Name: Kshitij Hundre **Div: D15C** **Roll No: 18**

AIDS-1
Assignment 02
Q.1: Use the following data set for question 1
82, 66, 70, 59, 90, 78, 76, 95, 99, 84, 88, 76, 82, 81, 91, 64, 79, 76, 85, 90

1. Find the Mean (10pts)
2. Find the Median (10pts)
3. Find the Mode (10pts)
4. Find the Interquartile range (20pts)

Dataset:
82, 66, 70, 59, 90, 78, 76, 95, 99, 84, 88, 76, 82, 81, 91, 64, 79, 76, 85, 90

1. Mean
Step 1: Count the values
Total numbers = 20
Step 2: Calculate the sum
82 + 66 + 70 + 59 + 90 + 78 + 76 + 95 + 99 + 84 +
88 + 76 + 82 + 81 + 91 + 64 + 79 + 76 + 85 + 90 = 1611
Step 3: Apply formula
Mean = Sum of values ÷ Number of values
Mean = 80.55

2. Median
Step 1: Sort the data
59, 64, 66, 70, 76, 76, 76, 78, 79, 81,
82, 82, 84, 85, 88, 90, 90, 91, 95, 99
Step 2: Since there are 20 values (even), find the average of 10th and 11th values
10th value = 81
11th value = 82
Median = 81.5

3. Mode
Frequency count:

- 76 appears 3 times
- 82 appears 2 times
- 90 appears 2 times
- Others appear once

Mode = 76

4. Interquartile Range (IQR)

Step 1: Sorted data (again)
59, 64, 66, 70, 76, 76, 76, 78, 79, 81,
82, 82, 84, 85, 88, 90, 90, 91, 95, 99

Step 2: Divide into halves

- Lower half: 59, 64, 66, 70, 76, 76, 76, 78, 79, 81
- Upper half: 82, 82, 84, 85, 88, 90, 90, 91, 95, 99

Q1 = Median of lower half = (5th + 6th)/2 = (76 + 76)/2 = **76**
Q3 = Median of upper half = (5th + 6th)/2 = (88 + 90)/2 = **89**
IQR = Q3 - Q1 = 89 - 76 = 13

Final Answers

- **Mean: 81.05**
- **Median: 81.5**
- **Mode: 76**
- **Interquartile Range (IQR): 13**

Q.2 1) Machine Learning for Kids 2) Teachable Machine

1. For each tool listed above
   - identify the target audience
   - discuss the use of this tool by the target audience
   - identify the tool's benefits and drawbacks
2. From the two choices listed below, how would you describe each tool listed above? Why did you choose the answer?
   - Predictive analytic
   - Descriptive analytic
3. From the three choices listed below, how would you describe each tool listed above? Why did you choose the answer?

- Supervised learning
- Unsupervised learning
- Reinforcement learning

Tool 1: Machine Learning for Kids

Target Audience

- Primarily designed for kids, educators, and absolute beginners.
- Ideal for use in schools, coding clubs, and introductory AI workshops.

How They Use It

- Provides hands-on experience by letting users train models using examples.
- Users can build projects like:
  - Text classifiers (e.g., spam or not spam)
  - Image recognition (e.g., recognize types of food)
  - Chatbots using Natural Language Processing (NLP)

- The tool integrates with platforms like Scratch and Python to help visualize and use trained models in fun, interactive ways.

- Learning Goal: Understand the basic principles of AI and machine learning through experimentation.

Benefits

- Simple drag-and-drop interface, no coding required (optional Python).
- Supports real ML algorithms (powered by IBM Watson).
- Encourages experimentation and creativity.
- Great documentation and curriculum support for teachers.

Drawbacks

- Limited algorithm choices – not for advanced projects.
- May oversimplify concepts, missing depth for serious learners.
- Data privacy concerns when uploading images/texts (especially with kids).

Analytics Type
Predictive Analytic – Models are trained to make predictions based on labeled training data (e.g., predict whether an input is a cat or dog based on previous examples).

Learning Type
Supervised Learning – Users label examples during training (e.g., tagging images or texts), and the model learns to predict similar labels.

