CS 2810 Final Project Documentation

Authors : Xiangxin Ji, Xiaofei Xie

CS 2810 Final Project Documentation

I. Algorithm

For this project, we choose to implement two supervised data mining algorithm, the K-Nearest-Neighbor algorithm and the Gaussian-Naive-Bayes algorithm, to predict the class of the items in the unclassified data.

For KNN algorithm, we first split the provided training dataset into a training portion and a testing portion in a ratio of 75% : 25%. Then, we find the most effective k, which predicts the testing portion data with the highest accuracy. Finally, the most effective k will be applied in the KNN algorithm and make predictions on the targeting file. The final output will be a csv file with applied features and the predicted label column.

For the Gaussian Naive Bayes algorithm, we first split the provided training dataset into a training portion and a testing portion in a ratio of 75% : 25%. Then, we apply the Gaussian Naive Bayes algorithm on the training and testing to make prediction and calculate the prediction accuracy. The accuracy result gives us a clue about how well our algorithm performed in the provided dataset, and it is represented as percentage. Finally, we applied this algorithm to the targeting file and make a prediction. The final output will be a csv file with applied features and the predicted label column.

II. Dataset

The chosen supervised datasets are listed above, please refer to Appendix: Dataset Details for detailed context and attribute information:

1. Iris Data Set (lable: Species)
2. Student Alcohol Consumption (lable: Walc/Dalc)
3. fruit (lable: fruit_name)
4. Heart Failure Prediction Dataset (lable: HeartDisease)
5. Drug Classification (lable: Drug)
6. Mushroom Classification (lable: class - edible VS poisonous)

- Note: we shuffled and split the original data into two subsets: data for training, and data for comparing; they are in `./training_data`, and `./sample_data` respectively. Data under the `./target_data` is just copies of the sample data with the label column removed, for running the prediction algorithm.

III.Code Design

Data Preprocessing

`KNNTrainRun` and `GNBTrainRun` share a portion of common code, which is for data preprocessing. We don't abstract the common part out because the rest of the code in each method depends on the value generated by this process, and if we put these values as arguments for the algorithm core, the signature would be too giant to read.

- The first step of data preprocessing is to verify whether users provided valid label, training file, targeting file, as well as the optional selected attributes.

```python
### Verify section
trainingDF = varifyTraining(label, trainingFile)
trainingAttr = ([col for col in list(trainingDF.columns) if not col.startswith("Unnamed")]
    if len(selectedAttr) == 0 else selectedAttr + [label])
try:
    checkValidAttr(trainingDF, trainingAttr)
except AttributeError as e:
    raise AttributeError(str(e) + " in training file!")


targetingDF = varifyTargeting(label, targetingFile)
targetingAttr = [attr for attr in trainingAttr if attr != label]
try:
    checkValidAttr(targetingDF, targetingAttr)
except AttributeError as e:
    raise AttributeError(str(e) + " in targeting file!")
### End verify section
```

- Then, we convert every string columns to integer columns, by labeling string entry with the corresponding integer tag.

```python
### String labeling section
attributeNominalMaps = {}
for attribute in trainingAttr:
    if(trainingDF[attribute].dtype == "object" or attribute == label):
        attributeNominalMaps[attribute] = (quantCol(trainingDF, attribute))
labelNominal = attributeNominalMaps[label]
labelDict = dict([reversed(i) for i in labelNominal.items()])
### End string labeling section
```

