Machine Learning Assignment 1

1st Ming-Xuan Wu
Institute of Data Science
National Cheng Kung University
Tainan, Taiwan
Email: RE6124019@gs.ncku.edu.tw

***Abstract*—This paper delves into a comprehensive investigation of supervised learning in machine learning, conducting in-depth research and exploration. Through a series of practical implementations, the study focuses on the analysis of a dataset obtained from Kaggle, specifically the Automobile Insurance Claims Prediction dataset. The objective is to analyze the data and predict whether policyholders are likely to file claims within the next 6 months. In the initial phase, data preprocessing is undertaken to facilitate subsequent tasks. During the implementation, the Perceptron is employed as the Linear Classifier, and training and prediction for KNN (K-Nearest Neighbors) and Decision Trees are realized. The paper also presents the task of feature engineering, wherein feature importance analysis is conducted to identify crucial features. Through feature fusion, the model's accuracy is enhanced. Finally, Cross-Validation is employed to assess the stability of the model, and robust algorithms are proposed to optimize model performance.**

***Index Terms*—Machine Learning, supervised learning, Classification, Feature engineering, Cross-Validation.**

I. Introduction

In this study, we undertake the task of implementing and evaluating several fundamental machine learning classifiers. The primary objective is to compare the performance of these classifiers and gain insights into their strengths and weaknesses. The classifiers under consideration include a basic linear classifier, a k-nearest neighbors (K-NN) classifier employing three distinct distance metrics, a naive decision tree classifier, and a decision tree classifier with a pruning algorithm.

Additionally, we explore feature engineering by developing an algorithm to assess "feature importance" for both linear classifiers and decision trees.To assess the classifiers' performance, we employ a carefully chosen evaluation criterion and validate the models using k-fold cross-validation. The evaluation criterion, yet to be specified, plays a crucial role in determining the effectiveness of each algorithm. Furthermore, we propose an algorithm to derive new features, acknowledging the potential limitations of the original feature set in improving model accuracy.To gain deeper insights into feature importance, we integrate the SHAP (SHapley Additive exPlanations) library with our implemented algorithms. This enables a comparative analysis between our findings and those obtained through SHAP, offering a comprehensive understanding of feature relevance.

Finally, we address the challenge of aggregating predictions from k classifiers in k-fold cross-validation. We design an algorithm for merging and aggregating these predictions, comparing the performance and complexity with the results obtained in the initial problem. To establish the true superiority of one model over another, we compare the results of 5-fold cross-validation with those obtained in a single train-test split scenario (Problem 1). This comparative analysis allows us to justify the robustness and reliability of the chosen classifiers, shedding light on their generalization capabilities and overall performance.

II. Method

A. Classification Task

1) Linear Classifier (Perceptron)
In the linear classifier, I employ the perceptron [1] as the foundational model construction class. This class encompasses essential parameters such as weights (w), bias (b), and learning rate (lr), with the default number of training epochs set to 100. Various methods are incorporated within the class, including 'predict,' which utilizes the fitted regression line for classification predictions. Additionally, there are methods for computing evaluation parameters and generating plots. Of utmost significance is the 'fit' method, wherein matrix operations are utilized to adjust the weights (w) and bias (b) in a manner that optimally converges for expedited learning. The training and validation dataset plot is illustrated in Figure 1.

