

## ⚠ MESSAGE FROM AMBROSE ⚠

This review sheet (a.k.a. a “cheat sheet”) follows the order of topics in the [ACTEX Study Manual for Exam PA](#) and provides a “helicopter”  view of the entire PA exam syllabus. As you know, the focus of PA is on **conceptual understanding**  and **written communication** . To write well and score high, there are quite a lot of things you have to memorize as part of your exam prep, and this cheat sheet collects, I believe, the most important items all in one place for your convenience. Although no substitute for a thorough study of the manual , this cheat sheet should be a valuable aid to enhance retention and memorization. When I took PA in December 2019, I used a preliminary version of this cheat sheet and found it quite useful. (It was only 7 pages long back then!) Please feel free to refine it and put in additional facts and tips that you think are valuable.



# 1 General Model Building Steps

## 1.1 Problem Definition

- **Three main categories of predictive modeling problems:**

(More than one category can apply in a given business problem.)

Category	Focus	Aim
Descriptive	What happened in the <u>past</u>	To “describe” or explain observed trends by identifying <u>relationships</u> between variables
Predictive	What will happen in the <u>future</u>	To make accurate “predictions”
Prescriptive	The impacts of different “prescribed” <u>decisions</u>	To answer the “what if?” and “what is the <u>best course of action</u> ” questions

- **Characteristics of predictive modeling problems:**

- ▷ (*Issue*) There is a clearly identified and defined business issue to be addressed.
- ▷ (*Questions*) The issue can be addressed with a few well-defined questions. 
- ▷ (*Data*) Good and useful data is available for answering the questions above.
- ▷ (*Impact*) The predictions will likely drive actions or increase understanding.
- ▷ (*Better solution*) Predictive analytics likely produces a solution better than any existing approach.
- ▷ (*Update*) We can continue to monitor and update the models when new data becomes available.

- **How to produce a meaningful problem definition?**

- ▷ *General strategy:* Get to the root cause of the business issue and make it specific enough to be solvable.

- ▷ *Specific strategies:*
  - (*Hypotheses*) Use prior knowledge of the business problem to ask questions **?** and develop testable hypotheses.
  - (*KPIs*) Select appropriate key performance indicators for assessing the outcome. Desirable qualities of a KPI:
    - ① Align with the overall business strategy of the client.
    - ② Be easily measurable from data and provide an objective, quantitative basis to measure success.

- **Constraints to face:**

- ▷ The availability of easily accessible and high quality data
- ▷ Implementation issues, e.g.:
  - The presence of necessary IT infrastructure and technology to fit complex models efficiently
  - Cost and effort required to maintain the selected model

## 1.2 Data Collection and Validation

### Data design

- **Relevance:** Need to ensure that the data is unbiased, i.e., representative of the environment where the model will operate.
  - ▷ *Population:* Important for the data source to be a good proxy of the true population of interest.
  - ▷ *Time frame:* Choose the time period that best reflects the business environment of interest.
 

In general, recent history is better than distant history.
- **Sampling:** The process of taking a subset of observations from the data source to generate the dataset

- ▷ **Random sampling:** “Randomly” draw observations from the underlying population without replacement. Each record is equally likely to be sampled.
- ▷ **Stratified sampling:** Divide the underlying population into a no. of non-overlapping “strata” (w.r.t. target or predictors) non-randomly, then randomly sample a set no. of observations from each stratum ⇒ get a more representative sample.
 

*A special case—systematic sampling:* Draw observations according to a set pattern; no random mechanism controlling which observations are sampled.

- **Granularity:** Refers to how precisely a variable is measured, i.e., level of detail for the information contained by the variable.

*Example:* Month is more granular than quarter.

### Data quality issues

- **Reasonableness:** Data values should be reasonable (make sense) in the context of the business problem, e.g., variables such as age, time, and income **\$** should be non-negative.
- **Consistency:** Data records should be inputted consistently on the same basis and rules, e.g., same measurement unit for numeric variables, same coding scheme for categorical variables.
- **Sufficient documentation:** Examples of useful elements:
  - ▷ A description of the dataset overall, including the data source
  - ▷ A data dictionary, with a clear description of each variable (name, definition, and format)
  - ▷ Notes about past updates or other irregularities of the data
  - ▷ A statement of accountability for the correctness of the data
  - ▷ A description of the governance processes used to manage the data

## Other data issues

- **Structured vs. unstructured data:**

	Structured Data	Unstructured Data
Definition	Data that fit into a <u>tabular arrangement</u> 	Data that do not, e.g., text, image, audio
Pros	Easier to manipulate	More flexible
Cons	Cannot represent information that does not naturally fit into a tabular arrangement	Harder to access and have to be carefully pre-processed

- **Personally identifiable information (PII):** Information that can be used to trace an individual's identity, e.g., name, SSN, address, photographs , and biometric records

How to handle PII?

- ▷ *Anonymization:* Anonymize or de-identify the data to remove the PII.
- ▷ *Data security:* Ensure that the data receives sufficient protection, e.g., encryption, access/transfer restrictions.
- ▷ *Terms of use:* Be well aware of the terms and conditions, and the privacy policy related to the collection and use of data.

- **Variables with legal/ethical concerns:**

- ▷ *Sensitive variables:* Differential treatment based on sensitive variables may lead to unfair discrimination and raise equity concerns.  
*Examples:* Race, ethnicity, gender, age, income \$, disability status , or other prohibited classes 

- ▷ *Proxy variables:* Variables that are not prohibited variables themselves, but are closely related to (hence serve as a “proxy” of) prohibited variables.

Examples:

- Occupation (possibly a proxy of gender)
- Geographical location (possibly a proxy of age, income)

- **Target leakage:**

- ▷ *Definition:* When predictors in a model “leak” information about the target variable that would not be available when the model is deployed in practice
- ▷ *Key to detecting target leakage—Timing:* These variables are observed at the same time as or after the target variable. 
- ▷ *Problem with this issue:* These variables cannot serve as predictors in practice and would lead to artificially good model performance if mistakenly included.
- ▷ *An extreme and commonly tested form of target leakage:* When the target variable itself is included as a predictor or is used to develop new features, e.g., PCs, cluster groups.

## 1.3 Exploratory Data Analysis (EDA)

- **Aims:**

- ▷ (*Data validation*) Clean and validate the data for inappropriate and anomalous entries (outliers)  
⇒ make the data ready for analysis.
- ▷ (*Generating insights for modeling*) Understand the characteristics of variables (on marginal + joint bases)  
⇒ identify potentially useful predictors, generate new features, decide which type of model (e.g., GLMs vs. decision trees) is more suitable, etc.

- **Variable type:** Determines the type of exploratory tools to use.

▷ *Numeric (a.k.a. quantitative):* Take the form of numbers with an associated range.

*Discrete:* Restricted to only certain numeric values, e.g., non-negative integers.

*Continuous:* Can assume any value within the range.

▷ *Categorical (a.k.a. qualitative, factor):* Take predefined values in a countable collection of “categories,” a.k.a. levels, classes.

*Nominal:* Levels have no natural order.

*Ordinal:* Levels have a natural order.

