COMP3009 Data Mining

Assignment

Polawat Srichana 20696984

Abstract-This assignment is reported as a part of COMP3009: Data Mining course. This assignment is produced to apply the knowledge from Data Mining to handle the real-world data and predict to obtain the best accuracy for classification

Introduction

The specified dataset is a csv file named 'data2022.student.csv'. From analysis of the data, the data has many problems and need to clean it properly but nowadays, Python have many libraries to handle this problem. For instance, data imbalance with imbalanced-learn. After cleaning we need to do classification, in this project we need to define the best two model that obtain the best accuracy for prediction which classifier that we decide to use are as follow; K-nearest neighbor, Random Forest, Decision Tree, Gradient Boosting, Naïve Bayes. After finishing the assignment, the knowledge which we can gain in this assignment can be apply for Data Scientist to deal with the real life data and real world classifier.

	ID	Class	C1	C2	C3	C4	C5	C6	C7	C8	...	C23	C24	C25	C26	C27	C28	C29	C30	C31	C32
0	1	0.0	V1	V3	59.0	1	27	V3	V3	V5	...	V3	NaN	1.0	V2	59	36734	V1	1.0	V2	V1
1	2	0.0	V4	V5	18.0	4	42	V7	V3	V5	...	V3	NaN	1.0	V3	18	49442	V1	2.0	V2	V1
2	3	0.0	V4	V2	23.0	2	28	V3	V2	V2	...	V3	NaN	1.0	V3	23	60773	V1	1.0	V2	V1
3	4	0.0	V4	V3	10.0	4	40	V7	V4	V4	...	V3	NaN	1.0	V3	10	21595	V1	1.0	V2	V1
4	5	0.0	V1	V3	5.0	3	29	V3	V2	V1	...	V3	NaN	1.0	V3	5	52357	V1	1.0	V2	V1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1095	1096	NaN	V4	V5	18.0	4	36	V7	V4	V1	...	V1	NaN	1.0	V3	18	26481	V1	2.0	V2	V2
1096	1097	NaN	V2	NaN	29.0	2	30	V2	V2	V1	...	V3	NaN	1.0	V2	29	71302	V1	1.0	V2	V1
1097	1098	NaN	V1	V3	17.0	2	23	V3	V2	V1	...	V3	NaN	1.0	V3	17	92832	V1	1.0	V2	V1
1098	1099	NaN	V1	V3	47.0	2	22	V3	V2	V1	...	V3	NaN	1.0	V3	47	57740	V1	1.0	V2	V1
1099	1100	NaN	V3	V3	23.0	1	32	V9	V3	V1	...	V3	NaN	1.0	V3	23	102461	V1	1.0	V3	V1

Table.1. data2022.student dataframe

	ID	Class	C3	C4	C5	C10	C11	C12	C13	C16	C18	C20
count	1100.000000	1000.00000	1093.000000	1100.000000	1100.000000	1100.000000	1100.000000	5.000000	1100.000000	1100.000000	1100.0	1100.000000
mean	550.500000	0.27700	20.508692	2.976364	35.242727	3295.070909	3295.070909	3.000000	1.403636	4998.934545	0.0	2.852727
std	317.686953	0.44774	12.134215	1.117886	11.596656	2816.551353	2816.551353	1.581139	0.574548	1008.922370	0.0	1.102624
min	1.000000	0.00000	3.000000	1.000000	18.000000	249.000000	249.000000	1.000000	1.000000	1068.000000	0.0	1.000000
25%	275.750000	0.00000	11.000000	2.000000	26.000000	1371.500000	1371.500000	2.000000	1.000000	4353.500000	0.0	2.000000
50%	550.500000	0.00000	18.000000	3.000000	32.000000	2341.000000	2341.000000	3.000000	1.000000	5025.000000	0.0	3.000000
75%	825.250000	1.00000	24.000000	4.000000	42.000000	4006.500000	4006.500000	4.000000	2.000000	5692.500000	0.0	4.000000
max	1100.000000	1.00000	72.000000	4.000000	74.000000	18423.000000	18423.000000	5.000000	4.000000	8270.000000	0.0	4.000000

Table.2. data2022.student statistics describe

Methodology

2.1 1-unique values data

After plotting the data of every attribute (see in Figure. 1), note that there are some attribute columns with 1 unique value, i.e., C18 C21 C22 C25, so delete that data. Due to the inability to bring Classification and Prediction, then drop Column 'ID' that identifies an individual student which is not use in classification.

2.2 Duplication

Let's check the Duplicate data by starting from data in Row first, i.e.,

```
array([900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912,
       913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925,
       926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938,
       939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951,
       952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964,
       965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977,
       978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990,
       991, 992, 993, 994, 995, 996, 997, 998, 999], dtype=int64)
```

Figure1. Duplicate data in Rows

After checking Duplicate has Row 900-999(see in Figure.2), drop all rows 900-999, and then check Duplicate data in Column with results as shown in the Table. 1, therefore drop column last and keep first column where the stored column is C7, C10, C17 (see in Figure. 3)

```
C14 is a duplicate column of C7
C11 is a duplicate column of C10
C19 is a duplicate column of C17
```

Figure2. Duplicate data in Columns

2.3 Correlation

Plot the heat map to see the correlation values (see in Figure. 4). Note that, C3 and C27 have correlation close to 1, which mean they have similar data in Column (not duplicate) and they can affects the classification.

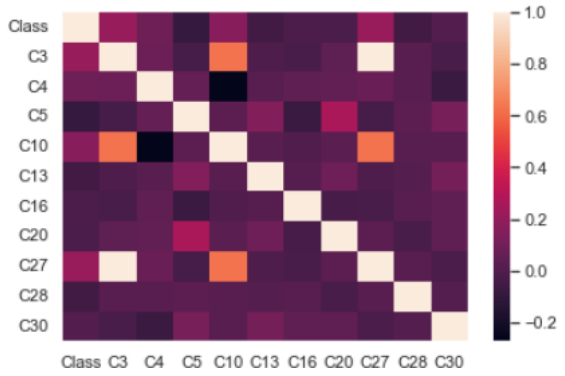


Figure3. Correlation checking (Before)

2.4 Missing values

Checking the missing data in attribute with the result as shown below.

```
C2 has 7 missing data
C3 has 7 missing data
C12 has 995 missing data
C23 has 6 missing data
C24 has 995 missing data
C30 has 6 missing data
```

Figure4. Missing values

Dealing with missing value have to do based on the type of data. Note that, there has some missing values in C3 but after referring to correlation C3 has correlation to C27 so the missing value of C3 has been replaced with the value in C27 because these 2 attributes have correlation value is close to 1, so C27 was dropped. Next attribute is C3, replace the Missing Value of C30 with the Mode (Most frequently data). Next step is attribute C2 and C23 with data type as object so replace missing value with Most frequently use and finally drop the C12 and C24 attributes due to excessive Missing value (1095 missing values).