Tool 2: Teachable Machine (by Google)
Target Audience

- Geared toward students, artists, educators, and hobbyists.
- Suitable for people with no programming background.

How They Use It

- Users can create machine learning models by providing examples through webcam, microphone, or file uploads.
- Project types include:
    - Image classification (e.g., gestures, objects)
    - Audio recognition (e.g., speech commands)
    - Pose detection (e.g., yoga poses, dance moves)
- Learning Goal: Help users understand how training and prediction work in ML without needing code.

Benefits

- No installation or signup required, runs in the browser.
- Extremely intuitive interface – great for quick demos.
- Can export models to TensorFlow.js, TensorFlow Lite, or embed in websites/apps.
- Supports real-time feedback, making it engaging and interactive.

Drawbacks

- Limited customization – can't tweak model architecture or hyperparameters.
- Doesn't support advanced datasets or complex use cases.
- Data privacy issues if webcam/mic is used without consent.

Name: Kshitij Hundre **Div: D15C** **Roll No: 18**

Analytics Type
Predictive Analytic – Models are trained to make real-time predictions (e.g., detect a specific pose or sound based on prior training data).

Learning Type
Supervised Learning – Models are trained using labeled input (e.g., label each pose/image/audio before training).

Summary Table

Tool Name	Target Audience	Analytics Type	Learning Type	Benefits	Drawbacks
Machine Learning for Kids	Kids, educators, beginners	Predictive Analytic	Supervised Learning	Easy UI, Scratch integration, great for schools	Limited models, privacy concerns
Teachable Machine	Students, hobbyists	Predictive Analytic	Supervised Learning	No-code, real-time training, easy exports	No deep customization, privacy risks

Q.3 Data Visualization: Read the following two short articles:

- Read the article Kakande, Arthur. February 12. "What's in a chart? A Step-by-Step guide to Identifying Misinformation in Data Visualization." Medium
- Read the short web page Foley, Katherine Ellen. June 25, 2020. "How bad Covid-19 data visualizations mislead the public." Quartz
- Research a current event which highlights the results of misinformation based on data visualization. Explain how the data visualization method failed in presenting accurate information. Use newspaper articles, magazines, online news websites or any other legitimate and valid source to cite this example. Cite the news source that you found.

Ans.
Case Study: Misleading Electricity Price Chart in Spain
In a government presentation, a chart showing Spain's electricity prices from 2004 to 2013 was widely criticized for its misleading structure. The chart initially used yearly data from 2004 to 2012,
but suddenly switched to quarterly data in 2012–2013—without clearly indicating the change. This
design choice made it seem like electricity prices were stabilizing, even though they were not.
How the Data Visualization Method Failed:
● Inconsistent Time Intervals: The sudden shift from yearly to quarterly data created an illusion

of more frequent and smaller changes, deceiving the viewer into thinking prices were improving.

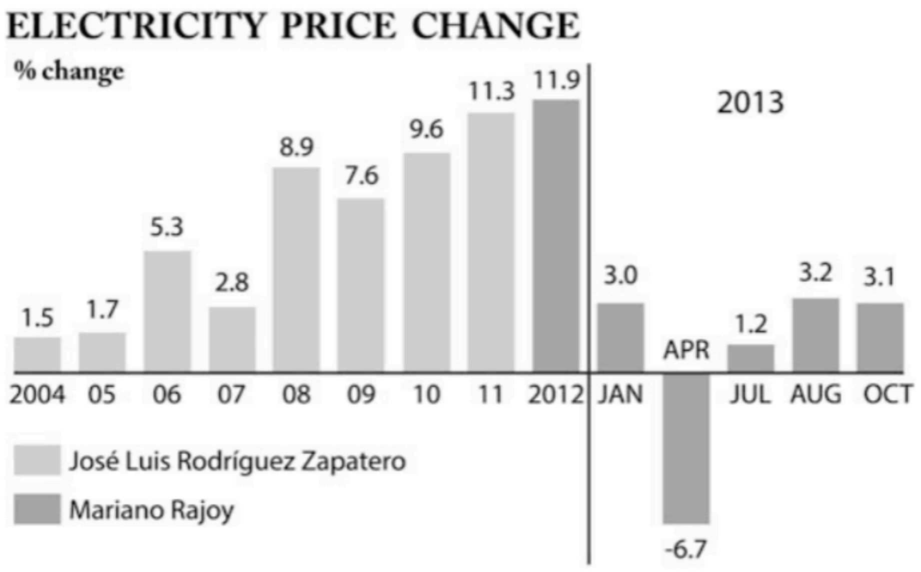
● Misleading Bar Heights: Bars representing just 3 months were nearly the same height as bars representing full years. This visually distorted the magnitude of the data.

