Car Evaluation

Alexandre Chaker, Amir Salah, Caleb Asress, Dennis Hantman, Myles Ezeanii

Max Mahdi Roozbahani

CS4641/CS7641

03 December 2024

Introduction

It's undeniable that cars are a central part of American culture. Over ninety percent of American households own at least one vehicle, and over seventy percent of commuters rely on cars for transportation. Because of this, the choice of what car to purchase is an extremely important one, and has a significant impact on the owner's lifestyle. This decision is especially crucial for those who may not have access to resources or individuals who can help them make the best choice. Thus, our group decided to make a car evaluation system that recommends vehicles based on their features. In one study regarding a similar concept, Singh et al. proposed a car recommendation system that utilized NLP and machine learning techniques to analyze both user reviews and user preference to ultimately recommend a car or not. Bauer et. al. have also analyzed car data with a specific focus on the used market, and Mou et. al. have used machine learning to help customer decision making by using machine learning algorithms such as SVM and random forest. Our project has a similar target with more of a focus on the features of the car, and stands out with our unique use of models, including a neural network.

Our dataset contains information on price, size, safety, tech, and more features for over a thousand different cars. It includes the target concept as well as three intermediate concepts:

price, tech, and comfort. We thought this would be a good dataset to base our models on as there are multiple features we can use to judge the car.

Dataset Link: https://archive.ics.uci.edu/dataset/19/car+evaluation

Problem Definition

The process of selecting the proper car can be difficult, particularly for people who lack considerable automotive knowledge or access to adequate resources. Our project tries to solve this problem by creating an automobile recommendation system using machine learning techniques. The major goal is to develop a model capable of categorizing cars as "unacceptable," "acceptable," "good," and "very good" based on their qualities. This will allow consumers to make informed selections based on their preferences and needs.

This project's dataset consists of a variety of car parameters from the UCI Car Evaluation dataset, including the purchase price, maintenance cost, number of doors, seating capacity, luggage boot size, and safety ratings. The dataset is well-suited to this task since it includes a wide variety of features that can influence car acceptability. By emphasizing these characteristics, our project aims to create an effective tool that streamlines the car purchasing process for prospective purchasers.

To achieve this, we used supervised learning models, which are intended to examine and forecast the acceptability of a car based on the attributes presented. The models were chosen because they are good at handling categorical and numerical data, resulting in robust and dependable suggestions.

Methods

To reach our goal of a machine learning solution that can help consumers decide on cars, we implemented several models based on the Car Evaluation data set. The features included in this data set are buying price, maintenance price, number of doors, seating capacity, luggage boot size, and estimated car safety.

We took many preprocessing steps before analyzing the data. First, we split the dataset into training and test sets. We do this in order to gauge how well the model is learning the training data as it goes along. Then, we use mean imputation to replace any missing data points in our set with the mean. This allows us to have a more complete set to work with, and increases our model accuracy. Then, we separate the categorical features, such as car safety, from the numerical features, such as number of doors. We then create transformers for our categories, using the mean of values for our numerical features and One-Hot encoding on our categorical features. Finally, we fit the preprocessor on the training data.

Now that the data is preprocessed, we must start supervised learning. We defined several models to achieve this. This first one is a Neural Network whose architecture consists of an input layer with 6 features, a hidden layer with 64 units and ReLU activation, and an output layer with 4 units and softmax activation. The second is a KNN model with 5 neighbors. The final one is an XGBoost classifier with 100 trees, 4 classes, 6 max depth, and a learning rate of 0.1.

We chose each type of model for a different reason. Our first model was a Neural Network because we had the most confidence in it to handle the large car dataset and found it to be a powerful model for generating predictions. We then chose the KNN model because of its strength in classifying data as well as its simplicity and transparency – however, we anticipated some challenges because of the complex nature of our dataset. Finally, we added the XGBoost model because of its very high prediction power as well as many of its desirable features like built-in regularization and efficiency.

Scikit-learn Tools Used:

- OneHotEncoder() or LabelEncoder() for encoding.

