UNIVERSIDAD DE CARABOBO
Facultad Experimental de Ciencias y Tecnología
Dirección de Postgrado FACT

ACTA VEREDICTO DEL TRABAJO ESPECIAL DE GRADO
PROGRAMA: ESPECIALIZACIÓN EN DESARROLLO DE SOFTWARE

Quienes suscribimos, profesores Mirella Herrera C.I.V- 8.044.677, Francisca Grimón C.I.V-5.521.244 y José Canache C.I.V-15.607.108, integrantes del Jurado designado por el Consejo de Postgrado de la Facultad Experimental de Ciencias y Tecnología de la Universidad de Carabobo, en su reunión ordinaria virtual No 03/2021 de fecha 15/06/2021, para conocer y decidir acerca del trabajo especial de grado titulado: "HOW TO BUILD, DEVELOP AND DEPLOY A MACHINE LEARNING MODEL TO PREDICT CARS PRICE USING NEURAL NETWORKS", el cual aparece en la publicación electrónica Google Launchpad - Applied machine learning best practices, con fecha de publicación 31/05/2019, (https://medium.com/thelaunchpad/how-to-build-develop-and-deploy-a-machine-learning-model-to-predict-cars-price-using-neural-7f7439a37300), presentado por el Ingeniero PEDRO RICARDO CASTILLO DELGADO, C.I. V-20.497.561, bajo la tutoría académica de la Profesora Mirella Herrera C.I.-V- 8.044.677, como requisito para optar al título de Especialista en Desarrollo de Software, todo ello conforme a lo estipulado en los artículos 79, 128 y 136 del Reglamento de los Estudios de Postgrado de la Universidad de Carabobo (REPUC), dejamos constancia de lo siguiente:e

1. Es un proyecto de innovación en el dominio del mercado de vehículos usados en 8 países con economías emergentes, usando técnicas de Aprendizaje de Máquinas (*Machine Learning*) para predecir sus precios, tanto para vendedores como para compradores. Las técnicas y modelos empleados, están orientados a reducir el tiempo y mejorar la efectividad en la toma de decisiones.
2. En este artículo se observa la aplicación del cuerpo de conocimientos de la Ingeniería del Software y del ciclo de vida, desde los requisitos hasta las pruebas, de una forma integral y confiable. Todo ello, aplicado a un caso real, que es el objetivo ulterior de un trabajo final en el programa de Especialización en Desarrollo de Software.

3. El uso del Aprendizaje de Máquinas empleado en esta investigación, aborda elementos de las ciencias de la computación, ciencia de datos y estadística; cuyos resultados obtenidos durante la experimentación, evidencian la capacidad alcanzada de predecir hasta en un 50% el precio de los vehículos con un margen de error inferior al 5%, representando así una solución factible y quedando abierta la investigación para trabajos futuros en esta área.

Cumplida la revisión de rigor del trabajo antes mencionado por cada uno de los miembros del Jurado, decidimos por UNANIMIDAD que cumple con los requisitos exigidos para su aprobación, por lo que emitimos el veredicto APROBADO, todo conforme a lo dispuesto en la normativa legal vigente.

En fe de lo cual se levanta y firma la presente Acta a los siete días del mes de julio de 2021

Mirella Herrera
C.I. 8.044.677
Coordinador del Jurado (Tutor)
FACYT- UC

Francisca Grimón
C.I.: 5.521.244
Miembro del Jurado
FACYT - UC

José Canache
C.I.- V- 15.607.108
Miembro del Jurado
FACYT- UC

You have **2** free member-only stories left this month. Sign up for Medium and get an extra one

How to Build, Develop and Deploy a Machine Learning Model to predict cars price using Neural Networks

Pedro Castillo Follow

May 31, 2019 · 9 min read ★

Authors: Pedro Castillo, Alessandro Regonini.

In this post we want to share our experience building a machine learning model that can predict used car prices by using neural networks. We started from scratch with a small team in place but a clear goal in mind: be able to accurately predict the selling price of an inspected used car without any established price guidelines.

