DATA MINING ASSIGNMENT

Student Name: Pradyumna Agrawal

Student Number: 20046165

Email Address: pradyumna.agrawal@postgrad.curtin.edu.au

School/Department: School of Electrical Engineering, Computing and Mathematical Sciences

Unit: Data Mining COMP5009

Lecturer/Tutor: Dr Paul Hancock

Date Due: 15th October, 2021

Abstract

This report presents analysis of 'Assignment2021.sqlite' dataset. The dataset has 1200 rows and 32 columns. One of the columns is, 'class'. The class column has a total of three unique labels. These labels are, 0, 1 and 2. The business goal is to create a classification model that has accuracy score greater than 75%. The next section presents steps for preparing the dataset. It is followed by data classification section. Four models are trained in the classification section. These models are, Random Forest, Gaussian Naïve-Bayes, k-NN, and Decision Tree. Stratified k-fold cross validation is done for hyperparameter tuning for each model and selection of model with most appropriate hyperparameters. The summary of all the four models is given in the table below –

Classifier	Name and Value of first Hyperparameter	Name and Value of first Hyperparameter	Average Accuracy
Random Forest	n_estimators = 200	Criterion = entropy	0.8207
Decision Tree	min_samples_leaf = 3	Criterion = entropy	0.7121
k-NN	n_neighbors = 5	Distance = Euclidian	0.688
Gaussian Naïve-Bayes	var_smoothing = 1	None	0.6604

The best accuracy is given by Random Forest Classifier and is equal to 0.8207. Therefore, this report concludes that the **estimate of accuracy is 82.07%**.

The key takeaway from this analytics exercise are –

1. It is important to understand data at hand for training better classifiers.
2. One can take the simplest route first to create a baseline model. If the baseline model in itself fulfils the business goal, then a lot of time will be saved. If the baseline model is not able to do that, then few tweaks can be brought into the model for better results.

Issues with Data –

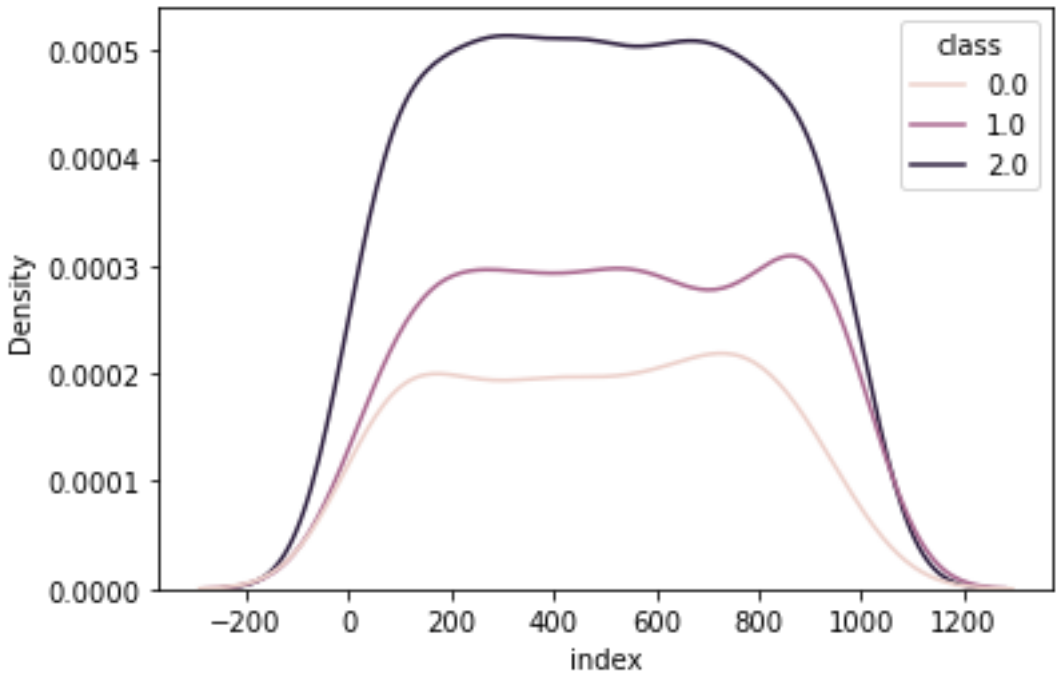
1. Index dimension is not useful.
2. Att00 and Att09 have missing value.
3. 'Att07', 'Att10', 'Att11', 'Att18', and 'Att28' are duplicate attributes.
4. 'Att08', 'Att01' and 'Att09' are categorical attributes with datatype as object.
5. The class ratio is 2:3:5 (data imbalance)

