ML Assignment: Classification

Raktim Dey-MDS202132, Sucheta Jhunjhunwala-MDS202151

18th March,2022

1 Data Cleaning

- After importing the data, we plotted histograms of the numerical and categorical variables separately to understand the relationship between the input variables and output variable. We found that variables like **'pdays'**, **'day_of_week'** and **'housing'** did not affect the output variable strongly, so we dropped those attributes from our dataframe.
- The **'default'** variable mostly had unknown or no as it's entries, so we eliminated that as well.
- We removed all the samples having unknown values for any of the categorical variables.
- We then used one-hot encoding to substitute dummy values for the categorical variables such as **'marital'**, **'loan'**, **'poutcome'** and **'contact'**.
- Categorical variables such as **'job'**, **'education'** and **'month'**, having more than 3 values, were given numeric values using Label Encoder.
- We used train_test_split to split the data into training and testing data in the ratio 70:30.

2 Applying Different Classifiers

Our goal was to predict whether given the various information about marketing campaigns of a bank, a customer will subscribe for a term deposit at the bank or not. Our target hence was to find all the cases when a customer actually opened a term deposit at the bank, which is the true positive case. Hence, we decided to maximise the metric **Recall**. Keeping this in mid we proceeded as follows with each of the classifiers:

- **Decision Tree Classifier**: We built several decision trees with variable depths(**'max_depth'**) and maximum number of leaf nodes(**'max_leaf_nodes'**) to find the optimum values of both the parameters which would maximise the recall metric.
- **Random Forest Classifier**: We tried to find the optimum values of the two parameters, maximum depth of any tree(**'max_depth'**) and maximum number of trees(**'n_estimators'**) to maximise Recall.
- **Naïve Bayes Classifier**: We chose GaussianNB as our Naïve Bayes classifier and built our classifier to fit our training data.

At the end, we also built an ensemble classifier- a voting classifier which aggregates the predictions of each classifier and predicts the class that gets the most votes.

3 Observations

- We tried to avoid dropping certain variables but that didn't create a significant difference in the different metrics.
- We also observed that more than 70% of the target variables had yes as its value. This was creating an imbalance in the number of yes and no. We tried to remove this imbalance but this didn't show any improvement.
- Initially without assigning the various parameters to our Decision Tree Classifier created a highly complicated tree, which might later on lead to overfitting. To overcome this we evaluated our metric on a certain range of values for these parameters.
- The Voting Classifier also gives satisfactory results with 53% recall and 91% accuracy.

4 Comparative Evaluation

Classifier	**Accuracy**	**Recall**	**Precision**	**Time**	**Memory**
Decision Tree	90%	49%	57%	0.78s	267.39Mb
Random Forest	91%	45%	67%	0.83s	265.61Mb
Naive Bayes	84%	58%	37%	0.77s	272.80Mb

5 Conclusion

All the three classifiers, take almost the same time and occupy almost the same amount of space. Although Random Forest has the highest accuracy, Naive Bayes has the highest recall when we had to identify the 1's, or in other words, it maximises the chances of finding the customers who will take a term deposit and minimise the chance of getting False Negatives. If we can manage to increase the Recall for Random Forest then in terms of accuracy, recall, precision, time taken and the space occupied, Random Forest Classifier would be the best suited one. Here, since we fixed our goal to maximise recall, Naive Bayes should be used for classification purposes of any unknown data from the bank.

6 External Packages Used

External packages used:
1) Memory_profiler- To calculate the memory consumption by our models.
2) Sklearn- To build the learning models.
3) Numpy- For mathematical calculations.
4) Pandas- For dataframe analysis.
5) Matplotlib- To make and visualize histograms and bar plots of the numerical variables.
6) Seaborn- To make and visualize heatmaps and histograms of categorical variables.
7) Timeit- To calculate the time taken by the models to fit the training data.
8) Os- To store the path of the input.
9) Six, IPython, pydotplus- To visualize the decision tree.

7 Link to the Python code

https://colab.research.google.com/drive/1KZS9YdLfqvBJt2jAC1eHdI2ttdp4UR2c?usp=sharing