Assignment-1: Data Imputation Techniques

Name:	Aryan Soni
Roll No:	22111009
Semester:	5th
Subject:	AI
Department:	Biomedical Engineering
College:	NIT Raipur

1 Abstract

Data imputation is crucial for missing data in various datasets. This work outlines five common data imputation techniques, with descriptions, sample Python code, and case studies.

2 Central Tendency based Imputation:

2.1 Description

Mean or median imputation involves replacing missing values with the mean or median of the observed data or sample value.

2.2 Python Code

```
import pandas as pd
import numpy as np

# Sample DataFrame with missing values
data = {'A': [1, 2, np.nan, 4, 5], 'B': [np.nan, 2, 3, 4, 5]}
df = pd.DataFrame(data)

# Mean Imputation
df_mean_imputed = df.copy()
df_mean_imputed['A'].fillna(df_mean_imputed['A'].mean(), inplace=
    True)
df_mean_imputed['B'].fillna(df_mean_imputed['B'].mean(), inplace=
    True)

print("Mean Imputed DataFrame:")
print(df_mean_imputed)

# Median Imputation
```

```
df_median_imputed = df.copy()
df_median_imputed['A'].fillna(df_median_imputed['A'].median(),
    inplace=True)
df_median_imputed['B'].fillna(df_median_imputed['B'].median(),
    inplace=True)

print("Median Imputed DataFrame:")
print(df_median_imputed)
```

2.3 Case Study

Title: *Impact of mean and median imputation on the accuracy of predictive models for healthcare data*

Summary: A study on healthcare data compared the accuracy of predictive models using mean and median imputation. Median imputation provided better accuracy due to its robustness against outliers.In a study focusing on healthcare data, researchers evaluated the effectiveness of mean and median imputation methods on the accuracy of predictive models. The dataset contained patient records with various missing values replaced by NaN. Researchers implemented mean and median imputation to handle the missing data and then trained predictive models to forecast patient outcomes and get very powerful conclusions over the data sets.

Results: The median imputation method outperformed mean imputation in terms of accuracy. Median imputation was particularly effective in handling skewed (packed at either side) data distributions and outliers, which are common in healthcare datasets.

Conclusion: Median imputation is recommended for healthcare data as it is more robust to outliers and non-normal data distributions.

3 Manifold Imputation

3.1 Description

Manifold imputation generates several complete datasets by imputing missing values multiple times, analyzing each dataset separately, and combining the results effectively.

3.2 Python Code

```
import pandas as pd
import numpy as np
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer

# Sample DataFrame with missing values
data = {'A': [1, 2, np.nan, 4, 5], 'B': [np.nan, 2, 3, 4, 5]}
```

```
df = pd.DataFrame(data)

# Multiple Imputation using Iterative Imputer
imputer = IterativeImputer(max_iter=10, random_state=0)
df_multiple_imputed = pd.DataFrame(imputer.fit_transform(df),
    columns=df.columns)

print("Multiple Imputed DataFrame:")
print(df_multiple_imputed)
```

3.3 Case Study

Title: *Manifold Imputation for Missing Data in Large-Scale Surveys*

Summary: This study showed that manifold imputation provided more reliable estimates and better captured variability in large-scale survey data compared to single imputation methods. This study examined the application of manifold imputation techniques on a large-scale survey dataset with extensive missing values. Researchers aimed to improve the reliability of statistical inferences by employing manifold imputation, which generates multiple data-logs and combines the results to predict the missing values.

Results: Manifold imputation provided more precise and reliable estimates compared to single imputation methods. It successfully captured the variability and uncertainty associated with missing data, leading to better model performance. Conclusion: Manifold imputation is highly effective for large-scale surveys, providing more robust and comprehensive results than traditional single imputation techniques.

4 K-Nearest Neighbors (KNN) Imputation

4.1 Description

KNN imputation uses the k-nearest neighbors algorithm to find the nearest neighbors based on the Euclidean distance and imputes the missing values with the average (or mode) of these neighbors.

4.2 Python Code

```
import pandas as pd
import numpy as np
from sklearn.impute import KNNImputer

# Sample DataFrame with missing values
data = {'A': [1, 2, np.nan, 4, 5], 'B': [np.nan, 2, 3, 4, 5]}
df = pd.DataFrame(data)

# KNN Imputation
imputer = KNNImputer(n_neighbors=2)
```

```
df_knn_imputed = pd.DataFrame(imputer.fit_transform(df), columns=df
    .columns)

print("KNN Imputed DataFrame:")
print(df_knn_imputed)
```

4.3 Case Study

Title: *KNN-based Imputation Techniques for Agricultural Yield Data*

Summary: In agricultural studies, KNN imputation was used to predict missing crop yield values, outperforming mean imputation and significantly improving accuracy of yield predictions.

