Bring your own dataset

Introduction to Machine Learning

Kristina Plazonic
Office of Advanced Research Computing

Agenda for the morning

- Basic concepts and the titanic dataset
- Demo of Datastudio for data exploration, visualization, and dashboards
- Hands on exercise

- What is machine learning? How to pose a problem and ML workflow
- Demo of Jupyter notebooks, Colab, RStudio, AutoML Tables
- Hands on exercise

- An overview of deep learning

Expectations

- Be collaborative! Work in pairs or groups
- Be hands on!
- Don’t get bogged down in code - this is a skill acquired slowly and painfully
- Feel free to leave if you need to

Resources - to be revisited

- Where to get the datasets
- Where can you get free computational cycles
- Where to find tutorials
- What is available on campus as help
- Let’s organize something - a meetup, study group!

About OARC, Office of Advanced Research Computing

Office of 20 sysadmins, research scientists/facilitators, and grant support

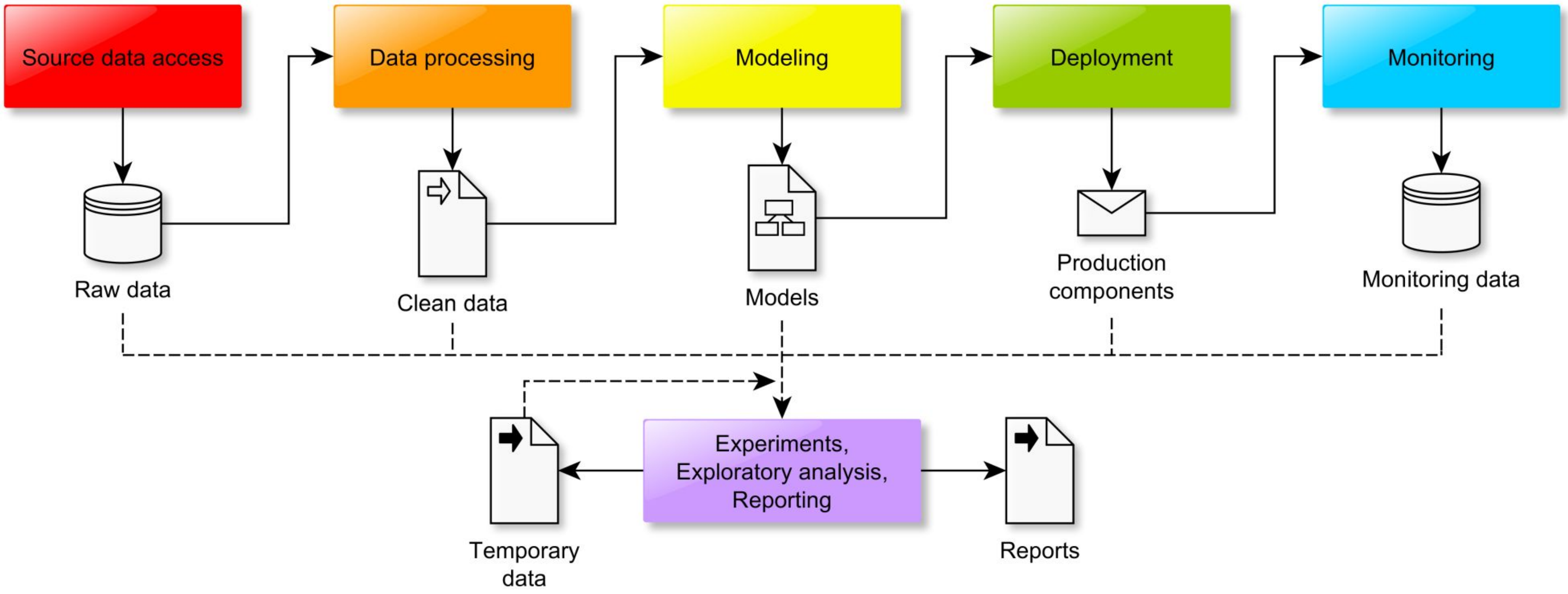
Administering several clusters under the Amarel umbrella: 600+ servers on all three campuses

Newest addition: 4 servers with 8 GPU cards each

Office hours and email support
help@oarc.rutgers.edu

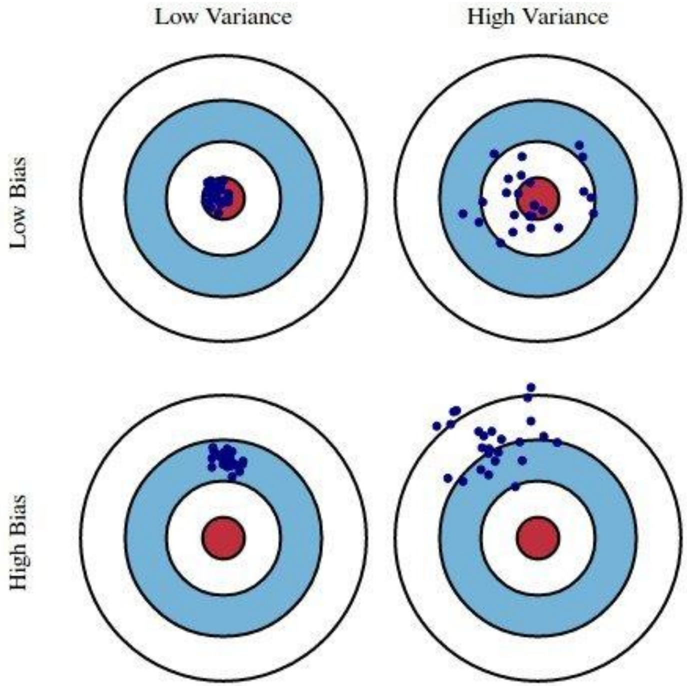
Part 1: Data Exploration

Machine Learning Pipeline

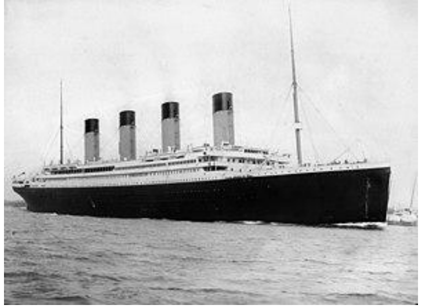


Basic concepts

- supervised vs semisupervised vs unsupervised machine learning - https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html
- features and labels
- continuous vs categorical variables
- classification vs regression
- structured vs unstructured data
- variance and bias - how complex is your model



The Titanic dataset



Competition Description

The sinking of the RMS Titanic is one of the most infamous shipwrecks in history. On April 15, 1912, during her maiden voyage, the Titanic sank after colliding with an iceberg, killing 1502 out of 2224 passengers and crew. This sensational tragedy shocked the international community and led to better safety regulations for ships.

One of the reasons that the shipwreck led to such loss of life was that there were not enough lifeboats for the passengers and crew. Although there was some element of luck involved in surviving the sinking, some groups of people were more likely to survive than others, such as women, children, and the upper-class.

