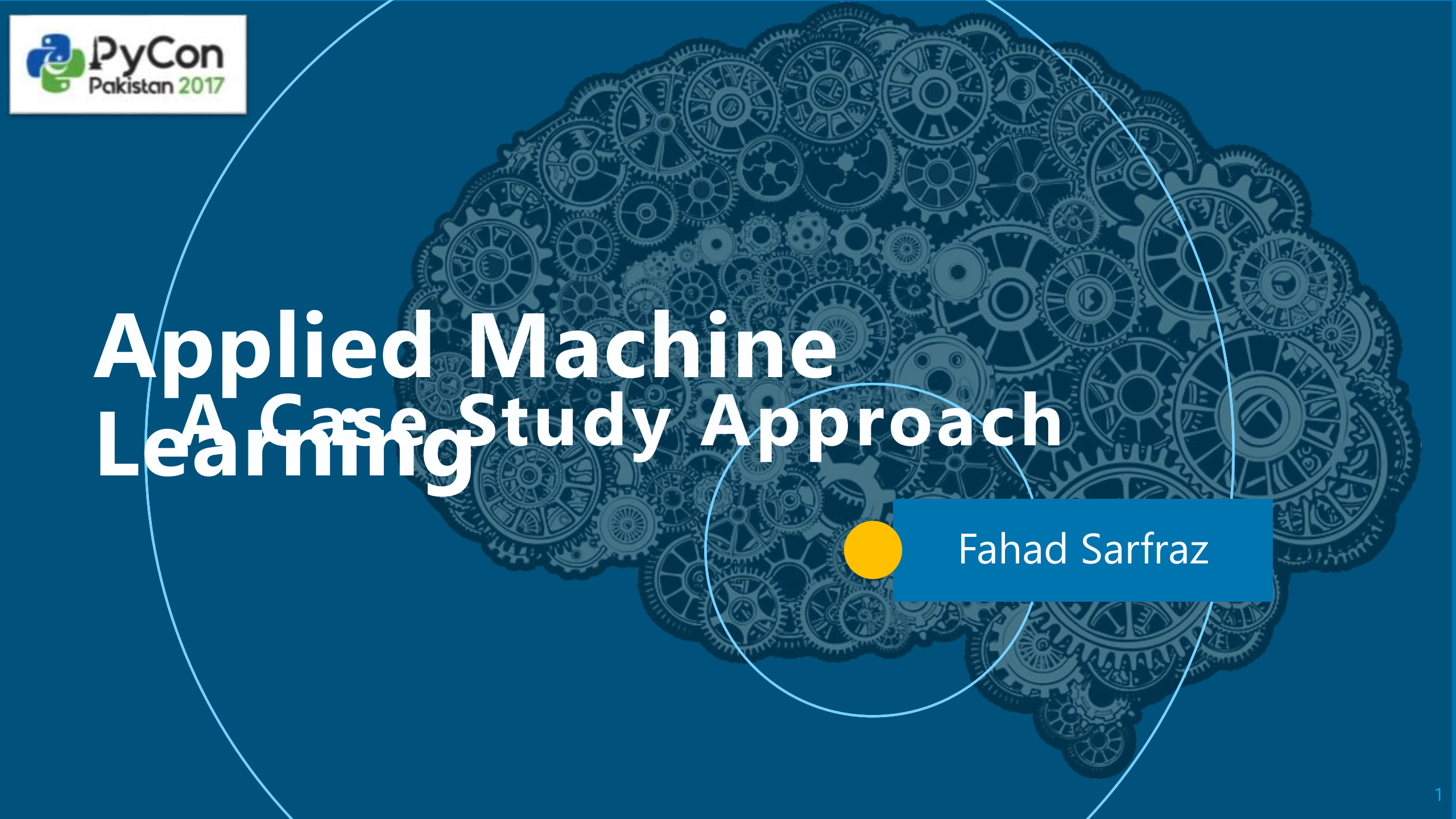


Applied Machine Learning
A Case Study Approach

Fahad Sarfraz

INTRODUCTION

Fahad Sarfraz

Software Engineer at Arbisoft, Machine Learning Department

A Machine Learning Engineer and a Data enthusiast with professional experience in employing machine learning algorithms and tools for image classification, text analytics and financial data prediction

Contact:

Email id: fahad.sarfraz@outlook.com | **Tel: +92-341-4400551**

LinkedIn: https://www.linkedin.com/in/fahadsarfraz/ | **Github:** https://github.com/fahad92virgo/

Applied vs Theoretical Machine Learning

Applied Machine Learning is about understanding the Machine Learning concepts at an abstract level sufficient enough to solve problems using machine learning. This involves gaining expertise in using the tools and libraries which implement the Machine Learning Algorithms at their core.

