Linear Regression

Linear regression is a statistical method used to model the relationship between a dependent variable (often called the response or target variable) and one or more independent variables (often called predictors or features). The goal is to find the best-fitting straight line through the data points that predicts the dependent variable as a linear function of the independent variable(s).

In its simplest form, with one independent variable, linear regression fits a line to the data points in the form:

$$y=mx+b$$

where:

- y is the dependent variable.
- x is the independent variable.
- m is the slope of the line, indicating how much y changes for a unit change in x.
- b is the y-intercept, indicating the value of y when $x=0$.

Steps in Linear Regression

1. **Collect Data**: Gather the data points of x and y.
2. **Fit the Line**: Determine the best-fitting line using the least squares method, which minimizes the sum of the squares of the vertical distances of the points from the line.
3. **Make Predictions**: Use the line to predict y values for given x values.

Example

Let's consider a simple example with a small dataset:

Suppose we have data on the number of hours studied (x) and the score obtained in an exam (y):

Hours Studied (xx)	**Exam Score (yy)**
1	50
2	55
3	65
4	70
5	75

We want to fit a line through these points.

1. **Plot the Data**: First, plot the data points on a graph.
2. **Calculate the Line**:
    - We find the slope (m) and y-intercept (b) using the least squares method.
    - For simplicity, let's assume the calculations give us $m=5$ and $b=45$.
3. **Equation of the Line**:
    - The best-fitting line is $y=5x+45$.
4. **Make Predictions**:
    - If a student studies for 3 hours, the predicted exam score is $y=5(3)+45=60$.

Visual Representation

Imagine a scatter plot with "Hours Studied" on the x-axis and "Exam Score" on the y-axis. The line $y=5x+45$ should run through the data points, minimizing the distance between the points and the line.

Summary

Linear regression finds the line that best represents the data. In this example, as the hours studied increase, the exam score increases linearly. The slope indicates that for each additional hour studied, the score increases by 5 points. The intercept tells us that without studying at all, the predicted score is 45.

Steps in Data Cleaning

- Identifying Anomalies: Detecting errors or inconsistencies in the data.
- Dealing with Missing Data: Handling null or missing values.
- Data Transformation: Standardizing and normalizing data.
- Removing Duplicates: Identifying and deleting repeated data points.
- Error Correction: Fixing or removing data anomalies and errors.

If we assume that our coffee app has been collecting a variety of data, such as transaction records, customer feedback, data from Internet of Things machines, and so on, so be it. The data needs to be cleaned up at this point.

Handling Missing Values

Some customer feedback entries are missing ratings or comments.

Solution: Replace missing values with a substitute, like the mean or median of that column. If the missing data is significant, consider removing those entries.

Example:

```
# Assuming 'df' is our DataFrame

# Impute missing ratings with the median

df['rating'].fillna(df['rating'].median(), inplace=True)

# Drop rows where feedback is missing

df.dropna(subset=['feedback'], inplace=True)
```

Correcting Inconsistencies

The format of timestamps or product names is inconsistent.

Solution: Convert all entries into a consistent format.

Example:

```
# Standardizing timestamp format
```

```
df['timestamp'] = pd.to_datetime(df['timestamp'])

# Standardizing product names to lowercase

df['product'] = df['product'].str.lower()
```

Removing Duplicates

The dataset has duplicate entries due to a glitch in data collection.

Solution: Identifying and removing duplicate records.

Example:

```
# Dropping duplicate rows

df.drop_duplicates(inplace=True)
```

Dealing with Outliers

Outlier values in transaction amounts or quantities that don't make sense (e.g., extremely high values).

Solution: Use statistical methods to identify and handle outliers.

Example:

```
# Identifying outliers using IQR

Q1 = df['quantity'].quantile(0.25)

Q3 = df['quantity'].quantile(0.75)

IQR = Q3 - Q1

df = df[~((df['quantity'] < (Q1 - 1.5 * IQR)) |(df['quantity'] > (Q3 + 1.5 * IQR)))]
```

Error Correction

Data entry errors in customer or machine IDs.

Solution: Cross-reference IDs with a master list and correct or remove invalid entries.

Example:

```
# Validating customer IDs

valid_ids = get_valid_customer_ids() # Assume this function gets valid IDs

df['customer_id'] = df['customer_id'].apply(lambda x: x if x in valid_ids else np.nan)

df.dropna(subset=['customer_id'], inplace=True)
```

Data Transformation

Data is in different scales or formats that are not suitable for analysis.

Solution: Normalize data scales and encode categorical variables.

Example:

```
# Normalizing a column

df['quantity'] = (df['quantity'] - df['quantity'].mean()) / df['quantity'].std()

# Encoding categorical variables

df = pd.get_dummies(df, columns=['product'])
```

Feature Engineering

Feature engineering plays a critical role in enhancing the performance of machine learning models. It involves the creative process of constructing new features from the existing dataset to improve the model's ability to recognize patterns and make accurate predictions. This step can involve various techniques, such as aggregating data into new combinations, extracting parts of a date-time stamp (like the day of the week), or creating interaction features that combine two or more variables. The goal is to provide the model with additional, meaningful input that can lead to more nuanced and informed decision-making.

Create a new feature that could be useful for prediction, like total sales.

```
df['Total_Sales'] = df['Espresso'] + df['Latte'] + df['Cappuccino']
```