(**Note:** Can't say one level is higher than another by how much, unlike numeric variables. ⚠)

- **Two types of general tools, with relative pros and cons:**

	Summary Statistics	Graphical Displays
Pros <b>+</b>	<ul style="list-style-type: none"> <li>▷ Precise and objective</li> <li>▷ Easily <u>comparable</u> across variables.</li> </ul>	<ul style="list-style-type: none"> <li>▷ Can gain a quick <u>visual impression</u> of a variable's distribution.</li> <li>▷ Can reveal information not easily captured by summary statistics</li> </ul>
Cons <b>-</b>	<ul style="list-style-type: none"> <li>▷ Can only capture a certain aspect of a variable's distribution.</li> <li>▷ Some statistics (e.g., mean) can be easily <u>distorted by outliers</u>.</li> </ul>	<ul style="list-style-type: none"> <li>▷ Information provided is not as precise as summary statistics.</li> <li>▷ Less comparable across variables.</li> <li>▷ Poorly designed graphs can be hard to interpret.</li> </ul>

- **Univariate exploration tools:** Look at one variable at a time.

Type	Statistics	Plots	Observations
<b>Numeric</b>	<b>Mean, median, variance,</b> min., max.	<b>Histograms, boxplots</b>	<ul style="list-style-type: none"> <li>▷ Shape of distribution (e.g., mode, any skew)?</li> <li>▷ Any unusual values?</li> </ul>
<b>Categorical</b>	Class frequencies	<b>Bar charts</b> 	<ul style="list-style-type: none"> <li>▷ Which levels are most common?</li> <li>▷ Any sparse levels?</li> <li>▷ (For binary Y) Presence of imbalance</li> </ul>

- **Bivariate exploration tools:** Look at pairs of variables.

Pair	Statistics	Plots	Observations
Numeric × Numeric	<b>Correlations</b> (only for <u>linear</u> relations)	<b>Scatterplots</b>	Any noticeable relationships, e.g., linear, monotonic, non-monotonic?
Numeric × Categorical	Mean/median of numeric variable split by categorical variable	<b>Split boxplots, histograms</b> (stacked or dodged)	Any sizable differences in the means/medians among the factor levels?
Categorical × Categorical	2-way frequency table 	<b>Bar charts</b> (stacked, dodged, or filled)	Any sizable differences in the class proportions among different factor levels?

▷ *Stacked, dodged, vs. filled bar charts:*

- The best one for showing the variation of proportions of a categorical variable across the levels of another categorical variable:

Filled bar chart.

(*Downside:* Lose information about the no. of observations in each level of the variable in  $x$  aesthetic.)

- The best one for showing the variation of proportions of a categorical variable within each level of another categorical variable:

Dodged bar chart.

(*Reason:* Proportions within each level are aligned and put side by side on a common baseline.)

● **Common data issues for numeric variables:**

---

**Issue 1: Highly correlated predictors**

---

Problems	<ul style="list-style-type: none"> <li>▷ Difficult to <u>separate out</u> the effects of individual predictors on the target variable</li> <li>▷ For GLMs, coefficients become widely varying in sign and magnitude, and difficult to interpret.</li> <li>▷ For decision trees, strong collinearity can adversely <u>skew variable importance scores</u>.</li> </ul>
Possible Solutions	<ul style="list-style-type: none"> <li>▷ <u>Drop</u> one of the strongly correlated predictors.</li> <li>▷ Use <b>PCA</b> to compress the correlated predictors into a few representative <b>PCs</b>.</li> </ul>

---



---

**Issue 2: Skewness (esp. right skewness due to outliers)**

---

Problems	<p>Extreme values:</p> <ul style="list-style-type: none"> <li>▷ Exert a disproportionate effect on model fit</li> <li>▷ Distort visualizations (e.g., axes of a plot expanded inordinately to take care of outliers)</li> </ul>
Possible Solutions	<ul style="list-style-type: none"> <li>▷ Remove outliers if there is a strong rationale, e.g., they are recording errors, they make up a tiny part of the data.</li> <li>▷ Apply <b>transformations</b> to reduce right skewness:             <ul style="list-style-type: none"> <li>□ <b>Log transformation</b> <math>\ln x = \log x</math> (works only for <u>strictly +ve</u> variables; remedy: add a small positive number to each value of the variable if there are zero or -ve values)</li> <li>□ <b>Square root transformation</b> <math>\sqrt{x}</math> (works for non-negative variables)</li> </ul> </li> </ul>

---



---

**Issue 3: Should they be converted to a factor?**

---

Important considerations	<p>“Yes” if...</p> <ul style="list-style-type: none"> <li>▷ Variable has a rather small no. of distinct values, e.g., quarter of the year (1 to 4).</li> <li>▷ Variable values are merely numeric <u>labels</u> (no sense of numeric order), e.g., group no.</li> <li>▷ Variable has a <u>complex relationship</u> with target variable <math>\Rightarrow</math> factor conversion gives models (esp. GLMs) more flexibility to capture the relationship.</li> </ul>
--------------------------	--

---

---

**Issue 3: Should they be converted to a factor? (Cont.)**


---

- Important considerations “No” if...
- ▷ Variable has a large no. of distinct values, e.g., hour of the day, with 24 values.  
(Reason: Would cause a high dimension and overfitting if converted into a factor.)
  - ▷ Variable values have a sense of numeric order possibly useful for predicting the target variable.
  - ▷ Variable has a simple monotonic relationship with the target  
⇒ its effect can be effectively captured by treating it as a numeric variable.
  - ▷ Future observations will have new variable values, e.g., calendar year.
- 

- **Common data issue for categorical predictors: Sparse levels**

- ▷ *Problem with high dimensionality/granularity:* Sparse factor levels reduce robustness of models and may cause overfitting.
- ▷ *A solution:* Combine sparse levels with more populous levels where the target variable behaves similarly to form more representative and interpretable groups.
- ▷ *Trade-off:* To strike a balance  between:
  - Ensuring each level has a sufficient no. of observations
  - Preserving the differences in the behavior of the target variable among different factor levels for prediction

- ▷ *Tip:* Knowledge of the meaning of the variables is often useful when making combinations, e.g., regrouping hour of day as “morning,” “afternoon,” and “evening.”  
(Try to use common sense and check the data dictionary!)

- **Interaction:**

- ▷ *Definition:* Relationship between a predictor and the target variable depends on the value/level of another predictor.  
(*Tip:* Good to include the definition in your response whenever an exam subtask tests interaction!)
- ▷ *Graphical displays to detect interactions:*

Predictor Combination	Numeric Target	Categorical Target
Numeric × Categorical	Scatterplot colored by categorical predictor	Boxplot for numeric predictor split by target and faceted by categorical predictor
Categorical × Categorical	Boxplot for target split by one predictor and faceted by the other predictor	Bar chart for one predictor filled by target and faceted by the other predictor
Numeric × Numeric	Bin one of the predictors (i.e., cut it into several ranges), and consider a numeric × categorical pair.	

- ▷ *Interaction vs. correlation:* Literally similar, but fundamentally different
  - Interaction:* Concerns a 3-way relationship (1 target variable and 2 predictors)
  - Correlation:* Concerns the (linear) relationship between two numeric predictors only

## 1.4 Model Construction and Evaluation

### Training/test set split

- How does it work?

Before fitting models, split the data into the training set (approx. 70-80%) and the test set (approx. 20-30%).



Models are fitted to  $\rightarrow$  Training set  
 Prediction performance is evaluated on  $\rightarrow$  Test set

Test set observations must be truly unseen to the trained model.

- Why do the split?

- ▷ Model performance on the training set tends to be overly optimistic and favor complex models.
- ▷ Test set provides a more objective ground for assessing the performance of models on new, unseen data.
- ▷ Split replicates the way the models will be used in practice.

- Trade-off about the sizes of the two sets:

Larger training set  $\Rightarrow$   $\begin{cases} \text{Training is more robust} \\ \text{Evaluation on test set is less reliable} \end{cases}$

- How to do the split?

