

Name: Yeo Chee En Luke
Student Num: 46746267 (S4674626)
COMP4702 Assignment
Date: 26/5/2023

Problem objective:

Using Machine learning models to predict if the food is solid or liquid (classification)

Proposed methods :

Machine learning model use for this assignment are logistic regression , KNN-classifier and MLP (ReLu and Sigmoid)

Reason for picking logistic regression:

- Logistic regression is a simple and interpretable model, making it easy to understand and implement.
- It handles high-dimensional data well.

Reason for picking KNN-classifier:

- It is non-parametric: it does not make any assumption about the data distribution which makes allows us to do complex and non-linear decision boundaries allowing It to handle high-dimensional data extremely well.
- Irrelevant features have less impact on KNN as it is predict by closest neighbour which makes KNN less affected by the curse of dimensionality

Reason for picking MLP (Relu and sigmoid)

- It can capture complex non-linear relationships in the data which allows better accuracy
- It scale well with large dataset and can handle high-dimensional spaces effectively
- Relu Reason:
 - It has a constant gradient for positive inputs which avoids the vanishing gradient problem that sigmoid suffers that effects the models ability to learn comlex patterns effectively
 - It has property of sparse activation which makes it more efficient as only relevant features are selected ad contribute to the classification process.
- Sigmoid Reason:
 - It handles imbalance data more effectively as compare to Relu. Which is 1 of the issue for the dataset (explain below)

Expectation:

Base on the reasoning above, i expect logistic regression to be the worst among the 3, using it as a “baseline” model for comparison for results while neural network or KNN to be the best choice to solve the objective.

Data Analysis

- From an inspection of 'Rel_2_nutrient_file.xlsx', in both datasheet, Sheet 1 of the datasheets has "All solids & liquids per 100g," but Sheet 2 only contains "Liquids per 100mL." There are numerous numerical columns for each different food-related element. The columns "Public Food Key" and "Food Name" are the only ones that lack a numerical value.
- Another issue with the dataset is the imbalance of the data, in the Sheet1 having the shape of (1616, 293), there are only 221 liquid while 1395 are solid which means the model will be biased to solid since it is the majority class. Thus, it may impact the evaluation (High accuracy due to majority class but lower on minority class) as there is not enough liquid class to learn from which gives poor generalization and overfitting . Hence we base it off F1-score than accuracy.
- There were Number of issues with the dataset :
 - There were many missing data Nan Values , 60.523181157706205% of entire dataset were Nan for Sheet 1
 - For the 'All solids & liquids per 100g', 2 columns were Nan for classification column with Public Food Key 'F009803' and 'F009834'
 - columns that may not provide any information

Solution to solve them:

- 1) Missing values in classification column, i check through other files (Rel_2_Food_detail) , to find the classification for them and added them into the dataset.
- 2) In regards to the many missing Nan value in terms of the dataset, i replaced all Nan (empty) with value '0.0'. Then check if an entire column is made of 0.0 , if it is , i would delete that column as it provides no useful information at all. Leaving me with 252 columns left after deleting 41 columns.
- 3) In regards to many columns that may have useless information, i have done feature selection (discussed below)
- 4) For data imbalancing, ive used the F1-score instead of accuracy score and changing the ratio of the train-test split.

Reason of solution:

- 1) Using the other sources (other data file provided in the assignment zip) as a method to get the missing value correctly is better than using classification number of similar food
- 2) This is to ensure that there are no more Nan value. The reason why i did not use the mean of the column or of specific food based (e.g. different type of yogurt) is due to the fact that there maybe added or lessen ingredients compare to other food which gives a totally different number. Hence i picked 0.0 as it is the safest option.
- 3) This is because there might be useless columns that does not provide any information in terms of solving the objection. Hence i have decided to do feature selection, using correlation. However to find out if there are any columns that are useless, ive decided to use the full dataset column in comparison with the top 20 correlation.
- 4) Due to data imbalancing, the accuracy score can be high due to achieving high prediction in the majority class (solid) . As such, F1 score is more reliable as a

measurement in performance as the goal is to identify the minority (liquid) class correctly. This is because F1 score take account precision and recall which makes it more suitable for data imbalance distribution dataset. The reason to split the train and test data is to ensure that the model learns on both minority class and the majority class. So that it can make better prediction.

Changes Made to dataset for Classification Model: Solid or Liquid

Since the classification column are not based by solid or liquid (in binary) but instead by a 4-numerical value, i used the 'classification on the sheet 2: 'Liquids only per 100mL' to set those numbers in the sheet 1: 'All solids & liquids per 100g' to value of '1' in a new column call 'category' while those that are not in sheet 2 as value of '0' . Hence making it binary for classification.

For this report , 'all columns' doesnt not include the 'category' column to prevent data leakage (giving the answer directly) to the model.

Solid = '0'

Liquid = '1'

Library used throughout the entire Assignment:

```
!pip install mlxtend
!pip install seaborn
!pip install tensorflow
import seaborn as sns
!pip install openpyxl
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
import tensorflow as tf
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.metrics import classification_report
from sklearn.metrics import f1_score
```

Feature Selection: (using all columns vs top 20 correlation)

Firstly , before feature selection , i split the dataset into train and test and setting the random state to random state 42 to ensure consistency among the data models.

Reason 1: If feature selection is done after splitting the data, it may introduce risk of data leakage, where information from the test set leak into the training, which results in over-optimistic performance estimates. Hence by doing feature selection after splitting the data, it prevents any biasness .

Reason 2: Feature selection also helps to reduce the complexity of the data, which prevents overfitting as if the model becomes too complex and capture noises and useless patterns, it will affect the model performance.

I have decided to use a correlation on the train dataset after splitting the data to train and split to find the top 20 correlation value. The reason why i used the top 20 correlation is to get as much distinct columns as possible in a range or value (0.1 to 0.3) which may help the models to better predict if a food is a solid or liquid.

How i did this was using the absolute value. By using the absolute value, it determines how closely the columns are related regardless of whether the relationship is positive or negative. Which allows us to identify columns with the strongest overall relationship.

Using the results of the correlation accuracy and F1 score vs the entire dataset column, i would be able to know if there are any columns that gives irrelevant information for the objective.