Methodology

Data Preparation

The data preparation methodology follows 'keep it simple' approach. That is, at first trial, choose the easiest option available. If the goal of 75% accuracy is not after this approach, then bring changes in data to improve efficiency.

1. It was observed that the first dimension is 'index'. It only has numbers from 1 to 1200 in sequence. Pandas have indexes by default. Therefore, this dimension is redundant for exploratory purpose. It is also redundant from classification purpose as the classes of data are not dependent on their location in dataset as shown in the graph below –



   The frequency distribution for all three classes from index 1 to 1000 is approximately normal. Therefore, it makes the most sense to drop 'index' from the data frame.

2. 2 dimensions (Att00 and Att09) have missing data. For Att00, 0.75% data is missing. For Att09, 48% data is missing. Available options to deal with the missing data are, delete rows with missing data, delete columns with missing data, and impute data.

a. Att00 – As only 9 rows have missing value for Att00 dimension, we will lose 1191 available values if we drop the entire column. Therefore, it makes more sense to delete those particular rows. The reason why imputation has not been selected as the best course of action here is that our strategy is to choose the easiest option at the first trial.
b. Att09 – 581 rows have missing values for Att09. Dropping all the rows would lead to loss of around 50% of the training dataset. Therefore, it makes more sense to drop the dimension Att09 instead of individual rows.

3. Duplicate attributes can bring bias in the training models. Therefore, it is advisable to remove them (Sarracino and Francesco 2016). However, it was found that there are no duplicate instances in the data frame.

4. There are three dimensions in the dataset with datatype as object. It can be inferred that they are categorical variables as they have limited number of unique instances as shows in the table below –

Dimension	Number of Unique Instances
Att01	10
Att08	3
Att29	7

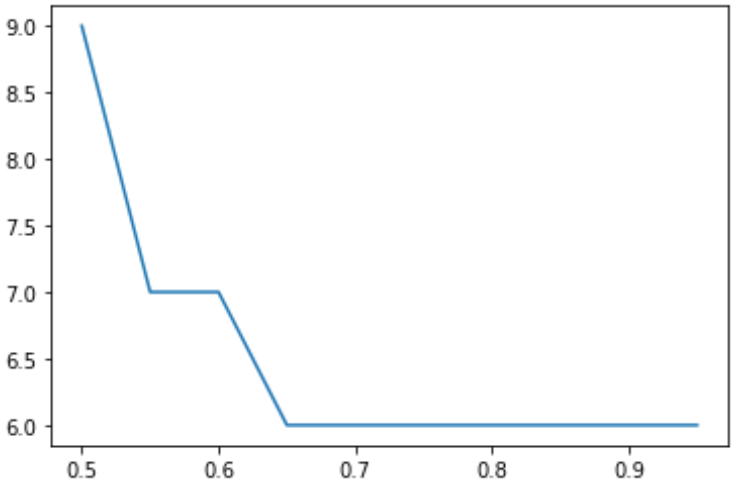
According to Tom (2018), KNeighborClassifier() has a weakness that it is not able to handle categorical data. We will be using this classifier for predictions. Therefore, it is important that we deal with categorical data at this stage. There are various encoding methods available for handling categorical data. For example, one-hot encoding and mean-encoding. Dataset already has two dimensions with binary data, therefore, one-hot encoding scheme will create more binary dimensions.
This project uses one-hot encoding. This encoding method creates a dummy dimension for every unique value of a categorical variable (Senger 2018). Therefore, it will create 10 + 3 + 7 = 20 dummy variables and remove the original 3 categorical variables.

