Machine Learning Assignment-1

Pei-Chun Chen

Nation Cheng Kung University

***Abstract*—This paper aims to meticulously construct a classification model, navigating through the realms of data preprocessing, feature engineering, K-fold cross-validation, and SHAP analysis. The overarching goal is to delve into data analysis and seamlessly integrate features, all the while scrutinizing the efficacy of the artisanal model.**

I. Introduction

This paper primarily focuses on establishing algorithms, including a linear classifier, K-nearest neighbors, and decision tree, for training and validating the model. Additionally, it incorporates feature engineering to assess feature importance and employs SHAP (SHapley Additive exPlanations) for enhancing model performance. Finally, k-fold cross-validation is used to compare model performances and select the most suitable model for this dataset.

II. Algoriyhm

A. Linear Classifier

- Description: A linear classifier is a fundamental machine learning model that aims to discriminate between different classes by establishing a linear decision boundary in the feature space. The perceptron algorithm, a pioneering development in the field, embodies the essence of linear classification. Proposed by Frank Rosenblatt in 1957, the perceptron algorithm laid the groundwork for neural network models.
- Algorithm: The perceptron algorithm operates by iteratively updating its weight parameters to minimize the misclassification error. At each iteration, the perceptron examines training instances and adjusts the weights based on misclassified samples until convergence is achieved. This process involves computing the weighted sum of input features and comparing it to a threshold, determining the predicted class.
- Formula:

  $f(x) = sign(\sum_{i=1}^{n} w_i x_i + b)$

  where:

  $f(x)$ is the predicted class

  w_i represents the input features

  x_i denotes the corresponding weights

  b is the bias term

  sign(·) is the sign function

  The weight adjustments are performed using the perceptron learning rule:

  $w_i \leftarrow w_i + \eta \cdot (y - \hat{y}) \cdot x_i$

  where:

  η is the learning rate

  $\hat{y}$ is the predicted class label

- Principles: The perceptron's underlying principle is rooted in the idea of finding a hyperplane that separates the classes in the feature space. Convergence of the perceptron algorithm is guaranteed when the data is linearly separable. However, in cases where perfect separation is not feasible, convergence is not assured, leading to the introduction of more advanced algorithms like the multi-layer perceptron.

B. K Nearest Neighgors

- Description: The k-Nearest Neighbors algorithm is a non-parametric and instance-based machine learning approach for classification and regression tasks. It operates on the principle that instances in the feature space with similar characteristics tend to belong to the same class or exhibit similar behaviors.
- Algorithm: Given a dataset with labeled instances, the k-NN algorithm classifies a new data point by identifying the k-nearest neighbors based on a chosen distance metric (commonly Euclidean distance). The class of the majority of these neighbors determines the classification of the new data point. For regression tasks, the algorithm computes the average of the k-nearest neighbors' target values.
- Formula:

  $\hat{y} = argmax(\sum_{i=1}^{k} \delta(y_i = c))$

  where:

  $\hat{y}$ is the predicted clas

  y_i represents the class labels of the k-nearest neighbors

  c denotes a specific class

  b is the bias term

  $\delta(\cdot)$ is the Kronecker delta function

C. Decision Tree

- Description: A Decision Tree is a hierarchical and non-linear predictive model used for both classification and regression tasks in machine learning. It recursively partitions the input space into regions and assigns a predictive label or value to each region. The tree structure is composed of nodes representing decision points and leaves containing the final predictions.
- Algorithm: The algorithm constructs the decision tree through a top-down, recursive process known as recursive binary splitting. At each node, the algorithm selects the feature that best splits the data into subsets, optimizing a predefined criterion (such as Gini impurity for classification or mean squared error for regression). This process continues until a stopping condition is met, such as reaching a maximum depth or a minimum number of samples in a leaf.

- Advantages: Decision Trees offer interpretability, ease of understanding, and are capable of handling both numerical and categorical data. They require minimal data preprocessing, handle non-linear relationships effectively, and can be visualized for insights into decision-making processes. However, care must be taken to avoid overfitting, and ensemble methods like Random Forests address this concern by aggregating multiple trees.