Correlation table (unable to fit all)									
	Energy with dietary fibre, equated (kJ)	Energy, without dietary fibre, equated (kJ)	Moisture (water) (g)	Protein (g)	Nitrogen (g)	Fat, total (g)	Ash (g)	Ti diet fibre	
Energy with dietary fibre, equated (kJ)	1.000000	0.997643	-0.865172	0.119396	0.132005	0.844911	0.010301	0.155	
Energy, without dietary fibre, equated (kJ)	0.997643	1.000000	-0.845697	0.125247	0.137084	0.854102	0.003248	0.087	
Moisture (water) (g)	-0.865172	-0.845697	1.000000	-0.127595	-0.143633	-0.504953	-0.237102	-0.380	
Protein (g)	0.119396	0.125247	-0.127595	1.000000	0.998392	0.021909	0.008337	-0.069	
Nitrogen (g)	0.132005	0.137084	-0.143633	0.998392	1.000000	0.029011	0.010496	-0.056	
Fat, total (g)	0.844911	0.854102	-0.504953	0.021909	0.029011	1.000000	-0.015208	-0.023	
Ash (g)	0.010301	0.003248	-0.237102	0.008337	0.010496	-0.015208	1.000000	0.104	
Total dietary fibre (g)	0.155191	0.087130	-0.380972	-0.069644	-0.056998	-0.023209	0.104189	1.000	
Alcohol (g)	-0.053694	-0.050691	0.082367	-0.119112	-0.119365	-0.061545	-0.034045	-0.049	
Fructose (g)	-0.061537	-0.065850	-0.024361	-0.264957	-0.265093	-0.128648	-0.014508	0.059	
Glucose (g)	-0.042411	-0.046000	-0.051816	-0.272905	-0.273192	-0.124854	-0.010510	0.049	
Sucrose (g)	0.219054	0.222000	-0.321308	-0.186748	-0.184591	0.018485	-0.019185	-0.014	
Maltose (g)	0.152748	0.150097	-0.265138	-0.087220	-0.080939	-0.038880	0.023596	0.052	
Lactose (g)	0.117082	0.120251	-0.139837	0.003324	-0.000080	0.048202	0.042126	-0.029	
Galactose (g)	-0.041296	-0.041798	0.032698	-0.033578	-0.032647	-0.026629	-0.002635	0.002	
Maltotriose (g)	0.045002	0.046108	-0.078043	-0.046470	-0.047192	-0.010296	-0.011136	-0.010	
Total sugars (g)	0.198056	0.197846	-0.344236	-0.299097	-0.297096	-0.039121	-0.012491	0.038	
Added sugars (g)	0.213025	0.217509	-0.322788	-0.215514	-0.214887	-0.000156	-0.024019	-0.037	
Free sugars (g)	0.208582	0.213073	-0.322033	-0.225888	-0.225154	-0.006629	-0.026246	-0.037	
Starch (g)	0.343830	0.326765	-0.586982	-0.148763	-0.136514	-0.071586	0.040244	0.272	
Dextrin (g)	-0.054191	-0.052969	0.066623	-0.057472	-0.057613	-0.030076	-0.019356	-0.024	
Glycerol (g)	0.019631	0.018946	-0.036372	-0.018845	-0.019030	-0.009565	-0.004252	0.007	
Glycogen (g)	-0.036546	-0.034795	0.037719	0.031451	0.030409	-0.026769	-0.002740	-0.029	

Top 20 correlation result	
Nitrogen \n(g)	0.387011
Protein \n(g)	0.386514
Tryptophan \n(mg)	0.383890
Niacin derived from tryptophan \n(mg)	0.383775
Phosphorus (P) \n(mg)	0.345831
Niacin derived equivalents \n(mg)	0.299194
Zinc (Zn) \n(mg)	0.284724
Alcohol \n(g)	0.275428
C20:4w6 (mg)	0.232458
C10 (%T)	0.231035
Moisture (water) \n(g)	0.218761
Niacin (B3) \n(mg)	0.212431
C8 (%T)	0.207888
Cholesterol \n(mg)	0.192631
Vitamin D3 equivalents \n(ug)	0.191616
C6 (%T)	0.188427
Starch \n(g)	0.187559
C12 (%T)	0.186664
C14 (%T)	0.186258
25-hydroxy cholecalciferol (25-OH D3) \n(ug)	0.184740

Using the 70-30 datasplit to compare F1 score and accuracy by all models for feature selection
(Test data only)

Model	F1- score with all columns	F1-score with top 20 correlation features
Logistic regression	Classification Report for test set, LR, ratio 70-30: <pre> precision recall f1-score support 0 0.94 1.00 0.97 415 1 0.96 0.61 0.75 70 accuracy 0.94 485 macro avg 0.95 0.80 0.86 485 weighted avg 0.94 0.94 0.93 485 </pre>	Classification Report for test set, LR Corr, ratio 70-30: <pre> precision recall f1-score support 0 0.89 1.00 0.94 415 1 1.00 0.26 0.41 70 accuracy 0.89 485 macro avg 0.94 0.63 0.68 485 weighted avg 0.90 0.89 0.86 485 </pre>
KNN-classifier	Classification Report for Test Data, KNN: 70-30: <pre> precision recall f1-score support 0 1.00 1.00 1.00 415 1 1.00 1.00 1.00 70 accuracy 1.00 485 macro avg 1.00 1.00 1.00 485 weighted avg 1.00 1.00 1.00 485 </pre>	Classification Report for Test Data, KNN Corr: 70-30: <pre> precision recall f1-score support 0 1.00 1.00 1.00 415 1 0.99 1.00 0.99 70 accuracy 1.00 485 macro avg 0.99 1.00 1.00 485 weighted avg 1.00 1.00 1.00 485 </pre>
MLP (Relu)	Average Training loss Relu:70-30: 0.3022768996655941 Average Test loss Relu:70-30: 0.5225952535867691 Average Training accuracy Relu:70-30: 0.9651635766029358 Average Test accuracy Relu:70-30: 0.9400000035762787 Average F1 score Relu:70-30: 0.7336911493950578	Average Training loss Relu Corr:70-30: 0.37354654669761655 Average Test loss Relu Corr:70-30: 0.43932426869869234 Average Training accuracy Relu Corr:70-30: 0.9163571894168854 Average Test accuracy Relu Corr:70-30: 0.9008247435092926 Average F1 score: Relu Corr:70-30 0.5038944400708754
MLP (Sigmoid)	Average Training loss Sigmoid:70-30: 0.1754472777247429 Average Test loss Sigmoid:70-30: 0.20879860371351242 Average Training accuracy Sigmoid:70-30: 0.9410256564617157 Average Test accuracy Sigmoid:70-30: 0.9265979349613189 Average F1 score Sigmoid:70-30: 0.6445898850266827	Average Training loss Sigmoid Corr:70-30: 0.23852526694536208 Average Test loss Sigmoid Corr:70-30: 0.25597364008426665 Average Training accuracy Sigmoid Corr:70-30: 0.8929266214370728 Average Test accuracy Sigmoid Corr:70-30: 0.8861855745315552 Average F1 score Sigmoid Corr:70-30: 0.3391902167744864

