Assignment 2

CMSC 691 — Intoduction to Data Science

Faisal Rasheed Khan

VB02734

vb02734@umbc.edu

Question 1--

1.(Assign2_Question1.py)

a. I have performed Exploratory data analysis for the given data set lending.csv
The .describe() function gives all the statistics for the columns present in lending.csv which is mean, standard deviation, Inter-Quartile Range.

- On seeing the dependent variable loan_default, its mean is closer to 0 (and also visualized in histogram) and it has values 0 and 1, where we can conclude that the data is class baised and has more values of 0, which is a bad sign if we want to train a model. So we need to balance our data first, this can be done via sampling and sampling depends on us how we do. I have done using undersampling where it considers number of rows of data of the minimum class labels, which is here 0 for loan_default.
  0 – 11086
  1-77365
  I have selected random 11086 samples of 1 label for loan_default.
- We see dependency of the other variables to the dependent variable via correlation. Remaining column fields are not correlated to dependent variable loan_default, this we conclude with the help of our code performing .corr() on the dataset. Only loan_amt is correlated with pct_loan_income.
  If one variable is correlated to other, then we can consider any one variable.
- Also preprocessed the data for example checked for any missing data, outliers, or any null data (showed in the code) but there are no any such cases.
- So with the help of equation of probability p(y = label | x= input data), we build the model. Python library has both logistic and naïve bayes model inbuilt function
  Here y= loan_default and x = input lending.csv file without loan_default column.
  With the help of this information we proceed further to train the model.

b.

- We will see the class bias of the dependent variable when we train the model without sampling and it fares very badly, it doesn't learn any thing but its accuracy will be high and the metrics precision and recall are also not that much efficient when the class is biased over one label.
  Accuracy without sampling is 87%
  Because precision is True positive/ (sum of (true positive + false positive))

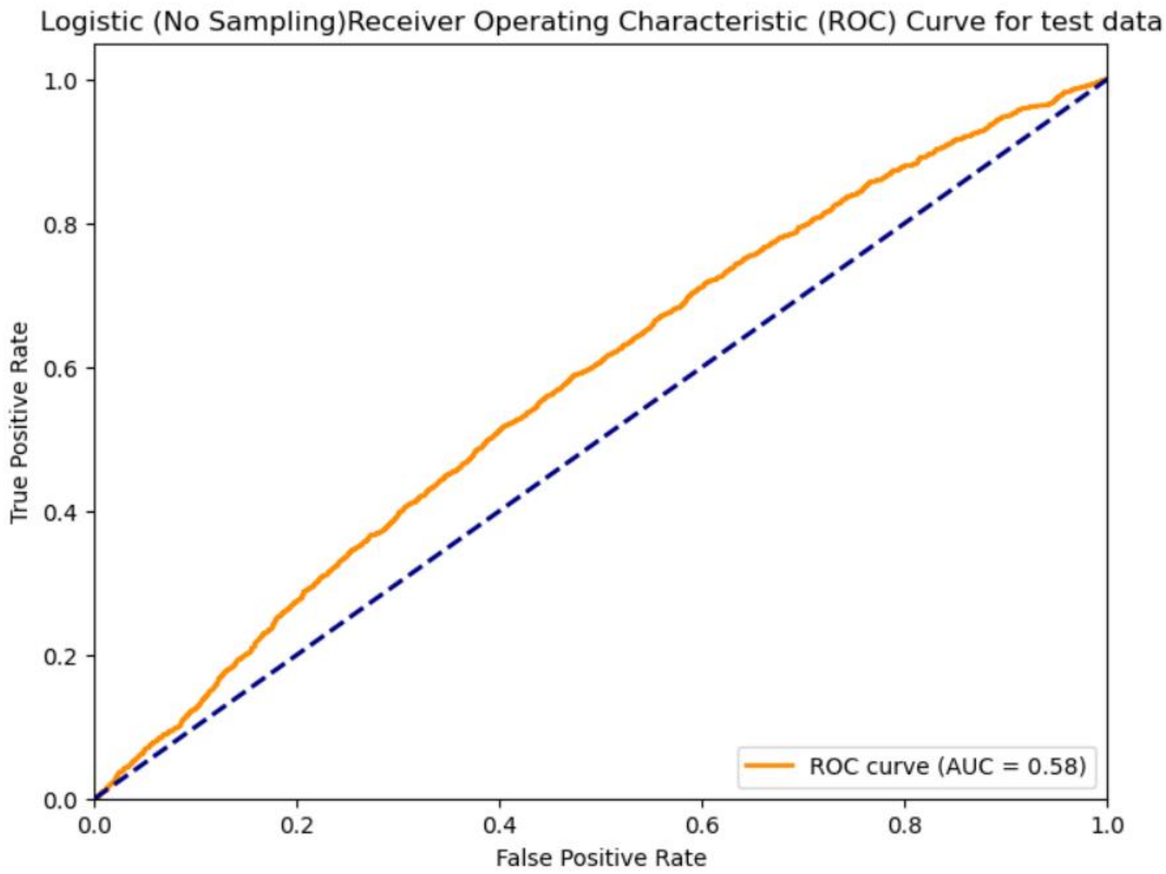
recall is True positive/ (sum of (true positive + false Negative)), if the class is biased then precision and recall will be high, for example TP = 100, FP=10, TN =100, FN =10, here the model is overfit and accuracy, precision, recall is high . but this model fails to perform good for unseen data(refer the results I have done in the code). So the other metrics use comes into the picture where they consider other terms also if the class is biased(other metrics: f1 score)