- ▷ By stratified sampling w.r.t. the target variable  $\Rightarrow$  produce representative training and test sets.
- ▷ Split based on a time variable, e.g., year  $\Rightarrow$  to evaluate how well a model extrapolates past time trends to future, unseen years.

### Common performance metrics

- General

- ▷ *Regression vs. classification problems:*
  - Regression:** When the target is numeric
  - Classification:** When the target is categorical  
 (**Note:** The predictors can be numeric or categorical. **!**)
- ▷ What do metrics computed on training and test sets measure:
  - Training:** Goodness of fit to training data
  - Test:** Prediction performance on new, unseen data
- ▷ *Loss function:* Most performance metrics use a loss function to capture the discrepancy between the actual and predicted values for each observation of the target variable numerically.

Examples:

Loss Function	Metric (the <u>lower</u> the value, the better)
Square loss	$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$ or RMSE = $\sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$
Absolute loss	$\text{MAE} = \frac{1}{n} \sum_{i=1}^n  y_i - \hat{y}_i $ <p>(<b>Note:</b> Absolute function places much less relative weight on the large errors and reduces the impact of outliers on the fitted model.)</p>
0-1 loss	$\text{Classification error rate} = \frac{1}{n} \sum_{i=1}^n 1_{\{y_i \neq \hat{y}_i\}}$

- **Additional metrics for (binary) classification problems:**

▷ *Classification rule:*

$$\text{Predicted probability for "+"} > \text{cutoff} \Leftrightarrow \text{Predicted class} = \text{"+"}$$

▷ *Confusion matrices:* Typical arrangement:

Prediction	Reference (= Actual)	
	-	+
-	TN	FN
+	FP	TP

□ **Accuracy** =  $\frac{TN + TP}{n}$  = proportion of correctly classified obs.

(Classification error rate =  $1 - \text{Accuracy} = \frac{FN + FP}{n}$ )

□ **Sensitivity** =  $\frac{TP}{TP + FN}$  = proportion of +ve obs. correctly classified as +ve

□ **Specificity** =  $\frac{TN}{TN + FP}$  = proportion of -ve obs. correctly classified as -ve

□ **Precision** =  $\frac{TP}{FP + TP}$  = proportion of +ve predictions truly belonging to +ve class

Properties of these metrics:

□ The higher (closer to 1), the better.

□ How certain metrics vary with cutoff:

$$\text{Cutoff} \uparrow \Rightarrow \begin{cases} \text{Sensitivity} \downarrow \\ \text{Specificity} \uparrow \end{cases}$$

May use a cost-benefit analysis to optimize cutoff.

□ Weighted average relation:

$$\text{Accuracy} = \frac{n_-}{n} \times \text{Specificity} + \frac{n_+}{n} \times \text{Sensitivity}.$$

▷ *ROC curves:*

□ Plot sensitivity against specificity for all cutoffs from 0 to 1.

□ All ROC curves share two special points:

$$(\text{specificity}, \text{sensitivity}) = \begin{cases} (0, 1), & \text{if cutoff} = 0, \\ (1, 0), & \text{if cutoff} = 1. \end{cases}$$

□ *A graphical assessment of classifier performance:* The closer the curve is to the top-left corner, the better.

▷ *Area under the ROC curve (AUC)*

□ Typically ranges between 0.5 (random classifier) and 1 (perfect classifier).

□ *A quantitative assessment of classifier performance:* The higher (closer to 1), the better.

## Cross-validation (CV)

- **How it works:**

For a fixed +ve integer  $k$  (e.g., 10), randomly split the training data into  $k$  folds of approximately equal size

↓

Train the model on all but one folds and measure performance on left-out fold

↓

Repeat with each fold left out in turn to get  $k$  performance values

↓

Average to get overall CV metric

- **Common uses of CV:**

▷ *Model assessment:* To evaluate a model's test set performance without using any test set.

▷ *Model selection (hyperparameter tuning):* To tune **hyperparameters** (= parameters with values supplied in advance; not optimized by the model fitting algorithm) by picking the values that produce the best CV performance, e.g., lowest MSE or highest accuracy.

- **Considerations when selecting the best model:**

▷ (*Prediction performance*) The model should perform well on test data w.r.t. certain performance metrics.

▷ (*Interpretability*) The model should be reasonably interpretable, i.e., the predictions should be easily explained in terms of the predictors and lead to specific insights.

▷ (*Ease of implementation*) The easier for a model to be implemented (computationally, financially, or logistically), the better the model.

## Sidebar: Unbalanced data (for binary targets)

- **Meaning:** One class is much more dominant than the other.

- **Problems with unbalanced data:**

▷ A classifier implicitly places more weight on the majority class and tries to fit those observations well at the expense of the minority class, which may be the class of interest.

▷ A high accuracy can be deceptive.

- **Solution 1—Undersampling:** Keep all observations from the minority class, but draw fewer observations (“undersample”) from the majority class.

▷ *Drawback:* Less data  $\Rightarrow$  training becomes less robust and the classifier becomes more prone to overfitting.

- **Solution 2—Oversampling:** Keep all observations from the majority class, but draw more observations (“oversample”) from the minority class.

▷ *Drawback:* More data  $\Rightarrow$  heavier computational burden

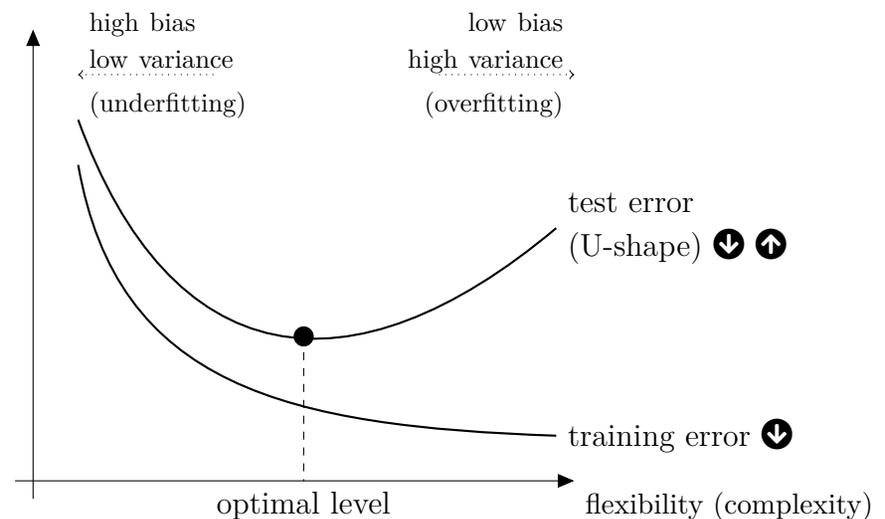
▷ *Caution:* Should be done after training/test set split (Otherwise, some observations may appear in both training and test sets.)

- **Effects of undersampling and oversampling on model results:**

+ve class becomes more prevalent in the balanced data  
 $\Downarrow$   
 Predicted probabilities for +ve class will tend to increase  
 $\Downarrow$   
 For a fixed cutoff, sensitivity  $\Uparrow$  but specificity  $\Downarrow$

## Controlling model complexity

- **How model complexity affects various quantities:**



- **Underfitting vs. overfitting:**

	Underfitting	Overfitting
Training error	Large	Small
Test error	Large	Large
Behavior	<u>Not complex</u> enough to capture the signal effectively ⇒ Poor performance on training + test sets	<u>Too complex</u> and trying too hard to capture the signal and the noise specific to the training data. ⇒ Not generalize well to new, unseen data.