We are **Frontier Car Group**, a Berlin based startup focused on the digitalization of the automotive sales sector in frontier markets. Currently, we are running used car online marketplaces in 8 emerging countries. One of our main business models is C2B2B, where we buy used cars from

consumers and after a deep technical inspection sell them for profit to dealerships via an online auction portal (learn more about **FCG**).

The challenge

We operate in countries where there are unique challenges:

- A lack of resources to compile and create price guidelines for vehicles. (e.g. Blue Books).
- Price dynamics that are market specific and counter-intuitive. (e.g. specific car models gaining value over time due to difficult provisioning).
- Price insensitivity to certain vehicle damages in particular markets. (e.g. a car in Nigeria has damage to its body, yet its price is unaffected).
- Small datasets of selling transactions for specific models.

How to price a car in an emerging market… and not die trying

As a data science project, we try to predict the price that dealers are likely to pay for a car based on an inspection report. In terms of the ML scope, this is a regression problem.

During our inspection process we try to be quick, yet thorough. We examine the vehicle for 30 minutes and work to collect a comprehensive amount of features for every car — over 200 to be exact. These competing goals create a unique data set where from one inspection to another you can find an interesting, and at times arbitrary, distribution of missing fields.

The composition of these fields is broad and ranges from technical specifications to tax and legal statuses. As you can expect, we have a significant number of categorical features, such as manufacturer and model. However, we also have some quirks, such as text input fields with a lack of structure to account for local slang and input errors.

Overall, the majority of our features are not numeric. For example, in order for a user to register damages to a car, they create a report by choosing from a bundle of images, all portraying different types of problems, and then select which image best represents the condition of their vehicle.

Yet, even with these features, operating in different markets implies different ways to inspect and value cars. This easily translates into one of the most common challenges in machine learning projects, a lack of standardization.

Start with the simplest model you can and don't fool yourself

We were excited for our first meeting with the Launchpad Mentors. We brought all our EDA reports, the SQL terminal was ready for quick queries and fast answers, but the advice we received was simple yet valuable: ***"Start with the simplest model you can and don't fool yourself."***

And in just one quick sentence, we received two incredibly valuable pieces of advice:

1. At first, most ML projects start with incredibly high expectations, but you have to get your hands dirty to actually prove feasibility.
2. Care about the bias that you can put on data — it can eventually break your models after months of work.

This way of thinking inspired us to create what we called a development log, a document to register the progress and ideas for the project. This proved useful in keeping track of our interactions and collaborations with the Launchpad Mentors.

Later on, the development log became essential in making the onboarding of a new team member quick and smooth. Having a document that noted the progress of the project, provided comments from the Mentors, and explained the decisions we made allowed our new hire to easily understand where we were and how we got there.

After one week she was already contributing to the project and to the development log itself.

The simplest model...

What is the simplest model with the least bias? For us, it was first achieved by:

1. Dropping all columns with missing values.
2. Dropping all free text input without a clear pattern.
3. One-hot encoding all the categorical features.
4. Randomly selecting the training and test sets.
5. Doing simple linear regression.

During this process we realized that for some features the classification into "*boolean"* was not enough, as the "*not available*" value was significant too. Therefore, we had to change the related fields into categorical.

After one-hot encoding, we ended up with 3.2k features for 10k samples. We found this result to be far from optimal, because with techniques like

deep learning, you need way more samples. But we did not let this stop us from trying.

The outcome was to fine tune our model as follows:

1. Drop all the columns with missing values.
2. Encode Thumbs Up/Down fields: [ON, OFF, NotAvailable] -> [1, -1, 0]
3. Drop a free text input that didn't have a clear pattern.
4. One-hot encode all the categorical features.
5. Randomly select the training and test sets.
6. Perform simple linear regression.

