Feature Selection
and
Feature Engineering

And why it Matters

Who am I?

- Dirk Biesinger linkedin.com/in/dirkbiesinger/ @Saravji
- Interested in AI and intelligence in general
- Lived and worked in Germany, Canada, USA
- Industries: Technology, Aerospace, Oil and Gas, Banking, Food, Manufacturing, Telecommunications, Consulting
- Sr. Data Scientist Consultant at Unify Consulting, focusing on ML
- teaching at UW Continuum College (Data Science and Python)
- creating educational materials:

saravji.github.io/saravjis_hut   youtube.com/channel/UCbHQVBE_i-4FlJUcU7Inl-g

I literally grew up in a machine shop. Consequentially, I developed a curiosity re optimization of workflows and minimizing of waste.

My journey ultimately took me into aerospace, where I formalized this passion by acquiring a Lean Six Sigma Black Belt Certification.

After transitioning into Data Science, this passion did not change and naturally, I was curious about:

Where is the time best spend when working on a typical Data Science project or Machine Learning pipeline?

Consequently, I spend about 200 hours exploring this question.

This Is My Answer:

Feature Selection
and
Feature Engineering

Matter.

Let me show why.

Don’t trust me – trust these:

The algorithms we used are very standard for Kagglers. […] We spent most of our efforts in feature engineering.

— Xavier Conort, on “Q&A with Xavier Conort” on winning the Flight Quest challenge on Kaggle

Actually the success of all Machine Learning algorithms depends on how you present the data.

— Mohammad Pezeshki, answer to “What are some general tips on feature selection and engineering that every data scientist should know?”

Basically, xgboost or lightgbm and Neural Nets are the algorithms used in Kaggle competitions. Feature Selection and Creative Feature Engineering decides which Teams will win.

— Anthony Goldbloom (CEO Kaggle), during a conversation with me.

you have to turn your inputs into things the algorithm can understand

— Shayne Miel, answer to “What is the intuitive explanation of feature engineering in machine learning?”

…some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used.

— Pedro Domingos, in “A Few Useful Things to Know about Machine Learning”

Syllabus

- Welcome
- Links
- Methodology used
- Data Set used
- Feature Selection
- Feature Engineering

Links

- This Presentation and related Materials: saravji.github.io/saravjis_hut/DS_ML/FS_FE/
- Saravji’s Hut all schedules: saravji.github.io/saravjis_hut
- Saravji’s Hut YouTube channel: youtube.com/channel/UCbHQVBE_i-4FIJUcU7Inl-g
- boruta feature selection method: Boruta all-relevant feature selection method
- boruta_py implementation (Daniel Homola): Related blog post
- boruta_py conda package: anaconda.org/conda-forge/boruta_py
- MADELON Dataset: archive.ics.uci.edu/ml/datasets/madelon

Introducing the Methodology used:

- To evaluate the contribution of individual components, each needs to be assessed individually.
- To create a neutral picture, I selected randomly one ML algorithm out of the ML groupings in scikit-learn. Ref.
- Unless tuning of algorithms is the assessment, the default settings are used.
- The exception is setting the random seed to ensure repeatability (and make the results comparable)

Data set used: MADELON

The data set used for the demonstrations is synthetically created by Isabelle Guyon (Prof. for DS @ Université Paris-Saclay; inventor of SVM-RFE) for the purpose of demonstrating feature selection and feature engineering.

It consists of 2000 training observations, 500 features with integers in the range 0…1000 and a target of -1 or 1 (equally distributed).

Further, 600 test observations in the same manner.

