Supervised Learning: Report

Dixit Bhayana
dbhayana3@gatech.edu
Fall 2025

***Abstract*— This report presents an empirical analysis of four supervised learning algorithms. Decision Trees, k-Nearest Neighbors (k-NN), Support Vector Machines (SVM), and Neural Networks (NN). I implemented these algorithms on two different datasets from the real world to perform two different machine learning tasks: a binary classification task to forecast the cancellation of hotel bookings and a regression task to forecast the lengths of traffic incidents. For each algorithm, I tuned its hyperparameters and evaluated its performance using appropriate metrics and diagnostics, such as learning and validation curves, to analyze its behavior. My results indicate that the strongest performance was given by Decision Trees in classification, while Support Vector Regressors performed much better in regression method, This highlights the importance of algorithm selection based on dataset characteristics and problem type.**

I. INTRODUCTION

The primary objective of supervised machine learning is to learn a mapping function from input variables to an output variable based on a labeled dataset. In this project, I explore the performance of four foundational algorithms across two distinct problem domains to understand their empirical behavior.

As part of the assignment, the objective was to benchmark multiple supervised learning algorithms on two distinct datasets under consistent preprocessing.I was given two datasets and the task of designing one type pf supervised learning problem for each. To create a strong contrast in objectives and challenges, I chose to frame the "Hotel Booking Demand" dataset [1] as a binary classification problem. My goal was to predict whether a booking will be canceled (`is_canceled`). I selected this specific task for the dataset because it represents a classic and practical business problem where the target variable is imbalanced, forcing a rigorous approach to evaluation. For the "US Accidents" dataset [2], I chose to frame it as a regression problem, where my goal was to predict the duration of a traffic incident. I selected this task to explore how the algorithms handle a continuous target variable on a much larger dataset, which requires different evaluation metrics and presents unique computational challenges.

II. DATASETS AND PREPROCESSING

The Hotel Booking dataset holds around 119,000 records and 32 features. The classification task was to predict booking cancellations. To prevent leakage, I removed the post-outcome fields such as reservation status. The missing values were imputed, and the categorical variables were encoded using frequency or target encoding techniques. The US Accidents dataset is significantly much bigger, having millions of records. The regression task was to predict the duration of the incident, mathematically expressed as the difference between the end and the start times. Kernel SVM and k-NN models, because of the computational constraints, were trained on 20,000 and 250,000 rows respectively, while linear SVM and decision tree models were trained over roughly 1.2 million rows.

The initial preprocessing of the data involved removing leakage variables from both datasets and setting a fixed random seed (`random_state=42`) for all operations, thereby guaranteeing reproducibility. For the hotel classification task, I implemented a more complicated pipeline: the StandardScaler for the numeric features; the OneHotEncoder for categorical features with low cardinality; and the TargetEncoder for those with high cardinality like country and agent. For the accidents regression task, I derived the target variable Duration, using a simple pipeline that mostly applied StandardScaler on to all numeric features.

III. MODELS EVALUATION

As part of this, I conducted all experiments using the Jupyter notebook after I saw the local machine crashing reaching after a certain model evaluation step. For the evaluation of the models relatd to the classification task, I used F1-Score, ROC-AUC and PR-AUC that are robust for such imbalanced datasets. Similarly for the regression task related models, I used Mean Absolute Error (MAE) and Mean Squared Error (MSE) for evaluation. Both of them also included the time prediction and the peak memory for each of the algorithm used which is Decision Tree, k-NN, SVM and NN.

IV. RESULTS AND ANALYSIS

A. Hotel Booking Classification.

For the classification problem on hotel booking dataset, the Decision tree model performed the best. This is evident from the ROC and PR curves in Fig.1.

