TU 257 – Fundamentals of Data Science

Data Analytics

L6– Tuning & AutoML

Brendan Tierney

Agenda

- Model Tuning
- Automating the process
- AutoML

Model Tuning

Previous Examples

- Our Previous Examples all used the default settings
- Each Algorithm has their own settings
- These parameters are often called Hyperparameters

- Lots of testing and Experiments have worked out the best settings to use.
- These work best for most cases/scenarios
- But may not work best for all cases/scenarios

Model Tuning

- Model Tuning is the process where you try to optimize the mode
    - By modifying the parameters
    - To give a better / more accurate model
    - To get better predictions on new data

- Why is this important
    - Minor changes can have a big impact
    - On € / $ Profit / Loss
    - Or reduce fraud / breakages / better health predictions, etc

- Experimentation is needed
- Evaluate the results to see if they are really useful

Model Tuning

- Some Algorithms have 10+ parameters
- Each parameter can have 10+, or 100+ possible values
- Search Space becomes huge

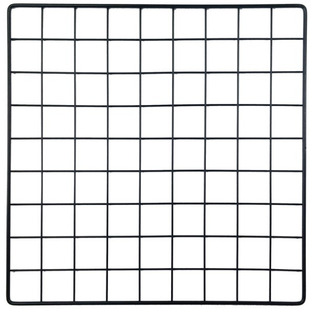
- Don't do it manually!
- Use in-built Functions to do this

- But it will take some time, maybe a long, long, long time

How to do this

- There are 2 main approaches
  - Random Grid Search – Randomly select values for parameters from list/range
  - Grid Search – Walks through all combinations
- These approaches can be used to find the best combination of Parameters and their Settings

- What's a Grid?
  - It's a List of Parameters and the Values to be included in the Search
  - The Values can be a List of values, or you can give a Range of values
  - Or some combination of these



```
#parameters with a list of values
a1: [0,1,2,3,4,5]
a2: [10,20,30,40,5,60]
a3: [105,105,110,115,120,125]
```

```
#parameters with list & range of values
a1: [0,1,2,3,4,5]
a2: list(range(10,60))    #all values between 10 & 60
a3: [105,105,110,115,120,125]
```

Random Grid Search

```
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV

param_grid = {
    'n_estimators': [25, 50, 100, 150],
    'max_features': ['sqrt', 'log2', None],
    'max_depth': [3, 6, 9],
    'max_leaf_nodes': [3, 6, 9],
}

#RandomizedSearchCV will select a Random selection of values for each parameter.
# This might not be suitable as it might miss important values

random_search = RandomizedSearchCV(RandomForestClassifier(),
                                   param_grid)

random_search.fit(X_train, y_train)

# random random search results
print('Best random search hyperparameters are: '+str(random_search.best_params_))
print('Best random search score is: '+str(random_search.best_score_))
```

```
  Best random search hyperparameters are: {'n_estimators': 25, 'max_leaf_nodes': 9,
'max_features': 'log2', 'max_depth': 6}
  Best random search score is: 0.8438924650439015
```

Check out this webpage for more RandomizedSearchCV details
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html

Grid Search

```
rfc = RandomForestClassifier()

#GridSearch can take a lot of time!   We will only use these 2 parameters as an example
forest_params = [{'max_depth': list(range(2, 6)),
                  'max_features': list(range(3, 8))}]

grid_search = GridSearchCV(rfc, forest_params, cv = 10, scoring='accuracy')

#this next command will take some time!
grid_search.fit(X_train, y_train)

GridSearchCV(cv=10, estimator=RandomForestClassifier(),
            param_grid=forest_params,  scoring='accuracy')


print('Best hyperparameters are: '+str(grid_search.best_params_))
print('Best score is: '+str(grid_search.best_score_))

  Best hyperparameters are: {'max_depth': 5, 'max_features': 6}
  Best score is: 0.853106644958161
```

How does this compare to RandomGrid Search?

Can you explain the difference?