D. *Decision Tree With Pruning*

- Description: A Decision Tree with pruning is an enhanced variant of the conventional Decision Tree algorithm that incorporates a process to mitigate overfitting by selectively removing or collapsing branches of the tree. Pruning is introduced to improve the model's generalization performance and prevent it from capturing noise in the training data.
- Algorithm: The Decision Tree with pruning algorithm follows a similar top-down recursive binary splitting approach as the standard Decision Tree. However, it introduces a pruning step after the initial tree construction. During pruning, the algorithm evaluates the impact of removing branches based on a cost-complexity measure, such as the Minimal Cost-Complexity Pruning (MCCP) algorithm. Pruning involves iteratively assessing subtrees and removing branches that do not significantly contribute to improving the model's predictive performance on unseen data.
- Comparison with Decision Tree: While traditional Decision Trees may develop overly complex structures that memorize noise in training data, a pruned Decision Tree aims for a more generalized model. This pruning process prevents the tree from becoming too specific to training data, improving accuracy on new, unseen data. The primary advantage lies in enhanced generalization, as pruning reduces sensitivity to noise, leading to improved performance on new data. Additionally, it contributes to model interpretability by simplifying complex tree structures.

III. EXPERMENTS

In the initial data preprocessing phase, a pie chart was constructed to visualize the distribution of the two classes labeled 0 and 1, revealing a pronounced imbalance. To rectify this class imbalance issue, undersampling was implemented, resulting in a resampled dataset with a balanced 2:1 ratio between label 0 and label 1. Subsequently, another pie chart was generated to illustrate the newfound balance.

Upon closer inspection, it was discovered that the 'max_torque' variable signifies the maximum Newton-meters (Nm) at a specific rpm.

To encapsulate this correlation, a novel variable 'torque_to_rpm_ration' was crafted, representing the ratio of torque to rpm. The 'max_power' variable underwent a similar transformation to 'power_to_rpm_ratio'.

Additionally, categorical variables were converted into dummy variables.

To mitigate the impact of varying feature scales on the model, certain continuous variables exhibiting substantial differences were standardized. This crucial step aimed to harmonize scales, mitigating potential distortions arising from disparate feature magnitudes.

A. *Linear classifier*

```
Epoch 1/100 - Training Accuracy: 0.6620344635908838
Epoch 2/100 - Training Accuracy: 0.665591995553085
Epoch 3/100 - Training Accuracy: 0.665591995553085
Epoch 4/100 - Training Accuracy: 0.6430239021678711
Epoch 5/100 - Training Accuracy: 0.6608115619788771
Epoch 6/100 - Training Accuracy: 0.44869371873262925
Epoch 7/100 - Training Accuracy: 0.5177320733740968
Epoch 8/100 - Training Accuracy: 0.3869927737632018
.
.
.
Epoch 94/100 - Training Accuracy: 0.40444691495275154
Epoch 95/100 - Training Accuracy: 0.6438021122846026
Epoch 96/100 - Training Accuracy: 0.33440800444691493
Epoch 97/100 - Training Accuracy: 0.5209560867148416
Epoch 98/100 - Training Accuracy: 0.5290717065036131
Epoch 99/100 - Training Accuracy: 0.6654808226792662
Epoch 100/100 - Training Accuracy: 0.6563646470261256
Accuracy of perceptron is: 0.6611827478879502
```

After training the Linear Classifier model for 100 epochs, it achieved an accuracy of 0.66.

Fluctuations in training accuracy across epochs, rather than a steady increase or decrease, suggest incomplete convergence. The initially high accuracy followed by later fluctuations hints at potential sensitivity to training data, indicating possible partial overfitting.

B. *K Nearest Neighgors*

The KNN model, utilizing a K value of 3, was trained on over 8000 data points and tested on more than 2000 data points using three different distance metrics: Euclidean, Manhattan, and Chebyshev.

method = "euclidean"

```
Accuracy of knn_euclidean: 0.6082703423743886
```

method = "manhattan"

```
Accuracy of knn_manhattan: 0.6024899955535794
```

method = "chebyshev"

```
Accuracy of knn_chebyshev: 0.6162738995108937
```

The model's performance, assessed by accuracy, exhibited negligible differences across all three distance metrics. This observation implies that, for this specific dataset, the selection of a distance metric had minimal impact on the model's predictive capability.

C. *Decision Tree*

```
Accuracy of Decision Tree: 0.57
```

The Decision Tree model was employed without explicit parameter configurations, yet certain optimizations were implemented to enhance its efficiency. By segmenting continuous variables into branches based on values at 10 percentiles, the model experienced accelerated training, yielding an accuracy of 0.57.