5. Appendix-1 shows spread of data for different dimensions. It must be noted that there is a contrast among ranges of various dimensions. As k-NN is a distance based classifier, it is impacted by the scale of dimensions (Sharma 2019). Therefore, it is necessary to scale all the dimensions. There are two possible options for scaling. The

first is min-max scaling and the second is standardisation. Min-max scaling bring the value of data between 0 and 1. On the other hand, standardisation maps the values to standard normal distribution.

This project uses standardisation as most of the numerical columns are approximately normally distributed.

6. Highly correlated dimensions are not useful for classification problems (Tolo̧şi 2011). They can impact stability of classification models and they also increase processing time (Tolo̧şi 2011). Therefore, it was decided to drop dimensions that are causing multilinearity in data. The graph below shows the number of dimensions that would be removed for different cut-off values of correlation coefficients –



It can be seen that there are six dimensions that have correlation coefficient almost equal to 1. After removing these six dimensions, there are no dimensions with correlation coefficient greater than 0.65. Therefore, we remove these six dimensions. These dimensions are, 'Att07', 'Att10', 'Att11', 'Att18', 'Att28', 'Att08_YIFL' . This also solves the issue of duplicate dimensions.

7. The data preparation is now complete. We keep aside the last two-hundred rows for prediction. We have a dataset with 991 instances and 32 dimensions for training and testing.

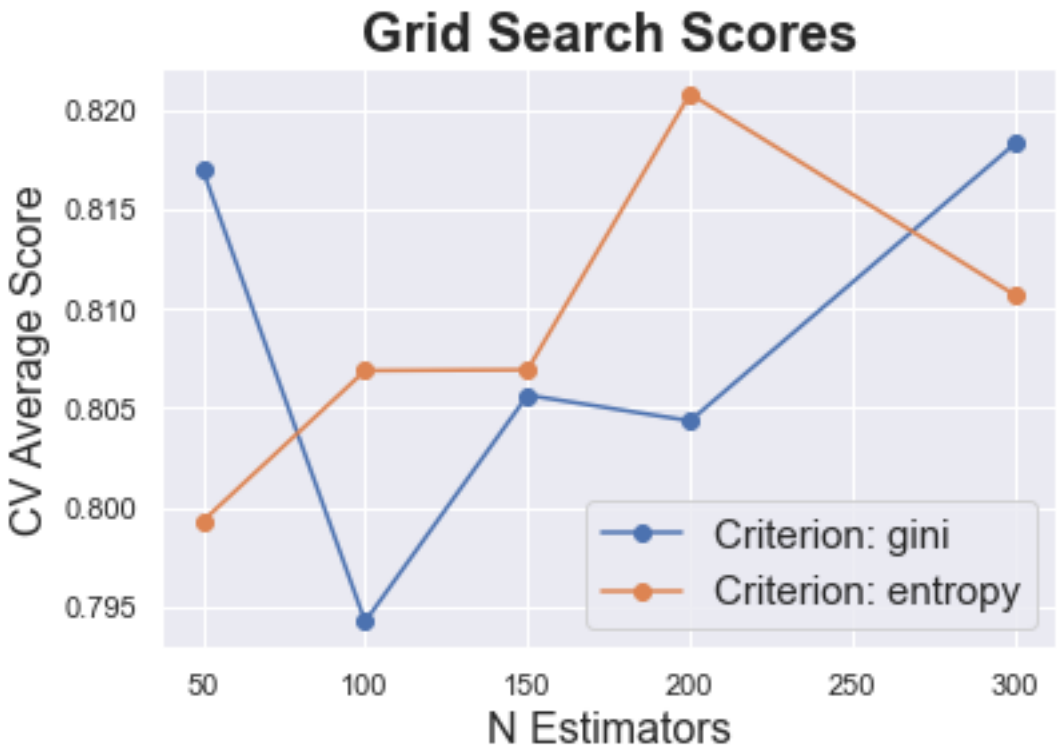
Data Classification

The first task is to perform a test-train split. We use 20% of 991 rows as testing data and the remaining 80% for training. Four models were trained. These models are, Random Forest, k-NN Classifier, Naïve Bayes Classifier, and Decision Tree Classifier.

