Assumptions of Linear Regression

1. Linear in terms of beta coefficients
2. Error is normally distributed and has population mean of 0
3. Homoskedasticity
4. Errors are uncorrelated across observations
5. Little to no multi-collinearity

① Linear in terms of beta coefficients

$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \varepsilon$ - linear

$Y = \beta_0 + e^{\beta_1} X^{\beta_2}$ - non-linear

OK to transform your features and target variables i.e. log

Not possible to use linear algebra to solve non-linear

② Error is normally distributed and has population mean of 0

$y = \hat{y} + \varepsilon \implies$

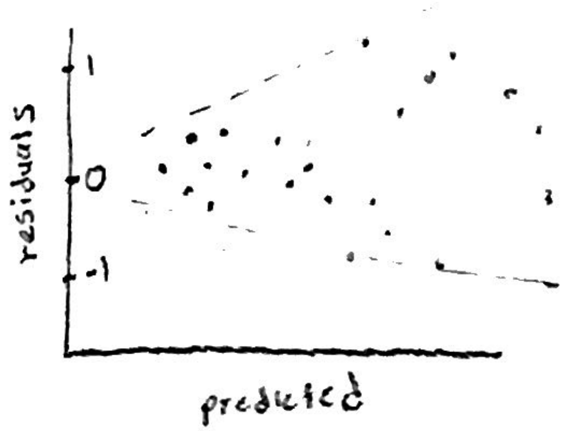
actual, estimate, error

$$y = X\beta + \varepsilon$$

$$\varepsilon = y - X\beta \sim N(0, \sigma^2)$$

$$N(0, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-\varepsilon^2}{2\sigma^2}\right)$$

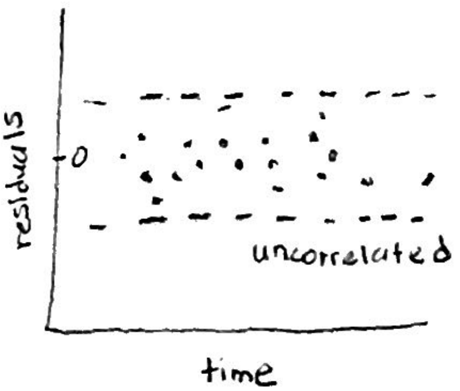
③ Homoskedasticity



Error of model has constant variance across all observations

Possible to use Weighted Least Squares if max residual variance is greater than 4 times min residual variance

④ Errors are uncorrelated across observations



Observed errors that follow a pattern are serially correlated or auto-correlated use time series prediction instead

⑤ Little to no multi-collinearity

$$L = \varepsilon_1 \times \varepsilon_2 \times \ldots \times \varepsilon_n = \prod_{i=1}^{N} \varepsilon_i$$

$$= \prod_{i=1}^{N} \left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp \frac{-\varepsilon_i^2}{2\sigma^2} \right)$$

Multiplying all the ε essentially gives the likelihood or the probability of having all these errors

$$\ln L = \sum_{i=1}^{N} \ln \left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp \frac{-\varepsilon_i^2}{2\sigma^2} \right)$$

Use log-likelihood to bring the exponential term down

$$= \sum_{i=1}^{N} \left(\ln \left(\frac{1}{\sqrt{2\pi\sigma^2}} \right) - \frac{\varepsilon_i^2}{2\sigma^2} \right)$$

$$= N \ln \left(\frac{1}{\sqrt{2\pi\sigma^2}} \right) - \sum_{i=1}^{N} \frac{\varepsilon_i^2}{2\sigma^2}$$

Move the constant term outside

$$= N \ln \left(\frac{1}{\sqrt{2\pi\sigma^2}} \right) - \left(\frac{1}{2\sigma^2} \right) (y - X\beta)^T (y - X\beta)$$

Substitute $\varepsilon = y - X\beta$

$$\frac{\partial \ln L}{\partial \beta} = 0 - \left(\frac{1}{2\sigma^2} \right) 2(-X^T)(y - X\beta) = 0$$

$$\Rightarrow X^T (y - X\beta) = 0$$

$$X^T y - X^T X \beta = 0$$

$$X^T y = X^T X \beta$$

$$\beta = (X^T X)^{-1} X^T y$$

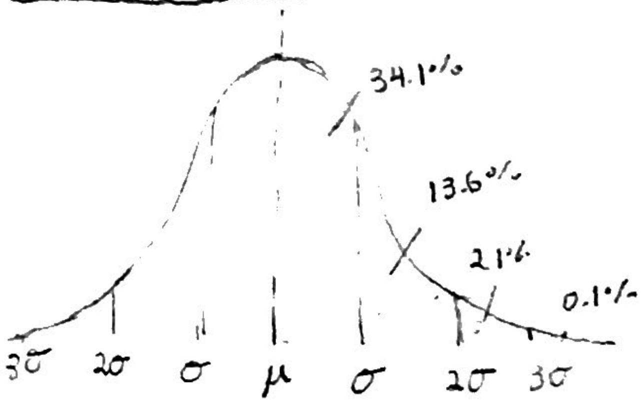
If two or more features are perfectly correlated, the matrix is not invertible.