Details: http://archive.ics.uci.edu/ml/datasets/madelon

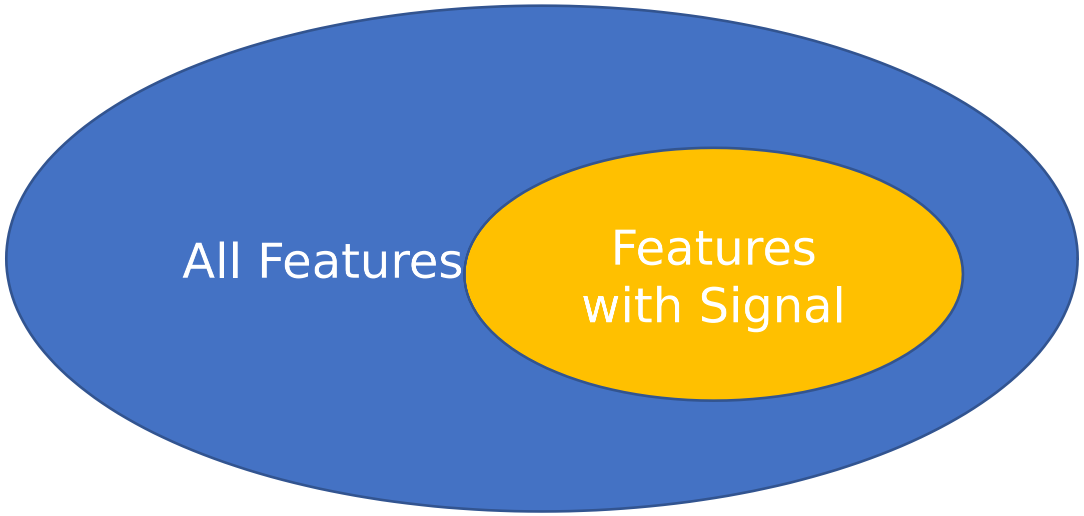
Feature Selection
and
Feature Engineering

And why it Matters

Feature Selection. Why?

- The Intuition behind feature selection is to only provide features supporting the signal to an ML algorithm and suppress the features adding noise.
- Feature Selection is fundamentally different from feature elimination and dimensionality reduction:
  - Feature elimination eliminates the feature that adds the least value while examining against the remainder of features.
  - Dimensionality reduction does not eliminate any data, but alters the composition of the data in a mathematical way. The original data can be reconstructed out of the result.
  - Feature Selection selects the features that add value.

Feature Selection. Why?



Feature Selection. Is it necessary?

- Would a ML algorithm not disregard the unimportant features no matter what?
  - No. The intuition behind each ML algorithm is to fit to the data presented. This includes all the noise presented as well. The result is a model fitting to the data plus noise presented during training. Provisions in the algorithm can only limit the adverse effects to a degree, but not eliminate them.
  - Especially vulnerable are boosted algorithms, as these can’t learn basic and learn-able combinations of weak hypotheses when presented with random noise.
- Can tuning, cross-validation, train-test-validate split compensate for this?
  - To a certain extend. But it is computationally expensive to achieve small gains.

Feature Selection. How?

- One method is to add one feature out of all not included features at a time and evaluate which one results in the biggest improvement. Repeat until no further improvement can be achieved. (Forward Selection)
- One method is to subtract one feature out of all remaining features at a time and evaluate which one results in no or the least reduction in the result. Repeat until no feature can be removed without effecting the results. (Backward Elimination)
- Both these approaches have the negative effect of being effected by the actual data (over-fitting), the sequence (of adding / subtracting features), the evaluation method used and the remainder of features.

Feature Selection. Then How?

- One method evaluates each feature against itself:
  - The feature to be evaluated is being copied as additional (shadow) feature into the same data set.
  - The sequence of the values in this feature gets randomized.
  - The importance of the original feature and the shadow feature gets evaluated after several model runs. If the shadow feature has a higher or equal importance then the original feature, the original feature does not add value.
  - Repeat for each feature.

Feature Selection. Demo Notebook

Feature Selection. The Results:

Using MCC as evaluation metric

- Naïve approach: (using all data and default models): Best: 0.51 in 45 sec.
- Brute Force: (all data, parameter tuning in models): Best 0.72 in 99 hrs. (4 days, 3hrs.)
- Univariant Feature Selection: (selective data and default models): Best 0.63, in < 10 sec.
- Recursive Feature Elimination: (reducing data and default models): Best 0.61, in < 15 sec.
- Dimensionality Reduction: (modified data and default models): Best 0.55, in < 10 sec.
- Selecting important features: (selective data and defaults models): Best 0.60, in < 10 sec.
- Feature Selection: (only selection of data and default models): Best 0.77, in < 2 min.

Note: only deviation from default model is setting a random state for reproducibility and fair comparison

Feature Selection and Feature Engineering

And why it Matters

Feature Engineering. Why?

- Equalizing the signal strength between features to establish equal importance between features.
- Amplifying signal in features
- Abstracting timeless meaning to allow application of past configurations to current configurations
- Creating additional amplitude or distinction out of existing features
- Creating additional dimensionality
- Reducing dimensionality

Feature Engineering. Is it necessary?

