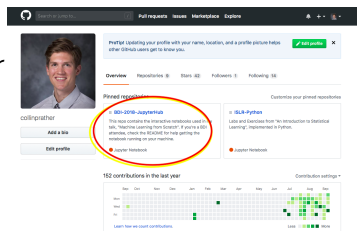


Welcome!

While you're waiting... I've prepared a Jupyter notebook that we will use to explore our data and build a machine learning algorithm from scratch. In order to get the notebook up and running on your computer:

- 1.) Head to
<https://github.com/collinprather>
- 2.) Click on the
"BDI-2018-JupyterHub"
- 3.) Scroll down and follow the
step-by-step instructions in
the readme.md



Machine Learning From Scratch

└ Welcome!

Before we get going here, I'd like to direct you to colab.research.google.com to get things set up for the live coding demo that will take place in the second half of our time today. Follow the instructions on my github at...

Welcome!

While you're waiting... I've prepared a Jupyter notebook that we will use to explore our data and build a machine learning algorithm from scratch. In order to get the notebook up and running on your computer:

- 1.) Head to <https://github.com/collinprather>
- 2.) Click on the "BDI-2018-JupyterHub" repository
- 3.) Scroll down and follow the step-by-step instructions in the readme.md



Machine Learning From Scratch

Collin Prather

September 21st, 2018

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GR Crash dataset

Good morning, and thank you all for making it out this early! These conference days are long, and to be here bright and early on day 3 is no joke, so I'm glad you're here.

We are going to be tackling machine learning from a ground-up approach today. Can I get a quick show of hands, how many of us here have experience applying machine learning to solving problems?

Before we really dive in, I want to tell you a bit about my talk today, and to do so, we're going to start with two things that help tremendously in successfully applying machine learning... (1) Understanding the math at play: a great way to build that intuitive understanding is to implement a machine learning algorithm from scratch! (2) Knowing how to process/manipulate the data to be ML ready. Now, the skills required for these two steps are admittedly a bit disparate, however, the skills needed to successfully apply ML is inherently interdisciplinary - so I think speaking on these two topics together is rather fitting. And we will, after a brief intro to ml, We will start with talking a lot about the data and do a bit of a case study on the steps in the ML process, and then we're

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GR Crash dataset

What is Machine Learning?



Machine Learning

Arthur Samuel:

Machine learning is “Field of study that gives computers the ability to learn without being explicitly programmed”.

Machine Learning From Scratch

└ Machine Learning Overview

└ What is Machine Learning?

What is Machine Learning?



Machine Learning

Arthur Samuel:

Machine learning is "Field of study that gives computers the ability to learn without being explicitly programmed".

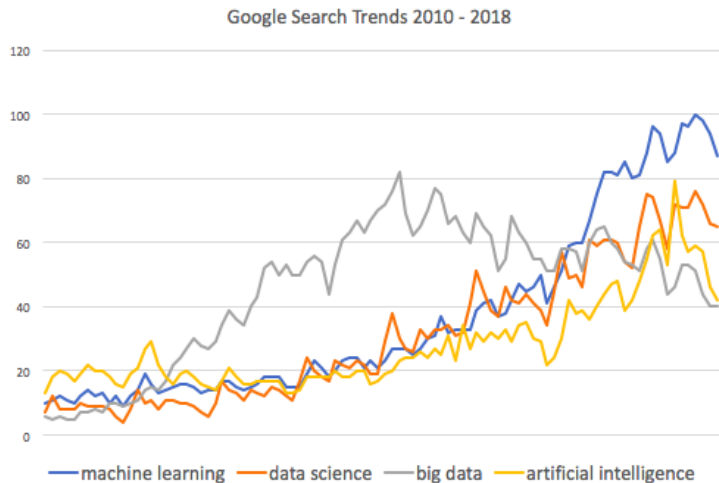
All this talk begs the question, what is machine learning? Even according to the experts, the exact definition of the field of machine learning is a bit fuzzy, but As early as 1959, Arthur Samuel was famously quoted as defining machine learning as...

ML techniques can be applied to a wide range of problems in diverse industries. In fact, ML has become ubiquitous in our everyday lives

- * Siri/ Amazon Alexa
- * Recommendation systems (amazon, netflix)
- * Fraud Detection
- * Disease diagnosis
- * Supply Chain Optimization

The list goes on and on. ML algorithms are used to learn from historical data in order to make predictions about novel data.

According to Google...



2018-09-14

Machine Learning From Scratch

└ Machine Learning Overview

└ According to Google...

According to Google...



This may not be a surprise to you, but the world's google search history reflects a steady increase in interest in machine learning and other related terms like Data Science, Big Data, and Artificial Intelligence.

This graph was pulled pretty simply from trends.google.com, which is actually fascinating, they make it very simple to look up search history trends. It's worth checking out.

What has caused this spike?

1. Data Availability

- ▶ digital data
- ▶ IoT (sensor data)

2. Computational Scale

- ▶ Moore's Law

Machine Learning From Scratch

└ Machine Learning Overview

└ What has caused this spike?

What has caused this spike?

1. Data Availability
 - digital data
 - IoT (sensor data)
2. Computational Scale
 - Moore's Law

What has caused this spike in recent years? (Not just it's spike in google searches, but also in its use in industry!) The math that powers machine learning algorithms has been around for quite a few years... so what's changed? It really boils down to two things.

1. Data Availability

- Think of all the data that comes from cell phones, it's cheap to collect and there's a ton of it.
- Machines embedded with software/sensors (commonly referred to as the Internet of Things or IoT) produce a lot of data as well.
- These are two of the most prominent examples.

2. Computational Scale (NG MLY 01 pg 10)

The rise of the big data era has given us access to astounding amounts of data. That phenomenon paired with the exponential growth we've experienced in computational advances, has created the perfect storm for the emergence of the field of machine learning.

5:30

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GR Crash dataset

2018-09-14

Machine Learning From Scratch

└ Steps in the Machine Learning Process

└ Step 0: Identify The Problem

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GR Crash dataset

Now that we have some motivation for what machine learning is and what it can do, let's move into the steps of the machine learning process.

Identify the Problem



Machine Learning From Scratch



Steps in the Machine Learning Process

Step 0: Identify The Problem

Identify the Problem

Step 0 is to identify a problem that can be framed in such a way that it can be solved using machine learning.

Let's say that you work for the city of Grand Rapids, and you've observed an influx in hit and run car crashes in recent years. This is a very concerning thing for the city and you decide that it is time to really start cracking down on the perpetrators. Now the investigative team is obviously interested in gaining any info they can on these fleeing drivers. One characteristic that may heavily inform their investigation is whether or not the driver was drunk. For example, If they knew that the driver at fault was drunk, maybe they could start at local bars near the crash site, trace some steps back. It may not be certain, but it surely be helpful, at least could point them in the right direction. Is it possible to know if a driver was drunk after the fact? Do we have data that help us answer this question?

This is a preliminary step of any machine learning project, asking the questions, can this problem be solved with data? Can we learn from

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GR Crash dataset

Get the Data

This may look like:

- ▶ SQL query
- ▶ CSV download
- ▶ Web-scraping
- ▶ Designing experiments/surveys and collecting data yourself

Get the Data

This may look like:

- ▶ SQL query
- ▶ CSV download
- ▶ Web-scraping
- ▶ Designing experiments/surveys and collecting data yourself

In our case, we'll head to [GRData](#).

Machine Learning From Scratch

└ Steps in the Machine Learning Process

└ Step 1: Get the Data

└ Get the Data

[Get the Data](#)

This may look like:

- SQL query
- CSV download
- Web-scraping
- Designing experiments/surveys and collecting data yourself

In our case, we'll head to [GRData](#).