● Lack of Transparency: No labels or annotations clarified the change in data intervals, causing further confusion and misleading interpretation.

Why This Is Problematic:

Data visualizations are meant to simplify and clarify information, but when time scales or measurements are altered without explanation, it can manipulate perception. Viewers may draw inaccurate conclusions, especially in politically or economically sensitive topics.

Visual Aid:



Misleading Electricity Price Chart link

Conclusion:

This example demonstrates the importance of consistent scales, accurate labeling, and clear communication in data visualizations. Even when data is accurate, poor visual representation can result in unintentional or intentional misinformation, affecting public opinion and policy decisions.

Name: Kshitij Hundre **Div: D15C** **Roll No: 18**

Q.4 Train Classification Model and visualize the prediction performance of trained model required information

- Data File: Classification data.csv
- Class Label: Last Column
- Use any Machine Learning model (SVM, Naïve Base Classifier)

Requirements to satisfy

- Programming Language: Python
- Class imbalance should be resolved
- Data Pre-processing must be used
- Hyper parameter tuning must be used
- Train, Validation and Test Split should be 70/20/10
- Train and Test split must be randomly done
- Classification Accuracy should be maximized
- Use any Python library to present the accuracy measures of trained model

Pima Indians Diabetes Database

Objective:

Train a classification model (SVM / Naïve Bayes) on the given dataset, resolve class imbalance, perform preprocessing and hyperparameter tuning, and evaluate its prediction performance.

Step 1: Import Required Libraries

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.svm import SVC
from imblearn.over_sampling import SMOTE
import seaborn as sns
import matplotlib.pyplot as plt
```

Step 2: Load the Dataset

```
df = pd.read_csv("Classification data.csv")
```

Step 3: Separate Features and Class Label

```
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
```

Step 4: Handle Class Imbalance

```
from imblearn.over_sampling import SMOTE
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X, y)
```

Step 5: Data Splitting (Train/Validation/Test = 70/20/10)

```
# Train (70%) and Temp (30%)
X_train, X_temp, y_train, y_temp = train_test_split(X_resampled, y_resampled, test_size=0.3, random_state=42, stratify=y_resampled)

# Validation (20%) and Test (10%) split from 30%
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=1/3, random_state=42, stratify=y_temp)
```

Step 6: Data Preprocessing

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_val_scaled = scaler.transform(X_val)
X_test_scaled = scaler.transform(X_test)
```

Step 7: Model Selection and Hyperparameter Tuning

```
params = {'C': [0.1, 1, 10], 'kernel': ['linear', 'rbf']}
svc = SVC()
grid = GridSearchCV(svc, params, cv=3)
grid.fit(X_train_scaled, y_train)
```