1. Random Forest Classifier – Random Forest Classifier are a collection of various individual decision-tree classifiers. They create various trees and then assign the class predicted by the majority of trees (Breiman 2001). We can change the number of trees to get better results. Generally, more the number of trees, better are the results. However, excessive number of trees can also lead to the problem of overfitting (Ali et al. 2012). Another important hyperparameter that impacts a random forest classifier is the measure that is used to split the trees. There are two commonly approaches. The first is Gini impurity and the second is Entropy. The Gini impurity is a measure of probability of mislabelling element in a node if they are labelled randomly. Entropy measures contrast between a dimension and the target. The dimension with least entropy is chosen for split (Laber and Marco 2018).

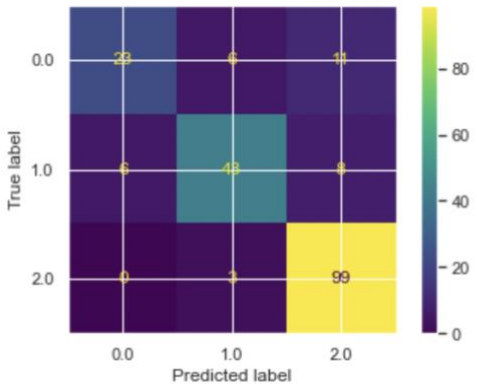
   We can train models for all the combination of hyperparameters and select the model with highest accuracy. However, making decision on the basis of one-set of data is not enough. Therefore, this project uses stratified k-fold cross-validation technique. Stratified k-fold technique divides data into k parts with each part having similar class distribution as the entire data frame. This project uses 5-fold stratified cross-validation. So it means, that we train models on four of these five partitions and test it on the fifth partition. Next time, we leave a different partition for testing and train models on the remaining four partitions (Raschka 2018). We repeat this process five times and then we take average of accuracy. We select the model with highest accuracy.

   The figure below shows average accuracy for different combinations of two hyperparameters (Number of trees and measure used for split).



It is evident that model with 200 trees and entropy criteria as measure of split has the highest average accuracy. Therefore, we use this model to check against the 20% testing dataset that we kept aside during test-train split. The image below shows diagnostic figures for the model.

	precision	recall	f1-score	support
0.0	0.79	0.57	0.67	40
1.0	0.83	0.75	0.79	57
2.0	0.84	0.97	0.90	102
accuracy			0.83	199
macro avg	0.82	0.77	0.79	199
weighted avg	0.83	0.83	0.82	199



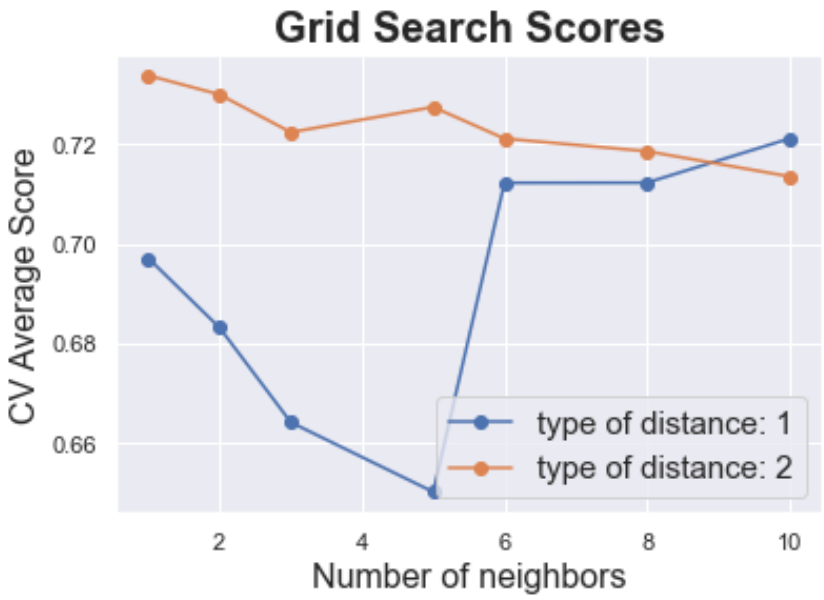
Precision tells the probability of an assigned label to be true for a model. We can see that precision for all three labels is above 75%. If a data point is assigned label 0, then we can say with 79% probability that that data point actually belongs to label 0.