Theoretical Machine Learning on the other hand is about understanding the underlying algorithms, mathematics, probability theory, statistics and definitely a lot of other subjects/concepts at the fundamental level.

Advice: Aim to strike a balance between the two.

Applied vs Theoretical Machine Learning cont..

Theoretical Machine Learning

Applied Machine Learning

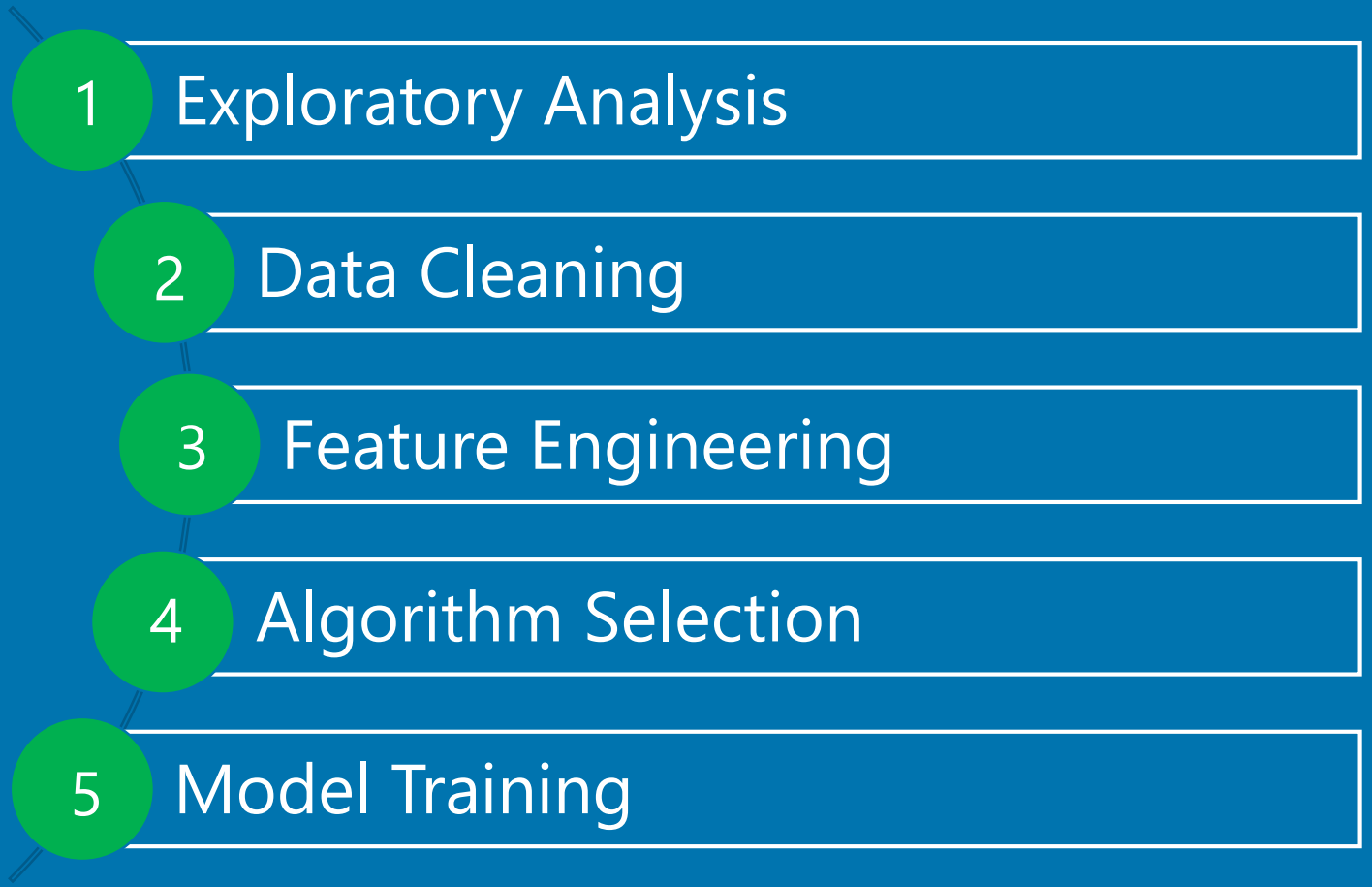
- Applied machine learning is about solving real world problems.
- This is where the potential and impact of new inventions/discoveries made through advancements in theoretical machine learning are realized.

