Assignment-1

Machine Learning

Wen-Hai Tseng
Nation Cheng Kung University
w13281328@gmail.com

***Abstract*—The purpose of this paper is to handcraft a classification model, and through data preprocessing, feature engineering, K-fold cross-validation, and SHAP analysis, achieve data analysis and feature integration, while also assessing the effectiveness of the handcrafted model.**

I. Introduction (*Heading 1*)

In this paper, we conducted automobile insurance claims prediction using Linear Classifier, K-Nearest Neighbors (K-NN), Decision Tree, and Pruned Decision Tree models, with the constraint of not utilizing external libraries. We also performed some feature engineering in between, which resulted in improved model performance. Finally, we conducted K-fold cross-validation to further analyze the model's performanceEase of Use

II. Method

Data Preprocessing: Text, continuous variables, and other data types were transformed using LabelEncoder and OneHotEncoder. The SMOTE method was employed to address the issue of label imbalance, whereby the less frequent labels were oversampled to match the frequency of the other labels. Subsequently, StandardScaler was used to standardize the data. Finally, the data was split into a ratio of 8:1:1 for training, validation, and testing, respectively.

A. Linear Classifier

The model in your code is trained through a process of gradient descent and binary cross-entropy loss minimization. Initially, the model is initialized with specific hyperparameters, including the learning rate and the number of training iterations. The training loop then begins, iterating over the specified number of epochs. In each iteration, the model makes predictions on the training data, transforms these raw predictions into binary class probabilities using the sigmoid function, and computes gradients that indicate the direction of parameter adjustments needed to reduce the loss. The model's weights and bias are updated using these gradients and the learning rate. The training loss is recorded in the loss_history list at each iteration.Additionally, the model calculates the validation loss to monitor its performance on unseen data, which is stored in the val_loss_history. Optionally, training accuracy can be printed at each iteration, and K-fold cross-validation can be performed to assess the model's performance across different subsets of the data. The training process continues for the specified number of iterations, and after training, the model can make binary class predictions using the `predict` method. The goal is to adjust the model's parameters so that it can learn patterns in the training data and make accurate predictions on new, unseen data.

$$L(y,\hat{y}) = -[y\log(\hat{y}) + (1-y)\log(1-\hat{y})] \quad (1)$$

$$\frac{\partial L}{\partial w} = \frac{1}{m} \cdot \boldsymbol{X}^{T} \cdot (\hat{y} - y) \quad (2)$$

$$\frac{\partial L}{\partial b} = \frac{1}{m}\sum_{1}^{m}(\hat{y}^{(i)} - y^{(i)}) \quad (3)$$

B. K Nearest Neighbors

My versatile kNN classifier that can be used for various classification tasks. It offers flexibility in choosing distance metrics (euclidean, manhattan, or minkowski) and different values of k for nearest neighbor search. Additionally, it supports k-fold cross-validation to evaluate the model's performance and allows for custom loss metric computation and visualization.. I also optimized the search method through KDTree, significantly improving the speed, but it is still much slower than sklearn. However, the performance has improved significantly

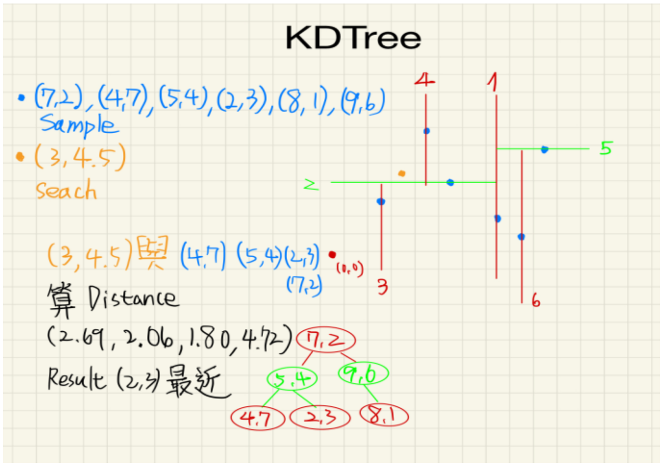


Fig. 1. The Process Diagram of Building and Searching a KD -Tree with Two-Dimensional Data

C. Decision Tree

The decision tree classifier I've implemented employs s everal key features to enhance its performance. It selects the optimal splitting feature and threshold based on the reductio n in Gini impurity to ensure the best data partitioning. This h elps in creating more effective decision boundaries. Additio nally, the inclusion of a random feature subset at each node mitigates overfitting, ultimately improving the model's gene ralization capabilities.