KNN

The first step of the KNN prediction (apart from the data preprocessing) is to find the most effective k to maximize the accuracy for predicting the testing portion of the training data, we implement this method for that desire:

```python
def classify(classified, unclassified, k):
    '''
    classify a list of objects using knn algorithm, based on the classified data and the provided k.

    Parameters
    ----------
        classified : [(int, numpy.array)]
            a list of tuples representing a list of classified objects, the first item of a tuple
            represents the integer tag of the object, the second item represents its feature vector

        unclassified : [numpy.array]
            a list of vectors representing unclassifed objects
        k : int
            the number of the nearest neighbors to find

    Returns
    -------
        out : [ [(int, numpy.array)] ]
            generate a list contaning lists of candidates (k-neighbors) for every unclassified item.
             The count of the candidates depends on the k.
    '''
    result = []
    for target in unclassified:
        neighbors = [(None, float('inf'))] * k
        for classifiedNode in classified:
            maxValIndex = neighbors.index(max(neighbors, key = lambda neighbor : neighbor[1]))
            currDistance = distance(target, classifiedNode[1])
            if(currDistance < neighbors[maxValIndex][1]):
                neighbors[maxValIndex] = (classifiedNode[0], currDistance)
        result.append(neighbors)
    return result

def findKAccuracyPair(training, testing, upperLimit, step):
    '''
    to find a k that maximize the prediction accuracy for the testing data, provided the training
    reference.

    Parameters
    ----------
        training : [(int, numpy.array)]
            a list of tuples representing a list of training objects, the first item of a tuple
            represents the integer tag of the object, the second item represents its feature vector
        testing : [(int, numpy.array)]
            a list of tuples representing a list of testing objects, the first item of a tuple
            represents the integer tag of the object, the second item represents its feature vector
        upperLimit : int
            the upper limit for k
        step : int
            the increase step for k-finding process, the step depends on the possible values for the
            labels, to avoid tie situations

    Returns
    -------
        out : (int, float)
            the k-accuracy pair with the k value maximize the prediction accuracy for the testing
            data
    '''
    kAccuracyPair = (-1, -1)
    expectLabels = getLabels(testing)
    for i in range(1,upperLimit + 1, step):
        actualLabels = list(map(lambda knn : mostFreq(getLabels(knn)),
            classify(training, getVectors(testing), i)))
        currAccuracy = compSimilarity(expectLabels, actualLabels)
        if (currAccuracy > kAccuracyPair[1]):
            kAccuracyPair = (i, currAccuracy)
    return kAccuracyPair
```

```

```

The fundamental idea for `findKAccuracyPair` is it repetitively call `classify` with possible k's (i.e. the k within the bound between 1 and an appropriate upper limit; and the k would not generate tie situations). For every k, it compares the actual label list generated by the `classify` with the expected list, and update the k-accuracy pair if the k reaches a higher accuracy. Eventually we will get a k which predicts the testing data with the highest accuracy.

We then use that k to predict the targetVectors:

```
classify(trainingSub, targetVectors, k)
```

and map the integer labels to the original class label:

```
list(map(lambda knn : labelDict[mostFreq(getLabels(knn))],
        classify(trainingSub, targetVectors, k)))
```

GNB

We implement the Gaussian Naive Bayes algorithm based on the formula:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

where x_i represents the value of the feature for each item in the targeting dataset, and y represents the feature for the known classes. The probability of one target item belongs to a particular class is the probability of the appearance of that classes multiplied by the z-score distance between the data point and the class-mean. We will find the highest probability among, and give a prediction.

> For example, if we select 3 features width, mass, and height for fruit prediction, with the target has entry (width=7.6, mass=3, height=3.25). The probability of this target belongs to the apple class is
>
> $$\frac{1}{\sqrt{2\pi\sigma_{\text{appleWidth}}^2}} \exp\left(-\frac{(7.6-\mu_{\text{appleWidth}})^2}{2\sigma_{\text{appleWidth}}^2}\right) \times \frac{1}{\sqrt{2\pi\sigma_{\text{appleMass}}^2}} \exp\left(-\frac{(3-\mu_{\text{appleMass}})^2}{2\sigma_{\text{appleMass}}^2}\right) \times \frac{1}{\sqrt{2\pi\sigma_{\text{appleHeight}}^2}} \exp\left(-\frac{(3.25-\mu_{\text{appleHeight}})^2}{2\sigma_{\text{appleHeight}}^2}\right) \times \Pr(\text{apple})$$

- `gnbTrainRun`

  We use this function to predict the expected label by applying Gaussian Naive Bayes algorithm on the provided targeting file.

```
def gnbTrainRun(label, trainingFile, targetingFile, selectedAttr):
    '''
    Predict labels for the targeting file by applying Gaussian Naive Bayes algorithm on the provided
    targeting file.

    Parameters
    ----------
        label : str
            the label which is going to predict
        trainingFile: str
            the file path of the traning file which is used to train Gaussian Naive Baye algorithm
        targetingFile : str
            the file path of the targeting file for prediction
        selectedAttr : [str]
            valid features in the provided dataset which uses to train the machine

    Returns
```

```
     -------
        out : (pandas.core.frame.DataFrame, str)
            the processed targeting dataframe to be saved in the machine, with a result string
            containing the the accuracy when run the Gaussian Naive Bayes algorithm on the
            testing portion of the training data.
    '''
```

