Mini project #1: Naive Bayesian Classifier--implementation and evaluation.

By Thanoj Muddana (923669081)

1. **Compile Code and Run the Program:**
   I've implemented the project using Python 3.9 interpreter in Visual Studio Code. Attaching .py file and data source file below.
   Utilized **pandas** Library for dataframe Operations and **numpy** for randomizing arrays/series.

2. **A brief discussion on the main methods adopted in the implementation:**
   User-Defined Functions (UDFs):
   udf_binning: This function is used to bin continous columns into discrete bins. It calculates the minimum and maximum values of a column, creates bin boundaries based on a specified bin size or custom thresholds, and applies binning to create a new column with bin labels.
   replace_unknown_with_mode: This function replaces missing values (marked as '?') in a column with the mode of that column, excluding the missing values.

   Observations: Total rows ~ 32561
   a) Work-Class (5.62% ~ 1836 missing values)
   b) Occupation (5.66% ~ 1843 missing values)
   c) Native Country (1.79% ~ 583 missing values)

   calculate_prior_probability: Calculates all different combinations of feature/class label , class label probabilities based on the training data.
   calculate_class_probability: Computes the class probabilities.
   calculate_likelihood_probability: Calculates the likelihood probabilities for each feature and class label.
   predict: Predicts the class label for a given set of features using the Naive Bayes classifier.
   apply_laplace_smoothing: Applies Laplace smoothing to handle zero likelihood records.
   Functions for calculating accuracy, precision, recall, F1 score, and confusion matrix.

   Data Preprocessing:
   The code reads the Adult dataset from a CSV file and sets column headers and data types according to a data dictionary.
   It performs data preprocessing tasks such as binning of numerical columns based on inputs from EDA done in Power BI and handling missing values using mode imputation, and selecting only required columns columns for the Naive Bayes implementation.

   Train-Dev-Test Splitting:
   The code splits the data into training (70%), development (20%), and testing (10%) sets. This split is performed by randomizing the dataset and selecting rows based on predefined ratios.

   Naive Bayes Model Implementation:
   The Naive Bayes algorithm is implemented using the functions defined earlier. It calculates prior probabilities, class probabilities, likelihood probabilities, and applies Laplace smoothing.

It then makes predictions on the test set and evaluates the model's performance in terms of accuracy, precision, recall, F1 score, and confusion matrix.

K-Fold Cross-Validation:

The code performs 10-fold cross-validation to assess the model's generalization performance. It shuffles the data and iterates through the folds, calculating accuracy for each fold and averaging them to obtain the final cross-validation accuracy.

3. **A brief description/discussion of your evaluation strategies:**

code implements two main evaluation strategies to assess the performance of the Naive Bayes classifier with evaluation metrics like accuracy, precision, recall value, F1 value & confusion matrix with a train-dev-test split strategy and K-fold cross-validation.

Train-Dev-Test Split Evaluation:

I've split the source data into 3 subsets for Training (70%), development (20% - to perform performance improvements) and test (to test the model). Using training data, I've calculated Likelihood & Class probabilities using training dataset and used them to calculate P(features/class label) , for the provided features, Bayes Implementation will refer to the conditional & class probability for both classes and returns the class label with maximum probability. Laplace smoothing is applied on the likelihood probabilities dictionary to handle 0 probability scenarios as we'll multiply it for calculating final probability of class label.