The cross-validation technique is utilized to assess the model's performance reliably. By employing K-fold cross-v alidation, the model's average performance is accurately esti mated, ensuring that it can generalize well to unseen data.

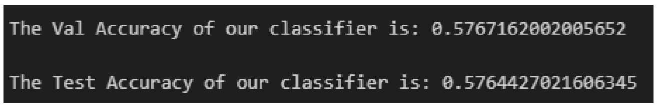
Furthermore, your implementation also calculates feature importance, enabling an understanding of which features have the most significant impact on the model's predictions. This insight is invaluable for feature selection and interpretation of the model's behavior.

D. Decision Tree with Pruning

The provided decision tree code employs pruning techniques to enhance its performance and mitigate overfitting. It includes a maximum depth limit (max_depth) that stops further node splitting when the tree's depth reaches the specified maximum. This is essential for controlling the tree's complexity and preventing excessive fitting to the training data. Additionally, there is a minimum sample split criterion (min_samples_split) that halts node splitting when the number of samples in a node falls below or equals the specified threshold, preventing over-specialization in small nodes. Moreover, the stopping condition considers situations where a node's target variable (y) contains only a single unique value, indicating no further need for splitting to distinguish these samples. These pruning measures collectively ensure the decision tree maintains an appropriate level of simplicity, thus enhancing its generalization performance while avoiding overfitting

III. EXPERMENTS

A. Linear Classifier



```
The Val Accuracy of our classifier is: 0.5767162002005652

The Test Accuracy of our classifier is: 0.5764427021606345
```

Fig. 2. The accuracy of the Linear Classifier model's predicti ons on the Validation and Test datasets.

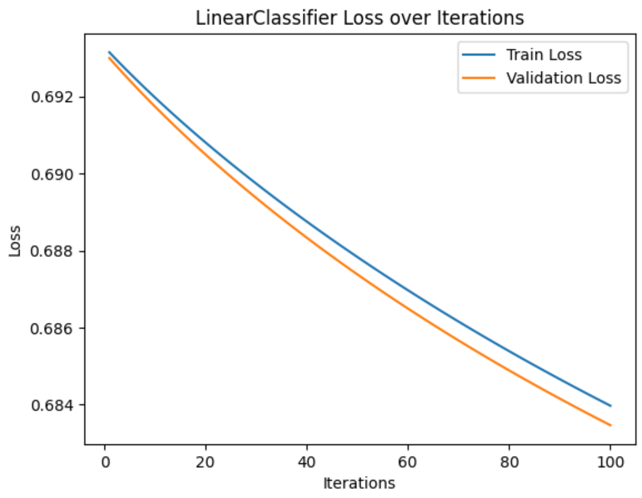
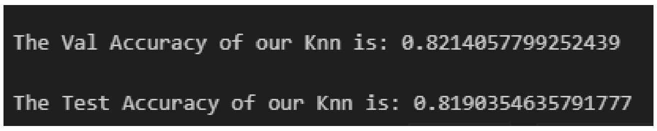


Fig. 3. The loss curve plot of the Linear Classifier model.

After training the Linear Classifier model for 100 epochs, it achieved an accuracy of 58%. Observing the loss curve ind icates that the model converged properly. However, due to th e complexity of the data, the model couldn't achieve a strong performance.

B. K Nearest Neighbors

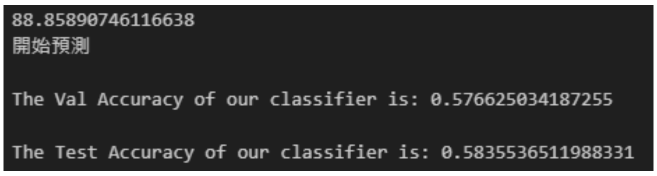


```
The Val Accuracy of our Knn is: 0.8214057799252439

The Test Accuracy of our Knn is: 0.8190354635791777
```

Fig. 4. The accuracy of the K Nearest Neighbors model's predictions on the Validation and Test datasets

Using K-Nearest Neighbors (KNN) with 80,000+ data points to classify more than 10,000 data points resulted in an accuracy of 81%. The model was configured with a K value of 3, and KDTree was utilized for acceleration. The execution took approximately 10 minutes.

C. Decision Tree

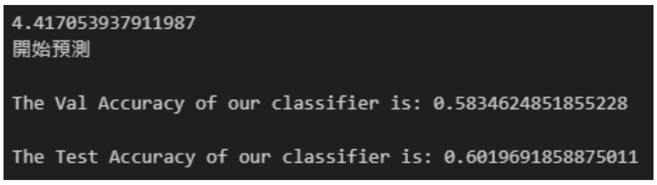


```
88.85890746116638
開始預測

The Val Accuracy of our classifier is: 0.576625034187255

The Test Accuracy of our classifier is: 0.5835536511988331
```

Fig. 5. The accuracy of the Decision Tree model's predictions on the Validation and Test datasets.

