Data Inputation Techniques

Saksham Nandanwar
Roll no. 22111048

July 12, 2024

1 Mean/Median Imputation

Mean or median imputation replaces missing values in a data set with the mean (average) or median (middle value) of the missing values for the corresponding value. This process is simple and easy to use, but it can affect the change of data, especially if missing values are not distributed.

1.1 Advantages

1. Ease of Implementation
Requires minimum computation and no complex algorithms.
2. Fast Execution
Suitable for lagre dataset

1.2 Disadvantages

Reduction in Variability

1.3 Case Study

Consider a dataset with missing values in the 'Blood Pressure' and 'Cholesterol' columns. We will demonstrate mean and median imputation using Python.

1.4 code

```
import pandas as pd
import numpy as np
# Sample data creation
data =
'Age': [25, 30, 22, 28, 35, 40],
'Blood Pressure': [120, 130, np.nan, 140, np.nan, 135],
'Cholesterol': [200, np.nan, 180, 190, np.nan, 210]
```

df = pd.DataFrame(data)
print("Original Data:")
print(df)

Mean Imputation
df$_{m}ean_{i}mputed = df.copy()$
$df_{m}ean_{i}mputed['BloodPressure'].fillna(df['BloodPressure'].mean(), inplace = True)$
$df_{m}ean_{i}mputed['Cholesterol'].fillna(df['Cholesterol'].mean(), inplace = True)$

$print("afterMeanImputation:")$
$print(df_{m}ean_{i}mputed)$

Median Imputation
df$_{m}edian_{i}mputed = df.copy()$
$df_{m}edian_{i}mputed['BloodPressure'].fillna(df['BloodPressure'].median(), inplace = True)$
$df_{m}edian_{i}mputed['Cholesterol'].fillna(df['Cholesterol'].median(), inplace = True)$

$print("afterMedianImputation:")$
$print(df_{m}edian_{i}mputed)$

2 Mode Imputation

Mode imputation involves replacing missing values in a dataset with the mode. This technique is generally used for categorical data but can also be used for numerical data when appropriate.

2.1 Advantages

1. Fast Execution
Suitable for large dataset
2. Ease of Implementation

2.2 Disadvantages

1. Not Suitable for All Data Types
Less affective for numerical data 2. Reduction in Variability

2.3 Case Study

Consider a dataset with missing values in the 'Gender' and 'Smoker' columns. We will demonstrate mode imputation using Python.

2.4 code

import pandas as pd
import numpy as np
Sample data creatio
data =
'Age': [25, 30, 22, 28, 35, 40],
'Gender': ['Male', 'Female', np.nan, 'Female', 'Male', np.nan],
'Smoker': ['No', 'Yes', 'No', np.nan, 'Yes', np.nan]

df = pd.DataFrame(data)
print("Original Data:")
print(df)

Mode Imputation
df$_mode_imputed = df.copy()$
$\#Functiontoimputemode$
$defimpute_mode(series):$
$mode = series.mode()[0]$
$returnseries.fillna(mode)$

Apply mode imputation
df$_mode_imputed['Gender'] = impute_mode(df['Gender'])$
$df_mode_imputed['Smoker'] = impute_mode(df['Smoker'])$
$print("afterModeImputation:")$
$print(df_mode_imputed)$

3 K-Nearest Neighbors (KNN) Imputation

K-nearest neighbor (KNN) imputation is a method of imputing missing values by looking at the "k" closest points (neighbours) to missing data already in the feature space. Missing values are then imputed to the nearest neighbor's values

3.1 Advantages

1. Flexible
Can handle different types of data
2. Can capture non-linear relationships between features

3.2 Disadvantages

1. Scalability Issues
Not suitable for large dataset

3.3 Case Study

A real estate company want to develop a model to predict house prices based on various features such as size, no. of bedrooms, no. of bathrooms,etc. The dataset contains missing values in some features, which we need to handle to build a model.

