ITD105 Case Study #1

Comparing Machine Learning Algorithms

Name: Clint Joshua O. Velasquez

I CLASSIFICATION

Train the **classification dataset** using various machine learning algorithms designed for classification. Evaluate and compare these models by applying different resampling techniques and utilizing appropriate performance metrics.

Classification Dataset

Dataset Name: Car Evaluation

Features: Buying Price; Maintenance Cost; Number of Doors; Number of Persons; Lug_Boot; Safety; Classification

Set A

Resampling Technique: Split into train and test sets

Classification Metric: Confusion Matrix and Classification Report

ML Algorithm (Classification)	Confusion Matrix *(Provide the matrix and classification report of each algorithm)*
CART (Classification and Regression Trees)	Confusion Matrix: [[234 1] [2 109]] Classification Report: precision recall f1-score support 0 0.99 1.00 0.99 235 1 0.99 0.98 0.99 111 accuracy 0.99 346 macro avg 0.99 0.99 0.99 346 weighted avg 0.99 0.99 0.99 346
Gaussian Naive Bayes/Naive Bayes	Confusion Matrix: [[220 15] [19 92]] Classification Report: precision recall f1-score support 0 0.92 0.94 0.93 235 1 0.86 0.83 0.84 111 accuracy 0.90 346 macro avg 0.89 0.88 0.89 346 weighted avg 0.90 0.90 0.90 346
Gradient Boosting Machines (AdaBoost)	Confusion Matrix: [[228 7] [11 100]] Classification Report: precision recall f1-score support 0 0.95 0.97 0.96 235 1 0.93 0.90 0.92 111 accuracy 0.95 346 macro avg 0.94 0.94 0.94 346 weighted avg 0.95 0.95 0.95 346
K-Nearest Neighbors (K-NN)	Confusion Matrix: [[235 0] [6 105]] Classification Report: precision recall f1-score support 0 0.98 1.00 0.99 235 1 1.00 0.95 0.97 111 accuracy 0.98 346 macro avg 0.99 0.97 0.98 346 weighted avg 0.98 0.98 0.98 346

Logistic Regression	Confusion Matrix: [[220 15] [26 85]] Classification Report: precision recall f1-score support 0 0.89 0.94 0.91 235 1 0.85 0.77 0.81 111 accuracy 0.88 346 macro avg 0.87 0.85 0.86 346 weighted avg 0.88 0.88 0.88 346
Multi-Layer Perceptron (MLP)	Confusion Matrix: [[231 4] [11 100]] Classification Report: precision recall f1-score support 0 0.95 0.98 0.97 235 1 0.96 0.90 0.93 111 accuracy 0.96 346 macro avg 0.96 0.94 0.95 346 weighted avg 0.96 0.96 0.96 346
Perceptron	Confusion Matrix: [[190 45] [5 106]] Classification Report: precision recall f1-score support 0 0.97 0.81 0.88 235 1 0.70 0.95 0.81 111 accuracy 0.86 346 macro avg 0.84 0.88 0.85 346 weighted avg 0.89 0.86 0.86 346
Random Forest	Confusion Matrix: [[235 0] [2 109]] Classification Report: precision recall f1-score support 0 0.99 1.00 1.00 235 1 1.00 0.98 0.99 111 accuracy 0.99 346 macro avg 1.00 0.99 0.99 346 weighted avg 0.99 0.99 0.99 346
Support Vector Machines (SVM)	Confusion Matrix: [[233 2] [10 101]] Classification Report: precision recall f1-score support 0 0.96 0.99 0.97 235 1 0.98 0.91 0.94 111 accuracy 0.97 346 macro avg 0.97 0.95 0.96 346 weighted avg 0.97 0.97 0.96 346

Set B *(should use different resampling technique and classification metric)*
Resampling Technique: K-fold Cross Validation
Classification Metric: Classification Accuracy