Step 1! Obtaining the data you'll need looks very different depending on what domain you're working in. In some instances, it can be fairly simple and straightforward, for example, In a business context, most often it will require querying some sort of internal database. Could also be downloading a csv file. In other instances, it may require a bit more creativity – For particular social research, you may need to scrape the web. In some cases, you may even need to collect some data yourself! Here are two examples:

1. You're developing a new data product at your company and are collecting data to fuel it
2. You're in public health and are working to make healthcare accessible to all residents of the greater GR area. You may need to conduct your own research to identify what may be inhibiting people from reaching healthcare.

In our case, we're lucky enough to have access to a meticulously maintained public database on the city of GR: GRData. Scroll through,

Get the Data

In our case, we'll head to [GRData](#)

```
In [33]: crash = pd.read_csv('Data/CGR_Crash_Data.csv')
         crash.head()
```

```
Out[33]:
```

| | X | Y | OBJECTID | ROADSOFTID | BIKE | CITY | CRASHDATE | CRASHSEVER | CRASHTYPE | WORKZNEACT | ... |
|---|------------|-----------|----------|------------|------|--------------|------------|----------------------|-----------------|---------------------------|-----|
| 0 | -85.639647 | 42.927216 | 6001 | 929923 | No | Grand Rapids | 2007-02-16 | Property Damage Only | Side-Swipe Same | Uncoded & Errors | ... |
| 1 | -85.639487 | 42.927213 | 6002 | 935745 | No | Grand Rapids | 2007-06-22 | Property Damage Only | Side-Swipe Same | Uncoded & Errors | ... |
| 2 | -85.639387 | 42.927212 | 6003 | 926813 | No | Grand Rapids | 2007-01-08 | Property Damage Only | Head-on | Work on Shoulder / Median | ... |
| 3 | -85.639288 | 42.927210 | 6004 | 943813 | No | Grand Rapids | 2007-11-12 | Property Damage Only | Side-Swipe Same | Uncoded & Errors | ... |
| 4 | -85.639288 | 42.927210 | 6005 | 943791 | No | Grand Rapids | 2007-11-09 | Property Damage Only | Parking | Uncoded & Errors | ... |

5 rows × 77 columns

2018-09-14

Machine Learning From Scratch

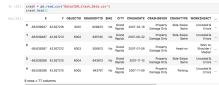
Steps in the Machine Learning Process

Step 1: Get the Data

Get the Data

[Get the Data](#)

In our case, we'll head to [GRData](#)



The screenshot shows a Jupyter Notebook interface. At the top, there's a code cell with the command `!curl -L get.data.grdata.com/GRData.csv -o GRData.csv`. Below it, a pandas DataFrame is displayed, containing columns: INDEX, X, Y, OBJECTID, OBJECTIDTO, AREA, CITY, TRANSPORT, CRASH-SEVER, INJURYTYPE, WORKSHEET, and a truncated column. The first few rows of data are visible, showing various incident details like location (e.g., JAGGERSF), date (e.g., 41.16.17.0), and severity (e.g., 004).

| INDEX | X | Y | OBJECTID | OBJECTIDTO | AREA | CITY | TRANSPORT | CRASH-SEVER | INJURYTYPE | WORKSHEET |
|-------|----------|------------|----------|------------|------|-------|--------------|-------------|------------|-------------|
| 1 | JAGGERSF | 41.16.17.0 | 004 | 000001 | 10 | GRAND | ROAD 24.0 | Property | Wid. Area | Worksheet 1 |
| 2 | JAGGERSF | 41.16.17.0 | 004 | 000100 | 10 | GRAND | Central 24.0 | Property | Wid. Area | Worksheet 1 |
| 3 | JAGGERSF | 41.16.17.0 | 004 | 000100 | 10 | GRAND | 2007.00.00 | Property | Wid. Area | Worksheet 1 |
| 4 | JAGGERSF | 41.16.17.0 | 004 | 000100 | 10 | GRAND | 2007.01.00 | Property | Wid. Area | Worksheet 1 |
| 5 | JAGGERSF | 41.16.17.0 | 004 | 000100 | 10 | GRAND | 2007.01.00 | Property | Wid. Area | Worksheet 1 |
| 6 | JAGGERSF | 41.16.17.0 | 004 | 000100 | 10 | GRAND | 2007.01.00 | Property | Wid. Area | Worksheet 1 |
| 7 | JAGGERSF | 41.16.17.0 | 004 | 000100 | 10 | GRAND | 2007.01.00 | Property | Wid. Area | Worksheet 1 |
| 8 | JAGGERSF | 41.16.17.0 | 004 | 000100 | 10 | GRAND | 2007.01.00 | Property | Wid. Area | Worksheet 1 |

After downloading the csv file, we can read the file into a pandas dataframe and explore it in our Jupyter notebook.

* note something about how we'll often refer to each row as an observation and each column as a feature

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GR Crash dataset

Machine Learning From Scratch

└ Steps in the Machine Learning Process

└ Step 2: Data Exploration

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GR Crash dataset

Step 2 is to explore the data! This is kind of an unstructured approach to understanding initial patterns in the data and potentially points of interest. This process isn't meant to reveal every bit of information a dataset holds, but rather give you direction in your analysis and potentially give you clues in how to process/model the data.[1]

Explore the Data

- ▶ Verify data
- ▶ Visualize data
- ▶ Identify patterns
- ▶ Give direction to analysis

Machine Learning From Scratch

└ Steps in the Machine Learning Process

└ Step 2: Data Exploration

└ Explore the Data

Explore the Data

- Verify data
- Visualize data
- Identify patterns
- Give direction to analysis

Now, if you're just emailed a csv file, this step is especially crucial, and it may take you some time to explore the data, get a feeling for what you're dealing with. If you are analyzing data that you work with day in and day out, this "exploration" process may be a bit more implicit.

The main idea here is to build an understanding of your data. Without an appreciation for the context of the data, it's just numbers. But when you see the data in context, it's fascinating, it's a story.

More often than not, your exploration of the data leads to more questions than answers.

Machine Learning From Scratch

└ Steps in the Machine Learning Process

└ Step 2: Data Exploration

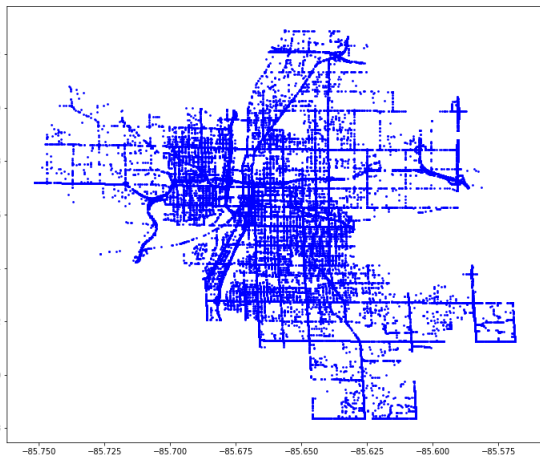
| | X | Y |
|---|------------|-----------|
| 0 | -85.639647 | 42.927216 |
| 1 | -85.639487 | 42.927213 |
| 2 | -85.639387 | 42.927212 |
| 3 | -85.639288 | 42.927210 |
| 4 | -85.639288 | 42.927210 |
| 5 | -85.639188 | 42.927208 |
| 6 | -85.639168 | 42.927208 |
| 7 | -85.639108 | 42.927207 |

Machine Learning From Scratch

└ Steps in the Machine Learning Process

└ Step 2: Data Exploration

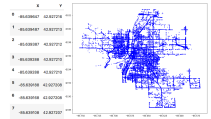
| | X | Y |
|---|------------|-----------|
| 0 | -85.639647 | 42.927216 |
| 1 | -85.639487 | 42.927213 |
| 2 | -85.639387 | 42.927212 |
| 3 | -85.639288 | 42.927210 |
| 4 | -85.639288 | 42.927210 |
| 5 | -85.639188 | 42.927208 |
| 6 | -85.639168 | 42.927208 |
| 7 | -85.639108 | 42.927207 |