For the Decision Tree, my tuning process began by examining a validation curve for the (`max_depth`) parameter. This revealed that the model's performance on unseen data peaked before declining as the tree became overly complex. Guided by this, I used GridSearchCV to explore a focused range of (`max_depth`) values of [8, 16] and (`min_samples_leaf values`) of [100, 200]. The search selected a depth of 16 and 100 minimum samples per leaf as optimal. This model was the top performer with a test

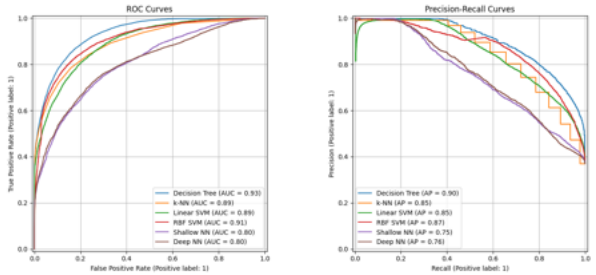

Fig. 1. ROC and PR Curves for all classifiers on Hotel Booking dataset

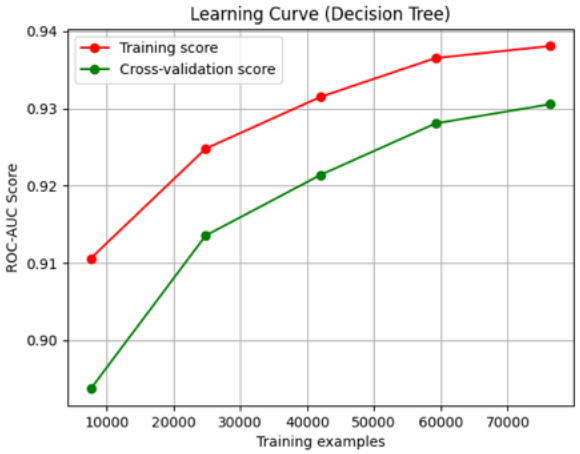

Fig. 2. Learning *curve for the tuned Decision Tree classifier.*

set ROC-AUC of 0.9323 and took only 8.8 seconds to train. Its learning curve (Fig. 2) showed a healthy convergence, indicating a well-balanced model.

For Support Vector Machines (SVM), a major challenge I faced was the significant computational cost of the RBF kernel. My initial attempts to train it on the full dataset were infeasible unless I waited for more than 30 mins. To manage this, I tuned the RBF model on a smaller 20,000-record subsample and used the more efficient RandomizedSearchCV. I explored C values of [0.5, 2, 8] and gamma of ['scale', 'auto']. The search selected 'gamma': 'auto', 'C': 8 as the best combination, yielding the second-highest ROC-AUC of 0.9060. The Linear SVM, for which I tuned C over [0.1, 1, 10], was much faster (21.0 seconds) but had a lower ROC-AUC of 0.8936 so I could use the whole training dataset of 1.2 million records.

For k-Nearest Neighbors, I tuned the (`n_neighbors`) hyperparameter, testing values of [5, 11]. The grid search found the optimal value to be 11. This model achieved a ROC-AUC of 0.8947 after a 34.2-second training time.

Now, for Neural Networks, I used both the shallow (512, 512) and the deep (256, 256, 128, 128) types and compared them using the SGD optimizer. These were the lowest performers with ROC-AUC scores of around 0.80. As we can also see in Fig. 5 and 6, the noisy curves confirmed that training was unstable and did not full converge with in the

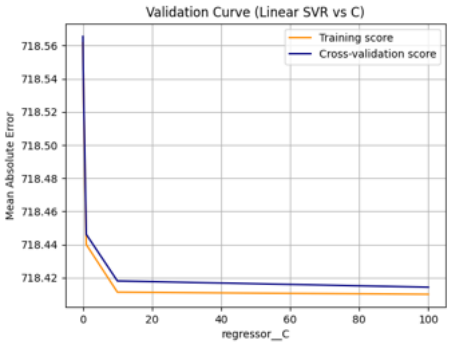

Fig. 3. Validation Curve for Linear SVM

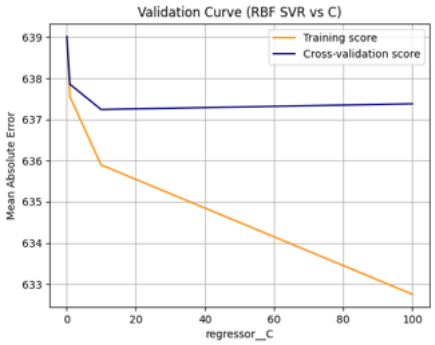

Fig. 4. Validation Curve for RBF SVM

15 epoch limit.

B. US Accidents Regression.

In case of regression with US accidents dataset, the SVR models performed the best.