In this challenge, we ask you to complete the analysis of what sorts of people were likely to survive. In particular, we ask you to apply the tools of machine learning to predict which passengers survived the tragedy.

https://www.kaggle.com/c/titanic

Data Dictionary

Variable	Definition	Key
survival	Survival	0 = No, 1 = Yes
pclass	Ticket class	1 = 1st, 2 = 2nd, 3 = 3rd
sex	Sex	
Age	Age in years	
sibsp	# of siblings / spouses aboard the Titanic	
parch	# of parents / children aboard the Titanic	
ticket	Ticket number	
fare	Passenger fare	
cabin	Cabin number	
embarked	Port of Embarkation	C = Cherbourg, Q = Queenstown, S = Southampton

Example rows

	# Passen...	# Survived	# Pclass	A Name	A Sex	# Age	A SibSp	# Parch	A Ticket	# Fare	A Cabin	A Embark...
1	1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.25		S
2	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	PC 17599	71.2833	C85	C
3	3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282	7.925		S
4	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	53.1	C123	S
5	5	0	3	Allen, Mr. William Henry	male	35	0	0	373450	8.05		S
6	6	0	3	Moran, Mr. James	male		0	0	330877	8.4583		Q
7	7	0	1	McCarthy, Mr. Timothy J	male	54	0	0	17463	51.8625	E46	S
8	8	0	3	Palsson, Master. Gosta Leonard	male	2	3	1	349909	21.075		S

Explaining basic concepts on this example

- **supervised problem** because there is something to predict, an outcome variable, and examples were given to us of who survived
    - If we didn't know who survived and instead were asked to group passengers together by similarity, that would be unsupervised
- features and labels
    - **Feature** = independent variable = predictor = variables like sex, pclass, age
    - **Label** = dependent variable = to be predicted = single column = "survival" column
- continuous vs categorical variables
    - **Continuous** variable = one that is numeric and can have decimals, e.g. fare
    - **Categorical** variable = it's values are categories = e.g. sex, pclass
- classification vs regression
    - **Classification** = if label is categorical
    - **Regression** = if label is continuous
- structured vs unstructured data
    - **Structured** = comes in a table - easier to deal with!
    - **Unstructured** = text, pdfs, audio
- variance and bias - at the end

Hands-on exercises

- What kind of variable is “Pclass”? (values 1,2,3)
- What kind of machine learning problem is this?
- Name all the numerical and all the categorical variables

Basics of visualization

Dimension vs. Metric

dimension=categorical
metric = numeric
(date and geographic have special types)

Aggregations

(are default)
Sum, average, count, min, max, ...

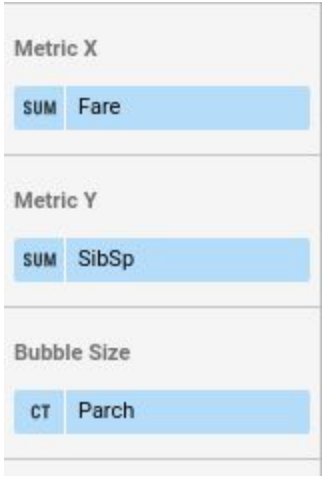


Chart types

Each chart type has a defined combo of data types

Table = dimension + agg(metric)
Bar plot = dimension +agg(metric)

Heat map = 2 dimensions + agg(metric)
Scatter = 2 metrics
Bubble chart = 3 metrics + 1 dimension (color)

Filter, Sort

Allow focus on particular portion of the dataset

Chart types

Heat map = 2 dimensions + 1 agg

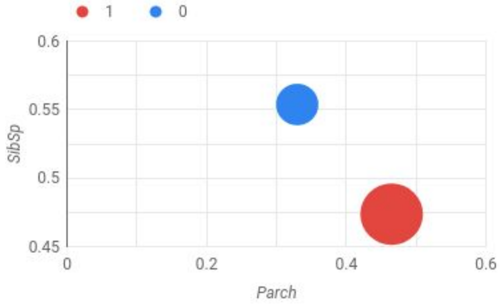
		Survived / PassengerId	
Sex	0	1	Grand total
male	468	109	577
female	81	233	314
Grand total	549	342	891

Bar, table =

1 dimension + 1 agg (+1 optional dimension in color)

Scatter, bubble

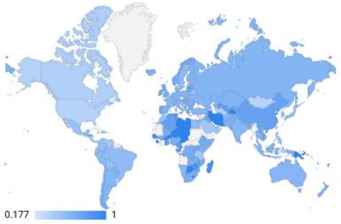
- 3 metrics + 1 dimension
- needs a unique ID if you do not want aggregation



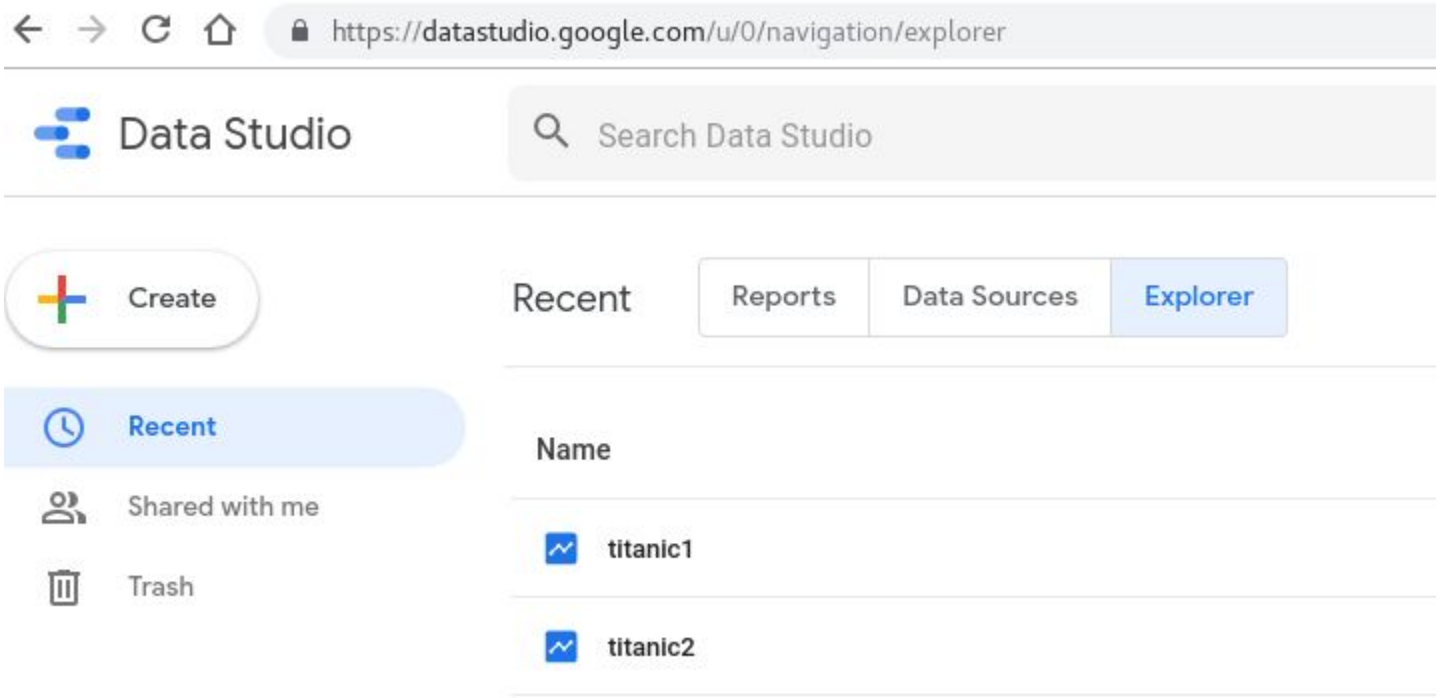
Time series

1 time + n aggs

(or choropleth)



Demo - datastudio



Hands-on exercise

- Go to https://datastudio.google.com - choose “Data Source” and “Shared with me”
- Create a bar chart with Sex and Survived
- Create something with your own dataset

Lessons/Warnings

Data acquisition

- Your data may not come as structured data
- Creating datasets can be publishable result - because creating datasets may be hard!

Data check

- Data validation - e.g. watch out for outliers or nonsensical - do a visual check of your data!
- Filling in the missing values
- Creating new features may help - your domain knowledge will be important here

Reproducibility

- Once you have a good way to preprocess your data, save the code, and the result in a timestamped directory. Clean up and document your code.