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

Automating the Process

Why Automate

- To make you life easier
- To make the job easier
- Allows you to concentrate on the important things -> the Business Problem
- No run like boring, repetitive tasks
- Avoid mistakes due to boring, repetitive tasks
- Things can go wrong where there is so many different tasks and dependencies between these

How do we automate

- Identify what do we need to do every time
- Can we Automate it in some way
  - Writing code is a way to do
  - Creating a Notebook with all steps
  - Re-run the Notebook – when we have new data

- Can we really Automate every step?
  - Should we automate
  - Some legal requirements – See topic later in the semester
  - Human oversight is vital

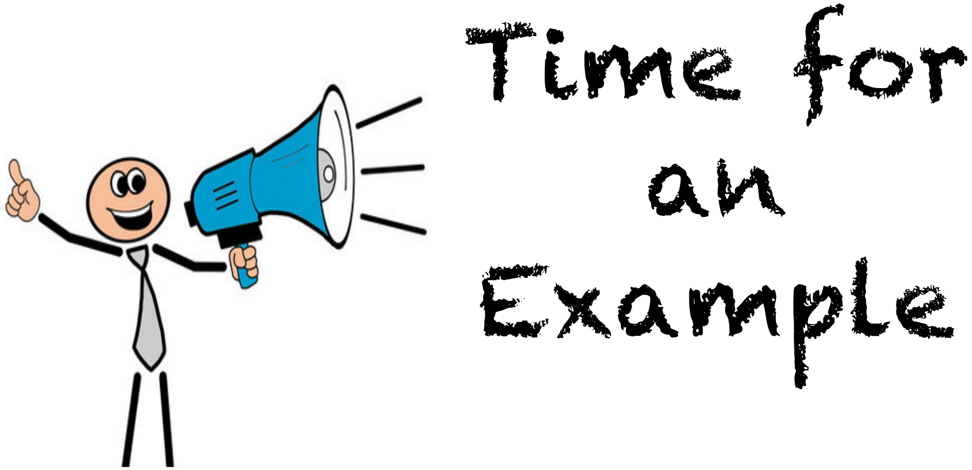
- What happens when the automation goes wrong?

How do we automate

- Document your code
- Document decisions
- Document outcomes
- Document edge cases
- Etc

- Create loops
- Integrate Charts
- Integrate Results
- Format the Outputs
- Make it easier to following and to understand

- How hands free can you be
- Create time to focus on Business Problem



AutoML

Automate the Boring Stuff

- We have seen examples of Automation before
  - Data Exploration
  - Graphs for Data
  - Data Preparation
- They are useful up to a point

