Lecture 20: Missing Data and Model Visualization

CS109A: Introduction to Data Science

Harvard University

- **Course:** CS109A: Introduction to Data Science
- **Lecture:** Lecture 20
- **Instructors:** Pavlos Protopapas, Kevin Rader, Chris Gumb
- **Topics:** Missing Data, MCAR/MAR/MNAR, Imputation Methods, Model Visualization, Partial Dependence Plots

Lecture Overview

This lecture covers two important practical topics in data science:

1. **Missing Data:** Real datasets almost always have missing values. Understanding *why* data is missing and handling it properly is crucial to avoid biased models.
2. **Model Visualization:** For "black-box" models like KNN and decision trees, we can't simply look at coefficients. Partial Dependence Plots (PDPs) help us understand how features affect predictions.

Key Insight: Both topics are about dealing with the messiness of real-world data science—incomplete data and complex models that resist simple interpretation.

Contents

1 What is Missing Data?

1.1 The Problem

Definition:

Missing Data **Missing data** (or **missingness**) refers to values that should exist in a dataset but are absent. In Python/Pandas, these appear as `NaN` (Not a Number) or `NA`.

Missing data creates two problems:

1. **Technical:** Most ML libraries (including sklearn) throw errors when they encounter NaN values
2. **Statistical:** Improper handling can introduce **bias** into predictions and estimates

Warning

What is Bias?

Bias means your estimates or predictions are **systematically wrong**—not just randomly off, but consistently wrong in a particular direction.

Analogy: A scale that always reads 2 lbs too high is biased. No matter how many times you weigh yourself, you'll always be 2 lbs over your true weight.

In modeling:

- **Biased prediction:** Your $\hat{y}$ systematically over- or under-estimates the true y
- **Biased estimation:** Your $\hat{\beta}$ systematically differs from the true β

1.2 Why Does Missing Data Occur?

Missing data can arise for many reasons:

- **Data entry errors:** Someone accidentally skipped a field
- **Non-response:** Survey respondent chose not to answer a question
- **New variable collection:** You started measuring something new midway through data collection
- **Measurement limits:** Sensor couldn't detect values below a threshold
- **Study dropout:** Participant left a clinical trial

Important:

The Million-Dollar Question **Why** is the data missing? The answer determines:

- Whether simple approaches will work
- How much bias you might introduce
- What imputation strategy to use

The reason for missingness is the key to proper handling.

1.3 Simple (But Problematic) Approaches

Before taking this class, you might have done:

Option 1: Drop rows with missing values

```
df_clean = df.dropna()  # Remove any row with NaN
```

Option 2: Fill with mean/median/mode

```
df['X'].fillna(df['X'].mean(), inplace=True)
```

Warning

Problems with Simple Approaches:

Dropping rows:

- Loses valuable data
- If missingness is related to outcome, introduces bias
- If 30% of rows have missing values, you lose 30% of data!

Mean imputation:

- Artificially reduces variance (all missing values become the same)
- Weakens relationships between variables
- Doesn't account for uncertainty in imputed values

2 Types of Missing Data: MCAR, MAR, MNAR

Understanding *why* data is missing is crucial. Statisticians classify missing data into three types:

2.1 MCAR: Missing Completely at Random

Definition:

MCAR Data is **Missing Completely at Random** if the probability of missingness is unrelated to:

- The missing value itself
- Any other observed variable

Missingness is purely random—like someone randomly punched holes in your spreadsheet.

Example: While entering survey data into Excel, the data entry person randomly skipped some values due to fatigue. The skipping had nothing to do with what the values were or who the respondents were.

Good news: With MCAR, dropping rows or simple imputation won't introduce bias (though you may lose statistical power).

Bad news: True MCAR is rare in practice.

2.2 MAR: Missing at Random

Definition:

MAR Data is **Missing at Random** if the probability of missingness can be fully explained by **other observed variables**—but not by the missing value itself.