Recall is the measure of probability that data points of a particular label will be assigned to that label. Recall for label 2 is 97%. It means that the model was able to identify 97% of data points with label 2 to be from label 2. However, recall for other two labels is quite low. It shows that data imbalance is affecting our model.

However, we have reached the accuracy above 75%. So, we will not take any additional action to correct this. Moreover, there is no mention anywhere that recall of minority classes is an important thing that should be kept in mind.

2. k-NN Classifier – k Nearest Neighbour is a distance based classification technique. It finds out distance of new data points from the existing data points and assign them class label based on the majority class of its k nearest neighbours. The most basic hyperparameter for this classifier is number of neighbours. Another important hyperparameter is the type of distance measure that is used. We can use Manhattan distance or the Euclidian distance. We can also use other measures of distance.

   We use the same 5-fold stratified cross-validation strategy for hyperparameter tuning. The graph below shows how models perform for various sets of hyperparameters.



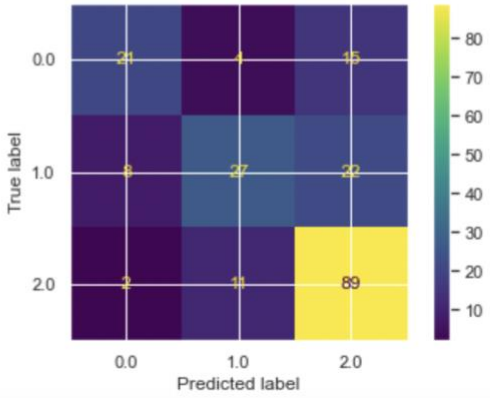
Model seems to be performing good for k = 1, k = 2, and k = 5. We choose k = 5 because using k = 1 is simply assigning the label of the nearest data instance. Also, there is not any scope of voting with k = 2. We also choose Euclidian distance as the way of identifying neighbours. Image below show diagnostics of the model –

```
Accuracy: 0.6884422110552764
              precision    recall  f1-score   support

         0.0       0.68      0.53      0.59        40
         1.0       0.64      0.47      0.55        57
         2.0       0.71      0.87      0.78       102

    accuracy                           0.69       199
   macro avg       0.68      0.62      0.64       199
weighted avg       0.68      0.69      0.68       199
```

```
Out[71]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f
```