1. First, we preprocess the data.

2. Next, we separate the data into a training set and a testing set.

```
trainingAttrTable = trainingDF[trainingAttr].astype("object")
trainingSub = trainingAttrTable.sample(frac=0.75,random_state=200)
testingSub = trainingAttrTable.drop(trainingSub.index)
separated = {}

for row, entry in trainingSub.iterrows():
    if not entry[label] in separated:
        separated[entry[label]] = []
    separated[entry[label]].append((entry[targetingAttr].values))
```

3. Then, we create a summaries dictionary has the structure of:
    - {int : ([float], [float], int)}
        - the key of the dictionary representing each distinct label
        - the value of the dictionary is a tuple consists of
            0. list of mean values for each feature
            1. list of std values for each feature
            2. the size of the entries that belong to this label

```
summaries = {}
for key in separated:
    labelStats = pd.DataFrame(separated[key])
    mean = labelStats.mean().values
    stdev = labelStats.std().values
    if 0 in stdev or np.isnan(stdev).any():
        raise AttributeError("The provided training data is not normally distributted,"\
        +" GNB cannot be performed!")
    sampleSize = len(labelStats)
    summaries[key] = (mean, stdev, sampleSize)
```

4. After that, we calculate the accuracy based on the expected output and actual output.

```
expect = list(testingSub[label].values)
actual = gnbPredict(summaries, testingSub[targetingAttr])
accuracy = compSimilarity(expect, actual)
```

5. Finally, we create the targeting data frame and make a prediction. This function will return a tuple of the processed targeting dataframe to be saved in the machine, and the result string containing the accuracy information.

```python
targetDataFrame = genTargetDataFrame(targetingDF, targetingAttr, attributeNominalMaps)
targetingDF[label] = list(map(lambda x : labelDict[x], gnbPredict(summaries,targetDataFrame)))
resultString = "GNB prediction result has been saved to {filePath}"\
    + " (with an accuracy that correctly predict " + str(round((accuracy * 100), 2)) \
    + "% of the testing data)"
return (targetingDF, resultString)
```

- `gnb`
  - We implement the Gaussian Naive Bayes algorithm based on the formula above.
- `predict`
  - We implement a predict method in order to classify each testing data inside the target dataframe. And since the probability is small, we normalized the result and got the percentage of the probability in 2 decimal.

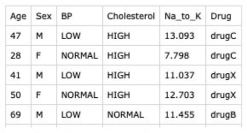
Main

- To run the sample, use the `--sample` as the only argument, it will run the six sample dataset using the two algorithm, and save the result file into `./out/`
- For other dataset, please use the syntax:
  - `$ python3 final.py [predictiing-label] [training-file] [targeting-file] [optional]`, where
    - predicting-label is the label to be predicted, and it should appear as a column name in the training-file
    - both training-file and targeting-file should be csv format files
    - targeting-file either does not have the label column or has an empty column, non-empty label column in the targeting-file would fail the program.
    - optional argument(s) can specify which features should involve for the prediction, and if nothing is specified all features will be count
  - The output will be saved in the `./out/` with the prefix of the algorithm applied.

IV. Result

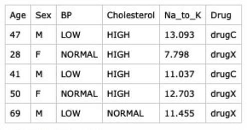
Due to the size of the data, we will only put a small portion of the result files, along with their sample files (for comparison) in this section. For more details, please refer to the result file under the `./out` and the sample file under the `./sample_data`

1. `$ python3 final.py Drug ./training_data/drug200_training.csv ./target_data/drug200_target.csv`
   - `>> KNN prediction result has been saved to ./out/knn_drug200_target_result.csv (with k = 1 that correctly predict 76.32% of the testing data)`
   - `>>ERROR: The provided training data is not normally distributed, GNB cannot be performed!`
   - Data from drug200_training is not normally distributed, GNB cannot be performed



Age	Sex	BP	Cholesterol	Na_to_K	Drug
47	M	LOW	HIGH	13.093	drugC
28	F	NORMAL	HIGH	7.798	drugC
41	M	LOW	HIGH	11.037	drugX
50	F	NORMAL	HIGH	12.703	drugX
69	M	LOW	NORMAL	11.455	drugB

