Vehicle Sales Time

Abstract

In the 21st century where almost every transaction is held online, e-commerce is also becoming a great platform for car trade. There are great market places to buy and sell used cars online. However, one constraint that seems to have a great effect on car sellers is the amount of time taken to get the car off their hands. Although there is no set time that a car should spend online, the amount of time spent to sell a car can be predicted through the type of vehicle and its estimated selling price. Therefore, this paper will discuss the data, the different unsupervised learning techniques, and steps as well as accuracies in the process of creating a model that can predict the time it would take a specific car to be sold once posted online. The research will be constrained to the online trading conditions and events of German vehicles, and will not be ensured to be as efficient on other vehicles.

Introduction

With the recent development of the Internet, Internet-of-Things, communication and other infrastructures that facilitate the process of e-commerce, we are able to sell and purchase commodities that satisfy our interests with ease. Likewise, the trade of vehicles online is also being somewhat customary. One of the main factors that make some vendors and delivery companies better than others is the amount of time they take to complete the purchase and hand the items to their customers. Therefore, being able to accurately predict the amount of time an item would take to be sold can be very important, both for the customers and the vendors.

The amount of time a vehicle could take to be purchased, however, can be predicted based on the type of car and its features. Fortunately, the posted prices of cars online have a great effect on the time it has to spend online (More on this below). Other features such as the type of gas a car uses and its brand also play a huge role.

Therefore, taking these features into count, one can predict the amount of time it would take a specific vehicle to be purchased.

Structure of the Data at Hand

The research depended on a web scrape containing over 370,000 German used vehicles scraped with Scrapy from Ebay-Kleinanzeigen. Initially, the dataset contains about 20 features including the name or model of the vehicle, seller, type of offer, number of pictures of the car posted and price that are fairly sales-related and more features concerning details and specifications of the vehicles. Some of those features include the type of transmission, the type of fuel the vehicles take, the brand, the distance on their odometers, and the type of vehicles. The data set also contains the date at which the vehicles were posted online for sale and the date at which they were removed from the websites.

Some of the fields in the data set will not be used in the training or testing stages of the model as they need heavy natural language processing. The data set is created from different websites and companies, therefore some features like the names of the vehicles are not regulated and do not have specific patterns to identify the vehicles through them. Fortunately, these fields can be replaced by other fields such as the models, brands and year of registrations of the vehicles.

Cleaning the Data

The most important changes to the dataset that were initially made were changes made on the different features. Some features were fairly modified so they can fit in the model training stage, some were modified to get further calculated fields, and others have been completely removed from the dataset. The motive of this research is to be able to predict the amount of time vehicles might take to be sold once posted online. Since the research will be dependent upon supervised learning we need a label for each record

that states the number of days the vehicle spent online. However, there isn't a field that specifically had such labels, so the first step was to create that field.

Fortunately, there are two fields that can be used to produce the desired outcome. The "lastSeen" column and the "dateCreated" columns, along with some string manipulation and calculations, can create the "daysOnline" field that contains the number of days the vehicle has spent online. Both fields are in a "yyyy-mm-dd hh:mm:ss" format, which can easily be manipulated to attain the date in years, months and days numerically for further calculations. Once the year, month and date are excerpted from the fields, one can use the following method to easily create a vector containing the number of days spent online:

```
def dateValue(lst):
    values = []
    for i in lst:
        string = i
        string = string.split(" ")[0]
        string = string.split("-")
        year = int(string[0])*365
        month = int(string[1])*30
        day = int(string[2])
        dateValueString = year+month+day
        values.append(dateValueString)
    return values
```

Later, the daysOnline field was created and the two other fields were removed as they were of no significance from this time on.

```
dateCreated = dateValue(data.dateCreated)
lastSeen = dateValue(data.lastSeen)
data = data.drop(columns="lastSeen")
data = data.drop(columns = "dateCreated")
data.insert(17, "lastSeen", lastSeen)
data.insert(17, "dateCreated", dateCreated)
```

This process could have been done more efficiently through libraries such as “Lubridate”, but it has not been applied in this research due to tight time constraints. Once, the labels that are to be used for supervised learning later have been created, it was time to focus on the cleanness of the data and the necessity of every other field included in the data.

