Milestone III Report - Pilot Study
Predicting Used Car Prices using Machine Learning

CSCE 3602 - Fundamentals of Machine Learning

Omar Bahgat
Computer Engineering Department
The American University in Cairo
Cairo, Egypt
omar_bahgat@aucegypt.edu

Omar Saleh
Computer Engineering Department
The American University in Cairo
Cairo, Egypt
Omar_Anwar@aucegypt.edu

Introduction

After conducting further cleaning and processing of the dataset to enhance its quality from the previous phase, our dataset now contains 748,000 instances and 975 features. The current phase involves selecting the most optimal model for the project. This report encompasses the various models we explored, along with the experimental setup for each, parameter selection, performance evaluation, and an analysis of their suitability for the problem at hand. We will examine the pros and cons of each model to make an informed decision regarding the most fitting approach for predicting used car prices.

Experimental Setup

Regarding the training of our data, we focused on using a consistent setup and we divided our dataset using an 80-20 train-test split for regression models, while opting for binning the price label into 8 numerical labels for classification models. This approach allowed us to effectively train our models on a substantial portion of the data while reserving a separate subset for evaluation.

For regression models, we primarily relied on the R-squared (R^2) metric. R^2 measures the proportion of the variance in the target variable explained by the model's predictions. A higher R^2 score, closer to 1, indicates a better fit of the model to the data.

For the classification models, we utilized accuracy, precision, and recall metrics as well to assess model performance. Accuracy measures the overall correctness of predictions, precision quantifies the proportion of true positive predictions among all positive predictions made, and recall calculates the proportion of true positive predictions among all actual positive instances.

By employing these concise performance metrics, we aim to comprehensively evaluate the effectiveness of our machine learning models across both regression and classification tasks.

Supervised Machine Learning Models Implemented

We implemented the following machine learning models: K-Nearest Neighbors (KNN), Linear Regression, Decision Trees, Random Forests, XGBoost, Neural Networks, Logistic Regression, and Naive Bayes. A detailed description of each model is provided below.

K-Nearest Neighbors (KNN)

Model Description: K-Nearest Neighbors (KNN) is an instance-based learning algorithm utilized for both classification and regression problems. For regression, KNN predicts the value of a target variable by averaging the values of its k nearest neighbors in the feature space. In the context of predicting used car prices, KNN considers the similarity of features among different cars to estimate their prices, making it a suitable candidate for the task.

Parameter selection: A critical aspect of KNN is the selection of the hyperparameter *k*, which determines the number of neighbors considered during prediction. As a rule of thumb, the value of *k* should be less than the square root of the number of instances. After some experimentation with a random sample of the dataset, a *k* value of 100 was chosen as it struck a balance between model complexity and performance. This choice allows for generalizing well and capturing sufficient data while reducing the risk of overfitting. The Euclidean distance served as the similarity measure.

Performance evaluation: The calculated R^2 score for the KNN model is 0.855, indicating a high degree of predictive accuracy. However, KNN may struggle with computational efficiency, especially with large datasets, due to its reliance on storing and computing distances to all data points and sorting them to find the *k*-nearest neighbors. From a user perspective, this model would be inefficient and not suitable for real-time applications as it will take a lot of time to predict a single car.

Linear Regression

Model Description:

Linear Regression is a straightforward and widely used method for modeling the relationship between a dependent variable and one or more independent variables. It assumes a linear relationship between the variables, aiming to find the best-fitting linear equation that predicts the dependent variable based on the independent variables.

Parameter selection:

For Linear Regression, no specific hyperparameters need to be optimized as it's a relatively simple model.

Performance evaluation:

After fitting the Linear Regression model to the training data and making predictions on the test data, the R-squared score is calculated to evaluate the performance of the model and it turns out to be negative. This indicates that the model fits the data worse than a horizontal line. However, this is expected as the dataset is huge and a single line isn’t expected to fit the data well.