- The model to perform good, we balance the data using sampling, such that the model is not biased to one class.
  I have divided the data into 3: Train data, Development/validation data, Test data.
  First we will train the model on train data and validation data is seen to improve the model a bit and we use test data to see how model performs.
  Then we build the models Logistic regression and Naïve Bayes. There is inbuilt library in python which does this.
  And the evaluation metrics is also there in scikit library

c.

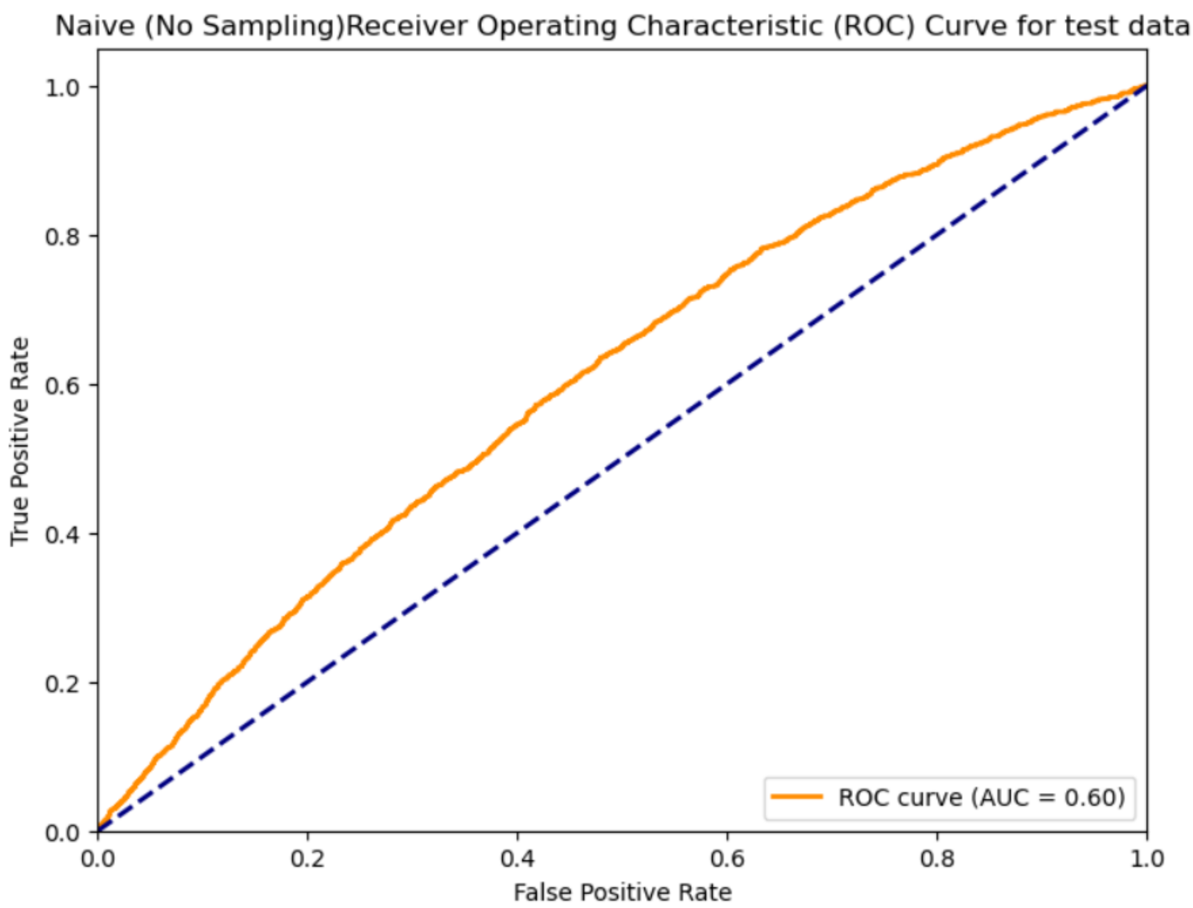
Logistic Regression(Without Sampling):