ML Algorithm (Classification)	
CART (Classification and Regression Trees)	Mean Accuracy: 0.8473753192633419 Standard Deviation: 0.11940050981103918
Gaussian Naive Bayes/Naive Bayes	Mean Accuracy: 0.8993816373168437 Standard Deviation: 0.06628282497867485
Gradient Boosting Machines (AdaBoost)	Mean Accuracy: 0.917898911143971 Standard Deviation: 0.06281304813811818
K-Nearest Neighbors (K-NN)	Mean Accuracy: 0.8941457185105526 Standard Deviation: 0.0398902191529915
Logistic Regression	Mean Accuracy: 0.8709067078908456 Standard Deviation: 0.06944590627065844
Multi-Layer Perceptron (MLP)	Mean Accuracy: 0.9375352870009408 Standard Deviation: 0.044445170561649996
Perceptron	Mean Accuracy: 0.8270668100551151 Standard Deviation: 0.1008261198126333
Random Forest	Mean Accuracy: 0.9063314961688398 Standard Deviation: 0.08069074088344796
Support Vector Machines (SVM)	Mean Accuracy: 0.9288546847694583 Standard Deviation: 0.04578712545956918

Set C *(should use different resampling technique and classification metric)*
Resampling Technique: Repeated Random Train-Test splits
Classification Metric: Logarithmic Loss

ML Algorithm (Classification)	
CART (Classification and Regression Trees)	Log Loss: 0.3400 (±0.1841)
Gaussian Naive Bayes/Naive Bayes	Log Loss: 0.2055 (±0.0157)
Gradient Boosting Machines (AdaBoost)	Log Loss: 0.4945 (±0.0083)
K-Nearest Neighbors (K-NN)	Log Loss: 0.1165 (±0.0218)
Logistic Regression	Log Loss: 0.2463 (±0.0234)
Multi-Layer Perceptron (MLP)	Log Loss: 0.0633 (±0.0165)
Perceptron	Log Loss: N/A
Random Forest	Log Loss: 0.0623 (±0.0070)
Support Vector Machines (SVM)	Log Loss: 0.0904 (±0.0171)

Results interpretation (Set A, Set B and Set C):
Set A (Train-Test Split with Confusion Matrix and Classification Report):

In Set A, I performed a train-test split, which is a common way to evaluate machine learning models. I tested several classifiers, and here are the results:

- **CART (Classification and Regression Trees):** CART demonstrated excellent performance, with an accuracy of 99%. It had a high precision, recall, and F1-score for both classes, indicating strong class separation.
- **Naive Bayes:** The Naive Bayes classifier achieved an accuracy of 90%. It exhibited good performance in terms of precision and recall, especially for the "0" class.
- **AdaBoost:** AdaBoost achieved an accuracy of 95% with good precision, recall, and F1-scores for both classes. It showed the ability to handle the data and boost classification performance.
- **K-NN:** K-NN performed well with a 98% accuracy. It had excellent recall for the "0" class, and overall high precision and F1-scores.
- **Logistic Regression:** Logistic Regression had an 88% accuracy and showed good precision and recall for the "0" class.
- **MLP:** The Multi-Layer Perceptron achieved a high accuracy of 96% and had balanced precision, recall, and F1-scores for both classes.
- **Perceptron:** The Perceptron had an 86% accuracy and showed good recall for the "1" class but lower precision, making it better at detecting "1" but not as precise.
- **Random Forest:** Random Forest demonstrated exceptional performance with a 99% accuracy and high precision, recall, and F1-scores for both classes.
- **SVM:** The Support Vector Machine had an accuracy of 97% with strong precision and recall for both classes.

Set B (K-fold Cross-Validation with Classification Accuracy):

In Set B, I used K-fold cross-validation, which provides a more robust assessment of model performance. Here are the results:

- All classifiers showed similar accuracy as in Set A, with minor variations.
- The highest accuracy was achieved by the **MLP classifier** (93.75%), indicating that it consistently performs well across different data splits.
- The **Perceptron** had the lowest average accuracy (82.71%), suggesting less consistency in performance.

Set C (Repeated Random Train-Test Splits with Logarithmic Loss):

In Set C, I used repeated random train-test splits and assessed classifiers based on logarithmic loss. Here are the results:

- Most classifiers had low logarithmic loss, indicating good probability estimation.
- The **Random Forest** had the lowest average logarithmic loss (0.0623), suggesting it provides accurate probability estimates for class labels.
- The **Perceptron** doesn't provide probability estimates and is therefore listed as "N/A."

In summary, Set A provides insights into classifier performance with a single train-test split, Set B uses K-fold cross-validation for more robust accuracy measurements, and Set C assesses probability estimation using logarithmic loss.

Based on the results, perform algorithm/hyperparameter tuning (at least 3) of the chosen ML algorithm.