- **Quantitative framework—Bias-variance trade-off:**

▷ *Bias-variance decomposition of expected test MSE:*

$$\mathbb{E}_{\text{Tr}, Y_0} \left[ \left( Y_0 - \hat{f}(\mathbf{X}_0) \right)^2 \right]$$

$$= \underbrace{[\text{Bias}_{\text{Tr}}(\hat{f}(\mathbf{X}_0))]^2 + \text{Var}_{\text{Tr}}[\hat{f}(\mathbf{X}_0)]}_{\text{reducible error}} + \underbrace{\text{Var}(\varepsilon_0)}_{\text{irreducible error}}$$

Quantity	<b>Bias</b>	<b>Variance</b>
Mathematical definition	Difference between the expected value of prediction and the true signal value	Amount of variability of prediction
What it measures	Accuracy	Precision
Behavior	⬇ with complexity	⬆ with complexity

▷ *Practical implications:*

Need to set model complexity to a reasonable level

⇓

$\left\{ \begin{array}{l} \text{optimize bias-variance trade-off} \\ \text{avoid underfitting \& overfitting} \end{array} \right. \Rightarrow \text{improve prediction performance}$

- **Feature generation:** To generate new features (= derivations from original variables) with the following objectives:

▷ (*Predictive power*) The features transform the information contained in the original variables into a more useful form ⇒ a predictive model can “absorb” the information and capture the signal more effectively.

▷ (*Interpretability*) New features may make a model easier to interpret.

- **Sidebar: Dimensionality vs. granularity**

▷ Granularity ⬆ ⇒ model complexity tends to ⬆

▷ Two main differences between the two concepts:

Concept	Applicability	Comparability
Dimensionality	Specific to categorical variables	Two categorical variables can always be ordered by dimension.
Granularity	Applies to both numeric and categorical variables	Not always possible to order two variables by granularity

## 1.5 Model Validation

- **Aim:** To check that the selected model has no obvious deficiencies and the model assumptions are largely satisfied.
- **Validation method based on the training set:** For a “nice” GLM, the deviance residuals should:
  - ① (*Purely random*) Have no systematic patterns (on their own and w.r.t. the predictors).
  - ② (*Homoscedasticity*) Have approximately constant variance upon standardization.
  - ③ (*Normality*) Be approximately normal (for most target distributions).

For ① & ②: Check a “**Residuals vs Fitted**” plot.

For ③: Check a **Q-Q plot**.

- **Validation methods based on the test set:**
  - ▷ *Predicted vs. actual values of target:* The two sets of values should be close (can check this quantitatively or graphically).
  - ▷ *Benchmark model:* Show that the recommended model outperforms a benchmark model, if one exists (e.g., intercept-only GLM, purely random classifier), on the test set.

## 1.6 Recommendations for Next Steps

- (*Adjust the business problem*) Changes in external factors, e.g., market conditions, regulations, may cause initial assumptions to shift  $\Rightarrow$  need to modify the business problem to incorporate the new conditions.
- (*Consult with subject matter experts*) Seek validation of model results from external subject matter experts.

- (*Gather additional data*) Enlarge training data with new obs. and/or variables, and retrain the model to improve robustness.
- (*Apply new types of models*) Try new types of models when new technology or implementation possibilities are available.
- (*Refine existing models*) Try new combinations or transformations of predictors, alternative hyperparameter values, alternative accuracy measures, etc.
- (*Field test proposed model*) Implement the recommended model in the exact way it will be used to gain users’ confidence.

## 2 Specific Types of Model

### 2.1 GLMs

- **Assumptions: Linear models (LMs) vs. GLMs**

	LMs	GLMs
Independence	Given the predictor values, the observations of the target variable are <u>independent</u> . (Same for LMs & GLMs.)	
Target distribution	Given the predictor values, the target variable follows a <u>normal</u> distribution.	Given the predictor values, the target distribution is a member of the linear exponential family.
Mean	The target mean directly equals the linear predictor: $\mu = \eta$ $= \beta_0 + \beta_1 X_1 + \dots + \beta_p X_p.$	A function (“link”) of the target mean equals the linear predictor: $g(\mu) = \eta \quad \text{link} \quad \text{linear predictor}$
Variance	Constant, regardless of predictor values	Varies with $\mu$ and the predictor values

- Two key components of a GLM:

- ① **Target distribution:** Choose one (in the linear exponential family) that aligns with the characteristics of the target.
- ② **Link functions:** Some important considerations:
  - ▷ Ensure the predictions match the range of values of the target mean.
  - ▷ Ensure ease of interpretation, e.g., **log link**.
  - ▷ (Minor) **Canonical links** make convergence more likely.

- Common e.g. of target distributions and link functions:

Variable Type	Common Dist.	Common Link
Real-valued with a bell-shaped dist.	Normal (Gaussian)	Identity
Binary (0/1)	Binomial	Logit
Count ( $\geq 0$ , integers)	Poisson	Log
Strictly +ve, continuous with right skew	Gamma, inverse Gaussian	Log
$\geq 0$ , continuous with a large mass at zero	Tweedie	Log

**Note:** ⚠

- ▷ For gamma and inverse Gaussian, the target variable has to be strictly positive. Values of zero are not allowed. (A *remedy*: Add a small positive amount, e.g., +0.1, to each target value.)
- ▷ The link function is applied to the target mean  $\mu$ ; the target variable itself is not transformed, so the log link may still work even for a target distribution with zero values, e.g., Poisson. (See Exercise 4.1.4 (c) in the manual.)

## Feature generation

- **Methods for handling non-monotonic relations:** GLMs, in their basic form, assume that numeric predictors have a monotonic (not necessarily linear) relationship with the target variable.

- ① **Polynomial regression:** Add polynomial terms to the model equation:

$$g(\mu) = \beta_0 + \beta_1 X + \underbrace{\beta_2 X^2 + \dots + \beta_m X^m}_{\text{polynomial terms}} + \dots$$

Pros. ⊕ Can take care of more complex relationships between the target variable and predictors. The more polynomial terms included, the more flexible the fit.

Cons. ⊖

- ▷ Coefficients become harder to interpret (all polynomial terms move together).
- ▷ Usually no clear choice of  $m$ ; can be tuned by CV or informed by EDA

- ② **Binning:** “Bin” the numeric variable and convert it into a categorical variable with levels defined as non-overlapping intervals over the range of the original variable.

Pros. ⊕ No definite order among the coefficients of the dummy variables corresponding to different bins  $\Rightarrow$  target mean can vary highly irregularly over the bins.

Cons. ⊖

- ▷ Usually no clear choice of the no. of bins and the associated boundaries
- ▷ Results in a loss of information (exact values of the numeric predictor gone)

③ Adding **piecewise linear functions**: Add features of the form  $(X - c)_+$ .

- Pros. **+** A simple way to allow the relationship between a numeric variable and the target mean to vary over different intervals
- Cons. **-** Usually no clear choice of the break points

● Handling categorical predictors—**Binarization**::

▷ *How it works*: (Automatically done in R behind the scenes.)

Categorical predictor  
↓  
A collection of dummy (binary) variables  
indicating one and only one level  
(= 1 for that level, = 0 otherwise)

↓

Dummy variables serve as predictors in model equation

▷ **Baseline level**: The level at which all **dummy variables** equal 0.

- *R's default*: The alpha-numerically first level
- *A good practice*: Set it to the most common level.

