US USED CARS DATASET

Authors: Anusha Valasapalli, Naga Sai Lohitha Karmuru, Rishabh Sureshkumar Shah, Wethanie Law
Department of Information Systems, California State University Los Angeles
CIS 5560: Intro to Big Data Science

Abstract: The dataset "Used Car Prediction" contains statistics on used automobiles sold in the United States. This dataset predicts the selling price of a used car based on factors such as manufacturer, model, year, and mileage. It can also help in identifying the important elements that influence used car prices and determining the popularity of various automobile models in the secondhand market. The dataset can be used to train machine learning models that predict the prices of secondhand cars based on their qualities. We intend to estimate car prices and advise consumers on whether they are getting a fair deal using various criteria utilizing regression models and machine learning technologies. We will evaluate the price, mileage, owner count, production year, and other factors to estimate better prices and more accurate results.

1. Introduction

The US used car market is a multi-billion-dollar industry with more than 10 million vehicles sold annually [1]. With the growth of e-commerce and online marketplaces, buyers can access a vast selection of used cars from all over the country. However, ascertaining a vehicle's genuine value might be difficult, given the variety of choices available. In this term paper, we will explore a dataset from Kaggle that contains information on thousands of used cars sold in the US. The dataset includes several pieces of information, including the car's make and model, production year, price, and mileage. We will use different machine learning methods to glean insights, including the Linear regressor, Gradient Boost Tree Trees (GBT) regressor, Random Forest (RF) regressor, Factorization Machine Regressor, and Logistic regression.

We can create predictive models using these algorithms that can calculate the worth of a used car based on its features. We will start by downloading the dataset from Kaggle and then upload the data to Databricks. Later we begin cleaning the dataset using Panda's framework. Data will then be preprocessed and divided into training and testing sets. The five machine learning algorithms will then be trained and tuned with cross-validation and train validation split, and their performance will be assessed using metrics like Root Mean Square Error (RMSE) and R-squared. The best algorithm for estimating the value of a used car will be determined after comparing the results of the five algorithms. Overall, this term paper aims to show how machine learning algorithms effectively estimate the price of used cars and help suggest to the customer whether they are getting a fair deal.

2. Related Work

(1) "Forecasting Used Car Prices Using Time Series Analysis" by M. Al-Tarawneh and A. Al-Madi (2021): This study uses a dataset of used car sales in the USA to forecast prices using time series analysis. This paper differs from ours because we have used 5 Machine learning algorithms to predict the price of used cars in the USA.

(2) Used Car Price Prediction using Linear Regression." by S. B. Kadam and N. K. Choudhari (2021): This study uses a dataset of used car sales in the USA to predict prices using linear regression. This paper is not similar to ours because we used four more algorithms, such as GBT, Decision tree classifier, and Factorization Machine regressor, to predict the prices based on various factors.

(3) "Prediction of Used Car Prices Using Machine Learning Algorithms" by H. Al-Majali and F. Al-Kabi (2020): This study uses a dataset of used car sales in the USA to predict prices using various machine learning algorithms. This paper is quite similar to ours, but they focused on only a few factors affecting the used car's price. Still, we considered mileage, owners count, car's make and model, year of production, and various other factors in the paper.

3. Specification

The dataset comprises information regarding the US used cars. The dataset is of size 2.0 GB.

Table 1 Databricks Specification

Number of nodes	1
File format	CSV
Databricks Community Version	9.1 LTS (includes Apache Spark 3.1.2, Scala 2.12)

Table 2 Spark submit (HDFS) Specifications

Number of nodes	3
CPU cores	4
Storage	390.71GB
Hadoop Cluster Version	3.2.1
Python Version	3.10.4
PySpark Version	3.0

4. Background/Existing work

In our project, we have used 4 regression algorithms for Car Price Prediction and 1 Logistic regression with Classification to suggest to the customer whether they are getting a fair deal or not.

4.1 Regression

Regression helps to predict the continuous values using a collection of input variables. A regression algorithm is also called as supervised algorithm. This type of algorithm requires input values to predict the output values. In our project, we have used 5 algorithms, of which 4 are regression algorithms. The algorithms are Linear regression, Gradient Boost Tree regression, Random Forest regressor, and factorization machine regressor. With these algorithms, we used input variables to predict the used cars' prices in the USA. To choose the input variables, we passed various columns from the dataset to the Random Forest regressor to predict the feature importance. Finally, we chose the Year, City fuel economy, Mileage, Owner count, Highway fuel economy, Engine displacement, and Savings amount columns to work with 4 regression algorithms. To find the best algorithm, we considered RMSE and R2 metrics. If R2 is more and RMSE is less, we can say that is the best algorithm for this dataset.