After running the model, these were our results:

- **Simple Linear Regression:** R^2 score 0.830455
- **Lasso Regression:** R^2 score 0.852544
- **Random Forest:** R^2 score 0.616342

By doing a deeper inspection, we found out that our predictions were not very accurate. One even said to sell a car for negative $2,000. Obviously the simplest model was not the best, but at least we proved that the project was feasible.

The difficult reality

An exclusive role at a startup, such as data scientist, is tricky. There is a lot of work to do, things are fast paced, and you have to wear different hats every day. This experience is exciting, but also means you have to cover multiple jobs at once. It also means that it's impossible to dedicate 100% of your time to a single project when answers are often needed quickly, and your team is made up of just 2 people. We were not only dealing with the complexity of the data; we were dealing with the complexity of the workload.

Work smarter, not harder

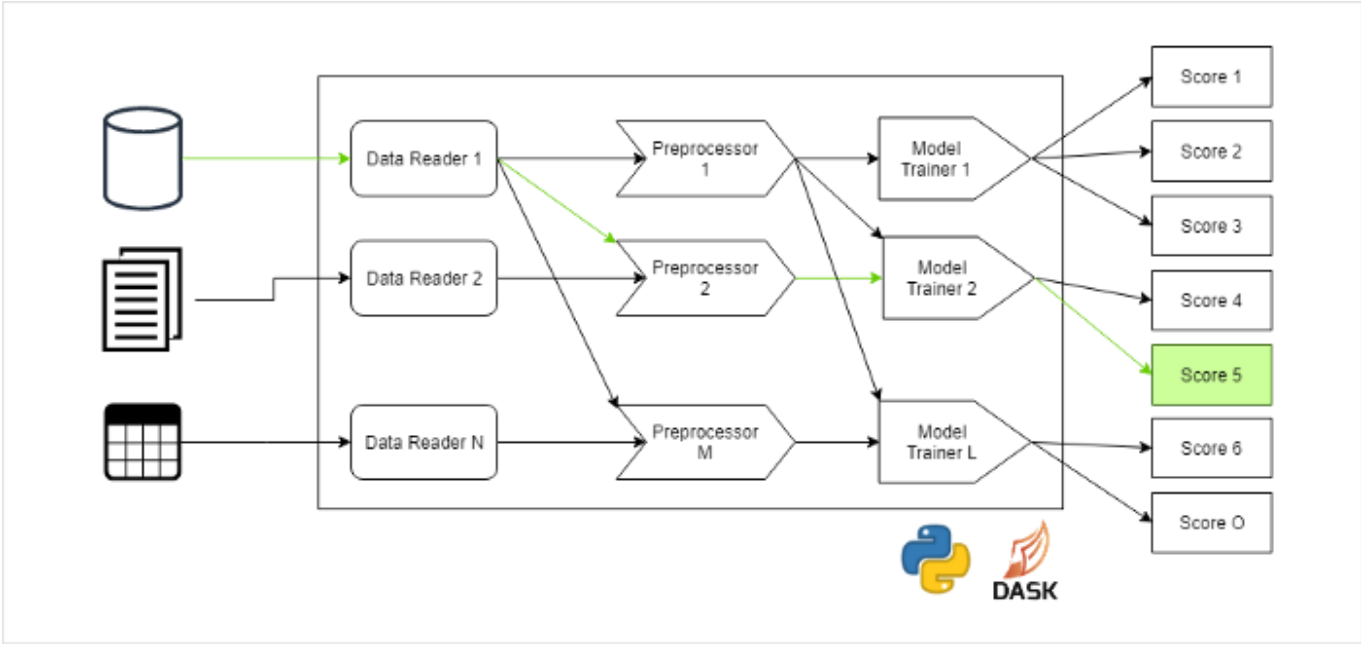
After a few weeks, we realized that we already had 20 Jupyter notebooks with different experiments, but there was always a pattern: *read data*, *preprocess,* and *train*.

The problem was that many experiments seemed promising and we didn't want to discard those approaches, so we decided to create an internal library to streamline the development process.

