Machine learning solution manual pdf

When I was young, I always hated being called Dale. This is mainly because my main image of what Dayles looked like was formed by Dale Gribble of King Hill as well as Dale Earnhardt Jr., a NASCAR driver. Dale Gribble image (credit), Dale Earnhardt Jr. image credit None of these Dales fit my aspirational self-esteem. On the contrary, I wanted to be called Sailor Moon. I didn't like the fact that my name was androgynous - 14 male Dales are born for every dale woman. When I asked my parents about it, their rationale was: A. women with androgynous names are potentially more successful. B. Their hipster friends just called their daughter Dale and it was just so cute! To their credit as an adult, I sure feel like I've benefited from pretending to be a person (or not directly denying it) on my resume, on Github, in my email signature, or even here. READ ALSO: Problems with tennis pitches and penalties? There's AI for this But sexism aside that if there's really something of nominal determinism - the idea that people tend to hire or lifestyle that match their names?1 And if your name has some impact on the life you lead, what responsibility should it be to choose a name for an entire person. I would not leave this responsibility to taste or randomness or trends. Of course not - I'd turn to deep learning (yes!). In this post, I'll show you how I used machine learning to build a generator named after a child (or predictor, more precisely) that takes a description of (the future) of the person and returns the name, i.e.: My baby will be born in New Jersey. She will grow up to be a software developer at Google who loves bike and coffee running. Given the biography, the model will return a set of names, Sorted by probability: Name: Linda Score: 0.04895663261413574 Name: Kathleen Score: 0.0423438773530626 3 Name: Suzanne Score: 0.0353787776665592 Name: Catherine Grade: 0.03052548505365848885 ... Theoretically, I was supposed to be Linda, but at the moment I'm very attached to Dale. If you want to try this model yourself, take a look here. Now you definitely don't have to put a lot of weight into these predictions because. they are biased and b. they are about as scientific as a horoscope. But still - wouldn't it be great to have your first child, named AI? Data set Although I wanted to create a name generator, what I really ended up building was the name of the predictor. I figured I would find a bunch of descriptions of people (biographies), block their names, and build a model that would predict what those (blocked) names should be. Fortunately, I found just such a data set here in Repo Github called Wikipedia-biography-dataset by David Grangier. The dataset contains the first paragraph of 728,321 biographies from Wikipedia, and different metadata. Naturally, there is a choice of bias when it comes to receives a biography on Wikipedia (according to Lily, only 15% of Wikipedia biographies are women, and I suppose the same can be said about non-white people). In addition, the names of people with biographies on Wikipedia tend to distort older, as many more famous people have been born in the last 500 years than in the last 30 years. To explain this, and because I wanted my name generator to give names that are popular today, I downloaded a census of the most popular baby names and cut my Wikipedia dataset only to include people with a census of popular names. I also only considered names for which I had at least 50 biographies. It left me with 764 names, most men. The most popular name in my dataset was John, who matched 10092 Wikipedia biographies (shocker!), followed by William, David, James, George, and the rest of the biblical male title list. The least popular names (which I still have 50 examples of) were Clark, Logan, Cedric, and another couple, with 50 counts each. To explain this massive skew, I'm down my dataset once again by randomly selecting 100 biographies for each name. After wrapping up the model after I had a sample of the data, I decided to train a model that, given the text of the first paragraph of the Wikipedia biography, would predict the name of the person about whom the biography was. If it's been a while since you've read a Wikipedia biography, they usually start something like this: Dale Alvin Gribble is a fictional character in the animated series Fox King of the Hill, voiced by Johnny Hardwick (Stephen Ruth, who voices Bill, and actor Daniel Stern both originally auditioned for the role). He is the creator of the revolutionary Pocket Sand defense mechanism, a fighter, a bounty hunter, a Daletech owner, a chain smoker, a weapons fanatic, and a paranoid supporter of almost all conspiracy theories and urban legends. Since I didn't want my model to be able to cheat, I replaced all the first and person's surname with an empty line: I am. Thus, the bio above becomes: - Alvin - a fictional character in the animated series Fox ... This is input into my model, and its corresponding mark of Dale's output. As soon as I prepared the dataset, I started to create a model of deep language learning. There were many different ways I could do this (here's one example in Tensorflow), but I decided to use AutoML's natural language, a code-free way of creating deep neural networks that analyze text. I downloaded my dataset into AutoML, which automatically divided it into 36,497 training examples, 4570 test examples, 4,570 test examples: Even though I tried to delete names, a few from the