4.2 Classification

Classification is a supervised machine learning algorithm. Here we can only predict categorical values (TRUE/FALSE, YES/NO). Using the regression algorithms, we predicted the price of the used cars; later, we wanted to suggest whether the customer was getting a fair deal through the Logistic regression classification algorithm. We have predicted the evaluation metrics: TP, FP, TN, FN, Precision, Recall, and AUC. We can say it's a good deal when the TT count is more, Precision/Recall/AUC is more than 0.8.

5. Implementation Flowchart

US Used car dataset is downloaded from the official website of Kaggle. The entire implementation process is shown in the flow chart below (Figure 1). The raw data set, which has 9.98 GB, was downloaded from Kaggle to our local system in CSV format, sliced 33Mb of data, and uploaded to Databricks. Later the dataset is spilt to Train and Test the data.

In our project, we used 70% of the data for training and 30% for the test data set. The trained data is further used to build a model. On the Train data set, the model is pre-built. We run the model using the Test Data set after it has been fitted. We have used five different algorithms Linear regressor, Gradient Boost Tree (GBT) regressor, Random Forest (RF) regressor, Factorization Machine Regressor classifier to predict the price of the used cars in the USA, and Logistic Regression classifier to suggest the customer whether they are getting a fair deal or not. Later we compared the results of regression (RMSE, R2) to determine the best algorithm for estimating the price of the used car and classification results (AUC, Recall, Precision) which helps in suggesting to the customer whether they are getting a fair deal or not.

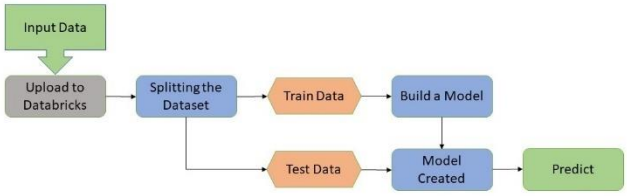

Figure 1 - Implementation Flowchart

6. Machine Learning Models

6.1 Linear Regression Model:

Initially, we started with Linear Regression. We split the data into 70% Train and 30% testing. To tune the data, we used Train Validation and Cross-Validation split. We created a Linear regression and cross-validator. We trained the data with the cross-validator, linear regression, and a few other param grid builder parameters, and then we evaluated the data with a regression evaluator.

Figure-2 shows the data bricks result using cross-validation. We got R2 as 0.61 and RMSE as 7693.73. The run time for the Cross validator is 58.50 seconds.

```
print("r2 For LR_CV: %.2f" % lr_evaluator_CV_r2.evaluate(prediction_CV_lR))
print("RMSE For LR_CV: %.2f" % lr_evaluator_CV_rmse.evaluate(prediction_CV_lR))
```

```
▸ (2) Spark Jobs
r2 For LR_CV: 0.61
RMSE For LR_CV: 7693.73
```

Figure 2 – Databrick result for Cross validator

Figure-3 shows the Spark submit the result of the Linear regression using Cross validator. We got R2 as 0.63 and RMSE as 6790.33. When compared to data bricks results, spark-submit results are better as it has more R2 and less RMSE.

```
r2 For LR_CV: 0.63
RMSE For LR_CV: 6790.33
```

Figure 3 – Spark submit the result for Cross Validator.

Figure-4 shows the data bricks result using Train Validator. We got an R2 value of 0.61 and an RMSE value of 7693.73. The run time for the train validator is 20.46 seconds. Even though the results are the same for both cross and train validators, the run time for the Cross validator is more than the train validator.

```
print("r2 For LR CV: %.2f" % lr_tv_evaluator_r2.evaluate(prediction_tv_lr))
print("RMSE for LT TV: %.2f" % lr_tv_evaluator_rmse.evaluate(prediction_tv_lr))
```

```
▸ (2) Spark Jobs
r2 For LR CV: 0.61
RMSE for LT TV: 7693.73
```

Figure 4 – Databrick result for Train Validator

Figure-5 shows that Spark submit the result of the Linear regression using the Train validator. We got R2 as 0.63 and RMSE as 6790.33. The results are the same for both train validation and cross-validation. When compared to data bricks results, spark-submit results are better as it has more R2 and less RMSE.

```
r2 For LR CV: 0.63
RMSE for LT TV: 6790.33
```

Figure 5 – Spark submit the result for Train Validator

6.2 Gradient Boost Tree Regression Model:

The second model that we worked on is the Gradient Boost Tree Regression. In this model, we trained, tuned, and evaluated the data with a regression evaluator for GBT using cross validator and train-validator.