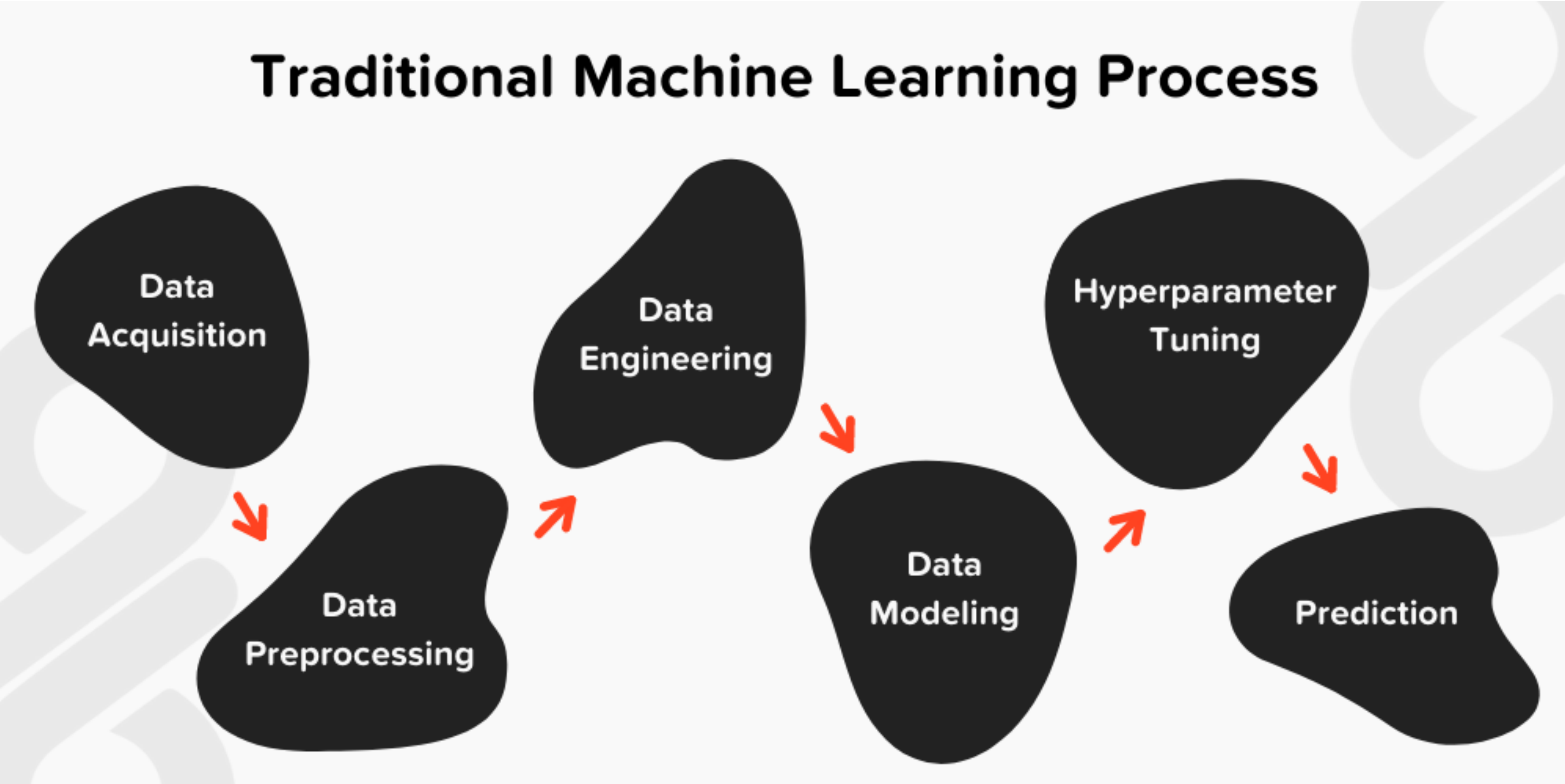
- AutoML -> Automate Machine Learning
- Was very popular "buzz" word over past few years

- Can help to guide the Analytics – but doesn't give some magic answer
  - It can give the wrong result -> just like ChatGPT