Example: In a survey about workplace harassment, people may choose not to answer based on their gender (which you observed), not based on whether they were actually harassed.

Key point: "Random" here is misleading. It means "random *conditional on observed data*." If you know gender, the missingness becomes random.

Good news: With MAR, we can use other observed variables to model and correct for missingness.

2.3 MNAR: Missing Not at Random

Definition:

MNAR Data is **Missing Not at Random** if the probability of missingness depends on:

- The unobserved (missing) value itself, OR
- Some unobserved variable

Example 1: High-income people are less likely to report their income *because* it's high. The missing value itself affects missingness.

Example 2: Patients drop out of a clinical trial because they experienced severe side effects. The side effect severity (unmeasured) causes both dropout and likely worse outcomes.

Bad news: MNAR is the hardest case. No statistical method can fully correct for bias because the information needed is... missing!

Table 1: *Summary of Missing Data Types*

Type	Missingness Depends On	Example	Handling Difficulty
MCAR	Nothing (pure random)	Data entry error	Easiest
MAR	Observed variables only	Gender affects response rate	Moderate
MNAR	Missing value or unobserved variables	High earners hide income	Hardest

Important:

The Fundamental Problem **Can you ever know which type you have?**

No. You cannot definitively determine from the data alone whether missingness is MCAR, MAR, or MNAR. MNAR involves unmeasured variables—by definition, you can't see them.

Practical approach: Assume MAR and use the best imputation methods available. This at least handles the recoverable bias.

3 Imputation Methods

3.1 Important Considerations

Before choosing an imputation method, consider:

1. **Where is the missingness?**
   - In predictors (X): Most imputation methods handle this
   - In response (Y): Much harder—often just drop these rows
2. **Variable type:** Numeric vs categorical (different methods apply)
3. **Amount of missingness:**
   - $< 10\%$: Usually safe to impute
   - $10 - 50\%$: Be careful; imputation may add noise
   - $> 50\%$: Consider dropping the variable entirely

3.2 Method 1: Missingness Indicator Variable

This simple method treats "missing" as **information itself**:

1. For variable X_1 with missing values:
2. Create X_1^*: Impute missing values with 0 (or mean)
3. Create X_1^{miss}: Binary indicator (1 if was missing, 0 otherwise)
4. Use **both** X_1^* and X_1^{miss} in your model

Example:

Missingness Indicator

Original		**Imputed**		**Indicators**	
X1	X2	X1*	X2*	X1_miss	X2_miss
10	0	10	0	0	0
5	1	5	1	0	0
21	NaN	21	**0**	0	**1**
NaN	0	**0**	0	**1**	0

Now the model can learn: "When X2_miss=1, treat X2*=0 differently than when X2_miss=0"

Key Information

Why This Works:

This creates a "did not respond" category. The model can estimate separate effects for:

- Observed zeros
- Imputed zeros (actually missing)

This is especially useful when MNAR is suspected—the fact that someone didn't respond might itself be predictive!

3.3 Method 2: Model-Based Imputation

Treat imputation as a **prediction problem**:

1. Identify variable X_j with missing values
2. **Training set:** Rows where X_j is observed
3. **Test set:** Rows where X_j is missing
4. Fit a model: $X_j \sim X_1, X_2, ..., X_{j-1}, X_{j+1}, ..., X_p$
5. Predict $\hat{X}_j$ for the test set
6. Fill in the missing values with $\hat{X}_j$

Models you can use:

- **Linear regression** (for numeric X_j)
- **Logistic regression** (for binary X_j)
- **KNN** (for any type)
- **Decision trees** (for any type)

Example:

KNN Imputation Variable Y (numeric) has missing values. Variable X (color) is fully observed. Using KNN with K=2:
For a missing value at color = "medium red":

- Find 2 nearest neighbors: "dark red" (Y=1) and "light red" (Y=0.5)
- Impute: $\hat{Y} = \frac{1+0.5}{2} = 0.75$

For a missing value at color = "yellow":

- Find 2 nearest neighbors: "orange" (Y=0.1) and "green" (Y=10)
- Impute: $\hat{Y} = \frac{0.1+10}{2} = 5.05$

3.4 Method 3: Imputation with Uncertainty

The problem with Method 2: **imputed values are too perfect**.

