Predict whether the Gene has function “Cell Communication” Using micro expression data and functional data Using various Classification techniques.

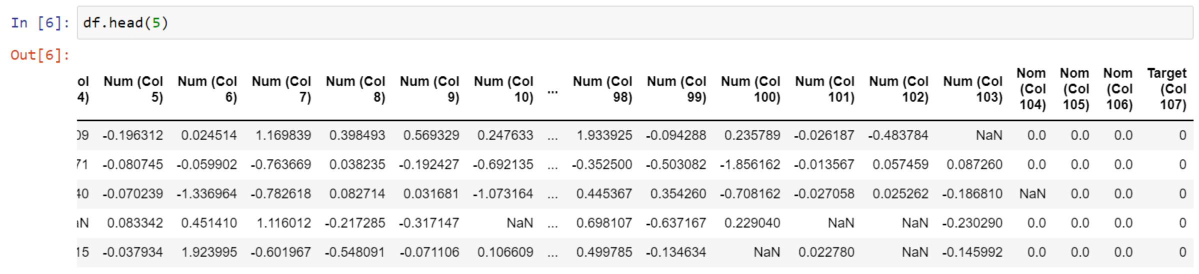
Ninaad Akella
School of Information Technology and Electrical Engineering
The University of Queensland, QLD., 4072, Australia.

Abstract

As the problem statement suggests, the objective of this project is to accurately predict whether the Gene has function “Cell Communication” using micro expression data and functional data available in the “Ecoli.csv” dataset using various Classification techniques. But before we can apply the classification on the data we have to check the if the data is consistent, complete and usable in its current state. If the data has issues, we will have to solve these issues first by using various pre-processing techniques like data imputation, normalization and feature engineering. The Classification algorithms that I plan to use are Decision Tree, K-Nearest Neighbors and Naïve Bayes. To improve the performance of these models we will use K-fold Cross Validation.

1 Understanding the Data

The dataset provided “Ecoli.csv” has 107 columns. Of these 107 columns, 103 are numerical features and three columns are nominal features (column 104 to 106). And the final column (107th) is the target column which indicates the label of the dataset representing “Cell communication”. In this column, the positive class “1” denotes the presence of “Cell communication” and the negative class “0” indicates its absence. This dataset has 1500 datapoints.



```
In [6]: df.head(5)
Out[6]:
```

ol 4)	Num (Col 5)	Num (Col 6)	Num (Col 7)	Num (Col 8)	Num (Col 9)	Num (Col 10)	...	Num (Col 98)	Num (Col 99)	Num (Col 100)	Num (Col 101)	Num (Col 102)	Num (Col 103)	Nom (Col 104)	Nom (Col 105)	Nom (Col 106)	Target (Col 107)
)9	-0.196312	0.024514	1.169839	0.398493	0.569329	0.247633	...	1.933925	-0.094288	0.235789	-0.026187	-0.483784	NaN	0.0	0.0	0.0	0
71	-0.080745	-0.059902	-0.763669	0.038235	-0.192427	-0.692135	...	-0.352500	-0.503082	-1.856162	-0.013567	0.057459	0.087260	0.0	0.0	0.0	0
40	-0.070239	-1.336964	-0.782618	0.082714	0.031681	-1.073164	...	0.445367	0.354260	-0.708162	-0.027058	0.025262	-0.186810	NaN	0.0	0.0	0
iN	0.083342	0.451410	1.116012	-0.217285	-0.317147	NaN	...	0.698107	-0.637167	0.229040	NaN	NaN	-0.230290	0.0	0.0	0.0	0
15	-0.037934	1.923995	-0.601967	-0.548091	-0.071106	0.106609	...	0.499785	-0.134634	NaN	0.022780	NaN	-0.145992	0.0	0.0	0.0	0

Figure 1

From the above image it is clear that the dataset is not complete.

2 Data Pre-processing

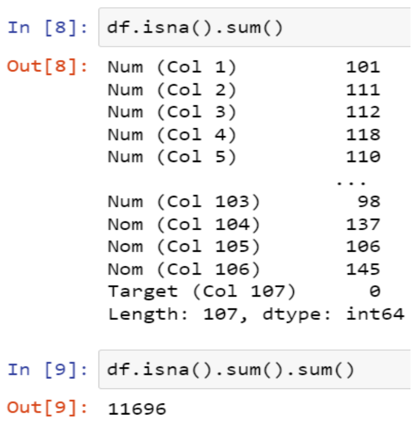
2.1 Feature Selection

Because of the **curse of dimensionality** in data mining, we need to consider Feature Selection. Curse of dimensionality refers to the fact “when dimensionality (number of features) increases, the amount of data we require to generalize accurately grows **exponentially**”.