One of the major problems of the dataset was that part of it was in German and had to be translated into English to have a better understanding of its structure and meaning. Fields like “seller”, “OfferType”, “gearbox”, and “notRepairedDamage” had to be translated. The following function was used in R to translate the German terms field by field.

```
# Translations from German to English.
# translations of the German words.
#   Angebot   -   Offer
#   Gesuch    -   Application
#   ja        -   yes
#   nein      -   no
#   manuell   -   manual
#   automatik -   automatic
# A function that translates the German terms in the data to English.
translate = function(x){
  if(x == "Angebot"){
    return("offer")
  }
  else if(x == "Gesuch"){
    return("application")
  }
  else if(x == "ja"){
    return("yes")
  }
  else if(x == "nein"){
    return("no")
  }
  else if(x == "automatik"){
    return("automatic")
  }
  else if(x == "manuell"){
    return("manual")
  }
  else if(x == "gewerblich"){
```

```
    return("commercial")
  }
  else if(x == "privat"){
    return("private")
  }
  else{
    return(x)
  }
}
```

With the use of methods such as lapply(), the function above can be used to translate every element in a vector one by one. As it can be seen in the thesaurus, most of the German words were fairly similar to their English translations, which made the process much easier. For the words that do not resemble any English terms, google translator had been used to translate them into English.

Later, some fields that were not important to the predictions (fields that could throw the models off) were removed. The “name” column was the first to be removed as it had too many factors and was not essential to our predictions as well. The “name” field in the data was the names given by the people that posted the car’s specifications online, and therefore were inconsistent and did not follow any pattern that could easily be identified without a massive amount of natural language processing. In addition, all the necessary information in the column could be found in other fields such as the “brand”, so there was no need to keep it there. Another field that was removed was the “dateCrawled” field. This field represented the date at which the specific record was found during the collection of the data and should not have any effect on the predictions.

So as to fit my models, especially the regression ones, I had to factorize most of the columns. This enabled my data to be more numerical than categorical so as to train the regression models.

Looking at the “nrOfPictures” column, it seems like the column only consists of 0’s. Which means that none of the records really had pictures attached with them in the Ads.

Therefore, there was no need to keep the column in the data as well, so it was removed.

Another problem that was in the data was the presence of outliers. There seem to be cars that took more than 700 days to sell, which I believe should not be considered while training the models. In fact, there are only a few cars that took more than a year to sell, and most of them might have been cars that sellers forgot to take down. Therefore, I decided to remove those records.

Another outlier that needed to be taken care of was based on the price. Some cars were overpriced to the extent that it looked like an error in the web scraping or when the data was uploaded by the sellers. Some cars were priced for over 99 billion dollars, which obviously cannot be the price of a car. Therefore, as it seemed reasonable to get rid of those cars, I set up a cut point to 200,000 dollars on the price and removed every record that had a greater price.

Visualization and Analysis

Getting to the visualization and analysis part of the research, I tried to see the effects of some of these fields on the number of days a vehicle spends online before being purchased. Initially, I tried to see how many days cars take to sell throughout the data and see the distribution of the field. Figure 1 below shows a plot of the number of records in the y-axis versus the number of days spent in the x-axis. It can be easily seen that most cars are sold between the first and tenth day the vehicles are posted online. The plot then eventually decreases showing that not a lot of cars take more than 30 days to sell.

