Tabular Data Classification

This report is related to Digikala assessment task

Rayehe Hosseinipour - August 14, 2020

Introduction

A tabular dataset has been provided with an average of 28.7% missing values. The objective is to preprocessing the dataset, dealing with missing values, feature selection, and choosing a proper model to classify data. The attached Jupiter notebook contains all codes used for this assessment and corresponding cell description. You can find the hole project in this GitHub repository

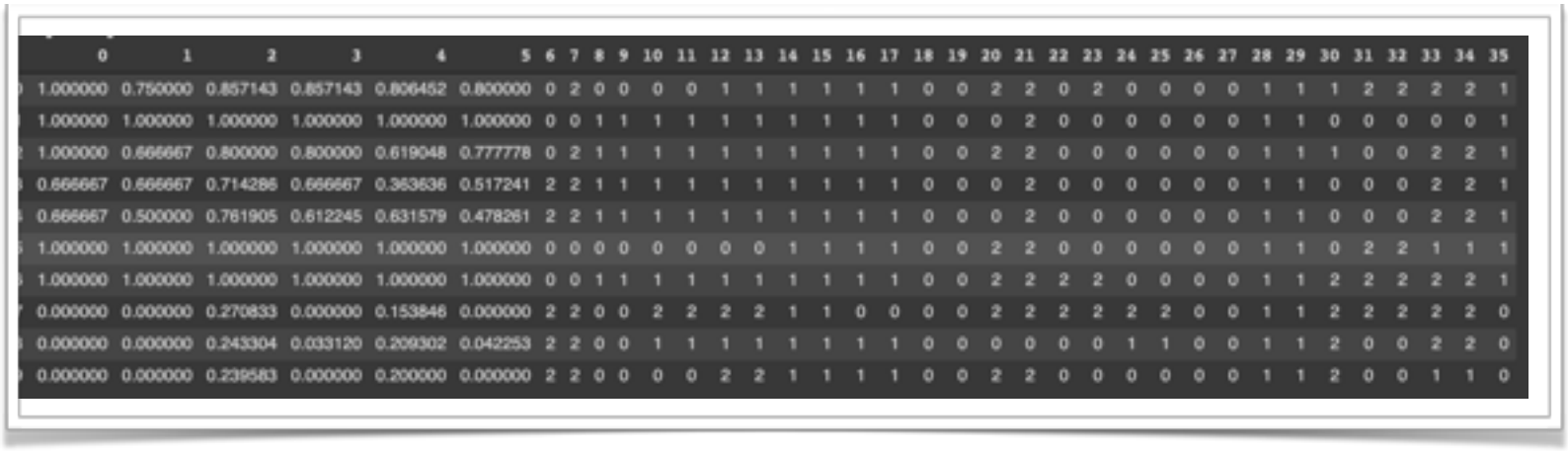
Data Preparation

provided data set consist of 31018 labeled samples and 3447 unlabeled samples.

- In (Reading the Data) cell : dataset has been loaded as a pandas data frame, and missing values are replaced with standard NaN value.
- In (Pre Processing) : I just kept the columns with less than 1% missing values, the reason behind choosing this threshold is that missing values rate in columns was either above 28% or about 1%. Afterward, I searched for common candidate features in labeled and unlabeled data to ensure I'm keeping the same features for both. As a result, there are 34 columns without any missing values and 20 columns with less than 1% missing values.
- In (Dealing with missing values) : we could either drop all columns with NaN values or Impute missing ones using an imputation method. The code includes both functions to

doing so, but in practice imputing the missing values of preserved columns doesn't help at all. so I just dropped NaNs.

- In(Categorical Labels Encoding) : next challenge is to encode categorical features to integers. To do so, I applied Sikitlearn LabelEncoder to columns containing string values.