Why the Case Study Approach

- Build an intuitive sense of the core concepts
- Gain a strong understanding of the machine learning pipeline
- Get motivation to dive deeper
- Solve real world problems

Case Study: Predicting House Prices

Core Steps


1
Exploratory Analysis
2
Data Cleaning
3
Feature Engineering
4
Algorithm Selection
5
Model Training

Exploratory Analysis

The purpose of exploratory analysis is to "get to know" the dataset.

Objective:

- gain valuable hints for Data Cleaning
- think of ideas for Feature Engineering
- get a "feel" for the dataset

Guidelines:

- Distributions of numeric features: plot distributions to identify outliers, binary features, potential measurement error
- Distributions of categorical features: Use bar charts to look out for sparse classes
- Segmentations: use box plots to observe the relationship between categorical features and numeric features
- Correlations: look at the relationships between numeric features and other numeric features.

Data Cleaning

Key Takeaway: '*Better data beats fancier algorithms.*'

- **Unwanted observations:** remove duplicates or irrelevant observations
- **Structural errors:** fix typos, inconsistent capitalization, mislabeled classes,
- **Outliers:** Outliers can cause problems with certain types of models. you must have a good reason for removing an outlier, such as suspicious measurements that are unlikely to be real data.
- **Missing data:** you cannot simply ignore missing values in your dataset. You must handle them in some way for the very practical reason that most algorithms do not accept missing values
  - Dropping missing values is sub-optimal because when you drop observations, you **drop information.**
  - Imputing values is suboptimal because you are not adding any real information. You're just reinforcing the patterns already provided by other features.

  Always tell your algorithm that a value was missing because missingness is informative
  - for *categorical* features is to simply label them as 'Missing'!
  - For missing *numeric* data, you should **flag and fill** the values.

Feature Engineering

"**Applied machine learning**" is basically **feature engineering**." - Andrew Ng

Feature engineering is about **creating new input features** from our existing ones.

Feature Engineering allows us to:

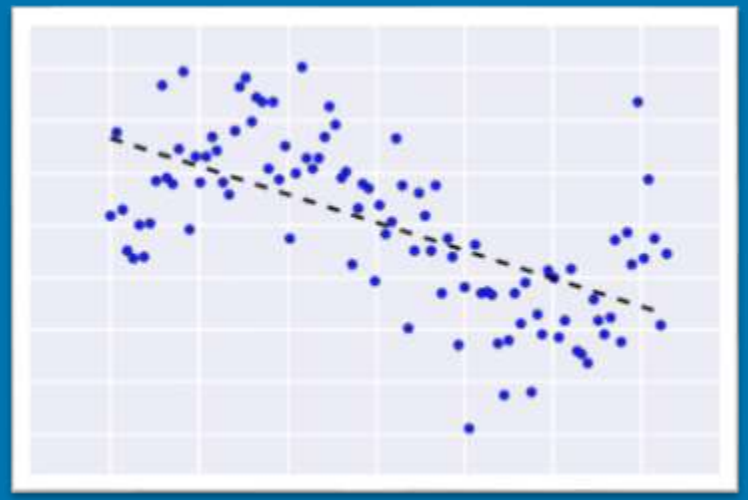
- isolate and highlight key information, which helps your algorithms "focus" on what's important.
- bring in our own **domain expertise**.

Guidelines:

- **Interaction features:** combinations of two or more features.
- **Sparse Classes:** group sparse classes together that have very few total observations.
- **Dummy Variables:** Most machine learning algorithms cannot directly handle categorical features.
- remove unused or redundant features from the dataset.

Algorithm Selection

- **Linear Regression**



Simple linear regression models fit a "straight line", or more precisely a a *hyperplane* depending on the number of features.

Their main advantage is that they are easy to interpret and understand

However, simple linear regression suffers from two major flaws:

- It's prone to overfit with many input features.
- It cannot easily express non-linear relationships.

Algorithm Selection cont..