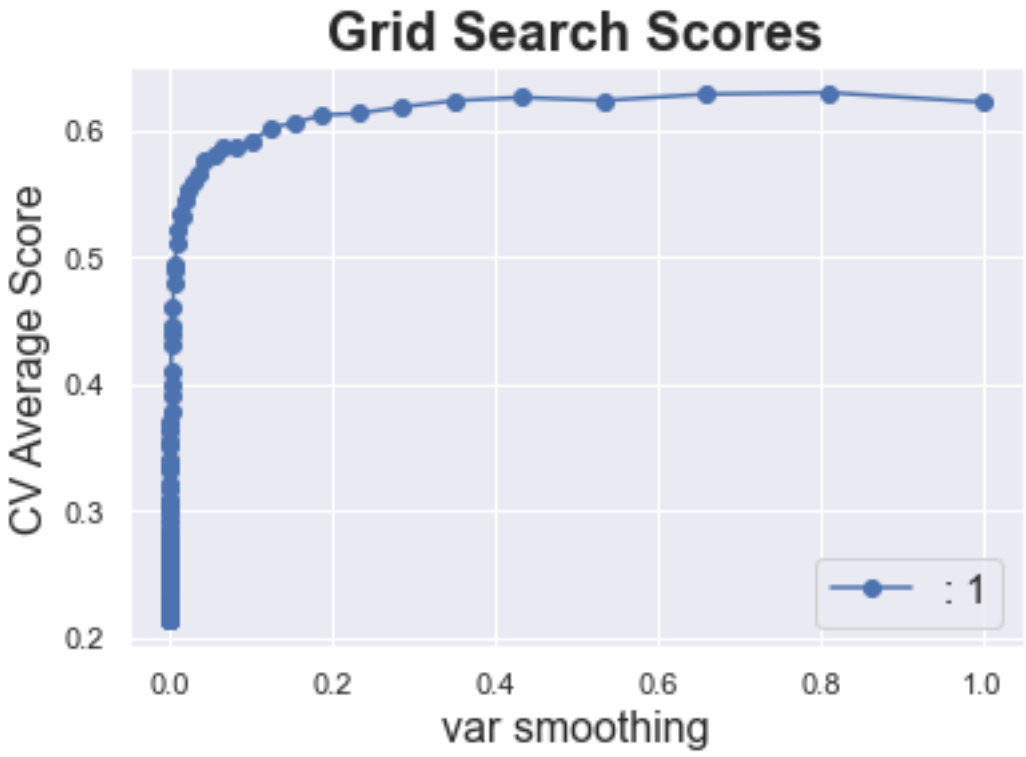


Here, we can see that precision for all the labels is less than the precision given by random forest model. Recall has also decreased drastically. It means that k-NN is affected by class imbalance more than random forest classifier. It could be due to the reason that due to high frequency of label: 2, most of the testing data points may end up having data points with label: 2 as neighbours.

3. Naïve-Bayes Classifier – We use a gaussian naïve-bayes classifier for this project. A gaussian naïve-bayes classifier assumes that that various dimensions of the data are distributed normally. Therefore, this classifier is not suitable if the data set has categorical variables (Dobilas 2021). Our data set contains three categorical variables. However, we would still train a gaussian naïve-bayes model without dropping one-hot encoded columns (taking the easiest path first).

   The sklearn implementation of Gaussian Naïve-Bayes has two hyperparameters. The first is prior probabilities. We can define the starting probabilities for the model using

this hyperparameter. The second one is variance smoothing. Scikit-learn 1.0 defines it as ‘Portion of the largest variance of all features that is added to variances for calculation stability’. The graph below shows performance of various models with different variance smoothing.



The model performance is at best when variance smoothing is at 0.8111.

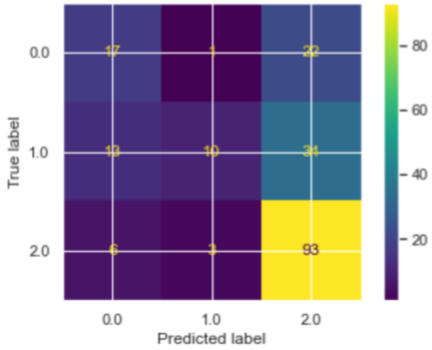
The diagnostics for naïve-bayes model with variance smoothing = 0.8111 are shown in image below –

```
Accuracy: 0.6030150753768844
              precision    recall  f1-score   support

         0.0       0.47      0.42      0.45        40
         1.0       0.71      0.18      0.28        57
         2.0       0.62      0.91      0.74       102

    accuracy                           0.60       199
   macro avg       0.60      0.50      0.49       199
weighted avg       0.62      0.60      0.55       199
```

```
Out[72]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f8d0005cca0>
```



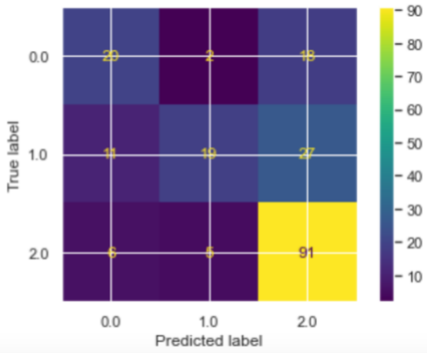
The model has a bad recall for label: 2. It is only able to identify 18% of all the data with label 2. The precision of label 2 has also decreased. It means that it is misclassifying various instances with label 0 and label 1 as label 2. We can change prior probabilities to increase recall for label: 1. After setting prior probabilities as [0.2, 0.45 and 0.35], we get the following results –

```
Accuracy: 0.6532663316582915
              precision    recall  f1-score   support

         0.0       0.54      0.50      0.52        40
         1.0       0.73      0.33      0.46        57
         2.0       0.67      0.89      0.76       102

    accuracy                           0.65       199
   macro avg       0.65      0.58      0.58       199
weighted avg       0.66      0.65      0.63       199
```