Machine Learning From Scratch

Steps in the Machine Learning Process

Step 2: Data Exploration

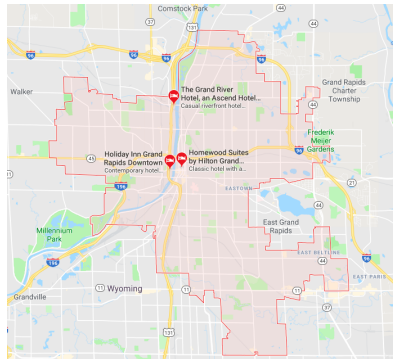
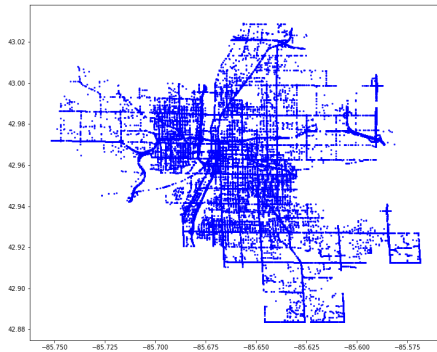


With our dataset, on car crashes, a logical place to begin would be the first two columns, containing latitudes and longitudes of each crash. This is just a snapshot of the data on the left side, and on the right, each dot represents a car crash.

Machine Learning From Scratch

└ Steps in the Machine Learning Process

└ Step 2: Data Exploration



2018-09-14

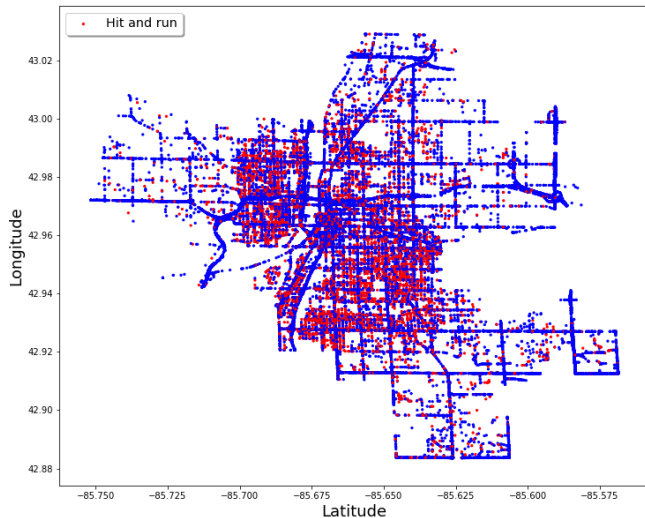
Machine Learning From Scratch

└ Steps in the Machine Learning Process

└ Step 2: Data Exploration



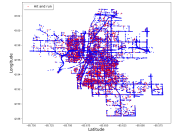
Now, this is pretty telling about our data, remember, there is nearly 73,000 crashes recorded, and if we juxtapose this plot with the city of GR, we actually see that the plot of crashes outline the city boundaries!



2018-09-14

Machine Learning From Scratch

- └ Steps in the Machine Learning Process
 - └ Step 2: Data Exploration



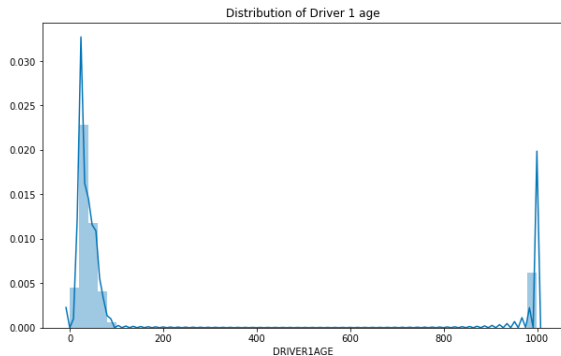
We may also be interested in hit and runs...

This plot again shows each car crash, but this time with each red dot representing a hit and run

Check the variable's distribution

```
In [41]: fig, ax = plt.subplots(figsize=(10,6))  
         ax.set_title('Distribution of Driver 1 age')  
         sns.distplot(crash.DRIVER1AGE)
```

```
Out[41]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1a742080>
```

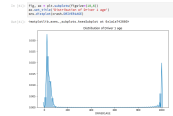


2018-09-14

Machine Learning From Scratch

- Steps in the Machine Learning Process
 - Step 2: Data Exploration
 - Check the variable's distribution

Check the variable's distribution

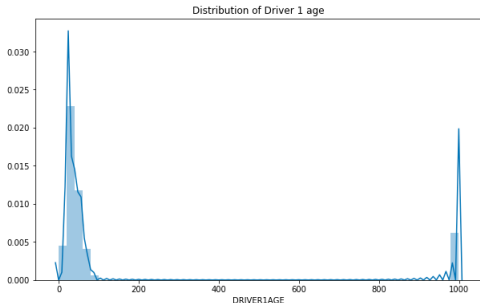


Continuing on, here we check the distribution of the age's of all the "DRIVER1"s recorded in the dataset. In my head, I would think that age would be an important feature in car crashes. And so we check it out, and it appears that there are a ton of instances bunched up between 1 and 100... that makes sense... then there is also a decent cluster of observations around 1000 years old... that does not make sense.

Check the variable's distribution

```
In [45]: fig, ax = plt.subplots(figsize=(10,6))  
ax.set_title('Distribution of Driver 1 age')  
sns.distplot(crash.DRIVER1AGE)
```

There are 8979 Driver 1's recorded as being 999 years-old.



```
In [46]: print('There are', crash.DRIVER1AGE[crash.DRIVER1AGE == 999].count(), "driver 1's recorded as being 999 years-old.")
```

There are 8979 driver 1's recorded as being 999 years-old.

Machine Learning From Scratch

- └ Steps in the Machine Learning Process
 - └ Step 2: Data Exploration
 - └ Check the variable's distribution

Check the variable's distribution



Nearly 9,000 driver 1's are recorded as being 999 years-old.. That is a good thing to know before trying to build a predictive model, as it seriously skews the data. We'll address that in our data processing stage.

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GR Crash dataset

Machine Learning From Scratch

└ Steps in the Machine Learning Process

└ Step 3: Data Preparation

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GR Crash dataset

Next, we move onto data preparation. This is the stage where we make the final manipulations to our data before feeding it into our ML algorithm. Now, you could make an argument that preparing the data is the most important part of the machine learning workflow. After all it's the data that fuels the algorithm, garbage in, garbage out! It's been shown time and time again that more/bigger data beats a better algorithm everytime. Though it usually appears to be straightforward, this step can often require a lot of creativity.

Data Selection

Use as minimal features as possible

1. Computationally efficient
2. Easier to interpret
3. Simpler is better

Data Selection

Use as minimal features as possible

1. Computationally efficient
2. Easier to interpret
3. Simpler is better

```
In [8]: crash = pd.read_csv('Data/CGR_Crash_Data.csv')
crash = crash[['X', 'Y', 'CRASHSEVER', 'DRIVER1AGE', 'DRIVER1SEX',
               'EMRGVEH', 'HITANDRUN', 'SPEEDLIMIT', 'HOUR', 'MOTORCYCLE',
               'NUMOFINJ', 'D1COND', 'D1DRINKIN']]
crash.columns

Out[8]: Index(['X', 'Y', 'CRASHSEVER', 'DRIVER1AGE', 'DRIVER1SEX', 'EMRGVEH',
               'HITANDRUN', 'SPEEDLIMIT', 'HOUR', 'MOTORCYCLE', 'NUMOFINJ', 'D1COND',
               'D1DRINKIN'],
              dtype='object')
```

Use as minimal features as possible

1. Computationally efficient
2. Easier to interpret
3. Simpler is better

[illegible]

The first part in preparing the data is simply choosing which features (you can think of as columns in the spreadsheet) you'll want to use in your model. It's generally regarded as best practice to use as minimal amount of features as possible, such that your predictive model still predicts as accurately as you need it to.

