Problem

The breast cancer data set `breast-cancer-wisconsin.data.txt` from https://archive.ics.uci.edu/dataset/15/breast+cancer+wisconsin+original (description available at the same URL) has missing values.

1. Use the mean/mode imputation method to impute values for the missing data.
2. Use regression to impute values for the missing data.
3. Use regression with perturbation to impute values for the missing data.
4. (Optional) Compare the results and quality of classification models (e.g., SVM, KNN) build using
   (1) the data sets from questions 1,2,3;
   (2) the data that remains after data points with missing values are removed; and
   (3) the data set when a binary variable is introduced to indicate missing values.

Methodology

Imputation Methods

Method 1: Mean Imputation

Missing values were replaced with the arithmetic mean of all observed Bare Nuclei values. This is the simplest approach and assumes missing data is missing completely at random (MCAR).

Method 2: Regression Imputation

A linear regression model was trained using the 8 other clinical features as predictors to estimate Bare Nuclei values. The model learned relationships from complete cases and predicted values for missing cases based on their other feature values.

Method 3: Regression with Perturbation

Building on Method 2, random noise was added to predictions to account for uncertainty. The noise was drawn from a normal distribution with mean 0 and standard deviation equal to the residual standard error from the regression model. This prevents underestimation of variance in the imputed dataset.

Method 4: Listwise Deletion

All rows containing any missing values were removed, reducing the dataset from 699 to 683 observations. This is the traditional approach but results in information loss.

Method 5: Missing Indicator Method

A binary indicator variable was created to flag which observations originally had missing values. Missing Bare Nuclei values were then filled with the mean. This approach allows classifiers to potentially learn patterns associated with missingness itself.

Solution

Step-1: Import necessary library and load the data from given url.

```
C: > Users > Varshini > Documents > R > HW10 > 14.1.py > ...
  1  #load library
  2  import pandas as pd
  3  import numpy as np
  4  from sklearn.linear_model import LinearRegression
  5  from sklearn.model_selection import train_test_split
  6  from sklearn.neighbors import KNeighborsClassifier
  7  from sklearn.svm import SVC
  8  from sklearn.metrics import accuracy_score
  9
 10  #loading data
 11  url = "https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/breast-cancer-wisconsin.data"
 12  columns = ["ID", "ClumpThickness", "CellSize", "CellShape",
 13             "MarginalAdhesion", "EpithelialSize", "BareNuclei",
 14             "BlandChromatin", "NormalNucleoli", "Mitoses", "Class"]
 15  bcDf = pd.read_csv(url, names=columns)
 16  print(bcDf.head())
```

Output

```
PS C:\Users\Varshini> & C:/Users/Varshini/anaconda3/python.exe c:/Users/Varshini/Documents/R/HW10/14.1.py
        ID  ClumpThickness  CellSize  CellShape  MarginalAdhesion  EpithelialSize BareNuclei  BlandChromatin  NormalNucleoli  Mitoses  Class
0  1000025               5         1          1                 1               2         1               3               1        1      2
1  1002945               5         4          4                 5               7        10               3               2        1      2
2  1015425               3         1          1                 1               2         2               3               1        1      2
3  1016277               6         8          8                 1               3         4               3               7        1      2
4  1017023               4         1          1                 3               2         1               3               1        1      2
```

Step-2: Replacing null values with NaN. Then we are checking datatype for all columns. BareNuclei column is text type because of the '?' symbol that was in data. We are converting it into numeric column.

```
#replace null values with NaN
bcDf = bcDf.replace("?", np.NaN)

#checking datatype for all columns
print(bcDf.dtypes)

#converting BareNuclei column from text to numeric type
bcDf['BareNuclei'] = pd.to_numeric(bcDf['BareNuclei'])
```

Output

```
ID                 int64
ClumpThickness     int64
CellSize           int64
CellShape          int64
MarginalAdhesion   int64
EpithelialSize     int64
BareNuclei        object
BlandChromatin     int64
NormalNucleoli     int64
Mitoses            int64
Class              int64
```

->

```
ID                 int64
ClumpThickness     int64
CellSize           int64
CellShape          int64
MarginalAdhesion   int64
EpithelialSize     int64
BareNuclei       float64
BlandChromatin     int64
NormalNucleoli     int64
Mitoses            int64
Class              int64
dtype: object
```