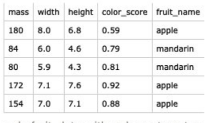
knn_drug200_target_result



Age	Sex	BP	Cholesterol	Na_to_K	Drug
47	M	LOW	HIGH	13.093	drugC
28	F	NORMAL	HIGH	7.798	drugX
41	M	LOW	HIGH	11.037	drugC
50	F	NORMAL	HIGH	12.703	drugX
69	M	LOW	NORMAL	11.455	drugX

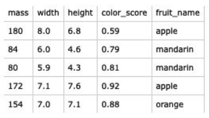
drug200_sample

2. `$ python3 final.py fruit_name ./training_data/fruit_data_with_colours_training.csv ./target_data/fruit_data_with_colours_target.csv mass width height color_score`
   - `>>KNN prediction result has been saved to ./out/knn_fruit_data_with_colours_target_result.csv (with k = 1 that correctly predict 63.64% of the testing data)`
   - `>>GNB prediction result has been saved to ./out/gnb_fruit_data_with_colours_target_result.csv (with an accuracy that correctly predict 72.73% of the testing data)`

- 

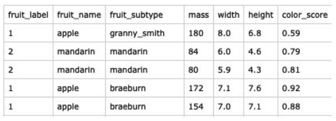
mass	width	height	color_score	fruit_name
180	8.0	6.8	0.59	apple
84	6.0	4.6	0.79	mandarin
80	5.9	4.3	0.81	mandarin
172	7.1	7.6	0.92	apple
154	7.0	7.1	0.88	apple

gnb_fruit_data_with_colours_target_result



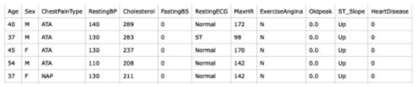
mass	width	height	color_score	fruit_name
180	8.0	6.8	0.59	apple
84	6.0	4.6	0.79	mandarin
80	5.9	4.3	0.81	mandarin
172	7.1	7.6	0.92	apple
154	7.0	7.1	0.88	orange

knn_fruit_data_with_colours_target_result



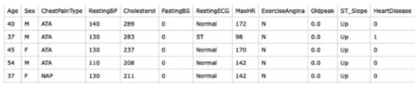
fruit_label	fruit_name	fruit_subtype	mass	width	height	color_score
1	apple	granny_smith	180	8.0	6.8	0.59
2	mandarin	mandarin	84	6.0	4.6	0.79
2	mandarin	mandarin	80	5.9	4.3	0.81
1	apple	braeburn	172	7.1	7.6	0.92
1	apple	braeburn	154	7.0	7.1	0.88

fruit_data_with_colours_sample

3. `$ python3 final.py HeartDisease ./training_data/heart_training.csv ./target_data/heart_target.csv`
   - `>>KNN prediction result has been saved to ./out/knn_heart_target_result.csv (with k = 9 that correctly predict 71.51% of the testing data)`
   - `>>GNB prediction result has been saved to ./out/gnb_heart_target_result.csv (with an accuracy that correctly predict 88.37% of the testing data)`
   - 

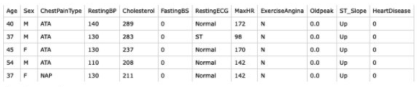
Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0
37	M	ATA	130	283	0	ST	98	N	0.0	Up	0
45	F	ATA	130	237	0	Normal	170	N	0.0	Up	0
54	M	ATA	110	208	0	Normal	142	N	0.0	Up	0
37	F	NAP	130	211	0	Normal	142	N	0.0	Up	0

gnb_heart_target_result



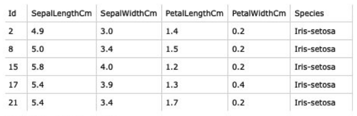
Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0
37	M	ATA	130	283	0	ST	98	N	0.0	Up	1
45	F	ATA	130	237	0	Normal	170	N	0.0	Up	0
54	M	ATA	110	208	0	Normal	142	N	0.0	Up	0
37	F	NAP	130	211	0	Normal	142	N	0.0	Up	0

knn_heart_target_result



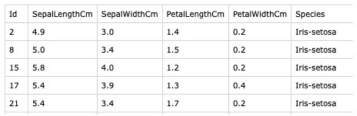
Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0
37	M	ATA	130	283	0	ST	98	N	0.0	Up	0
45	F	ATA	130	237	0	Normal	170	N	0.0	Up	0
54	M	ATA	110	208	0	Normal	142	N	0.0	Up	0
37	F	NAP	130	211	0	Normal	142	N	0.0	Up	0

heart_sample

4. `$ python3 final.py Species ./training_data/Iris_training.csv ./target_data/Iris_target.csv`
   - `>>KNN prediction result has been saved to ./out/knn_Iris_target_result.csv (with k = 1 that correctly predict 100% of the testing data)`
   - `>>GNB prediction result has been saved to ./out/gnb_Iris_target_result.csv (with an accuracy that correctly predict 100% of the testing data)`
   - 