In this dataset, there are 106 features and 1500 data points. So, based on the performance of the models, we may have to perform feature selection on this dataset.

2.2 Data Imputation

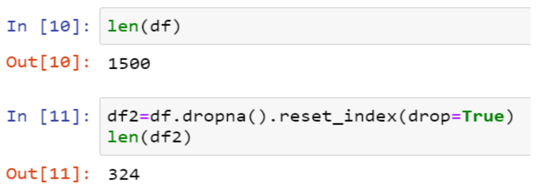
From figure 1, we can clearly see that there are “NaN” values present in this dataset. To decide whether we should perform data imputation or simply just remove all the data points with “NaN” values we need to know how many rows have these values.



```
In [8]:  df.isna().sum()

Out[8]:  Num (Col 1)          101
         Num (Col 2)          111
         Num (Col 3)          112
         Num (Col 4)          118
         Num (Col 5)          110
                             ...
         Num (Col 103)         98
         Nom (Col 104)        137
         Nom (Col 105)        106
         Nom (Col 106)        145
         Target (Col 107)       0
         Length: 107, dtype: int64

In [9]:  df.isna().sum().sum()

Out[9]:  11696
```

Figure 2

As we can clearly see, that each column has around 100 “NaN” values and in total there are 11696, though a lot of them in different columns but same data point. Let us how many rows remain after removing the “Nan” values.



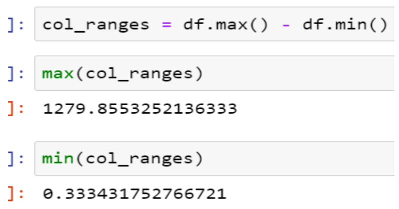
```
In [10]:  len(df)

Out[10]:  1500

In [11]:  df2=df.dropna().reset_index(drop=True)
          len(df2)

Out[11]:  324
```

Figure 3

As we can see, that after removing all the “Nan” values we have just 324 rows left. So, clearly, we need to use data imputation for this dataset. Imputation can be performed either by feature’s all values or class specific values. Based on the feature and performance of the models we will choose which one to use.

2.3 Normalization

Different numerical features can have different ranges of values. And the magnitude of these ranges can impact the performance of the classification. In such cases, we need to normalize the feature values. Let us see if our dataset has this issue.



```
]:  col_ranges = df.max() - df.min()

]:  max(col_ranges)

]:  1279.8553252136333

]:  min(col_ranges)

]:  0.333431752766721
```

Figure 4

Clearly, the magnitudes of ranges of the numerical features present in the dataset varies between 0.33 and 1279.855. So, we have to perform normalization on this dataset. There are two methods to perform this “min-max normalization” and “z-score normalization (standardization)”. We will decide which one to use after further investigation on the dataset.

2.4 Anomaly Detection

Anomaly can be defined as something that “deviates” from what is standard or considered as normal. In anomaly detection we find these rare data points as they might affect the performance of our classification model. There are five methods of anomaly detection which are “Density-based technique”, “Model-based technique”, “Distance-based techniques”, “Cluster-based technique” and “Isolation-based technique”. After further investigation of our data, we will decide which method to use.

3 Classification Methods

All Classification methods are grouped under Supervised Learning. Supervised Learning means we supervise the learning process of the model by providing it with label on which it can form the algorithm. For this project we will be using only classification methods as the objective requires us to train on a dataset with labels present.

3.1 Decision Tree

This classification method where the data is continuously split according to a certain parameter. The tree that forms can be explained by two entities, nodes and leaves. Nodes are where data is split based on a specific condition. The leaves are the final outcomes or decisions that we reach because of the nodes. The “certain parameter” I mentioned previously refers to “The measure of purity”. There are three of these “Information Gain”, “Gain ratio” and “Gini index”. We will try all three to figure out which one yields the best result for our dataset. The algorithm we are using is known as ID3 and has predefined functions in skit-learn. Other algorithms include C4.5 which we might try based on time constraints.