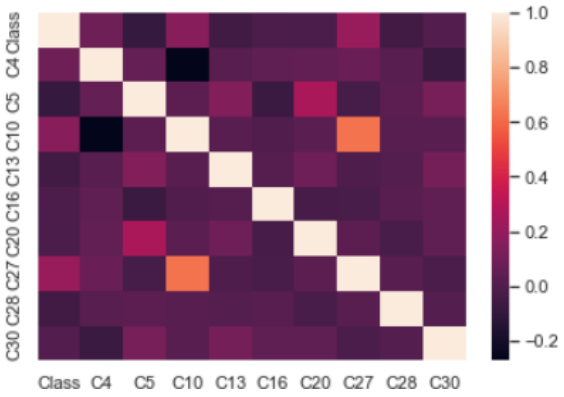


Figure5. Correlation checking (After)

2.5 Scaling and Standardization

Scaling the data in some attributes which are numerical data type to make model easier to predict. C5 C10 C16 C28 which are numerical attribute have to scale but for C27, it does not have to scale because after checking C27 values, it does not contain high values. After scale data C5, C10, C16, C28 will have standard deviation close to 1 as show in Table. 3.

2.6 Label Encoding

Due to making categorical data type changes to numerical data, it is easier to classification, so label encoding transform data becomes numeric, for example attribute 6 that is categorical (V1, V2,…, Vn) becomes 0,1,2 as shown in Table. 4.

	Class	C1	C2	C4	C5	C6	C7	C8	C9	C10	...	C17	C20	C23	C26	C27	C28	C29	C30	C31	C32
0	0.0	0	2	1	-0.703563	3	2	4	1	1.469970	...	1	1	2	1	3.198996	-0.095598	0	1.0	1	0
1	0.0	3	4	4	0.611017	7	2	4	1	-0.132112	...	1	4	2	2	-0.199165	0.338833	0	2.0	1	0
2	0.0	3	1	2	-0.615925	3	1	1	1	-0.133176	...	1	1	2	2	0.215245	0.726191	0	1.0	1	0
3	0.0	3	2	4	0.435740	7	3	3	1	-0.690715	...	1	4	2	2	-0.862221	-0.613135	0	1.0	1	0
4	0.0	0	2	3	-0.528286	3	1	0	1	-0.975687	...	3	1	2	2	-1.276631	0.438485	0	1.0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1095	NaN	3	4	4	0.085185	7	3	0	1	-0.845961	...	1	4	0	2	-0.199165	-0.446104	0	2.0	1	1
1096	NaN	1	5	2	-0.440647	2	1	0	1	-0.057679	...	0	4	2	1	0.712537	1.086132	0	1.0	1	0
1097	NaN	0	2	2	-1.054116	3	1	0	1	-0.028969	...	0	2	2	2	-0.282047	1.822150	0	1.0	1	0
1098	NaN	0	2	2	-1.141757	3	1	0	1	0.949648	...	1	2	2	2	2.204412	0.622506	0	1.0	1	0
1099	NaN	2	2	1	-0.265370	9	2	0	1	0.209571	...	3	2	2	2	0.215245	2.151323	0	1.0	2	0

Table.3. data2022.student statistics dataframe after dealing scaling and labelling

	Class	C1	C2	C4	C5	C6	C7	C8	C9	C10	...	C17
count	900.000000	1000.000000	1000.000000	1000.000000	1.000000e+03	1000.000000	1000.000000	1000.000000	1000.000000	1.000000e+03	...	1000.000000
mean	0.277778	1.536000	2.405000	2.973000	1.190159e-16	4.255000	1.745000	1.105000	0.963000	7.105427e-18	...	1.577000
std	0.448152	1.215805	1.222485	1.116715	1.000500e+00	2.756733	1.055978	1.560023	0.188856	1.000500e+00	...	1.257638
min	0.000000	0.000000	0.000000	1.000000	-1.492311e+00	0.000000	0.000000	0.000000	0.000000	-1.071032e+00	...	0.000000
25%	0.000000	0.000000	2.000000	2.000000	-7.912020e-01	2.000000	1.000000	0.000000	1.000000	-6.755621e-01	...	0.000000
50%	0.000000	1.000000	2.000000	3.000000	-2.653699e-01	3.000000	1.000000	0.000000	1.000000	-3.375121e-01	...	1.000000
75%	1.000000	3.000000	4.000000	4.000000	5.233782e-01	7.000000	3.000000	2.000000	1.000000	2.482938e-01	...	3.000000
max	1.000000	3.000000	5.000000	4.000000	3.415455e+00	9.000000	4.000000	4.000000	1.000000	5.370617e+00	...	3.000000

Table.4. data2022.student statistics describe after dealing scaling and labelling

2.7 Imbalanced Data

In the process of preparing data the problem that will occur is imbalance data problem., i.e., in attributes it's contain 2 variable which has 2 for 99 data and 1 for 1 data (see in Figure.) so “oversampling” or “undersampling” have to apply for dealing with this problem. SMOTE is library to deal with it by setting the random state as 99 and do oversampling (it should do both oversampling and undersampling so the data will fix the quantity to deal with imbalanced problem. Figure. and Figure. will show the result after applying SMOTE. The others plot for each attribute were shown in appendices.

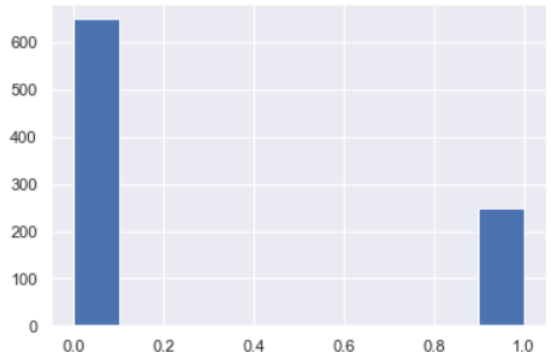


Figure6. Histogram plot: Class attributes (Before oversampling and undersampling)

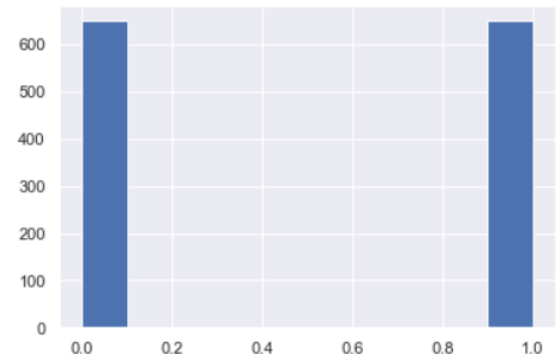


Figure6. Histogram plot: Class attributes (After oversampling and undersampling)