Figure-6 shows the data bricks result using cross-validation. We got R2 as 0.67 and RMSE as 7059.37. The run time for the Cross validator is 12.81 minutes.

```
gbt_evaluator_r2 = RegressionEvaluator(labelCol="label", predictionCol="prediction", metricName="rmse")
print("RMSE_GBT_CV: %.2f" % gbt_evaluator_r2.evaluate(prediction_GBT))

gbt_evaluator_rmse = RegressionEvaluator(predictionCol="prediction", \
                labelCol="label",metricName="r2")
print("R2_GBT_CV: %.2f" % gbt_evaluator_rmse.evaluate(prediction_GBT))
```

```
▸ (2) Spark Jobs
RMSE_GBT_CV: 7059.37
R2_GBT_CV: 0.67
```

Figure 6 – Databricks result for Cross Validator

Figure-7 shows that Spark submit the Gradient Boost Tree regression result using Cross validator. We got R2 as 0.77 and RMSE as 5427.82. When compared to data bricks results, spark-submit results are better as it has more R2 and less RMSE.

```
RMSE_GBT_CV: 5427.82
R2_GBT_CV: 0.77
```

Figure 7 – Spark submit the result for Cross Validator

Figure-8 shows the data bricks result using Train Validator. We got an R2 value of 0.67 and the RMSE value of 7009.82. The run time for the train validator is 53.94 minutes. Even though the results are almost similar for both cross and train validators, the run time for the Train validator is more than the Cross validator.

```
gbt_evaluator_rmse = RegressionEvaluator(labelCol="label", predictionCol="prediction", metricName="rmse")
print("RMSE_GBT_TV: %.2f" % gbt_evaluator_rmse.evaluate(prediction_GBT))


gbt_evaluator_r2 = RegressionEvaluator(predictionCol="prediction", \
                 labelCol="label",metricName="r2")
print("R2_GBT_TV: %.2f" % gbt_evaluator_r2.evaluate(prediction_GBT))

```

▸ (2) Spark Jobs

```
RMSE_GBT_TV: 7009.82
R2_GBT_TV: 0.67
```

Figure 8 – Databricks result for Train Validator

Figure-9 shows that Spark submit the Gradient boost Tree regression result using the Train validator. We got R2 as 0.80 and RMSE as 5041.78. When compared to data bricks results, spark-submit results are better as it has more R2 and less RMSE.

```
RMSE_GBT_TV: 5041.78
R2_GBT_TV: 0.80
```

Figure 9 – Spark submit the result for Train Validator

6.3 Random Forest Regression Model:

In Random Forest, we worked with different input columns from our dataset; we trained and tuned the data and evaluated the data with the regression model. We used parameters like maximum depth and the number of trees to tune the data to increase the predictive performance. We have performed tuning with cross validator and train validator and without tuning. Through this process, we understood that Random Forest Regression is an important model which allowed us to predict the important columns or features in our dataset to predict the price of the used cars.
Figure-10 shows the data bricks result using cross-validation. We got R2 as 0.74 and RMSE as 6197.13. The run time for the Cross validator is 32.08 minutes.

```
rf_evaluator_CV_rmse = RegressionEvaluator(labelCol="label", predictionCol="prediction", metricName="rmse")
print("RMSE for CV_RF: %.2f" % rf_evaluator_CV_rmse.evaluate(prediction_CV_rf))


rf_evaluator_CV_r2 = RegressionEvaluator(predictionCol="prediction", \
                 labelCol="label",metricName="r2")
print("R2 for CV_RF: %.2f" % rf_evaluator_CV_r2.evaluate(prediction_CV_rf))
```

▸ (2) Spark Jobs

```
RMSE for CV_RF: 6197.13
R2 for CV_RF: 0.74
```

Figure 10 – Databrick result for Cross validator

Figure-11 shows that Spark submit the result of the Random Forest regression using Cross validator. We got R2 as 0.87 and RMSE as 4115.92. When compared to data bricks results, spark-submit results are better as it has more R2 and less RMSE.

```
RMSE for CV_RF: 4115.92
R2 for CV_RF: 0.87
```