● **Interactions**:

Need to “manually” include interaction terms  
of the product form  $X_j X_k$

↓

Coefficient of  $X_j$  will vary with the value of  $X_k$

## Interpretation of coefficients

● **General statements**:

- ▷ Coefficient estimates capture the effects (magnitude + direction) of features on the target mean.
- ▷ p-values express statistical significance of features; the smaller, the more significant.

● **Specific statements based on log link**: Assume all else equal.

▷ *Numeric case*: For a unit change in a numeric predictor with estimated coefficient  $\hat{\beta}_j$ ,

$$\begin{array}{l} \text{multiplicative change} \\ \text{in target mean} \end{array} = e^{\hat{\beta}_j},$$

equivalently

$$\begin{array}{l} \text{percentage (\%) change} \\ \text{in target mean} \end{array} = 100(e^{\hat{\beta}_j} - 1)\%.$$

▷ *Categorical case*: For a non-baseline level of a categorical predictor with estimated coefficient  $\hat{\beta}_j$ ,

$$\begin{array}{l} \hat{\mu} \\ \text{@non-baseline level} \end{array} = e^{\hat{\beta}_j} \times \begin{array}{l} \hat{\mu} \\ \text{@baseline level} \end{array}$$

equivalently

$$\begin{array}{l} \hat{\mu} \\ \text{@non-baseline level} \end{array} \text{ is } 100(e^{\hat{\beta}_j} - 1)\% \text{ higher than } \begin{array}{l} \hat{\mu} \\ \text{@baseline level} \end{array}$$

## Other modeling techniques: Offsets vs. weights

	Offsets	Weights
Form of the target variable	Aggregate (e.g., <u>total</u> # claims in a group of similar policyholders)	Average (e.g., <u>average</u> # claims in a group of similar policyholders)
Do they affect the target mean or variance?	Target <u>mean</u> is <u>directly proportional</u> to <b>exposure</b> , e.g., with log link, $\mu_i = E_i \exp(\dots).$	<u>Variance</u> is <u>inversely</u> related to exposure: $\text{Var}(Y_i) = \frac{(\text{some terms})}{E_i}.$ Observations with a larger exposure will play a more important role in model fitting.

## Stepwise selection

- **Selection process:** Sequentially add/drop features, one at a time, until there is no improvement in the selection criterion. More computationally efficient than best subset selection (= fitting a separate model for each possible combination of features and selecting the “best subset”).

Area	Backward	Forward
① Which model to start with?	Full model	Intercept-only model
② Add or drop variables?	Drop	Add
③ Which method tends to produce a simpler model?		Forward selection

- Selection criteria based on penalized likelihood:

▷ *Idea:* Prevent overfitting by requiring an included/retained feature to improve model fit by at least a specified amount.

▷ *Two common choices:*

Criterion	Definition	Penalty per Parameter
<b>AIC</b>	$-2l + 2(p + 1)$	2
<b>BIC</b>	$-2l + [\ln(n_{\text{tr}})](p + 1)$	$\ln(n_{\text{tr}})$

(In R,  $-2l$  is treated as the deviance.)

▷ *AIC vs. BIC:*

- For both, the lower the value, the better.
- BIC is more conservative and results in simpler models.

- **Manual binarization:** Convert factor variables to dummy variables manually before running stepwise selection.

Pros. **+** To be able to add (resp. drop) individual factor levels that are statistically significant (resp. insignificant) w.r.t. baseline level

Cons. **-** ▷ More steps in the **stepAIC()** procedure ⇒ heavier computational burden

▷ Possibly non-intuitive results, e.g., only few levels of a factor are retained

## Regularization

- **Idea:** Reduce overfitting by shrinking the size of the coefficient estimates, especially those of non-predictive features.

- **How it works:** To optimize training loglikelihood (equivalently, training deviance) adjusted by a penalty term that reflects the size of the coefficients, i.e., to minimize

deviance + regularization penalty.

The formulation serves to strike a balance  $\Delta$  between goodness of fit and model complexity.

- **Common forms of penalty term:**

Method	Penalty	Characteristic
<b>Lasso</b>	$\mathbb{L} = \lambda \sum_{j=1}^p  \beta_j $	Some coef. may be zero
<b>Ridge regression</b>	$\mathbb{R} = \lambda \sum_{j=1}^p \beta_j^2$	None reduced to zero
<b>Elastic net</b>	$\alpha \mathbb{L} + (1 - \alpha) \mathbb{R}$	Some coef. may be zero

- **Two hyperparameters:**

①  $\lambda$ : Regularization (a.k.a. shrinkage) parameter

▷ Controls the amount of regularization:

$$\lambda \begin{matrix} \uparrow \\ \text{more shrinkage} \end{matrix} \Rightarrow \text{complexity} \begin{matrix} \downarrow \\ \end{matrix} \Rightarrow \begin{cases} \text{bias}^2 \uparrow \\ \text{variance} \downarrow \end{cases}$$

▷ **Feature selection property:** For elastic nets with  $\alpha > 0$  (lasso, in particular), some coefficient estimates become exactly zero when  $\lambda$  is large enough.

▷ *Typically tuned by CV:* Choose  $\lambda$  with the smallest CV error.

②  $\alpha$ : Mixing parameter

▷ Controls the mix between ridge ( $\alpha = 0$ ) and lasso ( $\alpha = 1$ ) (**Note:** Need to remember  $\alpha = 0$  is ridge regression and  $\alpha = 1$  is lasso.  $\blacktriangle$ )

▷ Provided that  $\lambda$  is large enough, increasing  $\alpha$  from 0 to 1 makes more coefficient estimates zero.

▷ Cannot be tuned by `cv.glmnet()`; need to tune manually.

## 2.2 Single Decision Trees

- **Basics:**

▷ *Idea:* Divide  $\times$  the feature space into a set of non-overlapping regions containing relatively homogeneous observations (w.r.t. target).

▷ *Deliverable:* A set of classification rules based on the values/levels of predictors and represented in the form of a “tree”  $\blacktriangle$

▷ *Predictions:* Observations in the same terminal node share the same predicted mean (for numeric targets) or same predicted class (for categorical targets).

- **Recursive binary splitting:**

▷ *Two terms:* The algorithm is...

□ **Greedy:** At each step, adopt the split that leads to the greatest reduction in impurity (or largest information gain) at that point, instead of looking ahead and selecting a split that results in a better tree in a future step. (Repeat until a stopping criterion is reached.)

□ **Top-down:** Start from the “top” of the tree, go “down,” and sequentially partition the feature space in a series of splits.

▷ *Node impurity measures:*

Tree Type	Name of Measure	Formula
<b>Regression</b>	Residual sum of squares	$\sum_{i \in R_m} (y_i - \hat{y}_{R_m})^2$
	<b>Classification error rate</b>	$1 - \max_{1 \leq k \leq K} \hat{p}_{mk}$
<b>Classification</b>	<b>Gini index</b>	$\sum_{k=1}^K \hat{p}_{mk}(1 - \hat{p}_{mk})$
	<b>Entropy</b>	$-\sum_{k=1}^K \hat{p}_{mk} \log_2(\hat{p}_{mk})$

Properties:

- The smaller, the purer the observations in the node.
- Gini index and entropy are similar numerically.
- Gini index and entropy are more sensitive to node impurity than classification error rate  
(Reason: They depend on all  $\hat{p}_{mk}$ , not just the max. class proportion.)