Part 2: Machine learning

Agenda:

- Split data into training, validation and testing
- Fit a particular algorithm to produce a model
- Evaluate how well the algorithm will perform on unseen data

Machine learning on this example

- Given examples (rows of the table, individual passengers), learn a pattern of when a particular row is associated with 0 (died), and when with 1 (survived)
- We do this by computing an error function between actual and predicted
- The algorithm adjusts the learned model to minimize that error

E.g. the **model** may be:

“if (sex == “male” and pclass == 3) then 0 else 1” - a “decision tree”

“if (- 0.6*pclass + 0.1*fare > 0.5) then 1 else 0” - a “linear model”

The **parameters** found by the model training would be “male” and 3; or 0.6,0.1,0.5

Examples of ML problems

- predict wine price from rainfall, temperature, soil characteristics
- predict whether a person will die in the following year from heart attack based on some measurements and disease risks (Framingham study started in 1948)
- predict who will win in a Supreme Court case - https://www.sciencemag.org/news/2017/05/artificial-intelligence-prevails-predicting-supreme-court-decisions
- predict house price from number of rooms, sqft, number of bathrooms, apartment or house - "housing dataset"
- predict the price of a particular stock the next day on the stock exchange
- predict whether the stock price will go up or down

- predict the gender of the person from the retinal image of the person - https://ai.googleblog.com/2018/02/assessing-cardiovascular-risk-factors.html
- predict the age of person from their photo - https://www.how-old.net/
- predict the species of an animal from the snapped picture from a camera trap

- predict the next word in a sentence - "language model"
- predict whether a yelp review is positive or negative - "sentiment analysis"
- predict the profession of the political candidate from their biography

Examples of badly posed problems

- Predict where will be a computer bug in computer code
- Predict which portion of a web page will be “About” purely from html structure

How to pose a problem correctly?

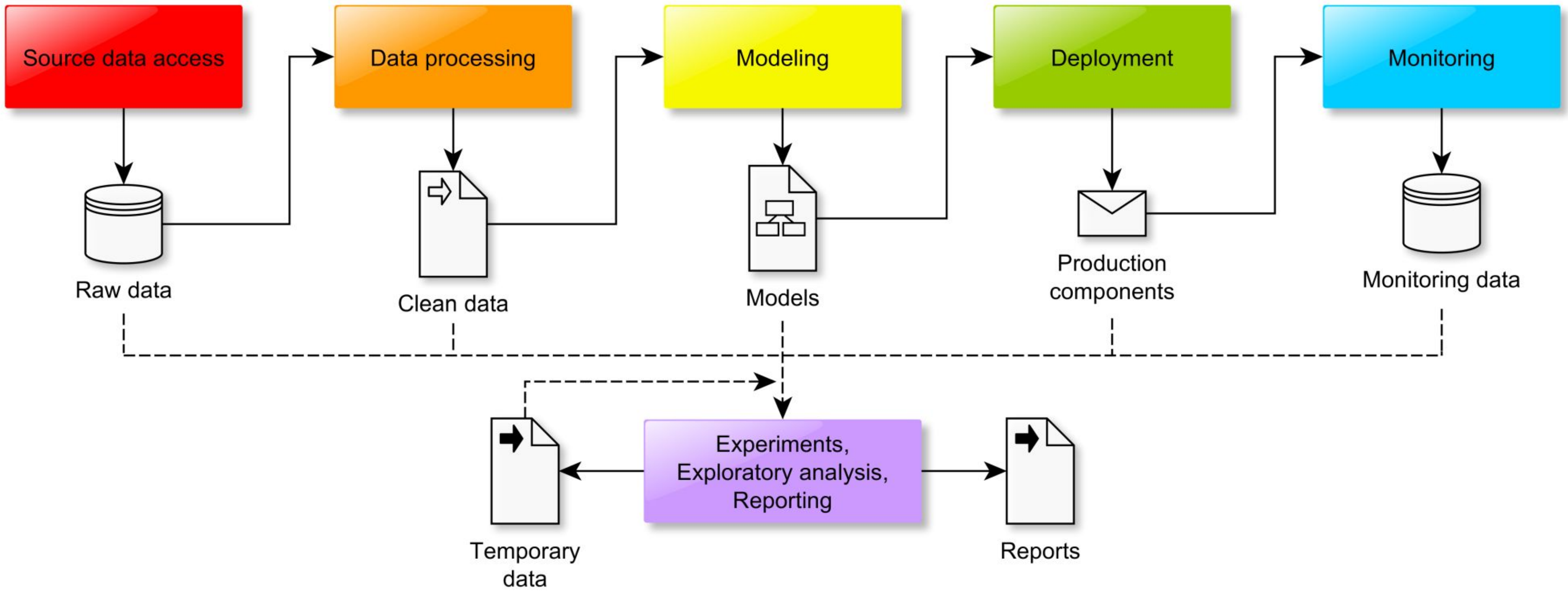
How to choose machine learning algorithm

https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

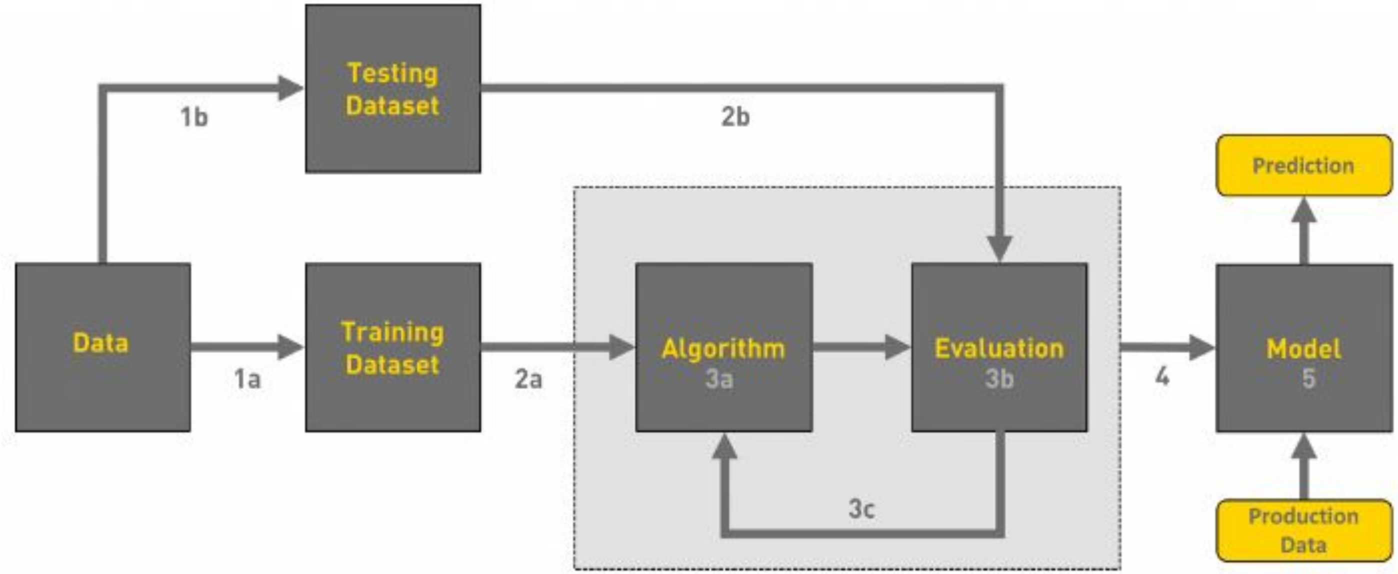
Common algorithms:

- **Linear regression** - and related - Lasso, Ridge, ElasticNet
- **Logistic regression** - despite the name, this is a classification algo
- Naive Bayes - (was used for NLP; it’s counting frequencies)
- Decision trees - and derived - **Random Forest**, Gradient Boosting Machine
- Nearest neighbors - kNN - (good but can be slow)
- SVM (Support Vector Machine) - (popular in early 2000’s, can be slow)
- Neural networks (deep learning)