2.8 Separation of Data

After cleaning data, data will contain 900 rows of data and 23 columns., i.e.,

Data. Shape () = (900,23)

Data will store in X and y which X does not have attributes “Class”

2.9 Splitting Data

Split data into two parts, 90% and 10%, using test size of 0.1, where 90% is the training dataset and 10% is testing dataset., i.e.,

trainX, ValidX, trainY, ValidY= train_test_split (X, y, test_size=0.1, stratify=y)

2.10 Cross Validation

StratifedKfold cross validation was picked to make cross validation and set K-fold as follow

skf = StratifiedKFold(n_splits=10, shuffle=True)

2.11 Data Classification

•K-nearest neighbors

One of the most straightforward non-parametric strategies for classification and regression is nearest neighbors. All accessible data is saved by the K-NN algorithm, which then classifies each new data point according to how similar it is to the preceding data point. This suggests that new data can be quickly categorized using the K-NN algorithm into an appropriate category when it is generated. It is supervised learning, which means that labels are necessary for learning.

• Decision Tree

Although it is most frequently used to address classification issues, Decision Trees are a sort of Supervised Learning approach that can

be used to address both regression and classification problems. It is a tree-structured classifier, with internal nodes denoting dataset properties, branches denoting choice criteria, and each leaf node denoting the result.

- Naïve-Bayes

Naive Bayes is an easy-to-use supervised machine learning technique that uses the Bayes theorem and strong independence presumptions between the features to produce results. In other words, the method implicitly assumes that every input variable is independent. [1].

- Random Forest

Even without hyperparameter tweaking, the flexible, user-friendly machine learning technique known as Random Forest regularly produces outstanding results [2]. Due to its versatility (it can be used for both classification and regression problems), it is also one of the most widely used algorithms. A technique for supervised learning is random forest. An ensemble of Decision trees, which are frequently trained using the bagging approach, are combined to form a forest. The fundamental tenet of the bagging approach is that the result is improved when multiple learning models are combined [3].

- Gradient Boosting

Gradient boosting is one type of machine learning boosting. It is based on the suspicion that the overall prediction error is decreased when earlier models are combined with the best upcoming model. The main idea is to set the desired outcomes for this future model to minimize error. The desired outcome will differ for each instance in the data depending on how changing a case's forecast affects the overall amount of prediction error [7].

2.12 Hyper parameter tuning

Hyper parameter was selected from GridsearchCV which give the best accuracy and F1-score. GridsearchCV are as follow, i.e.,

- K-nearest neighbors

param_grid = {'n_neighbors':[1,2,3,4,5,6,7,8,9,10],'n_jobs':[-1,1,2,None],'weights':['distance',None],'p':[1,2,3,4]}– KNeighborsClassifier (n_neighbors=5, n_jobs = -1, p=1)

- Decision Tree

param_grid = {'max_features':['auto','sqrt','log2',None],'max_depth':[12,14,16,None],'criterion':['gini','entropy',None],'min_samples_split':[40,50,60,None],'splitter':['random',None]}– DecisionTreeClassifier (max_depth=12, criterion='entropy', , 'random_state':[42])

- Naïve-Bayes

Except Naïve-Bayes which use GaussianNB that we decide to use only var-smoothing so we pick it by random.

– GaussianNB ()

– GaussianNB(var_smoothing=0.0003)

– GaussianNB(var_smoothing=3e-9)

- Gradient Boosting

param_grid = {'n_estimators':[100,500,600,None],'learning_rate':[0.5,0.01,0.05,None],'max_features':['auto','sqrt','log2',None],'n_estimators':[100,150,200,None],'criterion':['friedman_mse','squared_error',None],'max_depth':[12,14,16,None],'min_samples_split':[40,50,60,None] , 'random_state':[42] }

- Random Forest

param_grid={'n_estimators':[100,150,200, None],'max_features':['auto','sqrt','log2', None],'max_depth': [20,40,60, None],'criterion': ['gini','entropy', None], 'random_state':[42] }

After obtaining the best parameter, the cross

validation will be used, and the other 3 set of parameters will be plot with the best accuracy's set of parameters for comparison and plotting within same model.

2.13 Training-Prediction process

The model was trained using the training set, and it was validated using the test set. The steps are described in detail below.

• Training process:

–The models were built based on the hyperparameters tuning,

–Accuracy from Cross Validation were recorded,

–F1-Score from Cross Validation were recorded,

– Confusion Matrix were recorded.

• Prediction Process

– Class prediction was obtained via the modelfitting with the best 2 classifier,

– Accuracy from test dataset were recorded,

– F1-Score from test dataset were recorded,

– Confusion Matrix were recorded,

–Estimated accuracy for prediction werecalculated by accuracy approximate

Result

After applying GridsearchCV, the result should be as follow., i.e.,

•K-nearest neighbors

For K Nearest Neighbor Classifier, the best parameter that obtain best accuracy is n_jobs= -1, n_neighbors=2, p=1 and Weights are None. Weights determine the weight function used in the prediction and how distributed between neighbor values [4], which it gives weight function as None. P determines the type of distance computation and P equals to 1. N_neighbor mean the number of neighbors that close to K and in this section is 3[8]. After applying Cross Validation and test data, Table. 5 shown the result of accuracy, F1-score, and Figure.7 and Figure.8 shown the visualization of the best one parameter compared to another parameter and Figure.9 show confusion matrices of K-nearest neighbors.

The estimated accuracy would be 68%.