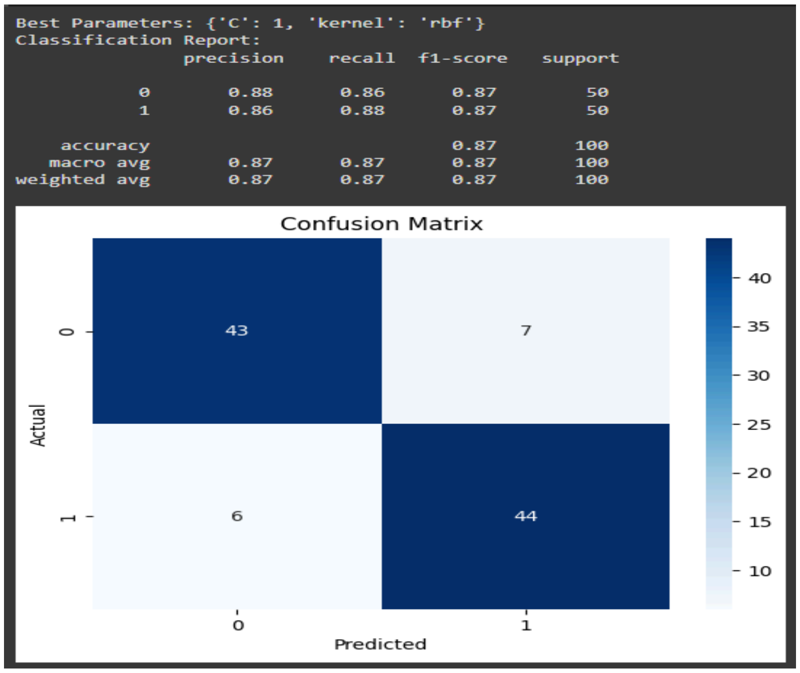
Step 8: Evaluate the Model

```
y_pred = grid.predict(X_test_scaled)

print("Best Parameters:", grid.best_params_)
print("Classification Report:\n", classification_report(y_test, y_pred))
```

Confusion Matrix

```
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```



```
Best Parameters: {'C': 1, 'kernel': 'rbf'}
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.86      0.87        50
           1       0.86      0.88      0.87        50

    accuracy                           0.87       100
   macro avg       0.87      0.87      0.87       100
weighted avg       0.87      0.87      0.87       100
```

Q.5 Train Regression Model and visualize the prediction performance of trained model

- Data File: Regression data.csv
- Independent Variable: 1st Column
- Dependent variables: Column 2 to 5

Use any Regression model to predict the values of all Dependent variables using values of 1st column.

Requirements to satisfy:

- Programming Language: Python
- OOP approach must be followed
- Hyper parameter tuning must be used
- Train and Test Split should be 70/30
- Train and Test split must be randomly done
- Adjusted R2 score should more than 0.99
- Use any Python library to present the accuracy measures of trained model

https://github.com/Sutanoy/Public-Regression-Datasets
https://raw.githubusercontent.com/selva86/datasets/master/BostonHousing.csv

- URL:
  https://archive.ics.uci.edu/ml/machine-learning-databases/00477/Real%20estate%20valuation%20data%20set.xlsx

(Refer any one)

Objective

To train a regression model using Ridge Regression with Polynomial Feature Expansion to predict real estate prices based on several features (like distance to MRT station, number of convenience stores, etc.). We aim to:

- Tune hyperparameters (alpha and polynomial degree)
- Use a modular, object-oriented design
- Evaluate using R², Adjusted R², and MSE
- Visualize predicted vs actual prices

Step-by-Step Explanation

1. Importing Required Libraries

```
import pandas as pd, numpy as np
from sklearn.linear_model import Ridge
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import r2_score, mean_squared_error
import matplotlib.pyplot as plt
import seaborn as sns
```

We use:

- Pipeline to streamline polynomial features → standardization → ridge regression.
- GridSearchCV for hyperparameter tuning
- r2_score, mean_squared_error for evaluation

2. Defining the RegressionModel Class

```
class RegressionModel:
   def __init__(self, model_pipeline, param_grid):
       ...
```

- Encapsulates model training, evaluation, and hyperparameter tuning.
- Reusable and extensible for other regression problems.

3. Loading the Dataset

```
url =
"https://archive.ics.uci.edu/ml/machine-learning-databases/00477/Real%20estate%20valuation%20data%20set.xlsx"
df = pd.read_excel(url)
```

The dataset contains real estate records including:

- Distance to MRT station
- Number of nearby convenience stores
- Age of building
- Geographic coordinates

4. Data Preprocessing

```
df = df.drop(columns=['No'])  # Drop irrelevant index
X = df.drop(columns=['Y house price of unit area'])  # Features
y = df['Y house price of unit area']  # Target
```

We define:

- X: all columns except house price
- y: the target variable (house price)

5. Train-Test Split

```
X_train, X_test, y_train, y_test = train_test_split(...)
```

- 70% training data, 30% testing
- random_state=42 ensures reproducibility

6. Model Pipeline & Hyperparameter Grid

