Machine Learning 2023 – Mini Project 1

Introduction

The purpose of this project was to analyze a dataset containing data on a marketing campaign of a banking institution. The goal was to produce two machine learning models trained on the data that can predict the success of a marketing campaign with at least 80% accuracy.

The outcome to be predicted is a “yes” or “no” for whether the campaign was successful for the current record, and this calls for using algorithms that can handle binary classification.

The project is executed in a python Jupyter notebook, using libraries sklearn and LightGBM for the algorithms and various tools for data processing.

Data processing

Before prediction can be done on the data steps need to be taken to process the data into a usable format. First step is loading the data and exploring it to see what issues may need to be addressed before proceeding to training a model. This entails formatting activities such as ensuring that numerical values are handled correctly, formatting categorical columns as onehot encoded if the selected algorithm calls for it, and dealing with missing values. In the exploration insights about the data may also be gathered.

Data load

The data was provided in a .csv format. For reading this data into the Jupyter notebook the python package pandas was used, with the function read_csv. To correctly load the data a delimiter was specified as the .csv file used semicolons as delimiters. Excess quotes in the data appeared in this step and needed to be removed.

Exploration

Once data was loaded the first step of exploring the data was to use pandas functions .info() and .head() to get an overview of the data. From this can be observed that there are a number of categorical columns and a number of numeric columns. The target column for prediction is the column “y”, that is a binary category formatted as “object”.

In the next step the data was checked for missing values by selecting rows from the data where the pandas function .isna().any(axis=1) is true. The function .isna() returns a pandas dataframe where any null values show as “True” Boolean value and all else as false. The addition of .any(axis=1) applies the “True” value to the entire row. The main dataframe is then sliced with this Boolean data frame to produce a slice of the original dataframe where any row containing a null value is kept. No rows are returned by this, so no null values exist in the data.

The columns were formatted to ensure that the algorithms handle them correctly. The numeric columns were formatted with the pandas function .to_numeric() which sets the columns as integer or float type as appropriate. The categorical were changed to type “category”. (Pandas Documentation, 2024)

Next, histograms are generated for each column to show the distribution of each category of the column. This will help with discovering any imbalances in the categories. The most valuable insight from this was the imbalance in the target column “y”. There were 36548 “no” values compared to 4640 “yes” values. This would be something that came into consideration in the preprocessing of the data. The generation of the graphs were done with the python library matplotlib.

A graph showing the correlation of the numeric and categorical columns with the target column was also made with the help of matlplotlib. First the pandas function .get_dummies() was used to turn the categorical columns into sets of binary column, with one column for each category. This was done to be able to show the individual impact of each category, for each column. The pandas function .corr() was then used to create a matrix of correlation coefficients between all columns in the table. From this matrix was then selected the correlation coefficients for the value “yes” in the column “y” to show which correlations exists between all columns and a positive result. The highest positive correlations apart from “duration” are with the value “success” in column “poutcome”, a higher value in the column “previous”, and “cellular” in the column “contact”. Highest negative correlations were from a higher value in the column “nr.employed”, a higher value in the column “pdays”, and a higher value in the column “euribor3m”.

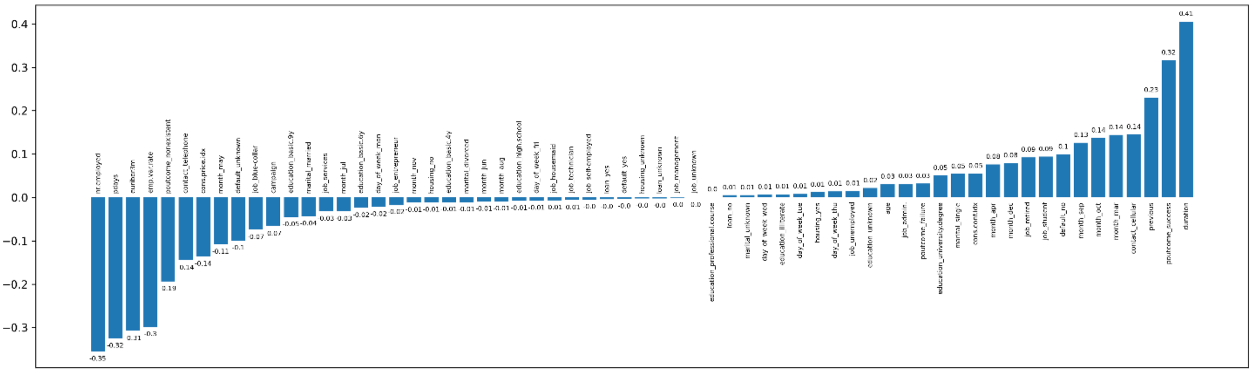

Image 1, Correlation between all columns and “yes” in column “y”.

Train and test split

The data was separated into two datasets, one for training the model and one for validating the accuracy of the model. The validation dataset was selected as 20% of the total data, randomly sampled.

Oversampling

To address the imbalance between the classes “yes” and “no” in the column “y”, the “yes” rows in the train dataset were duplicated until the counts of each class were roughly balanced.

