Future Sales Prediction - Problem Definition and Design Thinking

Problem Definition

The primary objective of this project is to develop a predictive model that leverages historical sales data to forecast future sales for a retail company. The ultimate aim is to create a powerful tool that empowers the company to optimize inventory management and make informed business decisions grounded in data-driven sales predictions. This project comprises several key phases, including data preprocessing, feature engineering, model selection, model training, and model evaluation.

Design Thinking

Data Source

The dataset provided contains the following attributes: TV, Radio, Newspaper, and Sales. These features represent the marketing expenses in different advertising channels (TV, Radio, Newspaper) and the corresponding sales figures. To implement our predictive model, we will focus on the "Sales" attribute as our target variable, and the "TV," "Radio," and "Newspaper" attributes as potential predictors.

Data Preprocessing

Data Loading: We start by loading the dataset into a pandas DataFrame.

```
import pandas as pd
data = pd.read_csv("Sales.csv")
```

Data Exploration: To understand the data better, we examine the first few rows, data types, and check for missing values and duplicated entries.

```
print(data.head())
```

Here, we use data.head() to display the first few rows of the dataset. This helps us get an initial view of the data, including the columns and their values.

```
print(data.info())
```

This line of code provides information about the dataset, including the data types of each column and the number of non-null entries. It's essential to check for missing values and understand the data types before proceeding with analysis

```
print("The sum of duplicated items:", data.duplicated().sum())
```

This code checks for and prints the sum of duplicated rows in the dataset. Duplicated rows may need to be removed to avoid biasing the analysis.

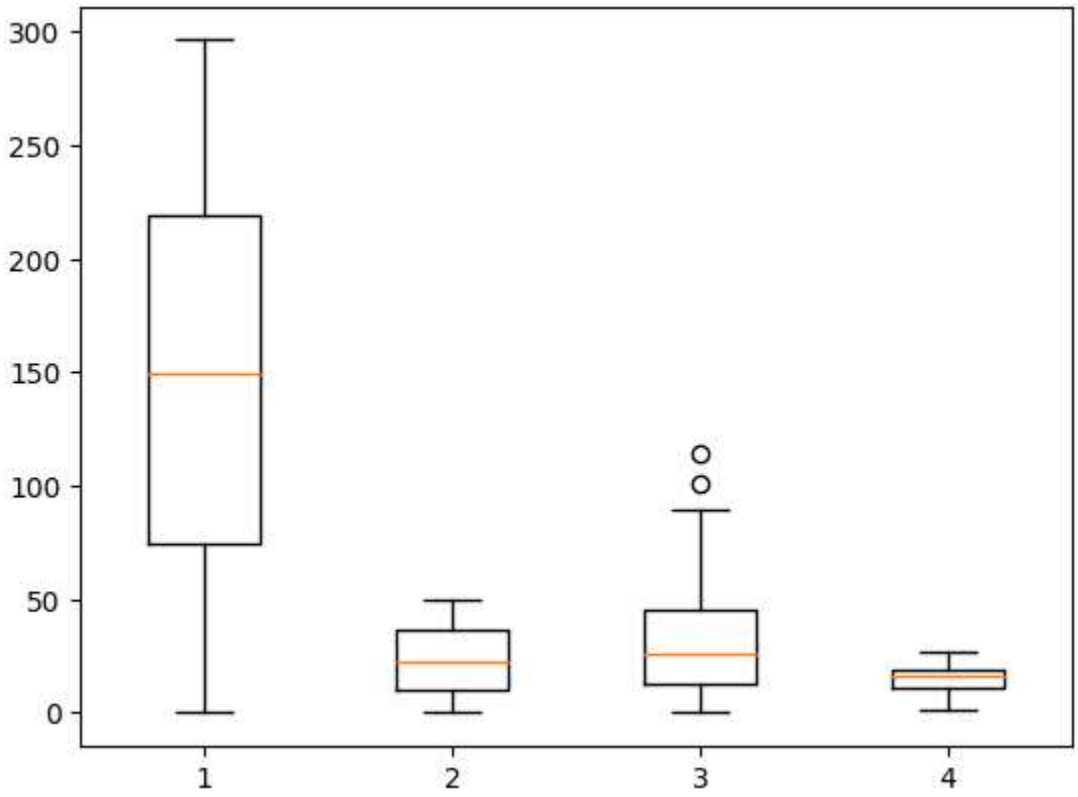
print("To check the null values:", data.isna().sum())

Here, we use data.isna().sum() to check for null (missing) values in the dataset. Handling missing data is a crucial step in data preprocessing.