Model	Accuracy with all columns	Accuracy with top 20 features
Logistic regression	Training loss LR, ratio 70-30: 0.05216622458001763 Test loss LR, ratio 70-30: 0.0597938144329897 Training accuracy LR, ratio 70-30: 0.947833775419982 Test accuracy LR, ratio 70-30: 0.9402061855670103	Training loss LR Corr, ratio 70-30: 0.09991158267020339 Test loss LR Corr, ratio 70-30: 0.10721649484536078 Training accuracy LR Corr, ratio 70-30: 0.9008884173297966 Test accuracy LR Corr, ratio 70-30: 0.8927835051546392
KNN-classifier	Training loss KNN: 70-30: 0.0 Test loss KNN: 70-30: 0.0 Training accuracy KNN: 70-30: 1.0 Test accuracy KNN: 70-30: 1.0	Training loss KNN Corr, 70-30: 0.0 Test loss KNN Corr, 70-30: 0.0020618556701030855 Training accuracy KNN Corr, 70-30: 1.0 Test accuracy KNN Corr, 70-30: 0.9979381443298969
MLP (Relu)	Average Training loss Relu:70-30: 0.3022768996655941 Average Test loss Relu:70-30: 0.5225952535867691 Average Training accuracy Relu:70-30: 0.9651635766029358 Average Test accuracy Relu:70-30: 0.9400000035762787 Average F1 score Relu:70-30: 0.7336911493950578	Average Training loss Relu Corr:70-30: 0.37354654669761655 Average Test loss Relu Corr:70-30: 0.43932426869869234 Average Training accuracy Relu Corr:70-30: 0.9163571894168854 Average Test accuracy Relu Corr:70-30: 0.9008247435092926 Average F1 score: Relu Corr:70-30 0.5038944400708754

MLP (Sigmoid)	Average Training loss Relu:70-30: 0.3022768996655941 Average Test loss Relu:70-30: 0.5225952535867691 Average Training accuracy Relu:70-30: 0.9651635766029358 Average Test accuracy Relu:70-30: 0.9400000035762787 Average F1 score Relu:70-30: 0.7336911493950578	Average Training loss Sigmoid Corr:70-30: 0.23852526694536208 Average Test loss Sigmoid Corr:70-30: 0.25597364008426665 Average Training accuracy Sigmoid Corr:70-30: 0.8929266214370728 Average Test accuracy Sigmoid Corr:70-30: 0.8861855745315552 Average F1 score Sigmoid Corr:70-30: 0.3391902167744864
---------------	---	---

Observation and explanation:

From the results from each of the models used, for both accuracy and f1 score, using the full data columns would give a better classification result between solid and liquid. This shows that there other columns that are not included in the top 20 correlation does give us useful information to classify. The reason why i only show the test data, is to show the result of prediction and if the score is high for test , means that it is high on the train data too.

Results from KNN-Classifier (K value)

I expect that the K(hyperparameter) value should be pretty low due to data imbalance and having so many features (for both top 20 correlation and the entire columns) .

Finding the best hyper-parameter, I used the cross validation on the training dataset to find the best hyperparameter , changing my score from range from 1 to 20 . Comparing the accuracy to the hyperparameter. From the result graph below, it shows that the best K value will be 1 though the accuracy throughout all values from 1 - 20 are 0.9 and above.

Reason for cross validation: it gives a more optimal hyperparameter. This is because the evaluation is more reliable by mitigating the impact of data variability. It also reduce the dependence of a single train-test split. It also uterlize the available data and provide a fair assessment.

Reason i did cross validation on training dataset: it is to tune the model's hyperparameter and estimate its performance on unseen training data. The test dataset is then use for an unbiased evaluation. The Separation helps to prevent overfitting to the test set and ensures a more realistic assessment of the model performance

Reason for K = 1 being the best value: This maybe due to the fact that there are 20 values (feature selection) for 'X' in the model.

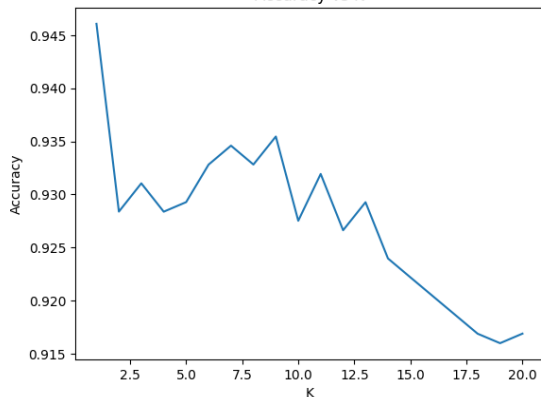
Due to the curse of dimensionality , where the more dimensions there are, the effectiveness and performance get impact negatively by the number of features selected. This is because the data points become increasingly sparse in the feature space which impacts the model classification algorithm to find meaningful patterns and relationship to make correct predictions.

Model	Best hyperparameter with full dataset	Best hyperparameter with top 20 features
KNN-classification	- It takes too long for tuning on all columns	K = 1

Accuracy vs K (hyperparameter)

Average: 0.9460720385033381
Standard deviation: 0.023545954859757592
Average: 0.9283884489986025
Standard deviation: 0.019137053896047508
Average: 0.9310510790249962
Standard deviation: 0.022251131516196693
Average: 0.9283806862288465
Standard deviation: 0.01916873548068842
Average: 0.9292811675205714
Standard deviation: 0.021614679305036674
Average: 0.9328132277596646
Standard deviation: 0.01729280457404331
Average: 0.934598664803602
Standard deviation: 0.020124454709082817
Average: 0.9328209905294209
Standard deviation: 0.017247756411514215
Average: 0.9354603322465456
Standard deviation: 0.017700483690208206
Average: 0.927526781555659
Standard deviation: 0.017877414369721012
Average: 0.9319360347772084
Standard deviation: 0.017633377129842774
Average: 0.9266340630336904
Standard deviation: 0.01760757897660069
Average: 0.929273404750815
Standard deviation: 0.015792856652868444
Average: 0.9239791957770531
Standard deviation: 0.016778075302126494
Average: 0.9222015215028723
Standard deviation: 0.01756506735800204
Average: 0.9204316099984474
Standard deviation: 0.021281573469597668
Average: 0.9186616984940226
Standard deviation: 0.021940224206996952
Average: 0.9168917869895978
Standard deviation: 0.019847815951015504
Average: 0.9159990684676291
Standard deviation: 0.020675520455586002
Average: 0.9168995497593541
Standard deviation: 0.02096049179008905

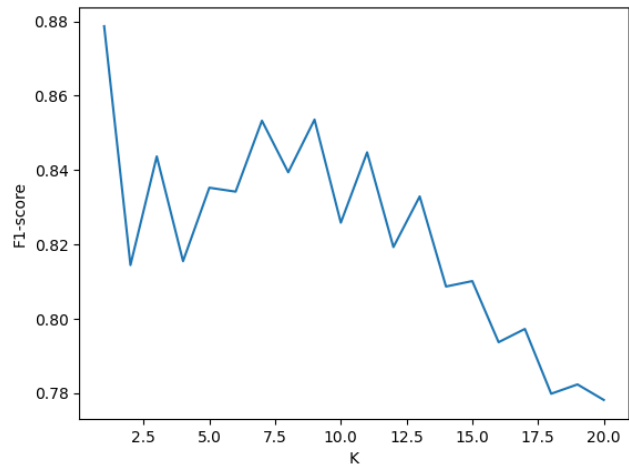
Accuracy vs K



F1-score vs K (hyperparameter)