K-nearest neighbors	Train	Test
Accuracy	81.7%	85.4%
F1-score	83.1%	85.7%

Table.5 K-nearest neighbors Accuracy and F1-score for Train and Test dataset

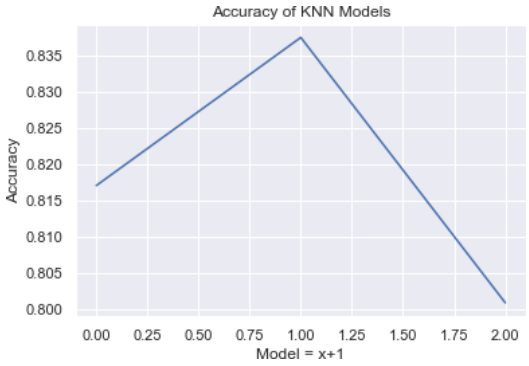


Figure7. K-nearest neighbors: Accuracy plot with Cross Validation

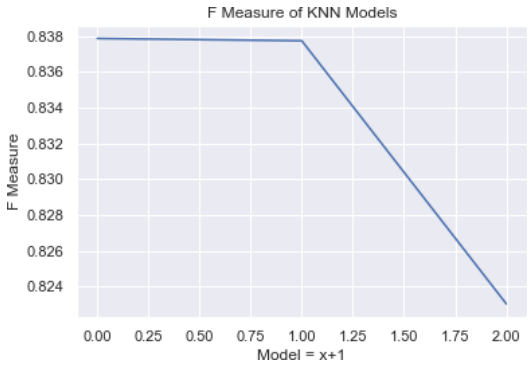


Figure8. K-nearest neighbors F1-score plot with Cross Validation

```
[[52  7]    [[51  9]
 [ 6 52]]    [ 8 49]]
[[50 14]    [[46  7]
 [ 8 45]]    [13 51]]
[[46  8]    [[51 11]
 [12 51]]    [ 8 47]]
[[48  8]    [[49  6]
 [10 51]]    [10 52]]
[[48  6]    [[50  4]
 [10 53]]    [ 9 54]]
```

Figure9. Confusion Matrix for K-nearest neighbors (Cross Validation 1-5: left, Cross Validation 6-10:)

- Decision Tree

For Random Forest Classifier, the best parameter that obtain best accuracy are DecisionTreeClassifier(random_state=42). Random state controls the randomness of the estimator. It needs to be a fixed integer to obtain a deterministic behavior. n_estimators define the number of trees in the forest. Criterion is the function which is used to measure the quality of the split. Choosing entropy uses information gain as the criterion. min_samples_split defines the minimum number of samples which is needed to split an internal node. Max_depth is parameter used to define the maximum depth of the tree. But with GridsearchCV, it gives result as simple way: random_state=42.After applying Cross Validation and test data, Table. 6 shown the result of accuracy, F1-score, and Figure.10 and Figure.11 shown the visualization of the best one parameter compared to another parameter and Figure.12 show confusion matrices of K-nearest neighbors.

The estimated classification accuracy would be 60%.

Decision Tree	Train	Test
Accuracy	75.9%	74.6%
F1-score	76.1%	76.9

Table.6 Decision Tree: Accuracy and F1-score for Train and Test dataset

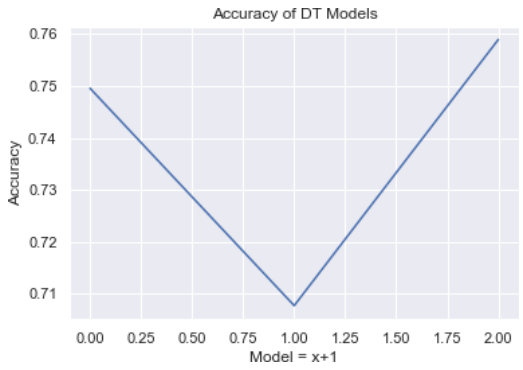


Figure10. Decision Tree Accuracy plot with Cross Validation

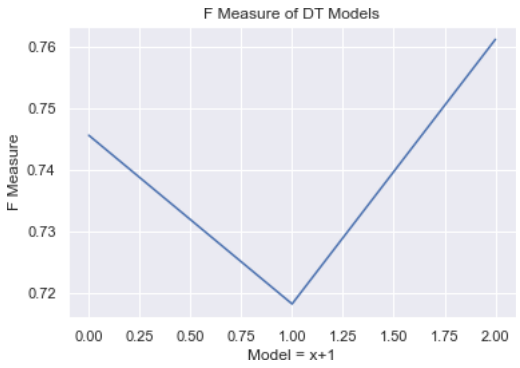


Figure11. Decision Tree F-1 score plot with Cross Validation

```
[[44 21]    [[42 13]
 [14 38]]    [17 45]]
[[45 13]    [[47  5]
 [13 46]]    [12 53]]
[[41 13]    [[44 16]
 [17 46]]    [15 42]]
[[49 14]    [[47 13]
 [ 9 45]]    [12 45]]
[[42 14]    [[41 14]
 [16 45]]    [18 44]]
```

Figure12. Confusion Matrix for Decision Tree (Cross Validation 1-5: left, Cross Validation 6-10: right)

- Naïve-Bayes

For Naive Bayes classifier, it gives result as var smoothing=0.0003 which var smoothing is a smoothing technique that help tackle problem of zero probability [9]. Using higher alpha values will push the likelihood towards a value. After applying Cross Validation and test data, Table. 7 shown the result of accuracy, F1-score, and Figure.13 and Figure.14 shown the visualization of the best one parameter compared to another parameter and Figure.15 show confusion matrices of Naïve-Bayes classifier.

The estimated classification accuracy would be 68%.

Naïve-Bayes	Train	Test
Accuracy	76.9%	76.9%
F1-score	78.4%	80.0%

Table.7 Naïve-Bayes: Accuracy and F1-score for Train and Test dataset

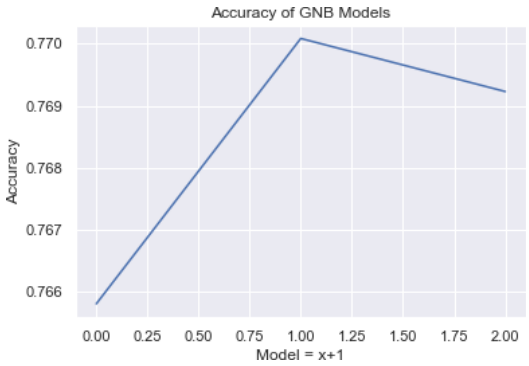


Figure13. Naïve-Bayes Accuracy plot with Cross Validation

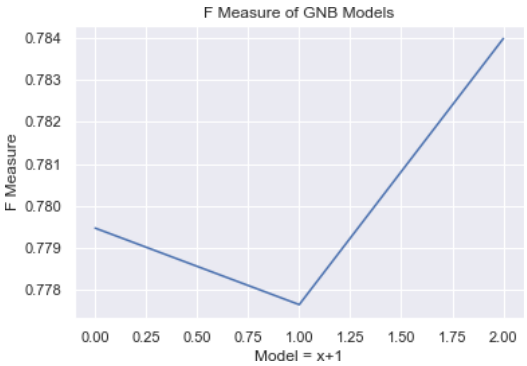


Figure14. Naïve-Bayes F1-score plot with Cross Validation

```
[[40  7]     [[36 12]
 [18 52]]     [23 46]]
[[45 14]     [[48 18]
 [13 45]]     [11 40]]
[[42  8]     [[36  5]
 [16 51]]     [23 53]]
[[36  8]     [[40  9]
 [22 51]]     [19 49]]
[[47 13]     [[38  6]
 [11 46]]     [21 52]]
```