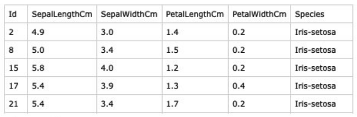
Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
2	4.9	3.0	1.4	0.2	Iris-setosa
8	5.0	3.4	1.5	0.2	Iris-setosa
15	5.8	4.0	1.2	0.2	Iris-setosa
17	5.4	3.9	1.3	0.4	Iris-setosa
21	5.4	3.4	1.7	0.2	Iris-setosa

gnb_Iris_target_result



Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
2	4.9	3.0	1.4	0.2	Iris-setosa
8	5.0	3.4	1.5	0.2	Iris-setosa
15	5.8	4.0	1.2	0.2	Iris-setosa
17	5.4	3.9	1.3	0.4	Iris-setosa
21	5.4	3.4	1.7	0.2	Iris-setosa

knn_Iris_target_result



Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
2	4.9	3.0	1.4	0.2	Iris-setosa
8	5.0	3.4	1.5	0.2	Iris-setosa
15	5.8	4.0	1.2	0.2	Iris-setosa
17	5.4	3.9	1.3	0.4	Iris-setosa
21	5.4	3.4	1.7	0.2	Iris-setosa

Iris_sample

5. `$ python3 final.py class ./training_data/mushrooms_training.csv ./target_data/mushrooms_target.csv`
   - `>>KNN prediction result has been saved to ./out/knn_mushrooms_target_result.csv (with k = 1 that correctly predict 98.8% of the testing data)`
   - `>>ERROR: The provided training data is not normally distributed, GNB cannot be performed!`
   - The dataset is too large to fit, please refer to `./out/knn_mushrooms_target_result.csv`
6. `$ python3 final.py Walc ./training_data/student_training.csv ./target_data/student_target.csv sex age famsize Pstatus Medu Fedu studytime activities freetime health absences`
   - `>>KNN prediction result has been saved to ./out/knn_student_target_result.csv (with k = 6 that correctly predict 29.73% of the testing data)`
   - `>>GNB prediction result has been saved to ./out/gnb_student_target_result.csv (with an accuracy that correctly predict 33.78% of the testing data)`
   - The dataset is too large to fit, please refer to `./out/knn_student_target_result.csv` and `gnb_student_target_result.csv`

- Note: the accuracy is calculated based on the percentage of successful prediction for the testing portion of the training data.

V. Conclusion

Based on the experience of implementing the algorithm, the observation of the result file and the estimated accuracy, and the time performance, we make the conclusion that both of these algorirthms have their strengths and drawbacks, and in some way they are complementary.

- KNN
  - Strength
    - Independent of the data type (discrete or continuous) or the distribution of the data, KNN will always generate a result.
    - The nature of the algorithm is very intuitive and easy to understand
    - The most effective k is once for all.
    - The algorithm itself doesn't assume anything.
    - The KNN itself doesn't require a training phase, yet it may take some time for finding the most effective k when the

sample is big enough.

- Limitations
    - The amount of calculations is astronomical (memory consuming) when the sample gets big
    - Finding the most effective k is time-consuming if the sample is big, we may have to make an empirical assumption that the most effective k is witin 10.
        - An anormally result appears for the prediction for `mushrooms_target.csv`, where the sample is large and feature is various, yet KNN can have 99.8% accuracy on the testing data. We make a conclusion that knn may perform better if there's fewer classifications (binary/trinary).
    - Because Euclidean distance is used for similarity comparison, the accuracy goes down as the dimension increases.
    - All the attributes will be treated as equally, so features have to have the same scale to avoid bias.

