TED UNIVERSITY

ADA 442

Statistical Learning

Team Project Report

30/05/2024

Streamlit:

Team Members:

Batuhan İşcan 11863132364

Yiğit Özarslan 13888064750

Introduction

In this project, we worked on training an accurate machine training model which will use the training data to predict whether a client will subscribe a term deposit. The used datatypes' description were all listed in the UCI ML Repository website with the id 222. While training the model, 10% of it's original data is used to create the test model.

Data Cleaning:

First; we check for null columns and since there is none, we don't modify them. There are some columns such as 'education' or 'poutcome' which have 'unknown' or 'nonexistent'. As dropping these columns would cause a huge data loss, these values are kept.

Preprocessing:

In preprocessing, we used Ordinal encoding for month, day_of_week and the education data because education may have difference in significance. Label encoding was used for binary data the 'y' column because it was a more appropriate way of turning it into numerical data. One-Hot encoding was used to seperate and enumerate the categorical data. Afterwards, the already numerical data was scaled using StandardScaling because the difference of the value according to the standard variation is a more appropriate way to use the numerical data as they are less likely to skew the results.

Model Selection:

We applied three models:

- **RandomForestClassifier:** This method improves classification performance by combining multiple decision trees, introducing randomness to reduce overfitting and providing robust predictions.
- **XGBClassifier:** Utilizes sequence of decision trees, each one improving on the errors on the errors of the previous ones to create a highly accurate model.
- **Logistic Regression:** A straightforward yet powerful classification algorithm used to predict binary outcomes. By modelling the probability of an event occurring and applying a threshold, it provides clear and interpretable predictions.

Pipelining:

We used three datasets, one for the normal, one for one with ages column having a logarithm operation done on it and one on an oversampled dataframe. These dataframes are all splitted to make training and testing sets by the ratio of 4 to 6. There were two pipelines made to observe the differences between the models that used their FeatureSelection – which didn't have an effect.

Hyperparameter Tuning:

We used the cross_validate function to find data about the different datasets and used only the oversampled dataset on GridSearchCV to find which values of classifier specific parameters gave out a better value.

Evaluation:

	Precision	Recall	F1-score	Support
No	1.00	0.60	0.75	2190
Yes	0.24	0.99	0.39	282
Accuracy			0.65	2472
macro avg	0.62	0.79	0.57	2472
weighted avg	0.91	0.65	0.71	2472

	Precision	Recall	F1-score	Support
No	1.00	0.65	0.79	2190
Yes	0.27	1.00	0.42	282
Accuracy			0.69	2472
macro avg	0.63	0.82	0.61	2472
weighted avg	0.92	0.69	0.75	2472

	Precision	Recall	F1-score	Support
No	1.00	0.70	0.82	2190
Yes	0.30	1.00	0.46	282
Accuracy			0.73	2472
macro avg	0.65	0.85	0.64	2472
weighted avg	0.92	0.73	0.78	2472

Deployment:

After uploading the repository to GitHub, the host.py file is used as the file to launch the app. It takes the fields as inputs and predicts the output after pressing a button. There are a few column elements, selectbox elements, button elements, number_input and subheader elements in the file.

Conclusion:

To conclude, we have explored the data, created preprocessing elements and added them into a pipeline, used to fit the model. We used a few different dataframes and compared their elements while training the model. For the most successful model with the oversampled dataframe, although there is a low precision for ‘yes’ values, the weighted average is quite high.