Figure15. Confusion Matrix for Naïve-Bayes (Cross Validation 1-5: left, Cross Validation 6-10:)

- Gradient Boosting

For Gradient Boosting classifier, it gives result as n_estimators=1000, learning_rate=0.05,max_features='auto', max_depth=20,min_samples_split=20, criterion='friedman_mse',random_state=42) which max_depth use to control overfitting as higher dept will allow model to learn relations very specific to a particular sample, max_features is the number of features while choosing the best split, learning_rate control the magnitude of this change in the estimates, n_estimators is the number of sequential trees to be modeled, min_samples_split is used to define the minimum number of samples which are required in a node to be considered for splitting, criterion is used for define loss[8]. After applying Cross Validation and test data, Table. 8 shown the result of accuracy, F1-score, and Figure.16 and Figure.17 shown the visualization of the best one parameter compared to another parameter and Figure.18 show confusion matrices of Gradient Boosting classifier.

The estimated classification accuracy would be 70%.

Gradient Boosting	Train	Test
Accuracy	85.9%	88.5%
F1-score	86.0%	89.1%

Table.8 Gradient Boosting: Accuracy and F1-score for Train and Test dataset

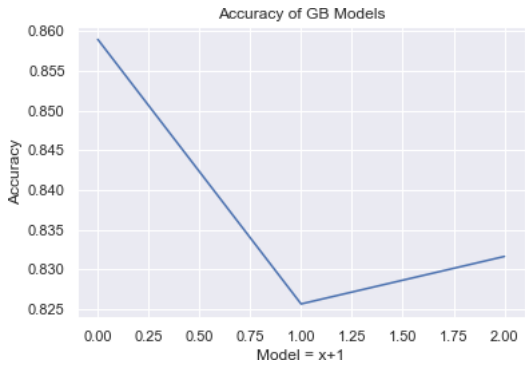


Figure16. Gradient Boosting for Accuracy plot with Cross Validation

Figure17. Gradient Boosting for F1-score plot with Cross Validation

```
[[54  8]     [[46 13]
 [ 5 50]]     [12 46]]
[[48  7]     [[49  6]
 [11 51]]     [ 9 53]]
[[56  7]     [[49  4]
 [ 3 51]]     [ 9 55]]
[[49 12]     [[50  7]
 [10 46]]     [ 8 52]]
[[48  6]     [[50  9]
 [11 52]]     [ 8 50]]
```

Figure18. Confusion Matrix for Gradient Boosting (Cross Validation 1-5: left, Cross Validation 6-10:)

- Random Forest

For Random Forestclassifier, it gives resultas n_estimators=200,min_samples_split=3, random_state=42,max_features= 'auto',criterion='entropy',bootstrap=False,max_depth=18,n_jobs= -1 which n_jobs tells the

engine how many processer is it allowed to use[6], bootstrap is a method that involves drawing of sample data repeatedly with replacement from a data source to estimate a population parameter[5]. After applying Cross Validation and test data, Table. 8 shown the result of accuracy, F1-score, and Figure.19 and Figure.20 shown the visualization of the best one parameter compared to another parameter and Figure.21 show confusion matrices of Radom Forest classifier.

The estimated classification accuracy would be 70%.

Gradient Boosting	Train	Test
Accuracy	86.1%	85.4%
F1-score	86.2%	85.7%

Table.9 Random Forest: Accuracy and F1-score for Train and Test dataset

Figure19. Random Forest for Accuracy plot with Cross Validation

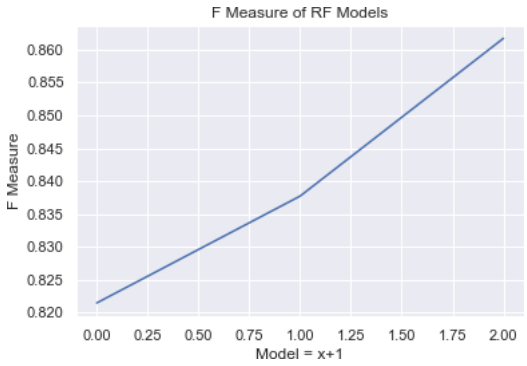


Figure20. Random Forest for F1-score plot with Cross Validation

```
[[46 11]     [[52  5]
 [13 47]]     [ 6 54]]
[[51  6]     [[49  8]
 [ 8 52]]     [ 9 51]]
[[50  7]     [[51  6]
 [ 9 51]]     [ 7 53]]
[[49 10]     [[49  8]
 [10 48]]     [ 9 51]]
[[53  7]     [[49  9]
 [ 6 51]]     [ 9 50]]
```

Figure21. Confusion Matrix for Random Forest (Cross Validation 1-5: left, Cross Validation 6-10:)

Comparison

From Figure22 and Figure 23 show that the best accuracy and F1-score on test data is Gradient Boosting but for second place, Random Forest and K-nearest neighbors has closely values so we must check for train set on cross validation. From Figure24 and Figure25 show that the best accuracy on training dataset is Random Forest with the second place is Gradient Boosting, so we decide to pick Random Forest and Gradient Boosting as the best 2 classifier in this work.