By using all the data(i.e. without Sampling), its Accuracy is 87%



Naïve Bayes(Without Sampling):

By using all the data(i.e. without Sampling), its Accuracy is 87%



Takeaways:
Due to biased class model, both the accuracy is almost similar because the model is biased to one label

<u>Now we will be discussing the results after we sample the data:</u>

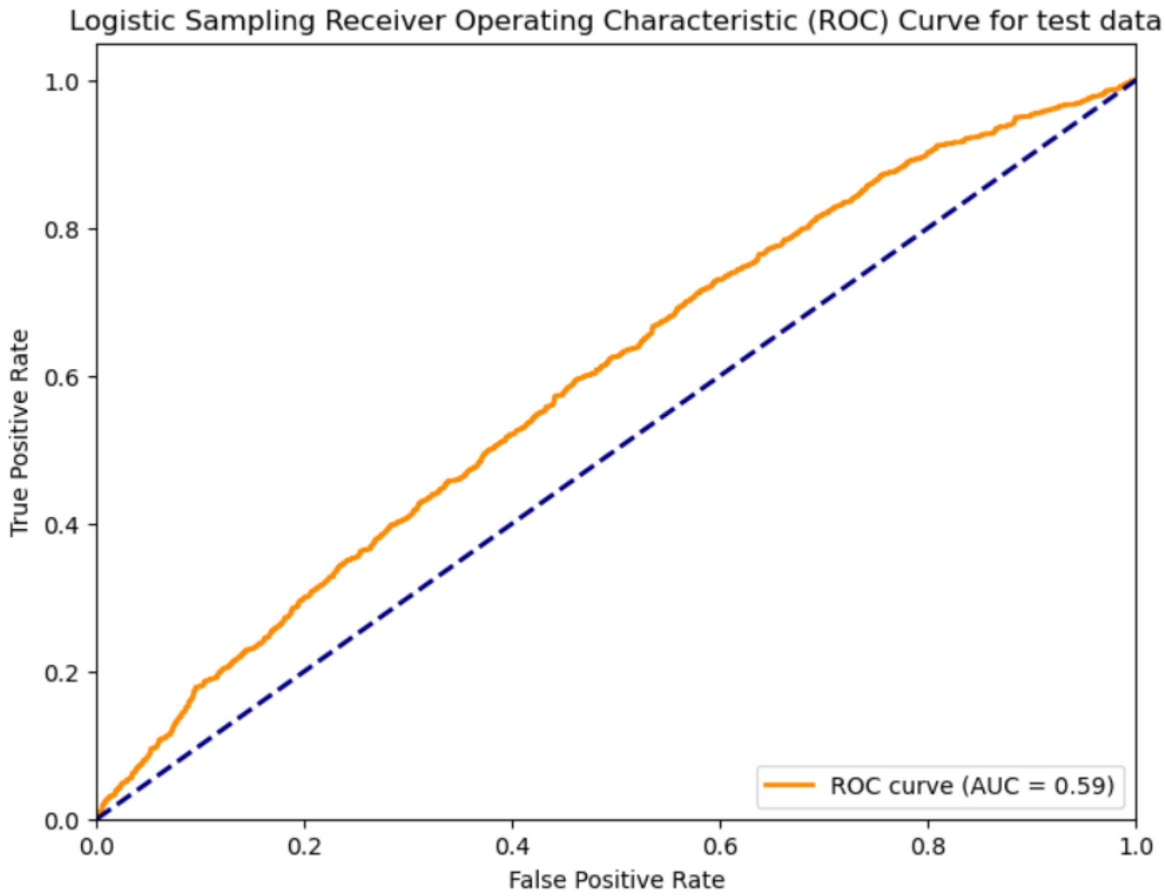
Logistic Regression(With Sampling):

```
Dev Accuracy: 0.5706554419723392
Dev F1 Score: 0.5963821368004523
Dev Precision: 0.5605738575982997
Dev Recall: 0.6370772946859904
Dev Confusion Matrix:
 [[ 843  827]
 [ 601 1055]]
Classification Report:
              precision    recall  f1-score   support

           0       0.58      0.50      0.54      1670
           1       0.56      0.64      0.60      1656

    accuracy                           0.57      3326
   macro avg       0.57      0.57      0.57      3326
weighted avg       0.57      0.57      0.57      3326

Test Set Accuracy: 0.5601322910402886
Test Set F1 Score: 0.5909980430528377
Test Set Precision: 0.551958224543081
Test Set Recall: 0.6359807460890493
Test Set Confusion Matrix:
 [[ 806  858]
 [ 605 1057]]
```