F1 score for k = 1 : 0.8787173124447356
F1 score for k = 2 : 0.814453693769169
F1 score for k = 3 : 0.8436867181119629
F1 score for k = 4 : 0.8155004482562127
F1 score for k = 5 : 0.8352632434388054
F1 score for k = 6 : 0.8342009456840499
F1 score for k = 7 : 0.8532975777740189
F1 score for k = 8 : 0.8394106861673555
F1 score for k = 9 : 0.8535872074984938
F1 score for k = 10 : 0.8258555525049969
F1 score for k = 11 : 0.8447756388690613
F1 score for k = 12 : 0.8192927744193644
F1 score for k = 13 : 0.8329219930820635
F1 score for k = 14 : 0.8086815372622919
F1 score for k = 15 : 0.8101416218555583
F1 score for k = 16 : 0.7936995143958347
F1 score for k = 17 : 0.7972693643157105
F1 score for k = 18 : 0.7798356129610718
F1 score for k = 19 : 0.7823412923766643
F1 score for k = 20 : 0.778164201444545

F1-score vs K



Observation for accuracy vs K and F1 score vs K:

From the graph and numerical values , both accuracy and f1 score decreases as K value increases. This is due to the of the number of features, curse of dimensionality as stated above.

Conclusion for feature selection:

Having all features gives better accuracy score and F1-score which shows that other columns not in the top 20 correlation do give important bits of information in terms of. However, the top 20 correlated features score is extremely close to the scores of full features.

KNN-Classifier (MinMaxScalar)

In my KNN-classifier model and finding K , i use the MinMaxScalar for the model instead of standard normalization. This is because the data number doesnt fit normal distribution. Not just that but Min-Max helps to preserve the original range of the data and maintain the relative relationships between data points which is good for KNN. Another reason why i use it is because in the dataset, there are non-linear relationships between the features and Minmaxscalar allows non-linear relationships to be maintain and ensures that the transformed features remains interpretabe in their original scale.

Confusion matrix:

Reason i use confusion matrix: it is easier to see what prediction did the model between solid and liquid :

- True Positive: top left , model predicted correctly for solid
- True Negative: top right , model predicted liquid when it is suppose to be a solid
- False Positive: bottom right , model predicted correctly for liquid
- False Negatie: bottom left, model predicted solid when it is suppose to be a liquid

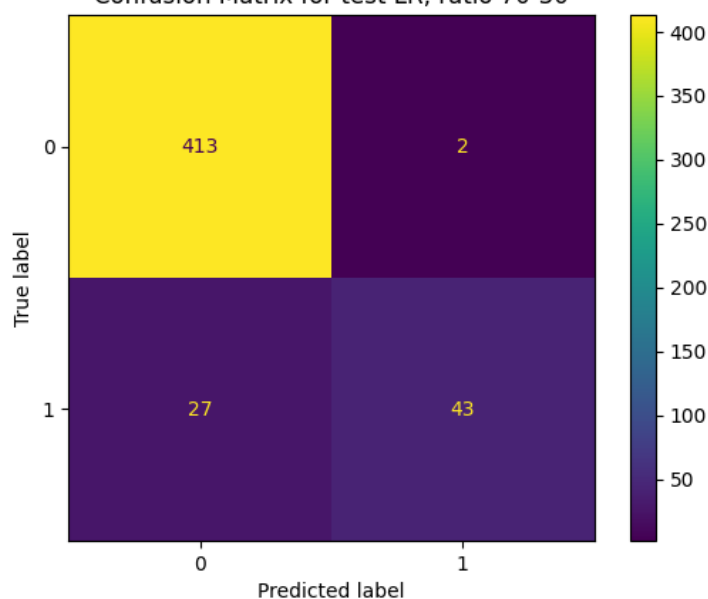
Logistic Regression expectation:

I do expect high accuracy for both all column and top 20 correlation though all column should have better prediction as it has more features which provide informations to work with.

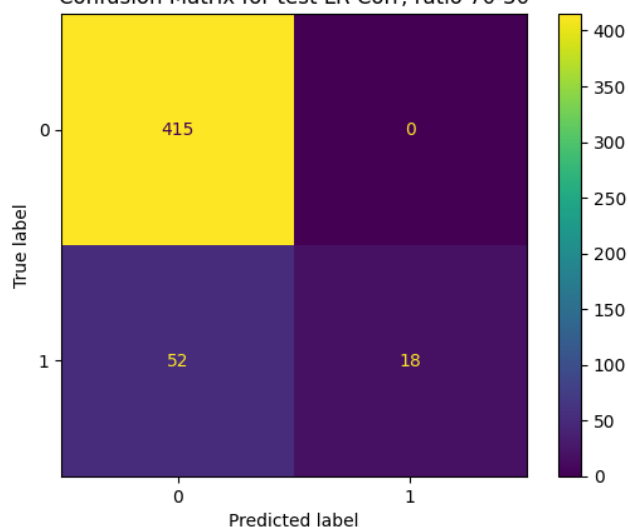
Logistic Regression results:

All columns	Top 20 correlation
<pre>Classification Report for test set, LR, ratio 70-30: precision recall f1-score support 0 0.94 1.00 0.97 415 1 0.96 0.61 0.75 70 accuracy 0.94 485 macro avg 0.95 0.80 0.86 485 weighted avg 0.94 0.94 0.93 485 Training loss LR, ratio 70-30: 0.05216622458001763 Test loss LR, ratio 70-30: 0.0597938144329897 Training accuracy LR, ratio 70-30: 0.9478337754199824 Test accuracy LR, ratio 70-30: 0.9402061855670103</pre>	<pre>Training loss LR Corr, ratio 70-30: 0.09991158267020339 Test loss LR Corr, ratio 70-30: 0.10721649484536078 Training accuracy LR Corr, ratio 70-30: 0.9000884173297960 Test accuracy LR Corr, ratio 70-30: 0.8927835051546392 Classification Report for test set, LR Corr, ratio 70-30: precision recall f1-score support 0 0.89 1.00 0.94 415 1 1.00 0.26 0.41 70 accuracy 0.89 485 macro avg 0.94 0.63 0.68 485 weighted avg 0.90 0.89 0.86 485</pre>