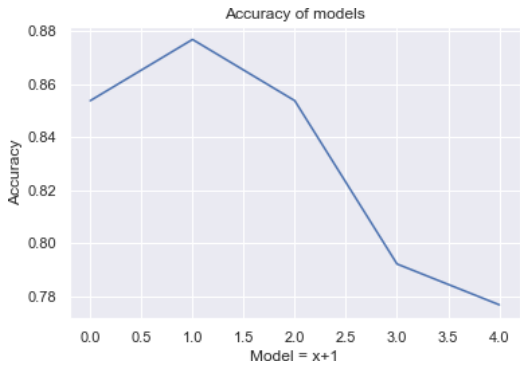


Figure22. Accuracy on test dataset in every classifier

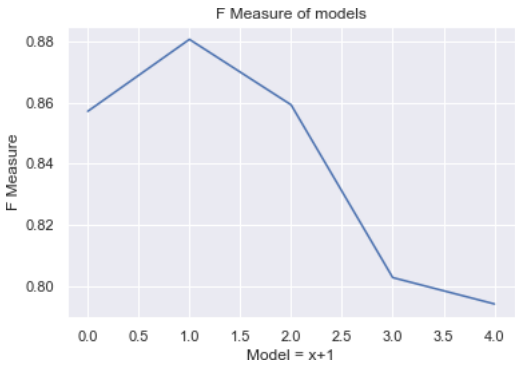


Figure23. F1-score on test dataset in every classifier

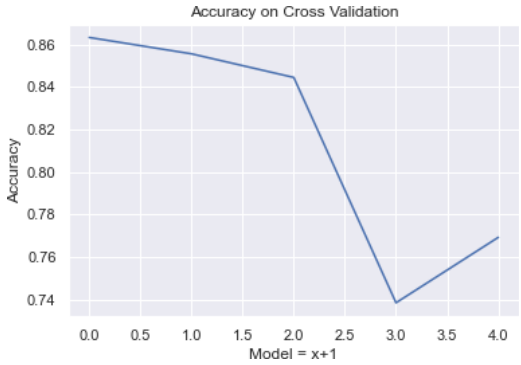


Figure24. Accuracy on training dataset with cross validation in every classifier

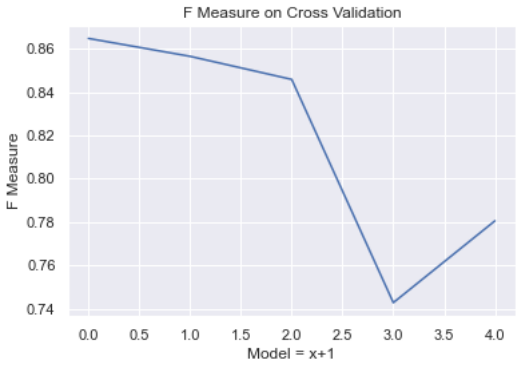


Figure25. F1-score on training dataset with cross validation in every classifier

The estimated classification accuracy would be 68%.

Conclusion

From the above experiment, note that the estimate accuracy quite low so from GridsearchCV should be improved to gain more suitable hyperparameter to use in classification and obtain the new higher accuracy than before.

References

[1] Sunil Ray. 6 Easy Steps to Learn Naive Bayes Algorithm with codes in Python and R. url:https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/.

[2] Jason Brownlee. How to Calculate Correlation Between Variables in Python. url: https://machinelearningmastery.com/how-to-use-correlation-to-understand-the-relationship-between variables/.

[3] Kamila Hamalcikova. Random Forest for imbalanced dataset: example with avalanches in FrenchAlps.url:https://towardsdatascience.com/random- forest- for-imbalanced-dataset-example with-avalanches-in-french-alps-77ffa582f68b.

[4] kNN Optimization . url: https://holypython.com/knn/k-nearest-neighbor-optimization-parameters/?fbclid=IwAR2PhKykUR9UA_34mgtwXB-hRB3sb0fOX6y_iwu6NeAEuElyTUM7zVFlwIM.

[5] Jason Brownlee. A Gentle Introduction to the Bootstrap Method.url: https://machinelearningmastery.com/a-gentle-introduction-to-the-bootstrap-method/?fbclid=IwAR1zB_UJMxb4MPcX_diKMYxJilX_ZDyFcsa36hp7yI24eJK_0JpQChz_WBE.

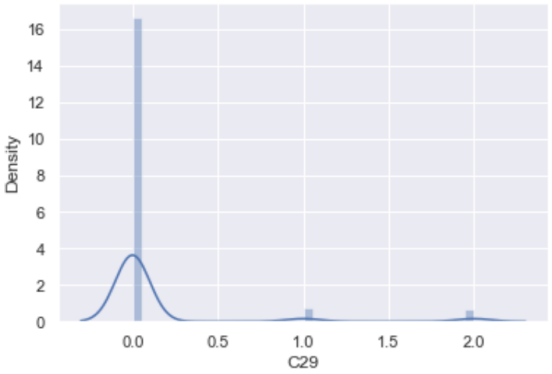
[6] Tavish Srivastava. Tuning the parameters of your Random Forest model. url: https://www.analyticsvidhya.com/blog/2015/06/tuning-random-forest-model/?fbclid=IwAR2RHhrelH8augz_HterZRVag_h0Yrf29Hmz3zhfeIAQYDmeyVevY6jCLNQ.

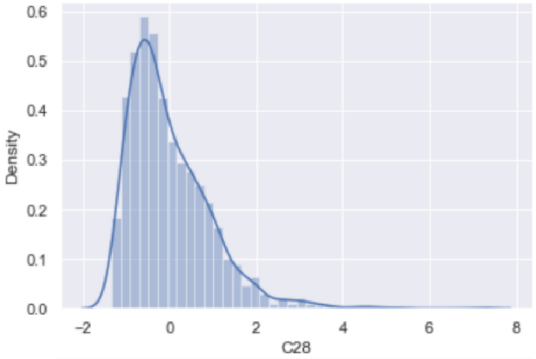
[7] Aarshay Jain. Complete Machine Learning Guide to Parameter Tuning in Gradient Boosting (GBM)in Python. url: https://www.analyticsvidhya.com/blog/2016/02/complete-guide-parameter-tuning-gradient-boosting-gbm-python/?fbclid=IwAR0HgY6NS9FP_1Pr6itytF95kySe8jE2AUWv7Zoeztp3QriwAdBA8Iz9MlI

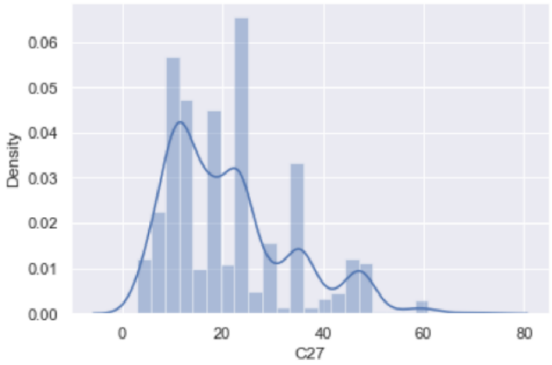
[8] Mohtadi Ben Fraj. In Depth: Parameter tuning for KNN. url: https://medium.com/@mohtedibf/in-depth-parameter-tuning-for-knn-4c0de485baf6

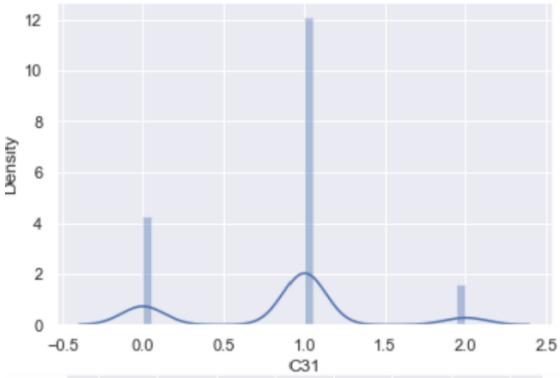
[9] Kopal Jain. How to Improve Naive Bayes? url: https://medium.com/analytics-vidhya/how-to-improve-naive-bayes-9fa698e14cba

