Machine Learning Engineer Task: Resume Categorization

Objective: Design, implement, and train a machine learning model to automatically categorize resumes based on their domain (e.g., sales, marketing, etc.). Following this, develop a script that can be run via the command line to process a batch of resumes, categorize them, and output results to both directory structures and a CSV file.

Approach:

Initially, I have downloaded the dataset from Kaggle, and perform Exploratory Data Analysis (EDA) to understand the pattern of the data. EDA also allows to take decision whether it needs any necessary pre-processing steps. All of my experiments and data can be found in this drive folder. My workflow can be divided into some parts that is discussed below:

EDA and exploring machine learning (ML) algorithm: I have performed some basics EDA and found dataset is slightly imbalanced for 3 classes. I did not address this issue; it might not have that much impact on the final classification. If the ratio was very high, then I would perform SMOTE or other oversampling technique to address this issue. After that, I explored some machine learning algorithm including K-nearest neighbors (KNN), random forest (RF), and XGBoost. I have utilized TF-IDF Vectorizer to extract features from the textual data before classification. From experimental results, I found that XGBoost outperformed other two classifiers. KNN reported low accuracy and took larger training time compared to others. The experimental results of these classifiers are represented in table 1:

Classifier	Accuracy	Precision	Recall	F1 score	Septicity
KNN	0.51	0.55	0.51	0.49	-
RF	0.61	0.62	0.61	0.58	0.98
XGBoost	0.75	0.75	0.75	0.74	0.98
XGBoost (tuned)	0.76	0.78	0.77	0.76	0.98

Table-01: Results comparison on test data of ML models

The experimental results demonstrated the performance of fine-tuned XGBoost. RF was also fine-tuned by applying grid search cross-validation to find the best parameters. The experiments can be found in this notebook.

File handling based on requirements: After exploring ML algorithms, I started to work for the file handling that will be needed to develop script. Here, I separated train and test data into two folders, and wrote all necessary functions to develop the script. I also classified test data with the trained fine-tuned XGBoost in this experiments, the notebook is here.

Exploring deep learning (DL) model: LSTM with two different variants was employed to determine the performance of LSTM in this dataset. I got very bad classification results and huge loss values for this experiment. The experiment is conducted in this notebook. In the beginning of this notebook, I have separated train set into validation and train 80:20 ratio, extracted text data from the resumes, cleaned those raw texts, and save into two different csv files. I also converted test data into csv file. From here, I have made three different csv dataset for training, validation and test.

Final model implementation and script development: After those experiments, I found XGBoost is the best performer model in our data. I have chosen this model as the final model. In this final experiment, I

also applied 'Chi-square' feature selection technique to discard redundant features. This technique takes the relevant features, which boost the performance as well as reduce computational cost. After feature selection, the accuracy improved without fine tuning the model. The experimental results are presented below:

FS	Tuning	T. acc.	V. acc.	AUC	MCC	SP	SN	Precision	F1 Score
No	No	1.00	0.73	0.97	0.72	0.98	0.74	0.75	0.74
	Yes	0.81	0.74	0.97	0.72	0.98	0.74	0.76	0.73
Yes	**No**	**1.00**	**0.76**	**0.97**	**0.75**	**0.98**	**0.77**	**0.78**	**0.76**
	Yes	0.82	0.75	0.97	0.74	0.98	0.75	0.77	0.77

Table-02: Final experimental results of the selected XGBoost model

The experimental results based on various evaluation metrices demonstrates that feature selection technique has positive impact on the final classification results. Here, T.acc. stands for training accuracy and V.acc. stands for validation accuracy. MCC is the Matthew's correlation coefficient, SP is specificity, SN is sensitivity or recall. We can see that, this model obtained a higher AUC score, which indicates true positive and false positive rates of the model. Besides this, I have also plotted log-loss and classification error plots for both training and validation data. The experiment can be found in this notebook. This model achieved an accuracy of 71.48% for the test data, other evaluation metrices are presented in the notebook. I have saved the feature extractor, vectorizer, and this model to develop the script.

Based on the performance of validation set with different metrices, XGBoost demonstrates its superior performance compared to other classifiers. Therefore, I have chosen this as our final model. Deep learning model might perform better than this model, but I was unable to explore more DL models due to lacking o time. I have run the script, and it has worked perfectly based on your requirements. My GitHub repository contains all the files and codes during the experiments. A csv file is also added in the repository that has generated after running the script.

Instructions to run the code

Model folder and `script.py` should be kept in same directory to run perfectly the script from the command line. Package requirements are provided in the `requirement.txt` file in the repository. You need the provide the test data path, where uncategorized resumes are kept, it will load those resumes and categorize them into predicted category folder. It will also create a csv file that will have two columns, one is for file name and another one is for predicted category.