middlemen slipped in! To train the model, I'll re-engage on the Train tab and press the Start learning button. After about four hours Training is done. So how well did the generator name Do? If you've created models before, you know go to metrics to evaluate quality, usually accuracy and recall (unless you're familiar with these terms or need retraining, check out this good interactive demo my colleague zack Aqil built to explain them!). In this case, my model had a accuracy of 65.7% and a 2% recall. But in the case of our name generator model, these metrics aren't really what I'm saying. Because the data is very noisy - there is no correct answer to what a person should be named based on his or her life story. The names are pretty much arbitrary, which means that no model can make really great predictions. My goal was not to build a model that with 100% accuracy can predict a person's name. I just wanted to create a model that understood something about names and how they work. One way to delve into what the model has learned is to look at a table called the confusion matrix, which indicates what types of errors the model makes. This is a useful way to fine-tune or make a quick sanity check. In the Rate AutoML tab provides a confusion matrix. Here's a tiny corner of it (cut off because I had so many names in the data set): In this table, the line blanks are the true tags and the column blanks are the projected tags. The lines indicate what the person's name was, and the columns indicate which model predicted the person's name. So, for example, look at the line with the inscription ahmad. You will see a light blue box with the inscription 13%. This means that of all the biographies of people named Ahmad in our dataset, 13% were labeled Ahmad by the model. Meanwhile, looking one box on the right, 25% of the biographies of people named ahmad were (wrongly) labeled as Ahmed. Another 13% of people named Ahmad were incorrectly labeled as Alec. Although these are technically incorrect tags, they tell me that the model probably learned something about the name because Ahmed is very close to Ahmad. Same for people named Alec. The model is labeled by Alecs as Alexander 25% of the time, but in my read, Alec and Alexander are very close names. Running sanity checks Next, I decided to see if my model understands the basic statistical rules about the naming. For example, if I described someone like her, would a model predict a woman's name compared to the male name he is? For the suggestion she loves to eat, the top predicted names were Frances, Dorothy, and Nina, followed by a few other female names. Sounds like a good sign. For the suggestion He likes to eat, the top names were Gilbert, Eugene, and Elmer. So it seems the model understands some concept of gender. Next, I thought I'd check whether he was able to understand how geography played out in the names. Here are some suggestions that I've been testing and Predictions: He was born in New Jersey - Gilbert She was born in New Jersey - Francis He was born in Mexico. - Armando She was born in Mexico - Irene He was born in France. - Gilbert She was born in France. - Edith He was born in Japan - Gilbert She was born in Japan - Frances I was very impressed with the model, it seemed particularly poor to understand which names are popular in Asian countries, and usually in these cases only to return the same small set of names (i.e. Gilbert, Francis). This tells me that I haven't had enough global diversity in my training data set. Model bias Finally, I thought I'd check on one last thing. If you've read at all about the model of fairness, you may have heard that it's easy to accidentally build biased, racist, sexist, age-related, etc. models, especially if your training data set doesn't reflect the population you're building this model for. I mentioned before there is a skew in who gets a biography on Wikipedia, so I already expected more men than women in my data set. I also expected that this model, reflecting the data she was trained on, would learn about gender biases - that programmers are men and nurses are women. Let's see if I'm right: They'll be a programmer. Joseph They'll be a nurse. - Francis They will be a doctor. - Albert They will be an astronaut. - Raymond They will be a novelist. - Robert They will be a parent. - Jose They will be a model. - Betty Well, it seems the model has made to learn traditional gender roles when it comes to the profession , the only surprise (to me, at least) that the parent was predicted to have a male name (Jose) rather than a female. So obviously this model learned something about what people are called, but not quite what I hoped it would be. Guess I'm back to normal when it comes to choosing a name for my future offspring... Dale Jr.? This article was written by Dale Markowitz, an APPLIED AI engineer at Google based in Austin, Texas, where she works on machine learning in new fields and industries. She also likes to solve her life's problems with AI, and talks about it on YouTube. Published August 2, 2020 - 11:00AM UTC UTC understanding machine learning solution manual. foundations of machine learning solution manual. understanding machine learning solution manual pdf. pattern recognition and machine learning solution manual. bayesian reasoning and machine learning solution manual. tom mitchell machine learning solution manual. bayesian reasoning and machine learning solution manual pdf. foundations of machine learning solution manual pdf