Step-3: Finding how many missing values we have in each column. Only BareNuclei has missing values.

```
#finding no of null values in each column
print(bcDf.isnull().sum())

print(f"\nNumber of missing values in BareNuclei: {bcDf['BareNuclei'].isna().sum()}")
```

Output

```
ID                  0
ClumpThickness      0
CellSize            0
CellShape           0
MarginalAdhesion    0
EpithelialSize      0
BareNuclei         16
BlandChromatin      0
NormalNucleoli      0
Mitoses             0
Class               0
dtype: int64

Number of missing values in BareNuclei: 16
```

Step-4: Finding out mean of BareNuclei column and imputing it in missing values.

```
#Method 1 - MEAN IMPUTATION

#Make copy of the data
df1 = bcDf.copy()

#calculate the mean of BareNuclei column
mean_value = df1['BareNuclei'].mean()
print(f"Mean value for BareNuclei: {mean_value:.2f}")

#missing value before imputation
print("Missing values before imputation:", df1['BareNuclei'].isna().sum())

#fill missing values with the mean
df1['BareNuclei'].fillna(mean_value, inplace=True)

#missing value after imputation
print("Missing values after imputation:", df1['BareNuclei'].isna().sum())
```

Output

```
Mean value for BareNuclei: 3.54
Missing values before imputation: 16
Missing values after imputation: 0
```

Step-5: In regression imputation, it at complete rows to learn patterns. Then apply these patterns to predict the missing values.

```
# METHOD 2 - REGRESSION IMPUTATION

#Make a copy of the data
df2 = bcDf.copy()

#Get rows WITHOUT missing values
complete_data = df2[df2['BareNuclei'].notna()]

#Prepare the data for regression
feature_cols = ["ClumpThickness", "CellSize", "CellShape",
                "MarginalAdhesion", "EpithelialSize", "BlandChromatin",
                "NormalNucleoli", "Mitoses"]

X_complete = complete_data[feature_cols]
y_complete = complete_data['BareNuclei']

#missing value before imputation
print("Missing values before imputation:", df2['BareNuclei'].isna().sum())

#Train a regression model
model = LinearRegression()
model.fit(X_complete, y_complete)

#Find rows WITH missing values
missing_mask = df2['BareNuclei'].isna()
X_missing = df2[missing_mask][feature_cols]

#Predict the missing values
predicted_values = model.predict(X_missing)
print("Predicted BareNuclei values for missing rows:")
print(predicted_values)

#Fill in the missing values
df2.loc[missing_mask, 'BareNuclei'] = predicted_values

#missing value after imputation
print("Missing values after imputation:", df2['BareNuclei'].isna().sum())
```

Output

```
Missing values before imputation: 16
Predicted BareNuclei values for missing rows:
[5.36606663 8.22591014 0.88928053 1.66055745 1.08993002 2.22087363
 2.78188891 1.76056172 2.12086936 5.84594766 0.97967265 2.39182817
 5.54199423 1.76056172 0.88928053 0.56870337]
Missing values after imputation: 0
```

Step-6: Same as Method 2 (or Step 5) – but we + Noise to the predicted values. It is a slightly different variation of Step-5.

```
# METHOD 3 - REGRESSION WITH PERTURBATION

#Make copy of the data
df3 = bcDf.copy()

#missing value before imputation
print("Missing values before imputation:", df3['BareNuclei'].isna().sum())

# Use the same model from Method 2
# Calculate predictions
predicted_values = model.predict(X_missing)

#Calculate the prediction errors on complete data
predictions_complete = model.predict(X_complete)
residuals = y_complete - predictions_complete
std_error = np.std(residuals)

print(f"Standard error of predictions: {std_error:.2f}")

# Add random noise to predictions
noise = np.random.normal(0, std_error, size=len(predicted_values))
predicted_with_noise = predicted_values + noise

# Fill in the missing values
df3.loc[missing_mask, 'BareNuclei'] = predicted_with_noise

#missing value after imputation
print("Missing values after imputation:", df2['BareNuclei'].isna().sum())
```

Output

```
Missing values before imputation: 16
Standard error of predictions: 2.26
Missing values after imputation: 0
```

Step-7: In listwise deletion, we remove all rows with missing values. This results in data reduction.

```
#METHOD 4 - LISTWISE DELETION

df4 = bcDf.dropna()
print(f"Dataset reduced from {len(bcDf)} to {len(df4)} rows")
```