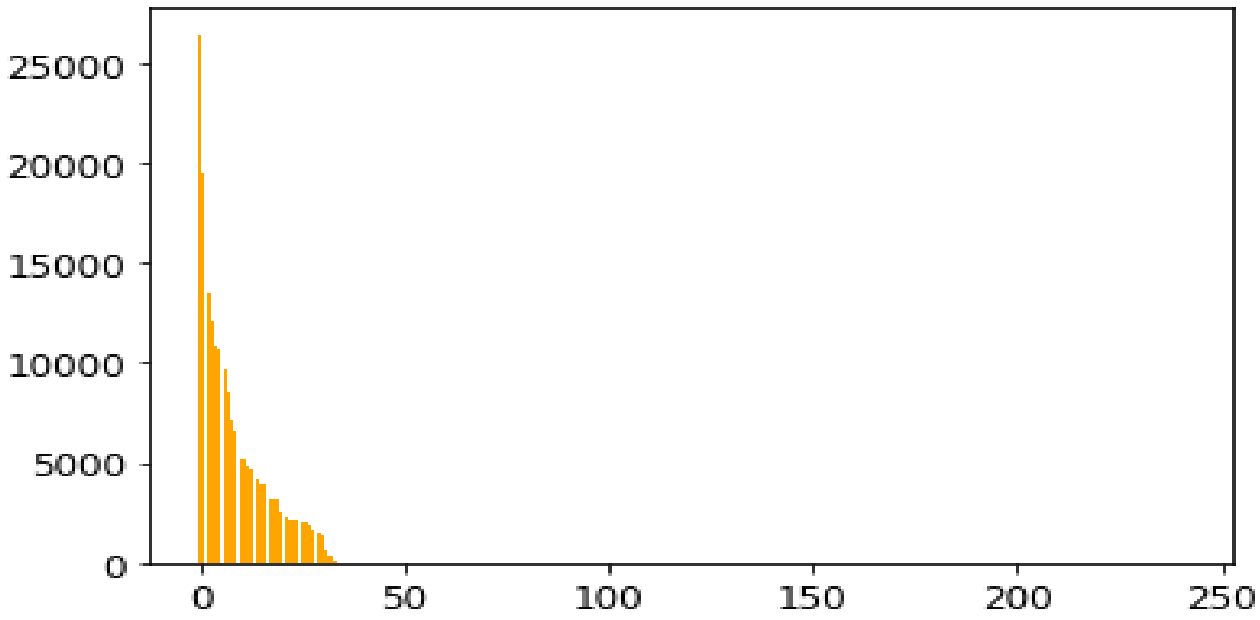

Figure 1: Number of records Vs. The number of days online.

Considering the fields that may have more effects on the number of days online, naturally, the type of seller should affect the number of days the car spends online because commercial sellers tend to have better networks and are more trusted to purchase a car from. Thus, cars sold by commercial sellers should spend much less time online. However, the records with private sellers had an average number of days online of 8.38 whilst the records with commercial sellers had an average number of days of 17.0. Although this did not make a lot of sense, the difference should be enough evidence to know that the values of this field matter, and should be considered in the training of the models later on.

Another field I focused on was the fuel type of the records. As it turns out, the records that take benzene have an average number of online days of about 8.4 whilst the records with diesel have an average of 8.5 days online. This can also easily be seen in the two figures below. The two plots seem almost exactly the same.