If you use linear regression to impute, all imputed values fall exactly on the regression line. But real data has scatter!

Warning

The Problem with Deterministic Imputation:
If you impute with $\hat{y}$ (the prediction), your imputed values have **no variance around the prediction**. This:

- Artificially strengthens relationships
- Makes your model overconfident
- Underestimates uncertainty in downstream analyses

Solution: Add randomness!

For linear regression:

1. Predict $\hat{y}$
2. Randomly sample a residual ϵ from training data (or from $N(0, \hat{\sigma}^2)$)
3. Impute: $\hat{y} + \epsilon$

For KNN:

1. Find K neighbors
2. Instead of averaging, **randomly sample one** of the K values
3. Impute that sampled value

For decision trees:

1. Traverse to the leaf node
2. Instead of using the leaf mean, **randomly sample one** observation from that leaf
3. Impute that sampled value

For classification:

1. Get predicted probability $\hat{p}$
2. **Flip a coin** with probability $\hat{p}$ to determine class
3. Don't just use the majority class!

3.5 Method 4: Iterative Imputation (Multiple Variables)

What if **multiple** variables have missing values?

You can't impute X_1 from X_2, X_3 if X_2 and X_3 also have missing values!

Solution: Iterate!

1. Initialize: Fill all missing values with simple imputation (mean/median)
2. Round 1: Use current X_2, X_3 to impute X_1
3. Round 1: Use current X_1, X_3 to impute X_2
4. Round 1: Use current X_1, X_2 to impute X_3
5. Repeat rounds until values converge (stop changing significantly)

```
from sklearn.impute import IterativeImputer

# Iteratively imputes using all other features
imputer = IterativeImputer(max_iter=10, random_state=42)
X_imputed = imputer.fit_transform(X)
```

3.6 Sklearn Imputation Tools

```
from sklearn.impute import SimpleImputer, KNNImputer, IterativeImputer
from sklearn.impute import MissingIndicator

# 1. Simple imputation (mean, median, most_frequent, constant)
```

```
simple = SimpleImputer(strategy='mean')

# 2. KNN imputation
knn = KNNImputer(n_neighbors=5)

# 3. Iterative imputation (most sophisticated)
iterative = IterativeImputer(max_iter=10, random_state=0)

# 4. Create missingness indicators
indicator = MissingIndicator()

# Usage
X_imputed = simple.fit_transform(X)
X_indicator = indicator.fit_transform(X)
```

4 Visualizing Black-Box Models

4.1 The Interpretation Problem

For **parametric models** (linear/logistic regression):

- We have coefficients β_j
- Interpretation: "One unit increase in X_j leads to β_j change in Y"
- These are "white-box" models—we can see inside

For **non-parametric models** (KNN, decision trees, random forests):

- No simple coefficients to interpret
- Complex decision rules or distance calculations
- These are "black-box" models—hard to see inside

Key Information

Why Do We Need Interpretation?
Even with black-box models, we need to:

- Understand what the model learned
- Check if relationships make domain sense
- Communicate findings to stakeholders
- Debug unexpected behavior

Solution: Plot the predictions!

4.2 Partial Dependence Plots (PDPs)

Definition:

Partial Dependence Plot A **Partial Dependence Plot (PDP)** shows the relationship between a feature X_j and the predicted outcome $\hat{Y}$, while holding all other features at fixed values (typically their medians).
It answers: "How does changing X_j affect predictions, controlling for everything else?"

How to create a PDP:

1. Choose the feature of interest, X_j
2. Fix all other features at their median values
3. Create a sequence of X_j values: $X_j = \{x_1, x_2, ..., x_n\}$
4. For each x_i, predict $\hat{Y}$ using the model
5. Plot X_j vs $\hat{Y}$

Example:

PDP for Heart Disease Prediction Model: KNN (K=50) predicting heart disease from max heart rate and other variables.