Figure 11 – Spark submit the result for Cross Validator

Figure-12 shows the data bricks result using Train Validator. We got an R2 value of 0.73 and an RMSE value of 6758.30. The run time for the train validator is 35.74 minutes.

```
rf_tv_evaluator_rmse = RegressionEvaluator(labelCol="label", predictionCol="prediction", metricName="rmse")
print("RMSE for RF_TV: %.2f" % rf_tv_evaluator_rmse.evaluate(prediction_tv_rf))


rf_tv_evaluator_r2 = RegressionEvaluator(predictionCol="prediction", \
                 labelCol="label",metricName="r2")
print("R2 for RF_TV: %.2f" % rf_tv_evaluator_r2.evaluate(prediction_tv_rf))
```

▸ (2) Spark Jobs

```
RMSE for RF_TV: 6758.30
R2 for RF_TV: 0.73
```

Figure 12 – Databrick result for Train Validator

Figure-13 shows that Spark submit the result of the Random Forest regression using the Train validator. We got R2 as 0.86 and RMSE as 4123.82. The results are almost the same for both train validation and cross-validation. When compared to data bricks results, spark-submit results are better as it has more R2 and less RMSE.

```
RMSE for RF_TV: 4123.82
R2 for RF_TV: 0.86
```

Figure 13 – Spark submit the result for Train Validator

After we worked with most of the columns in our dataset, we figured out the columns in figure-14 are important and show the feature importance. So, in all the columns, Mileage has the first importance, and Year has the second importance. Higher fuel economy and engine displacement have third and fourth importance compared to other columns/features.

Figure 14 – Feature importance

	feature	importance
3	owner_count	0.044117
6	savings_amount	0.065726
1	city_fuel_economy	0.092745
4	highway_fuel_economy	0.118275
5	engine_displacement	0.145099
0	year	0.230173
2	mileage	0.303864

6.4 Factorization Machine Regression Model:

The last regression model that we worked on is the Factorization Machine Regression. In this model, we trained, tuned, we evaluated the data with a regression evaluator for Factorization Machine Regressor using cross validator and train validator.
Figure-15 shows the data bricks result using cross-validation. We got R2 as -48.92 and RMSE as 95536.98. The run time for the Cross validator is 3.88 minutes.

```
fm_cv_evaluator_rmse = RegressionEvaluator(labelCol="label", predictionCol="prediction", metricName="rmse")
print("RMSE for FM_CV: %.2f" % fm_cv_evaluator_rmse.evaluate(prediction_fm))

fm_cv_evaluator_r2 = RegressionEvaluator(predictionCol="prediction", \
                 labelCol="label",metricName="r2")
print("R2 for FM_CV: %.2f" % fm_cv_evaluator_r2.evaluate(prediction_fm))

```

▸ (2) Spark Jobs

```
RMSE for FM_CV: 95536.98
R2 for FM_CV: -48.92
```

Figure 15 – Databrick result for Cross validator

Figure-16 shows the Spark submit the result of the Factorization Machine regression using Cross validator. We got R2 as -103.69 and RMSE as 114680.51.

```
RMSE for FM_CV: 114680.51
R2 for FM_CV: -103.69
```

Figure 16 – Spark submit the result for Cross Validator

Figure-17 shows the data bricks result using Train Validator. We got the R2 value as -1.15 and the RMSE value as 19841.19. The run time for the train validator is 2.63 minutes.

```
fm_tv_evaluator_rmse = RegressionEvaluator(labelCol="label", predictionCol="prediction", metricName="rmse")
print("RMSE for FM_TV: %.2f" % fm_tv_evaluator_rmse.evaluate(prediction_tv_fm))

fm_tv_evaluator_r2 = RegressionEvaluator(predictionCol="prediction", \
                labelCol="label",metricName="r2")
print("R2 for FM_CV: %.2f" % fm_tv_evaluator_r2.evaluate(prediction_tv_fm))
```

▸ (2) Spark Jobs

RMSE for FM_TV: 19841.19
R2 for FM_CV: -1.15

Figure 17 – Databrick result for Train Validator

Figure-18 shows that Spark submit the result of the Linear regression using the Train validator. We got R2 as -1.48 and RMSE as 17640.81. This model isn't suitable for our dataset as it has inappropriate values of R2 and RMSE.

```
RMSE for FM_TV: 17640.81
R2 for FM_CV: -1.48
```

Figure 18 – Spark submit the result for Train Validator

6.5 Logistic Regression Classification Model:

In logistic regression, we have added some other input columns like daysonmarket, dealerzip, listing id, seller rating, and sp id to predict whether the user can buy this car or not. We worked with the vector assembler, vector indexer, and min-max features and passed them to parameters for logistic regression. We used regParam, elasticNetParam, and maxIter to minimize the loss for better prediction, and we also used binary classification to evaluate the data. Did the same process with the cross-validation and train-validation split to tune the data.