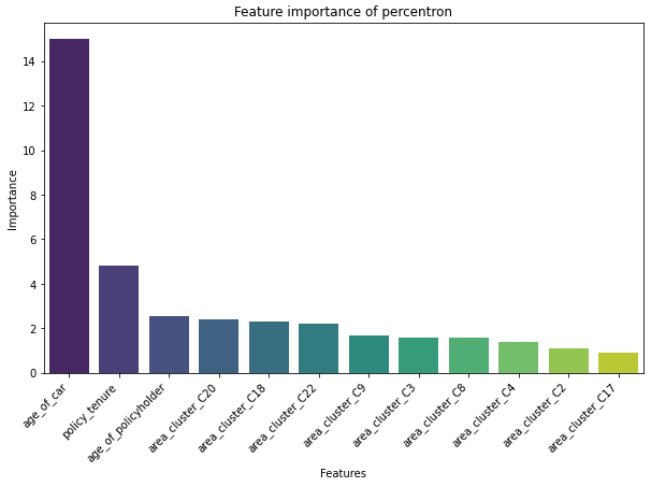
D. *Decision Tree With Pruning*

```
Accuracy of Tree with Pruning: 0.65
```

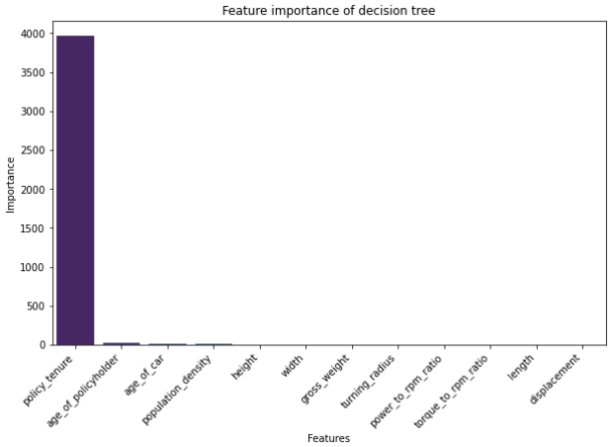
Pruning a decision tree is preferable to not pruning, as it serves to prevent overfitting. Overfitting occurs when a decision tree becomes too intricately tailored to the training data, leading to suboptimal performance on test data. Pruning mitigates this issue by restricting the growth of the tree, preventing it from overly fitting the training data and consequently enhancing its performance on test data.

Improved Generalization: In certain scenarios, pruning can enhance the generalization ability of a decision tree, enabling it to perform better on unseen data. Pruning helps strike a balance, ensuring that the decision tree doesn't become too specific to the training set, but rather captures underlying patterns that are more likely to apply to new, unseen data.

E. Features Importance

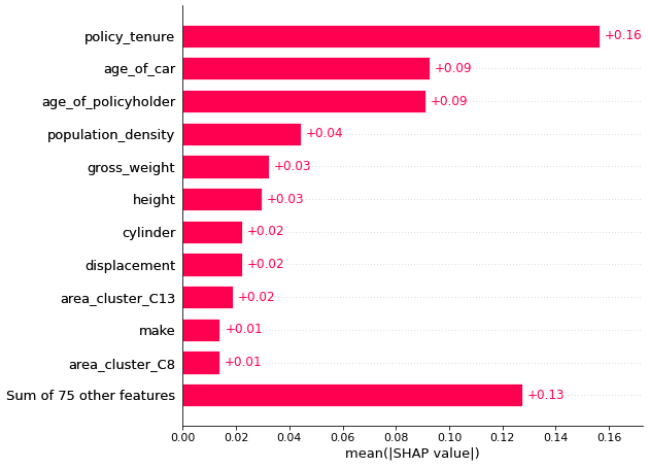


This figure depicts the feature importance of a perceptron, specifically the top 12 features ranked by their importance. The importance is determined by the absolute values of the weights, which are used to derive the significance of each feature.

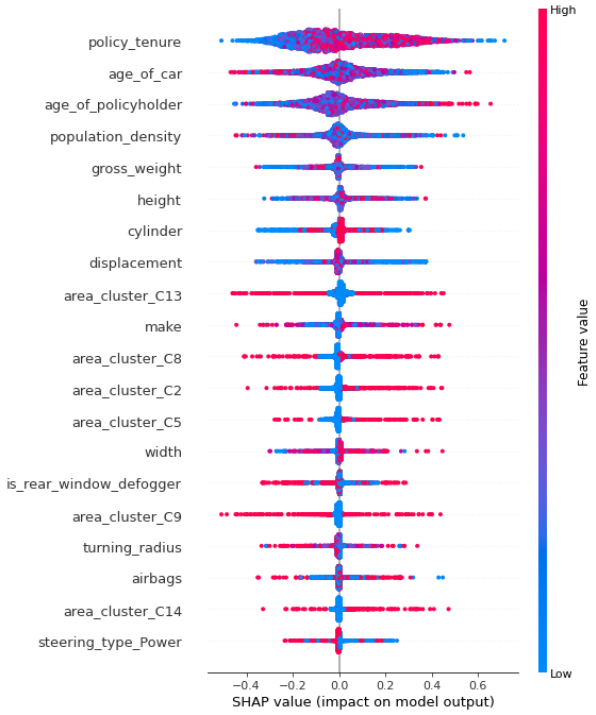


The image above depicts the accuracy of a decision tree, and it is notable that "policy_tenure" stands out significantly compared to others, which has raised my curiosity. However, after conducting a Shap analysis, I discovered that the importance ranking of the first six features is consistent.

