INT303 Big Data Analysis Assignment 2 Report

Brief by Mingzirui Wu

Introduction – **Feature Engineering techniques**, an indispensable pre-processing step of machine learning, convert and optimize raw data into features for creating a predictive model using machine learning techniques. It encapsulates various data science techniques, including selecting relevant attributes, handling missing data, encoding, and normalizing it. As its highly robust and universal feature engineering technique is promoted extensively to manage "Big Data."

Methodology – This report first applies data pre-processing techniques to deal with the IBM Dataset. Four classification algorithms are involved as the criterion to evaluate different feature engineering techniques.

A. Data Preprocessing

The IBM Dataset includes 1103 rows and 35 columns conduct the dataset, and all the data come from the same sources with no empty value in the dataset. Thus, there is no need to do data cleaning and data integration. However, the IBM Dataset is imbalanced, containing only 15.6% (172) of 'Yes' samples in the attrition attribute. SMOTE (Synthetic Minority Oversampling Technique) was applied to balance the Attrition class. Among the dataset, 'Over18' and 'EmployeeCount' are single-value features, and 'EmployeeCount' is a unique identifying number without duplication. Thus, the three attributes will be removed at the training stage.

This report applied two data scaling strategies for the model to better understand the dataset: Normalization (Applied for attributes that do not display normal distribution) and Standardization (Applied for features that are typically distributed but the range is too distinctive to other features). At the end of data preprocessing, the bottom fourteen irrelevant attributes were removed based on the correlation ranking of each attribute and Attrition. This change will be compared in the results with the case without the deletion.

B. Classification Algorithm

Decision Tree and Random Forest formed the control group during training phase. The main algorithms were two distributed, high-performance *gradient boost* frameworks proposed in 2016 and 2014: LightGBM (Light Gradient Boosting Machine) and XGBoost (eXtreme Gradient Boosting). *Gradient boosting* is a supervised learning algorithm that requires a set of labeled training elements as input and builds a model that aims to predict other non-label information correctly.

LGBM and XGBoost use tree-based learning algorithms, but LGBM differs from other algorithms because of the vertical growth direction. LGBM grows tree leaf-wise while others grow level-wise (XGBoost), and it chooses the leaf with max delta loss to grow, which means this algorithm can reduce more loss than a level-wise algorithm. The algorithm also spends lower memory while handling extensive data, but it is easy to overfit the small data. Here comes XGBoost, another main algorithm used in this report to compare cases where LGBM may overfit when there is little difference in performance between the two.

Results –The k-fold cross-validation is applied to the **training set**, and Table 1demonstrate training result by applying different strategies.

Classfier	Benchmark	With Scaler	With SMOTE
DecisionTreeClassifier	0.7919	0.8145	0.7014
RandomForestClassifier	0.8869	0.8869	0.8371
XGBClassifier	0.8824	0.8824	0.8009
LGBMClassifier	0.8959	0.8959	0.7964

Table 1 Accuracy of processing strategies with classifiers

The result of feature selection with a **public test set**, the 'VotingClassifier,' is an ensemble combination of all the valuable classifiers.

Classfier	16 Attributes	30 Attributes
LGBMClassifier	0.84313	0.86274
XGBClassifier	0.84313	0.86274
SVC	0.83333	0.84313
VotingClassifier	0.85294	0.87254

Table 2 Test accuracy of feature selection

Figure 1 Screenshot of the Final Result

Discussion – For removing irrelevant features, which reduces the column size of the dataset from 30 to 16, the **contributor thought** that even though this strategy could improve training efficiency, lower accuracy is inevitable. From the problem itself, the IBM dataset is a mini-dataset, and it is worth sacrificing training speed and limitations on model size to get better performance. An available solution already proposed in this report is applying the LGBM classifier, which spends fewer resources than other classifiers and is faster simultaneously.

Conclusion – To summarize, the essential points:

1. Choose suitable normalization or standardization strategies.
2. Make a good trade-off between training speed and final performance (i.e., remove some less critical features to improve training speed).