● **Tree parameters:**

Parameter	Name in R	Meaning	Effect
Minimum bucket size	<code>minbucket</code>	Min. # obs. in a terminal node	Higher, tree less complex
Complexity parameter	<code>cp</code>	Min. improvement required for a split to be made (not 100% right...)	Higher, tree less complex
Maximum depth	<code>maxdepth</code>	# edges from root node to furthest node	Higher, tree more complex

- ▷ *Learning tip:* Be sure to know how these parameters limit tree complexity!
- ▷ `cp` can be tuned by CV within `rpart()`; `minbucket` and `maxdepth` have to be tuned by trial and error.

● **Interpretation of trees:** Things you can comment on:

- ▷ No. of tree splits
- ▷ Split sequence, e.g., start with  $X_1$ , further split the larger bucket by  $X_2, \dots$

- ▷ Which are the most important predictors (usually those in early splits)?
- ▷ Which terminal nodes have the most observations? Any sparse nodes?
- ▷ Any prominent interactions?
- ▷ (*Classification trees*) Interesting combinations leading to the +ve event

● **Cost-complexity pruning:**

- ▷ *Rationale:* To reduce tree complexity by pruning branches from bottom that do not improve goodness of fit by a sufficient amount  $\Rightarrow$  prevent overfitting and ease interpretation.
- ▷ *How it works:*

Step 1. Grow a large tree  $T_0$ . (**Note:** Don't miss this step. )

Step 2. Minimize the penalized objective function

$$\text{relative training error} + \frac{c_p \times |T|}{(\text{tree complexity})},$$

(model fit to training data)

over all subtrees of  $T_0$ , where

$$\text{training error} = \begin{cases} \text{RSS}, & \text{for regression,} \\ \# \text{ misclassifications,} & \text{for classification.} \end{cases}$$

- ▷ *About the hyperparameter `cp`:*
  - `cp`   $\Rightarrow$  tree less complex (smaller)
  - Typically tuned by CV:* Set `cp` to the value that minimizes CV error (`xerror` in `cptable`).
- ▷ *Alternative: **One-standard-error (1-SE) rule***
  - How:* Select the smallest tree whose CV error is within 1 SE of the minimum CV error.
  - Rationale:* Select a simpler and more interpretable tree with comparable prediction performance. (Occam's razor)

## • Do variable transformations affect GLMs and trees?

	GLMs	Trees
Transformations on target variable	Yes (The transformations alter the values of the predictors and target variable that go into the likelihood function.)	Yes (The transformations can alter the values of node impurity measures, e.g., RSS, that define the tree splits.)
Transformations on predictors	Yes (Same reasoning as above)	Yes, unless the transformations are <u>monotonic</u> , e.g., log (Monotonic transformations will not change the <u>ranks</u> of the predictor values.)

## 2.3 Ensemble Trees

### Random forests (many trees!)

#### • Idea:

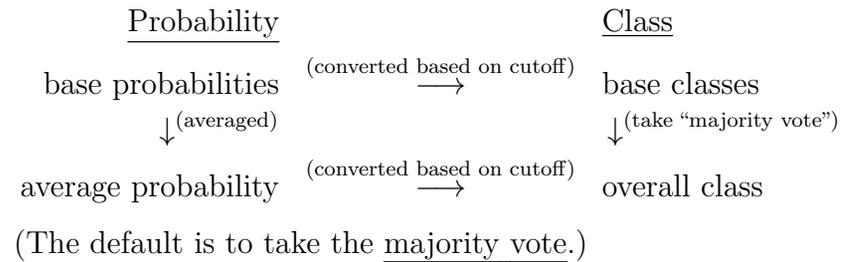
- ▷ (*Variance reduction*) Combine the results of multiple trees fitted to different **bootstrapped training samples** in parallel  $\Rightarrow$  reduce variance of overall predictions.
- ▷ (*Randomization*) Take a random sample of predictors as candidates for each split  $\Rightarrow$  reduce correlation between base trees  $\Rightarrow$  further reduce variance of overall predictions.

#### • Combining base predictions to form overall prediction:

- ▷ *Case 1—Regression trees:* By averaging:

$$\hat{f}_{\text{rf}}(\mathbf{x}) = \frac{1}{B} \sum_{b=1}^B \hat{f}^{*b}(\mathbf{x}).$$

- ▷ *Case 2—Classification trees:* Two methods:



#### • Key parameters:

- ▷ **mtry**: # features randomly sampled as candidates at each split
  - mtry**  $\downarrow$   $\Rightarrow$  greater variance reduction but bias  $\uparrow$
  - Common choice**:  $\sqrt{p}$  (classification) or  $p/3$  (regression)
  - Typically tuned by CV or out-of-bag estimation
- ▷ **ntree**: # trees to be grown
  - ntree**  $\uparrow$   $\Rightarrow$  greater variance reduction
  - Often overfitting does not arise even if set to a large no.
  - Set to a relatively small value to save run time

## Boosting

#### • Idea:

- ▷ In each iteration, fit a tree to the residuals of the preceding tree and subtract a scaled-down version of the current tree’s predictions from the residuals to form the new residuals.
- ▷ Each tree focuses on observations the previous tree predicted poorly.
- ▷ Overall prediction:  $\hat{f}(\mathbf{x}) = \sum_{b=1}^B \lambda \hat{f}^b(\mathbf{x})$ .

#### • Key parameters:

- ▷ **eta**: Learning rate (or shrinkage) parameter

□ *Effects of eta*: eta  $\uparrow$   $\Rightarrow$  algorithm converges faster but is more prone to overfitting.

□ *Rule of thumb*: Set to a relatively small value

▷ **nrounds**: Max. # rounds in the tree construction process

□ *Effects of nrounds*: nrounds  $\uparrow$   $\Rightarrow$  algorithm learns better but is more prone to overfitting.

□ *Rule of thumb*: Set to a relatively large value

□ *Interaction with eta*: A smaller eta requires a larger nrounds for sufficient training.

#### • Random forests vs. boosted trees:

Item	Random Forest	Boosting
Fitting process	In <u>parallel</u>	In <u>series</u> (sequential)
Focus	Variance	Bias
Overfitting	Less vulnerable	More vulnerable
Hyperparameter tuning	Less sensitive	More sensitive

### Two interpretational tools for ensemble trees

#### • Variable importance plots:

▷ *Definition of importance scores*: The total drop in node impurity (RSS for regression trees and Gini index for classification trees) due to splits over a given predictor, averaged over all base trees:

$$\text{importance score} = \frac{1}{B} \times \sum_{\text{all splits over that predictor}} \text{impurity reduction}$$

▷ *Use*: To identify important variables (those with a large score)

▷ *Limitation*: The score doesn't tell us how the important variables affect the target.

#### • Partial dependence plots:

▷ *Definition of partial dependence*: Model prediction obtained after averaging the values/levels of variables not of interest:

$$\text{PD}(x_1) := \frac{1}{n_{\text{tr}}} \sum_{i=1}^{n_{\text{tr}}} \hat{f}(\underbrace{x_1}_{\text{fixed}}, \underbrace{x_{i2}, \dots, x_{ip}}_{\text{averaged}})$$

▷ *Use*: Plot  $\text{PD}(x_1)$  against various  $x_1$  to show the marginal effect of  $X_1$  on the target variable.

▷ *Limitations*:

① Assume predictor of interest is independent of other predictors  $\Rightarrow$  some predictions may be based on practically unreasonable combinations of values.

② May miss heterogeneous relationships in the data.

### 2.4 Pros and Cons of Different Models

• **Tips for recommending a model**: Refer to the business problem (prediction vs. interpretation) and characteristics of data (e.g., any complex, non-monotonic relations?)