* computationally efficient, * more easily interpretable, * simpler is better

So how do we actually determine which features to use?, we can use various statistical tests, and even some algorithms to determine which features are going to be most relevant to predicting our target variable (which for us is whether or not the driver who caused the crash was drinking). We won't go into detail here.

When you are first beginning to iterate through different ml models, it is okay to kind of eyeball it, or use what you know about the context of the data to choose some features.

You'll see here the python code I've used to index our dataframe and select the feature's we'll use in building our predictive model.

[illegible]

So now we have our 13 features that we've chosen to use, and we could just send these raw features into our algorithm, and it may perform well, but it may not. It's important to remember that our ultimate goal is to build a predictive model that can make accurate predictions on new/unseen observations. When all the data comes in from a car crash that just happened, we want to be able to accurately predict whether or not it was a drunk driver that caused it.

We'll use a process called feature engineering to augment our model's ability to make accurate predictions. I'll give you a moment to read what these two prominent figures in ML have to say about machine learning.

Feature Engineering



'Coming up with features is difficult, time-consuming, requires expert knowledge. Applied machine learning is basically feature engineering.'

Prof. Andrew Ng



'At the end of the day, some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used.'

Prof. Pedro Domingos

Machine Learning From Scratch

└ Steps in the Machine Learning Process

└ Step 3: Data Preparation

└ Feature Engineering

Feature Engineering



"Coming up with features is difficult, time-consuming, requires expert knowledge. Applied machine learning is basically feature engineering."
Prof. Andrew Ng



"At the end of the day, some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used."
Prof. Pedro Domingos

Feature engineering is the process of transforming raw data into features that better represent the underlying problem to the predictive models, resulting in improved model accuracy on unseen data. - Jason Brownlee [2]

We acknowledge, however, that the data that's being recorded (from those UD10 reports) isn't necessarily guaranteed to accurately represent reality. Which is an important thing if we want to build accurate, stable predictive models.

We're going to walk-through two steps that I took to engineer features out of our data. The examples that we show here are absolutely not exhaustive when it comes to feature engineering, but I believe do build some intuition for the nature of feature engineering.

Feature Engineering: *transforming "hour" variable*

$$\begin{bmatrix} 12 \\ 2 \\ 4 \\ 18 \\ 19 \\ 6 \\ \vdots \end{bmatrix}$$

Feature Engineering: *transforming "hour" variable*

$$\begin{bmatrix} 12 \\ 2 \\ 4 \\ 18 \\ 19 \\ 6 \\ \vdots \end{bmatrix} \implies f(\text{hour}) = \frac{2 \cdot \pi \cdot (\text{hour})}{24}$$

Feature Engineering: *transforming "hour" variable*

$$\begin{bmatrix} 12 \\ 2 \\ 4 \\ 18 \\ 19 \\ 6 \\ \vdots \end{bmatrix} \Rightarrow f(\text{hour}) = \frac{2 \cdot \pi \cdot (\text{hour})}{24} \Rightarrow \begin{bmatrix} 3.14 \\ 0.52 \\ 1.03 \\ 4.71 \\ 4.98 \\ 1.57 \\ \vdots \end{bmatrix}$$

Machine Learning From Scratch

└ Steps in the Machine Learning Process

└ Step 3: Data Preparation

└ Feature Engineering: *transforming "hour" variable*Feature Engineering: *transforming "hour" variable*

$$\begin{bmatrix} 12 \\ 2 \\ 4 \\ 18 \\ 19 \\ 6 \\ \vdots \end{bmatrix} \implies f(\text{hour}) = \frac{2\pi(\text{hour})}{24} \implies \begin{bmatrix} 3.14 \\ 0.52 \\ 1.03 \\ 4.71 \\ 4.98 \\ 1.57 \\ \vdots \end{bmatrix}$$

For example, one of our columns contains the hour the crash occurred at (ranging from 0-23). It did not make sense to me to treat this column as a numerical variable, since the model would interpret 2pm as twice 1pm... while that does not make logical sense. However, I felt that to treat each hour as its own category wouldn't accurately capture the information that the hour conveys (for example, it would not capture the fact that 12pm and 12am are 12 hours apart, while 12pm and 1pm are next to each other), so I set out to transform this feature in order to convey more meaning to our algorithm. Here's what I did.

On the left we have a vector containing a few of the hours that the crashes took place at. I sent this vector through this trigonometric function that we have here.

Feature Engineering: *transforming "hour" variable*

$$\begin{bmatrix} 3.14 \\ 0.52 \\ 1.03 \\ 4.71 \\ 4.98 \\ 1.57 \\ \vdots \end{bmatrix} \Rightarrow \underbrace{\begin{bmatrix} 0.00 \\ 0.47 \\ 0.86 \\ -0.99 \\ -0.97 \\ 1.0 \\ \vdots \end{bmatrix}}_{\sin(f(\text{hour}))}, \underbrace{\begin{bmatrix} -1.0 \\ 0.87 \\ 0.51 \\ -0.002 \\ -0.26 \\ 0.001 \\ \vdots \end{bmatrix}}_{\cos(f(\text{hour}))}$$

Machine Learning From Scratch

└ Steps in the Machine Learning Process

└ Step 3: Data Preparation

└ Feature Engineering: *transforming "hour" variable*Feature Engineering: *transforming "hour" variable*

$$\begin{bmatrix} 3.14 \\ 0.52 \\ 1.03 \\ 4.71 \\ 4.98 \\ 1.57 \\ \vdots \end{bmatrix} \implies \begin{bmatrix} 0.00 \\ 0.47 \\ 0.86 \\ -0.99 \\ -0.97 \\ 1.0 \\ \vdots \end{bmatrix}, \begin{bmatrix} -1.0 \\ 0.87 \\ 0.51 \\ -0.002 \\ -0.26 \\ 0.001 \\ \vdots \end{bmatrix}$$

$\sin(f(\text{hour}))$ $\cos(f(\text{hour}))$

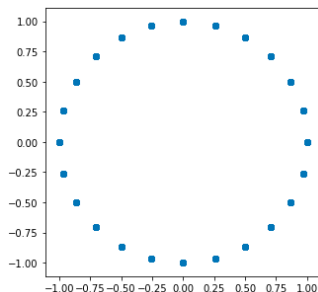
Okay, so we have this new vector, but it doesn't look too helpful at the moment. well, we take that output and map it to two separate vectors, a sin and cos transformed vector.

Feature Engineering: *transforming "hour" variable*

```
In [193]: crash['HOUR_X']=np.sin(2. * np.pi * crash.HOUR / 24.)  
          crash['HOUR_Y']=np.cos(2. * np.pi * crash.HOUR / 24.)
```

```
In [194]: # Hence, the time of day is now cyclic (just as in reality)  
          plt.figure(figsize = (5,5))  
          plt.scatter(crash.HOUR_X, crash.HOUR_Y)
```

```
Out[194]: <matplotlib.collections.PathCollection at 0x1a0daeca20>
```



Machine Learning From Scratch

Steps in the Machine Learning Process

Step 3: Data Preparation

Feature Engineering: *transforming "hour" variable*

Feature Engineering: *transforming "hour" variable*

```
In [109]: import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler

In [110]: # Create the time of day for each vehicle based on the hour of the crash
plt.figure(figsize=(10,10))
plt.scatter(hour, sin(hour), s=100)
```



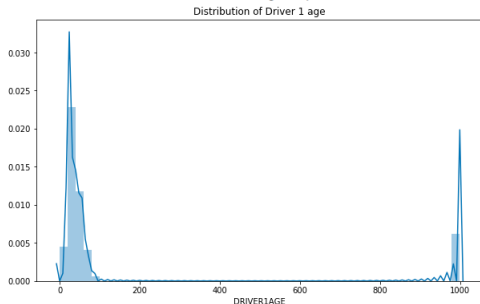
So.. why did we do all of that? Well, when we transfer that all into code, and graph the two vectors on the x-y plane, we see that they make a perfect circle... kind of like a clock..