Decision Trees

Model Description: Decision Trees are versatile and widely used in both classification and regression tasks. In regression, they partition the feature space into regions, assigning a constant value to each region based on the average target value of the training instances falling within it. In predicting used car prices, Decision Trees analyze various car features to create a tree-like structure, aiding in estimating prices.

Parameter selection: For the Decision Tree model, the squared error criterion was selected as the splitting criterion, aiming to minimize the mean squared error at each node. Additionally, the max depth parameter was optimized by testing a range of different depths to choose the optimal one while preventing overfitting.

Performance evaluation: The calculated R^2 score for the Decision Tree is 0.859, indicating a very good fit of the data. They are very efficient models with minimal training time, representing the lowest amongst all models we explored. However, Decision Trees are prone to overfitting, especially with deeper trees, and may struggle with capturing complex relationships in the data.

Random Forests

Model Description: Random Forests are ensemble learning methods that construct a multitude of uncorrelated decision trees during training. In regression tasks, Random Forests average the predictions of multiple decision trees to improve accuracy and reduce overfitting. This ensemble approach makes Random Forests robust and effective for a problem such as predicting used car prices.

Parameter selection: For the Random Forests model, parameters were carefully selected to optimize performance while managing computational complexity. The number of estimators, representing the number of decision trees in the forest, was set to 60 after experimentation with values ranging from 20 to 100 in increments of 20. This choice was based on achieving almost identical results to higher values while reducing training time. The squared error criterion and maximum depth were kept consistent with the Decision Tree setup to maintain consistency in the comparison.

Performance evaluation: The calculated R^2 score for the Random Forest is an impressive 0.924 indicating an excellent fit of the data. Additionally, Random Forest addresses the high variance problem of decision trees by utilizing parallel decision trees which lead to reduced variance. However, it's also important to note that training time and computational resources for Random Forests are generally much higher compared to other models.

XGBoost

Model Description: XGBoost is an ensemble learning method that utilizes gradient boosting to construct several independent decision trees during training. In regression problems, XGBoost sequentially adds trees to minimize the loss function, resulting in improved accuracy and reduced overfitting. This boosting technique makes XGBoost highly effective for predicting used car prices, as it can capture complex relationships in the data.

Parameter selection: For the XGBoost model, parameters were carefully chosen to optimize performance. Through experimentation, the number of trees was tuned, stabilizing around 100 trees, as observed from a plot of R^2 scores against n_estimators. Additionally, learning rate and max depth were fine-tuned, yielding optimal results for our dataset with a learning rate of 0.2 and a max depth of 20. A higher max depth led to overfitting and a lower R^2 score. These parameter choices were based on achieving the highest performance on our dataset while mitigating the risk of overfitting.

Performance evaluation: The calculated R-squared score for the XGBoost model is an impressive 0.926, outperforming all other models. XGBoost effectively addresses the limitations of individual decision trees, leading to improved predictive accuracy. Moreover, its exceptional speed in both training and prediction, typically taking only around one minute for our dataset, sets it apart from other models. However, XGBoost may be susceptible to overfitting in complex scenarios, requiring careful tuning of hyperparameters and regularization techniques for optimal performance.

Artificial Neural Networks

Model Description: The Artificial Neural Networks algorithm, inspired by the human brain, utilizes multilayer perceptrons to foresee problems and model intricate patterns. It operates through feedforward and backpropagation algorithms, employing gradient descent or stochastic gradient descent for training and weight updating. The model iteratively adjusts weights to minimize the loss function, leading to improved accuracy and reduced overfitting. This flexibility makes Neural Networks suitable for predicting used car prices in a complex scenario similar to what we have.

Parameter selection: For our Neural Networks model, parameters were chosen to optimize performance and reduce training time. The model was configured with two hidden layers of 40 and 20 neurons, using the rectified linear unit activation function, and trained for a maximum of 300 iterations to converge. These choices were based on experimentation and fine-tuning to achieve the highest performance on our dataset while avoiding overfitting.