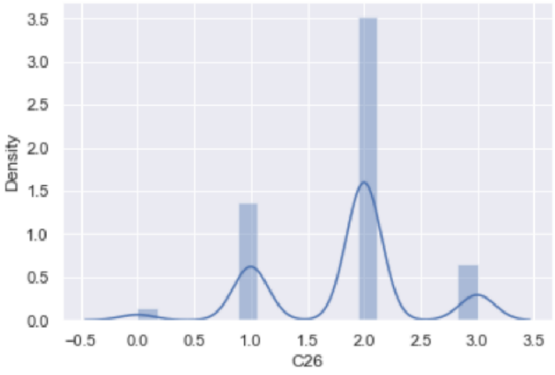
Appendices

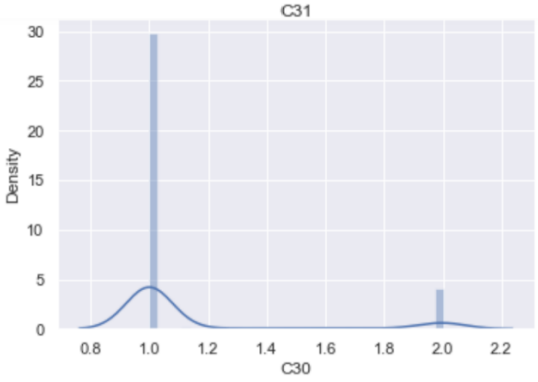


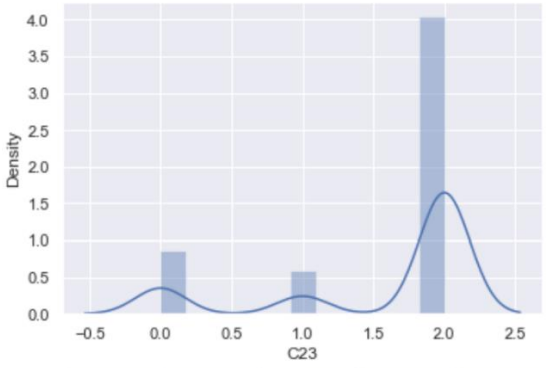


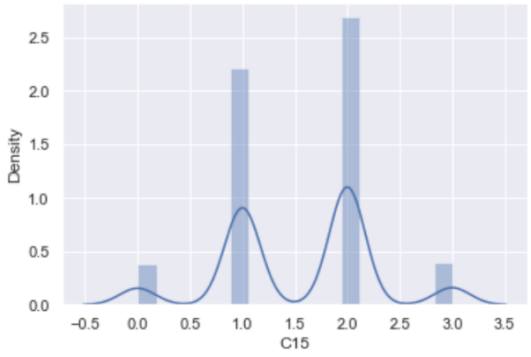


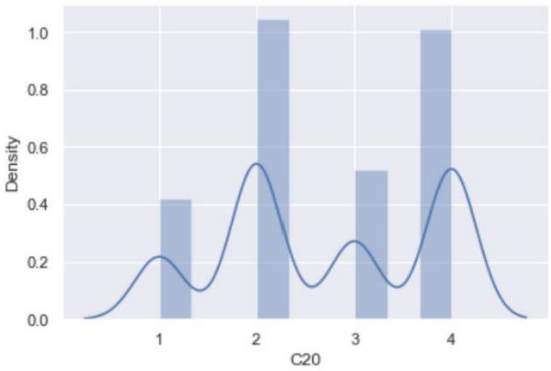


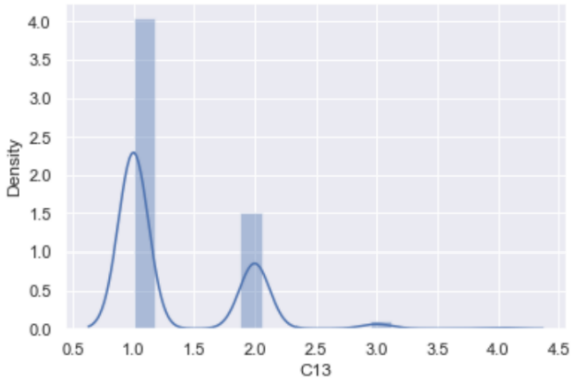


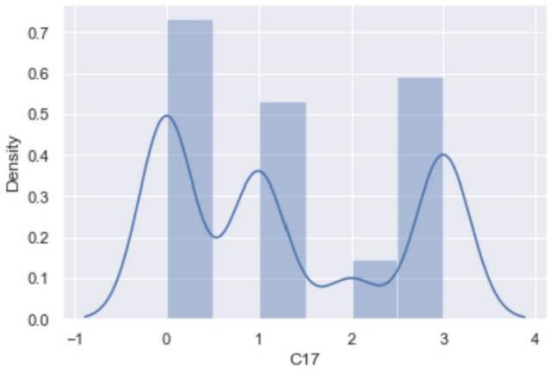


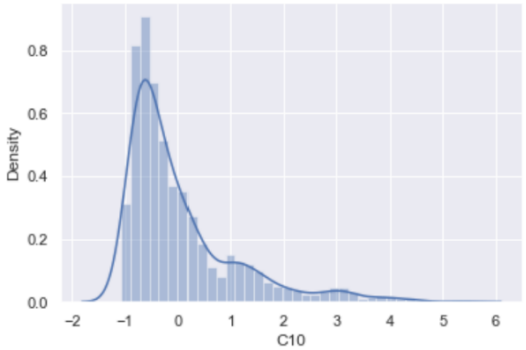

Density
C23

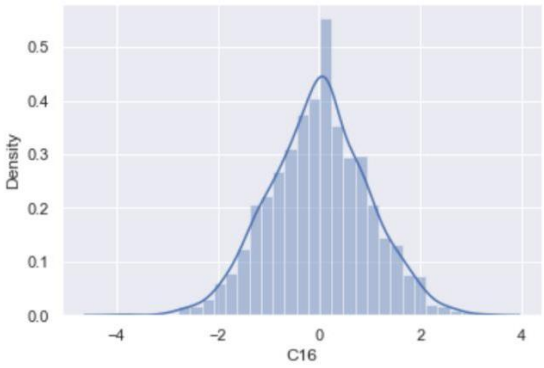

Density
C15

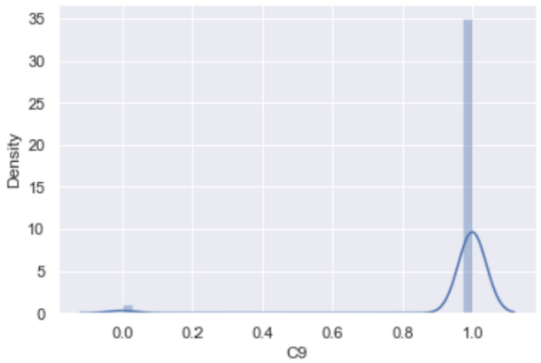

Density
C20

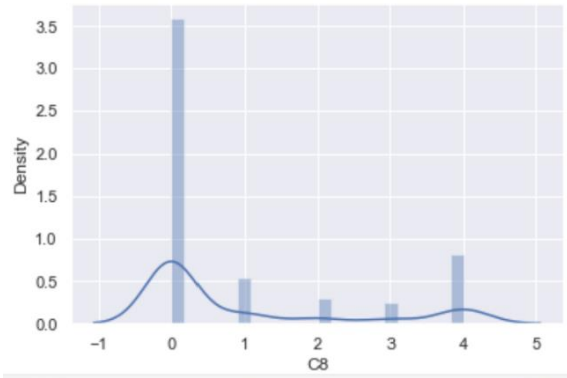

Density
C13

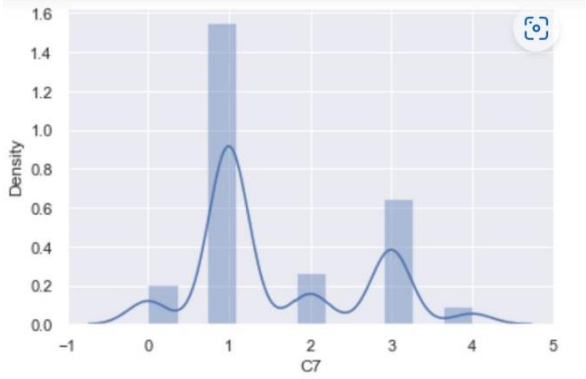

Density
C17

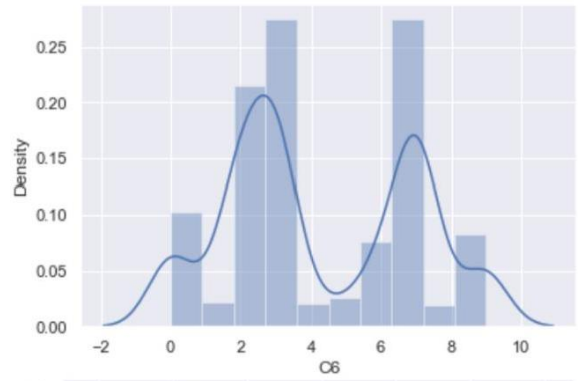