Regularization:

Regularization is a technique used to prevent overfitting by *artificially penalizing model coefficients*

- It can discourage large coefficients (by dampening them).
- It can also remove features entirely (by setting their coefficients to 0).
- The "strength" of the penalty is tunable

Lasso Regression

- Penalizes the absolute size of coefficients
- Practically this leads to coefficients that can be exactly 0
- Offers automatic feature selection because it can completely remove some features

Ridge Regression

- Penalizes the squared size of coefficients
- Practically this leads to smaller coefficients but it doesn't force them to 0
- Offers feature shrinkage

Model Training

Split dataset: **Training sets** are used to fit and tune your models. **Test sets** are put aside as "unseen" data to evaluate your models

Comparing test vs. training performance allows us to avoid overfitting... **If the model performs very well on the training data but poorly on the test data, then it's overfit.**

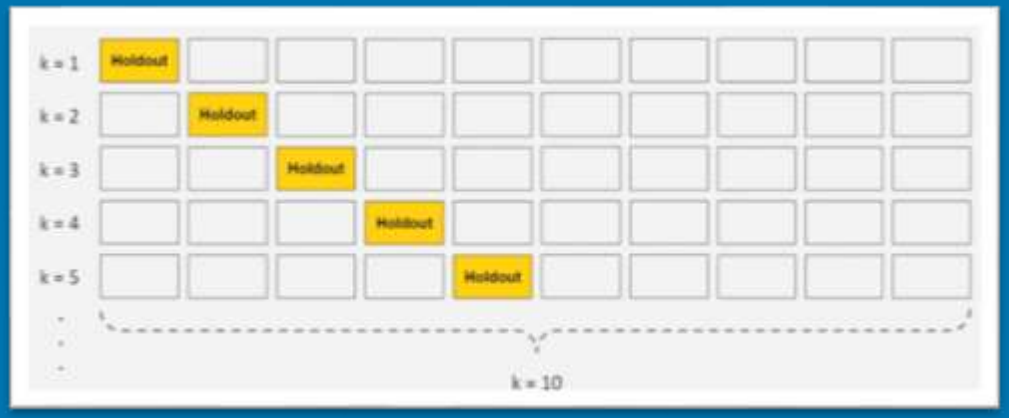
Parameter Tuning: There are two types of parameters in machine learning algorithms.

- **Model parameters** are learned attributes that define *individual models*. They can be **learned directly** from the training data
- **Hyperparameters** express "higher-level" *structural settings* for algorithms. They are **decided** before fitting the model because they can't be learned from the data

Model Training cont.

Cross-validation

- Cross-validation is a method for getting a reliable estimate of model performance using only our training data.
- There are several ways to cross-validate. The most common one, **10-fold cross-validation**, breaks your training data into 10 equal parts (a.k.a. folds), essentially creating 10 miniature train/test splits.



Steps for 10-fold cross-validation:

1. Split your data into 10 equal parts, or "folds".
2. Train your model on 9 folds (e.g. the first 9 folds).
3. Evaluate it on the 1 remaining "hold-out" fold.
4. Perform steps (2) and (3) 10 times, each time holding out a different fold.
5. Average the performance across all 10 hold-out folds

Model Training cont.

Fit and tune models

- perform the entire cross-validation loop detailed above on each **set of hyperparameter values** we'd like

```
For each algorithm (i.e. regularized regression, random forest, etc.):
  For each set of hyperparameter values to try:
    Perform cross-validation using the training set.
    Calculate cross-validated score.
```

- we'll pick the best set of hyperparameters *within each algorithm*.

```
For each algorithm:
  Keep the set of hyperparameter values with best cross-validated score.
  Re-train the algorithm on the entire training set (without cross-validat
```

Model Training cont.

Select winner

- Now it’s time to evaluate each model and pick the best one
- Because we've saved our **test set** as a truly unseen dataset, we can now use itto get a reliable estimate of each models' performance

There are a variety of **performance metrics** you could choose from. We won't spend too much time on them here, but in general **Mean Squared Error (MSE)** or **Mean Absolute Error (MAE)** is recommended for regression tasks

1. For each of your models, make predictions on your test set.
2. Calculate performance metrics using the predictions and the "ground truth" target variable from the test set

Model Training cont.

Finally, use these questions to help you pick the winning model:

- Which model had the best performance on the test set? **(performance)**
- Does it perform well across various performance metrics? **(robustness)**
- Did it also have (one of) the best cross-validated scores from the training set? **(consistency)**
- Does it solve the original business problem? **(win condition)**

Credits

- **Elite Data science 7 Days Crash Course. https://elitedatascience.com/**
- **Washington University's Machine Learning Specialization on Coursera. https://www.coursera.org/specializations/machine-learning**

Workshop Material

Note: The workshop material containing the python notebook and data file are available on guthub: https://github.com/fahad92virgo/pycon2017

You can find more tutorials on my blog in the future: pynerd.com (In progress)

"The key to artificial intelligence has always been the representation."

— Jeff Hawkins

THANK YOU