The library had the following components:

1. Data Readers.
2. Preprocessors.
3. Model Trainers.

We then registered and plugged these components in multiple configuration and ran multiple algorithms with multiple preprocessing methods. Since our dataset was relatively small, we could try all the combinations between methods in a reasonable time, especially using Docker and Dask to run them in parallel.



Main idea behind the internal library

Every run was like holding a small Kaggle competition where we picked the best solution. Instead of figuring out the best approach, we essentially used brute force to find all of the combinations and then kept the best one. This approach helped us leverage other team members to work and develop quickly without too much hassle.

We added the following steps in the winning approach:

1. Scale the numerical values.
2. Create exponential features for year and mileage.
3. Select the training and test sets with manufacturer distribution using random stratified sampling.
4. Train 5 different models.
5. Validate the results with fresh data, taking in consideration the current month.

After those iterations the results were the following:

- **Simple Linear Regression:** R^2 score: 3.0286×10^{-19}
- **Lasso Regression:** R^2 score: 0.893110
- **Random Forest:** R^2 score: 0.790125
- **XGBoost:** R^2 score: 0.778196

- **Tensorflow:** R^2 score: 0.916127
- **No more negative predictions.**
- Deep Learning was the leading algorithm.

During the project we kept collecting data. Our strategy was to freeze the dataset with information from the previous month, and then validate the algorithm with the new data from the current month. We called this the ***sanity check set***.

Talking it over

In our second meeting with the Launchpad Mentors we presented our new library and discussed crucial topics for the success of the project:

- Different sampling techniques for selecting the training and test sets.
- Strategies for sample weighting, data augmentation, and oversampling.
- Usage of a custom loss function.
- Papers with promising feature engineering techniques that can be applied to our dataset.

We left the meeting feeling confident about the progress we made, but we still needed to work on gaining more points on our accuracy score.