Density
C10

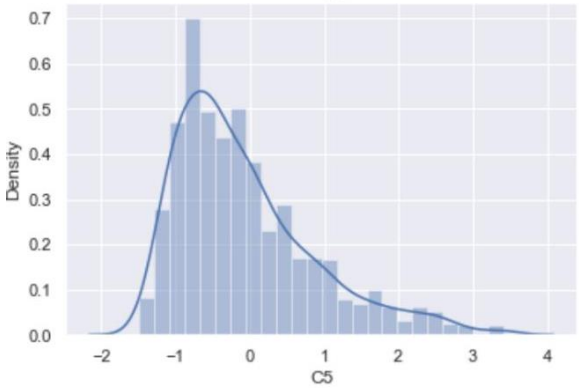

Density
C16

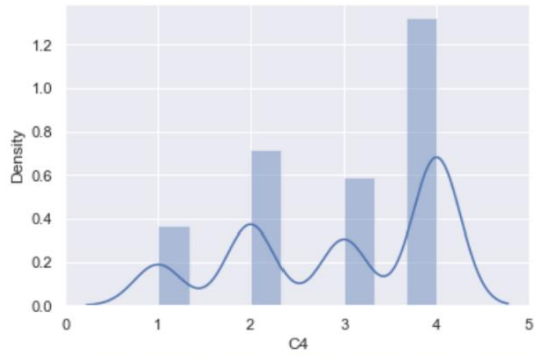

Density
C9

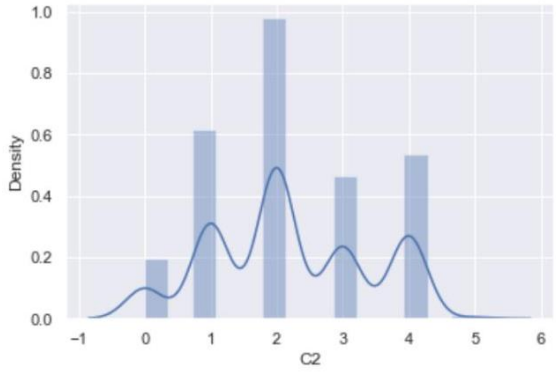


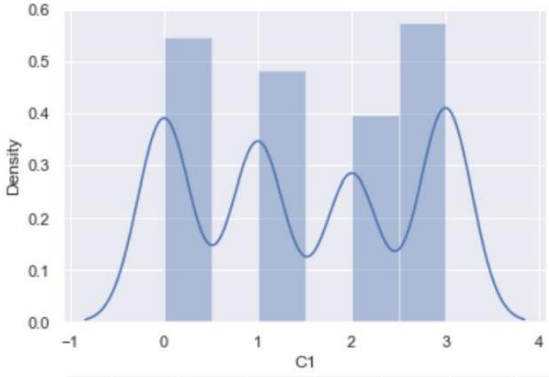


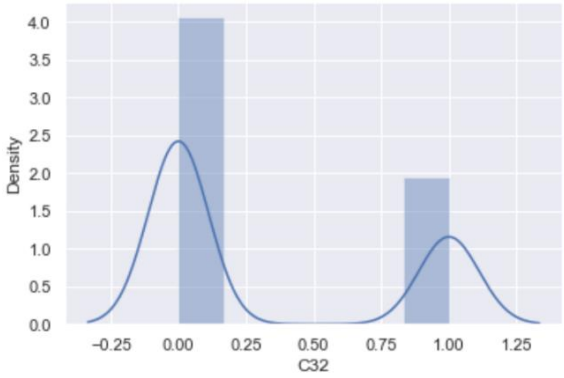












C16	int64	ID	int64
C17	object	Class	float64
C18	float64	C1	object
C19	object	C2	object
C20	int64	C3	float64
C21	object	C4	int64
C22	object	C5	int64
C23	object	C6	object
C24	object	C7	object
C25	float64	C8	object
C26	object	C9	object
C27	int64	C10	int64
C28	int64	C11	int64
C29	object	C12	float64
C30	float64	C13	int64
C31	object	C14	object
C32	object	C15	object