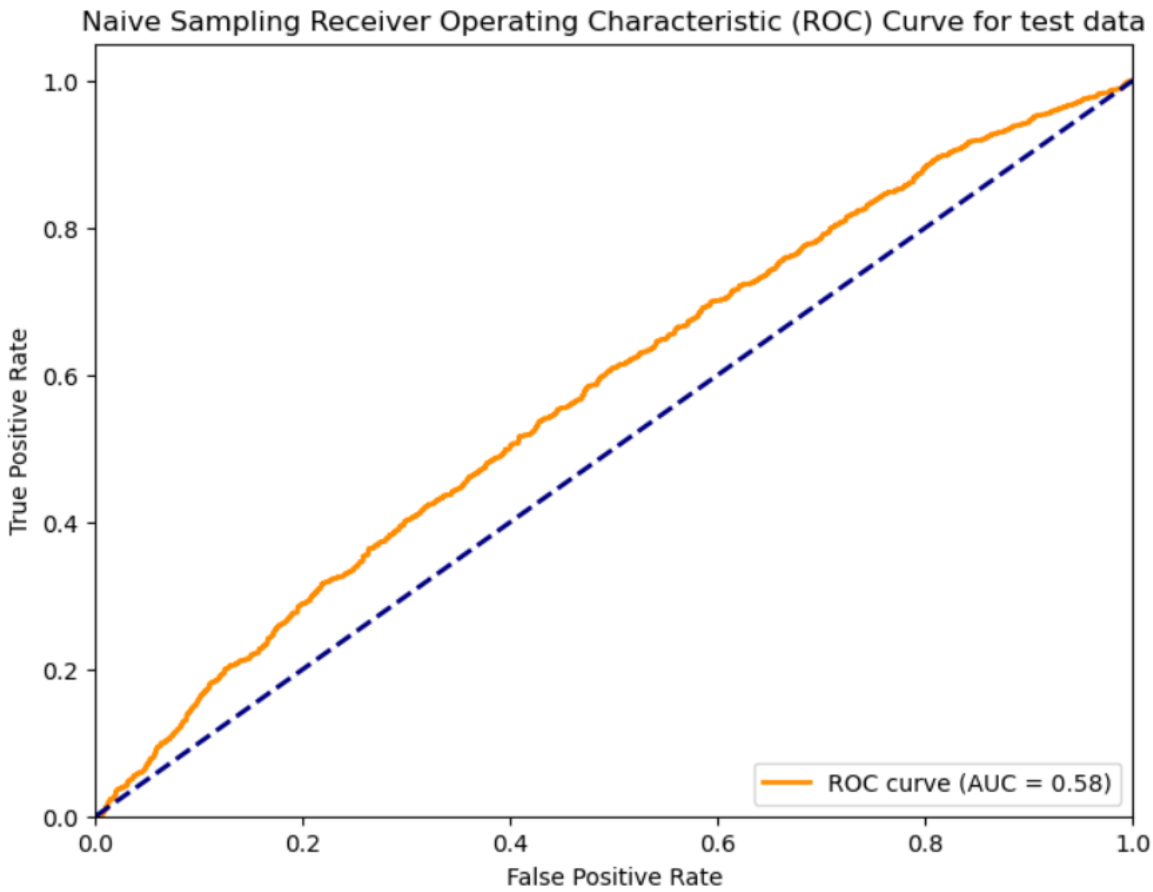
Naïve Bayes(With Sampling):

```
Dev Accuracy: 0.539386650631389
Dev F1 Score: 0.6600088770528185
Dev Precision: 0.5217543859649123
Dev Recall: 0.8979468599033816
Dev Confusion Matrix:
 [[ 307 1363]
 [ 169 1487]]
Classification Report:
               precision    recall  f1-score   support

           0       0.64      0.18      0.29      1670
           1       0.52      0.90      0.66      1656

    accuracy                           0.54      3326
   macro avg       0.58      0.54      0.47      3326
weighted avg       0.58      0.54      0.47      3326

Test Set Accuracy: 0.5408899579073962
Test Set F1 Score: 0.6593798795449476
Test Set Precision: 0.5239276852180078
Test Set Recall: 0.8892900120336944
Test Set Confusion Matrix:
 [[ 321 1343]
 [ 184 1478]]
```