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
1.000000	0.750000	0.857143	0.857143	0.806452	0.800000	0	2	0	0	0	0	1	1	1	1	1	1	0	0	2	2	0	2	0	0	0	0	1	1	1	2	2	2	2	1
1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	2	0	0	0	0	0	0	1	1	0	0	0	0	0	1
1.000000	0.666667	0.800000	0.800000	0.619048	0.777778	0	2	1	1	1	1	1	1	1	1	1	1	0	0	2	2	0	0	0	0	0	0	1	1	1	0	0	2	2	1
0.666667	0.666667	0.714286	0.666667	0.363636	0.517241	2	2	1	1	1	1	1	1	1	1	1	1	0	0	0	2	0	0	0	0	0	0	1	1	0	0	0	2	2	1
0.666667	0.500000	0.761905	0.612245	0.631579	0.478261	2	2	1	1	1	1	1	1	1	1	1	1	0	0	0	2	0	0	0	0	0	0	1	1	0	0	0	2	2	1
1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	0	0	0	0	0	0	0	0	1	1	1	1	0	0	2	2	0	0	0	0	0	0	1	1	0	2	2	1	1	1
1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	0	0	1	1	1	1	1	1	1	1	1	1	0	0	2	2	2	2	0	0	0	0	1	1	2	2	2	2	2	1
0.000000	0.000000	0.270833	0.000000	0.153846	0.000000	2	2	0	0	2	2	2	2	1	1	0	0	0	0	2	2	2	2	2	2	0	0	1	1	2	2	2	2	2	0
0.000000	0.000000	0.243304	0.033120	0.209302	0.042253	2	2	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	1	1	0	0	1	1	2	0	0	2	2	0
0.000000	0.000000	0.239583	0.000000	0.200000	0.000000	2	2	0	0	0	0	2	2	1	1	1	1	0	0	2	2	0	0	0	0	0	0	1	1	2	0	0	1	1	0

Feature Selection

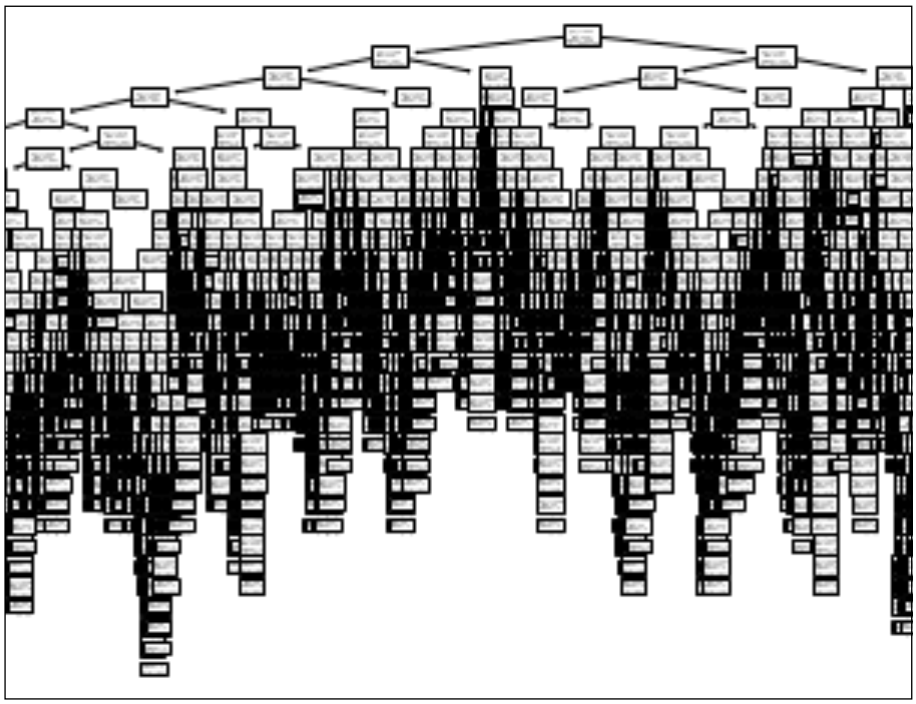
Now that we applied preprocessing techniques on our data, it is time to select the most informative features.

