optimal p-value 4. Last, in figure 14 uniformed weights vs distance weights shows that distance is superior as stated in th eknn parameter breakdown distance is design to help combact imbalance datasets. Running knn with the found optimal parameters (n\_nieghbers = 3 ,weight=’distance’, algorithm=’auto’, p=4) the classifaction report below is produced. There is an overall decrease in the f1 macro score and accuracy compared to the default model. 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.

A screenshot of a computer screen

Description automatically generated

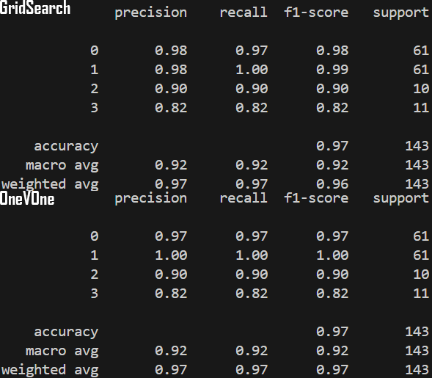
GridscearchCV

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. The best parameters are k = 3, p = 1, w= distance, a= auto. The k value makes since we have a minority classes and having a higher k values would make it hard for model to predict the minority class since the larger class would have a higher chance of being the majority. The p value is interesting since your think the default distance formula would be fine. However, the more features your data has the more dimension the graph that the points are plotted has. Weight being distance helps the model predict minority data points since if the unknown data is right next to a one minority class but the two majority class without distance it would say the point is the majority class but distance makes the model realize if it close to the minority class it most likely the minority class.



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 and average is the same as the hypertunned knn with the classififer. The only difference is that item and itemprice but the difference even out.



naïve baye:

Naïve Bayes uses the Bayes theorem with a “naive” assumption of conditional independence between every pair of features given the value of the class variable. Not the best model since it hard to find data that truly independent in the real world. The most important choice is the equation your naïve Bayes uses. Choosing the right equation is based on your desired data and output.

Grassuian: The basic equation that assumes your data follows the Gaussian distribution. The only parameter is var\_smoothing which controls how tight the curve is towards your data. This was the best for our data as Grassuian model works with continuous data that has multiple outputs.

A screenshot of a computer screen

Description automatically generated

Multinomial: Best used when you have a binary output and your data is discrete since it assumes your output can be estimated with relative frequency counting. The reason multinomial complete failed is because our data isn’t the best for the model. Multinomial works when there a word count. For example, if we have a set of text, if we count how frequency certain words show up, we can find the probability that the text is spam of not using multinomial. However, our data is normalized coordinated points that tell the computer where on the document it was found.

A screenshot of a computer screen

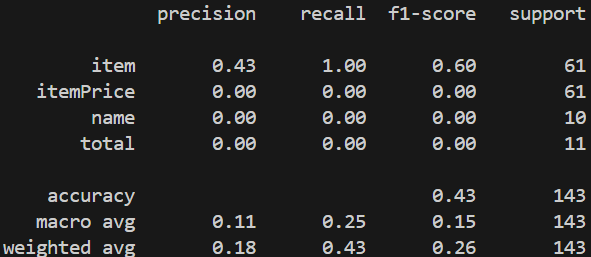
Description automatically generated

Bernoulli: Best used when you have a binary output and data since it assumes that your data follows the multivariate Bernoulli distributions. Since our data doesn’t follow Bernoulli distributions so it doesn’t work with Bernoulli model. Data that works best with Bernoulli is data that’s uses 1 and 0 to show the prensence or absence of the certain feature in that class.

A screenshot of a computer screen

Description automatically generated

Categorical: It’s a beef up version of Bernoulli that allow for features that hold more than binary value of 1 or 0. However, the values most be discrete and ordinal as 1 < 2 < 3. Since, our data doesn’t follow the restriction it gives out a poor classification report.



Complement: This uses the complement of the likelihood of getting each class. For example, if there was a 20% of getting a total complement model will store the percentage as 80% since there a 80% chance the unknown is not the total class. We then calculate the likelihood that the unknown value is not the class using naïve bayes and select the class with the smallest rate. Since we have minority class the percent an unknown is not one of the minority class is larger than a majority class without knowing its values. The main problem when running complement model on its own is that there no real difference between total and item or item price since on some receipts they can be in the same locate as a total on another receipt. So, by having the model only look at two class at a time it better able to distinguish between some item and itemprices with the total and make some predicts. However, it still scores lower than using the default

. 