Data Analysis stages



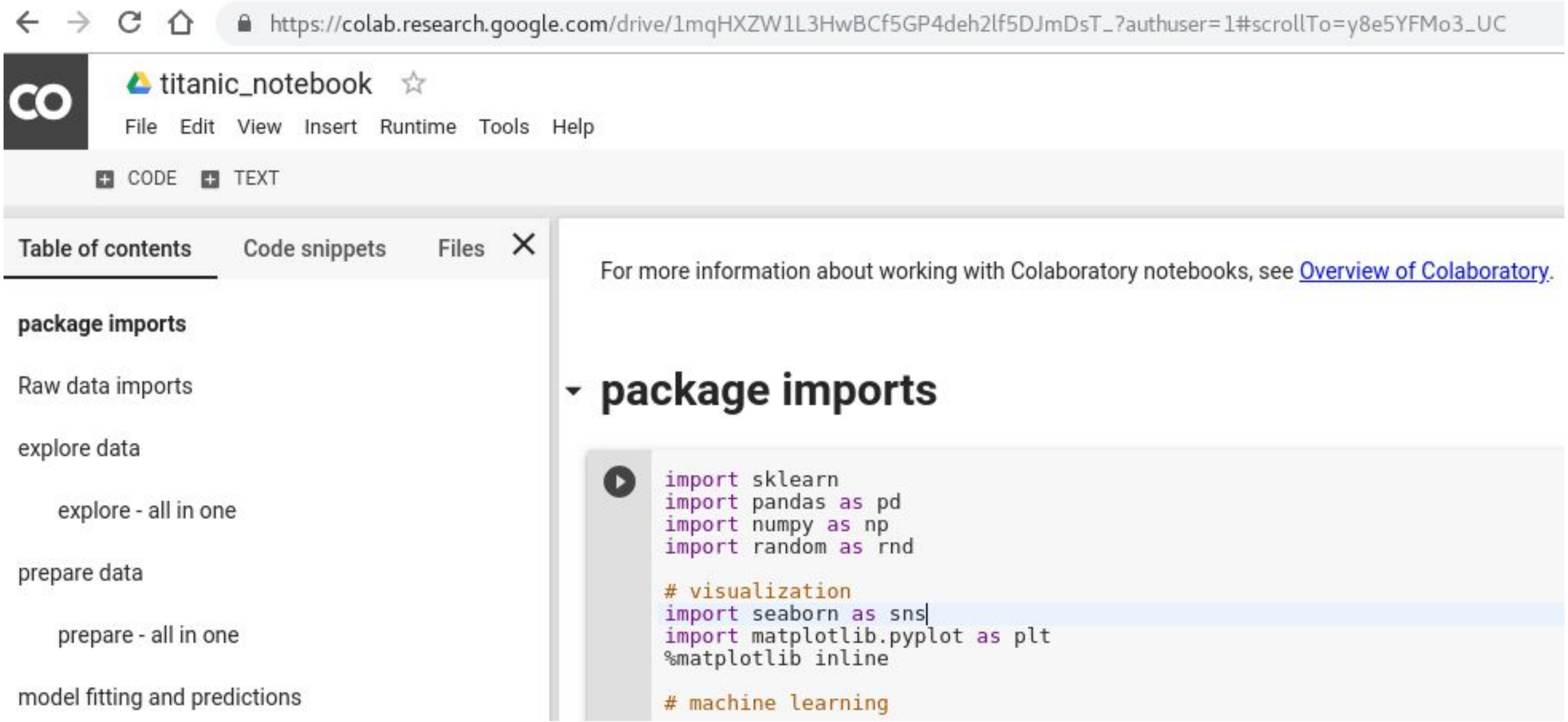
Machine learning workflow - model building



Where to run your code

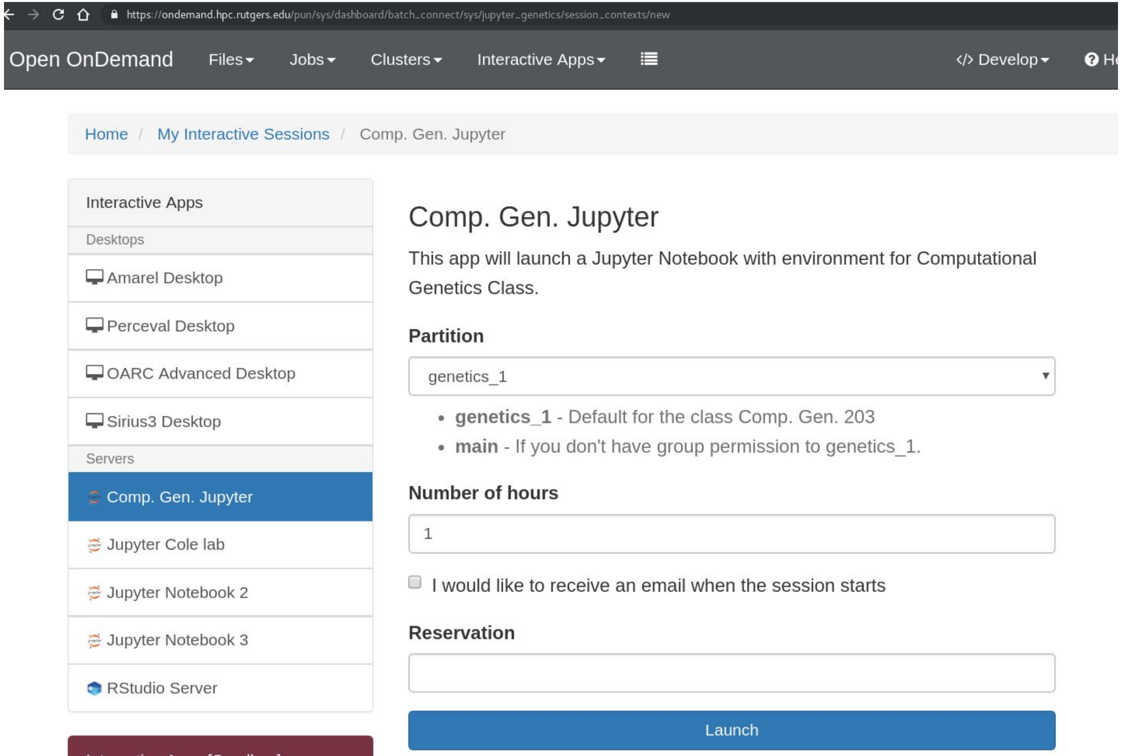
- Google Colab
- Amarel OnDemand has a Jupyter notebook interface

Google Colab



Jupyter notebook on Amarel

https://ondemand.hpc.rutgers.edu



Jupyter keyboard shortcuts

- 2 types of cells - code and text
- 2 modes of interacting with cells - “editing” and “command”
  - CTRL-Enter for executing the cell
  - ESC to enter the command mode
  - m to change the cell to text (i.e. “markdown”)
  - CTRL-/ for commenting out a piece of code
  - dd for deleting the cell
  - CTRL-a for new cell above, CTRL-b for new cell below
  - CTRL-minus for splitting a cell

Most important python tips

Tab-completion

```
In [ ]:  pd.|
pd.api
pd.bdate_range
pd.Categorical
pd.CategoricalIndex
pd.compat
pd.concat
pd.core
pd.crosstab
pd.cut
pd.DataFrame
```

Help on functions with ? at the end

```
In [3]:  pd.crosstab?

In [ ]:
```

```
Signature: pd.crosstab(index, columns, values=None, rownames=None, colnames=None, aggfunc=None, marg
opna=True, normalize=False)
Docstring:
Compute a simple cross-tabulation of two (or more) factors. By default
computes a frequency table of the factors unless an array of values and an
aggregation function are passed

Parameters
----------
index : array-like, Series, or list of arrays/Series
    Values to group by in the rows
columns : array-like, Series, or list of arrays/Series
    Values to group by in the columns
values : array-like, optional
    Array of values to aggregate according to the factors.
    Requires `aggfunc` be specified.
aggfunc : function, optional
    If specified, requires `values` be specified as well
```

Notebook demo

Basic code to create model from X (predictors) and Y (outcome)

```
model = RandomForestClassifier()           # declare algorithm type
model.fit(X_train, Y_train)                # find the model i.e. actual
weights
Y_pred = model.predict(X_new)              # predict outcomes on unseen data
model.score(X_known, Y_known)              # find accuracy of the model
```

Basic code to evaluate how good your model is on unseen data:

```
scores = cross_val_score(model, X_train, Y_train, cv=5) #crossvalidation
```

The rest of the basic code:

Data import

```
train_df = pd.read_csv("train.csv")
```

Data pre-processing

```
train_df['Age']      = train_df['Age'].fillna(mean_age)
```

Choosing features and outcome:

```
X_train = train_df[ ['Pclass', 'Sex', 'Age', 'SibSp', 'Parch'] ]
Y_train = train_df[ "Survived" ]
```

Dataset code - import breast cancer dataset

Scikit-learn has a submodule, datasets, with ~10 standard datasets; breast cancer is one of them. When you load data, you get an object with `data, target, feature_names, DESCR` - not a DataFrame - define a function that turns it into dataframe

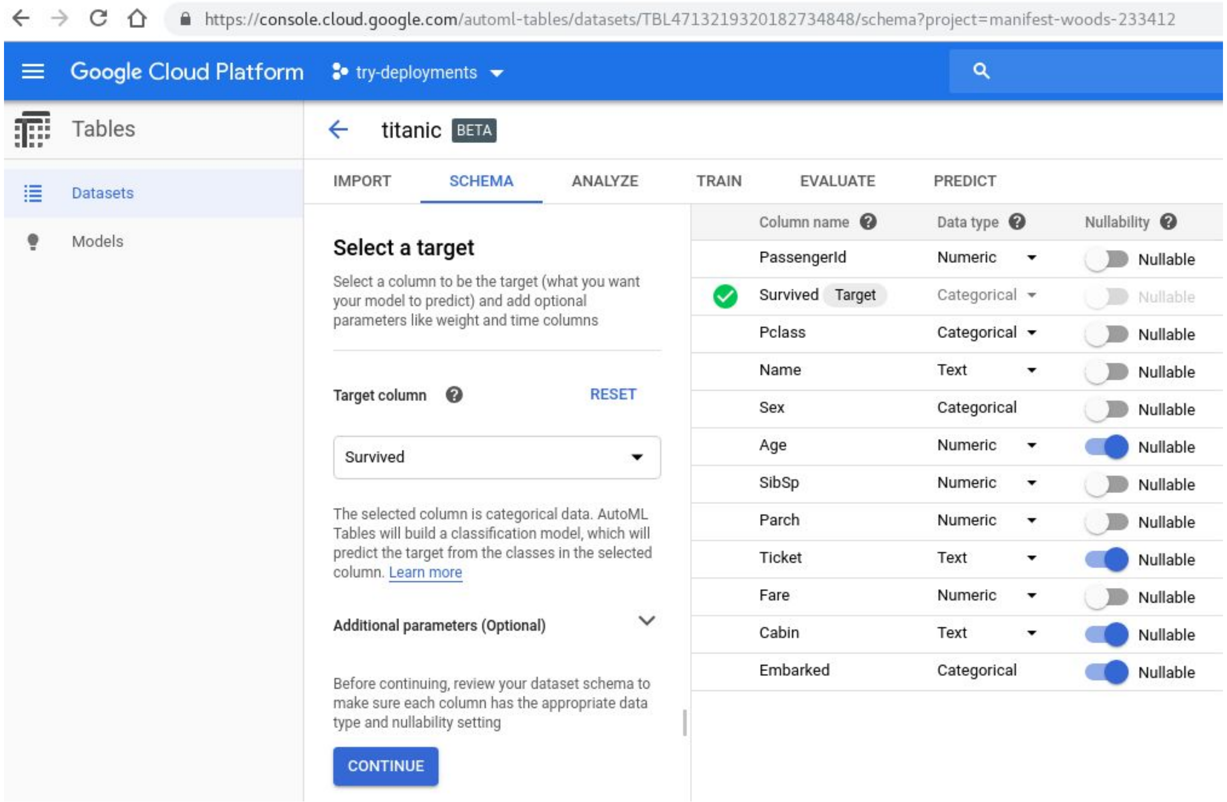
```
from sklearn import datasets

def sklearn_to_df(sklearn_dataset):
    df = pd.DataFrame(sklearn_dataset.data,
columns=sklearn_dataset.feature_names)
    df['target'] = pd.Series(sklearn_dataset.target)
    return df

df_breast = sklearn_to_df(datasets.load_breast_cancer())
df_breast.head()
```

Visualization code - Matplotlib

Hands-on exercise

- Open the Jupyter notebook
- Execute the cells
- Try to change the algorithm with another algorithm
- Try to change the data to your dataset
- Try to score on the training set - what accuracy do you get? (will revisit)

Cloud offerings - GCP AutoML Tables, AWS Sagemaker



GCP AutoML Tables - Output

IMPORT | SCHEMA | ANALYZE | TRAIN | EVALUATE | PREDICT

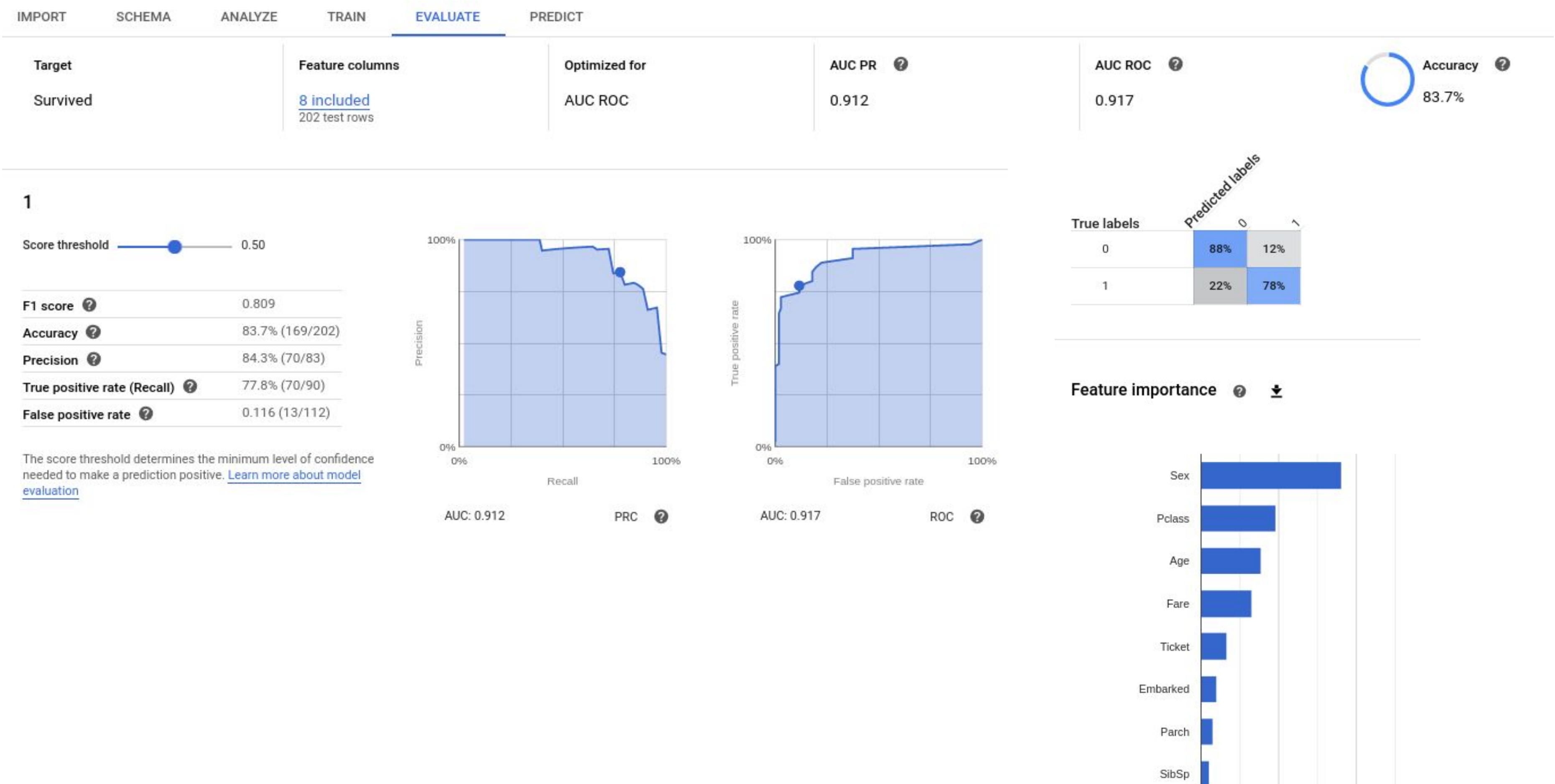
Up to date. Last modified on Jul 21, 2019, 3:46:25 PM