- train_test_split() to split data.

- SimpleImputer for missing value replacement.

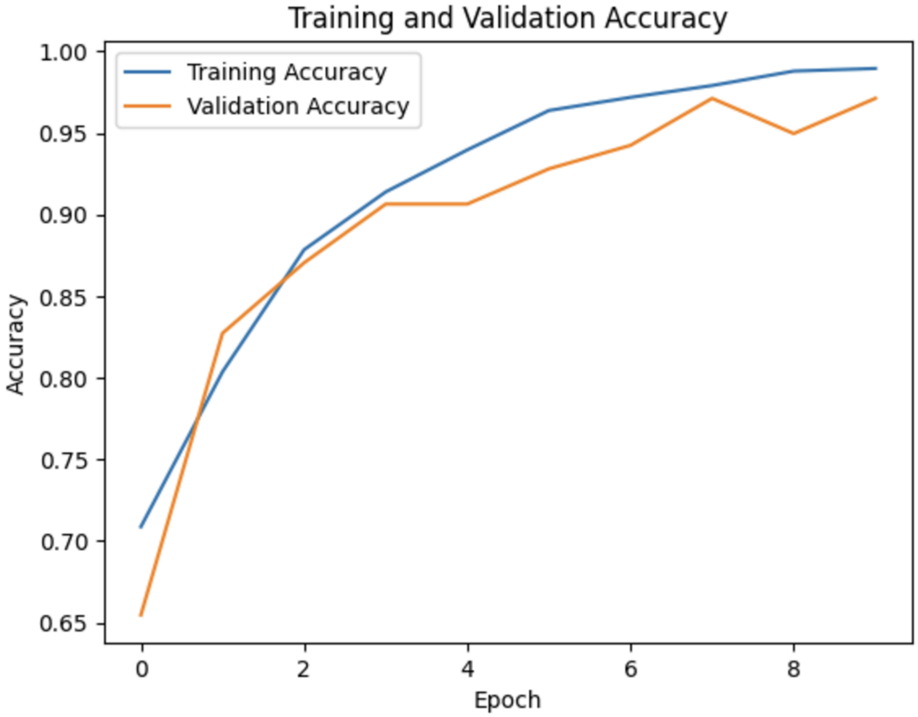
- ColumnTransformer() for column preprocessing.

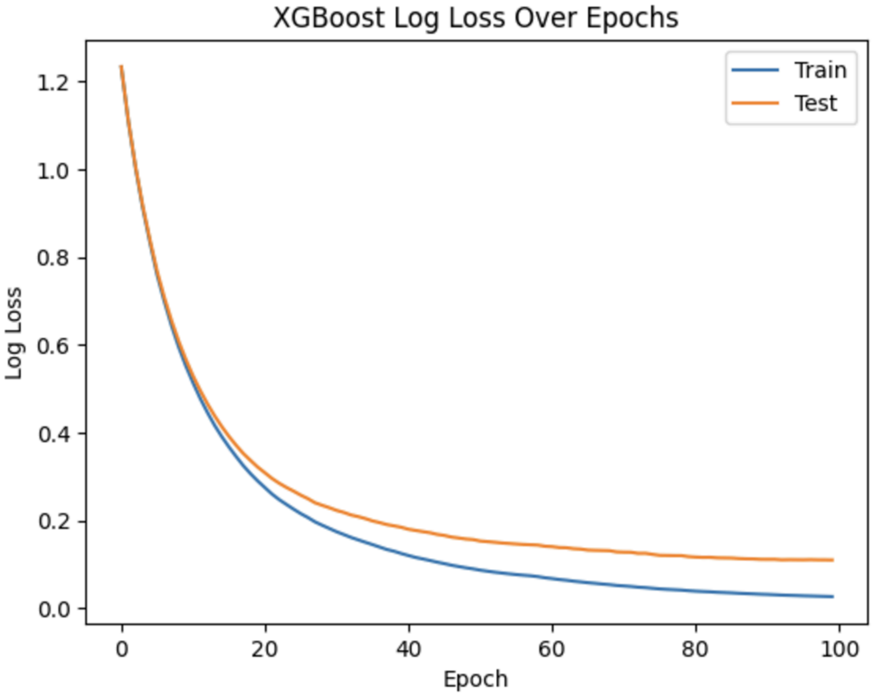
Results and Discussion:

For our project, we used five separate quantitative metrics to determine the efficiency of the model. Firstly, we used **confusion matrices** to break our results down to true and false positives and negatives, allowing us to see where our model is making mistakes and determine if these false positives and negatives are costly to our model. For our testing metrics, we used **accuracy** to help us find its overall correctness by measuring the proportion of cars correctly identified. We also used **precision and recall** to further determine the consequences and how significant they are. Precision checks how many cars predicted as acceptable are actually acceptable while recall measures how many of the actual acceptable cars were correctly classified. To ensure our models are not making false positive predictions, we used the **log loss function** to measure the loss of each model. This is done by summing the predicted probability of a true positive divided by the number of predictors. Finally, we used **Receiver Operating Characteristic - Area Under the Curve** (ROC-AUC) to see if the dataset is imbalanced. Our ROC-AUC will be equal to the area under the curve plotting the True Positive Rate.

Our goal was to create a model which can accurately determine the acceptability of a car based on the standards we provide. This section will ultimately discuss how our results can help us decide which model would allow for users to decide most accurately whether or not the cars they intend to use are adequate.

Neural Network:

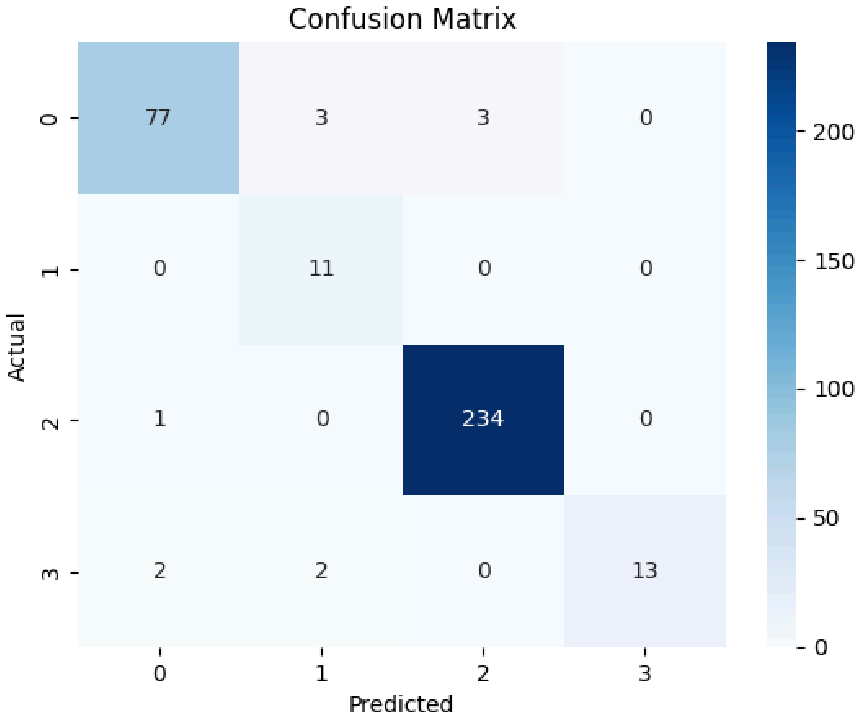


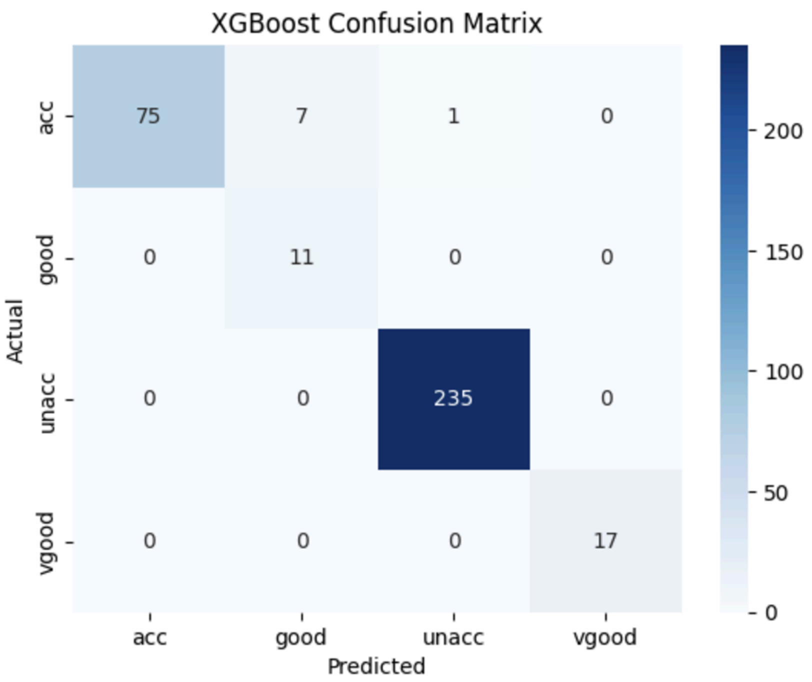


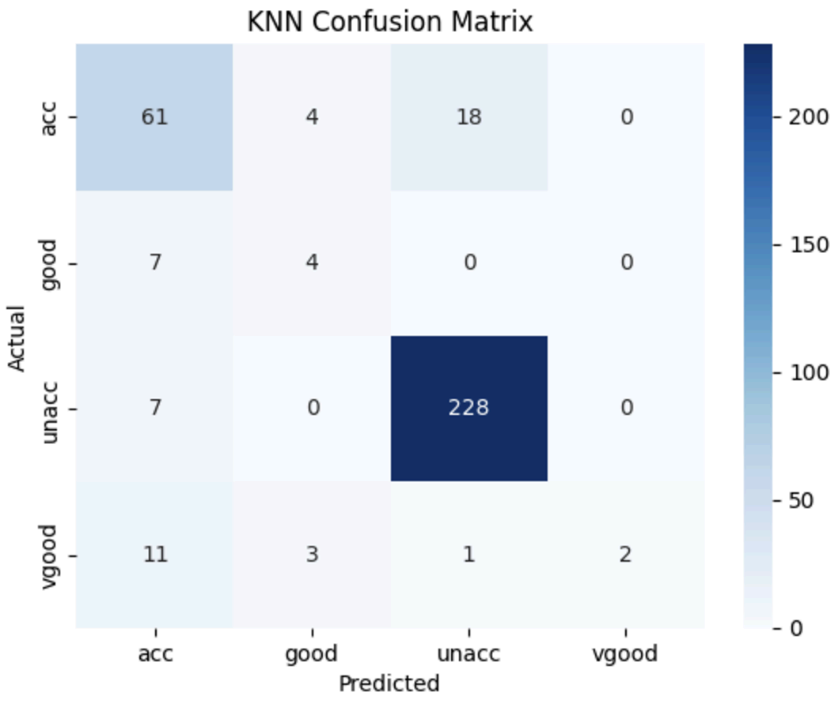
Graph not provided for KNN as it does not use epochs.

Looking at the graphs above, we see that the addition of 8 epochs significantly improved the performance of our neural network. Two hidden layers with ReLU activation functions, a large dataset, and preprocessing steps such as imputation and one-hot encoding likely improved performance as well. Looking at our XGBoost graph, we see that the log loss function begins to plateau around 70 epochs. Hence, the addition of 80 epochs in our XGBoost model significantly aided in decreasing the total log loss.