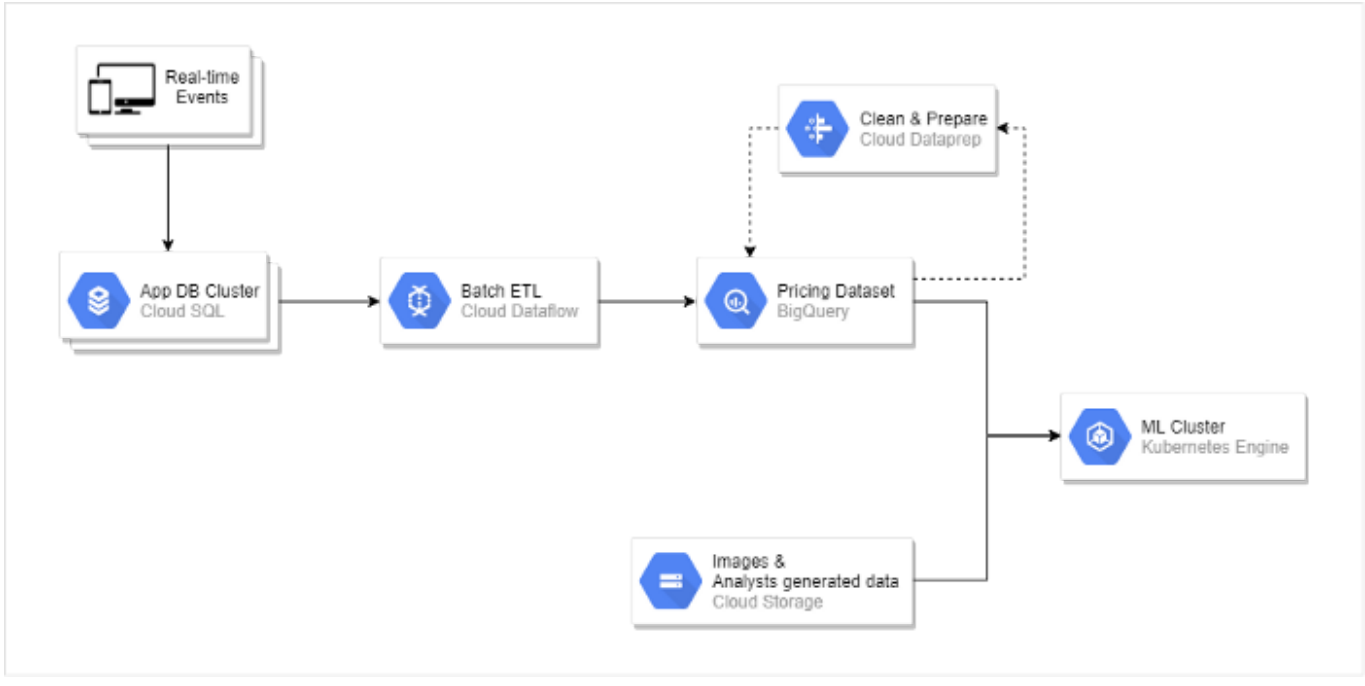
The big clean up

We decided to go deeper into the data itself and understand the features with high categories cardinality. There were interesting findings. For example, categories like “expensive” and “not cheap” existed, but there was no manual or standard description for understanding the difference between such categories. As you can guess, the selection was entirely subjective and could be grouped into a single category.

Also, we found pleasant surprises. In some of the free text inputs we detected an organic nomenclature which proved handy in obtaining extra information about a car.

Cleaning data is not the most exciting job, but we were introduced to Dataprep! This is a fantastic tool available on GCP for data cleaning and preparation. It let us concentrate on concepts instead of implementation. If you are fine coding your way out for preprocessing, then it won’t bring you much benefit. However, if you have different levels of coding skills within your team, Dataprep allows anyone contribute to the solution and then migrate the logic and concepts, already validated, to production ready code.

Finally, the overall architecture looked like this:



Architecture of the deployed solution

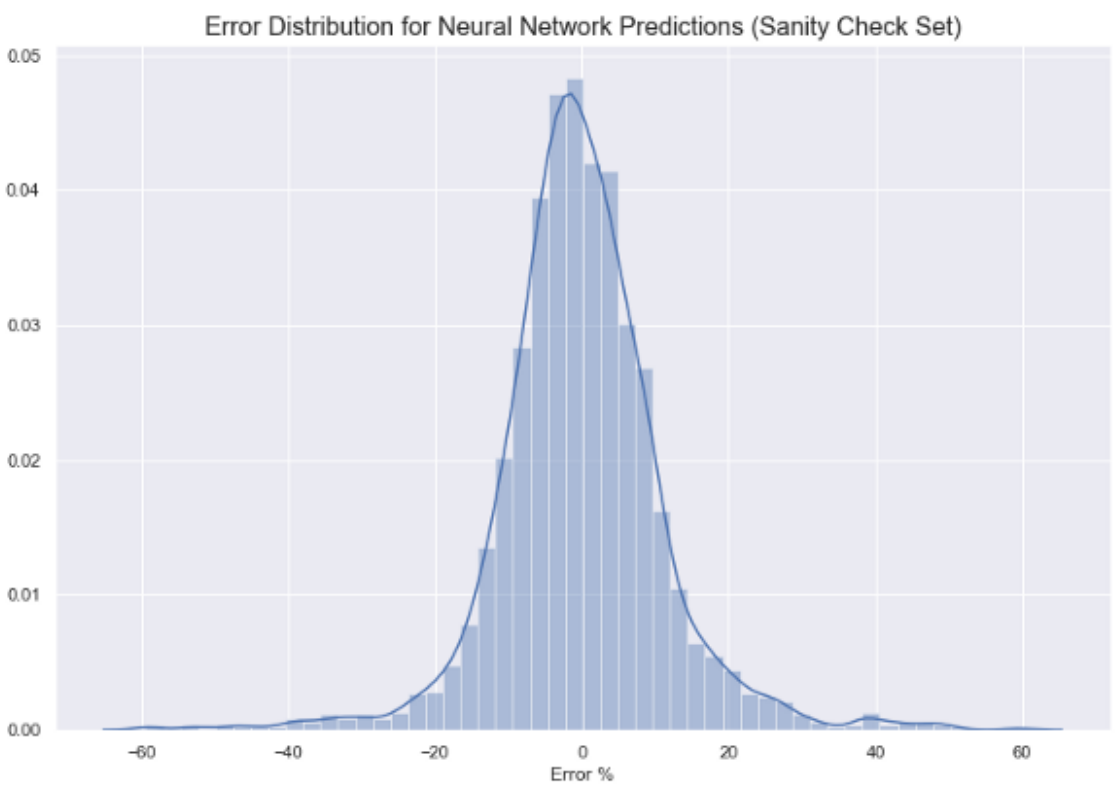
After the big clean up, we ended up with 960 features after one-hot encoding. Remember we started with 3.2k.

