ELITE DATA SCIENCE

CHECKLIST: FEATURE ENGINEERING IDEAS

Feature engineering, the process creating new input features for machine learning, is one of the most effective ways to improve predictive models. Check out the most up-to-date full guide here.

INDICATOR VARIABLES

- **Indicator variable from thresholds:** Let's say you're studying alcohol preferences by U.S. consumers and your dataset has an `age` feature. You can create an indicator variable for `age >= 21` to distinguish subjects who were over the legal drinking age.
- **Indicator variable from multiple features:** You're predicting real-estate prices and you have the features `n_bedrooms` and `n_bathrooms`. If houses with 2 beds and 2 baths command a premium as rental properties, you can create an indicator variable to flag them.
- **Indicator variable for special events:** You're modeling weekly sales for an e-commerce site. You can create two indicator variables for the weeks of Black Friday and Christmas.
- **Indicator variable for groups of classes:** You're analyzing website conversions and your dataset has the categorical feature `traffic_source`. You could create an indicator variable for `paid_traffic` by flagging observations with traffic source values of `"Facebook Ads"` or `"Google Adwords"`.

INTERACTION FEATURES

- **Sum of two features:** Let's say you wish to predict revenue based on preliminary sales data. You have the features `sales_blue_pens` and `sales_black_pens`. You could sum those features if you only care about overall `sales_pens`.
- **Difference between two features:** You have the features `house_built_date` and `house_purchase_date`. You can take their difference to create the feature `house_age_at_purchase`.
- **Product of two features:** You're running a pricing test, and you have the feature `price` and an indicator variable `conversion`. You can take their product to create the feature `earnings`.
- **Quotient of two features:** You have a dataset of marketing campaigns with the features `n_clicks` and `n_impressions`. You can divide clicks by impressions to create `click_through_rate`, allowing you to compare across campaigns of different volume.

FEATURE REPRESENTATION

- **Date and time features:** Let's say you have the feature `purchase_datetime`. It might be more useful to extract `purchase_day_of_week` and `purchase_hour_of_day`. You can also aggregate observations to create features such as `purchases_over_last_30_days`.
- **Numeric to categorical mappings:** You have the feature `years_in_school`. You might create a new feature `grade` with classes such as `"Elementary School"`, `"Middle School"`, and `"High School"`.
- **Grouping sparse classes:** You have a feature with many classes that have low sample counts. You can try grouping similar classes and then grouping the remaining ones into a single `"Other"` class.
- **Creating dummy variables:** Depending on your machine learning implementation, you may need to manually transform categorical features into dummy variables. You should always do this *after* grouping sparse classes.

EXTERNAL DATA

- **Time series data:** The nice thing about time series data is that you only need one feature, some form of `date`, to layer in features from another dataset.
- **External API's:** There are plenty of API's that can help you create features. For example, the Microsoft Computer Vision API can return the number of faces from an image.
- **Geocoding:** Let's say have you `street_address`, `city`, and `state`. Well, you can geocode them into `latitude` and `longitude`. This will allow you to calculate features such as local demographics (e.g. `median_income_within_2_miles`) with the help of another dataset.
- **Other sources of the same data:** How many ways could you track a Facebook ad campaign? You might have Facebook's own tracking pixel, Google Analytics, and possibly another third-party software. Each source can provide information that the others don't track. Plus, any differences between the datasets could be informative (e.g. bot traffic that one source ignores while another source keeps).

ERROR ANALYSIS (POST-MODELING)

- **Start with larger errors:** Error analysis is typically a manual process. You won't have time to scrutinize every observation. We recommend starting with those that had higher error scores. Look for patterns that you can formalize into new features.
- **Segment by classes:** Another technique is to segment your observations and compare the average error within each segment. You can try creating indicator variables for the segments with the highest errors.
- **Unsupervised clustering:** If you have trouble spotting patterns, you can run an unsupervised clustering algorithm on the misclassified observations. We don't recommend blindly using those clusters as a new feature, but they can make it easier to spot patterns. Remember, the goal is to understand why observations were misclassified.
- **Ask colleagues or domain experts:** This is a great complement to any of the other three techniques. Asking a domain expert is especially useful if you've identified a pattern of poor performance (e.g. through segmentations) but don't yet understand why.