Create PDP for max heart rate:

1. Fix Age, Sex, RestBP, etc. at their medians
2. Vary MaxHR from 70 to 200
3. For each MaxHR value, predict probability of heart disease
4. Plot MaxHR vs predicted probability

Result: A curve showing that higher max heart rate → lower probability of heart disease (negative relationship).

```python
import numpy as np
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier

# Fit model
knn = KNeighborsClassifier(n_neighbors=50)
knn.fit(X_train, y_train)

# Create synthetic X values for max heart rate
maxhr_range = np.linspace(X['MaxHR'].min(), X['MaxHR'].max(), 100)

# Hold other variables at their medians
other_medians = X.drop('MaxHR', axis=1).median()

# Create prediction data
pred_data = pd.DataFrame({
    'MaxHR': maxhr_range,
    **{col: [val]*100 for col, val in other_medians.items()}
})

# Predict probabilities
probs = knn.predict_proba(pred_data)[:, 1]

# Plot PDP
import matplotlib.pyplot as plt
plt.plot(maxhr_range, probs)
plt.xlabel('Max Heart Rate')
plt.ylabel('Predicted P(Heart Disease)')
plt.title('Partial Dependence Plot')
```

4.3 Using PDPs to Detect Interactions

The power of PDPs comes from comparing them across different subgroups:

Key insight: If the PDP shape changes when you vary another feature, there's an **interaction**.

Example:

Detecting Age x MaxHR Interaction Create three PDPs for MaxHR:

1. Age = minimum (29 years)
2. Age = median (55 years)
3. Age = maximum (77 years)

If the three curves are different, it means the effect of MaxHR on heart disease depends on Age—an interaction!

Typical finding:

- At low MaxHR: All ages have high disease probability
- At high MaxHR: Young people benefit more (lower probability) than older people

Interpretation: High max heart rate (fitness) is more protective for younger patients.

Warning

Limitations of PDPs:

- Assumes features are independent (can be misleading if features are correlated)
- Shows average effect; may hide heterogeneity
- Doesn't show uncertainty in the relationship

For more sophisticated interpretation, consider SHAP values or ICE plots.

5 Summary and Key Takeaways

Key Summary

Missing Data:

- Missing data causes both technical problems (sklearn errors) and statistical problems (bias)
- Three types: MCAR (random), MAR (explained by observed data), MNAR (depends on missing value itself)
- You can never know for sure which type you have
- Simple approaches (drop rows, mean imputation) can introduce bias

Imputation Methods:

- **Missingness indicator:** Create binary flag for "was missing"
- **Model-based:** Predict missing values from other features
- **With uncertainty:** Add randomness to imputed values
- **Iterative:** For multiple variables with missingness

Model Visualization:

- Black-box models (KNN, trees) need visualization for interpretation
- **Partial Dependence Plots:** Show feature effect while holding others constant
- Compare PDPs across subgroups to detect interactions

6 Learning Checklist

☐ Can you explain what bias means in the context of estimation and prediction?

☐ Do you understand the three types of missing data (MCAR, MAR, MNAR)?

☐ Can you explain why you can never definitively know which type of missingness you have?

☐ Do you understand the problems with simple approaches (dropping rows, mean imputation)?

☐ Can you implement the missingness indicator variable approach?

☐ Do you understand model-based imputation and how to add uncertainty?

☐ Can you explain iterative imputation for multiple variables?

☐ Do you understand what a Partial Dependence Plot shows?

☐ Can you use PDPs to detect interactions between features?

7 Looking Ahead

Next lectures cover **ensemble methods**:

- **Bagging:** Bootstrap aggregating—averaging many trees
- **Random Forests:** Bagging + random feature selection
- **Boosting:** Sequential models that correct previous errors
- **Gradient Boosting:** XGBoost, LightGBM—top performers on tabular data