Confusion Matrix for test LR, ratio 70-30



Confusion Matrix for test LR Corr, ratio 70-30



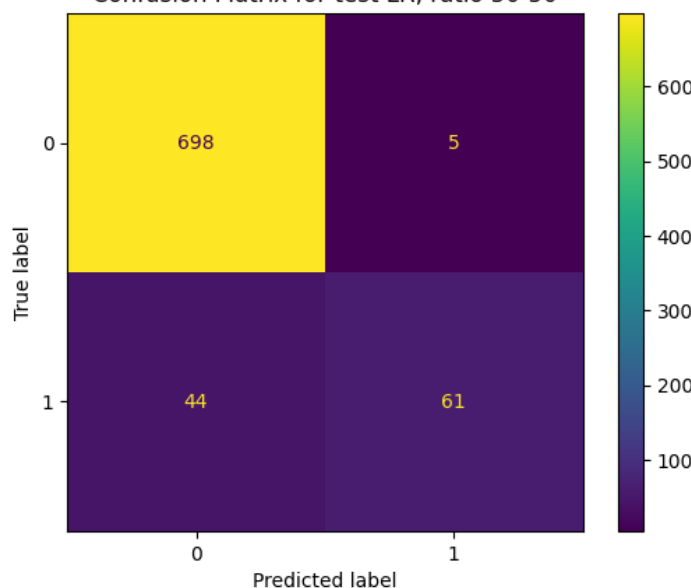
Training loss LR, ratio 50-50: 0.05074257425742579
 Test loss LR, ratio 50-50: 0.0606435643564357
 Training accuracy LR, ratio 50-50: 0.9492574257425742
 Test accuracy LR, ratio 50-50: 0.9393564356435643

Classification Report for test set, LR, ratio 50-50:

	precision	recall	f1-score	support
0	0.94	0.99	0.97	703
1	0.92	0.58	0.71	105

accuracy			0.94	808
macro avg	0.93	0.79	0.84	808
weighted avg	0.94	0.94	0.93	808

Confusion Matrix for test LR, ratio 50-50



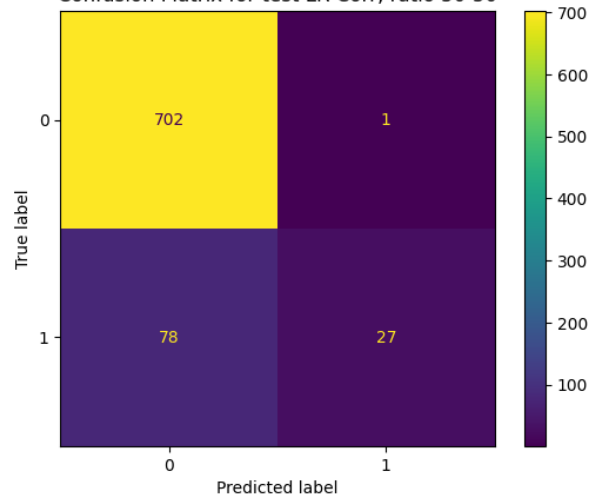
Training loss LR Corr, ratio 50-50: 0.10643564356435642
 Test loss LR Corr, ratio 50-50: 0.09777227722722725
 Training accuracy LR Corr, ratio 50-50: 0.8935643564356436
 Test accuracy LR Corr, ratio 50-50: 0.9022277227722773

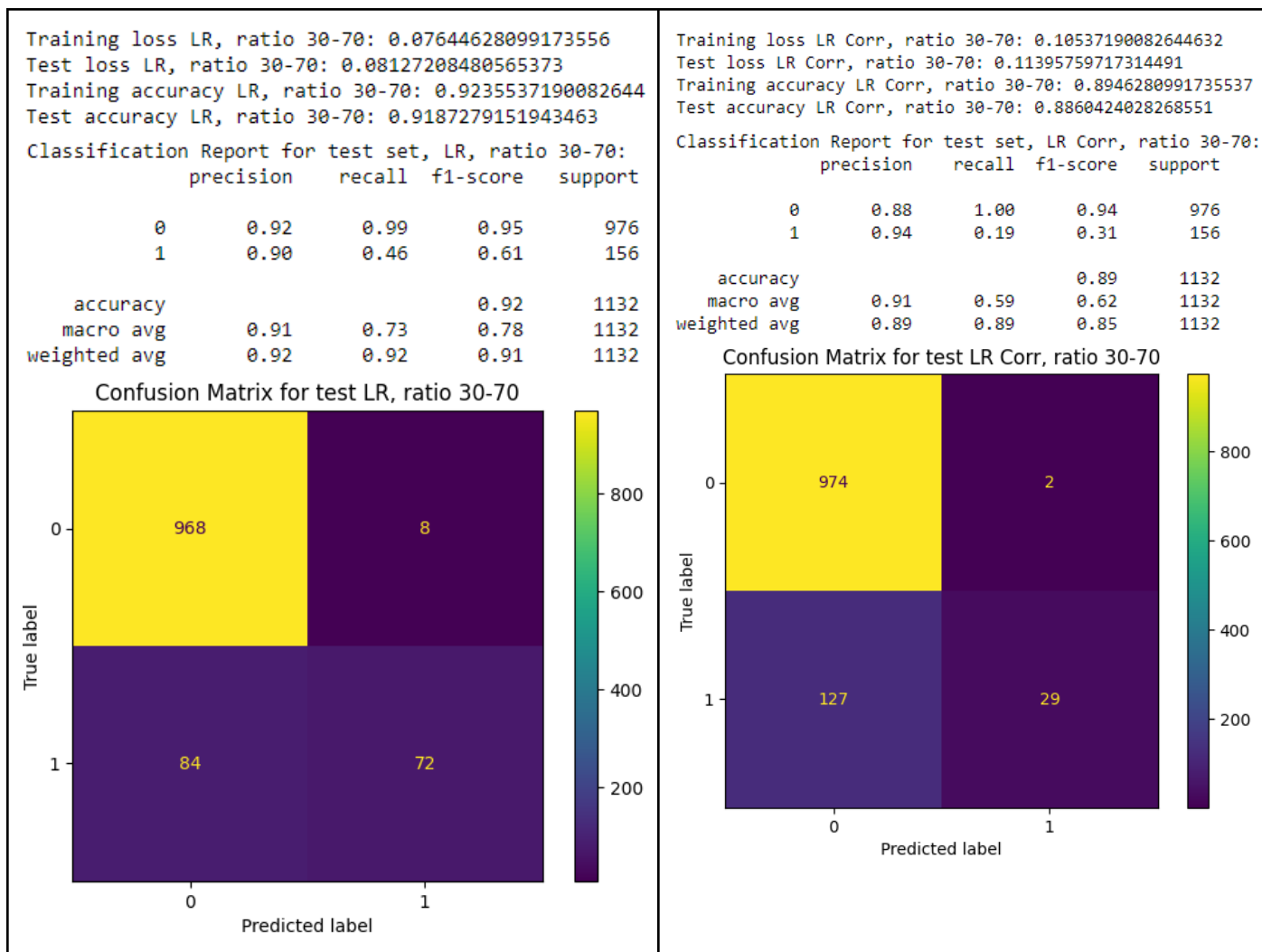
Classification Report for test set, LR Corr, ratio 50-50:

	precision	recall	f1-score	support
0	0.90	1.00	0.95	703
1	0.96	0.26	0.41	105

accuracy			0.90	808
macro avg	0.93	0.63	0.68	808
weighted avg	0.91	0.90	0.88	808

Confusion Matrix for test LR Corr, ratio 50-50





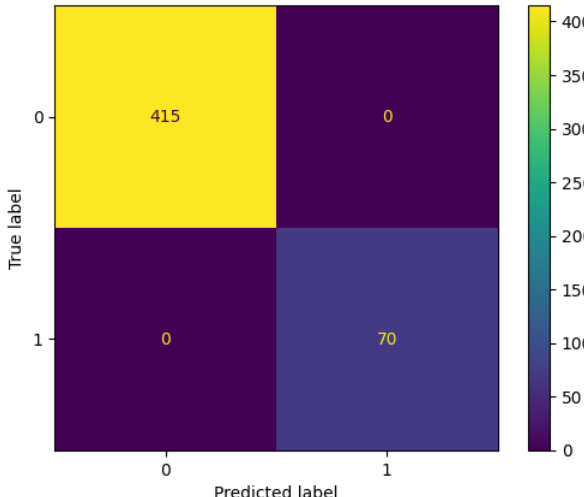
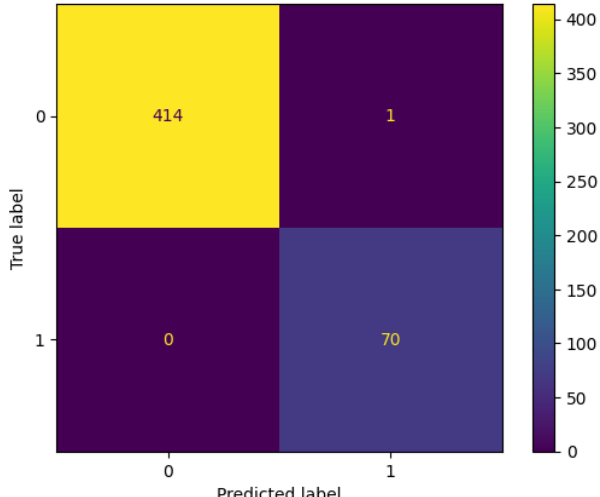
Observation and reason for Logistic Regression:

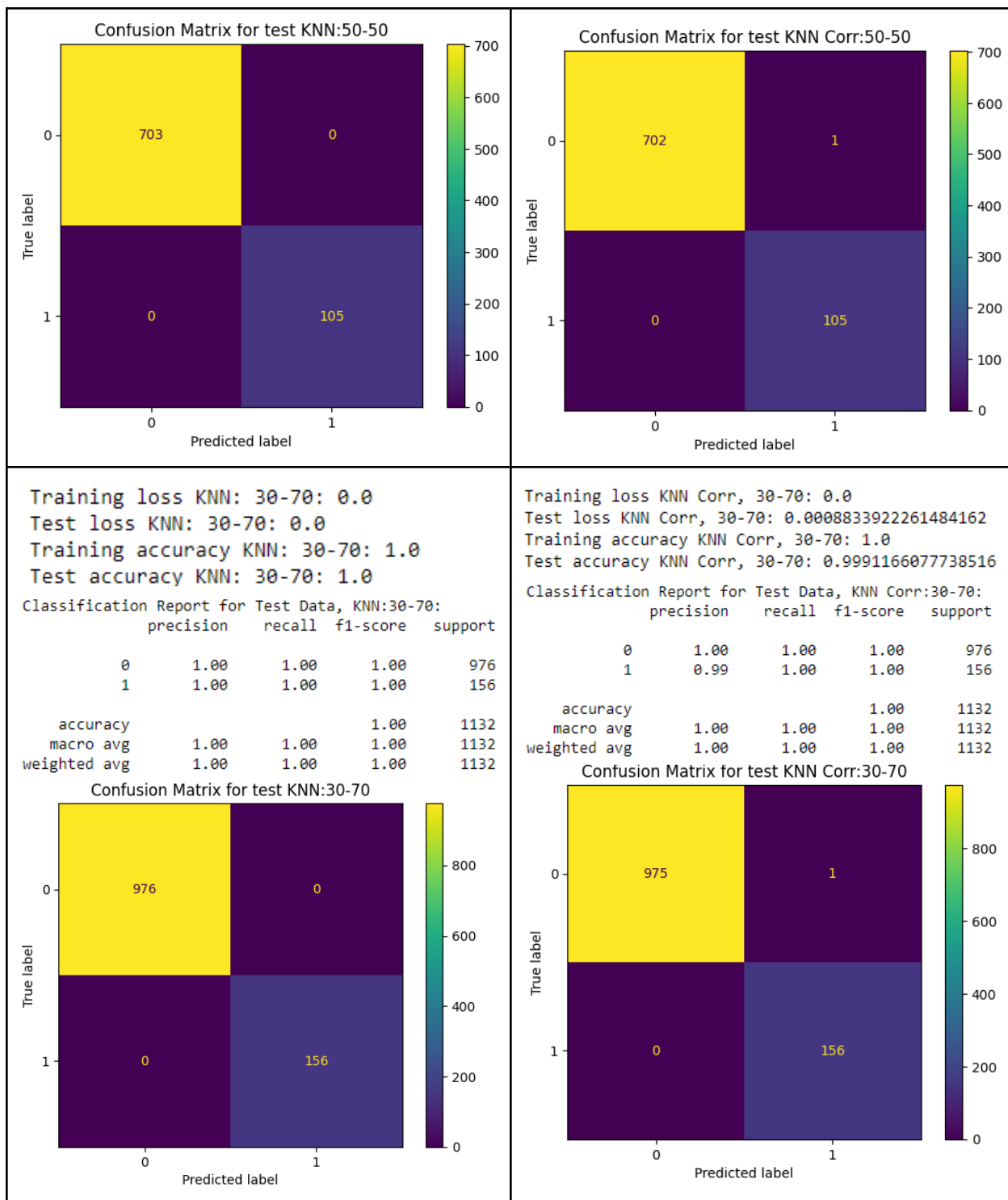
For logistic regression, the f1 and accuracy score is pretty high, estimating at 0.8-0.95 it perform. Although the 1 with all columns is has better prediction since there are useful information in other columns of the data outside of the Top 20 correlation. When splitting the data from 70-30 to 30-70 , the f1 score and accuracy is reduced, this means that the model does not have enough training. While 70-30 and 50-50 remains almost the same, which mean changing the value between them will give us almost the same result due it the model knowing the patterns and relationship to predict correctly.