ML Algorithm: Random Forest
Sampling Technique - Train/Test Split
Classification Metrics - Accuracy

	Random Forest Hyperparameters			
	N_Estimators	**Max_Depth**	**Min_Samples_Split**	**Accuracy**
Model I	50	None	2	99.7%
Model II	50	None	5	99.4%
Model III	50	None	10	99.1%
Model IV	50	10	2	99.7%
Model V	50	10	5	99.7%
Model VI	50	10	10	99.1%
Model VII	50	20	2	99.7%
Model VIII	50	20	5	99.4%
Model IX	50	20	10	99.1%
Model X	100	None	2	99.7%
Model XI	100	None	5	99.7%
Model XII	100	None	10	99.4%
Model XIII	100	10	2	99.7%
Model XIV	100	10	5	99.7%
Model XV	100	10	10	99.4%
Model XVI	100	20	2	99.7%
Model XVII	100	20	5	99.7%
Model XVIII	100	20	10	99.4%
Model XIX	200	None	2	99.7%
Model XX	200	None	5	99.7%
Model XXI	200	None	10	99.4%
Model XXII	200	10	2	99.7%
Model XXIII	200	10	5	99.4%
Model XXIV	200	10	10	99.4%
Model XXV	200	20	2	99.7%
Model XXVI	200	20	5	99.7%
Model XXVII	200	20	10	99.4%

Results interpretation:
Based on the results of the Random Forest hyperparameter tuning, the following observations can be made:

- The accuracy of the Random Forest model is consistently high across different hyperparameter settings, with accuracies ranging from 99.1% to 99.7%.
- Varying the number of estimators (n_estimators) from 50 to 200 does not significantly impact accuracy.
- Changing the maximum depth (max_depth) of the decision trees within the Random Forest also has minimal impact on accuracy.
- Altering the minimum number of samples required to split an internal node (min_samples_split) does not result in substantial changes in accuracy.
- The highest accuracy achieved is 99.7%, which is consistent across multiple models (e.g., Model I, Model IV, Model V, etc.) with different hyperparameter settings.

In summary, the Random Forest model appears to be robust to changes in these hyperparameters and consistently delivers high accuracy, making it a reliable choice for this classification task.

II REGRESSION

Train the **regression dataset** using various machine learning algorithms designed for regression. Evaluate and compare these models by applying different resampling techniques and utilizing appropriate performance metrics.

Regression Dataset
Dataset Name : Used Car Price Prediction
Features: model; brand; year; transmission; mileage; fuel_type; price

Set A
Resampling Technique : Split into train and test sets
Regression Metric : Mean Absolute Error (MAE)

Algorithm (Regression)	
CART (Classification and Regression Trees)	Mean Absolute Error = 196970.82
Elastic Net	Mean Absolute Error = 265543.15
Gradient Boosting Machines (AdaBoost)	Mean Absolute Error = 306335.08
K-Nearest Neighbors (K-NN)	Mean Absolute Error = 280099.82
Lasso Regression	Mean Absolute Error = 199079.56
Ridge Regression	Mean Absolute Error = 209465.26
Linear Regression	Mean Absolute Error = 199070.78
Multi-Layer Perceptron (MLP)	Mean Absolute Error = 360855.99
Random Forest	Mean Absolute Error = 180124.27

Set B *(should use different resampling technique and regression metric)*

Resampling Technique: K-fold Cross Validation

Regression Metric: R^2

ML Algorithm (Regression)	
CART (Classification and Regression Trees)	Mean R-squared (R^2) = -0.56
Elastic Net	Mean R-squared (R^2) = 0.15
Gradient Boosting Machines (AdaBoost)	Mean R-squared (R^2) = -0.64
K-Nearest Neighbors (K-NN)	Mean R-squared (R^2) = -0.08
Lasso Regression	Mean R-squared (R^2) = 0.53
Ridge Regression	Mean R-squared (R^2) = 0.53
Linear Regression	Mean R-squared (R^2) = 0.53
Multi-Layer Perceptron (MLP)	Mean R-squared (R^2) = -0.38
Random Forest	Mean R-squared (R^2) = 0.38

Results interpretation (Set A and Set B):

Here's the results interpretation for Set A and Set B based on the different resampling techniques and regression metrics:

Set A:

- In Set A, I used a train-test split resampling technique and evaluated the regression models using the

Mean Absolute Error (MAE) metric.

- Among the regression algorithms, Random Forest had the lowest MAE, indicating that it performed the best in terms of minimizing the absolute errors between predicted and actual prices.

