AI351 - Machine Learning Lab

Taiwanese Bankruptcy Statistics

Ali Hasan Khan
Artificial Intelligence(FCSE)
Ghulam Ishaq Khan Institute of Engineering and Technology
Topi,Swabi, Pakistan
u2021079@giki.edu.pk

I. Introduction

The provided code demonstrates a systematic and comprehensive approach to analyzing a dataset, likely centered around financial data. The primary objective is to effectively manage class imbalance while building a robust classification model. The code implements several crucial stages, including data preprocessing, correlation analysis, and the utilization of a Decision Tree Classifier for classification modeling. Each of these stages is integral in achieving the overarching goal of extracting valuable insights from the dataset.

The data preprocessing phase forms the foundation of the analysis. The code initializes by loading the dataset from a 'data.csv' file, presumably containing financial data for analysis. It subsequently proceeds to refine the dataset by removing the 'Net Income Flag' column and encoding categorical features using the LabelEncoder. Additionally, any missing values in the dataset are imputed with the mean of the respective column, ensuring data integrity and consistency throughout the analysis.

The Exploratory Data Analysis (EDA) component is instrumental in uncovering the underlying relationships within the dataset. The code generates a correlation heatmap, providing a visual representation of the interdependencies between different features. This heatmap serves as a valuable tool in identifying significant correlations between various attributes, thereby facilitating the process of feature selection and offering deeper insights for further analysis. Moreover, the code presents the correlation of each feature with the target variable 'Bankrupt?' through a bar plot, aiding in the identification of the most influential features critical for the classification task.

Moving on to the Classification Modeling stage, the dataset is divided into training and testing sets using the train-test-split function from the sklearn.model-selection module. This step allows for the comprehensive evaluation of the classification model's performance. The code employs a Decision Tree Classifier, trained on the training data to predict the target variable 'Bankrupt?'. It evaluates the classification model's performance using essential metrics such as accuracy, precision, F1-score, and a comprehensive classification report, providing a holistic assessment of its effectiveness and efficiency.

978-1-6654-5992-1/22/$31.00 ©2022 IEEE

To address the challenge posed by class imbalance, the code incorporates the implementation of a Random Under-Sampling technique, ensuring a balanced distribution of the target variable 'Bankrupt?'. This strategic step significantly enhances the reliability and robustness of the classification model, contributing to more accurate and dependable predictions.

The final stage of Model Evaluation involves the visualization of a Receiver Operating Characteristic (ROC) curve. This graphical representation offers an insightful assessment of the classification model's performance, providing a comprehensive understanding of its ability to distinguish between the two classes. The ROC curve serves as a valuable tool in gaining insights into the overall efficacy and performance of the model.

In summary, the code exemplifies a comprehensive analytical framework encompassing pivotal stages such as data preprocessing, exploratory data analysis, classification modeling, and model evaluation. Its implementation is designed to provide meaningful insights into the effective management of imbalanced datasets, thereby facilitating the development of a robust classification model tailored to the unique nuances of financial data analysis.

II. Exploratory Data Analysis (EDA)

The data preprocessing phase serves as a fundamental cornerstone of the analysis, playing a pivotal role in ensuring the quality and integrity of the dataset. The code commences the data preprocessing process by importing the dataset from a 'data.csv' file, presumably housing financial data for in-depth analysis. This initial step sets the stage for subsequent data refinement and exploration.

Following the dataset import, the code undertakes a meticulous data refinement process, eliminating the 'Net Income Flag' column. This strategic move is crucial for streamlining the dataset and focusing the analysis on the most relevant attributes, thereby enhancing the overall effectiveness of the subsequent analysis. Removing this column is a crucial data preprocessing step as it helps in minimizing noise and streamlining the dataset for better interpretability and accuracy in the subsequent modeling process.

Moreover, the code leverages the LabelEncoder for encoding the categorical features present within the dataset. By transforming categorical data into numerical form, the LabelEncoder facilitates the integration of categorical data

into the subsequent analysis pipeline, ensuring that the algorithms can effectively process and interpret the information. This transformation is essential as many machine learning algorithms require numerical data for modeling, and the LabelEncoder serves as an effective tool in achieving this data transformation.

Additionally, the code addresses the issue of missing values within the dataset by employing a mean imputation strategy. By replacing the missing values with the mean value of the respective column, the code ensures the preservation of the dataset's overall integrity and consistency. This approach is instrumental in maintaining the statistical properties of the data and preventing any potential bias that might arise from the presence of missing values during the subsequent stages of analysis and modeling.