KNN expectation:

I expect the result to be better than Logistic regression

KNN-Classifer results:

All columns	Top 20 correlation																																																												
<p>Training loss KNN: 70-30: 0.0</p> <p>Test loss KNN: 70-30: 0.0</p> <p>Training accuracy KNN: 70-30: 1.0</p> <p>Test accuracy KNN: 70-30: 1.0</p> <p>Classification Report for Test Data, KNN: 70-30:</p> <table><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr><tr><td>0</td><td>1.00</td><td>1.00</td><td>1.00</td><td>415</td></tr><tr><td>1</td><td>1.00</td><td>1.00</td><td>1.00</td><td>70</td></tr><tr><td>accuracy</td><td></td><td></td><td>1.00</td><td>485</td></tr><tr><td>macro avg</td><td>1.00</td><td>1.00</td><td>1.00</td><td>485</td></tr><tr><td>weighted avg</td><td>1.00</td><td>1.00</td><td>1.00</td><td>485</td></tr></table> <p>Confusion Matrix for test KNN:70-30</p> 		precision	recall	f1-score	support	0	1.00	1.00	1.00	415	1	1.00	1.00	1.00	70	accuracy			1.00	485	macro avg	1.00	1.00	1.00	485	weighted avg	1.00	1.00	1.00	485	<p>Training loss KNN Corr, 70-30: 0.0</p> <p>Test loss KNN Corr, 70-30: 0.0020618556701030855</p> <p>Training accuracy KNN Corr, 70-30: 1.0</p> <p>Test accuracy KNN Corr, 70-30: 0.9979381443298969</p> <p>Classification Report for Test Data, KNN Corr: 70-30:</p> <table><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr><tr><td>0</td><td>1.00</td><td>1.00</td><td>1.00</td><td>415</td></tr><tr><td>1</td><td>0.99</td><td>1.00</td><td>0.99</td><td>70</td></tr><tr><td>accuracy</td><td></td><td></td><td>1.00</td><td>485</td></tr><tr><td>macro avg</td><td>0.99</td><td>1.00</td><td>1.00</td><td>485</td></tr><tr><td>weighted avg</td><td>1.00</td><td>1.00</td><td>1.00</td><td>485</td></tr></table> <p>Confusion Matrix for test KNN Corr:70-30</p> 		precision	recall	f1-score	support	0	1.00	1.00	1.00	415	1	0.99	1.00	0.99	70	accuracy			1.00	485	macro avg	0.99	1.00	1.00	485	weighted avg	1.00	1.00	1.00	485
	precision	recall	f1-score	support																																																									
0	1.00	1.00	1.00	415																																																									
1	1.00	1.00	1.00	70																																																									
accuracy			1.00	485																																																									
macro avg	1.00	1.00	1.00	485																																																									
weighted avg	1.00	1.00	1.00	485																																																									
	precision	recall	f1-score	support																																																									
0	1.00	1.00	1.00	415																																																									
1	0.99	1.00	0.99	70																																																									
accuracy			1.00	485																																																									
macro avg	0.99	1.00	1.00	485																																																									
weighted avg	1.00	1.00	1.00	485																																																									
<p>Training loss KNN: 50-50: 0.0</p> <p>Test loss KNN: 50-50: 0.0</p> <p>Training accuracy KNN: 50-50: 1.0</p> <p>Test accuracy KNN: 50-50: 1.0</p> <p>Classification Report for Test Data, KNN: 50-50:</p> <table><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr><tr><td>0</td><td>1.00</td><td>1.00</td><td>1.00</td><td>703</td></tr><tr><td>1</td><td>1.00</td><td>1.00</td><td>1.00</td><td>105</td></tr><tr><td>accuracy</td><td></td><td></td><td>1.00</td><td>808</td></tr><tr><td>macro avg</td><td>1.00</td><td>1.00</td><td>1.00</td><td>808</td></tr><tr><td>weighted avg</td><td>1.00</td><td>1.00</td><td>1.00</td><td>808</td></tr></table>		precision	recall	f1-score	support	0	1.00	1.00	1.00	703	1	1.00	1.00	1.00	105	accuracy			1.00	808	macro avg	1.00	1.00	1.00	808	weighted avg	1.00	1.00	1.00	808	<p>Training loss KNN Corr, 50-50: 0.0</p> <p>Test loss KNN Corr, 50-50: 0.0012376237623762387</p> <p>Training accuracy KNN Corr, 50-50: 1.0</p> <p>Test accuracy KNN Corr, 50-50: 0.9987623762376238</p> <p>Classification Report for Test Data, KNN Corr: 50-50:</p> <table><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr><tr><td>0</td><td>1.00</td><td>1.00</td><td>1.00</td><td>703</td></tr><tr><td>1</td><td>0.99</td><td>1.00</td><td>1.00</td><td>105</td></tr><tr><td>accuracy</td><td></td><td></td><td>1.00</td><td>808</td></tr><tr><td>macro avg</td><td>1.00</td><td>1.00</td><td>1.00</td><td>808</td></tr><tr><td>weighted avg</td><td>1.00</td><td>1.00</td><td>1.00</td><td>808</td></tr></table>		precision	recall	f1-score	support	0	1.00	1.00	1.00	703	1	0.99	1.00	1.00	105	accuracy			1.00	808	macro avg	1.00	1.00	1.00	808	weighted avg	1.00	1.00	1.00	808
	precision	recall	f1-score	support																																																									
0	1.00	1.00	1.00	703																																																									
1	1.00	1.00	1.00	105																																																									
accuracy			1.00	808																																																									
macro avg	1.00	1.00	1.00	808																																																									
weighted avg	1.00	1.00	1.00	808																																																									
	precision	recall	f1-score	support																																																									
0	1.00	1.00	1.00	703																																																									
1	0.99	1.00	1.00	105																																																									
accuracy			1.00	808																																																									
macro avg	1.00	1.00	1.00	808																																																									
weighted avg	1.00	1.00	1.00	808																																																									



Observation and reason for KNN-classifier

When using KNN, regardless of how it is split or using all the features, the result for both the f1 score and the accuracy is almost perfect. However, all columns is able to predict that 1 outlier prediction correctly by the top 20 column due to the addition features it has that helped the model to predict it correctly. And why the prediction is so accurate is because $K=1$

Neuron Network

Default setting

```
model = Sequential()
model.add(Dense(16, activation='relu', input_shape=(X7_train_scaled.shape[1],)))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='relu'))

# Compile the model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Fit the model to the training data
history = model.fit(X7_train_scaled, y7_train, epochs=50, batch_size=32, validation_split=0.2, verbose=0)
```

Reason sequential:

I chose sequential is because it allows me to create a hierarchy of layers for the model to learn and extract more complex details between the features. Each layer can capture different patterns which allows the model to make better decision in terms of classification.

Reason for dense:

Why i put dense as multiple of 8 and 3 layers is because i didnt not want to make the layers too small but still allow the model to learn more about the relationship between the features while reducing the risk of overfitting.