Model Comparison:

1. Accuracy: In terms of accuracy, the Logistic Regression model performs slightly better on both the development and test datasets. It has a dev accuracy of 0.5707 compared to Naive Bayes' 0.5394. In the test set, Logistic Regression also has a higher accuracy (0.5601) compared to Naive Bayes (0.5409).

2. F1 Score: The F1 score is a measure that considers both precision and recall. The Naive Bayes model has a higher F1 score on the development dataset (0.6600) compared to Logistic Regression (0.5964). However, on the test set, their F1 scores are quite similar, with Naive Bayes at 0.6594 and Logistic Regression at 0.5910.

3. Precision and Recall: The Naive Bayes model has a higher recall (sensitivity) on both the development and test datasets, indicating its ability to correctly identify positive cases. However, it sacrifices precision, resulting in more false positives. Logistic Regression maintains a better balance between precision and recall.

Which Model to prefer?
Logistic Regression:
Logistic Regression is a linear model that learns to predict the probability (0 or 1)
It learns the relationship between input features and the binary outcome by finding the optimal coefficients for each feature.
It's a discriminative model that directly models the conditional probability of the output given the input.

Naive Bayes:
Naive Bayes is a probabilistic model that uses Bayes' theorem to predict the probability of a certain event based on prior knowledge.
It's a generative model that models the joint probability of input features and the output.
Model Choice:
The choice between Logistic Regression and Naive Bayes depends on the specific characteristics of your problem and the trade-off between precision and recall. In this case:
If you prioritize high recall (TP, even if it means more false positives), the Naive Bayes model is better.
If you prefer a balanced approach with higher accuracy and a reasonable F1 score, the Logistic Regression model is a better choice.
Consider the specific requirements of your application and whether false positives or false negatives are more critical when deciding which model to use.

Question 2--

2.(Association_Rules_Bread_Basket. R)

Using Breadbasket dataset, find out the descriptive statistics about transactions and list all the distinct items in the dataset. Do a market basket analysis and show the association rules. Use a suitable metric to sort the rules. Explain why did you choose that metric? Next, among the top five rules, choose your most favorite rule and describe it and explain what information it provides you.

- The descriptive statistics has been done for the BreadBasket_DMS.csv like Mean, Median, Mode, InterQuartile Range, Variance, Standard Deviation.

```
>
> #Descriptive statistics analysis of our data BreadBasket_DMS.csv
>
> summary(arules_data$Transaction)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      1    2548    5067    4952    7329    9684
> mean(arules_data$Transaction)
[1] 4951.991
> median(arules_data$Transaction)
[1] 5067
> mode=function(a){
+   uni=unique(a)
+   tab= tabulate(match(a,uni))
+   uni[tab==max(tab)]
+ }
> mode(arules_data$Transaction)
[1] 6474
> quantile(arules_data$Transaction)
  0%  25%  50%  75% 100%
   1 2548 5067 7329 9684
> var(arules_data$Transaction)
[1] 7771597
> sd(arules_data$Transaction)
[1] 2787.758
```

