Job Title Prediction with Machine Learning - Comprehensive Report

This report details the development of a machine learning model for predicting job titles using data from a CSV file (function_assignment.csv). It outlines the data exploration, preprocessing, feature engineering, model selection, validation, and analysis stages, adhering to professional standards in machine learning practice.

Data Exploration and Preprocessing

Findings:

- The initial data exploration phase utilizes functions like data.head() and data.info() to gain insights into the data structure (columns, data types, missing values). This understanding is crucial for selecting appropriate data cleaning and preprocessing techniques.

Preprocessing Steps:

1. **Missing Value Handling:**
   - To prevent errors during text processing, missing values in the jds column are replaced with a placeholder string like "No description available." This ensures consistent data handling throughout the pipeline.
2. **Categorical Variable Encoding:**
   - The categorical variable function is encoded using pd.Categorical and .cat.codes. This transforms categorical labels into numerical features suitable for machine learning algorithms.
3. **Text Cleaning:**
   - A dedicated function clean_text performs the following text cleaning operations on job descriptions (jds):
     - Lowercasing text for consistency in word representation.
     - Removing punctuation to focus on the core content of the text.
     - Removing stop words (common words like "the", "a") using stopwords.words("english") to reduce noise and improve model performance.
     - Applying stemming (PorterStemmer) or lemmatization (optional) to reduce words to their base form and capture variations.

Feature Engineering

Extracted Features:

1. **TF-IDF:**
   - The prepare_features function leverages TfidfVectorizer to generate TF-IDF features. These features represent the relative importance of words within a job description compared to the entire dataset. Words that are specific to a particular job title will have higher TF-IDF scores in those descriptions.
2. **N-grams:**
   - The function also extracts bigrams (2-word sequences) and trigrams (3-word sequences)

using nltk.util.ngrams from the cleaned job descriptions. These can capture skills and phrases frequently mentioned together, potentially indicative of specific job requirements.

3. **Description Length:**
    - The function calculates the length of each cleaned job description. This feature might be relevant for some job roles, where a longer description could indicate greater job complexity.

Rationale Behind Chosen Features:

- TF-IDF and n-grams directly target the content and terminology used in job descriptions, which is highly informative for predicting job titles.
- Description length can be a supplementary feature that adds context for certain job roles.

4. **Data Split**
    - The preprocessed data is subsequently split into training and testing sets using train_test_split from scikit-learn. This allows the model to learn from the training set and be evaluated on unseen data from the testing set, providing a more robust assessment of its generalization capability.

Modeling

Chosen Model:

- **Random Forest Classifier:** This report uses Random Forest Classifier as an illustrative example. It is a robust ensemble method known for its effectiveness in text classification tasks. Other options include Logistic Regression, Support Vector Machines (SVM), or Deep Learning models (LSTM, CNN) for potentially better performance (requires more data and computational resources).

Hyperparameter Tuning (not shown):

- Techniques like GridSearchCV or RandomizedSearchCV from scikit-learn can be employed to find optimal hyperparameters for the chosen model. Hyperparameters control the model's behavior and can significantly impact performance.

Validation

Train-Test Split:

- As mentioned earlier, the preprocessed data is split into training and testing sets using train_test_split. The model is trained on the training set and evaluated on the testing set.

Evaluation Metrics:

- The model's performance on the testing set is evaluated using various metrics:
    - **Accuracy:** Proportion of correctly predicted job titles, providing a general performance indicator.
    - **F1 Score:** Harmonic mean of precision and recall, considering both true positives and

false negatives, offering a more balanced view of performance.

Additional Considerations:

- **Label Encoding**
  - If the job title labels are strings, they need to be encoded using LabelEncoder from scikit-learn before feeding them to the model. This converts string labels into numerical representations suitable for the model.
- **Handling Object Columns**
  - If any features remain as object data type (strings) after preprocessing, consider removing them or converting them to numerical representations using appropriate techniques to ensure compatibility with the model.
-