All features	12
Numeric	5
Categorical	4
Text	3

Filter instances

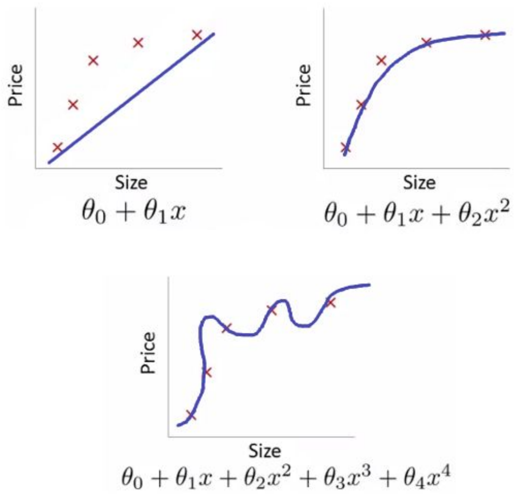
Feature name ↑	Type	Missing	Distinct values	Invalid values	Correlation with Target	Mean	Standard deviation
Age	Numeric	19.692% (333)	89	0	0.433	29.883	14.505
Cabin	Text	76.878% (1,300)	148	0	—	—	—
Embarked	Categorical	0.177% (3)	4	0	0.122	—	—
Fare	Numeric	0% (0)	248	0	0.512	32.341	50.044
Name	Text	0% (0)	891	0	—	—	—
Parch	Numeric	0% (0)	7	0	0.488	0.378	0.795
PassengerId	Numeric	0% (0)	891	0	0.483	467.526	246.256
Pclass	Categorical	0% (0)	3	0	0.129	—	—
Sex	Categorical	0% (0)	2	0	0.141	—	—
SibSp	Numeric	0% (0)	7	0	0.491	0.51	1.095
Survived Target	Categorical	0% (0)	2	0	—	—	—
Ticket	Text	0% (0)	681	0	—	—	—

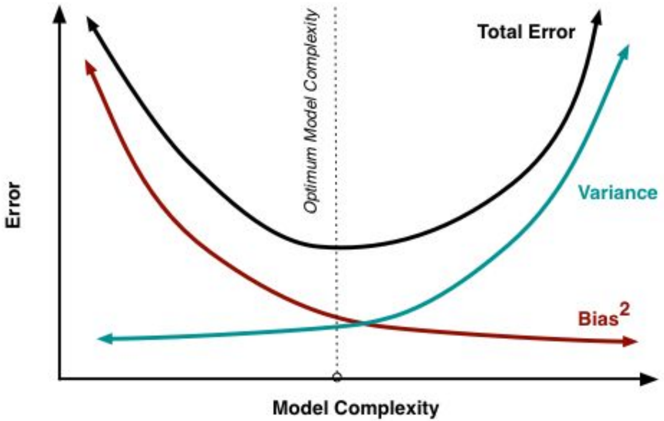
GCP AutoML Tables - Output

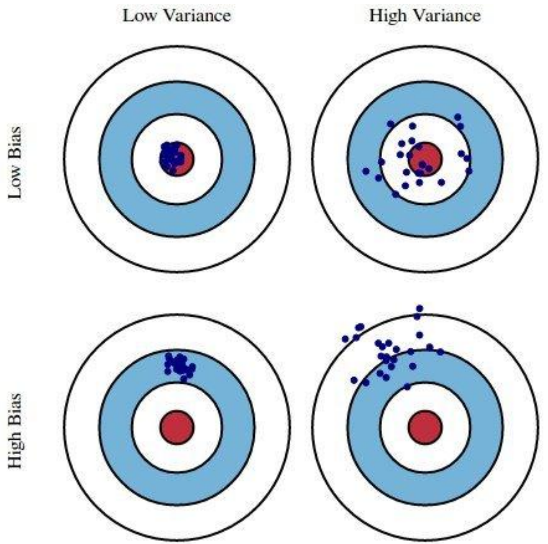


Bias-variance tradeoff

There is a sweet spot between model **complexity** and model **generalizability** - important when choosing the final model







Pitfalls to watch for

- Data on which you want to predict is not similar to data on which you train
- Data leakage - e.g. if your data has time component, separate train-test by time
- The preprocessing must be defined on training data only, but applied to every new data (e.g. in filling in missing values for the test data, you need to use the mean from the training data, not the test data)
- New categories appearing in the testing dataset may cause your model to error out on new predictions
- Imbalanced dataset (one label is overly represented)
- Overfitting - your model does not generalize well because it’s too complicated

Summary

- It is crucial how you represent the data (tabular, text, images are ok) - in the end, all your data must be a multi-dimensional numerical matrix
- Changing an ML model is as easy as swapping one line of code
- The real work is in acquiring and cleaning data, and analyzing the results
- Always produce a base model first (and fast) to compare progress
- More data is usually probably more useful than better algorithm
- Your expertise as a domain expert is crucial - you know if features are inadequate, if they can be created (and how) from the given data
- You will have a tough time if you have more features than examples (more columns than rows)

(continued)

- Sometimes creating two models on two subsets of the dataset is best
- Look at where the model is wrong (the confusion matrix) - you may get ideas for additional features

Challenging situations to model

- When number of examples (rows) is smaller than number of features(columns)
  - E.g. for genomic data - small number of samples, but large number of features (genes)
  - *limma* (microarrays), *edgeR*, *DEseq(2)* (RNAseq) = R packages that apply statistical methods
- Time series - they have a whole subfield of special methods - e.g. ARIMA
  - Usually take the previous *n* measurements to be features
- Categorical column with a high number of labels e.g. US zip code
  - E.g. apply mean encoding
- Not a clear tabular structure, e.g. data is naturally a graph - subfield of ML and active area of research

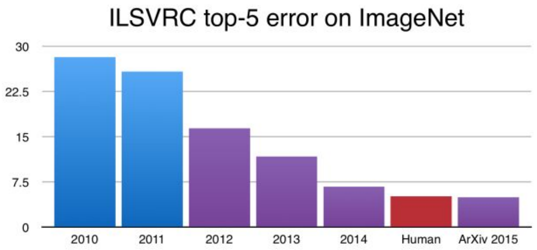
Deep learning

What is it?

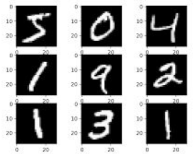
What is it good for?

Vision and text!