If two or more features are highly correlated, OLS cannot separably estimate beta coefficient values for them

check magnitude and order of beta coefficients for collinear features

check condition number in statsmodel summary

Standardization



$$z = \frac{x - \mu}{\sigma}$$

μ ← mean

σ ← standard deviation

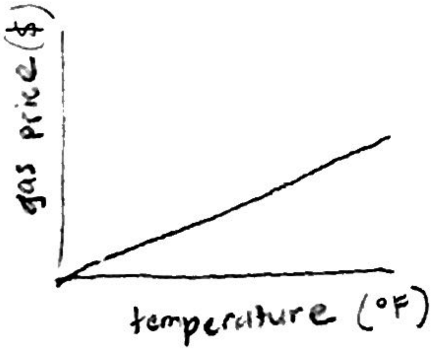
x ← feature value

z score has no units

standardize your data with StandardScaler before regularization

- StandardScaler works by subtracting off each feature column's mean and dividing by its standard deviation. This results in a z score – a measure of how many standard deviations a value is from the mean – for each value in the feature.

- How to make beta coefficient interpretable again after standardizing?
  A beta coefficient in linear regression tells us how much the outcome variable will change by for every one unit of change in the feature variable.



$$[\beta] = \frac{[\text{gas price}]}{[\text{temperature}]} = \frac{[\$]}{[°F]}$$ No standardizing

$$[\beta] = \frac{[\text{gas price}]}{[\text{standardized temperature}]} = \frac{[\$]}{\text{z-score}}$$ after standardizing

- Divide beta coefficient by standard deviation of feature to make beta coefficient interpretable again

- Standardizing allows us to interpret our intercept as the expected value of when are features values have a mean of 0. Not the same as our features being 0.

- If my temperature is 0°, does this mean gas price is not affected? Most likely not. If my temperature is average, there should be no effect though.

Regularization

- way to avoid overfitting by penalizing the model for having non-zero beta coefficients → increase bias, but reduces variance
- a model with some beta coefficients as 0 is less complex than a model with all non-zero beta coefficients

$$y = \beta_0 + \underset{\downarrow \atop 0}{\beta_1 X_1} + \beta_2 X_2 + \underset{\downarrow \atop 0}{\beta_3 X_3} \quad \text{v.s.} \quad y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3$$

$$= \beta_0 + \beta_2 X_2$$

- generally, a more complex model leads to overfitting

Ridge (L2)

X_1	X_2	X_3	X_4	Y
X_{11}	X_{12}	X_{13}	X_{14}	Y_1
X_{21}	X_{22}	X_{23}	X_{24}	Y_2

$X_{ij} \rightarrow X_{\text{row index}, \text{col index}}$

Loss function

$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

$$= RSS + \lambda \sum_{j=1}^{p} \beta_j^2$$

λ ← regularization strength parameter

LASSO (L1)

- performs feature selections in that features can go to 0
- cannot help with multi-collinearity, use ridge regression instead

Loss function

$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

$$= RSS + \lambda \sum_{j=1}^{p} |\beta_j|$$

λ ← regularization strength parameter

Elastic Net

- weighted combination of Ridge and LASSO, weighted by α value

Loss Function

$$RSS + \lambda \sum_{j=1}^{p} \left(\alpha \beta^2 + (1-\alpha) |\beta_j| \right)$$

Metrics

- R^2 value

$$R^2 = 1 - \frac{\sum_i (\underbrace{Y_i}_{\text{actual } Y} - \underbrace{\hat{Y}}_{\text{predicted } Y})^2}{\sum_i (Y_i - \underbrace{\bar{Y}}_{\text{mean of actual } Y})^2} = 1 - \frac{SSE}{SST}$$

$R^2 = 0$ model is good as guessing mean of Y_i or $\bar{Y}$

$R^2 < 0$ model is worse than guessing mean of Y_i or $\bar{Y}$

- assumes every feature explains variation in outcome variable

- adjusted R^2 value

$$\text{adjusted } R^2 = 1 - \frac{SSE}{SST} \cdot \frac{\overbrace{n}^{\text{number of observations}} - 1}{n - \underbrace{k}_{\text{number of variables, excluding constant}} - 1}$$

- generally better to use adjusted R^2
- penalizes model for adding features that do not fit the model, does not increase with every feature
- prevents overfitting

- Mean Absolute Error (MAE)