Neural Network Confusion Matrix (0 - acc, 1 - good, 2 - unacc, 3 - vgood):







Looking at our confusion matrices, we see generally similar results. There is a heavy bias towards “unacceptable” and “acceptable” predictions, however, this is likely due to a bias in the training data itself. Otherwise, the Neural Network and the XGBoost models had very similar results, while KNN had slightly less true positives for “acceptable” classifications. This may suggest that our KNN model performed worse than the other two models tested. However, test result metrics can give us a better understanding of how the models performed.

Model	**Accuracy**	**Precision**	**Recall**	**ROC-AUC**	**Loss**
Neural Network	0.98	0.99	0.98	1.00	0.05
KNN	0.85	0.86	0.85	0.94	0.76
XGBoost	0.98	0.90	0.98	0.99	0.11

Above are the test results of each model used in our project rounded to two decimals. We first see that our Neural Network performed extremely well. With an accuracy of 0.98, almost all of our neural network’s predictions were correct. This means our neural network performed well in identifying both true positives and true negatives. Furthermore, with a precision of 0.99, almost all predicted positives were true positives, hence our model rarely makes false positive predictions. This is an important metric for our project as falsely recommending cars could cost users large amounts. A recall of 0.98 and ROC-AUC of 1.00 also suggests our neural network model was successful in correctly classifying cars and distinguishing between classes. Finally, a loss of 0.05 indicates that the neural network’s predictions were close to the true values. Hence, our neural network performed very well and would be an excellent choice for our product.

Our XGBoost model performed similarly to our neural network in most metrics. The only significant difference was that the XGBoost had a slightly lower precision. This means that our XGBoost model made slightly more false positive predictions than our neural network. In the context of our project, high precision is an important metric. While XGBoost can often perform much faster than a neural network, ensuring that our users are not presented with incorrect recommendations is of high priority. Hence a neural network may still be a better option.

Our KNN model performed the worst, suffering in all metrics compared to our other models. One metric that may have had a large impact in our KNN model is the loss of 0.76. This is a relatively high loss, especially compared to neural network's 0.05 and XGBoost's 0.11. One main reason behind the loss is that the KNN model likely suffered from a high-dimensionality dataset. Each car in the dataset has several different features, resulting in a high dimensionality. This means that the distances KNN measures become very similar and it then struggles to distinguish between classes. One way to improve this model would be to implement some form of dimensionality reduction like PCA. Overall though, KNN did not perform well and should be avoided in our product.

Ultimately, based on the results discussed above, if our dataset is not too big and speed is not of importance, then the Neural Network model is the best choice for our project. It excels in all metrics, providing our users with accurate and precise car recommendations. If speed is of importance, then XGBoost is a solid choice given that its results are similar to those of the neural network other than precision. KNN did not perform well compared to the other models and therefore is best to be avoided.

Gantt Chart:

https://docs.google.com/file/d/1gD6TI02N_67U_YAIVeduaDORvfWMKicr/edit?usp=docslist_api&filetype=msexcel

Contribution Table:

Member Name	**Final Contribution**
Dennis Hantman	Results and Discussion
Alexandre Chaker and	KNN Model
Myles Ezeanii	Video Presentation
Amir Salah	Methods, Streamlit, Revised Intro
Caleb Asress	XGBoost Model

Works Cited

[1] I. Bauer, L. Zavolokina, and G. Schwabe, “Is there a market for trusted car data?,” Electronic Markets, vol. 30, no. 2, pp. 211–225, Sep. 2019, doi: https://doi.org/10.1007/s12525-019-00368-5.\n

[2] A. Das Mou, P. K. Saha, S. A. Nisher, and A. Saha, “A Comprehensive Study of Machine Learning algorithms for Predicting Car Purchase Based on Customers Demands,” 2021 International Conference on Information and Communication Technology for Sustainable Development (ICICT4SD), Feb. 2021, doi: https://doi.org/10.1109/icict4sd50815.2021.9396868.\n

[3] S. Singh, S. Das, A. Sajwan, I. Singh, and A. Alok, “Car Recommendation System Using Customer Reviews,” *International Research Journal of Engineering and Technology (IRJET)*, vol. 9, no. 10, pp. 983–989, Oct. 2022