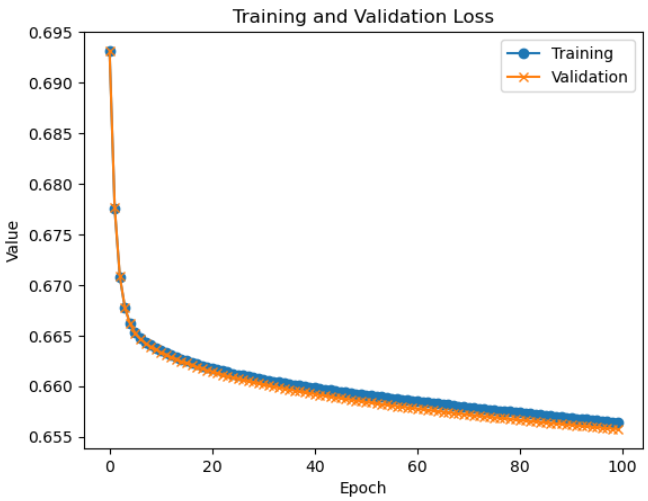

Fig. 1: Training/Validation loss plot

2) K-NN Classifier (Use K-D tree)
In the K-NN classifier [2], three classes have been implemented: KDNode, KDTree [3], and KNN. KDNode is responsible for the node structure in constructing the KDTree. Due to the computational intensity involved in calculating distances for all data points and identifying the nearest K neighbors in traditional K-NN, the speed can be significantly slow. To address this, a KDTree is employed to establish a hierarchical structure, facilitating the expedited identification of points with similar properties for subsequent distance calculations.
In the proposed methodology outlined in this paper, a limitation is imposed on the number of points within the tree leaves, specifically restricted to K*100, aiming to accelerate the search and distance computation process. The primary algorithm, KNN, incorporates essential parameters, including the number of nearest neighbors (K), and stores the constructed tree (tree).
The main methods within the class include 'fit,' which involves the construction of the tree through the KDTree, and 'predict,' which utilizes the tree to identify the most similar K*100 points for distance computation. Subsequently, the algorithm determines the K nearest points and assigns the predicted class based on the majority class among these points.
3) Naïve Decision Tree Classifier
In the implementation of the decision tree [4] [5], I adopted a straightforward approach. In this method, the selection of the optimal feature is accomplished through the _best_split method, followed by the construction of the entire tree structure using the _build_tree method. At each branching step, the data is divided into two categories. In the predict method, the data to be predicted is input into the tree structure for classification, yielding the final predicted category.
4) Decision Tree with Pruning [6]
In the pruned decision tree, the overall structure is largely similar to a conventional decision tree, with the primary distinction lying in the pruning process. This method involves a reassessment of the tree structure and a recalculation of the Gini index for the data beneath each node. If it is determined that the Gini index of the data after pruning is not greater than the original Gini index, subsequent branches are pruned; otherwise, they are retained. Consequently, this approach effectively optimizes the tree structure, resulting in accelerated prediction performance.

B. *Feature engineering*

1) Feature Importance [7] [8]
In the context of linear classifiers, I examine the model's weights and transform them into absolute values. This is done because both positive and negative correlations contain relevant information. We interpret weights close to zero as indicative of a lack of correlation with the output variable. Extracting these weights, I select the top five features with the highest absolute values as significant features. Subsequently, by retraining the model using these features, I observe performance similar to that of the original model.
In the case of decision trees, I employ a Depth-First Search (DFS) algorithm to identify important features. I calculate the depth of each node, presuming that a deeper depth implies clearer classification information, making it potentially more critical. Simultaneously, I reconstruct the tree structure, utilizing another set of data for prediction, and note a comparable performance to the original model.
2) SHAP
Exploring important features in the data using the SHAP [9] tool proves to be a valuable approach. The significant features identified by SHAP align closely with those discovered using a decision tree, as outlined in Table I. The important features identified by the three methods can be referenced in Table I, while a visual representation of the significant features unearthed by SHAP is provided in Figure 2 and 3.

TABLE I: Feature Importance in Difference Methods

Perceptron	Decision Tree	SHAP
policy_tenure	policy_tenure	policy_tenure
is_speed_alert	age_policyholder	age_policyholder
is_power_steering	age_of_car	age_of_car
steering_Power	population_density	displacement
rear_brakes_Drum	area_cluster_C19	torque_rpm

