**Flow of model building**

**Feature engineering:**

* Look for outliers and other abnormal data according to the business context. Drop the outliers
* look for missing values and impute them approximately
* do **sns.pairplot** to check how every feature interacts with the target variable
* for regression tasks do the **correlation plot** to check what features are positively correlated and what are negatively correlated
* a **heat map** is a great source to understand the strongly correlated features and the weak ones
* strongly correlated features mean they have higher probability to affect the target variable
* weakly correlated features can be either dropped or imputed appropriately
* also for regression log-transforming the feature values can have equal effect on predicting the target variable

**Model building**

* In case of a small dataset, one could probably build a simple baseline model and then go for fitting advanced, improved models for performance. Finally interpret whether the advanced model has really yielded better performance compared to the baseline.
* In case of a large dataset, one could split the dataset into training and testing sets. Then further divide the train dataset into: train and validation sets.
* Build a pipeline of models.
* Cross-validate to find out the best forming model by using the train and validation sets.
* In case one’s not satisfied with the performance of the models in the cross-validation stage, one could swing back to the feature engineering stage to impute the features differently or to add more data or drop some of the useless data and so on according to time and resources.
* Dump the built-ready model into joblib or pickle.
* Loaded the dumped model when it’s time to test.
* Finally test the winning model on test data.

**Post model building**

* Interpret the result.
* Visualize the result in order to present.

**Lead scoring**

**Exponential decay:**

The exponential decay of a function can be represented as:

y = e -t/τ

However, it is very hard for the human eye (and brain) to see how well data fall upon an exponential curve. On the other hand, humans are attuned to straight lines. The results become clearer if we take the natural log of both sides:   
  
ln(y) = - (1/τ) t   
  
In other words, the negative inverse of the slope will be the time constant of decay. **Note that bad things happen when you take the natural log of zero, so recognize this when you are attempting your fit.**   
  
Note that often this is written in an alternate form:  
  
y = e -λ t  
  
and ln(y) = - λ t

Finally, the half-life of the decay is related to the decay constants in the following way:   
  
t1/2 = ln(2) τ = **ln(2) / λ**

So in this formula all we have to find is the speed of the decay.

**Case on hand: WeTravel**

Historical data has informed us that after 5 months there is only 50% chance that our customers will sign-up on our platform. So, the decay half-life is set at 5 months (provided we had expressed other units in months too).

**How did I build the model?**

I added a column of probability values thereby accounting for the decay term (by converting the term into float values). So accounts that closed within and equal to 5 months get and the ones closed above 5 months get the calculated value: = ln(2) / λ where λ=5 and it equals to 0.1386

So added a column of probability values thereby accounting for the decay term by calculating the term with λ starting from 2 (that’s depending on the context as usually it takes the travel brands at least 2 months to start transacting. Therefore the accounts that had closed within 2 months (meaning the accounts with num\_month\_diff as 1 get the probability values as 1). So the λ keeps varying depending on the num\_month\_diff values. For the accounts that failed I added the values with a negative sign to signify failure. I added the column of probability values before splitting the training dataset into train and validation. Had a test set in 100s of thousands of records of open accounts to classify. Built the following models: random forest, logistic regression, and naïve baye’s classifiers. Achieved a 92% accuracy with the random forest classifier. It’s put to use and so far so good.

Potential questions:

* "Let's say you have an Excel spreadsheet with 10,000 leads from a few months back -- long enough that those leads' sales cycle has passed. The file contains information about each lead, like their industry, title, company size, and what they did to become a lead (like downloading an ebook). Also in the file is whether they closed as a customer and how much their order was for. Can you use this information to create a lead score? How would you do it?"
* "What's an example of a lead-generating campaign you'd be excited to work on here?"