Figure-19 shows the metric table. The precision value is 0.94, and the Recall value is 0.99. TP means the true predicted result with the highest count, so the prediction result is accurate and suggestable for the customer to buy the car. The run time for the cross-validator is 2 minutes.

metric	value
TP	417.0
FP	25.0
TN	68.0
FN	3.0
Precision	0.9434389140271493
Recall	0.9928571428571429

Figure 19 – Databrick result for Cross validator

Figure-20 shows the AUC value of 0.97, which is a good result as it is almost near 1.

```
AUC =  0.9795442908346135
```

Figure 20 – Databrick result for Cross validator

Figure-21 shows that Spark submit the result of the metric table. The precision is 0.81, and Recall is 1.00. When compared to data bricks, the results in Spark submit are more accurate and better. Here the TP value is high, so the prediction result is accurate and suggestable for the customer to buy the car.

metric	value
TP	39160.0
FP	9059.0
TN	0.0
FN	0.0
Precision	0.8121279993363612
Recall	1.0

Figure 21 – Spark submit the result for Cross Validator

Figure-22 shows the AUC value of 0.95, which is a good result as it is almost near 1. But the AUC value of data bricks is more when compared to the Spark submit result.

```
AUC =  0.9546786242181969
```

Figure 22 – Spark submit the result for Cross Validator

Figure-22 shows the metric table. The precision value is 0.93, and the Recall value is 0.98. TP means the true predicted result with the highest count, so the prediction result is accurate and suggestable for the customer to buy the car. The run time for the train validator is 4 minutes.

metric	value
TP	415.0
FP	28.0
TN	65.0
FN	5.0
Precision	0.9367945823927766
Recall	0.9880952380952381

Figure 22 – Databrick result for Train Validator

Figure-23 shows the AUC value of 0.98, which is a good result as it is almost near 1. The AUC result of the train validator is more accurate than the cross-validator.

```
AUC =  0.98
```

Figure 23 – Databricks result for Train Validator

Figure-24 shows that Spark submit the result of the metric table. The precision is 0.81, and Recall is 1.00. When compared to data bricks, the results in Spark submit are more accurate and better. Here the TP value is high, so the prediction result is accurate and suggestable for the customer to buy the car. The cross-validator and train validator results are the same in Spark submit.

metric	value
TP	39160.0
FP	9059.0
TN	0.0
FN	0.0
Precision	0.8121279993363612
Recall	1.0

Figure 24 – Spark submit the result for Train Validator

Figure-25 shows the AUC value of 0.95, which is a good result as it is almost near 1. But the AUC value of data bricks is more when compared to the Spark submit result.

```
AUC =  0.95
```

Figure 25 – Spark submit the result for Train Validator

7. Conclusion

After working with all the algorithms in Spark submit to predict the price of the used cars in the USA, we compared the RMSE and R2 results and decided that the Random Forest regressor is the best algorithm for our dataset. When we worked with the Random Forest regressor, we got the R2 value as 0.8 and RMSE as 4115. The second-best algorithm to predict the used car price is the Gradient Boost Tree Regression (GBT); we got the R2 value as 0.8 and RMSE as 5041. For the Linear Regression (LR), we got the R2 value as 0.6 and RMSE as 6790. The Factorization Machine Regressor model is the only model not a suggestible algorithm for our dataset. The Logistic regressor model is helpful with the recall, precision, and AUC values being accurate, allowing the customer to decide if they are getting a fair deal. Compared to the data bricks results, the Spark submit results are more precise, the R2 values are almost near 1, and the RMSE values are also low for all the algorithms.

Based on Spark Submit results, we arranged various Regression Algorithms below.

Based on accuracy:

RF> LGR> GBT> LR> FM (RF having the best accuracy and FM having the least)

Based on time:

GBT> RF> LR> FM> LGR (GBT taking the most time and Logistic Regression taking the least)

8. References

[1] https://www.cnbc.com/2023/01/06/2022-us-auto-sales-are-worst-in-more-than-a-decade-.html#:~:text=Industry%20estimates%20range%20from%2013.7,recovering%20from%20the%20Great%20Recession

[2] https://spark.apache.org/docs/2.2.0/ml-pipeline.html

[3] https://spark.apache.org/docs/latest/ml-tuning.html

[4] Dataset source- https://www.kaggle.com/datasets/ananaymital/us-used-cars-dataset

9. GitHub Link

https://github.com/anushavalasapalli-97/CIS-5560-US_USED_CARS_PROJECT-