3.2 K-Nearest Neighbors

The main idea of KNN lies in identifying the label of “k” nearest data point of the test point and decide the label of the test point by using majority rule. K-NN is sometimes called “lazy learning approach” or “instance-based learning” because the training data is just memorized without building any model in the training phase. The more similar two objects are, the less distance is between them and this distance is calculated using **“Minkowski Distance”**. Which is represented as:

$$\text{dist}(x_i,x_j) = \left[\sum_{u=1}^{d} |x_{iu} - x_{ju}|^{p}\right]^{1/p}$$

Based on the value of p:
P=1: Manhattan distance
P=2: Euclidean distance (Most common)
P=infinite: Chebyshev distance
These three are the most commonly used distance equaltions.

3.3 Naïve Bayes

This is a probabilistic classification model. It performs classification by estimating posterior distribution i.e., by estimating p(y|x) where x is the instance and y is the label. For a dataset with two labels y1 and y2, the label for the data point is decided by: p(y1|x)>p(y2|x) than the assigned label is y1 otherwise it is y2. In naïve bayes method, we assure attribute conditional independence which means “Given the class information, all features are conditionally independent from each other”.

3.4 Random Forest

Random Forest is a type of Ensemble Learning. Ensemble learning is usually considered as an easy and powerful method to achieve better generalization. In this algorithm, we construct various different decision trees instead of one and combine all of them together.

4 K- Fold Cross Validation

In K-fold cross validation, we **randomly** partition the data into k folders. Then, iteratively, we train the model with (k-1) folders and validate the model using the last folder. So, each folder is used for testing once. Finally, we average the score for each run. This method is used for tuning the hyper-parameters and get the highest score for each model. I may use grid search method along with K fold CV to get the best possible result.

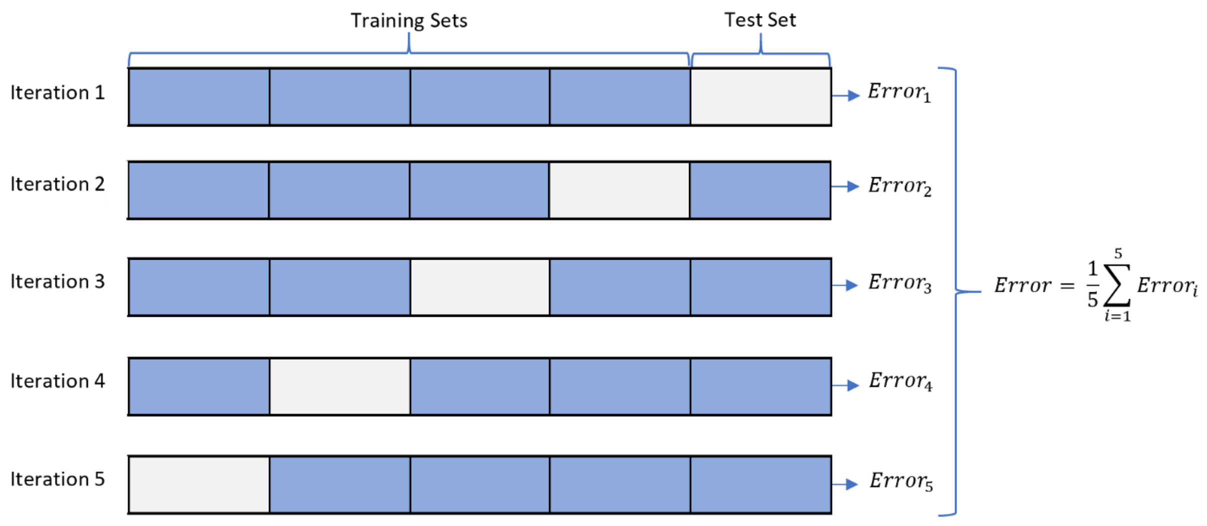


Figure 5

5 Evaluation

The main evaluation metric used for this project will be F1 score as it is the usually considered as the best evaluation metric for classification. F1 score can be defined as the harmonic mean between precision recall. It uses the confusion matrix to calculate precision and recall.

6 Timeline

The project will be carried out as follows:
[1] 09-09-2022 → 29-09-2022: Data Pre-Processing.
[2] 30-09-2022 → 15-10-2022: Classification.
[3] 16-10-2022 → 25-10-2022: Hyperparameter tuning.

References

[1] www.towardsdatascience.com.
[2] www.geeksforgeeks.com.
[3] www.stackoverflow.com.
[4] Charu C. Aggarwal Data Mining: The Textbook.
[5] Pang-Ning Tan, Michael Steinbach, Anuj Karptne and Vipin Kumar Introduction to Data Mining.