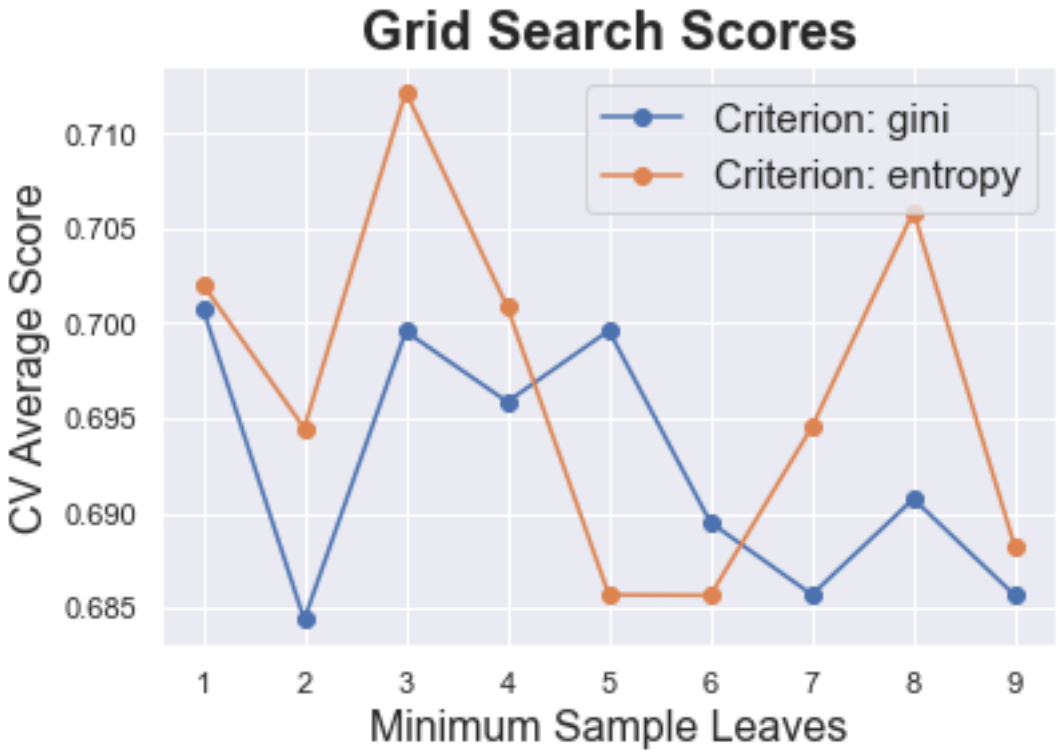
```
Out[87]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f8d419ca970>
```



We can see that the recall for label: 1 has increased. Accuracy also increased from around 60% to 65%.

4. Decision-Tree Classifier – We already trained a random forest above. A decision tree is one single unit out of 200 units that we used to form our random forest. We use similar stratified 5-fold cross validation scheme for hyperparameter tuning. The hyperparameters that we tune for this model are, measure that is used to split the tree (entropy and Gini), and the number of minimum sample leaves. The description of splitting criterion is mentioned in the random forest section. The second hyperparameter defines the minimum number of samples that would be needed to be considered a node. A tree is made up of nodes. So this hyperparameter would define when the model gets to create a new node (Mithrakumar 2019).

   The graph below shows performance of models for a varying range of these two parameters –



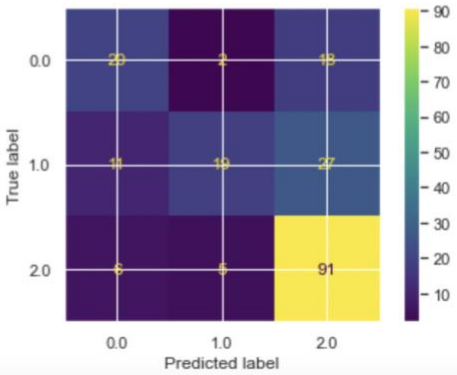
   Highest accuracy is achieved when minimum number of leaves is 3 and entropy criteria is used. The picture below shows performance of this model on the testing data that we set aside during test train split –

```
Accuracy: 0.7185929648241206
              precision    recall  f1-score   support

         0.0       0.50      0.50      0.50        40
         1.0       0.68      0.70      0.69        57
         2.0       0.83      0.81      0.82       102

    accuracy                           0.72       199
   macro avg       0.67      0.67      0.67       199
weighted avg       0.72      0.72      0.72       199
```