A Decision Tree model was used without setting specific parameters, but some optimizations were applied to speed up the model. It divided continuous variables into branches by taking values at 10 percentiles, which improved training speed, resulting in an accuracy of 58%.

D. Decision Tree with Pruning



```
4.417053937911987
開始預測

The Val Accuracy of our classifier is: 0.5834624851855228

The Test Accuracy of our classifier is: 0.6019691858875011
```

Fig. 6. The accuracy of the Decision Tree with Pruning model's predictions on the Validation and Test datasets.

Pruning the Decision Tree was performed with the settings max_depth=5 and min_samples_split=2. Additionally, optimizations were applied by dividing continuous variables into branches using 10 percentiles to improve training speed. As a result, the accuracy increased to 61%. Notably, this pruned model showed a significant reduction in execution time compared to the non-pruned version, decreasing from 110 seconds to 5 seconds.

E. Feature Importance

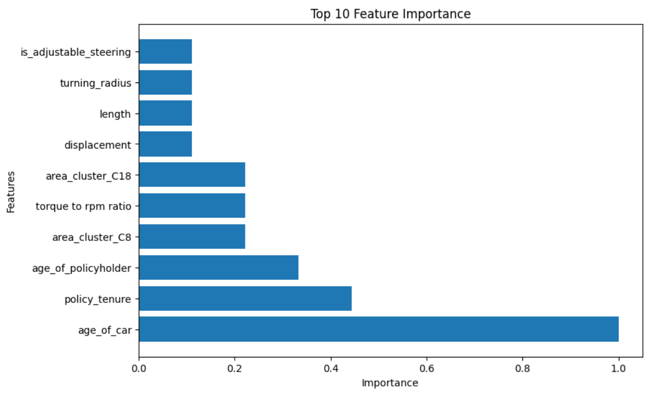


Fig. 7. Calculate Gini coefficient and rank feature importance using a decision tree.

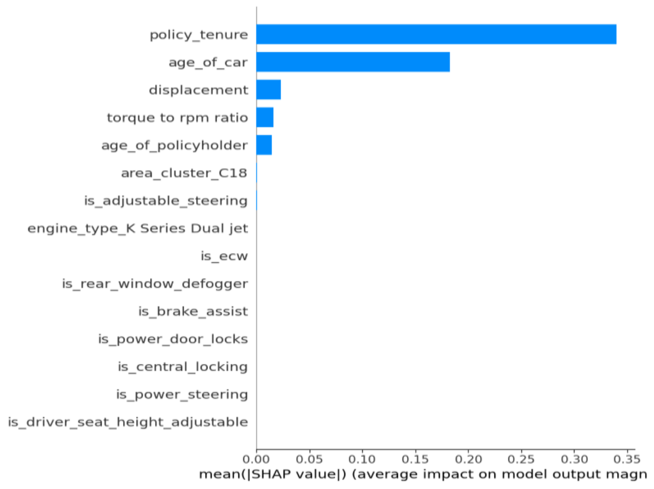


Fig. 8. Utilize SHAP with my decision tree to assess feature importance.

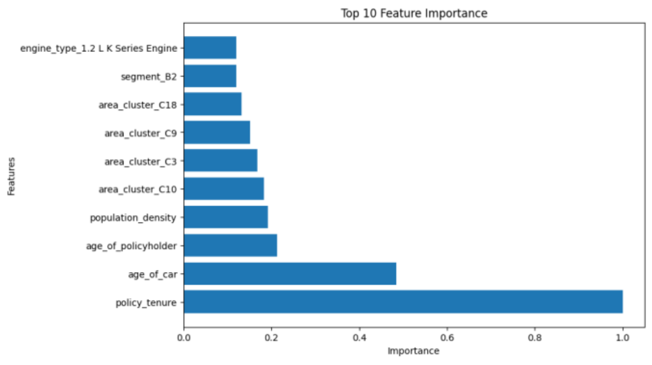
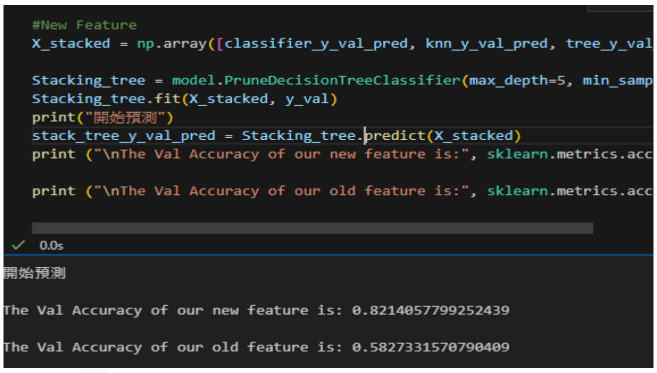


Fig. 9. Calculate Gini coefficient and rank feature importance using a linear classifier.