Feature selection

The column “duration” was dropped according to the project specifications, as including it would not create a realistic prediction. An additional step of feature selection was done for the dataset used for the RandomForestClassifier. To reduce the number of features with little impact on the prediction, the features that were found to have an absolute correlation lower than 0.02 were dropped. This is because having a high number of variables with few significant variables may cause poor performance for this type of algorithm. (Hastie, Tibshirani, & Friedman, 2008)

Training the models

The selected algorithms for this were RandomForestClassifier from the sklearn library and lightgbm. Both of these are suitable alternatives for a binary classification problem. Both work on a similar principle of tree-based learning, which means that the feature space is split recursively to create a boundary between the classes. Additionally, both combine multiple of these “weaker” classifying trees to make a prediction based the average of the trees. (Hastie, Tibshirani, & Friedman, 2008)

The learning was additionally done with a grid search approach where the parameters of the models were programmatically tweaked, and the results of each run were recorded to be able to select the best overall configuration of the model.

LightGBM

First the LightGBM model was trained, with the grid search being done for “num_leaves” and “learning_rate” in the ranges 50-150 increment 10, and 0.01-0.30 increment 0.03 respectively. “num_leaves” determines the complexity of the trees in the model, a higher number leading to higher complexity and potentially better defined partitions in the feature space. “learning_rate” determines the size of steps for the gradient descent, which means that a larger value may let the model approach the ideal value faster, but a higher value may also cause the model to overshoot the ideal value. (LightGBM Documentation, 2023)

The metrics tracked were accuracy and ROC AUC score, accuracy being the percentage of correct predictions. The threshold between classes in binary classification is normally set at 0.5, but ROC AUC (Receiver Operating Characteristic Area Under the Curve) evaluates the accuracy for the full range of the threshold from 0 to 1. This gives a more complete view of the accuracy of the model in the case of imbalanced classes. (Hastie, Tibshirani, & Friedman, 2008)

Early stopping was enabled to potentially reduce overfitting of the model.

The outcome of training the model showed that the best configuration for this problem is a “num_leaves” of 130 and a learning rate of 0.10. This resulted in an accuracy of ~87.7% and an ROC AUC score of ~75.9%.

Num_leaves	Learning_rate	Accuracy	ROC_AUC
130	0.10	0.877383	0.758550
110	0.10	0.873740	0.764291
120	0.10	0.873255	0.766992
140	0.13	0.872648	0.760877
130	0.13	0.872526	0.757773

Table 1, Top performing configurations of LightGBM.

	precision	recall	f1-score	support
0	0.93	0.93	0.93	7322
1	0.45	0.44	0.45	915
accuracy			0.88	8237
macro avg	0.69	0.69	0.69	8237
weighted avg	0.88	0.88	0.88	8237

Table 2, Classification score for the top performing LightGBM configuration.

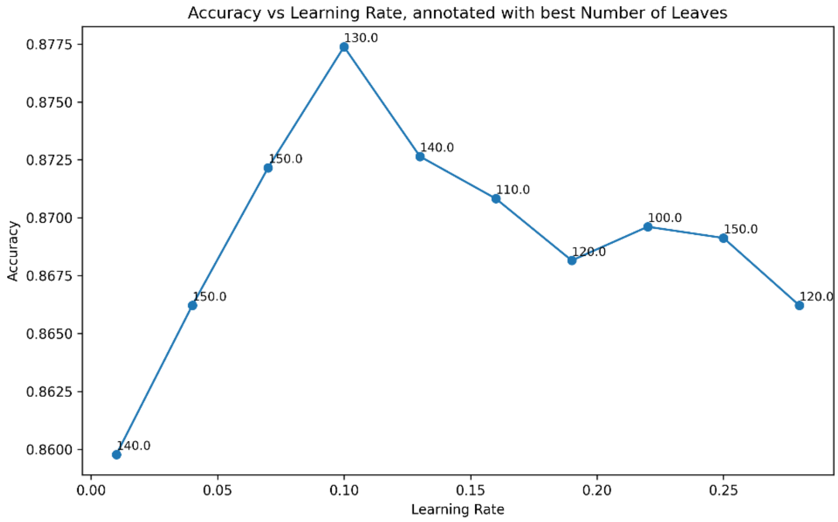

Image 2, The performance for each step of "learning_rate", with the best "num_leaves".