Performance evaluation: The calculated R^2 score for our Neural Networks model is 0.9, demonstrating its strong performance in capturing complex patterns in the data, leading to improved predictive accuracy compared to simpler models. However, it's important to note that the model is very computationally intensive, especially with larger datasets, and took the most training time out of all the models we explored. Additionally, the model may require extensive hyperparameter tuning to optimize performance. However, the Neural Networks model still offers significant potential for accurately predicting used car prices given our dataset.

Logistic Regression

Model Description:

Logistic Regression is a linear classification algorithm used for binary and multiclass classification tasks. It models the probability that a given input belongs to a particular class by fitting a logistic curve to the data. Despite its name, logistic regression is primarily used for classification rather than regression tasks.

Parameter selection:

Convergence Criteria: This parameter determines the stopping criterion for the optimization algorithm, specifying when to consider the optimization process as converged. It helps control the computational resources needed for training the model. The number of iterations was set to 300.

Performance evaluation:

The Logistic Regression model shows decent performance on the dataset, as indicated by the R-squared score of 0.7763. However, it's important to note that R-squared is not a typical metric for classification tasks.

The classification report provides more detailed insights, including precision, recall, and F1-score for each class, as well as overall accuracy.

The model performs reasonably well across most classes, but there may be room for improvement, especially for classes with lower precision and recall. In summary, while the Logistic Regression model demonstrates overall decent performance, further analysis and potential model tuning could enhance its effectiveness, especially for certain classes.

Naive Bayes

Model Description:

Naive Bayes is a probabilistic classification algorithm based on Bayes' theorem with an assumption of independence between features. Despite its simplicity, it is known to perform well in various real-world scenarios, particularly in text classification and spam filtering tasks.

Parameter selection:

For Gaussian Naive Bayes, there are typically no hyperparameters to tune during model instantiation. It assumes that the features follow a Gaussian distribution, which is already encoded in the algorithm.

Performance evaluation:

The R-squared score, although not typically used for classification tasks, is reported. A negative value (-2.327) indicates severe issues with the model's fit to the data.

Additionally, a classification report is generated, which includes metrics such as precision, recall, and F1-score for each class, as well as overall accuracy. The report provides insights into the model's performance across different classes.

From the classification report, it can be observed that the model has low precision, recall, and F1-score for most classes, indicating poor performance. The overall accuracy is also low at 0.23, suggesting that the model's predictions are not reliable. In summary, the Gaussian Naive Bayes model, as currently implemented, does not perform well on the given dataset. Further analysis, data preprocessing, or model tuning may be necessary to improve its performance.

Analysis of Top Three Performing Models

The top three performing models in our analysis are XGBoost, Random Forests, and Artificial Neural Networks. These models were evaluated based on their speed, accuracy, and underlying factors influencing their performance. Among these models, XGBoost had the highest R^2 score of 0.926. Its efficient implementation of gradient boosting allowed XGBoost to emerge as the fastest model, typically requiring around one minute for training, outperforming Random Forests and Neural Networks both in terms of accuracy and training time. While Random Forests followed closely behind XGBoost in terms of R^2 score, achieving 0.924, they were slightly slower due to constructing multiple decision trees in parallel. Neural Networks also exhibited a high R^2 score of 0.9; however, it was notably slower in training compared to the other two models as a result of the complex architecture and iterative optimization process. The differences in results among the models highlight the impact of factors like model complexity and optimization techniques. XGBoost's efficient gradient boosting, Random Forests' ensemble learning, and Neural Networks' pattern-capturing capabilities all contributed significantly to their respective performance

Final Choice

After careful consideration of performance, model characteristics, and suitability for our problem domain, XGBoost emerges as the preferred choice among the top three models. Its outstanding R^2 score and extremely fast training time make it highly appealing for predicting used car prices. While XGBoost's rapid training speed is a significant advantage, it's essential to acknowledge the potential risk of overfitting in more complex scenarios. However, I am aware of this limitation and intend to address it through careful hyperparameter tuning and regularization techniques. Furthermore, XGBoost's adaptability and robustness make it well-suited for handling complex datasets and capturing patterns in the used car market. Hence, XGBoost is the most suitable model for our machine learning project on predicting used car prices and is our final choice.