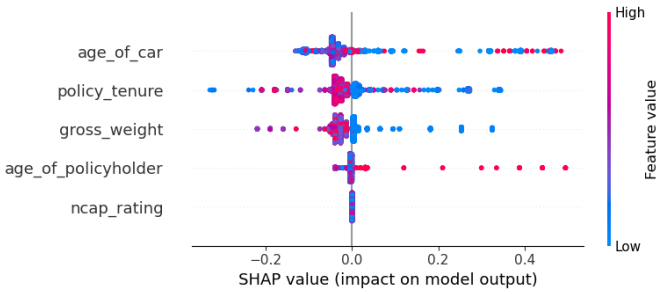

Fig. 2: shap_value

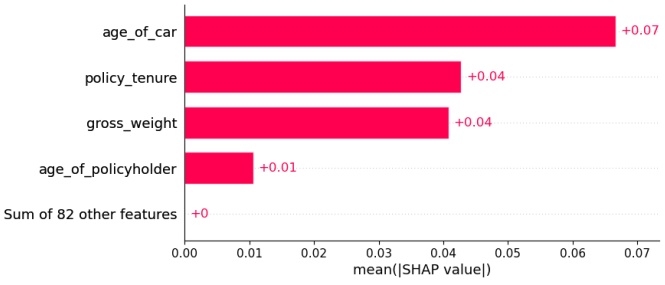

Fig. 3: mean($|shap\_value|$)

3) New Feature
I multiplied the most important parameters and performed feature fusion. Subsequently, I added the resulting features to the original dataset and retrained the model. Using a decision tree model for classification, there was a slight improvement in performance. Please refer to Table II for the predictive accuracy after feature fusion.

TABLE II: Feature Fusion Accuracy Compare

	Before	After
Decision Tree	0.59625	0.61875

C. Cross-Validation

1) Cross-Validation Implement [10]
The following describes an experiment involving cross-validation with k values of 3, 5, and 10, while simultaneously recording accuracy metrics. The results are presented visually. Overall, the model's performance exhibited a certain degree of stability after cross-validation. Please refer to Figures 4, 5, and 6.

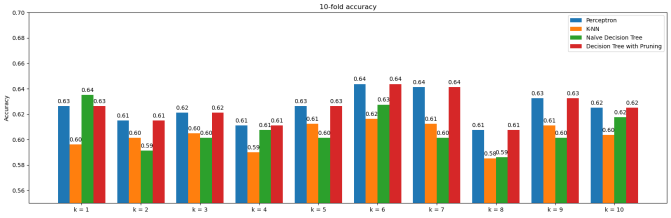

Fig. 4: k = 10

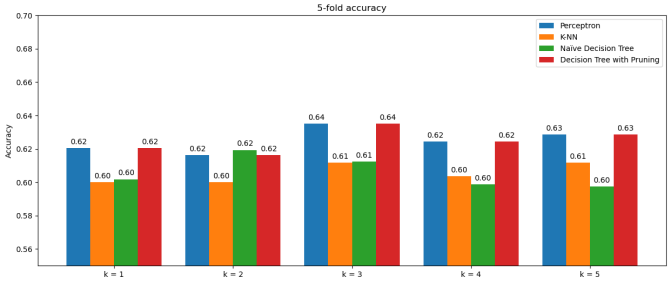

Fig. 5: k = 5

2) Merge Predict
This study utilizes a voting mechanism for each model within the context of cross-validation, wherein predictions are generated for k-fold models. Following this, the probabilities associated with these predictions undergo averaging to yield the final predicted classification. The resultant predictions are then juxtaposed with those of the original model, as delineated in Figure 7. The graphical representations unequivocally demonstrate the enhanced performance of the K-NN model post cross-validation.