The results looked like this:

- **Lasso Regression:** R^2 score: 0.801170.
- **Random Forest:** R^2 score: 0.902150.
- **XGBoost:** R^2 score: 0.889196.
- **Tensorflow:** R^2 score: 0.925422.

The models didn't improve that much, mainly because the critical features didn't change, but they were running faster and were easier to explain.

The relative error to the real selling price had the following distribution:



Error distribution of prediction on the sanity check set.

Having around 50% of predictions under 5% error.

What to do when the nature of the data is most likely the problem

If you clean correctly, use the right techniques, and still don't get the results you expected, then the nature of the data is most likely your problem.

For example, it was hard to predict the price of used Porsches when they were less than 1% of the dataset. Additionally, not having a standard method for assessing the significance of certain damages made it difficult to improve the algorithm.

The good news was that these problems could be solved by collecting more data and extending what was already collected.

Recommendations

- If you are using R, **switch to Python!** Everything in GCP is prepared to work with Python, it will make development and integration way easier.
- Leverage BigQuery (ML). Load your dataset and **validate quick ideas** with a SQL query.
- Use Dataprep for cleaning your dataset, then it's easier to translate the preprocessing logic to Python.
- Use GCP ML Engine as much as you can. **Try to run many models in parallel!**
- **Automate**. Instead of copy pasting code between notebooks, dedicate time to develop a well tested software library. Create scripts and Makefiles for deploying your training jobs quickly.

Next steps

After the *Google Cloud Next 19* event, the whole landscape for ML changed in GCP with the introduction of new products like AI Hub and new ML services.

We are excited about the announcements and are now planning to simplify our solution even further by:

- Improving our internal library with Kubeflow.
- Expanding our usage of Cloud ML Engine for training and hyper-parameter tuning.
- Lowering the entry barrier for ML using AutoML Tables.

As we now look out at what is hopefully a long future, with more changes in the landscape of ML ahead, we keep in mind a short saying: **start simple and don't fool yourself.**

Pedro Castillo *is the Lead BI Engineer at Frontier Car Group, leading the analytics and machine learning projects in the organization.*

Alessandro Regonini*, VP of Engineering at Frontier Car Group. Serial entrepreneur in enterprise and mobile internet software with over 15 years experience creating, managing, and selling service and product-based software companies in Italy and internationally.*

Machine Learning Startup Lessons Analytics Google Cloud Platform

Learn more.

Medium is an open platform where 170 million readers come to find insightful and dynamic thinking. Here, expert and undiscovered voices alike dive into the heart of any topic and bring new ideas to the surface. Learn more

Make Medium yours.

Follow the writers, publications, and topics that matter to you, and you'll see them on your homepage and in your inbox. Explore

Share your thinking.

If you have a story to tell, knowledge to share, or a perspective to offer — welcome home. It's easy and free to post your thinking on any topic. Write on Medium

About Help Legal