More often then not, the data has features that have implicit meaning easily comprehensible to humans. But: ML algorithms don’t have these capabilities (yet).

It is important to complement the implicit data with these explicit features to ENABLE the ML algorithms to utilize this information.

Feature Engineering is a vast field only limited by imagination. There are innumerable possibilities.

The following slides will be limited to a few basic approaches and ideas that can be used as starting points.

Feature Engineering. Is it necessary?

- If an important features is not presented, it can’t be fitted.
- If one feature dwarfs other features with its numbers, it may dominate.
- If the signal is barely noticeable, it might not get picked up.
- If the meaning is hidden within data presented, but not extracted, it will not get picked up.

Feature Engineering. How?

Intuitive Approaches to Feature Engineering:

- Normalizing or standardizing features to eliminate range imbalances. Example: house price forecast (number bedrooms vs. sq.ft. house)
- Abstracting features to counter drift over time. Example: Harddrives / SSD in laptops, stock prices after stock split
- Amplifying values to pronounce importance of a signal. Example: analyzing faint radio signals or background noise in audio / visual data.
- Creating features out of existing data. Example: Creating Day-of-Week out of date for weekly pattern
- Creating additional dimensionality. Example: volume out of three dimensions
- Reducing dimensionality. Example: grouping data by region or grouping languages by family.

Feature Engineering. How?

Creative Approaches to Feature Engineering:

- Combining existing numerical features mathematically
- Chaining existing categorical features
- Grouping categories in dependence of available observations and additional dimensionality
- Creating categories based on statistically inferred threshold values in available data
- Combining categorical features to create a matrix of occurring combinations
- Use ML algorithm output based on a sub-selection of features as additional feature for a second stage (be wary of data leakage)

Complimentary Approaches to Feature Engineering:

- Add external data

Feature Engineering. Demo Notebook:

Feature Engineering. Results:

Using MCC as evaluation metric and selected Features

- Baseline: (default models): Best: 0.763 in < 2 min.
- Deep Dive: (modified feature selection, default models): Best 0.793 in < 10 sec.
- Standard Scaler: (modified feature selection, default models): Best 0.860, in < 10 sec.
- MinMax Scaler: (modified feature selection, default models): Best 0.834, in < 10 sec.
- PCA: (modified feature selection, default models): Best 0.777, in < 10 sec.
- Std Scaler plus PCA: (modified feature selection, default models): Best 0.857, in < 10 sec.
- MinMax Sclr pls PCA: (modified feature selection, default models): Best 0.824, in < 10 sec.
- Std Sclr, PCA, Std Sclr: (modified feature selection, tuned models): Best 0.863, in < 15 sec.

The results of 0.86 and 0.863 mean: 43 and 41 incorrectly classified observations or an error rate of 7.167% and 6.83% respectively (relative to the target which has deliberately flipped labels)

Note: only deviation from default model is setting a random state for reproducibility and fair comparison

Feature Engineering: Remarks

- The Boruta package utilized for Feature Selection is a convenient shortcut. The paper it is based on has a few shortcomings which result in inconsistent - and depending on the data set unintentional - results. The implementation does not follow the paper and is at best a rough approximation on some of the core principles. Use cautiously and verify results.
- As demonstrated, a better feature selection is available.
- As demonstrated, Feature Selection and Feature Engineering outperform algorithm tuning by a wide margin, especially when combined.
- On this data set, it is hard and taxed with significant effort to get two (2) observations more correctly classified.

Overall Results:

Using MCC as evaluation metric

- Naïve approach: (using all data and default models):
  Best: 0.51 in 45 sec.
- Brute Force: (all data, parameter tuning in models):
  Best 0.72 in 99 hrs. (4 days, 3 hrs.)
- Feature Selection: (default models):
  Best: 0.763 in < 2 min.
- Deep Dive: (modified feature selection, default models):
  Best 0.793 in < 10 sec.
- Standard Scaler: (modified feature selection, default models):
  Best 0.860, in < 10 sec.

Significant effort required to tune the parameters:

- Std Sclr, PCA, Std Sclr: (modified feature selection, tuned models):
  Best 0.863, in < 15 sec.

Feature Selection
and
Feature Engineering

It Matters.

Thank You.