Brief history

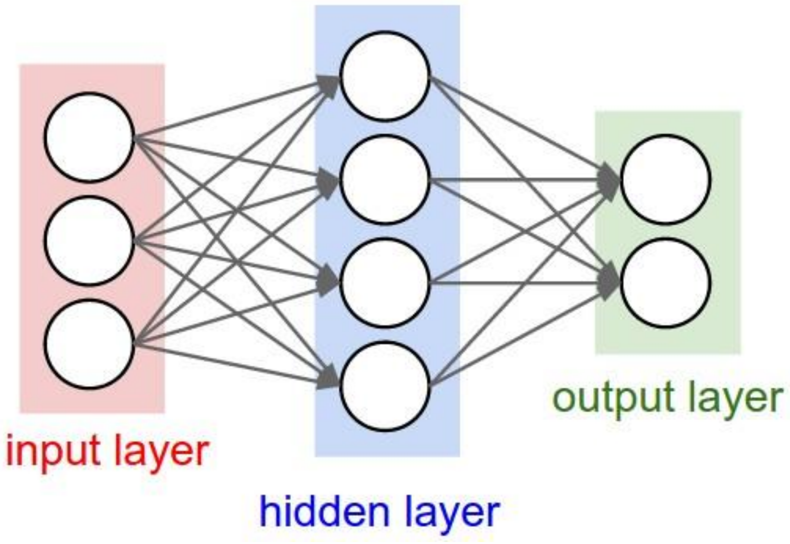


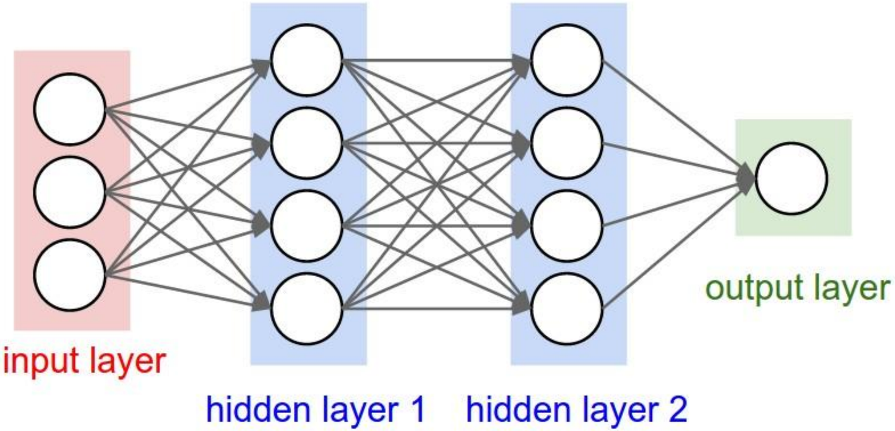
- Linear perceptron 1950s
  - Fell into the first nuclear winter when it was shown it can't model XOR function
- Resurgence in the 1980s-1990s - first digit recognizer - Yann LeCun
  - But computers were too slow to train models
  - Vanishing gradient problem - updates were just too slow, training took forever
  - Abandoned in favor of SVMs and RFs and GBMs in the late 90s, early 2000s



- Really gained prominence in 2012 - huge improvement in ImageNet accuracy
  - https://cdn-images-1.medium.com/max/800/1*Zz0iyMI4Ph1QRr2q8tFJzA.png
- Since then, many new ways to look the problems through the optic of DL
- Deep Learning is a subset of Machine Learning - some classical algorithms can be recast as 1-layer neural networks

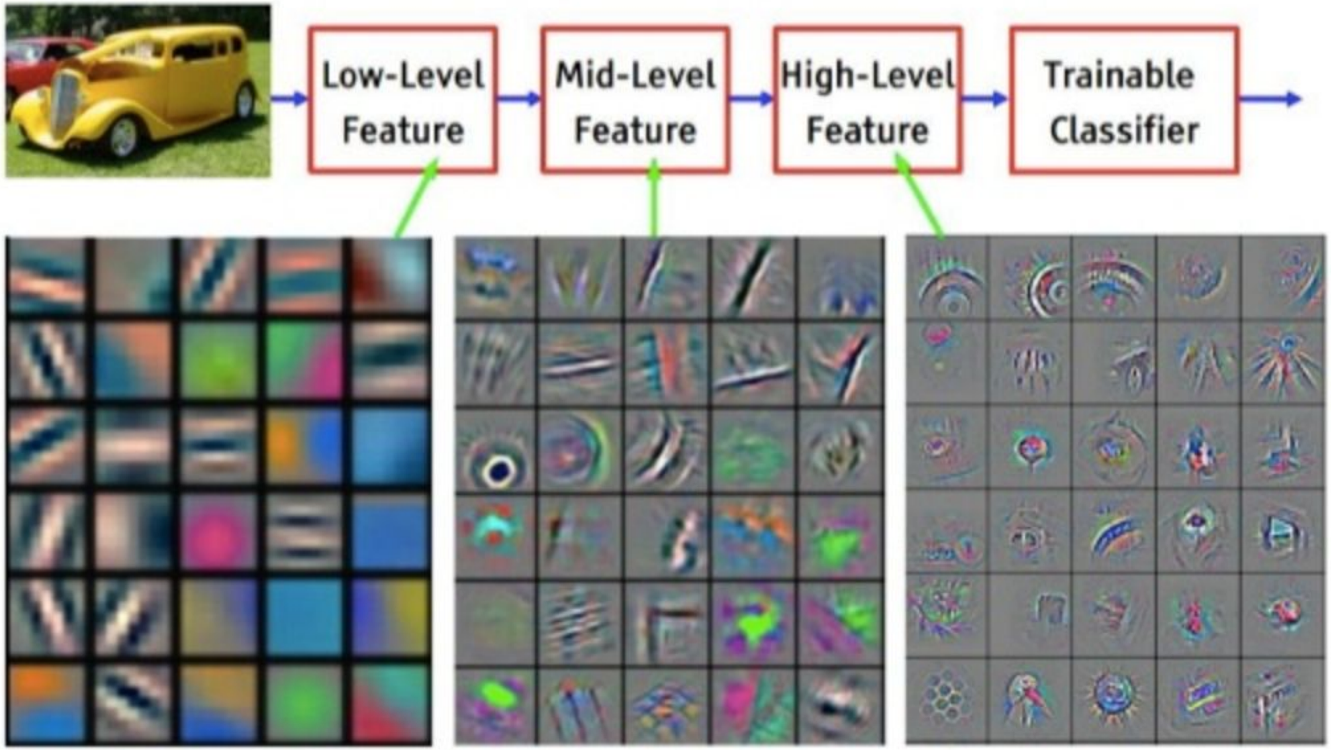
At its simplest:

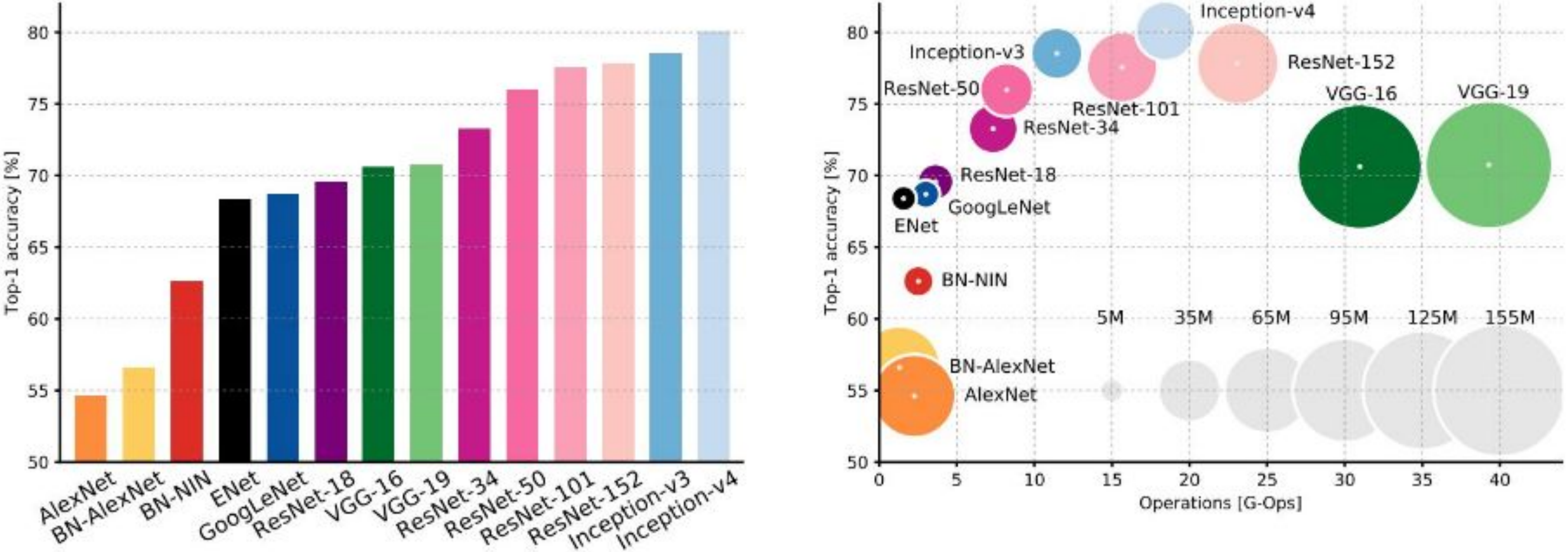




Problem: find the weights of the edges so that the error(predicted - actual) is minimum
- optimization problem, solved with gradient descent.
- many, many layers are used; commonly 16, 38, 50, 150 (but some up to a thousand layers)