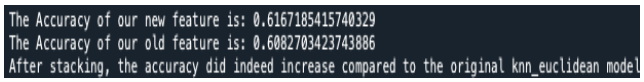
- SHAP: Using the "SHapley Additive exPlanations" approach, analyze the feature importance of a decision tree. This image displays the importance of the top 12 features.



And this figure below represents SHAP summary plot visualizations. It is crucial to closely examine the direction (whether positive or negative) and the magnitude of the SHAP values associated with each feature. This analytical approach provides valuable insights into comprehending how each feature contributes to the model's output for a specific instance.



- Stacking: I chose the stacking technique to design new features, as it combines predictions from multiple models to enhance overall performance. Leveraging the strengths of diverse models, it automatically performs feature engineering and adapts to different data distributions, making it effective in handling complex relationships within the data.



This figure compares the accuracy of the old and new features after stacking. Initially, the scores of perceptron and decision tree with pruning were higher than that of knn_euclidean. Consequently, during the stacking process, the overall score experienced an improvement.

F. Cross-Validation

The data was divided into 3, 5, and 10 subsets for cross-validation. In each case, the average accuracy was computed by combining the accuracies of each fold. It was observed that the perceptron exhibited greater variability across the three scenarios, leading to more substantial differences in the average scores obtained. It can also be observed that the scores of other models, such as k-Nearest Neighbors (KNN) and Decision Tree, are relatively stable across the different scenarios.

k-fold for perceptron (for k=3, 5, 10)

```
Fold 1 - Accuracy for k=3: 0.6568836712913554
Fold 2 - Accuracy for k=3: 0.6678228388473852
Fold 3 - Accuracy for k=3: 0.5691035218783351
Average Accuracy for k=3: 0.6312700106723586

Fold 1 - Accuracy for k=5: 0.666073810582481
Fold 2 - Accuracy for k=5: 0.6651845264562027
Fold 3 - Accuracy for k=5: 0.666073810582481
Fold 4 - Accuracy for k=5: 0.48599377501111607
Fold 5 - Accuracy for k=5: 0.33362989323843417
Average Accuracy for k=5: 0.563391163174143

Fold 1 - Accuracy for k=10: 0.6675555555555556
Fold 2 - Accuracy for k=10: 0.5368888888888889
Fold 3 - Accuracy for k=10: 0.4746666666666667
Fold 4 - Accuracy for k=10: 0.6373333333333333
Fold 5 - Accuracy for k=10: 0.650355871886121
Fold 6 - Accuracy for k=10: 0.3798932384341637
Fold 7 - Accuracy for k=10: 0.6663701067615658
Fold 8 - Accuracy for k=10: 0.5836298932384342
Fold 9 - Accuracy for k=10: 0.6663701067615658
Fold 10 - Accuracy for k=10: 0.6672597864768683
Average Accuracy for k=10: 0.5930323448003163
```

k-fold for KNN_euclidean (for k=3, 5, 10)

```
Fold 1 - Accuracy for k=3: 0.6226490596238495
Fold 2 - Accuracy for k=3: 0.6106442577030813
Fold 3 - Accuracy for k=3: 0.6128903122497998
Average Accuracy for k=3: 0.6153945431922435

Fold 1 - Accuracy for k=5: 0.6166666666666667
Fold 2 - Accuracy for k=5: 0.6130753835890593
Fold 3 - Accuracy for k=5: 0.590393595730487
Fold 4 - Accuracy for k=5: 0.5983989326217478
Fold 5 - Accuracy for k=5: 0.619079386257505
Average Accuracy for k=5: 0.6075227929730932

Fold 1 - Accuracy for k=10: 0.6293333333333333
Fold 2 - Accuracy for k=10: 0.62
Fold 3 - Accuracy for k=10: 0.6306666666666667
Fold 4 - Accuracy for k=10: 0.6173333333333333
Fold 5 - Accuracy for k=10: 0.5866666666666667
Fold 6 - Accuracy for k=10: 0.6026666666666667
Fold 7 - Accuracy for k=10: 0.5714285714285714
Fold 8 - Accuracy for k=10: 0.6315086782376502
Fold 9 - Accuracy for k=10: 0.6328437917222964
Fold 10 - Accuracy for k=10: 0.6114819759679573
Average Accuracy for k=10: 0.6133929684023143
```