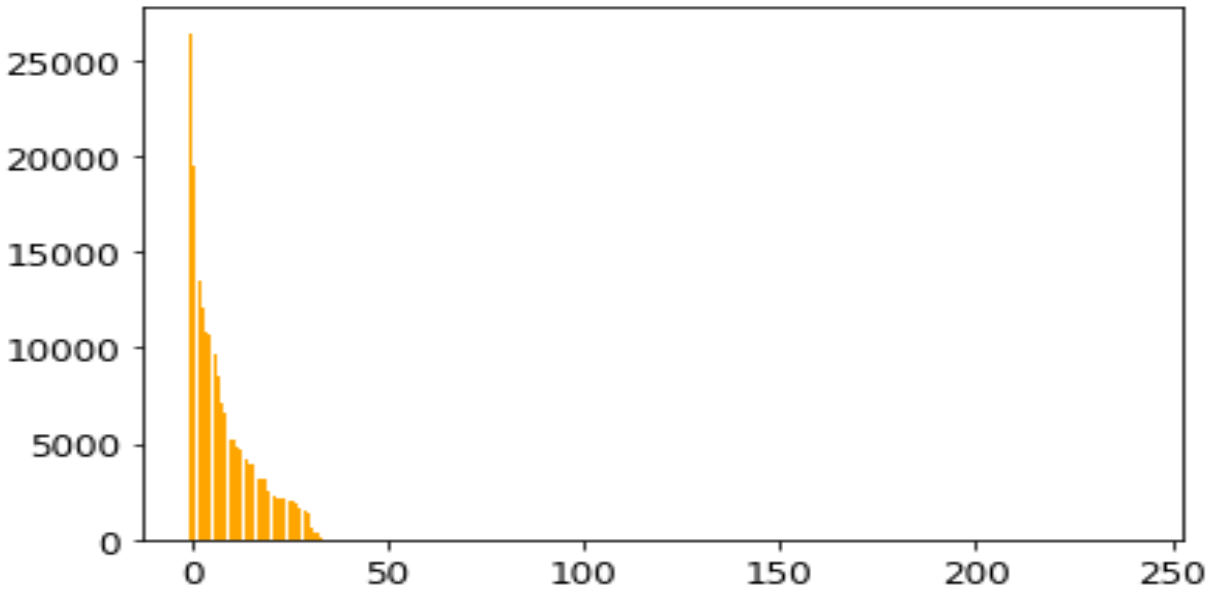

Figure 2: number of cars that take benzene Vs. the number of days spent online

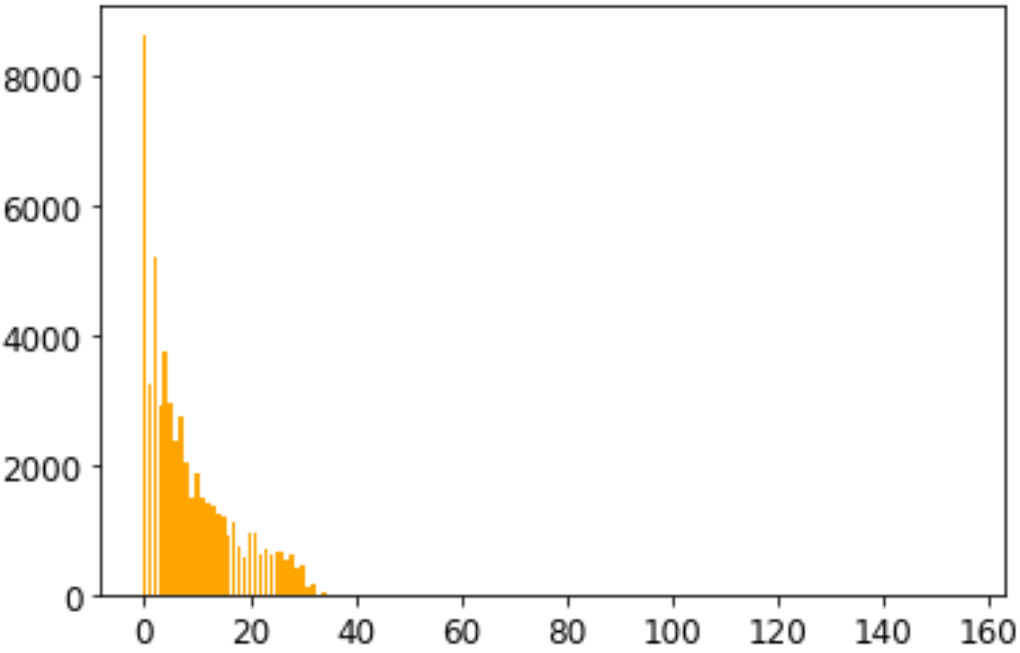

Figure 3: number of cars that take diesel Vs. the number of days spent online

Another main factor that should affect the number of days spent online is the price at which the cars were posted to be sold. Figure 4 below shows a scatter plot of the prices of each record in relation to the number of days spent online. (y-axis being the prices of

the cars and x-axis is the number of days spent online). From the figure, it can be inferred that cheaper cars tend to be sold earlier, which makes sense. Therefore, there must be a correlation between the two fields.