Reason binary_crossentropy:

It is well suited for models that output probabilities for each class, it measures the dissimilarity between the predicted probabilities and the true labels for binary classification task which allows the model to predict better

Reason for adam:

I picked adam due to the non-linear relationship and the sparse of data (curse of dimensionality). Adam helps adjusts the learning rate dynamically for each parameter which helps the model in converging faster and handling sparse gradients efficiently. Allowing better prediction

Reason for accuracy:

It provides straightforward measure of the model's performance. Which does not add to the complexity of the model

.

Reason for loop:

It helps to reduce overfitting. By exposing it to the training data repeatedly , the average of the results can help to average out any overfitting which product better result score.

Expected observation:

I do expect the result to be better than Logistic regression, but worse or equal to KNN results

Results for all column Relu:

Average Training loss Relu:70-30: 0.3022768996655941
Average Test loss Relu:70-30: 0.5225952535867691
Average Training accuracy Relu:70-30: 0.9651635766029358
Average Test accuracy Relu:70-30: 0.9400000035762787
Average F1 score Relu:70-30: 0.7336911493950578

Average Training loss Relu:50-50: 0.5510515570640564
Average Test loss Relu:50-50: 0.6225139737129212
Average Training accuracy Relu:50-50: 0.9470296978950501
Average Test accuracy Relu:50-50: 0.9358910918235779
Average F1 score Relu:50-50: 0.6495767944262509

Average Training loss Relu:30-70: 0.7170746810734272
Average Test loss Relu:30-70: 0.9035396695137023
Average Training accuracy Relu:30-70: 0.9361570179462433
Average Test accuracy Relu:30-70: 0.9146643102169036
Average F1 score Relu:30-70: 0.5432905976741307

Results for all column Sigmoid:

Average Training loss Sigmoid:70-30: 0.1754472777247429
Average Test loss Sigmoid:70-30: 0.20879860371351242
Average Training accuracy Sigmoid:70-30: 0.9410256564617157
Average Test accuracy Sigmoid:70-30: 0.9265979349613189
Average F1 score Sigmoid:70-30: 0.6445898850266827

Average Training loss Sigmoid:50-50: 0.19747358411550522
Average Test loss Sigmoid:50-50: 0.20788883566856384
Average Training accuracy Sigmoid:50-50: 0.9231435716152191
Average Test accuracy Sigmoid:50-50: 0.9207920789718628
Average F1 score Sigmoid:50-50: 0.5637586336922578

Average Training loss Sigmoid:30-70: 0.31826300323009493
Average Test loss Sigmoid:30-70: 0.3269763305783272
Average Training accuracy Sigmoid:30-70: 0.8706611394882202
Average Test accuracy Sigmoid:30-70: 0.8668727695941925
Average F1 score Sigmoid:30-70: 0.06525795408093746

Result for top 20 correlation Relu:

Average Training loss Relu Corr:70-30: 0.37354654669761655
Average Test loss Relu Corr:70-30: 0.43932426869869234
Average Training accuracy Relu Corr:70-30: 0.9163571894168854
Average Test accuracy Relu Corr:70-30: 0.9008247435092926
Average F1 score: Relu Corr:70-30 0.5038944400708754

Average Training loss Relu Corr 50-50: 0.5935872465372085
Average Test loss Relu Corr 50-50: 0.5592303737998009
Average Training accuracy Relu Corr 50-50: 0.9017326712608338
Average Test accuracy Relu Corr 50-50: 0.9054455399513245
Average F1 score Relu Corr 50-50: 0.4275830353225058

Average Training loss Relu Corr 30-70: 0.7579659387469292
Average Test loss Relu Corr 30-70: 0.8040758892893791
Average Training accuracy Relu Corr 30-70: 0.8913223147392273
Average Test accuracy Relu Corr 30-70: 0.8872791409492493
Average F1 score Relu Corr 30-70: 0.3139362069158713

Result for top 20 correlation Sigmoid:

```
Average Training loss Sigmoid Corr:70-30: 0.23852526694536208
Average Test loss Sigmoid Corr:70-30: 0.25597364008426665
Average Training accuracy Sigmoid Corr:70-30: 0.8929266214370728
Average Test accuracy Sigmoid Corr:70-30: 0.8861855745315552
Average F1 score Sigmoid Corr:70-30: 0.3391902167744864

Average Training loss Sigmoid Corr:50-50: 0.2737825185060501
Average Test loss Sigmoid Corr:50-50: 0.26418707370758054
Average Training accuracy Sigmoid Corr:50-50: 0.8736386120319366
Average Test accuracy Sigmoid Corr:50-50: 0.8830445528030395
Average F1 score Sigmoid Corr:50-50: 0.18505123807137372

Average Training loss Sigmoid Corr:30-70: 0.34952048063278196
Average Test loss Sigmoid Corr:30-70: 0.35058501064777375
Average Training accuracy Sigmoid Corr:30-70: 0.8657024502754211
Average Test accuracy Sigmoid Corr:30-70: 0.8628091633319854
Average F1 score Sigmoid Corr:30-70: 0.008588957055214723
```

Observation and explanation:

Surprisingly, the MLP model (Relu and Sigmoid) f1 score is lower than both KNN and Logistic Regression model. This maybe due to the fact that MLP has a difficulty generalizing and ended up overfitting. Data imbalancing may cause the result to be worse as it makes learning for the MLP alot harder. With so many different features, scaling becomes more complex while KNN and Logistic Regression can handle high-dimensional data better. Ways to improve the MLP is listed down below after conclusion. Also regard of split , for both Relu and sigmoid , the lesser data being use as train , the worse the accuracy

Another observation seen is that for both feature situation and regardless of split, the accuracy is less than 0.5-0.6 which is not good. For sigmoid in top 20 correlation , the f1-score and accuracy is much lower till less than 0.2 due to number of features which doesnt give much information to the model, hence the prediction is worse a way to improve this is to increase the dense size from [1,8,16] to [10,20,30] as listed below.

Conclusion:

In conclusion, for this problem objectivemain issue for the problem objective was high dimensionality and data imbalancing, KNN is the best model to use base on what i have done. The result was also surprising as i had expected that the MLPs model would do better than Logistic Regression but it didnt. However, there are ways to improve the MLPs used in this project to increase the f1-score and accuracy listed below.

Further improvement for result in this project:

- 1) Another way is by doing PCA, however by the time of learning PCA , i do not have enough time to complete the coding the PCA feature, comparing the results with what i have originally done with feature selection.
- 2) For Relu and Sigmoid component , change the batch size/ increase number of dense (instead of having 3 numbers , add more like [1,4,8,12,16] , increasing the numbers like [10,20,30]) and see the result.
- 3) Using more Machine Learning models such as Decision Tree to compare the results
- 4) Add it new liquid dataset and test the model to see how well the models used in this assignment predicts new unseen data.
- 5) Changing the MLP model from sequential to convolution neural networks (CNN) or other models