PROJECT REPORT

Illinois Institute of Technology

Professor Jawahar Panchal

Thi Thuy Dieu Cao

A20516306

CS 422 Data Mining

1. **Abstract**

This project focuses on developing a machine learning model using a Scikit-Learn based pipeline for training and an ONNX based model for deployment. The model used is a Decision Tree Classifier. The dataset was classified into three distinct categories. The initial performance evaluation of the model indicates moderate accuracy with some room for improvement. Future work includes experimenting with different models and fine-tuning hyperparameters to enhance classification performance.

2. **Overview**
   - Problem Statement

     The primary objective of this project is to develop a machine learning model capable of classifying a dataset into three distinct classes. The model is to be trained using Scikit-Learn and deployed in an edge environment using ONNX, which allows for efficient inference.
   - Relevant Literature

     Decision Tree Classifiers are renowned for their straightforward nature and interpretability, making them a popular choice for classification problems. They create a model by splitting the data into subsets based on feature values, which can be visualized as a tree structure. Despite their advantages, Decision Trees are susceptible to overfitting, particularly with complex datasets. Techniques such as pruning, setting maximum depth, and minimum samples per leaf are commonly employed to mitigate overfitting and enhance generalization. Feature selection based on feature importance can also improve model performance by reducing complexity and focusing on the most informative features.
   - Proposed Methodology
     1. Data Preprocessing:
        - Impute missing values using mean imputation.
        - Standardize features with StandardScaler to ensure consistent scaling.
     2. Model Training:
        - Train a Decision Tree Classifier using the preprocessed data.
        - Use feature importance to identify and select the most informative features.
        - Re-train the model using the selected features.
     3. Model Conversion:
        - Convert the trained Scikit-Learn model to ONNX format for deployment.
     4. Performance Evaluation:
        - Evaluate model performance using precision, recall, and F1-score.

3. **Data Processing**
    - Pipeline details

      The preprocessing pipeline includes the following steps:
        - Imputation:
            + Missing values are filled using the mean of the respective features if there is any. This approach ensures that no data is lost and maintains the statistical integrity of the dataset.
        - Scaling:
            + Features are standardized to have a mean of 0 and a standard deviation of 1. Standardization is crucial for many machine learning algorithms to perform optimally.
    - Data Issues
        - Missing values: There is no missing value in the dataset.
        - Class imbalance: The dataset exhibits a notable imbalance in class distribution, which may impact model performance.
    - Assumptions/ Adjustments
        - Dataset representativeness: It is assumed that the dataset is representative of the problem space and contains all relevant features for classification.
        - Feature informativeness: The features are assumed to be informative enough to distinguish between the classes.

4. **Data Analysis**
    - Summary Statistics

      The dataset comprises three classes with the following sample distribution:
        - Class 1: 180,594 samples
        - Class 2: 449,885 samples
        - Class 3: 569,521 samples
    - Visualization

      Exploratory data analysis included visualizations such as:
        - Histograms: to understand the distribution of individual features.

    - Feature extraction

      Original Feature: No additional features extraction was performed yet, the model utilized the original features provided in the dataset.

5. **Model Training**
    - Feature Engineering:

      Standardization: Features were standardized to improve model performance and ensure that all features contribute equally to the decision-making process.

Feature Selection: After the initial training of the Decision Tree Classifier, feature importance was analyzed to identify the most informative features. The model was then re-trained using only these selected features to potentially enhance performance and reduce complexity.

- Evaluation Metrics:
  - Precision: Measures the proportion of true positive predictions among all positive predictions made.
  - Recall: Indicates the proportions of true positive predictions out of all actual positives
  - F1-Score: The harmonic mean of precision and recall, providing a balanced measure of model performance.
- Model Selection:
  Decision Tree Classifier: Chosen for its simplicity, ease of interpretation, and ability to handle both numerical and categorical data.

6. Model Validation

- Test Results
  After performing feature selection and re-training the model, the Decision Tree Classifier yielded this best performance metrics on the test set:
  Class 1:
  - Precision: 0.37
  - Recall: 0.38
  - F1-Score: 0.37
  - Support: 36,189

  Class 2:
  - Precision: 0.75
  - Recall: 0.82
  - F1-Score: 0.78
  - Support: 89,845

  Class 3:
  - Precision: 0.63
  - Recall: 0.57
  - F1-Score: 0.60
  - Support: 113,966

  Overall:
  - Accuracy: 64%
  - Macro Average Precision: 0.58
  - Macro Average Recall: 0.59
  - Macro Average F1-Score: 0.58
  - Weighted Average Precision: 0.63
  - Weighted Average Recall: 0.64

- Weighted Average F1-Score: 0.63

- Performance Criteria
  The model's performance was assessed using classification metrics, with a focus on the balance between precision, recall, and F1-Score.
- Biases/ Risks
  Overfitting: There is a risk of the model overfitting the training data, which may result in reduced performance .

7. Conclusion

- Strengths:
  - Class 2 Performance: The model performs well on Class 2, with high precision (0.75), recall (0.82), and F!-Score(0.78). This indicates that the model is quite effective at identifying instances of Class 2.
  - Balanced Performance: The weighted averages (Precision, Recall, F1-Score) are relatively balanced around 0.62, indicating a fair performance across the classes.
- Weaknesses:
  - Class 1 Performance: The model struggles with Class 1, showing lower precision (0.37), recall (0.38), and F1-Score (0.37). This suggests that the model is not very effective at identifying instances of Class 1, possibly due to a smaller sample size or less distinguishable features for this class.
  - Class 3 Recall: The recall for Class 3 is lower (0.57), indicating that the model misses a significant number of Class 3 instances.
- Improvement areas:
  - Class imbalance: The dataset shows an imbalance with Class 1 having significantly fewer samples compared to Classes 2 and 3. Techniques such as Synthetic Minority over-sampling Technique or re-sampling can be employed to balance the dataset.
  - Advanced Models: Consider using more advanced models like Random Forests, Gradient Boosting, or even ensemble methods to improve performance.
  - Hyperparameter Tuning: Further hyperparameter tuning could help in finding a better model configuration that improves overall performance.
  - Feature Engineering: continue refining feature selection and engineering to ensure that the most informative features are used.

8. Data Sources

The data used in this project was provided by Professor Jawahar Panchal from the Illinois Institute of Technology as part of the CS422 Data Mining course.

Description of the Dataset:
Number of Features: The dataset comprises 15 features labeled from A to O.
Classes: The dataset includes three classes:

- Class 1
- Class 2
- Class 3

Sample Distribution: The classes have the following distribution:

- Class 1: 180,594 samples
- Class 2: 449,885 samples
- Class 3: 569,521 samples

9. Source Code

The source code for this project is organized into several sections, each corresponding to different stages of the data processing, model training, and evaluation pipeline. For full detailed implementation, please refer to the jupyter notebook file attached and named "ONNXProject.jpynb".

10. Bibliography

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830. Retrieved from https://scikit-learn.org/stable/documentation.html

ONNX. (n.d.). Open Neural Network Exchange (ONNX). Retrieved from https://onnx.ai/

Guyon, Isabelle & Elisseeff, André. (2003). An Introduction of Variable and Feature Selection. J. Machine Learning Research Special Issue on Variable and Feature Selection. 3. 1157 - 1182. 10.1162/153244303322753616.

J. R. Quinlan. 1996. Learning decision tree classifiers. ACM Comput. Surv. 28, 1 (March 1996), 71–72. https://doi.org/10.1145/234313.234346