- GNB
    - Strength
        - GNB is super fast as the calculation flow is linear.
        - Running time is independent of the sample size
        - Computationally efficient.
        - The prediction performance is decent for most of data.
    - Limitations
        - GNB highly depends on the type of the data. The data has to be continuious for precise predictionl
        - GNB highly depends on the distribution of the data. Every feature has to be normally distributted (variance assumed) for successful prediction, as the standard diviation cannot be zero when calculating the gaussian probability density function.
        - It makes a strong assumption that all the features are independent, this may be problemetic if the number of features goes big.

†Appendix: Dataset Details

1. Iris Data Set
    - Attribute Information:
        - Sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - Species : Iris Setosa, Iris Versicolour, Iris Virginica
2. Student Alcohol Consumption
    - Context:
        - The data were obtained in a survey of students math and portuguese language courses in secondary school. It contains a lot of interesting social, gender and study information about students. You can use it for some EDA or try to predict students final grade.
    - Attribute Information:
        - school - student's school (binary: 'GP' - Gabriel Pereira or 'MS' - Mousinho da Silveira)
        - sex - student's sex (binary: 'F' - female or 'M' - male)
        - age - student's age (numeric: from 15 to 22)
        - address - student's home address type (binary: 'U' - urban or 'R' - rural)
        - famsize - family size (binary: 'LE3' - less or equal to 3 or 'GT3' - greater than 3)
        - Pstatus - parent's cohabitation status (binary: 'T' - living together or 'A' - apart)
        - Medu - mother's education (numeric: 0 - none, 1 - primary education (4th grade), 2 – 5th to 9th grade, 3 – secondary education or 4 – higher education)
        - Fedu - father's education (numeric: 0 - none, 1 - primary education (4th grade), 2 – 5th to 9th grade, 3 – secondary education or 4 – higher education)

        - Mjob - mother's job (nominal: 'teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at_home' or 'other')
        - Fjob - father's job (nominal: 'teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at_home' or 'other')
        - reason - reason to choose this school (nominal: close to 'home', school 'reputation', 'course' preference or 'other')
        - guardian - student's guardian (nominal: 'mother', 'father' or 'other')
        - traveltime - home to school travel time (numeric: 1 - <15 min., 2 - 15 to 30 min., 3 - 30 min. to 1 hour, or 4 - >1 hour)
        - studytime - weekly study time (numeric: 1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - >10 hours)
        - failures - number of past class failures (numeric: n if 1<=n<3, else 4)
        - schoolsup - extra educational support (binary: yes or no)
        - famsup - family educational support (binary: yes or no)
        - paid - extra paid classes within the course subject (Math or Portuguese) (binary: yes or no)
        - activities - extra-curricular activities (binary: yes or no)
        - nursery - attended nursery school (binary: yes or no)
        - higher - wants to take higher education (binary: yes or no)
        - internet - Internet access at home (binary: yes or no)
        - romantic - with a romantic relationship (binary: yes or no)
        - famrel - quality of family relationships (numeric: from 1 - very bad to 5 - excellent)
        - freetime - free time after school (numeric: from 1 - very low to 5 - very high)
        - goout - going out with friends (numeric: from 1 - very low to 5 - very high)
        - Dalc - workday alcohol consumption (numeric: from 1 - very low to 5 - very high)
        - Walc - weekend alcohol consumption (numeric: from 1 - very low to 5 - very high)
        - health - current health status (numeric: from 1 - very bad to 5 - very good)
        - absences - number of school absences (numeric: from 0 to 93)
        - G1 - first period Math grade (numeric: from 0 to 20)
        - G2 - second period Math grade (numeric: from 0 to 20)
        - G3 - final Math grade (numeric: from 0 to 20, output target)

3. fruit
    - Attribute Information:
        - fruit_label: numeric label of fruit_name
        - fruit_name: the name of the targeting fruit
        - fruit_subtype: the subtype of the fruit
        - mass (numeric: 76-362)
        - width (numeric: 5.8-9.6)
        - height (numeric: 4-10.5)
        - color_score (numeric: 0.55-0.93)
4. Heart Failure Prediction Dataset
    - Context:
        - Cardiovascular diseases (CVDs) are the number 1 cause of death globally, taking an estimated 17.9 million lives each year, which accounts for 31% of all deaths worldwide. Four out of 5CVD deaths are due to heart attacks and strokes, and one-third of these deaths occur prematurely in people under 70 years of age. Heart failure is a common event caused by CVDs and this dataset contains 11 features that can be used to predict a possible heart disease.

