Python Assignment

M.K.S.Roshan
220633

1.Methodology

In this section, we outline the methodology employed for Preprocessing the data set and preparing it for model training.

1.1 Data Pre-processing steps

Data Preprocessing is a fundamental step in the data preparation pipeline for machine learning and data analyzing tasks. It involves transforming raw data into a clean, organized, and structured format that is suitable for analysis or model training. The primary objective of data Preprocessing is to ensure that the data is of high quality, consistent to be used by machine learning algorithms.

1.1.1 Currency Conversion

(Used in code under the function `currency_to_num`)
Currency conversion is performed to harmonize financial data across the data set. By converting currency strings into numeric values, this process establishes consistency and facilitates seamless integration of financial metrics into the analysis pipeline.

1.1.2 Missing Value Imputation

(Used in code with `SimpleImputer`)
Missing values within pertinent columns, such as 'Total Assets' and 'Liabilities', are diligently handled using mean imputation techniques. This ensures the dataset remains complete and robust for subsequent analyses.

1.1.3 One-Hot Encoding

(Used in code with `pd.get_dummies`)
Categorical variables undergo one-hot encoding, a pivotal Pre-processing step to transform them into binary indicators. This process ensures categorical data can be effectively utilized by machine learning algorithms without introducing spurious ordinal relationships.

1.2 Feature Engineering Steps

Feature engineering is the process of creating new features or transforming existing features in a dataset to improve the performance of machine learning models. It involves extracting meaningful information from the raw data and representing it in a way that enhances the model's ability to learn patterns and make accurate predictions. Feature engineering is a crucial step in building effective machine learning models, as the quality of features directly impacts the model's performance.

1.2.1 Discretization

(Used in code with `SimpleImputer`)
Continuous variables such as 'Total Assets' and 'Liabilities' undergo discretization into discrete intervals or bins. This simplifies the data, enhances model interpretability, and facilitates effective handling of outliers.

2.Experiment details

In this section, we provide details about the classifiers used for model training and evaluation.

Model:-

The classifiers utilized for model training are as follows:
- Bernoulli Naive Bayes (BernoulliNB)

Hyperparameters:-
The hyperparameters utilized for each classifier are as follows:
- BernoulliNB:-   Alpha: [0.1, 0.5, 1.0, 2.0]

These classifiers were trained using the training data and evaluated based on their performance metrics on the validation data (10% of the train data). The tuned hyperparameters were found using GridSearchCV.

The best score was obtained by the Bernoulli Naive Bayes model with the given parameters.

Bernoulli Naive Bayes is a probabilistic classification algorithm suited for binary feature vectors. It assumes that all features are binary-valued (Bernoulli, boolean), and thus it's particularly appropriate for features that take on only two values, such as presence or absence.

Hyperparameters:-

Alpha (α): It's a smoothing parameter used in Naive Bayes models. By adjusting alpha, the model can be regularized, effectively reducing overfitting. The range of alpha values tested during hyperparameter tuning was [0.1, 0.5, 1.0, 2.0].

Train-Test Split:

The dataset was split into training and validation sets using the train_test_split function from scikit-learn. The test_size parameter was set to 0.1, indicating that 10% of the training data was reserved for validation. Additionally, the random_state parameter was set to 49 to ensure reproducibility of results.

Cross-Validation (CV) in GridSearchCV:

Hyperparameter tuning was performed using GridSearchCV with a cross-validation (CV) value of 9. This means that the dataset was divided into 9 folds for cross-validation during the grid search process. Cross-validation helps in obtaining more reliable estimates of the model's performance and ensures robustness of the selected hyperparameters.

GIT LINK FOR CODE:-

https://github.com/Roshan-haisha/Assignment-python-