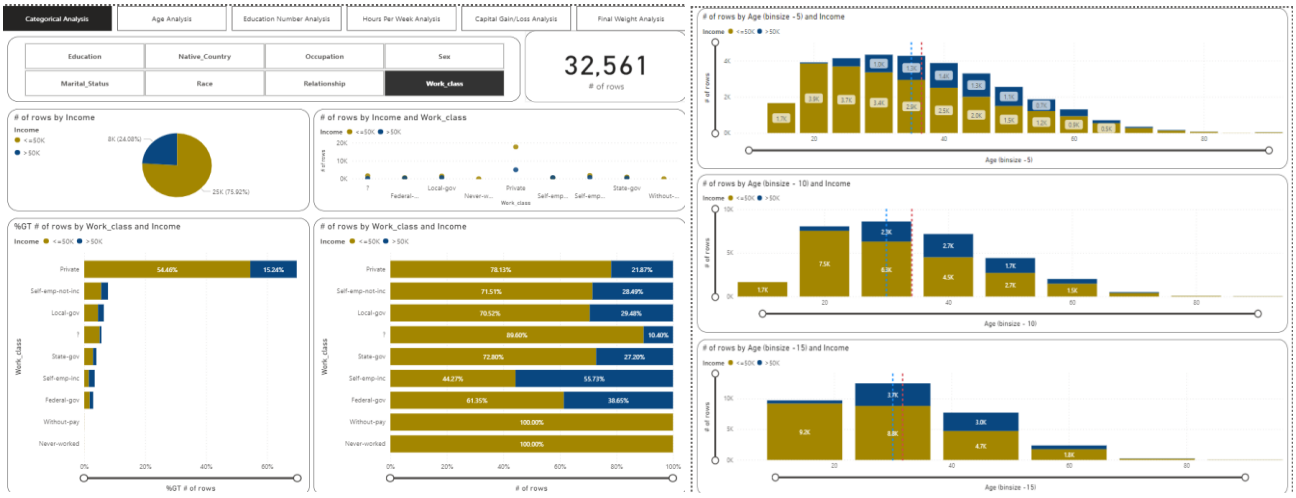
K-Fold Cross-Validation:

K-fold cross-validation is used to assess the model's performance by dividing the dataset into K subsets (folds). The model is trained and evaluated K times, with each fold serving as the test set once and the remaining K-1 folds used for training. In the provided code, K-fold cross-validation is used to assess the model's generalization performance, and the average accuracy over the K folds is reported. This approach provides a more robust estimate of how well the model is likely to perform on unseen data compared to a single train-dev-test split.

4. **Results analysis and suggestions for the next steps.**

During pre-processing of data, I've used Power BI for data visualization, below are some of the key observations from EDA.

Analyze Bin Size for continuous attributes:



Age:
Result: Almost symmetric with a right skew.
Suggestion: Bin size of 5 provides a good balance.
Education Number:
Result: Almost symmetric.
Suggestion: Bin size of 4 captures the distribution well.
Hours Per Week:
Result: Bin size of 10 is reasonable. Almost symmetric around Mean/median.
Capital Gain:
Result: Bin size of 2000 with outlier grouping is effective for >34000.
Capital Loss:
Result: Bin size of 50 is appropriate. Symmetric around mean.
Final Weight:
Result: Bin size of 25k with outlier grouping works well for >50000.

After discretizing continuous attributes, train_dev_test strategy is implemented and below are evaluation metrics for Naïve Bayes Model.

Results:

Accuracy: 0.819 (approximately 81.9%)

Precision: 0.596 (approximately 59.6%)

Recall: 0.786 (approximately 78.6%)

F1 Score: 0.678 (approximately 67.8%)

Confusion Matrix:

True Positives: 621

True Negatives: 2046

False Positives: 421

False Negatives: 169

K-Fold Average Accuracy: 0.820 (approximately 82.0%)

Next Steps/Suggestions:

1) Address class imbalance (>50k Class label record (24% approx.)) : apply techniques like oversampling or undersampling to handle class imbalances.
2) Evaluate other metrics: Consider other evaluation metrics like MCC, ROC-AUC to gain a comprehensive view of model performance.
3) Adjust the decision threshold for classification. By default, Naive Bayes uses a threshold of 0.5, we can experiment with different threshold values to achieve a balance between precision and recall.
4) Ensure your data preprocessing steps, such as handling missing values and encoding categorical variables, are optimized for your specific dataset.
5) Examine the specific cases where your model is making errors. This can provide insights into areas where improvements are needed and guide your optimization efforts.