This is exactly what we want. We've mapped our single hour column to two separate hour columns (think of them like x and y coordinates for each hour). This cyclic representation of time conveys the hour of the car crash in a way that has meaning to the algorithm. This is a really important aspect of feature engineering.

Feature Engineering: *imputing missing ages*

```
In [45]: fig, ax = plt.subplots(figsize=(10,6))  
         ax.set_title('Distribution of Driver 1 age')  
         sns.distplot(crash.DRIVER1AGE)
```

There are 8979 Driver 1's recorded as being 999 years-old.



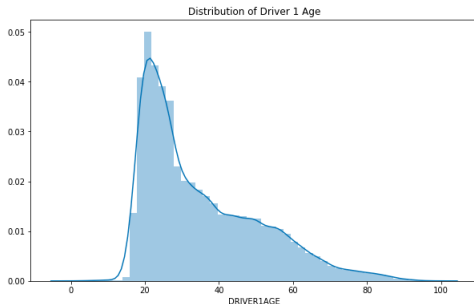
```
In [46]: print('There are', crash.DRIVER1AGE[crash.DRIVER1AGE == 999].count(), "driver 1's recorded as being 999 years-old.")
```

There are 8979 driver 1's recorded as being 999 years-old.

Feature Engineering: *imputing missing ages*

```
In [90]: # This age distribution looks much better!  
fig, ax = plt.subplots(figsize=(10,6))  
ax.set_title('Distribution of Driver 1 Age')  
sns.distplot(crash['DRIVER1AGE'])
```

```
Out[90]: <matplotlib.axes._subplots.AxesSubplot at 0x1a21b50240>
```



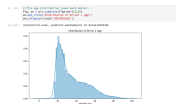
Machine Learning From Scratch

Steps in the Machine Learning Process

Step 3: Data Preparation

Feature Engineering: *imputing missing ages*

Feature Engineering: *imputing missing ages*



Now, earlier when we were exploring our data and checking the distribution of the ages, we found that there are about 9,000 erroneous ages.

When you have erroneous/outright missing data you have a couple different options.

1. drop them entirely
2. impute mean/median (make this choice depending on distribution of data)

For most cases, one of those two options will work just fine. In our case, with nearly 9,000 errors, I'm not sure any of those would make sense, so we'll opt for a bit more involved technique. (The technical term is "Multiple Imputation by Chained Equations", affectionately known as MICE). What this means is that we will fit a linear regression model to our data to predict what the missing ages could be. We can think of these as making an educated guess.

Data Processing

Two general types of data to deal with:

Data Processing

Two general types of data to deal with:

- ▶ Numerical variables (Quantitative)
 - ▶ Driver 1 age, number of injuries, etc

Data Processing

Two general types of data to deal with:

- ▶ Numerical variables (Quantitative)
 - ▶ Driver 1 age, number of injuries, etc
- ▶ Categorical variables (Qualitative)
 - ▶ Hit and run, motorcycle involved, etc

Machine Learning From Scratch

└ Steps in the Machine Learning Process

└ Step 3: Data Preparation

└ Data Processing

Data Processing

Two general types of data to deal with:

- Numerical variables (Quantitative)
 - Driver 1 age, number of injuries, etc
- Categorical variables (Qualitative)
 - Hit and run, motorcycle involved, etc

Now, we move onto processing the data that we've selected. The data processing stage naturally diverges into two substeps: dealing with numerical variables, and dealing with categorical variables.

It's important to note that some of the preprocessing steps we'll talk about here may actually be necessary for you to do to get the data in a format where you can explore it.

Numerical variables are quantitative – it's something you can measure. In our case, some numerical variables are Driver1 age, and number of injuries.

Categorical variables are qualitative, in our case, we have some binary categorical variables: like whether or not the crash was a hit and run, whether or not a motorcycle was involved. It's either one or the other. Variables can of course have multiple categories.

There is some grey area between the two... maybe mention speedlimits or hour of day?

Data Processing: *numerical variables*

```
In [91]: crash[['X', 'Y', 'DRIVER1AGE', 'NUMOFINJ']].head()
```

```
Out[91]:
```

| | X | Y | DRIVER1AGE | NUMOFINJ |
|---|------------|-----------|------------|----------|
| 0 | -85.639647 | 42.927216 | 62.0 | 0 |
| 1 | -85.639487 | 42.927213 | 31.0 | 0 |
| 2 | -85.639387 | 42.927212 | 22.0 | 0 |
| 3 | -85.639288 | 42.927210 | 30.0 | 0 |
| 4 | -85.639288 | 42.927210 | 44.0 | 0 |

Data Processing: *numerical variables*

```
In [34]: sc_X = StandardScaler()
X_scaled = sc_X.fit_transform(crash[['X', 'Y', 'DRIVER1AGE', 'NUMOFINJ']])
X_scaled_df = pd.DataFrame(X_scaled, columns = ['X', 'Y', 'DRIVER1AGE', 'NUMOFINJ'])
crash_ = crash.drop(['X', 'Y', 'DRIVER1AGE', 'NUMOFINJ'], axis=1)
crash = pd.concat([X_scaled_df, crash_], axis=1)
crash.iloc[:, :4].head()
```

```
Out[34]:
```

| | X | Y | DRIVER1AGE | NUMOFINJ |
|---|----------|-----------|------------|-----------|
| 0 | 0.406318 | -0.996140 | 1.685157 | -0.416816 |
| 1 | 0.411006 | -0.996237 | -0.269018 | -0.416816 |
| 2 | 0.413936 | -0.996298 | -0.836360 | -0.416816 |
| 3 | 0.416866 | -0.996358 | -0.332056 | -0.416816 |
| 4 | 0.416866 | -0.996358 | 0.550474 | -0.416816 |