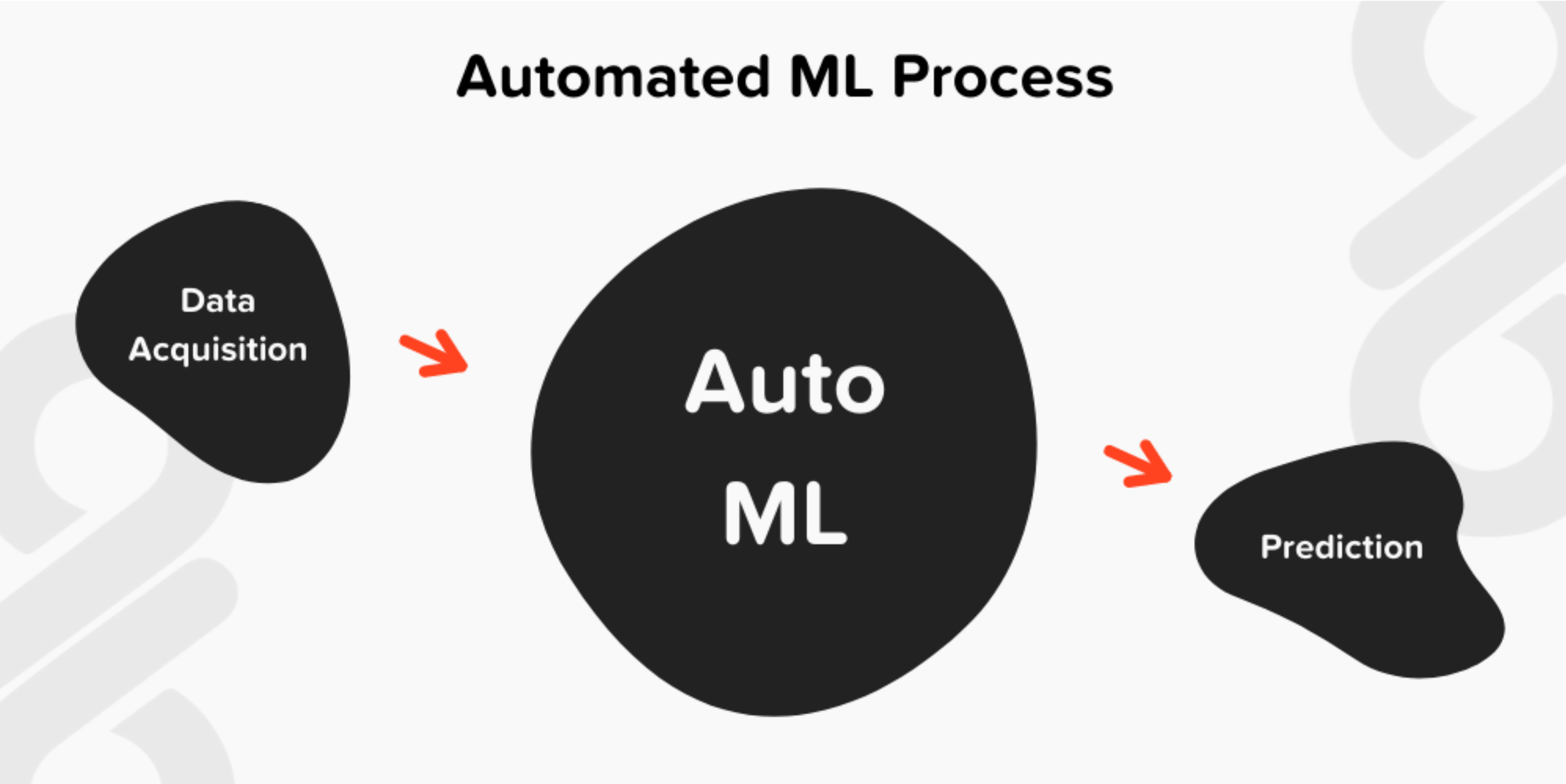
Pros vs Cons of AutoML

- **Pros**
  - Reduce the time it takes to implement traditional ML models
  - Reduce human effort by automatically running repetitive tasks
  - Reduce human errors
  - Save a lot of GPU and CPU processing, resulting in cost and power efficiency
  - Anyone without ML knowledge can enjoy the benefits of ML features
  - Opens doors for new opportunities to create a platform to provide AutoML apps for easier access to machine learning
- **Cons**
  - Human intelligence is neglected in complex problems, which can be more efficient than autoML
  - More emphasis on research and automating everything can lead to fewer jobs for data scientists
  - ML makes some decisions, like feature engineering, on the basis of domain knowledge which is lacking in the automation process
  - AutoML only focuses on supervised tasks that require labelled data as input and overlooks the more challenging tasks of unsupervised and reinforcement learning.