- Random Forest achieved a relatively low MAE of 180,124.27, which suggests that it is a good choice for this regression task. Other models also achieved varying levels of MAE, with Lasso Regression, Linear Regression, and CART also showing promising results.

Set B:

- In Set B, I used K-fold Cross Validation as the resampling technique and assessed the models using the R-squared (R^2) metric. R^2 measures the proportion of the variance in the dependent variable (price) that is predictable from the independent variables.

- Unfortunately, the R^2 values for most models are not very informative. Many models, including CART, AdaBoost, K-NN, and MLP, had negative R^2 values. This indicates that these models performed poorly in explaining the variance in used car prices.

- Lasso Regression, Ridge Regression, and Linear Regression had the highest R^2 values, all around 0.53, suggesting that they explained around 53% of the variance in used car prices. This may be seen as a moderate level of performance.

- Random Forest achieved an R^2 value of 0.38, indicating that it explained 38% of the variance in prices. While not the best result, it still outperformed many other models in Set B.

Overall Interpretation:

- Based on Set A and Set B, Random Forest appears to be a robust choice for regression in this dataset. It had the lowest MAE in Set A and achieved a decent R^2 value in Set B.

- Lasso Regression, Ridge Regression, and Linear Regression also performed reasonably well in Set B, providing moderate explanatory power for used car prices.

Based on the results, perform at algorithm tuning (at least 3) of the chosen ML algorithm.

EXAMPLE:

ML Algorithm: Random Forest
Sampling Technique - Train/Test Split
Regression Metrics - MAE

	Random Forest Hyperparameters			
	N_Estimators	**Max_Depth**	**Min_Samples_Split**	**MAE**
Model I	50	None	2	180439.18
Model II	50	None	5	185562.72
Model III	50	None	10	190003.10
Model IV	50	10	2	202510.47
Model V	50	10	5	204550.98
Model VI	50	10	10	206597.03
Model VII	50	20	2	183328.25
Model VIII	50	20	5	187038.28
Model IX	50	20	10	192585.02
Model X	100	None	2	182520.77
Model XI	100	None	5	186187.54
Model XII	100	None	10	188025.19
Model XIII	100	10	2	203340.83
Model XIV	100	10	5	204020.48
Model XV	100	10	10	206180.83
Model XVI	100	20	2	181403.25
Model XVII	100	20	5	186884.36
Model XVIII	100	20	10	189878.21
Model XIX	200	None	2	181609.10
Model XX	200	None	5	184501.76

Model XXI	200	None	10	190394.43
Model XXII	200	10	2	203063.00
Model XXIII	200	10	5	202895.77
Model XXIV	200	10	10	204924.82
Model XXV	200	20	2	181785.35
Model XXVI	200	20	5	184505.19
Model XXVII	200	20	10	189922.55

Results interpretation:

- The hyperparameter tuning process revealed that different combinations of hyperparameters had varying effects on the model's accuracy. I evaluated the model's performance using the MAE metric, which measures the average absolute difference between the predicted and actual car prices. Lower MAE values indicate a better fit to the data.

- The results of hyperparameter tuning showed that the optimal configuration for the Random Forest model was Model I with the following hyperparameters:
  N_Estimators: 50
  Max_Depth: None
  Min_Samples_Split: 2

- This model achieved an MAE of **180,439.18**, which represents the smallest deviation between the predicted car prices and the actual prices in the test dataset. It is important to note that hyperparameter tuning is crucial in improving the model's performance and achieving more accurate predictions. The Random Forest algorithm with these optimized hyperparameters can now be confidently used for predicting used car prices with high accuracy and reliability. The chosen hyperparameters should be used for future predictions, as they demonstrated the best performance in minimizing prediction errors. Additionally, further evaluation, such as cross-validation or testing on unseen data, can be performed to ensure the model's generalizability and reliability.

Submit the following:

a. Pdf copy of the results.
   Source Code: https://github.com/kiyojiii/All_ITD105
   PDF Link:
   https://drive.google.com/drive/folders/1GNB0AvxWVT2uokStK97WcVJg59U5EvYO?usp=sharing

b. Video link demonstrating the case study.
   GDrive Link:
   https://drive.google.com/drive/folders/1GNB0AvxWVT2uokStK97WcVJg59U5EvYO?usp=sharing
   Youtube Link:
   https://youtu.be/B3j8-HVlPOs