Output:

```
Dataset reduced from 699 to 683 rows
```

Step-8: In this method, we are adding a new column – BareNuclei_Missing which has Boolean value (1- Missing data, 0- Not missing data).

```
#METHOD 5 - MISSING INDICATOR

#Make a copy of the data
df5 = bcDf.copy()

#Create a new column: 1 if missing, 0 if not
df5['BareNuclei_Missing'] = df5['BareNuclei'].isna().astype(int)

#Fill missing values with mean
df5['BareNuclei'].fillna(mean_value, inplace=True)

print(df5.head())
```

Output

	ID	ClumpThickness	CellSize	CellShape	MarginalAdhesion	EpithelialSize	BareNuclei	BlandChromatin	NormalNucleoli	Mitoses	Class	BareNuclei_Missing
0	1000025	5	1	1	1	2	1.0	3	1	1	2	0
1	1002945	5	4	4	5	7	10.0	3	2	1	2	0
2	1015425	3	1	1	1	2	2.0	3	1	1	2	0
3	1016277	6	8	8	1	3	4.0	3	7	1	2	0
4	1017023	4	1	1	3	2	1.0	3	1	1	2	0

Step-9: Comparing accuracies in all the methods so far using KNN & SVM to see the best method to impute missing values.

```
print("COMPARING CLASSIFICATION PERFORMANCE")

def evaluate_dataset(dataset, method_name):
    df = bcDf.copy()
    # Remove ID column
    df = dataset.drop('ID', axis=1)
    # Split into features (X) and target (y)
    X = df.drop('Class', axis=1)
    y = df['Class']
    # Split into train and test sets (80% train, 20% test)
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, random_state=42
    )

    # KNN Classifier
    knn = KNeighborsClassifier(n_neighbors=5)
    knn.fit(X_train, y_train)
    knn_pred = knn.predict(X_test)
    knn_accuracy = accuracy_score(y_test, knn_pred)
    # SVM Classifier
    svm = SVC()
    svm.fit(X_train, y_train)
    svm_pred = svm.predict(X_test)
    svm_accuracy = accuracy_score(y_test, svm_pred)

    return knn_accuracy, svm_accuracy
```

```
# Evaluate each method
print("\nResults (Accuracy scores):")
print("-" * 50)

knn1, svm1 = evaluate_dataset(df1, "Mean Imputation")
print(f"Method 1 (Mean)          - KNN: {knn1:.3f}, SVM: {svm1:.3f}")

knn2, svm2 = evaluate_dataset(df2, "Regression")
print(f"Method 2 (Regression)    - KNN: {knn2:.3f}, SVM: {svm2:.3f}")

knn3, svm3 = evaluate_dataset(df3, "Regression+Noise")
print(f"Method 3 (Reg+Noise)     - KNN: {knn3:.3f}, SVM: {svm3:.3f}")

knn4, svm4 = evaluate_dataset(df4, "Listwise Deletion")
print(f"Method 4 (Deletion)      - KNN: {knn4:.3f}, SVM: {svm4:.3f}")

knn5, svm5 = evaluate_dataset(df5, "Missing Indicator")
print(f"Method 5 (Indicator)     - KNN: {knn5:.3f}, SVM: {svm5:.3f}")
```

Output

```
Results (Accuracy scores):
--------------------------------------------------
Method 1 (Mean)          - KNN: 0.986, SVM: 0.964
Method 2 (Regression)    - KNN: 0.979, SVM: 0.971
Method 3 (Reg+Noise)     - KNN: 0.979, SVM: 0.964
Method 4 (Deletion)      - KNN: 0.949, SVM: 0.949
Method 5 (Indicator)     - KNN: 0.986, SVM: 0.964
```

Results & Discussion

Classification Performance

Method	KNN Accuracy	SVM Accuracy	Dataset Size
Mean Imputation	0.986	0.964	699
Regression	0.979	0.971	699
Regression + Noise	0.979	0.964	699
Listwise Deletion	0.949	0.949	683
Missing Indicator	0.986	0.964	699

For general practice with missing data:

- Examine missingness patterns before choosing a method
  (Missing Completely at Random, Missing At Random, Missing Not At Random)
- Consider multiple imputation for higher missingness rates
- Always compare results across multiple imputation strategies
- Report the amount and pattern of missing data in any analysis

Conclusion

For the Breast Cancer Wisconsin dataset, regression-based imputation methods slightly outperformed simpler approaches while maintaining the full dataset. However, the practical differences were small due to the low proportion of missing data. The choice of imputation method becomes increasingly critical as missingness rates increase and when data is not missing completely at random.