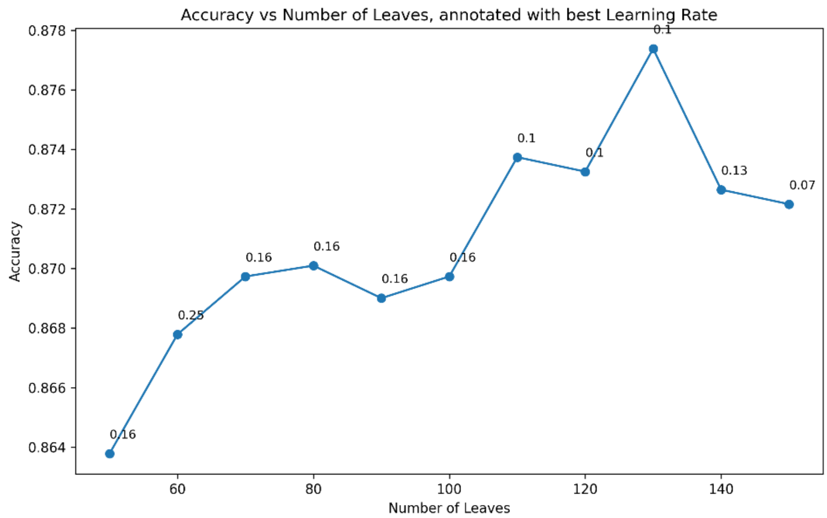

Image 3, The performance for each step of "num_leaves", with the best "learning_rate".

RandomForestClassifier

Similarly with LightGBM the training of this model was executed with a grid search, this time searching for the best configuration of "n_estimators". The best configuration for "n_estimators" was searched for in the range 50-500 with a step size of 5. The RandomForestClassifier works by running a decision tree algorithm and returning the mode of the classes of all trees. "n_estimators" is here the number of trees that are run. Increasing the number of trees will increase the complexity of the model, but it will not overfit, like some other models may, as the outcome is decided by the average of all trees. (Hastie, Tibshirani, & Friedman, 2008)

Same as for the previous model, the metrics tracked were accuracy and ROC AUC.

The best configuration with this algorithm was found at 290 estimators, which resulted in a model with ~88.7% accuracy and ~63.7% ROC AUC.

Estimators	Accuracy	ROC_AUC
220	0.886502	0.636621
170	0.886380	0.636553
200	0.886259	0.636967
180	0.886259	0.636484
210	0.886137	0.635934

Table 3, Top performing configurations of RandomForestClassifier.

	precision	recall	f1-score	support
0	0.92	0.96	0.94	7330
1	0.48	0.32	0.38	908
accuracy			0.89	8238
macro avg	0.70	0.64	0.66	8238
weighted avg	0.87	0.89	0.88	8238

Table 4, Classification score for the top performing RandomForestClassifier configuration.

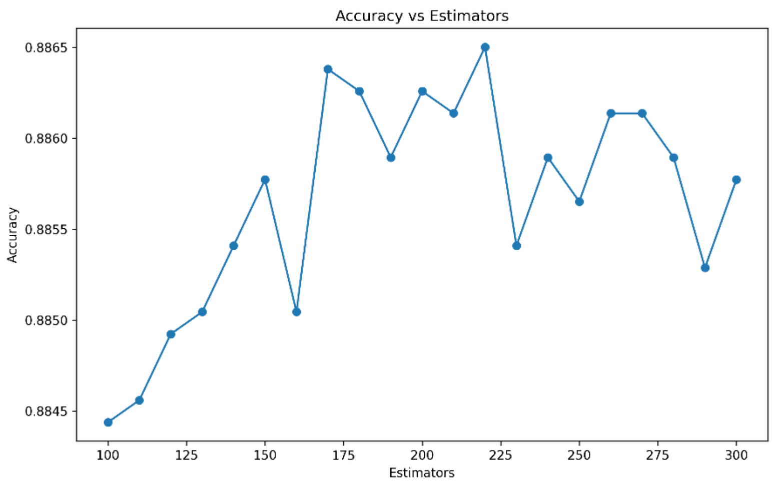

Image 4, The performance for each step of "n_estimators".

Conclusions

Both models were able to predict the correct classes with high accuracy, with a slight edge to RandomForestClassifier, but which may not have a big impact with a difference of only 1%. The main difference in the performance of the models was in the ROC AUC with a difference over 12%. This implies that at the standard threshold of 0.5 both models are equal in performance. But if the threshold would need to be adjusted for some application, such as if there is a priority to avoid false positives or false negatives, then the LightGBM model would be preferable.

It could be argued that other parameters could be selected for the LightGBM model, if instead prioritizing the ROC AUC. With a different configuration the ROC AUC improved a significant amount, while losing a bit of accuracy. Depending on the use case this could be the preferred configuration, but since the requirements for this project were stated in terms of the accuracy the configuration with the best accuracy performance was presented.

Although the models meet the overall accuracy requirement here is an imbalance in the accuracy for different classes, with a negative outcome being easier to predict. This could however still prove to be valuable in business decisions as being able to predict a negative outcome with good accuracy could potentially save a lot of time and effort by filtering down the potential targets for the campaign, even though the model can’t accurately predict the actual positive outcomes.

References

Hastie, T., Tibshirani, R., & Friedman, J. (2008). *The Elements of Statistical Learning.* Stanford: Springer.

LightGBM Documentation. (2023). Retrieved from https://lightgbm.readthedocs.io/en/stable/

Matplotlib Documentation. (2023). Retrieved from https://matplotlib.org/

Pandas Documentation. (2024). Retrieved from https://pandas.pydata.org/docs/index.html

Scikit-learn Documentation. (2024). Retrieved from https://scikit-learn.org/stable/