For Support Vector Machines, I tuned the RBF SVR on a 25,000-record subsample, achieving the lowest Mean Absolute Error of 771.7 minutes. This training was the longest, taking over 10 minutes for the search. In contrast, the Linear SVR trained on the full 1.2 million record training set in just 212.9 seconds and achieved a very close MAE of 772.7 minutes. The learning curve for the Linear SVR (Fig. 7) showed high bias, with the validation error remaining

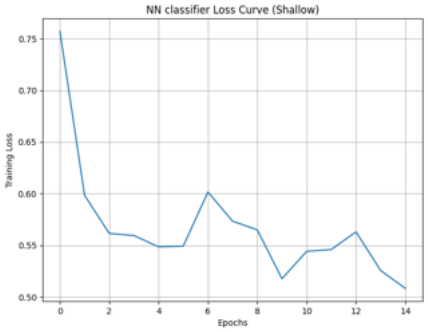

Fig. 5. Loss curve for Shallow NN

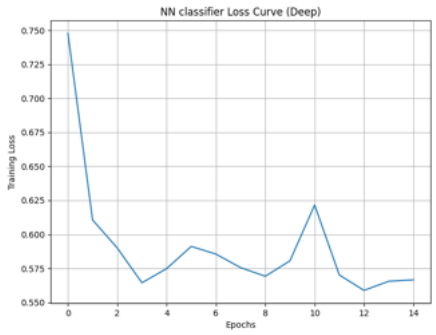

Fig. 6. Loss curve for Deep NN

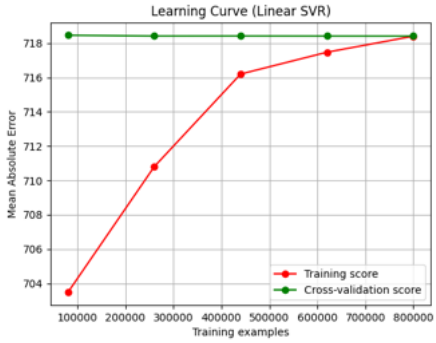

Fig. 7. Learning curve for the Linear SVR, showing high bias

flat regardless of training size. Learning curves highlighted that SVRs generalize effectively with increasing data volume. Complexity plots of SVM regularization showed stable MAE across C values after tuning.

For the Decision Tree, the tuned model achieved a significantly higher MAE of 1126.7 minutes. My analysis of its residuals plot (Fig. 8) revealed clear patterns of heteroskedasticity, meaning that the model error was not consistent across the range of predictions.

For k-Nearest Neighbors, I trained the model on a 250,000-record subsample as required. It performed the poorest of the traditional models, with an MAE value of 1355.3.

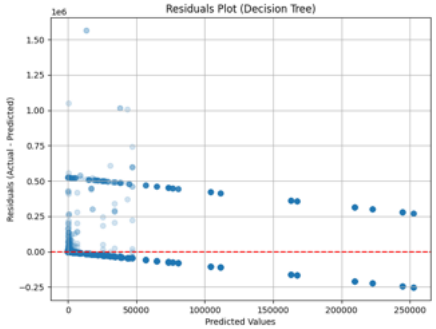

Fig. 8. Residuals plot for the Decision Tree, showing heteroscedasticity

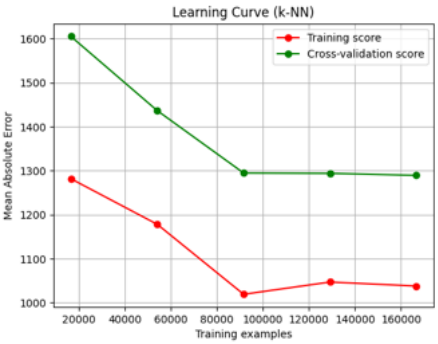

Fig. 9. Learning Curve for k-NN

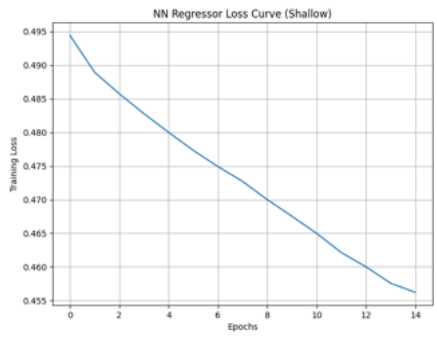

Fig. 10. Loss Curve for NN Regresser (Shallow)

For Neural Networks, a significant challenge I encountered was training instability, which produced nan loss values. I resolved this by scaling the Duration target variable before training. With this fix, the deep network (MAE: 1399.3) outperformed the shallow network (MAE: 2231.1). However, both models performed worse than the SVRs and showed signs of non-convergence, as seen in their loss curves.