          People with cardiovascular disease or who are at high cardiovascular risk (due to the presence of one or more risk factors such as hypertension, diabetes, hyperlipidaemia or already established disease) need early detection and management wherein a machine learning model can be of great help.
    - Attribute Information:
        - Age: age of the patient [years]
        - Sex: sex of the patient [M: Male, F: Female]
        - ChestPainType: chest pain type [TA: Typical Angina, ATA: Atypical Angina, NAP: Non-Anginal Pain, ASY: Asymptomatic]
        - RestingBP: resting blood pressure [mm Hg]
        - Cholesterol: serum cholesterol [mm/dl]

        - FastingBS: fasting blood sugar [1: if FastingBS > 120 mg/dl, 0: otherwise]
        - RestingECG: resting electrocardiogram results [Normal: Normal, ST: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV), LVH: showing probable or definite left ventricular hypertrophy by Estes' criteria]
        - MaxHR: maximum heart rate achieved [Numeric value between 60 and 202]
        - ExerciseAngina: exercise-induced angina [Y: Yes, N: No]
        - Oldpeak: oldpeak = ST [Numeric value measured in depression]
        - ST_Slope: the slope of the peak exercise ST segment [Up: upsloping, Flat: flat, Down: downsloping]
        - HeartDisease: output class [1: heart disease, 0: Normal]
5. Drug Classification
    - Context:
        - To predict the outcome of the drugs that might be accurate for the patient.
    - Attribute Information:
        - Age (numeric : 15-74)
        - Sex (binary : M/F)
        - Blood Pressure Levels (BP) (trinary: HIGH/LOW/NORMAL)
        - Cholesterol Levels (binary: HIGH/NORMAL)
        - Na to Potassium Ration (numeric: 6.27 - 38.2)
        - Drug: DrugY, DrugX, DrugA, DrugB, DrugC
6. Mushroom Classification
    - Context:
        - This dataset includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family Mushroom drawn from The Audubon Society Field Guide to North American Mushrooms (1981). Each species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. This latter class was combined with the poisonous one.
    - Attribute Information:
        - class: edible=e, poisonous=p
        - cap-shape: bell=b, conical=c,convex=x, flat=f, knobbed=k, sunken=s
        - cap-surface: fibrous=f, grooves=g, scaly=y, smooth=s
        - cap-color: brown=n, buff=b, cinnamon=c, gray=g,green=r, pink=p, purple=u, red=e, white=w, yellow=y
        - bruises?: bruises=t, no=f
        - odor: almond=a, anise=l, creosote=c, fishy=y, foul=f, musty=m, none=n, pungent=p, spicy=s
        - gill-attachment: attached=a, descending=d, free=f, notched=n
        - gill-spacing: close=c, crowded=w, distant=d
        - gill-size: broad=b, narrow=n
        - gill-color: black=k, brown=n, buff=b, chocolate=h, gray=g, green=r, orange=o, pink=p, purple=u, red=e, white=w, yellow=y
        - stalk-shape: enlarging=e, tapering=t
        - stalk-root: bulbous=b, club=c, cup=u, equal=e, rhizomorphs=z, rooted=r, missing=?
        - stalk-surface-above-ring: fibrous=f, scaly=y, silky=k, smooth=s
        - stalk-surface-below-ring: fibrous=f, scaly=y, silky=k, smooth=s
        - stalk-color-above-ring: brown=n, buff=b,cinnamon=c,gray=g, orange=o, pink=p, red=e, white=w, yellow=y
        - stalk-color-below-ring: brown=n, buff=b,cinnamon=c,gray=g, orange=o, pink=p, red=e, white=w, yellow=y
        - veil-type: partial=p, universal=u
        - veil-color: brown=n, orange=o, white=w, yellow=y
        - ring-number: none=n, one=o, two=t
        - ring-type: cobwebby=c, evanescent=e, flaring=f, large=l, none=n, pendant=p, sheathing=s, zone=z
        - spore-print-color: black=k, brown=n, buff=b, chocolate=h, green=r, orange=o, purple=u, white=w, yellow=y
        - population: abundant=a, clustered=c, numerous=n, scattered=s, several=v, solitary=y
        - habitat: grasses=g, leaves=l, meadows=m, paths=p, urban=u, waste=w, woods=d