Machine Learning From Scratch

└ Steps in the Machine Learning Process

└ Step 3: Data Preparation

└ Data Processing: *numerical variables*Data Processing: *numerical variables*

```

10 [10]: m2 = RandomForestRegressor()
11         X_scaled = StandardScaler().fit(X).transform(X)
12         y_scaled = StandardScaler().fit(y).transform(y)
13         m2.fit(X_scaled, y_scaled)
14         y_scaled = m2.predict(X_scaled)
15         y_scaled = StandardScaler().inverse_transform(y_scaled)
16         print('R^2 score: %.3f' % m2.score(X_scaled, y_scaled))
17
18 [10]: #
19         # R^2 score: 0.800000
20         # R^2 score: 0.800000
21         # R^2 score: 0.800000
22         # R^2 score: 0.800000
23         # R^2 score: 0.800000
24         # R^2 score: 0.800000
25         # R^2 score: 0.800000
26         # R^2 score: 0.800000
27         # R^2 score: 0.800000
28         # R^2 score: 0.800000
29         # R^2 score: 0.800000
30         # R^2 score: 0.800000
31         # R^2 score: 0.800000
32         # R^2 score: 0.800000
33         # R^2 score: 0.800000
34         # R^2 score: 0.800000
35         # R^2 score: 0.800000
36         # R^2 score: 0.800000
37         # R^2 score: 0.800000
38         # R^2 score: 0.800000
39         # R^2 score: 0.800000
40         # R^2 score: 0.800000
41         # R^2 score: 0.800000
42         # R^2 score: 0.800000
43         # R^2 score: 0.800000
44         # R^2 score: 0.800000
45         # R^2 score: 0.800000
46         # R^2 score: 0.800000
47         # R^2 score: 0.800000
48         # R^2 score: 0.800000
49         # R^2 score: 0.800000
50         # R^2 score: 0.800000
51         # R^2 score: 0.800000
52         # R^2 score: 0.800000
53         # R^2 score: 0.800000
54         # R^2 score: 0.800000
55         # R^2 score: 0.800000
56         # R^2 score: 0.800000
57         # R^2 score: 0.800000
58         # R^2 score: 0.800000
59         # R^2 score: 0.800000
60         # R^2 score: 0.800000
61         # R^2 score: 0.800000
62         # R^2 score: 0.800000
63         # R^2 score: 0.800000
64         # R^2 score: 0.800000
65         # R^2 score: 0.800000
66         # R^2 score: 0.800000
67         # R^2 score: 0.800000
68         # R^2 score: 0.800000
69         # R^2 score: 0.800000
70         # R^2 score: 0.800000
71         # R^2 score: 0.800000
72         # R^2 score: 0.800000
73         # R^2 score: 0.800000
74         # R^2 score: 0.800000
75         # R^2 score: 0.800000
76         # R^2 score: 0.800000
77         # R^2 score: 0.800000
78         # R^2 score: 0.800000
79         # R^2 score: 0.800000
80         # R^2 score: 0.800000
81         # R^2 score: 0.800000
82         # R^2 score: 0.800000
83         # R^2 score: 0.800000
84         # R^2 score: 0.800000
85         # R^2 score: 0.800000
86         # R^2 score: 0.800000
87         # R^2 score: 0.800000
88         # R^2 score: 0.800000
89         # R^2 score: 0.800000
90         # R^2 score: 0.800000
91         # R^2 score: 0.800000
92         # R^2 score: 0.800000
93         # R^2 score: 0.800000
94         # R^2 score: 0.800000
95         # R^2 score: 0.800000
96         # R^2 score: 0.800000
97         # R^2 score: 0.800000
98         # R^2 score: 0.800000
99         # R^2 score: 0.800000
100        # R^2 score: 0.800000

```

Here, we have our numerical variables. One problem that we still face with these numerical variables is that when we feed our data through the algorithm, the computer will view an increase in 1 latitude of latitude as equivalent to an increase in 1 year of Age of the driver. In actuality and increase in 1 degree of latitude could land you in an entirely different neighborhood, while 37 y/o and 38 y/o are basically exactly the same. We will rescale the numerical variables through a process called standardization. This is not necessary for all ml algorithms, but generally will not hurt if we do it.

When we standardize the variables, we rescale them so that they all have a mean of 0 and a S.D. of 1.

Notice that in this snippet, the number of injuries was 0 for all of them, and now is negative... this means that having 0 injuries in a car crash is below average...

Data Processing: *categorical variables*

```
In [111]: crash[['CRASHSEVER', 'DRIVER1SEX', 'EMRGVEH', 'HITANDRUN',  
                'MOTORCYCLE', 'D1COND', 'D1DRINKIN']].head()
```

```
Out[111]:
```

| | CRASHSEVER | DRIVER1SEX | EMRGVEH | HITANDRUN | MOTORCYCLE | D1COND | D1DRINKIN |
|---|----------------------|------------|---------|-----------|------------|-----------------|-----------|
| 0 | Property Damage Only | F | No | Yes | No | Appeared Normal | No |
| 1 | Property Damage Only | M | No | Yes | No | Unknown | No |
| 2 | Property Damage Only | F | No | No | No | Appeared Normal | No |
| 3 | Property Damage Only | M | No | Yes | No | Appeared Normal | No |
| 4 | Property Damage Only | M | No | No | No | Appeared Normal | No |

Data Processing: *categorical variables*

In [135]: `dummies.head()`

Out[135]:

| | CRASHSEVER_Fatal | CRASHSEVER_Injury | CRASHSEVER_Property Damage Only | DRIVER1SEX_F | DRIVER1SEX_M | DRIVER1SEX_U |
|---|------------------|-------------------|------------------------------------|--------------|--------------|--------------|
| 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 2 | 0 | 0 | 1 | 1 | 0 | 0 |
| 3 | 0 | 0 | 1 | 0 | 1 | 0 |
| 4 | 0 | 0 | 1 | 0 | 1 | 0 |

5 rows × 21 columns

Machine Learning From Scratch

└ Steps in the Machine Learning Process

└ Step 3: Data Preparation

└ Data Processing: *categorical variables*Data Processing: *categorical variables*

IN [107]: [display\(mtcars\)](#)

Out[107]:

| | mpg | wt | qsec | vs | am | gear | carb |
|----|-------|-------|-------|----|----|------|------|
| 1 | 16.43 | 2.62 | 16.99 | 0 | 4 | 4 | 4 |
| 2 | 17.82 | 2.875 | 17.05 | 1 | 4 | 4 | 4 |
| 3 | 15.83 | 2.32 | 15.42 | 0 | 4 | 4 | 4 |
| 4 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 5 | 17.01 | 2.645 | 16.99 | 1 | 4 | 4 | 4 |
| 6 | 15.83 | 2.32 | 15.42 | 0 | 4 | 4 | 4 |
| 7 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 8 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 9 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 10 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 11 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 12 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 13 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 14 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 15 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 16 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 17 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 18 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 19 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 20 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 21 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 22 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 23 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 24 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 25 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 26 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 27 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 28 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 29 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 30 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 31 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 32 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 33 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 34 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 35 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 36 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 37 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 38 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 39 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 40 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 41 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 42 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 43 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 44 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 45 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 46 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 47 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 48 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 49 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |
| 50 | 16.99 | 2.668 | 16.99 | 1 | 4 | 4 | 4 |

5 rows x 8 columns

Here we have our categorical variables. In our excel spreadsheet, this looks great, nice and clean.., but if we want to squeeze this data into a model, we need to manipulate it into a format that makes sense for math. Basically, we are going to turn all of our categories into 1's and 0's, representing yes' and no's. You can think of this as transforming the data to turn everything into a yes or no question.

Was the driver sex male? No. Was the driver sex female? yes.

I've seen this called creating dummy variables, or one-hot-encoding

This is just a snapshot, you see that our columns of categorical variables grew from 7 to 21

Come up with more concrete explanation

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GR Crash dataset

Choosing a Model/Representation

| | Classification | Regression |
|---------------------|---|---|
| Supervised | <ul style="list-style-type: none">• Logistic Regression• Naive-Bayes• KNN• SVM | <ul style="list-style-type: none">• Linear Regression• Decision Trees• Random Forests |
| Unsupervised | <ul style="list-style-type: none">• Apriori• Hidden Markov Model | <ul style="list-style-type: none">• PCA• K-means• SVD |

Machine Learning From Scratch

Steps in the Machine Learning Process

Step 4: Model Selection

Choosing a Model/Representation

Choosing a Model/Representation

| | Classification | Regression |
|--------------|---|---|
| Supervised | <ul style="list-style-type: none">• Logistic Regression• Naïve-Bayes• KNN• SVM | <ul style="list-style-type: none">• Linear Regression• Decision Trees• Random Forests |
| Unsupervised | <ul style="list-style-type: none">• Apriori• Hidden Markov Model | <ul style="list-style-type: none">• PCA• K-means• SVD |

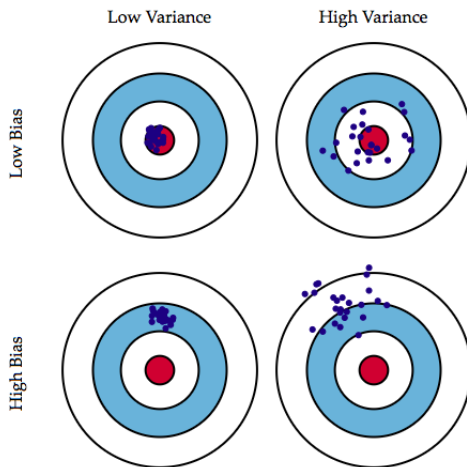
And with that, our data is ready to be fed into a predictive model! The final step is to choose which predictive model to use. There are hundreds to choose from, and trade-offs associated with each. The good news is, as we alluded to earlier there are different types of machine learning problems and each come with their own set of algorithms – so this narrows down our search considerably.