- In (Feature Selection) : I've used the sklearn SelectKBest module to choose the most correlated features to the corresponding label. Top 10 features were kept as training features due to accuracy and overhead tradeoff.
- Top 20 gives a slightly better result (about 0.5% in test accuracy) but is much more time consuming for training decision-tree.
- And by selecting less than 8 features, accuracy drops by 10%.

Learning Models

1. Decision Tree

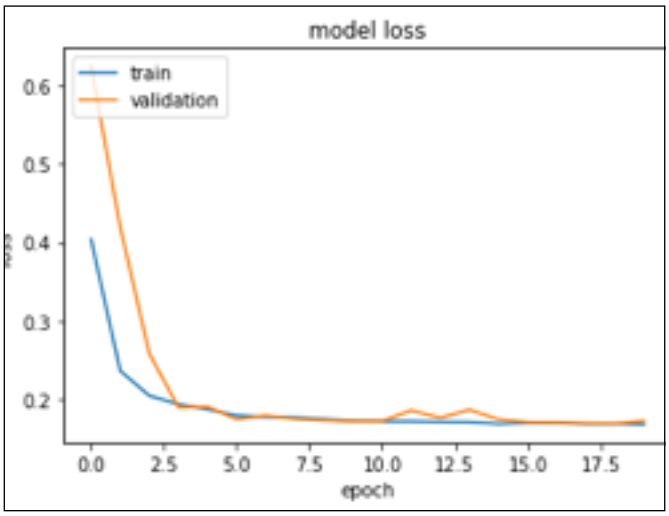
- I exploit the sikitlearn DecisionTreeClassifier to classify data as the first model. This model takes much more time than the CNN model to train.
- Test accuracy of the model in this section is 92.8%
- The final learned tree is plotted below:

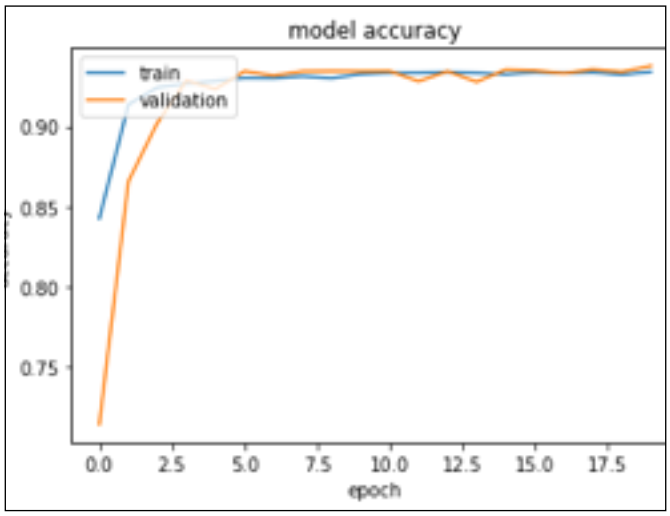


2. CNN model

- The input of the model is the same as the above decision tree. I used 90% of labels data for train and validation, and the rest 10% for the teat phase.
- My sequential model consists of 3 Convolutional layers and corresponding to those , 3 BatchNormalization layers.
- Final convolved outputs are flattened and passed to the classifier.
- The training process is repeated for 20 epochs, and the batch size is equal to 100.

- the model has a total of 918 parameters to learn, and it takes less than a minutes to finish training.
- Test Accuracy and Loss curve are plotted below according to model history
- this model achieved 93.85% test accuracy





Classifying unlabeled Data

- all preprocessing steps applies to the train data have also been applied to the provided test dataset.
- The label column has been added to "dk_data_test_users_withLabels.csv". this file will be stored in the same directory as training data.

Conclusion

according to the results, CNN is a better choice for classifying this dataset .both system overhead and Accuracy-wise

Regards, Rayehe