**Project Title**: **Data Science:: Bank Marketing (Campaign) -- Group Project**
**Date**: Feb 01, 2024

**Group Name**: Data Department_1
- **Name**: Minseok Kim
- **Email**: mxk230041@utdallas.edu
- **Country**: United States
- **College/Company**: The University of Texas at Dallas
- **Specialization**: Data Science

**Problem Description**
The objective of this project is to analyze the marketing campaign data of a Portuguese bank and predict the likelihood of customers subscribing to a term deposit product. This analysis enables the bank to efficiently allocate marketing resources and focus on customers with a high probability of subscription.

**GitHub Repository Link**
The work content and code were shared and collaborated through GitHub. The link to the GitHub repository is as follows: https://github.com/N0VA-code/Week9.git

**Data Cleansing and Transformation**
Handling Missing Values
For handling missing values within the dataset, average and median values were employed. For numeric variables, missing values were replaced with the mean, and for categorical variables, the mode was used as a substitute.
Outlier Detection and Treatment
Two approaches were adopted for handling outliers within the dataset:
- **Z-score**: This method calculates the Z-score of each data point using the mean and standard deviation of the data. Points with a Z-score less than -3 or greater than 3 were considered outliers and removed. The Z-score method showed high efficiency for columns following a normal distribution.

```
import pandas as pd
import matplotlib
import numpy as np

from matplotlib import pyplot as plt
from scipy.stats import norm
%matplotlib inline
matplotlib.rcParams['figure.figsize'] = (10, 6)
df = pd.read_csv("bank.csv", sep = ';')
numeric_columns = df.select_dtypes(include=['number'])
print(numeric_columns)

{'age', 'balance', 'day', 'duration', 'campaign', 'pdays', 'previous'}
def show_hist(df, column):
```

```
    plt.hist(df[column], bins=50, rwidth=0.9)
    plt.xlabel(column)  # Corrected typo here
    plt.ylabel('count')    # Corrected typo here
    plt.show()
show_hist('age')
df['zscore'] = (df.age - df.age.mean()) / df.age.std()
df_no_outliers_age1 = df[(df.zscore> -3) & (df.zscore < 3)]
def show_no_outliers (df, column):

    plt.hist(df[column], bins=50, rwidth=0.9)
    plt.xlabel(column)  # Corrected typo here
    plt.ylabel('count')    # Corrected typo here
    plt.show()
show_no_outliers (df_no_outliers_age1, 'age')
df = df_no_outliers

df['zscore'] = (df.duration - df.duration.mean()) / df.duration.std()
df_no_outliers_age1_balance2_duration3 = df[(df.zscore> -3) & (df.zscore < 3)]

show_no_outliers (df_no_outliers_age1_balance2, 'balance')
def no_outliers(df, column):
    df_copy = df.copy()  # Create a copy to avoid modifying the original DataFrame

    zscore = (df_copy[column] - df_copy[column].mean()) / df_copy[column].std()

    df_copy.loc[:, 'zscore'] = zscore

    df_no_outliers = df_copy[(zscore > -3) & (zscore < 3)]

    return df_no_outliers


# Remove outliers from the 'duration' column using the function
df_no_outliers = no_outliers(df, 'duration')

# Display the DataFrame without outliers
print(df_no_outliers)

df_no_outliers = no_outliers(df, 'campaign')
df = df_no_outliers
df_no_outliers = no_outliers(df, 'pdays')
df = df_no_outliers
df_no_outliers = no_outliers(df, 'previous')
df = df_no_outliers
df.shape
show_hist(df, 'age')
show_hist(df, 'balance')
show_hist(df, 'day')
```

```
show_hist(df, 'duration')
show_hist(df, 'campaign')
show_hist(df, 'pdays')
show_hist(df, 'previous')
df.shape
df.to_csv("Bank_cleaned.csv")
```

- **IQR (Interquartile Range)**: This method calculates the IQR as the range between the first quartile (Q1) and the third quartile (Q3) of the data. Data points falling below Q1-1.5*IQR or above Q3+1.5*IQR were deemed outliers and removed.

It is worth noting that when removing outliers using the Z-score method, any row containing at least one outlier in any column was entirely removed. This approach ensured a thorough cleansing of the dataset but also highlighted the method's potential for significant data reduction without regard to the column's distribution.

```
import pandas as pd
import matplotlib
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
matplotlib.rcParams['figure.figsize'] = (10, 6)

df = pd.read_csv("bank.csv", sep=';')
numeric_columns = df.select_dtypes(include=['number'])
print(numeric_columns)
def remove_outliers_IQR(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    df_filtered = df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]
    return df_filtered
for column in numeric_columns:
    df = remove_outliers_IQR(df, column)

def show_hist(df, column):
    plt.hist(df[column], bins=50, rwidth=0.9)
    plt.xlabel(column)
    plt.ylabel('count')
    plt.show()

for column in numeric_columns:
    show_hist(df, column)

print(df.shape)
```

```
df.to_csv("Bank_cleaned_IQR.csv")
```

**Natural Language Processing (NLP) Data Cleansing**

As our team is solely focused on the Data Science track and lacks an NLP specialization, we did not undertake natural language processing data cleansing tasks.

**Collaboration and Code Review**

Given that I am the sole member of the team, the data cleansing approach was independently conducted. Following the implementation of the Z-score and IQR methods for outlier treatment, I reviewed the advantages and challenges of each method:

- The **Z-score** method is effective for columns that follow a normal distribution and is straightforward to implement. However, it may not adequately identify outliers in significantly skewed data. Additionally, by removing rows containing any outliers across columns, this method could potentially lead to a substantial reduction in data, irrespective of the specific column's distribution characteristics.
- The **IQR** method offers a more flexible approach to identifying outliers, regardless of the data's distribution. However, this method could risk identifying too many data points as outliers, especially in unevenly distributed datasets.

Based on the results of applying each method, the most appropriate outlier treatment technique was chosen for the project, considering the characteristics of the data at hand.