In our case, we're working on a supervised classification problem, so the upper left hand corner displays some common algorithms for problems like ours. It's very common to test a handful of them and choose the one that performs best on your dataset (or even combine some of them into an ensemble model.)

Remember, the ultimate objective to choose a model that learns a predictive rule (which comes in the form of an equation) that can be generalized to new observations (both for classification and regression)

Talk about difference between supervised/unsupervised and maybe even mention reinforcement learning?

Bias-Variance Tradeoff



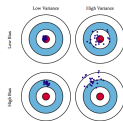
Machine Learning From Scratch

Steps in the Machine Learning Process

Step 4: Model Selection

Bias-Variance Tradeoff

Bias-Variance Tradeoff



One critical aspect to take into consideration when choosing a model is called the bias-variance tradeoff. These dartboards create a great analogy for understanding the bias-variance trade-off. I got this analogy from a paper written by Pedro Domingos, I'm not positive that he was the creator, but regardless, I think it's incredibly useful. Bias and Variance have some specific definitions in statistics, but in this context, they are words that we use to describe the behavior of a machine learning model. For example, If I were to say that our model has high variance, then that means that its predictions would vary significantly if we were to train it on a different sample of our dataset.

If I were to say that our model was highly biased, then that would mean that it's predictions are very consistent, regardless of the sample of data that we trained it on. (that does not necessarily mean accurate, as we see in the bottom left corner.)

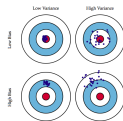
Generally, bias and variance are inversely related, and controlled by the complexity of our model. If it is a highly complex model, like a neural

Machine Learning From Scratch

└ Building a Support Vector Machine from Scratch

└ Bias-Variance Tradeoff

Bias-Variance Tradeoff



Here we'll first be going through a general example of building an svm from scratch on a toy dataset, then apply some of Python's ml tools to our crash data set to predict whether or not a car crash was caused by a drunk driver!

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GR Crash dataset

Machine Learning From Scratch

└ Building a Support Vector Machine from Scratch

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to OR Crash dataset

Here, we will begin by importing the necessary libraries – (audience should just run these). So here is what our plotted data looks like. And here is a snapshot of data set. Note that for each data point (each row/each dot), we have an x_1 value, and x_2 value, and a y value (which represents the target variable, which we are trying to predict). In this case, we've rather defined the red data points to be labeled as -1 and the blue data points to be labeled as 1. It may seem weird to call them x_1 and x_2 , but we'll address why we do that in a little bit.

Machine Learning From Scratch

└ Building a Support Vector Machine from Scratch

└ Representation

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GRU Crash dataset

Representation: Support Vector Machine

The first step in building any sort of predictive model is choosing a representation. For our example today, we choose a Support Vector Machine. Big picture - The goal of a SVM is to To find the optimal separating hyperplane which maximizes the margin of the training data. Now, a hyperplane is kind of like a fancy word for a line, or maybe better put, it's a general way to talk about a line. The black line on this graph of our dataset is a hyperplane that separate the two classes (red/blue). Since our data is two dimensional (we have an x , and a y (1, 2)) the hyperplane is a line

The margin of the training data refers to how much is between the hyperplane and the two classes of data on each side.

Machine Learning From Scratch

└ Building a Support Vector Machine from Scratch

└ Representation

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GR Crash dataset

How do we find the optimal separating hyperplane?:

Now, as we mentioned, the goal of the SVM is to find the optimal separating hyperplane, so how do we do that? It appears that there can be many different separating hyperplanes (often there is infinite). We're going to choose the hyperplane that is as far away as possible from each class of datapoints.

We want to maximize the margin because It generalizes better to classifying unseen observations - \hat{y} (meaning that it makes better predictions)

If we have our usual equation for a line: $y=mx+b$, then all we need to do is find that optimal m and b that characterize the optimal hyperplane!

Machine Learning From Scratch

└ Building a Support Vector Machine from Scratch

└ Representation

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GRU Crash dataset

Understanding the Definition of a hyperplane:

As we've seen previously, the separating hyperplane that we keep referring to (in 2-d) is just a line. I'm confident that you're familiar with the way we define lines:

Now, a quick refresher that b is just a constant that represents the y-intercept, m (usually defines the slope) is a coefficient to the variable x (also a constant) and the variable y even has a coefficient as well, in this case it's 1. if we try to expand this equation $y=mx+b$ to more and more dimensions(variables), it's not a real great form to express it in. So instead, we choose to set the whole equation equal to zero. With some algebraic manipulation, we get that:

So, that looks a bit unnatural..so its common convention to represent all coefficients/constants with β , and all variables with X_i , so that we don't run out of variables to use.

In our Crash data, we use 14 variables...

2018-09-14

Machine Learning From Scratch

└ Building a Support Vector Machine from Scratch

└ Evaluation

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GR Crash dataset

Evaluation

Machine Learning From Scratch

- └ Building a Support Vector Machine from Scratch
 - └ Evaluation

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GRU Crash dataset

Quick refresher on the dot product:

Now that we have established a representation for our SVM, we'll move on to defining how we'll evaluate its accuracy. Let's start with talking for a minute about the dot product. The dot product has many different names across fields of mathematics: inner product, and linear combination are also common. Regardless of its name, it is a useful operation to use between vectors. To put it simply, if you take the dot product of two vectors, you multiply all their corresponding elements and take the sum. (note that the vectors must have the same dimensions). The output of the dot product is not another vector, but just a scalar value.

Machine Learning From Scratch

- └ Building a Support Vector Machine from Scratch
 - └ Evaluation

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to OR Crash dataset

Dot product as distance from hyperplane:

Now you'll notice that this dot product indeed looks very similar to the way that we've defined the equation of a hyperplane. It turns out that, when we take the dot product of our weights vector \vec{w} and a single datapoint $\vec{x}^{(i)}$, the resulting number that we get can be thought of as the distance from the data point to the hyperplane (how far away it is). And since we're trying to maximize the width of our margin, having an efficient way to calculate how far away each datapoint is from the hyperplane proves to be crucial.

Now, if we're being mathematically rigorous, we may want to frame that a bit differently, but at least for all intensive purposes, we can think of it this way!

The fact that we've defined the equation of a hyperplane to be the dot product of our weights and features equal to zero implies that the hyperplane and the weights vectors are perpendicular (aka orthogonal).

And since they are orthogonal, we're able to exploit some linear algebra

Machine Learning From Scratch

- └ Building a Support Vector Machine from Scratch
 - └ Evaluation

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to OR Crash dataset

The Hinge Loss Function is defined as:

Now, armed with this ability to calculate how far away each data point is from the hyperplane, we can now use the Hinge Loss Function to quantify how wrong each of our predictions are!

One more important thing to note about the dot product is that it is signed, meaning that it can be either positive or negative, namely, data points below the hyperplane will have a negative distance from the hyperplane, and datapoints above the hyperplane, will have a positive distance from hyperplane.

Talk about how we will predict which class each data point comes from, then use the hinge loss to tell us how wrong we are.