The unique rules have also been listed with the function provided by R.

```
R 4.3.1 · ~/
> unique(arules_data$Item)
 [1] "Bread"                          "Scandinavian"
 [3] "Hot chocolate"                  "Jam"
 [5] "Cookies"                        "Muffin"
 [7] "Coffee"                         "Pastry"
 [9] "Medialuna"                      "Tea"
[11] "NONE"                           "Tartine"
[13] "Basket"                         "Mineral water"
[15] "Farm House"                     "Fudge"
[17] "Juice"                          "Ella's Kitchen Pouches"
[19] "Victorian Sponge"               "Frittata"
[21] "Hearty & Seasonal"              "Soup"
[23] "Pick and Mix Bowls"             "Smoothies"
[25] "Cake"                           "Mighty Protein"
[27] "Chicken sand"                   "Coke"
[29] "My-5 Fruit Shoot"               "Focaccia"
[31] "Sandwich"                       "Alfajores"
[33] "Eggs"                           "Brownie"
[35] "Dulce de Leche"                 "Honey"
[37] "The BART"                       "Granola"
[39] "Fairy Doors"                    "Empanadas"
[41] "Keeping It Local"               "Art Tray"
[43] "Bowl Nic Pitt"                  "Bread Pudding"
[45] "Adjustment"                     "Truffles"
[47] "Chimichurri Oil"                "Bacon"
[49] "Spread"                         "Kids biscuit"
[51] "Siblings"                       "Caramel bites"
[53] "Jammie Dodgers"                 "Tiffin"
[55] "Olum & polenta"                 "Polenta"
[57] "The Nomad"                      "Hack the stack"
[59] "Bakewell"                       "Lemon and coconut"
[61] "Toast"                          "Scone"
[63] "Crepes"                         "Vegan mincepie"
[65] "Bare Popcorn"                   "Muesli"
[67] "Crisps"                         "Pintxos"
[69] "Gingerbread syrup"              "Panatone"
[71] "Brioche and salami"             "Afternoon with the baker"
[73] "Salad"                          "Chicken Stew"
[75] "Spanish Brunch"                 "Raspberry shortbread sandwich"
[77] "Extra Salami or Feta"           "Duck egg"
[79] "Baguette"                       "Valentine's card"
[81] "Tshirt"                         "Vegan Feast"
[83] "Postcard"                       "Nomad bag"
[85] "Chocolates"                     "Coffee granules"
[87] "Drinking chocolate spoons"      "Christmas common"
[89] "Argentina Night"                "Half slice Monster"
[91] "Gift voucher"                   "Cherry me Dried fruit"
[93] "Mortimer"                       "Raw bars"
[95] "Tacos/Fajita"
```

- Association Rules if we keep support = 0.01 and confidence = 0.5, as coffee frequency is higher then buying the items will lead to buy coffee. And we don't get much information from that, as we can also easily guess that because coffee frequency is higher.
- So, we need to adjust the support and confidence such that appropriate rules/findings are there.

<u>Below are the association rules for support = 0.01 and confidence = 0.5:</u>

```
> #sorting the rules by using lift metric
> inspect(sort(rules_no_coffee_no_none,decreasing = TRUE,by="lift"))
     lhs                 rhs      support    confidence coverage   lift     count
[1]  {Toast}          => {Coffee} 0.02349979 0.7044025  0.03336131 1.483510 224
[2]  {Spanish Brunch} => {Coffee} 0.01080571 0.5988372  0.01804448 1.261183 103
[3]  {Medialuna}      => {Coffee} 0.03483005 0.5684932  0.06126731 1.197277 332
[4]  {Pastry}         => {Coffee} 0.04710449 0.5515971  0.08539656 1.161693 449
[5]  {Alfajores}      => {Coffee} 0.01951322 0.5406977  0.03608896 1.138738 186
[6]  {Juice}          => {Coffee} 0.02045741 0.5357143  0.03818716 1.128243 195
[7]  {Sandwich}       => {Coffee} 0.03797734 0.5323529  0.07133865 1.121164 362
[8]  {Cake}           => {Coffee} 0.05434326 0.5269583  0.10312631 1.109803 518
[9]  {Scone}          => {Coffee} 0.01793957 0.5229358  0.03430550 1.101331 171
[10] {Cookies}        => {Coffee} 0.02801091 0.5194553  0.05392363 1.094001 267
[11] {Hot chocolate}  => {Coffee} 0.02937474 0.5072464  0.05791020 1.068288 280
```