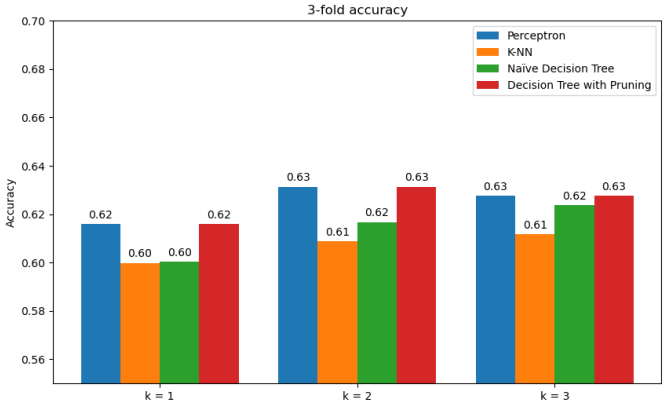

Fig. 6: k = 3

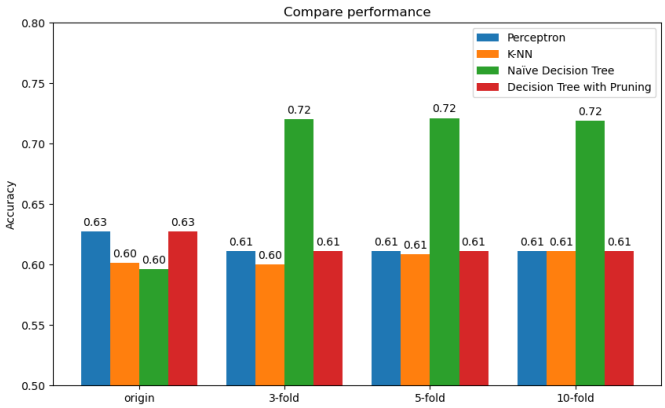

Fig. 7: Compare Accuracy with cross-validation and origin

3) Demonstrates superior performance in terms of F1-score.
The F1 score is a metric that combines precision and recall, commonly used to evaluate the performance of binary classification models. It is the harmonic mean of these two metrics and is often considered a more comprehensive indicator of a model's quality. Therefore, we calculated the F1 score for the k=5 model and compared it with the original model. Interestingly, the decision tree exhibited superior performance. For a detailed comparison table of F1 scores, please refer to Table III.

TABLE III: F1-Score Compare

	Before	After
Decision Tree	0.59625	0.61875

III. Conclusion

This machine learning assignment provides a comprehensive overview of core techniques in supervised learning, covering linear classifiers, k-nearest neighbors (KNN), decision trees, as well as advanced concepts such as feature engineering, model evaluation, and hyperparameter tuning. Through

systematic experimental design, we quantitatively assess the performance of each algorithm and qualitatively understand their strengths and weaknesses. While linear classifiers offer interpretability, they are prone to underfitting; KNN is flexible but computationally expensive; simple decision trees are susceptible to overfitting and require pruning for generalization. Feature engineering enhances accuracy, and algorithm fusion proves effective. Cross-validation reliably evaluates models and boosts predictions through voting, notably resulting in a significant improvement in the F1 score for decision trees. Through extensive references and learning [11] [12], it is recognized that this study demonstrates the fundamental concepts of machine learning by solving practical problems. Despite the use of simplified models, the constructed analytical framework lays the foundation for addressing more advanced challenges. This paves the way for further exploration in the field of machine learning.

REFERENCES

[1] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[2] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[3] J. H. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Transactions on Mathematical Software (TOMS)*, vol. 3, no. 3, pp. 209–226, 1977.

[4] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, pp. 81–106, 1986.

[5] L. Breiman, *Classification and regression trees*. Routledge, 2017.

[6] J. Mingers, "An empirical comparison of pruning methods for decision tree induction," *Machine learning*, vol. 4, no. 2, pp. 227–243, 1989.

[7] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of machine learning research*, vol. 3, no. Mar, pp. 1157–1182, 2003.

[8] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[9] M. Mayer, D. Meier, and M. V. Wuthrich, "Shap for actuaries: Explain any model," *Available at SSRN:*, 2023.

[10] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Ijcai*, vol. 14, pp. 1137–1145, Montreal, Canada, 1995.

[11] w5535586, "Ml_hw1," 2023. http://www.address_of_you_wannar_cite/, Last accessed on 2023-11-13.

[12] Robert0831, "Ml_hw1," 2023. https://github.com/Robert0831/ML_hw1, Last accessed on 2023-10-20.