Data Science Workflow (iterative process):

Step 1: Project Scoping / Data Collection  
Step 2: Exploratory Data Analysis (EDA)  
Step 3: Data Cleaning  
Step 4: **Feature Engineering**  
Step 5: Model Training (including cross-validation to tune hyper-parameters)  
Step 6: Project Delivery / Insights

What is Feature Engineering?

**Coming up with features is difficult, time-consuming, requires expert knowledge.**

**“Applied machine learning” is basically feature engineering. From Andrew Ng**  
  
Data Cleaning involves:

Handling missing values, scaling and normalization,removing duplicates...

Feature Engineering involves:

1. Feature Transformation
2. **Feature Creation**
3. Feature Selection

Before fitting into model

Examples to consider for Feature Creation

*1. Indicator variable from thresholds*

For example, we are studying alcohol preferences and collected a dataset with an **age** feature. We can create an indicator variable for **age >= 21** to distinguish subjects who were over the legal drinking age.

New feature: **Legal drinker [Yes/No]**

*2. Consolidate variable from multiple features*

For example in real-estate prices prediction dataset, we have the features **n\_bedrooms** and **n\_bathrooms**. By knowledge, houses with **2 beds and 2 baths** are classified as premium

New feature: **House Type [premium/normal]**

*3. Event variable from date*

If we are handling dataset with **dates** for marketing, we can create event variables

New feature: **Black Friday [0/1]** and **Christmas [0/1]**.

*4. Using mathematical calculation*

Addition

If we are going to predict revenue based on preliminary sales data. You have the features **sales\_blue\_pens** and **sales\_black\_pens**. Why not adding two features for new feature?

New feature: **total\_sales\_pens** = **sales\_blue\_pens + sales\_black\_pens**

Subtraction

Given we have dataset with features **house\_built\_date** and **house\_purchase\_date**.

Dates often can give us opportunities to do subtraction for new feature

New feature: **house\_age\_at\_purchase** = **house\_purchase\_date - house\_built\_date**

Multiplication

For accounting features like **price** and **conversion**, it is natural to create new feature by multiplying the two features

New feature: **earnings** = **price\*conversion**

Division

For marketing campaigns with the features **n\_clicks** and **n\_impressions**, we can divide clicks by impressions to create a rate to compare across campaigns with different volume.

New feature: **click\_through\_rate** = **n\_clicks/n\_impressions**

*5. Representing features*

Date and time features:

From **purchase\_datetime**

Create: **purchase\_day\_of\_week,** **purchase\_hour\_of\_day**.

Aggregate**:purchases\_over\_last\_30\_days**.

Numeric to categorical mappings:

From **years\_in\_school**,

New feature: **Grade : ["Elementary School", "Middle School", "High School"]**

Grouping sparse classes:

When handling feature with many classes that have low sample counts.

Grouping the remaining ones into a single **"Other" class**.

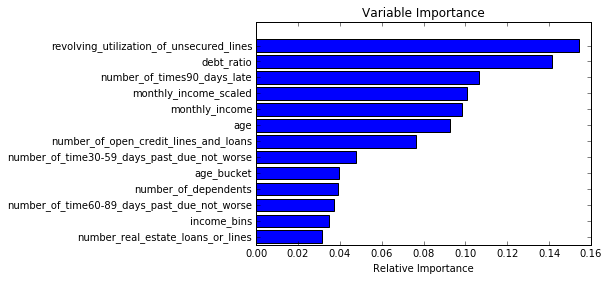
After fitting into model

Perform **Error Analysis** to understand why the model is meeting the benchmark

One usual function in tree-based model is **variable importance**

Variable importance in random forest

Feature importance in xgboost



References:

<https://elitedatascience.com/feature-engineering-best-practices>

<http://blog.yhat.com/tutorials/5-Feature-Engineering.html>

<https://campus.datacamp.com/courses/introduction-to-python-machine-learning-with-analytics-vidhya-hackathons/expert-advice-to-improve-model-performance?ex=2>