- That's the reason why we remove coffee value from the right side. And also we can not buy any item called "None", so it is also of no use. We should remove None such that we may not have/infer appropriate rules if we keep None as it could be misleading.
- We use lift metric to sort the rules as it is more popular with apriori algorithm.
- The appropriate adjustment after substituting many values for support and confidence for me were
- support = 0.001 and confidence = 0.001

```
> inspect(sort(rules_no_coffee_no_none,decreasing = TRUE,by="lift"))
     lhs                             rhs                    support     confidence
[1]  {Coffee, Salad}              => {Extra Salami or Feta} 0.001468737 0.22580645
[2]  {Coffee, Extra Salami or Feta} => {Salad}              0.001468737 0.45161290
[3]  {Extra Salami or Feta}       => {Salad}                0.001678556 0.42105263
[4]  {Salad}                      => {Extra Salami or Feta} 0.001678556 0.16161616
[5]  {Fudge}                      => {Jam}                  0.002517835 0.16901408
[6]  {Jam}                        => {Fudge}                0.002517835 0.17021277
[7]  {Alfajores, Cookies}         => {Juice}                0.001049098 0.43478261
[8]  {Juice, Sandwich}            => {Coke}                 0.001049098 0.18181818
[9]  {Salad}                      => {Spanish Brunch}       0.001258917 0.12121212
[10] {Spanish Brunch}             => {Salad}                0.001258917 0.06976744
[11] {Coke, Juice}                => {Sandwich}             0.001049098 0.47619048
[12] {Alfajores, Juice}           => {Cookies}              0.001049098 0.34482759
[13] {Jammie Dodgers}             => {Tiffin}               0.001154008 0.08800000
[14] {Tiffin}                     => {Jammie Dodgers}       0.001154008 0.07534247
[15] {Coffee, Jammie Dodgers}     => {Juice}                0.001363827 0.20634921
[16] {Coffee, Juice}              => {Spanish Brunch}       0.001993286 0.09743590
[17] {Coke, Sandwich}             => {Juice}                0.001049098 0.20408163
[18] {Bread, Cake}                => {Jammie Dodgers}       0.001573647 0.06787330
[19] {Coffee, Juice}              => {Jammie Dodgers}       0.001363827 0.06666667
[20] {Chicken Stew}               => {Spanish Brunch}       0.001154008 0.08943089
[21] {Spanish Brunch}             => {Chicken Stew}         0.001154008 0.06395349
[22] {Cake, Sandwich}             => {Soup}                 0.001154008 0.16923077
[23] {Truffles}                   => {Spanish Brunch}       0.001783466 0.08854167
[24] {Spanish Brunch}             => {Truffles}             0.001783466 0.09883721
[25] {Cookies, Juice}             => {Alfajores}            0.001049098 0.17543860
[26] {Coffee, Spanish Brunch}     => {Juice}                0.001993286 0.18446602
[27] {Coffee, Coke}               => {Juice}                0.001154008 0.18032787
[28] {Coffee, Coke}               => {Sandwich}             0.001993286 0.31147541
[29] {Mineral water}              => {Coke}                 0.001154008 0.08208955
[30] {Coke}                       => {Mineral water}        0.001154008 0.05978261
```

The above rules respective coverage, lift, count are here below:

	coverage	lift	count
[1]	0.006504406	56.641766	14
[2]	0.003252203	43.482568	14
[3]	0.003986572	40.540138	16
[4]	0.010386068	40.540138	16
[5]	0.014897188	11.425832	24
[6]	0.014792279	11.425832	24
[7]	0.002412925	11.385571	10
[8]	0.005770038	9.418972	10
[9]	0.010386068	6.717407	12
[10]	0.018044482	6.717407	12
[11]	0.002203105	6.675070	10
[12]	0.003042384	6.394740	10
[13]	0.013113722	5.745315	11
[14]	0.015316828	5.745315	11
[15]	0.006609316	5.403628	13
[16]	0.020457407	5.399761	19
[17]	0.005140579	5.344248	10
[18]	0.023185061	5.175747	15
[19]	0.020457407	5.083733	13
[20]	0.012903903	4.956135	11
[21]	0.018044482	4.956135	11
[22]	0.006819136	4.948183	11
[23]	0.020142677	4.906856	17
[24]	0.018044482	4.906856	17
[25]	0.005979857	4.861281	10
[26]	0.010805707	4.830577	19
[27]	0.006399496	4.722212	11
[28]	0.006399496	4.366152	19
[29]	0.014057910	4.252596	11
[30]	0.019303399	4.252596	11

Association: X -> Y

Support (s):

Fraction of transactions that contain itemset

It is the fraction of transactions in the dataset that contain both the LHS and RHS items.