```
pipeline = Pipeline([
    ('poly', PolynomialFeatures()),
    ('scaler', StandardScaler()),
    ('ridge', Ridge())
])
```

- Pipeline Components:
  - poly: adds non-linearity (degree=2 or more)
  - scaler: standardizes features (important for ridge!)
  - ridge: regularized regression
- Parameter Grid:

```
param_grid = {
    'poly__degree': [2, 3, 4],
    'ridge__alpha': [0.1, 1, 10, 100]
}
```

We let GridSearchCV try different combinations of:

- Polynomial degrees: 2 to 4
- Ridge regularization strengths (alpha): 0.1 to 100

7. Training and Evaluation

```
best_model = reg_model.train(X_train, y_train)
y_test_actual, y_pred = reg_model.evaluate(best_model, X_test, y_test)
```

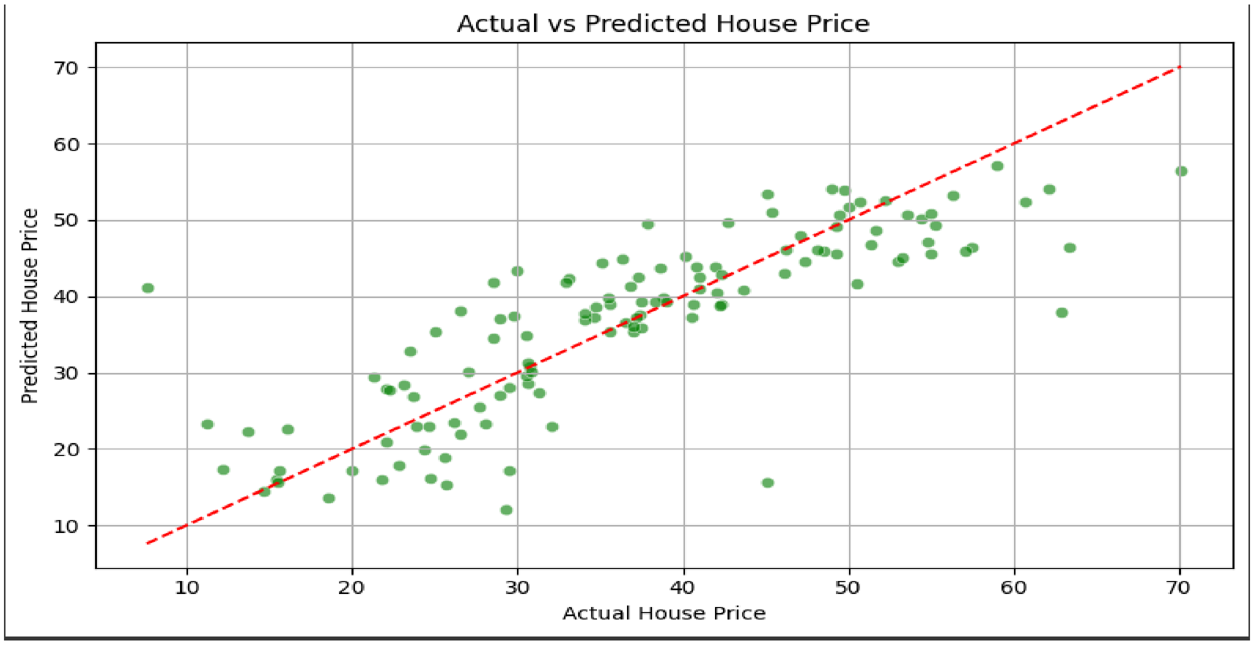
- The model is trained with 5-fold cross-validation
- Best estimator is used for prediction
- We calculate:
  - R^2: proportion of variance explained
  - Adjusted R^2: penalizes for extra features
  - MSE: average squared prediction error

8. Visualization