Using Decision Trees to calculate the Gini coefficient and Linear Classifier's weights as importance metrics, and applying the Decision Tree to the Shap package, we found that in each method, 'age_of_car' and 'policy_tenure' consistently ranked in the top three for importance.

F. Feature Fusion



```
#New Feature
X_stacked = np.array([classifier_y_val_pred, knn_y_val_pred, tree_y_val

Stacking_tree = model.PruneDecisionTreeClassifier(max_depth=5, min_samp
Stacking_tree.fit(X_stacked, y_val)
print("開始預測")
stack_tree_y_val_pred = Stacking_tree.predict(X_stacked)
print ("\nThe Val Accuracy of our new feature is:", sklearn.metrics.acc

print ("\nThe Val Accuracy of our old feature is:", sklearn.metrics.acc

 0.0s
開始預測

The Val Accuracy of our new feature is: 0.8214057799252439

The Val Accuracy of our old feature is: 0.5827331570790409
```

Fig.10. Train a new model by stacking the results of four individual models..

By stacking the results of four individual models, we were able to improve the accuracy from 58% to 82%. However, it's evident that the new model performs similarly to KNN, indicating that the other three models didn't outperform KNN in certain aspects.

G. Based on Problem 1,use k-fold cross-validation to verify the stability of each classifier.

	3 folds	5 folds	10 folds
Linear Classifier	57.75%	57.77%	57.77%
KNearest Neighbors	77.57%	78.62%	79.41%
Decision Tree	88.87%	89.26%	89.57%
Decision Tree with Pruning	64.36%	64.44%	64.48%

Table.1 The average accuracies of the four models in Problem 1 across 3-folds, 5-folds, and 10-folds.

KNearest Neighbors Performs Best:

In 5-fold and 10-fold cross-validation, KNearest Neighbors demonstrates significantly higher accuracy compared to the other models, achieving 78.62% and 79.41%, respectively. This suggests that KNearest Neighbors performs relatively well on the data and may be suitable for the problem at hand.

Decision Tree and Decision Tree with Pruning:

The accuracy of Decision Tree and Decision Tree with Pruning is relatively high, showing similar performance across different fold numbers. This indicates that the decision tree models exhibit a degree of stability with little impact from pruning on performance.

Poor Performance of Linear Classifier:

The linear classifier exhibits lower accuracy across all fold numbers, hovering around 57%. This suggests that the linear classifier's performance is subpar for your data, and further feature engineering or exploration of alternative algorithms may be necessary to improve performance.

In summary, KNearest Neighbors and Decision Tree are the better-performing options, particularly with KNearest Neighbors excelling across different fold numbers. Additionally, if interpretability is essential for the problem, Decision Tree is a promising choice. Further optimization of these models, such as parameter tuning or feature selection, may enhance their performance.

H. *Design an algorithm that can merge/aggregate the predicted results from k classifiers in k-fold cross-validation.*

	3 folds	5 folds	10 folds
VotingClassifier	67.43%	67.89%	68.22 %

Table.2 The average accuracy of the four k-fold models when combined through voting.

The voting method is an effective ensemble learning technique that can enhance model performance, but it also requires careful model selection and parameter tuning. It can be observed that my voting model does not achieve higher accuracy compared to the highest accuracy in Table 1. This may be because most of the other models made incorrect predictions, leading to no improvement in accuracy.

I. *compare the result in 5-fold cross-validation and the result of Problem 1 to justify which is "REALLY" better..*

Through ANOVA, the calculated p-value is very close to 1 (0.9977). This implies that there is no significant difference in the performance of the four models across different fold numbers. Therefore, we cannot reject the null hypothesis, which suggests that the average performance of the four models is equal. Thus, in the case of 5-fold cross-validation, we can confidently choose the best-performing model as indicated in Table 1, which is the Decision Tree.

References

[1] https://devpress.csdn.net/python/62f62634c6770329307fc080.html

[2] https://zhuanlan.zhihu.com/p/23966698