Pre-processing & Training Data

Overview of Preprocessing Workflow

The preprocessing stage is crucial in the predictive modeling process, aimed at refining the dataset to optimize the performance of subsequent models. This involves handling class imbalances, inputting missing values, encoding categorical variables, and scaling features. The data_transformation.py script, following on from data_ingestion.py, manages these tasks by accessing data stored in intermediate pickle files, specifically utilizing dataset number 24 out of 26 datasets that have been processed thus far. This script represents the 25th experiment out of a series of 29, each exploring different preprocessing configurations.

Feature Selection and Splitting

Irrelevant features are pruned from the dataset to concentrate on those variables that directly impact the target variable ‘isMissionSuccess’. The data is then divided into training and testing sets, maintaining a test size of 5% to optimize the amount of data available for training while ensuring enough data for effective model validation.

Handling Data Imbalance

Given the significant class imbalance noted in the dataset, a RandomOverSampler is employed to equalize the class distribution in the training data by oversampling the minority class. This adjustment is crucial for preventing model bias toward the majority class.

Data Transformation Strategy

The preprocessing phase aims to appropriately handle each feature based on its characteristics, such as data type and distribution. Here's a closer examination of the key features and the rationale for their specific preprocessing approaches:

1. 'Rocket Cost':
    - Data Type and Distribution: This feature is a continuous numerical variable with a wide range from a minimum of $5.3 million to a maximum of $5 billion, indicative of the varied costs associated with different rocket launches.
    - Imputation Strategy: Given the skewed nature of this distribution, median imputation is utilized to handle missing values, with a median value of $59 million. This approach is chosen to avoid the influence of extreme values on the imputation process, ensuring a more robust and representative fill value for missing data.
2. 'Year', 'Month', 'Day':
    - Data Type: These are discrete numerical features. 'Year' ranges from 1957 to 2020, 'Month' from 1 to 12, and 'Day' from 1 to 31.
    - Scaling: StandardScaler is applied to these features to normalize their scale. Although these are inherently discrete, their numerical nature and range make them suitable for scaling to ensure they contribute equally when modeling, avoiding any potential scale bias that could affect the algorithm's performance.
3. Geographical Coordinates ('Company Origin Lat', 'Company Origin Long'):
    - Data Type: Continuous numerical variables representing latitude and longitude.
    - Scaling: Given their float data type and range from negative to positive values, scaling these features is crucial. StandardScaler is used to ensure these geographically based measurements do not overpower other variables in model calculations due to their range and scale.
4. 'Unix Time':
    - Data Type: Continuous numerical variable, ranging from -0.4e9 to 1.6e9.
    - Scaling: The significant range of Unix Time values, which represent timestamps, benefits from scaling to normalize these values within a comparable range to other features, enhancing model accuracy and learning efficiency.
5. 'YearSine':
    - Data Type: Continuous numerical variable, ranging from -1 to 1.
    - Scaling: Although already within a standard range, applying StandardScaler helps maintain consistency in preprocessing across all continuous numerical variables, ensuring that this cyclic feature integrates well with other features in the model.
6. Categorical Variables ('Status Rocket', 'isWeekend'):
    - Data Type and Categories: 'Status Rocket' has categories like StatusActive and StatusRetired; 'isWeekend' is a binary feature with values True or False.

- Encoding: These features are transformed using OrdinalEncoder to convert them into a numerical format suitable for machine learning models. This encoding is straightforward as the features are binary or have a clear order, making them ideal for ordinal encoding.

Exclusion of OneHotEncoder Features:

- No features that required OneHotEncoder proved useful in enhancing the model's performance metrics, leading to their exclusion from this experiment. This decision highlights the importance of feature selection in optimizing model outcomes, ensuring only impactful features are included in the final model.

Pipelines and Column Transformers

Each preprocessing step is encapsulated within pipelines and combined in a column transformer. This structured approach ensures that transformations are applied consistently and are easily traceable, streamlining the preprocessing for model consumption.

Parameter Tracking and Documentation

All preprocessing steps, transformations, and configurations are meticulously documented in a master parameters dictionary, which includes details of all column transformations. This dictionary is saved in a YAML file alongside the processed datasets, facilitating reproducibility and transparency in the preprocessing decisions.

Exporting Processed Data

The processed data is exported as pickle files, ready for use in the modeling stage. This ensures that the preprocessing and modeling stages are distinct, promoting modularity and ease of adjustments in the analysis workflow.

Conclusion

By meticulously preparing the dataset through the outlined preprocessing steps, this script sets a robust foundation for developing predictive models. It addresses key challenges such as data imbalance, missing values, and feature encoding, crucial for enhancing the accuracy and effectiveness of the predictive models. The detailed documentation and parameter tracking underscore the project's commitment to reproducibility and clarity, which are essential for strategic decision-making by stakeholders in the space exploration industry.