Traditional ML



AutoML



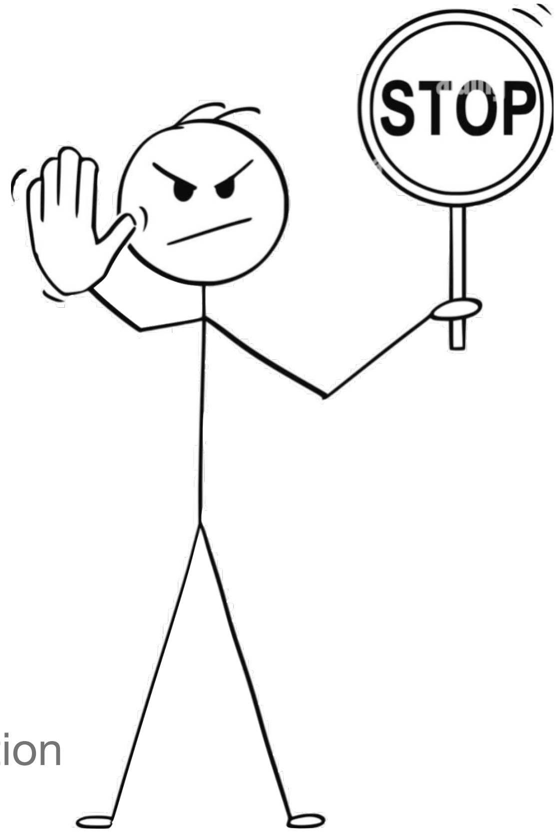
AutoML - Limitation

- It doesn’t work for all types of Algorithms or Problems

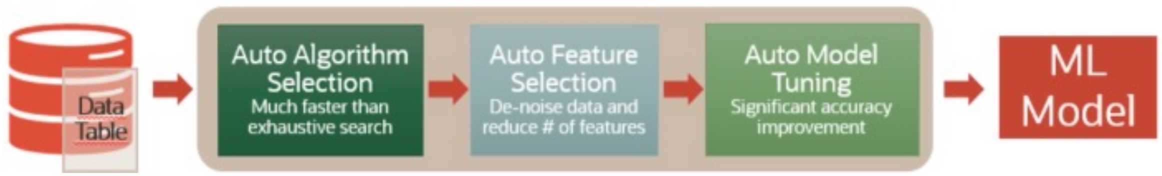
- Typically, suited to Classification
  - Yes/No
  - 1/0
  - Multi-Class e.g. 1, 2, 3, 4

- Some can do Regression

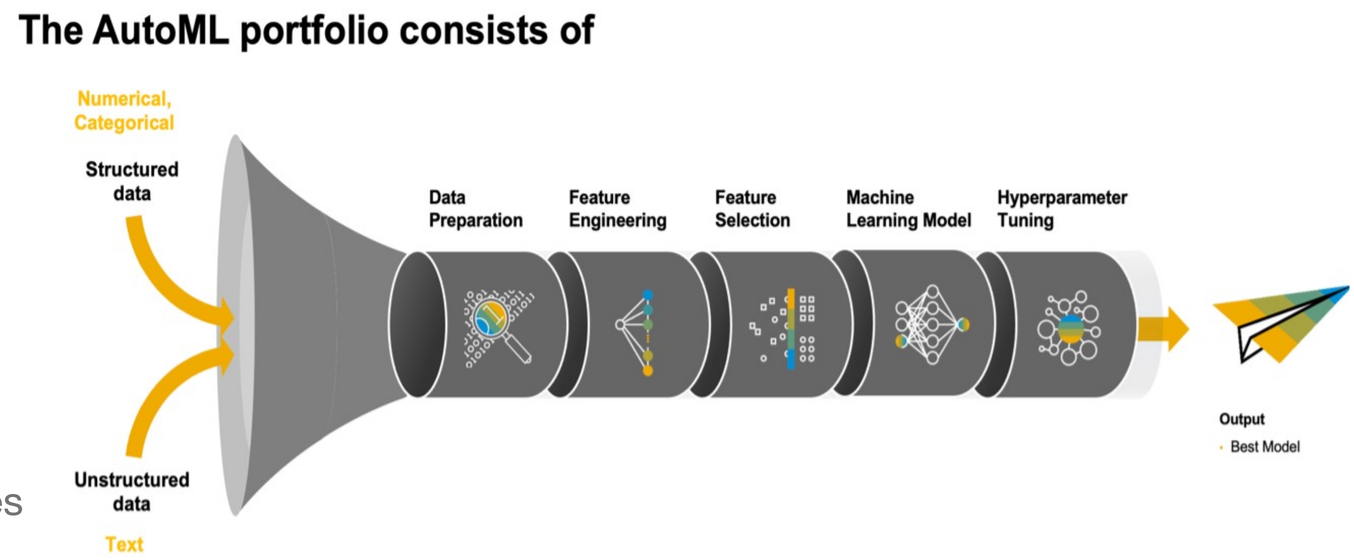
- Not much else -> But a larger percentage of problems are Classification



What does AutoML do



- Iterates through the process
  - Data Preparation
  - Feature Engineering
  - Feature Selection
  - Machine Learning
  - Tuning
- Output is “Optimal” model
  - Usually based on accuracy scores



- Meta Learning is used to iterate back over these steps to improve the results
  - Different Feature subsets selected
  - Selects appropriate Algorithms
  - Keeps iterating -> for a defined time, or a number of iterations or ….