In essence, the data preprocessing phase, as executed by the provided code, exemplifies a robust approach to enhancing the dataset's quality and usability. By addressing issues such as the removal of redundant columns, encoding categorical features, and handling missing values, the code lays a solid foundation for the subsequent stages of exploratory data analysis and classification modeling. This approach is crucial in ensuring that the dataset is adequately prepared for insightful analysis and accurate modeling, leading to more reliable and meaningful outcomes.

III. Decesion Tree Classifier

The presented code entails the application of a Decision Tree Classifier for a specific dataset, resulting in an accuracy of 1.0. The classifier's performance is evaluated using precision, recall, and F1-score metrics, with the classification report presenting a detailed overview of the model's predictive capabilities for both classes. The model's performance is further illustrated through the confusion matrix visualization, offering valuable insights into the true and predicted values and enabling a comprehensive understanding of the model's classification accuracy.

The code commences by executing a data split operation using the train-test-split function, enabling the division of the dataset into training and testing sets in a controlled and randomized manner. This critical step is fundamental in ensuring that the model's performance can be accurately evaluated on unseen data, thereby facilitating a more robust assessment of its generalization abilities. Subsequently, the code instantiates a Decision Tree Classifier, a popular and versatile algorithm known for its ability to handle both classification and regression tasks. By leveraging a tree-like model structure, the Decision Tree Classifier recursively segments the dataset into subsets based on the most discriminating features, thereby facilitating the identification of complex decision boundaries within the data.

Upon training the model on the training data, the code generates predictions for the test dataset using the predict function. The confusion matrix is then computed using the confusion-matrix function, providing a comprehensive summary of the model's classification performance. The confusion matrix visually illustrates the number of true positive, true negative, false positive, and false negative predictions, offering valuable insights into the model's classification accuracy and misclassification tendencies. Moreover, the confusion matrix visualization, depicted using a heatmap, provides a clear and intuitive representation of the model's predictive capabilities, enabling a comprehensive understanding of its performance across different classes.

The accuracy-score function is utilized to compute the model's overall accuracy, representing the proportion of correctly predicted instances in the test dataset. The resulting accuracy value of 1.0 signifies that the model achieved perfect classification performance on the test dataset, accurately predicting all instances' class labels. Additionally, the classification-report function is employed to generate a detailed report outlining the model's precision, recall, and F1-score for both classes. This comprehensive report offers insights into the model's predictive precision, its ability to correctly identify positive instances (recall), and the balanced harmonic mean of precision and recall (F1-score), providing a holistic assessment of the model's performance across various metrics.

In summary, the code exemplifies a comprehensive approach to classification modeling using a Decision Tree Classifier, with a thorough evaluation of the model's predictive performance using critical metrics such as accuracy, precision, recall, and F1-score. The code's implementation enables a detailed understanding of the model's classification accuracy and misclassification tendencies, thereby facilitating informed decision-making and further refinement of the modeling approach.

IV. ROC-Curve

The provided code snippet is used to generate a Receiver Operating Characteristic (ROC) curve, an essential tool in assessing the performance of a binary classification model. The ROC curve is a graphical representation of the trade-off between the true positive rate (sensitivity) and the false positive rate (1-specificity) across different probability thresholds. It is particularly useful for evaluating the discriminatory ability of a model and determining its effectiveness in distinguishing between the positive and negative classes.

The 'roc-curve' function from the sklearn.metrics module is employed to compute the true positive rate, false positive rate, and corresponding thresholds based on the model's predictions. These values are then utilized to plot the ROC curve using the 'plot' function, with the false positive rate plotted on the x-axis and the true positive rate on the y-axis. The plotted curve visually represents the model's performance in terms of its ability to correctly identify positive instances (true positives) and its tendency to misclassify negative instances as positive (false positives).

In the plot, the diagonal 'k–' line represents the ROC curve for a random guessing model, serving as a baseline reference for comparison with the actual model's performance. The area under the ROC curve (AUC) is a commonly used metric to

quantify the overall performance of a classification model. A model with a higher AUC value generally indicates better discrimination ability and overall predictive performance.

The resulting plot provides a comprehensive visualization of the model's ability to differentiate between the two classes, enabling a clear assessment of its classification performance across various probability thresholds. The 'ROC Curve' title succinctly denotes the purpose of the plot, and the labels and legend added to the plot enhance its interpretability, facilitating a comprehensive understanding of the model's discriminatory power and overall effectiveness.