A high support value implies that the rule is more frequently applicable.

Support = $\frac{\text{No. of transactions that contain both } \{\mathbf{X}, \mathbf{Y}\} \textit{itemset}}{\text{Total number of transactions}}$

Confidence (c):

Measures how often items in Y appear in transactions that contain X i.e. the strength of the rule.

It is the probability of finding the RHS items in a transaction given that the LHS items are present.

A higher confidence value means that the rule is more reliable.

Confidence = $\frac{\text{No. of transactions that contain both } \{\mathbf{X}, \mathbf{Y}\}}{\text{No. of transactions that contain X}}$

Lift:

Lift is a metric widely used for apriori algorithm, to measure of how much more likely it is to find the RHS items in a transaction when the LHS items are present compared to when they are not.

Higher value of lift means the items are more correlated

$$\boldsymbol{lift} = \frac{\textbf{Rule Confidence of } (\mathbf{A} \rightarrow \mathbf{B})}{\textbf{Prior proportion of the consequent B}}$$

Why lift metric to sort?

- As we know that lift value measures how 2 items are correlated, in identifying associations that are not independent then lift is a good choice.
- Here our objective is to find the associated items, i.e. highly correlated items that's why we have used lift
- We just doesn't need high reliable and high frequent occurrence like what the support and confidence does, we need more correlation of the items to have good and nicer association rules.

Which rule is my most favorite?

```
   lhs                     rhs                    support     confidence
[1]{Coffee, Salad} => {Extra Salami or Feta} 0.001468737 0.22580645
```

- The above is the first rule, it is selected because of the high lift value, which means that the lhs and rhs are more correlated.
- But for me that rhs item Extra Salami or Feta doesn't seems to be effective while we consider the business perspective
- If we are considering this association(according to business's perspective) then rule 1 to rule 6 are not the items what we will be looking for to order based on the association.
- The good rules I would say are rule 7/ rule 8 because this gives information of what we need to stock the items

```
[7] {Alfajores, Cookies} => {Juice}          0.001049098 0.43478261
[8]   {Juice, Sandwich}  => {Coke}            0.001049098 0.18181818
```

However, lift value is not high but we will look these types of associations for the business needs.

References: scikit-learn: machine learning in Python — scikit-learn 1.3.1 documentation

sklearn.linear_model.LogisticRegression — scikit-learn 1.3.1 documentation

1.9. Naive Bayes — scikit-learn 1.3.1 documentation

Welcome to Python.org

https://blackboard.umbc.edu/bbcswebdav/pid-6351621-dt-content-rid-70354645_1/xid-70354645_1

https://blackboard.umbc.edu/bbcswebdav/pid-6362730-dt-content-rid-70461367_1/xid-70461367_1

https://blackboard.umbc.edu/bbcswebdav/pid-6371899-dt-content-rid-70683694_1/xid-70683694_1

https://blackboard.umbc.edu/bbcswebdav/pid-6371900-dt-content-rid-70683695_1/xid-70683695_1

Home - RDocumentation

BINOMIAL distribution in R [dbinom, pbinom, qbinom and rbinom functions] (r-coder.com)

https://blackboard.umbc.edu/bbcswebdav/pid-6367033-dt-content-rid-70554826_1/xid-70554826_1

https://blackboard.umbc.edu/bbcswebdav/pid-6367033-dt-content-rid-70554827_1/xid-70554827_1

https://blackboard.umbc.edu/bbcswebdav/pid-6367033-dt-content-rid-70554828_1/xid-70554828_1

https://blackboard.umbc.edu/bbcswebdav/pid-6367033-dt-content-rid-70554829_1/xid-70554829_1

https://blackboard.umbc.edu/bbcswebdav/pid-6367033-dt-content-rid-70554830_1/xid-70554830_1

An Introduction to Statistical Learning (statlearning.com)

https://blackboard.umbc.edu/bbcswebdav/pid-6377459-dt-content-rid-70951694_1/xid-70951694_1

https://blackboard.umbc.edu/bbcswebdav/pid-6377461-dt-content-rid-70951698_1/xid-70951698_1

https://blackboard.umbc.edu/bbcswebdav/pid-6380392-dt-content-rid-71127366_1/xid-71127366_1

Welcome | R for Data Science (had.co.nz)