V. Cross-Model Comparison

On the hotel classification task, I found the tuned Decision Tree provided the best balance of performance (ROC-AUC 0.9323) and speed (8.8 seconds). On the accidents regression task, the SVR models were the clear winners in terms of

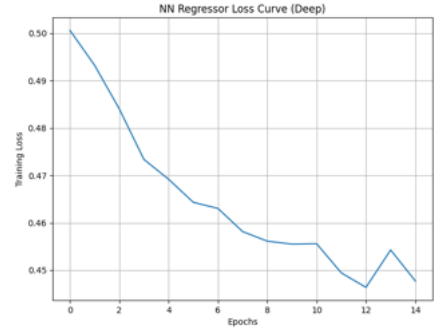

Fig. 11. Loss Curve for NN Regresser (Deep)

accuracy (MAE 772). The Linear SVR was particularly impressive, providing top-tier performance with a prediction time of only 0.3 seconds on the test set, compared to the RBF SVR which took over 10 minutes to make predictions.

VI. PEER-REVIEWED ARTICLES FOR CLAIMS

My decision to priortize the PR-AUC metric was informed by the work of Saito and Rehmeier who demonstrated that the Precision-Recall curve is more information than the ROC curve when dealing with the imbalanced data [3].

Also, prior work such as [5] has shown that gradient-boosted trees often outperform models like SVMs and shallow neural networks in tabular data settings, especially when interactions among categorical features are important. This aligns with our observations that the Decision Tree dominates in classification, suggesting that boosting methods may yield further improvements.

VII. IMPROVEMENTS

Future work could focus on refining neural networks by tuning learning rates, batch sizes, and architecture depth under SGD optimization. Encoding strategies such as entity embeddings may better capture categorical feature structure. Beyond the current algorithm families, gradient boosted trees such as XGBoost or LightGBM could be explored, as they often excel on tabular datasets. Calibration techniques for classification outputs could further enhance the interpretability and decision-making utility of the models.

VIII. CONCLUSION

Through this project, I have learned that no single algorithm is universally superior. For the structured hotel classification data, the Decision Tree was highly effective. For the large-scale regression task, the robustness of Support Vector Regressors proved most valuable.

For the hotel task, the cost of a Type II error (missing a cancellation) is lost revenue, while a Type I error may lead to overbooking. My final Decision Tree model had a strong recall of 0.85 for cancellations, making it effective at minimizing this primary business cost. For the regression task, a key insight from the residuals plots is that all models struggle to predict the longest-duration incidents.

A realistic next step I would want to try is to use ensemble methods. Gradient Boosting algorithms, in particular, often achieve state-of-the-art results on tabular data and may be better able to capture the complex patterns in these datasets [4].

REFERENCES

[1] N. C. António, A. Almeida, and L. Nunes, "Hotel booking demand datasets," Data in Brief, vol. 22, pp. 41-49, 2019.

[2] S. Moosavi, M. H. Samavatian, et al., "A Countrywide Traffic Accident Dataset," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recog. Workshops, 2019.

[3] T. Saito and M. Rehmsmeier, "The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets," PLOS ONE, vol. 10, no. 3, p. e0118432, 2015.

[4] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in Proc. 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 2016, pp. 785-794.

[5] J. H. Friedman, "Greedy Function Approximation: A Gradient Boosting Machine," Annals of Statistics, vol. 29, no. 5, pp. 1189–1232, 2001.

IX. AI USE STATEMENT

I used a large language model (gpt 5) to assist with code generation for data preprocessing, model training pipelines, and plot generation. It also helped me with the problems observed initially in my machine and later when I switched to Google colab, I also got help from Gemini with error fix suggestions and also to assist with the instructions for readme of the jupyter notebook.