Machine Learning From Scratch

- └ Building a Support Vector Machine from Scratch
 - └ Evaluation

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GR Crash dataset

The Hinge Loss Function is defined as (Part 2): Breaking it down
1 minus y times the dot product of the weights vector and the datapoint itself.

Let's break this down piece by piece. First, we'll look at the inputs. \vec{w} is a vector of weights (coefficients) in the linear equation. $\vec{x}^{(i)}$ is a vector representing a single datapoint (the i^{th} datapoint), or a single row in our dataset. $y^{(i)}$ is the label associated with the datapoint (which is either 1 or -1). The output is a scalar value (a number) representing a penalty for how wrong our prediction was. The greater the penalty, the worse our estimated weights were in classifying the data.

Machine Learning From Scratch

- └ Building a Support Vector Machine from Scratch
 - └ Evaluation

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to OR Crash dataset

The Hinge Loss Function is defined as (Part 3): little subscript plus sign

One thing I've neglected to mention thus far is the little subscript plus sign at the end of the loss function. That little plus sign denotes that our loss function only looks at positive values, meaning that the output of function is a negative value, it's just going to change it to zero. This intuitively makes sense! If we get a negative penalty for our algorithm... that means that the prediction must be very correct, so instead, we'll just assign a penalty of 0, basically no penalty at all.

Machine Learning From Scratch

- └ Building a Support Vector Machine from Scratch
 - └ Evaluation

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring SciKit-Learn and applying to OR Crash dataset

The Hinge Loss Function is defined as (Part 4): working it out

Now, if you actually try to work this out with the datapoints, it can be a bit confusing, as there are lots of negatives and the sign of the function is constantly changing, so I've included this little table that works out small examples for all 4 cases, when the data point is correctly and incorrectly classified from the negative class, and when the data point was correctly and incorrectly classified from the positive class. For these little examples, I've assumed that the datapoint is 3 units away from the hyperplane. You'll notice if you skip to the bottom that the penalty for both incorrectly classified datapoints is 4, while the penalty for the correctly classified datapoints is 0.

Machine Learning From Scratch

- └ Building a Support Vector Machine from Scratch
 - └ Evaluation

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to OR Crash dataset

The Hinge Loss Function is defined as (Part 5): essence of ml

This is really worth emphasizing, because it doesn't just apply to SVM's but is the basis of machine learning. Machines learn by associating a quantitative penalty to incorrect predictions. The machine then sets out to minimize the penalty, which ultimately will result in making the maximum amount of correct predictions.

This is the essence of the current state of artificial intelligence. All AI is based on this principle, everything from autonomous vehicles, to amazon's recommendation system.

Machine Learning From Scratch

- └ Building a Support Vector Machine from Scratch
- └ Evaluation

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to OR Crash dataset

Hinge Loss with regularization term:

Now, with the hinge loss function we've defined, when we try to minimize the loss function, we will (if possible) find a separating hyperplane that perfectly classifies the data i.e. no misclassifications. Which sounds great! The only problem with that is that if there are any outliers in our dataset, it can seriously skew the hyperplane. Below we have two nearly identical datasets...the one on the right has one outlier (circle it with mouse) and it has completely changed our hyperplane.

Again, we need to remember that our goal is to find a hyperplane that will best classify new datapoints...so we don't want it to be affected by outliers.

In order to defend against the outlier, we will introduce a regularization term to the end of our hinge loss function, parameterized by the coefficient λ . The λ coefficient acts as a tuning parameter (in our example, we will explicitly hard-code a value for lambda), that helps to balance the bias-variance tradeoff. Generally, as the value for lambda

2018-09-14

Machine Learning From Scratch

└ Building a Support Vector Machine from Scratch

└ Optimization

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GR Crash dataset

Optimization

Machine Learning From Scratch

- └ Building a Support Vector Machine from Scratch
 - └ Optimization

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to OR Crash dataset

Optimization

Now, we've been talking quite a bit about minimizing the penalty to our algorithm for mis-classifying data points, but it is not immediately obvious how to minimize the penalty.

We're going to use an algorithm called Stochastic Gradient Descent that will iteratively find the weights that minimize the total "loss" or "cost" to our Support Vector Machines, resulting in the very best possible predictions. **Look at this visually in two ways! 1. With stanford web demo, 2. with gradient descent drawing**

Machine Learning From Scratch

- └ Building a Support Vector Machine from Scratch
- └ Optimization

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to OR Crash dataset

Stanford Web App

For one of its course, Stanford has created this awesome web demo that actively animates gradient descent. Here, they have three different classes (red, blue, green), while we only have two (red, blue), yet the principles are exactly the same. Gradient Descent begins with totally random separating boundaries, this means that we begin with random weights, \vec{w} , random coefficients in the equation of the separating hyperplane. (click randomize a few times)

The algorithm then iteratively updates the weights little by little until the separating hyperplane is correctly classifying the data points.(start repeated update with softmax)

So this is how gradient descent works.

Machine Learning From Scratch

- └ Building a Support Vector Machine from Scratch
 - └ Optimization

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to OR Crash dataset

Gradient Descent Drawing

I want to show you another angle we can look at this from. Take a look at this drawing. Imagine that this red line is the hinge loss function (preface: it's not. But it is a representative of the general concept of gradient descent), which, remember, outputs how wrong our predictions are, so we want to minimize this function, by finding the weights that correspond to the smallest possible output. All we have to do is move towards the bottom of this function, which is exactly what gradient descent does.

Machine Learning From Scratch

└ Building a Support Vector Machine from Scratch

└ Optimization

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to OR Crash dataset

Transition to Math of GD

Pragmatically speaking, the algorithm is going to loop through each observation (row) in our dataset, and check if our current weights would have correctly or incorrectly classified it. If it would have incorrectly classified the data point, then the algorithm updates the weights accordingly.

A natural question to ask then is, how are we going to update the weights? How do we move from the top of the function to the bottom? We will do so by using partial derivatives of the hinge loss function. We do this because:

- The derivative of the hinge loss function gives us the slope of our hinge loss function at our current values for \vec{w} . Given the slope, we want to "move" or update our weights in the direction that the slope is decreasing i.e heading towards the bottom of the function.

Machine Learning From Scratch

└ Building a Support Vector Machine from Scratch

└ Optimization

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to OR Crash dataset

Calculating the partial derivatives:

Okay, so let's calculate the partial derivatives. Using a rule from calculus, we can separately calculate the partial derivative of the first and second terms of the hinge loss function (loss term and regularization term) separately.

As we can see here, the partial derivative of the loss term depends on whether the datapoint was correctly or incorrectly classified. (0 if correctly classified, or $-y$ times x if incorrectly classified). The partial derivative of the regularization term is 2 times λ times w .

2018-09-14

Machine Learning From Scratch

└ Building a Support Vector Machine from Scratch

└ Optimization

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GR Crash dataset

Update Rules:

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GR Crash dataset

Machine Learning From Scratch

Exploring Scikit-Learn and applying to GR Crash dataset

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GR Crash dataset

Conclusion Now, Admittedly, I found and explored this dataset for the purpose of presenting here, (look up notes from michael to better explain this?)

But truly, the challenge in ML is not the math (the machine will do it for you), and often it's not the data itself (as aforementioned, it's becoming increasingly more accessible). The challenge is often building a ML model that is stable, has "business" value, and is able to be used in production. I thought that maybe I can leave you with this challenge: If you knew, as we've shown today, that given data from the Police's U10 reports, you could accurately predict if causes were caused by drunk drivers, how would that be helpful? What could that be used for? How could we apply it/put it into use?

These are the questions that should drive us as data scientists.



<https://www.sisense.com/glossary/data-exploration/>



<https://towardsdatascience.com/understanding-feature-engineering-part-2-categorical-data-f54324193e63>



<http://scott.fortmann-roe.com/docs/BiasVariance.html>