REPORT ON VNB FoML 2024 Hackathon

POOJA VINOD MANE(CS22BTECH11035)-

We used Random Forest classification for training the VNB dataset.

MY CONTRIBUTION

Importing the essential libraries such as pandas, numpy, and sklearn. Used pd.read_csv() to read the train and test datasets from the appropriate CSV file.
The training data is split into features (x_train) and the target variable (y_train) using train_test_split libraries from sklearn, while the target is encoded using LabelEncoder to convert categorical labels into numeric values. The data is divided into training and validation sets, which means that such a model can be tested in unseen data for the point of performance while training data. The features in the training and validation sets are standardized by subtracting the mean and dividing by the standard deviation (calculated from the training data).:The predicted labels are converted back to their original categorical form using inverse_transform of the LabelEncoder.

BOTH : **Random Forest Model:** A Random Forest classifier is created with specific hyperparameters: 90 trees, a maximum depth of 8, and using the square root of the number of features for each split. The model is trained using the standardized training data.
We tried to do it by Gradient Boosting Classification, but we were not able to get a good f1-score because gradient boosting requires more computational power, less robust while dealing with large dataset and risk of overfitting. It is slower to train compared to models like Random Forest because it builds trees sequentially, which also limits parallelization. Gradient Boosting consumes more memory as it stores intermediate models during training.
BOTH:A submission DataFrame is created with the UID column from the test data and the predicted target values. This is saved as a CSV file (submission.csv).

Key Points:

- **Preprocessing**: It drops columns with all entries are NaN. Missing values are handled with mean imputation, and features are standardized to ensure consistency across datasets.
- **Model**: Random Forest is used with balanced class weights to account for potential class imbalance in the target variable.
- **Performance**: The F1 score is used to evaluate model performance on the validation set, providing a balanced measure of precision and recall.
- **Predictions:** The function predictions() processes the test data and saves the predictions in the required format for submission file.

In Predict Submission, we submitted a total of 14 submission files. Where 7 of the submissions are done by Pooja(cs22btech11035) and 7 of the submissions are done by Surbhi(cs22btech11057). We observed that, at first we were getting a 0.358 score.
We used a row-wise function to modify the data frames and get row mean and row standard deviation. But got no improvement in the score. Earlier in our code we were not standardizing the data appropriately due to which we were getting low scores.
While calculating standardized data, the problem of zero data division was taken care of later, thus we standardized the data for consistency contribution. We have features that have vastly different units or scales so standardizing helps in improving model performance and training stability. In the training dataset given each column has a particular value type some are numeric values, some are high- value measurements, some are data or time indicators, some are encoded values, so standardizing them for model training becomes important. Subsequently, we also improvised our random forest classifier by using various parameters keeping in mind the overfitting problem.
By implementation of all above, we achieved a 0.425 score.