COMP 540
Kaggle Advice

Please do not

1. Do nothing until the last month
2. Blindly put the data into some scikit-learn/keras models
3. Do some cursory hyperparameter tuning
4. Take the average of your predictions and call it a day

Also do not

1. Use code from the Kaggle kernels to get a semi-decent result
2. Pretend it is brilliant

The grading rubric will reflect that

- It isn’t worth your time as a student to throw together a mess of code and default setting models and stop when your accuracy looks “high enough”
- You need to spend a significant amount of time looking at your data

Recommended approach (% overall time/effort)

1. Look at your data. A lot. (30%)
2. Develop a workflow. (10%)
3. Try a bajillion models/features. (55%)
4. Investigate your final model (5%)

Looking at your data: surprising difficult (30%)

Start with lots of exploratory data analysis

- How much data is there? What does the average observation look like? How much variability is there?
- Are the training and test sets different in important ways?
- Are there outliers in the training data?
- Are there clusters in the data?
- Which features seem most predictive? Any guesses as to why?

Repeat for each different representation of the training data. Answer with pictures rather than numbers wherever possible.

Develop a workflow (10%)

Enable fast iteration

- Think about the computational infrastructure needed to train, test and compare many models
- Find examples of good hyperparameter tuning. Grid search is a low bar to clear
- Your work isn’t required to be fully reproducible but if it isn’t I will judge you
- Use version control
- Document your models and log their performance in a central, standard format

In particular: AWS / GPUs

- AWS is expensive
- Figure out how to prototype on your laptop and then train on the cloud
- At some point you’re going to want access to a GPU

Options:

- Jupyter Notebooks running on AWS
- Python scripts transferred via Git
- Etc

Note: 40% of energy not spent on modelling

Okay, now for the modelling part (55%)

Baselines

- For each feature representation, try all the standard models: penalized GLMs, random forests, GBMs, SVMs, a variety of kernelized methods, KNN, Naive Bayes, neural networks, etc
- Do good hyperparameter tuning
- These models form your baseline: your next goal is to beat them
- If you save the trained models, you can use them later in ensembles
- Investigate where these models do well and where they fail

Consider using autoML libraries or automatic hyperparameter tuning

Comparing Models

- Use consistent resampling across all your models to make comparisons as direct as possible
- Consider a holdout “blending” set for later ensembling
- Strongly consider using bayesian ROPE (region of practical equivalence) to compare models
- Never test a model on data it has seen before

Ensembles

- Taking an average of predictions is naive
- Try blending or stacking
- More models in = better predictions out

Feature Representations

- Feature representations (or the representations learned by a neural network) are going to make or break the competition this year
- Computer vision is dominated by neural nets, but that doesn't mean you should only use neural nets.
- Try a ton of architectures
- Consider extracting features from neural nets and using standard ML models on these extracted features

Finally: investigate the FrankenModel (5%)

FrankenModel + Medical Dataset = Bad

- You need to justify your final model
  - Good prediction is great
  - Demonstrations that the model is learning good features or that the model is interpretable are also valuable
- Use LIME (Locally Interpretable Model-Agnostic Explanations) to investigate predictions on average and outlier observations
- Use pictures to convince people your model is good

Remember: If a student does work but doesn’t document or explain it, did they actually do any work?