#### • GLMs:

▷ *Pros*:  $\oplus$

① (*Target distribution*) GLMs excel in accommodating a wide variety of distributions for the target variable.

② (*Interpretability*) The model equation shows how the target mean depends on the features; coefficients = interpretable measure of directional effect of features.

③ (*Ease of implementation*) Simple to implement

▷ *Cons*:  $\ominus$

① (*Complex relationships*) Unable to capture non-monotonic (e.g., polynomial) or non-additive relationships (e.g., interaction), unless additional features are manually incorporated.

- ② (*Interpretability*) For some link functions (e.g., inverse link), the coefficients may be difficult to interpret.

### • Regularized GLMs:

#### ▷ Pros:

- ① (*Categorical predictors*) Via the use of model matrices, binarization of categorical variables is done automatically and each factor level treated as a separate feature to be removed.
- ② (*Hyperparameter tuning*) An elastic net can be tuned by CV using the same criterion (e.g., MSE, accuracy) ultimately used to judge the model against unseen test data.
- ③ (*Variable selection*) For elastic nets with  $\alpha > 0$ , variable selection can be done by making  $\lambda$  large enough.

#### ▷ Cons:

- ① (*Categorical predictors*) Possible to see some non-intuitive or nonsensical results when only a handful of the levels of a categorical predictor are selected.
- ② (*Target distribution*) Limited/restricted model forms allowed by `glmnet()` (Weak point!)

### • Single trees:

#### ▷ Pros:

- ① (*Interpretability*) If there are not too many buckets, trees are easy to interpret because of the if/then nature of the classification rules and their graphical representation.
- ② (*Complex relationships*) Trees excel in handling non-monotonic and non-additive relationships without the need to insert extra features manually.

- ③ (*Categorical variables*) Categorical predictors are automatically handled by separating their levels into two groups without the need for binarization.

- ④ (*Variable selection*) Variables are automatically selected as part of the model building process. Variables that do not appear in the tree are filtered out and the most important variables show up at the top of the tree.

#### ▷ Cons:

- ① (*Overfitting*) Strongly dependent on training data (prone to overfitting)  $\Rightarrow$  predictions unstable with a high variance  $\Rightarrow$  lower user confidence
- ② (*Numeric variables*) Usually need to split based on a numeric predictor repeatedly to capture its effect effectively  $\Rightarrow$  tree becomes large, difficult to interpret.
- ③ (*Categorical variables*) Tend to favor categorical predictors with many levels (*Reason:* Too many ways to split  $\Rightarrow$  easy to find a spurious split that looks good just on training data, but doesn't really exist in the signal.)

### • Ensemble trees:

- ▷ *Pros:* Much more robust and predictive than base trees by combining the results of multiple trees

#### ▷ Cons:

- ① Opaque (“black box” ) , difficult to interpret (*Reason:* Many base trees are used, although variable importance or partial dependence plots can help.)
- ② Computationally prohibitive to implement (*Reason:* Huge computational burden with fitting multiple base trees.)

### 3 Unsupervised Learning

- **Supervised vs. unsupervised learning:** Two main differences:

	Supervised	Unsupervised
Target	Present	Absent (or ignored if present)
Goal	To make inference or <u>predictions</u> for the target	To extract <u>relationships</u> between variables

- **Two reasons why unsupervised learning is often more challenging than supervised learning:**

- ① (*Objectives*) Objectives in unsupervised learning are more fuzzy and subjective (no simple goal like prediction).
- ② (*Harder to assess results*) Methods for assessing model quality based on the target variable (e.g., CV) are generally not applicable.

#### 3.1 Principal Components Analysis (PCA)

- **Idea:**

- ▷ To transform a set of numeric variables into a smaller set of representative variables (**PCs**)  $\Rightarrow$  reduce dimension of data
- ▷ Especially useful for highly correlated data  $\Rightarrow$  a few PCs are enough to capture most information.

- **Properties of PCs:**

- ▷ Linear combinations of the original features:

$$z_{im} = \phi_{1m}x_{i1} + \phi_{2m}x_{i2} + \cdots + \phi_{pm}x_{ip}.$$

with  $\phi_{1m}^2 + \phi_{2m}^2 + \cdots + \phi_{pm}^2 = 1$  (normalization).

- ▷ (*Maximal variance direction*) PC loadings are chosen to capture as much information in the data (w.r.t. variance) as possible
- ▷ (*The line with minimal distance*) PC loadings define the line that minimizes the sum of the squared perpendicular distances between each data point and the line.
- ▷ Mutually uncorrelated (different PCs capture different aspects of the data)
- ▷ **Amount of variance explained** decreases with PC order, i.e., PC1 explains the **most variance** and subsequent PCs explain less and less.

- **Two applications of PCA:**

- ① *Data visualization:* Reduce the dimension of the data from  $p$  to  $M \Rightarrow$  easier to explore and visualize.
- ② *Feature generation:* Replace the original variables by PCs to reduce overfitting and improve prediction performance.

- **Interpretation of PCs:**

- ▷ *Signs and magnitudes of PC loadings:* What do the PCs represent, e.g., proxy, average, or contrast of which variables? Which variables are more correlated with one another?
- ▷ *Sizes of **proportions of variance explained** (PVEs):*

$$\text{PVE}_m = \frac{\text{Variance explained by } m\text{th PC}}{\text{Total variance}}.$$

Are the first few PVEs large enough (related to the strong correlations between variables)? If so, the PCs are useful.

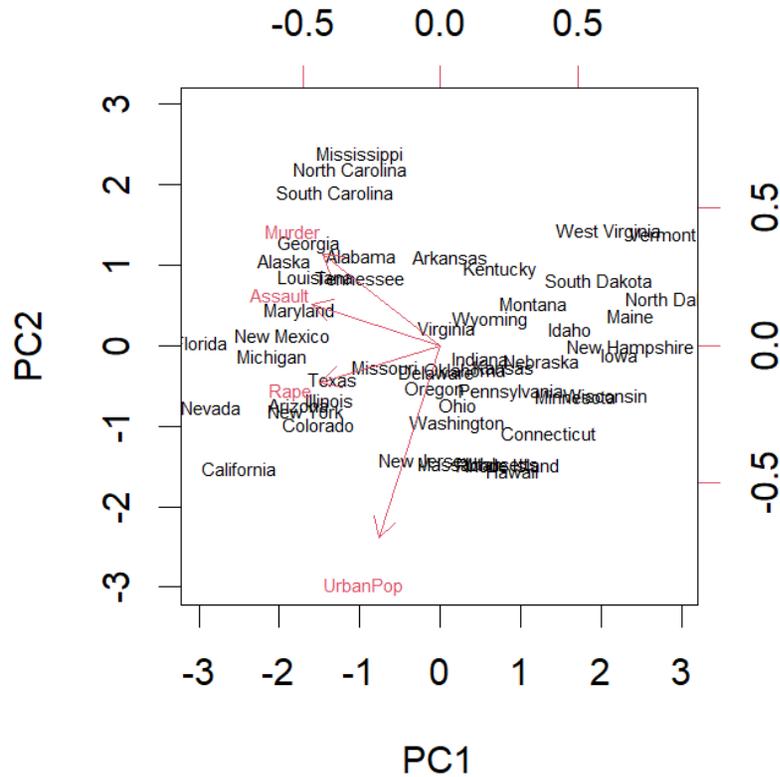
- **Biplots:** A useful tool to visualize PCA output

- ▷ (*Visualizing high-D data*) Graph the scores of the first two PCs against each other in a 2D scatterplot.