```
sns.scatterplot(x=y_test_actual, y=y_pred)
```

- Compares actual vs predicted prices
- Red dashed line shows ideal predictions (perfect match)

Name: Kshitij Hundre **Div: D15C** **Roll No: 18**



Final Results:
Best Params: {'poly__degree': 2, 'ridge__alpha': 1}
R² Score: 0.6552
Adjusted R² Score: 0.6376
Mean Squared Error: 57.6670

- The model captures ~66% of the variance in house prices.
- There's room for improvement, but this is reasonable for real-world data.
- Adjusted R² shows performance after accounting for model complexity.

Why we May Not Reach R² > 0.99

- Real-world datasets include noise, missing features, and non-linear interactions.
- Polynomial features help but too high a degree → overfitting.
- Ridge helps reduce overfitting, but can't add missing signal.

Q.6 What are the key features of the wine quality data set? Discuss the importance of each feature in predicting the quality of wine? How did you handle missing data in the wine quality data set during the feature engineering process? Discuss the advantages and disadvantages of different imputation techniques. (Refer dataset from Kaggle).

1. Introduction
The Wine Quality dataset available on Kaggle is a well-known dataset used for regression and classification tasks. It includes physicochemical properties of red or white vinho verde wine samples, along with a quality score rated by wine tasters.
Dataset Link:
[Wine Quality Dataset on Kaggle (Red and White)]

Name: Kshitij Hundre **Div: D15C** **Roll No: 18**

2. Key Features of the Wine Quality Dataset

Feature	Description
fixed acidity	Tartaric acid content. Too much affects the taste, while too little may reduce the wine's stability.
volatile acidity	Acetic acid content. High levels give an unpleasant vinegar taste.
citric acid	Adds freshness and flavor. Small amounts are desirable.
residual sugar	Sugar remaining after fermentation. Influences sweetness and body of wine.
chlorides	Salt content. High amounts can negatively affect taste.
free sulfur dioxide	Prevents microbial growth. Too much affects flavor.
total sulfur dioxide	Sum of free and bound forms. High levels can cause off-flavors.
density	Indicates sugar and alcohol content. Important for fermentation control.
pH	Acidity level. Affects freshness, taste, and preservation.
sulphates	Wine preservative. Also affects flavor.
alcohol	One of the strongest predictors of quality. Higher alcohol usually improves perception.
quality (target)	Score between 0–10 given by tasters. This is what we aim to predict.

3. Importance of Features in Predicting Wine Quality
Based on correlation studies and model performance, some features contribute more to wine quality prediction:

- Alcohol: Strongest positive correlation with quality.
- Volatile Acidity: Strong negative impact on taste and quality.
- Sulphates: Moderate positive impact; enhances flavor.
- Citric Acid: Mild but noticeable effect on freshness.
- Density: Inversely related to alcohol; indirectly affects quality.

Using models like Random Forest or XGBoost, feature importance plots show alcohol, volatile acidity, sulphates, and citric acid as top predictors.

Handling Missing Data During Feature Engineering

Name: Kshitij Hundre **Div: D15C** **Roll No: 18**

The wine quality dataset from Kaggle does not contain missing values in its original form. However, if encountered (e.g., in modified or extended versions), these are the standard approaches:
Techniques for Handling Missing Data

Method	Description	Advantages	Disadvantages
Drop rows	Remove rows with missing values	Simple, avoids introducing bias	Risk of losing valuable data
Mean/Median Imputation	Replace missing values with mean/median	Easy to implement, preserves dataset size	Can distort distribution; ignores variability
Mode Imputation	Replace with most frequent value (used for categorical data)	Maintains consistency	Can bias the data if mode is too dominant
KNN Imputation	Use nearest neighbors to predict missing values	Considers feature relationships	Computationally expensive; sensitive to outliers
Multivariate Imputation (MICE)	Iteratively predicts missing data using regression models	Accurate and flexible	Complex and slow on large datasets

Recommended Technique:

- For this dataset, mean or median imputation would be ideal if data were missing, as most features are continuous.
- For advanced cases, KNN imputation or MICE could improve model performance by preserving relationships among variables.

Conclusion

- The most important predictors of wine quality are alcohol, volatile acidity, and sulphates.
- Although the original dataset is clean, it is essential to understand missing value imputation techniques.
- The choice of imputation method depends on the nature of data, the amount of missingness, and the balance between performance and simplicity.