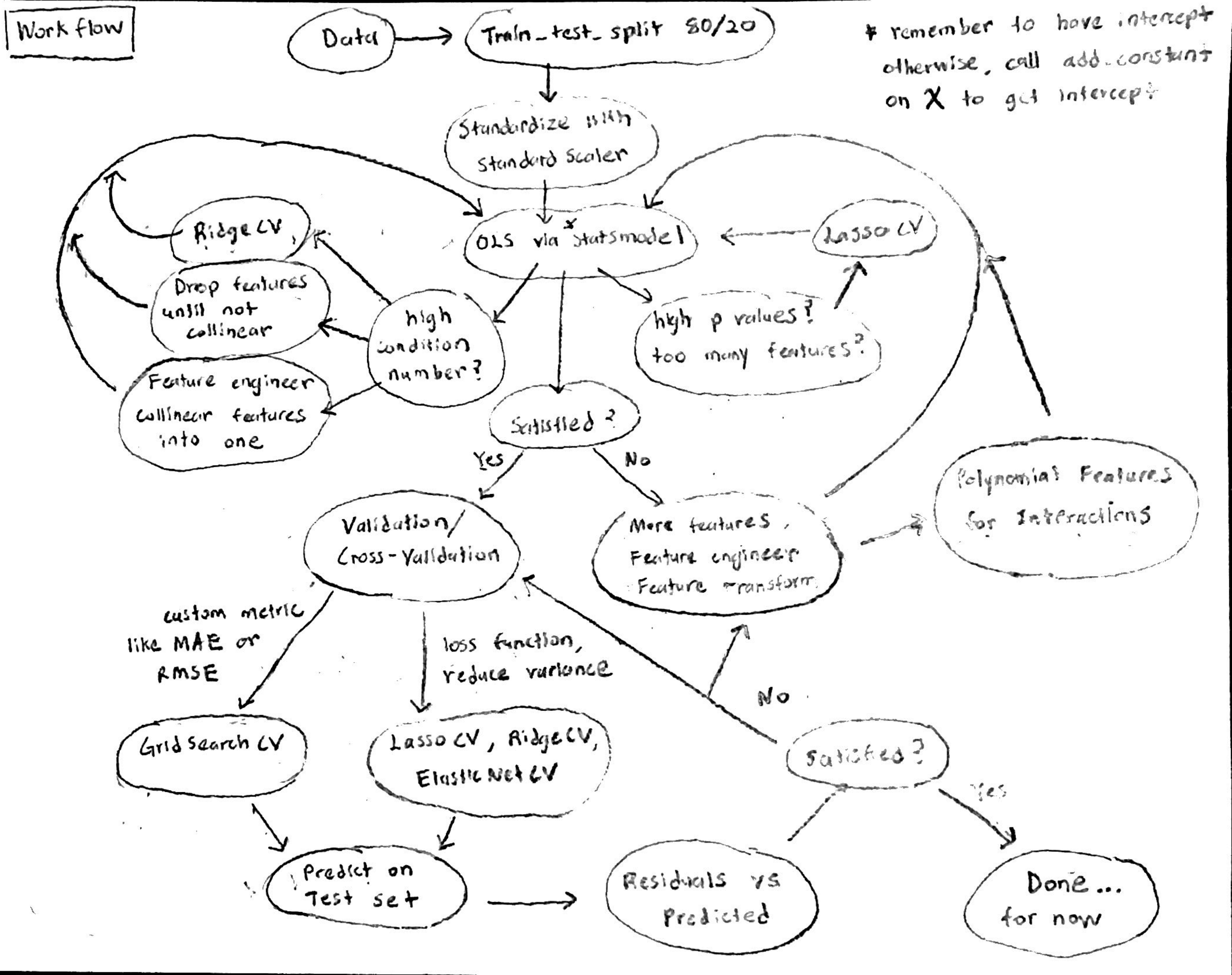
$$MAE = \frac{1}{\underbrace{n}_{\text{number of observations}}} \sum_{i=1}^{n} |\underbrace{y_i}_{\text{actual } Y} - \underbrace{\hat{y}_i}_{\text{predicted } Y}|$$

- does not indicate your model is underperforming or overperforming because we use absolute value of the residual
- small MAE $\rightarrow$ model is great at predicting
- large MAE $\rightarrow$ model has trouble in some areas
- interpretible, more business-oriented

- Root Mean Square Error (RMSE)

$$RMSE = \sqrt{\frac{1}{\underbrace{n}_{\text{number of observations}}} \sum_{i=1}^{n} (\underbrace{y_i}_{\text{actual } Y} - \underbrace{\hat{y}_i}_{\text{predicted } Y})^2}$$

- punishes more heavily against outliers in data and hence, large residual values
- do outliers matter to your model or not?



Work flow
Data
Train_test_split 80/20
* remember to have intercept
otherwise, call add_constant
on X to get intercept
Standardize with
Standard Scaler
OLS via *statsmodel
Lasso CV
Ridge CV,
Drop features
until not
collinear
Feature engineer
collinear features
into one
high
condition
number?
high p values?
too many features?
Satisfied?
Yes
No
Validation/
Cross-Validation
More features,
Feature engineer
Feature transform
Polynomial Features
for Interactions
custom metric
like MAE or
RMSE
loss function,
reduce variance
No
Grid Search CV
Lasso CV, Ridge CV,
ElasticNet CV
Satisfied?
Yes
Predict on
Test set
Residuals vs
Predicted
Done...
for now