Convolutional Neural Network





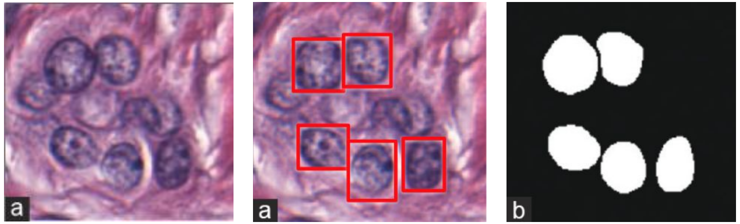
An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Applications

Vision

- Image Labeling - e.g. “does this image contain a blood cell”
- Object Detection - e.g. “give me a box in the picture where blood cell is”
- Image Segmentation - e.g. “identify all pixels where the blood cell is”

	Output
labeling	yes/no
detection	Set of “bounding boxes” - 4 numbers per box
segmentation	“Masked image”



Original bounding box mask

Natural Language Processing NLP

Old approach: bag of words, TF-IDF

New approach: word embedding; language model

Idea: represent each word with a vector of numbers. Similar words will be represented by similar vectors. (Dimensions of 50, 100, 200 are common)

Learn that representation as a part of your DL model!

Transfer learning

Idea: use pretrained models + data for your particular problem, to cut down on the number of examples you need in your data

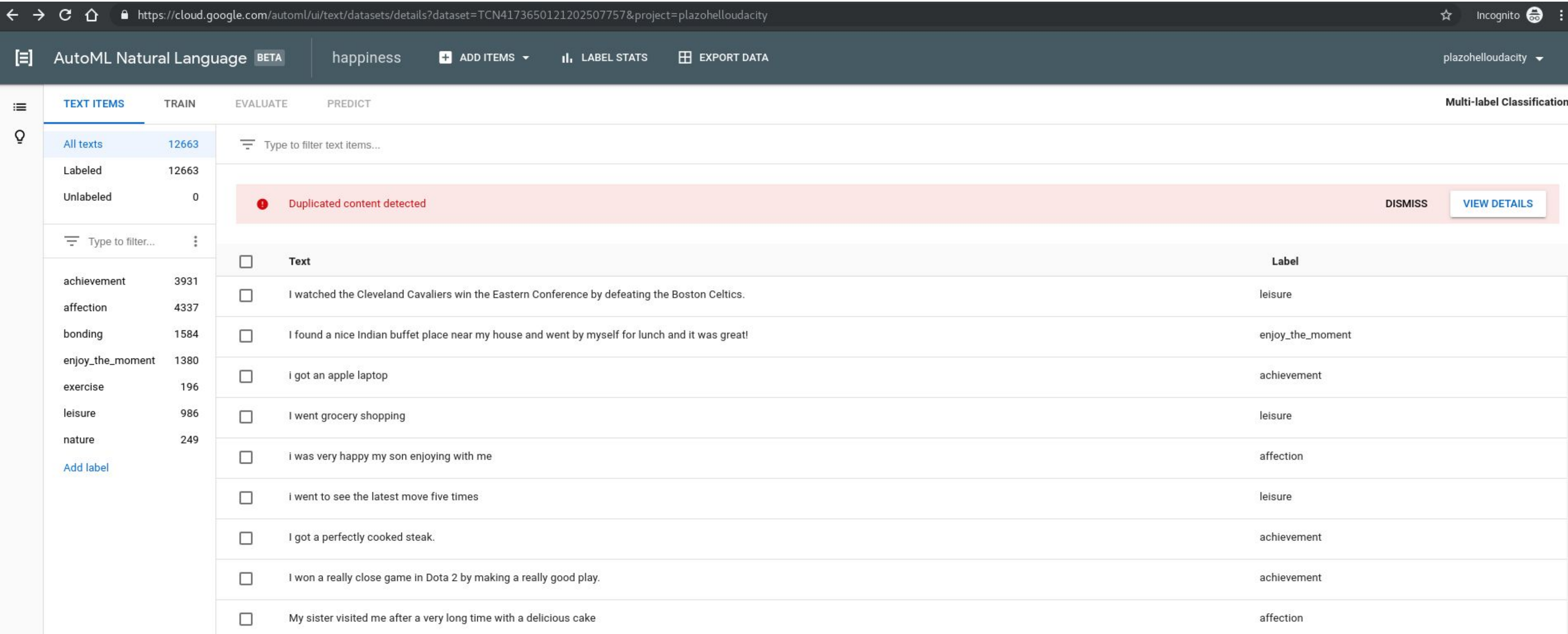
Example:

“Sentiment analysis” - i.e. if we had an understanding of the English language, it would be easy to train on your data because the parameters of the model are already close to what they need to be. You TRANSFER the problem from one domain to another.

Upshot: you pretrain on millions of images/documents, but fine-tune on thousands!

I.e. you cut down on the number of examples you need to label!

AutoML for NLP



TODO: concrete example of expected accuracies

Resources

Datasets: kaggle.com, data.gov, RU library, make your own!

Tutorials: kaggle.com (kernels), fast.ai, youtube

On campus help: graduate student specialist; OARC data science office hours

APIs: seaborn API, scikit documentation

Compute power: Amarel (free), Colab (free), GCP credits $300

Toolkits: scikit-learn (python), caret (R);

Resources (cont.)

On campus help:

OARC Data Science office hours - Mondays 3-4:30pm in CoRE 710

Graduate student specialist program - Library

Tutorials and courses:

https://www.fast.ai/ by Jeremy Howard - excellent practical courses - python

https://www.youtube.com/playlist?list=PLOg0ngHtcqbPTlZzRHA2ocQZqB1D_qZ5V "Introduction to Statistical Learning Series" by Hastie and Tibshirani (Stanford) - R

https://www.amazon.com/Introduction-Statistical-Learning-Applications-Statistics/dp/1461471370

Thank you!

Comments, questions, suggestions?

Please fill out the evaluation form

Future topics

- Feature engineering
- Feature importance - i.e. which features are the most important ones?
- Hypertuning - i.e. choosing the right parameters for the algorithms
- Categorical encodings - OHE or One-Hot Encoding
- Machine learning for time series
- Machine learning for geo data
- Visualization code

Work on your dataset

- Exchange information about your datasets
- Determine inputs and outputs for your problem
- Is your problem supervised or unsupervised?
- Are your variables categorical or numerical or time or geo?
- Do you have missing values? How are you going to impute them?
- Which variables do you think are going to be most important for prediction?
- Could you get more data?
- Could you get other features?
- Do you have to discard some variables?

concept	python	R
dot	property or function of an object	no special meaning - can be part of a name
help	`pd.read_csv?`	`?read.csv`
dataframe package	pandas	base
import package	`import pandas as pd`	no import necessary
dataframe	list of named columns	list of columns
define dataframe	`df = pd.DataFrame({})`	`df = data.frame(n, s, b)`
column "age"	`df.age` or `df['age']`	`df$age`
columns	`df.columns`	`names(df)`
show	`df.head()`	`head(df)`
read csv	`df = pd.read_csv(filename)`	`df = read.csv(filename)`
machine learning package	scikit-learn	caret
import ML package	`import sklearn`	`library(caret)`
install package	`pip install scikit-learn` in terminal	`install.packages("caret")` in R
list/vector	`x = ["a", "b", "c"]`	`x = c("a", "b", "c")`
index start	0	1
slicing	`x[1:3]` (2 elts)	`x[1:3]` (3 elts)