▷ (“bi”) Display two kinds of information in the same plot:

- ① PC Scores: For observations
- ② **PC loading vectors** (in red): For variables

Example:



▷ **PC loadings** on top and right axes  $\Rightarrow$  deduce meaning of PCs

▷ PC scores on bottom and left axes  $\Rightarrow$  deduce characteristics of observations (based on meaning of PCs)

• **Number of PCs ( $M$ ) to use:**

▷ *Trade-off:*  $M \uparrow \Rightarrow$   $\begin{cases} \text{cumulative PVE} \uparrow \\ \text{dimension} \uparrow \\ \text{(if } y \text{ exists) model complexity} \uparrow \end{cases}$

▷ *How to choose  $M$ :*

- **Scree plot:** Eyeball the plot and locate the “elbow” (= point at which the PVEs of subsequent PCs have dropped off to a sufficiently low level).
- *CV:* Treat  $M$  as a hyperparameter to be tuned if  $Y$  exists.

• **Drawbacks of PCA:**

- ▷ Loss of interpretability  
(*Reason:* PCs as composite variables can be hard to interpret.)
- ▷ Not good for non-linearly related variables  
(*Reason:* PCs rely on linear transformations of variables.)
- ▷ PCA does dimension reduction, but not feature selection.  
(*Reason:* PCs are constructed from all original features.)
- ▷ Target variable is ignored. (*Remember:* PC is unsupervised.)

### 3.2 Cluster Analysis

• **Basic idea:**

- ▷ To partition observations into a set of non-overlapping subgroups (“clusters”) generated with the following two goals:
  - ① (*Homogeneity*) Observations within each cluster share similar characteristics while observations in different clusters are rather different (well separated).
  - ② (*Interpretability*) The characteristics of the clusters are interpretable and meaningful within the context of the business problem.

• **Two feature generation methods based on clustering:**

- ▷ *Cluster groups:* As a new factor variable
- ▷ *Cluster means:* As a new numeric variable

## $K$ -means Clustering

- **Idea:** For a fixed +ve integer  $K$ , choose  $K$  clusters  $C_1, \dots, C_K$  to minimize the total within-cluster SS,  $\sum_{k=1}^K W(C_k)$ , where

$$W(C_k) = \sum_{i \in C_k} d(\mathbf{x}_i, \bar{\mathbf{x}}_k)^2$$

= sum of squared Euclidean distances between each observation in  $C_k$  and centroid of  $C_k$

- How the **algorithm** works:

- ▷ *Step 1 (Initialization):* Given  $K$ , randomly select  $K$  points in the feature space as initial cluster centers.
- ▷ *Step 2 (Iteration):* Repeat the following steps until the cluster assignments no longer change:
  - Assign each observation to the cluster with the closest center.
  - Recalculate the  $K$  cluster centers (hence “ $K$ -means”).

- **Good practice:** Set `nstart` to a large integer, e.g.,  $\geq 20$ .

Reason:

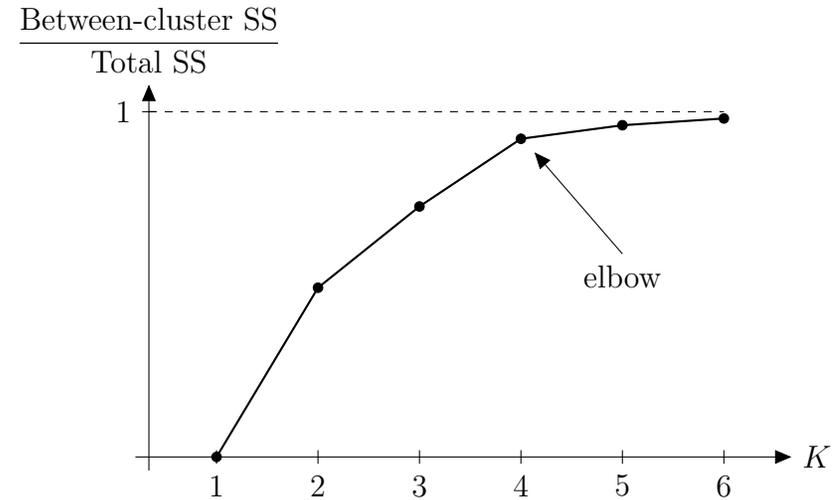
The algorithm produces a local optimum, which depends on the randomly selected initial cluster centers.

↓

Run the algorithm multiple times to improve the chance of finding a better local optimum, if not the global optimum.

- Selecting the value of  $K$  by **elbow method**:

- ▷ Plot the proportion of variation explained against  $K$ .



- ▷ Choose the “elbow,” beyond which the proportion of variation explained is marginal.

## Hierarchical Clustering

- **Idea:**

- ▷ **Algorithm:**

- Start with the individual observations, each treated as a separate cluster.
- Successively fuse the closest pair of clusters, one at a time.
- Stop when all clusters are fused into a single cluster containing all observations.

- ▷ *Output:* A “hierarchy” of clusters which can be visualized by a dendrogram

- **Linkage:** To measure the dissimilarity between two clusters, at least one of which has  $\geq 2$  observations

Linkage	Inter-cluster Dissimilarity
<b>Complete</b> (default)	Maximal pairwise distance
<b>Single</b>	Minimal pairwise distance
<b>Average</b>	Average of all pairwise distances
<b>Centroid</b>	Distance between the two centroids

Properties:

- ▷ Complete and average linkage are commonly used.  
(*Reason:* They tend to result in more balanced clusters.)
- ▷ Single linkage tends to produce extended, trailing clusters with single observations fused one-at-a-time.
- ▷ Centroid linkage may lead to inversion (some later fusions occur at a lower height than an earlier fusion).

- **Dendrogram:** An upside-down tree showing the sequence of fusions and the inter-cluster dissimilarity (“Height” on vertical axis) when each fusion occurs.

Some insights from a dendrogram:

- ▷ (*Similarities between clusters*) Clusters joined towards the bottom of a dendrogram are rather similar to one another, while those fused towards the top are rather far apart.
- ▷ *Key considerations when choosing the no. of clusters:*
  - $K$  should be small (e.g., 3 to 7) to ease interpretation.
  - The resulting clusters have a similar no. of observations (balanced).
  - The dissimilarity corresponding to the chosen  $K$  should not be way higher than the next lower threshold.  
( $\therefore$  If not, some clusters can be further split to reduce dissimilarity drastically.)

- **$K$ -means vs. hierarchical clustering:**

Item	$K$ -means	Hierarchical
Is randomization needed?	Yes (for initial cluster centers)	No
Is the no. of clusters pre-specified?	Yes ( $K$ needs to be specified)	No (Specify the height of the dendrogram later)
Are the clusters nested?	No	Yes (a hierarchy of clusters)

## Other Practical Issues

- **Scaling of variables matters for both PCA and clustering**

- ▷ *Without scaling:*

Variables on a large scale will  
dominate variance and distance calculations

↓

have a disproportionate effect on  
PC loadings & cluster groups

- ▷ *With **scaling** (generally recommended):* All variables are on the same scale and share the same degree of importance.

- **Correlation-based distance** Focuses on shapes of feature values rather than their exact magnitudes.
- **Validating the clusters obtained:** Apply the same clustering method to an independent set of observations and checking if those observations exhibit more or less the same set of clusters.
- **Clustering and curse of dimensionality:**
  - ▷ Visualization of the results of cluster analysis becomes problematic in high dimensions ( $p \geq 3$ ).
  - ▷ As  $p$  increases, intuition breaks down and it becomes harder to differentiate between observations that are close and those that are far apart.