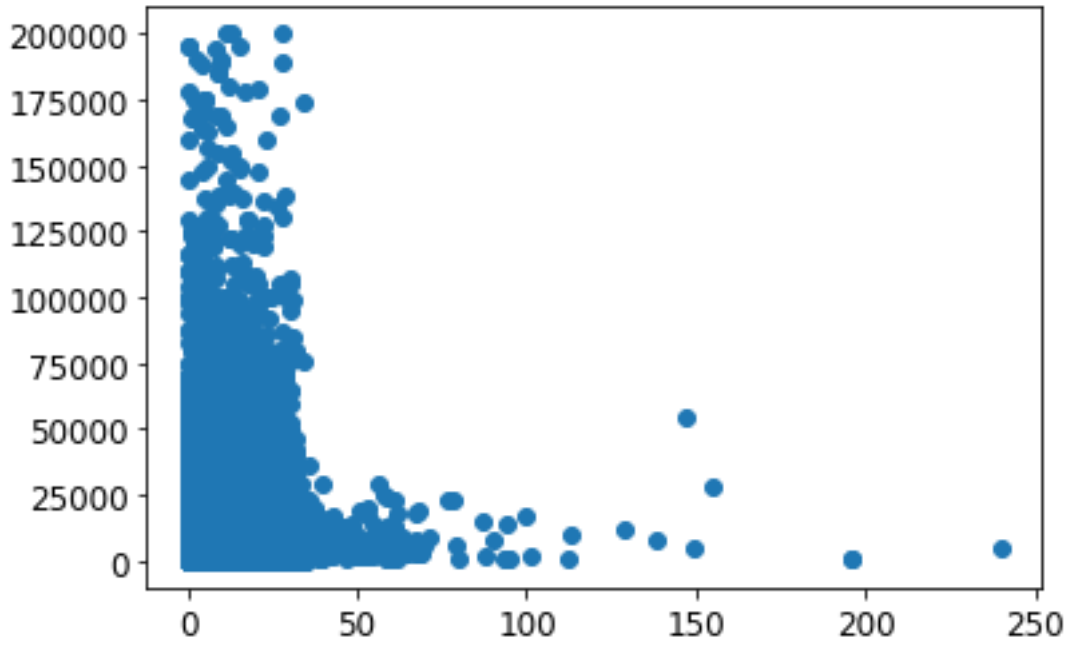

Figure 4: prices of cars Vs the number of days online.

To create the models and train them, I cut my data into a training set, a validation set and a test set. The data was cut with a proportion of 80:10:10 for training, validation and testing. Since I am trying to predict numerical values, the most logical models to use were regression type models, and maybe trees. Therefore, I fit a random forest regressor, an extra trees regressor and an XGBoost regressor. After successful training with the training set, I ran the models individually on the validation set and tried to see the RMSEs and cross-validations. It made more sense to think of the dispersion of my prediction away from my read values (the labels), and RMSE could in fact tell how far off the predictions were, rather than just declaring if the predictions were right or wrong.

Following that, I imported cross_val_score and ran it with my models on the validation set with 10 folds, then took the square root of the individual scores. Random forest

regressor had a mean score of 8.1 with a standard deviation of 0.01, Extra trees regressor resulted in 8.3 with a standard deviation of 0.1 and XGBoost resulted in an 8.1 with a standard deviation of 0.15. This marked the random forest regressor as the best model in this case, but, since all the scores were fairly similar, I decided to see if they improve when put in an ensemble. Although I put the random forest regressor and the Extra trees regressor in an ensemble, the score went up to 8.3 with a standard deviation of 0.11.

Choosing the random forest regressor, I decided to make a grid search to see what parameters would give out the best predictions. I used the grid search and its parameters as displayed below.

```
param_grid = [
 {'n_estimators': [3, 10, 30], 'max_features': [2, 4, 6, 8]},
 {'bootstrap': [False], 'n_estimators': [3, 10], 'max_features': [2, 3,
4]},
 ]
grid_search = GridSearchCV(random_forest_reg, param_grid, cv=5,
 scoring='neg_mean_squared_error',
return_train_score=True)
```

From the grid search, I found out that the best combinations of parameters was when ‘max_features’ was set to 4 and ‘n_estimators’ was set to 30. Therefore, I set the parameters on the random forest regressor and run it over the test set, which gave a score of about 8.3 and a standard deviation of 0.24, meaning that the predictions were off by about a week for every record. An error space of a week, I believe, is a good enough model to predict the time at which a car would be sold.

References

Leka, Orges. “Used Cars Database.” *Kaggle*, Orges Lega, 13 Nov. 2019, www.kaggle.com/orgesleka/used-cars-database.

Géron Aurélien. Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. OReilly Media, Inc., 2019.