Handling Outliers: We visualize potential outliers using box plots to identify any extreme values.

import matplotlib.pyplot as plt

print("Boxplot for handling outliers", plt.boxplot(data))



Descriptive Statistics

In this section, we calculate both the mean and median for each of the attributes in the dataset: TV, Radio, Newspaper, and Sales. The mean represents the average value, while the median is the middle value when the data is sorted. These statistics provide insights into the central tendency of each attribute.

```
# Calculate the mean and median for each attribute: TV, Radio, Newspaper, and Sales
print("Mean of TV:", data["TV"].mean())
print("Median of TV:", data["TV"].median())
print("Mean of Radio:", data["Radio"].mean())
print("Median of Radio:", data["Radio"].median())
print("Mean of Newspaper:", data["Newspaper"].mean())
print("Median of Newspaper:", data["Newspaper"].median())
print("Mean of Sales:", data["Sales"].mean())
print("Median of Sales:", data["Sales"].median())
```

The code calculates and prints the mode (the most frequent value) for each attribute. Knowing the mode can be useful for understanding the most common values in the dataset.

```
# Calculate the mode for each attribute
```

```
print("Mode of TV:", data["TV"].mode())
print("Mode of Radio:", data["Radio"].mode())
print("Mode of Newspaper:", data["Newspaper"].mode())
print("Mode of Sales:", data["Sales"].mode())
```

Frequency tables are generated for each attribute to understand the distribution of values. These tables show how often each value occurs in the dataset.

```
# Create frequency tables for each attribute to understand the distribution of values
print("Frequency Table of TV:", data["TV"].value_counts())
print("Frequency Table of Radio:", data["Radio"].value_counts())
print("Frequency Table of Newspaper:", data["Newspaper"].value_counts())
print("Frequency Table of Sales:", data["Sales"].value_counts())
```

Feature Engineering

Feature Creation: We will create additional features that could enhance the predictive power of the model. In particular, we'll consider time-based features like day of the week and month, which might be valuable for capturing sales seasonality.

Model Selection

For this project, we intend to predict future sales based on advertising spending, which can be considered a regression problem. We'll start with a linear regression model, but we could also explore more sophisticated models if necessary.

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

X = data[['TV', 'Radio', 'Newspaper']]
y = data['Sales']

# Splitting the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Creating a Linear Regression model
model = LinearRegression()

# Training the model
model.fit(X_train, y_train)

# Making predictions
```

```
y_pred = model.predict(X_test)
```

Evaluation

We evaluate the model's performance using common regression metrics:

```
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Squared Error (MSE): {mse}")
print(f"R-squared (R2): {r2}")
```

We import metrics from scikit-learn to evaluate the model. We calculate three common regression evaluation metrics:

Mean Absolute Error (MAE): This metric measures the average absolute difference between predicted and actual values. Lower MAE values indicate better model performance.

Mean Squared Error (MSE): MSE measures the average squared difference between predicted and actual values. Lower MSE values indicate better accuracy.

R-squared (R2): R2 measures the proportion of the variance in the target variable that is predictable from the features. It ranges from 0 to 1, with higher values indicating better model fit.
The calculated metrics help assess how well the linear regression model performs in making sales predictions based on the features (TV, Radio, and Newspaper advertising expenditures).

This initial model using linear regression allows us to make predictions based on advertising spending. Further iterations may involve refining features, exploring different algorithms, and considering time series forecasting techniques as outlined in the original problem statement for more accurate future sales predictions.

This Provides an overview of how we've approached the problem statement in terms of data preprocessing, feature engineering, model selection, training, and evaluation. Further steps will be based on the results of this initial model and the specific requirements and goals of the retail company.