k-fold for decision tree (for k=3, 5, 10)

```
Fold 1 - Accuracy for k=3: 0.581032412965186
Fold 2 - Accuracy for k=3: 0.5782312925170068
Fold 3 - Accuracy for k=3: 0.5876701361088871
Average Accuracy for k=3: 0.58231128053036

Fold 1 - Accuracy for k=5: 0.586
Fold 2 - Accuracy for k=5: 0.5943962641761175
Fold 3 - Accuracy for k=5: 0.5783855903935957
Fold 4 - Accuracy for k=5: 0.5703802535023349
Fold 5 - Accuracy for k=5: 0.5863909272848565
Average Accuracy for k=5: 0.583110607071381

Fold 1 - Accuracy for k=10: 0.5933333333333334
Fold 2 - Accuracy for k=10: 0.616
Fold 3 - Accuracy for k=10: 0.5493333333333333
Fold 4 - Accuracy for k=10: 0.584
Fold 5 - Accuracy for k=10: 0.596
Fold 6 - Accuracy for k=10: 0.5893333333333334
Fold 7 - Accuracy for k=10: 0.5634178905206942
Fold 8 - Accuracy for k=10: 0.5647530040053405
Fold 9 - Accuracy for k=10: 0.5967957276368492
Fold 10 - Accuracy for k=10: 0.6034712950600801
Average Accuracy for k=10: 0.5856437917222964
```

k-fold for decision tree with pruning (for k=3, 5, 10)

```
Fold 1 - Accuracy for k=3: 0.6425475158386129
Fold 2 - Accuracy for k=3: 0.6264176117411607
Fold 3 - Accuracy for k=3: 0.5783855903935957
Average Accuracy for k=3: 0.6157835726577897

Fold 1 - Accuracy for k=5: 0.6642579210672596
Fold 2 - Accuracy for k=5: 0.6331295163979989
Fold 3 - Accuracy for k=5: 0.6570316842690384
Fold 4 - Accuracy for k=5: 0.6536964980544747
Fold 5 - Accuracy for k=5: 0.5825458588104503
Average Accuracy for k=5: 0.6381322957198443

Fold 1 - Accuracy for k=10: 0.6422222222222222
Fold 2 - Accuracy for k=10: 0.6477777777777778
Fold 3 - Accuracy for k=10: 0.6155555555555555
Fold 4 - Accuracy for k=10: 0.6311111111111111
Fold 5 - Accuracy for k=10: 0.6766666666666666
Fold 6 - Accuracy for k=10: 0.6629588431590656
Fold 7 - Accuracy for k=10: 0.5817575083426029
Fold 8 - Accuracy for k=10: 0.6685205784204672
Fold 9 - Accuracy for k=10: 0.6718576195773082
Fold 10 - Accuracy for k=10: 0.6084538375973304
Average Accuracy for k=10: 0.6406881720430107
```

G. Merge/Aggregate

By training three models - Perceptron, KNN_euclidean, and Decision Tree - and employing a merge/aggregate method to predict results through a voting mechanism, it is not guaranteed that accuracy will improve. In certain cases, all models may make the same errors, leading to a voting outcome that is not necessarily more accurate. The obtained accuracy of 0.55 indicates that the performance is not exceptionally high.

```
Ensemble Accuracy: 0.5490928495197439
Ensemble Classification Report:
              precision    recall  f1-score   support

           0       0.64      0.76      0.69      7496
           1       0.21      0.13      0.16      3748

    accuracy                           0.55     11244
   macro avg       0.42      0.44      0.43     11244
weighted avg       0.49      0.55      0.51     11244
```

H. Model Performance Compare

	Model			
	Perceptron	KNN	Decision Tree	Decision Tree with pruning
Accuracy				
In Problem 1	0.66	0.61	0.57	0.65
5-fold Cross validation	0.56	0.61	0.58	0.64
RANK				
In Problem 1	2	3	4	1
5-fold Cross validation	4	2	3	1

From the above table, it is evident that Decision Tree with pruning consistently performs the best, maintaining the highest scores. It improves upon the traditional Decision Tree by addressing overfitting and leveraging the advantages of non-linear classification. On the other hand, Perceptron exhibits significant instability, with a notable 0.1 difference in accuracy between Problem 1 and cross-validation. Additionally, KNN_euclidean and the traditional Decision Tree consistently fall in between, with KNN_euclidean showing slightly better stability than the traditional Decision Tree.

REFERENCES

[1] https://www.kaggle.com/code/zubinrelia/car-insurance-claim-classifier