Problems with using AutoML

- Cannot fix for bad Business Problem
- Cannot fix bad/poor Data Quality
- Does not explain WHY things have changed, etc

- Rubbish in = Rubbish out

- Human Oversight is needed
- No or Limited Model Explainability
- Legal Implications
- Reinforce Data Biases

- But could give you a bit of a guide for you to do Manually -> Human Oversight

- It can be Slow -> But it's doing lots of work -> It would be slower to write all the code yourself
    - This isn't a bad thing – Just it isn't a magic solution

Lots of AutoML soluctions

- AutoWEKA
- Auto-sklearn
- Auto-PyTorch
- AutoGluon
- H2O AutoML
- MLBoX
- TPOT
- TransmogrifAI
- Amazon Lex
- AutoKeras
- Data Robot
- BigML AutoML
- Google Cloud AutoML
- Auto-WEKA

Plus lots, lots more

Some Blog Posts

- AutoML, what is it good for? It Depends!
- AutoML – using TPOT
- AutoML – using autosklearn in Python
- AutoML using Pycaret
- OML4Py – AutoML – Step-by-Step Approach

See Installation Tip on next slide

AutoML install/setup

- This can be a little challenging in Anaconda
- Some of these AutoML libraries need specific versions of other libraries
  - These might not be what you have installed!

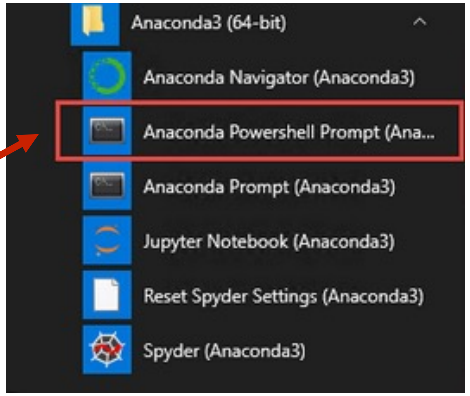
- Create a new Anaconda Virtual Environment
  - Install the AutoML into it
  - Here are some blog posts illustrating this
    - [Installing PyCaret in Anaconda]()
    - [Pycaret Installation Documentation]()

Similar needed for autosklearn

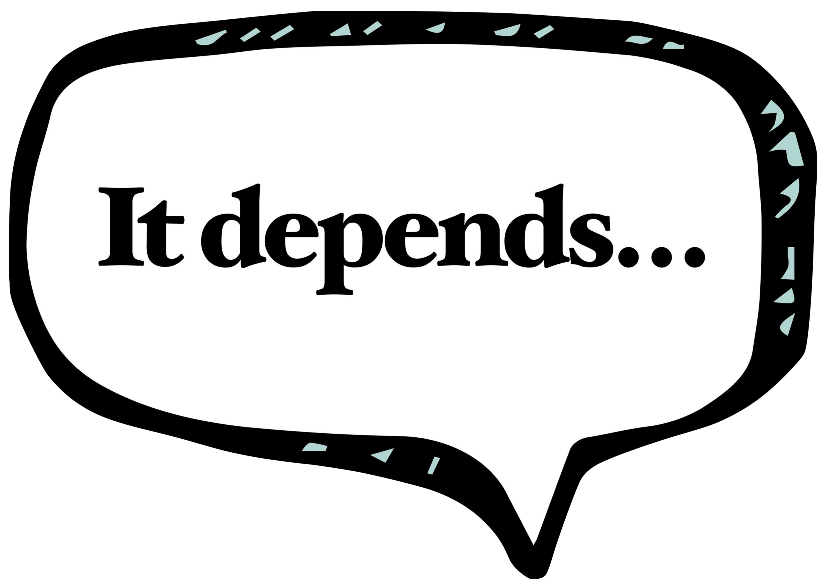
- Although some might work in your current Anaconda environment

tpot - It if isn't listed in available list of libraries to install, run the following

```
conda install -c conda-forge tpot
```




EASY
HARD
EASY
HARD


It depends…

Time for an Example

Any Questions ?

What Now/Next ?