Data Analysis and Missing Value Imputation

ROHIT RAGHUWANSHI

February 5, 2025

1 Introduction

This document presents a Python script for handling missing values, training a Linear Regression model, and evaluating predictions using visualizations.

2 Python Code

The following Python script performs the following tasks:

- Reads the dataset and converts data to numeric format.
- Introduces missing values randomly (20% of the dataset).
- Splits data into known and missing parts.
- Trains a Linear Regression model to predict missing values.
- Evaluates model performance using Mean Squared Error (MSE).
- Visualizes results with scatter plots, histograms, and a missing data matrix.

```python
# Install necessary libraries (if not installed)
# !pip install openpyxl missingno

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer

# Load the dataset (Replace with actual file path)
df = pd.read_excel("DSAICourseInterestRelevanceSurvey.xlsx")  # Adjust the file
↪  path as needed
```

```
# Convert to numeric, handling errors
df = df.apply(pd.to_numeric, errors='coerce')

# Function to introduce missing values
def introduce_missing_values(df, missing_percentage=20):
    df_missing = df.copy()
    total_values = df_missing.size
    missing_count = int((missing_percentage / 100) * total_values)

    np.random.seed(42)
    missing_indices = np.random.choice(total_values, missing_count,
    ↪  replace=False)

    row_indices, col_indices = np.unravel_index(missing_indices,
    ↪  df_missing.shape)
    df_missing.values[row_indices, col_indices] = np.nan

    return df_missing

# Introduce 20% missing values
df_missing = introduce_missing_values(df, 20)

# Splitting data into known and unknown (missing) values
known_data = df_missing[df_missing.notna().any(axis=1)]
missing_data = df_missing[df_missing.isnull().any(axis=1)]

# Preparing features (X) and target (y) for model training
X_known = known_data.dropna(axis=1, how='all')  # Drop completely empty columns
y_known = X_known.mean(axis=1)  # Target variable: row mean (as an
↪  approximation)

# Splitting into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X_known, y_known,
↪  test_size=0.2, random_state=42)

# Impute missing values using the mean
imputer = SimpleImputer(strategy='mean')
X_train = imputer.fit_transform(X_train)
X_test = imputer.transform(X_test)

# Train Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Predict missing values
X_missing = missing_data.dropna(axis=1, how='all')
predicted_values = model.predict(X_missing.fillna(X_train.mean()))

# Replace missing values with predictions
df_filled = df_missing.fillna(pd.Series(predicted_values,
↪  index=df_missing.index[df_missing.isnull().any(axis=1)]))

# Evaluate using MSE
mse = mean_squared_error(y_test, model.predict(X_test))
print(f"Mean Squared Error (MSE): {mse:.4f}")

# 1. Actual vs Predicted values
```

```
plt.figure(figsize=(8, 6))
plt.scatter(y_test, model.predict(X_test), alpha=0.7, color="blue",
↪  label="Predicted vs Actual")
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
↪  color="red", linestyle="dashed")
plt.xlabel("Actual Values")
plt.ylabel("Predicted Values")
plt.title("Actual vs. Predicted Values")
plt.legend()
plt.show()

# 2. Residual plot
residuals = y_test - model.predict(X_test)
plt.figure(figsize=(8, 6))
plt.scatter(model.predict(X_test), residuals, alpha=0.7, color="blue")
plt.axhline(y=0, color="red", linestyle="dashed")
plt.xlabel("Predicted Values")
plt.ylabel("Residuals")
plt.title("Residual Plot")
plt.show()

# 3. Distribution of actual and predicted values
plt.figure(figsize=(8, 6))
sns.histplot(y_test, label="Actual", color="blue", kde=True)
sns.histplot(model.predict(X_test), label="Predicted", color="red", kde=True)
plt.xlabel("Values")
plt.ylabel("Frequency")
plt.title("Distribution of Actual and Predicted Values")
plt.legend()
plt.show()

print(df_missing.isnull().sum().sum())  # Total number of missing values
print(df_missing.isnull().sum())  # Missing values per column

# Missing data matrix
plt.figure(figsize=(18, 8))
msno.matrix(df_missing, sparkline=False, fontsize=12)
plt.title("Missing Data Matrix", fontsize=14)
plt.show()
```

3 Results and Discussion

3.1 Mean Squared Error

The performance of the model is evaluated using the Mean Squared Error (MSE), which measures the difference between actual and predicted values.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{1}$$

The computed MSE in this experiment is displayed in the output.

3.2 Visualizations

- **Actual vs. Predicted Scatter Plot:** Shows the correlation between actual and predicted values.
- **Residual Plot:** Highlights the errors in predictions.
- **Distribution Plot:** Compares the distributions of actual and predicted values.
- **Missing Data Matrix:** Provides an overview of missing values.

4 Conclusion

This script effectively handles missing data by predicting values using a trained Linear Regression model. The results demonstrate the effectiveness of using statistical imputation and predictive modeling to fill missing values.