3.4 code

# Step 1: Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.impute import KNNImputer
from sklearn.preprocessing import StandardScaler

Step 2: Load the dataset
data =
'Size': [2100, 1600, np.nan, 2400, 1800],
'Bedrooms': [3, np.nan, 4, 3, 2],
'Bathrooms': [2, 2, 3, 2, 1],
'Location': [1, 2, 3, np.nan, 2],
'Age': [10, 5, 8, 20, 15],
'Price': [500000, 350000, 450000, 600000, 400000]

df = pd.DataFrame(data)

Step 3: Preprocess the data
Convert categorical variables to numerical format if necessary
In this example, 'Location' is already numerical

# Standardize the data (excluding the target variable 'Price')
scaler = StandardScaler()
scaled$_features = scaler.fit_transform(df.drop('Price', axis = 1))$

$\#Step4 : ApplyKNNImputation$
$imputer = KNNImputer(n_neighbors = 3)$
$imputed_data = imputer.fit_transform(scaled_features)$

$\#Converttheimputeddatabacktotheoriginalscale$
$imputed_data = scaler.inverse_transform(imputed_data)$

$\#CreateanewDataFramewiththeimputeddata$
$imputed_df = pd.DataFrame(imputed_data, columns = df.columns[: -1])$
$imputed_df['Price'] = df['Price']$

$print("OriginalDataFramewithMissingValues:")$
$print(df)$
$print("afterKNNImputation:")$
$print(imputed_df)$

$\#Step5: Evaluatetheresult$
$\#Forthiscasestudy, wesimplyprintthebeforeandafterdata.$
$\#Inareal-worldscenario, furtheranalysisandmodelingwouldfollow.$

4 Multiple Imputation by Chained Equations (MICE)

Multiple Imputation by Chained Equations (MICE) is a method for handling missing data by creating multiple complete datasets, each with a different imputed values. These datasets are then analyzed separately and the results are combined to create final estimate. This imputation process use chained equations which means each variable with missing values is imputed according to the model that includes other variables as predictors.

4.1 Advantages

1. Versatility
It Can be applied to a wide range of data types and structures
2. Flexibility

4.2 Disadvantages

1. Complex

4.3 Case Study

Consider a healthcare dataset with missing values in the 'Age' and 'BMI' columns. We will demonstrate MICE imputation using Python.

4.4 code

# Step 1: Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.experimental import enable$_iterative_imputer\#noqa$
$fromsklearn.imputeimportIterativeImputer$

$\#Step2 : Loadthedataset$
$data = 'Age' : [25, 45, np.nan, 35, 50], 'Gender' : [1, 0, 1, 0, 1], 'BMI' : [22.5, np.nan, 24.7, 26.8, np.nan], 'BloodP$

$df = pd.DataFrame(data)$

$\#Step3 : ApplyMICEImputation$
$imputer = IterativeImputer(max_iter = 10, random_state = 0)$
$imputed_data = imputer.fit_transform(df)$

$\#ConverttheimputeddatabacktoaDataFrame$
$imputed_df = pd.DataFrame(imputed_data, columns = df.columns)$

$\#Step4 : Evaluatetheresult$
$print("OriginalDataFramewithMissingValues :")$
$print(df)$
$print("afterMICEImputation :")$
$print(imputed_df)$

5 Regression Imputation

Regression imputation is a method of identifying missing values by using a regression mode to estimate missing values. In this method, the missing value of the variable is estimated based on the observed values of the other variables. These model uses the relationships between variables to produce imputed values that are consistent with the observed data.

5.1 Advantages

1. predictive Power
It Can improve the quality of the dataset
2. Single Imputation
Reduces complexity

5.2 Disadvantages

1. Model Dependency

5.3 Case Study

Consider a dataset with missing values in the 'BMI' column. We will demonstrate regression imputation using Python.

5.4 code

# Step 1: Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model$_{s}election import train_{t}est_{s}plit$
$from sklearn.linear_{m}odel import Linear Regression$

$\#Step2 : Load the dataset (assuming' data.csv' is your dataset file)$
$df = pd.read_{c}sv('data.csv')$

$\#Step3 : Split the dataset into training and test sets$
$\#Assume' BMI' is the column with missing values$
$train = df.dropna(subset = ['BMI'])Use rows where' BMI' is not NaN for training$
$test = df[df['BMI'].isnull()] Rows with missing' BMI' values for imputation$

$\#Separate features and target$
$X_{t}rain = train.drop(['BMI'], axis = 1)$
$y_{t}rain = train['BMI']$
$X_{t}est = test.drop(['BMI'], axis = 1)$

$\#Step4 : Train a regression model on non - missing data$
$model = Linear Regression()$
$model.fit(X_{t}rain, y_{t}rain)$

$\#Step5 : Predict missing' BMI' values using the trained model$
$predicted_{b}mi = model.predict(X_{t}est)$

$\#Step6 : Impute the missing values in the original dataset$
$df.loc[df['BMI'].isnull(),' BMI'] = predicted_{b}mi$

$\#Step7 : Evaluate the imputed dataset(optional)$
$print("Original DataFrame with Missing Values : ")$
$print(df)$