In agricultural research, accurate yield prediction is crucial. This case study explored the use of KNN imputation for handling missing values in crop yield datasets. Researchers compared KNN imputation with mean imputation to evaluate the improvement in prediction accuracy.

Results: KNN imputation significantly outperformed mean imputation, resulting in more accurate yield predictions. The method effectively utilized the similarity between neighboring data points to impute missing values, preserving the inherent patterns in the dataset.

Conclusion: KNN imputation is a powerful technique for agricultural datasets, enhancing the accuracy of yield predictions and providing better support for decision-making in agriculture.

5 Regression Imputation

5.1 Description

Regression imputation uses a regression model to predict missing values based on other observed variables.

5.2 Python Code

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression

# Sample DataFrame with missing values
data = {'A': [1, 2, np.nan, 4, 5], 'B': [np.nan, 2, 3, 4, 5]}
df = pd.DataFrame(data)

# Separate complete and incomplete cases
complete_cases = df.dropna()
incomplete_cases = df[df.isnull().any(axis=1)]

# Train regression model
model = LinearRegression()
```

```
model.fit(complete_cases[['B']], complete_cases['A'])

# Predict missing values
incomplete_cases['A'] = model.predict(incomplete_cases[['B']])

# Combine the complete and imputed cases
df_regression_imputed = pd.concat([complete_cases, incomplete_cases
    ]).sort_index()

print("Regression Imputed DataFrame:")
print(df_regression_imputed)
```

5.3 Case Study

Title: *Using Regression Imputation to Handle Missing Data in Customer Satisfaction Surveys*

Summary: A retail company applied regression imputation to handle missing responses in customer satisfaction surveys, accurately imputing missing satisfaction scores and improving feedback analysis. A retail company faced challenges with missing responses in customer satisfaction surveys. To address this, researchers applied regression imputation, using observed responses to predict missing values. The goal was to accurately impute missing satisfaction scores and improve the overall analysis of customer feedback.

Results: Regression imputation successfully predicted missing satisfaction scores based on other observed variables, such as demographic information and purchase history. This method resulted in a more complete dataset, enabling more accurate and meaningful analysis of customer satisfaction. Conclusion: Regression imputation is effective for survey data, particularly when there is a strong relationship between the observed and missing variables. It enhances the quality of customer feedback analysis and supports better business decision-making

6 Moment Rendering Imputation

6.1 Description

This imputation involves using techniques specific to time series data, such as forward fill, backward fill, or interpolation, to impute missing values.

6.2 Python Code

```
import pandas as pd
import numpy as np

# Sample time series data with missing values
data = {'A': [1, np.nan, 3, np.nan, 5]}
```

```
df = pd.DataFrame(data)

# Forward Fill Imputation
df_ffill = df.copy()
df_ffill['A'].fillna(method='ffill', inplace=True)

print("Forward Fill Imputed DataFrame:")
print(df_ffill)

# Backward Fill Imputation
df_bfill = df.copy()
df_bfill['A'].fillna(method='bfill', inplace=True)

print("Backward Fill Imputed DataFrame:")
print(df_bfill)

# Linear Interpolation
df_interpolated = df.copy()
df_interpolated['A'].interpolate(method='linear', inplace=True)

print("Linear Interpolated DataFrame:")
print(df_interpolated)
```

6.3 Case Study

Title: *Imputing Missing Values in Energy Consumption Data Using Time Series Methods*

Summary: Researchers used this imputation methods like linear interpolation and seasonal decomposition on energy consumption data from smart meters, providing accurate imputation and preserving temporal patterns crucial for load forecasting. Researchers explored various imputation methods to handle missing values in energy consumption data collected from smart meters. The study aimed to maintain the temporal patterns and improve the accuracy of load forecasting models. Methods Used: Researchers implemented forward fill, backward fill, and linear interpolation to impute missing values. Results: Linear interpolation provided the most accurate imputation results, preserving the temporal dynamics and seasonal patterns essential for load forecasting. Forward fill and backward fill methods were also effective but less accurate in capturing the nuances of the time series data. Conclusion: Moment Rendering imputation methods, particularly linear interpolation, are crucial for maintaining the integrity of temporal data in energy consumption datasets. These methods ensure accurate load forecasting and support efficient energy management.

GitHub Repository: Click here to re-direct to GitHub