Out[128]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f8d418ebfa0>



The accuracy of the model is 71.85% on the testing data. It has better recall for label 1 and label 2 than naïve bayes classifier. However, its precision of label 1 and label 2 are lesser than that of naïve-bayes classifier.

Conclusion

The business requirement (75% accuracy) was achieved based on 'keep it simple' approach using random forest classifier. Accuracy and f1 score of all the four models on the testing data is given in the table below –

Model	Accuracy	Weighted average of f1 score
Random Forest	0.834	0.83
Decision Tree	0.718	0.72
k-NN	0.688	0.69
Gaussian Naïve-Bayes	0.653	0.63

Therefore, the best model as per the business goal is random forest model.

ACCURACY ESTIMATE FOR THE MODEL IS = 82.07% (based on the average accuracy of five models trained during cross validation).

Random forest and Decision Tree models were used to predict labels of the 200 data points with missing labels. The labels predicted by them are in 73.5% agreement with each other. The labels are given in a table in appendix 2.

Scope of Improvement (Future Works)

1. Check if the missing values of Att09 are missing at random or not. Based on that, use a suitable imputation strategy.
2. Perform oversampling of minor classes so that they get a better precision and recall.
3. Only use normally distributed dimensions in gaussian-naïve bayes. We have used one-hot encoding. So a separate Bernoulli naïve-bayes model can be trained for these dimensions and we can use weighted average of both models to do better predictions.
4. Try tuning other hyperparameters of random forest classifier (maximum depth, minimum sample leaves, maximum features).
5. Further investigate dimensions with uniformly distributed data.

References

Ali, Jehad, Rehanullah Khan, Nasir Ahmad, and Imran Maqsood. "Random forests and decision trees." *International Journal of Computer Science Issues (IJCSI)* 9, no. 5 (2012): 272.

Breiman, Leo. "Random forests." *Machine learning* 45, no. 1 (2001): 5-32.

Dobilas, Saul. 2021. “Naive Bayes Classifier - How to Successfully Use It in Python?” Medium. Towards Data Science. August 22, 2021. https://towardsdatascience.com/naive-bayes-classifier-how-to-successfully-use-it-in-python-ecf76a995069.

Laber, Eduardo, Marco Molinaro, and Felipe Mello Pereira. "Binary partitions with approximate minimum impurity." In *International Conference on Machine Learning*, pp. 2854-2862. PMLR, 2018.

Mithrakumar, Mukesh. 2019. “How to Tune a Decision Tree?” Medium. Towards Data Science. November 12, 2019. https://towardsdatascience.com/how-to-tune-a-decision-tree-f03721801680.

Raschka, Sebastian. "Model evaluation, model selection, and algorithm selection in machine learning." *arXiv preprint arXiv:1811.12808* (2018).

Toloşi, Laura, and Thomas Lengauer. "Classification with correlated features: unreliability of feature ranking and solutions." *Bioinformatics* 27, no. 14 (2011): 1986-1994.

Tom. 2018. “How Does Knn Handle Categorical Features.” Data Science Stack Exchange. April 1, 2018. https://datascience.stackexchange.com/questions/26713/how-does-knn-handle-categorical-features.

Sarracino, Francesco, and Malgorzata Mikucka. "Estimation bias due to duplicated observations: a Monte Carlo simulation." (2016).

Seger, Cedric. "An investigation of categorical variable encoding techniques in machine learning: binary versus one-hot and feature hashing." (2018).

Sharma, Pulkit. 2019. “Why Is Scaling Required in KNN and K-Means?” Medium. Analytics Vidhya. August 5, 2019. https://medium.com/analytics-vidhya/why-is-scaling-required-in-knn-and-k-means-8129e4d88ed7.

Appendices – 1