I will generate a dataset with approximately 100 variables for use in Tasks 1 and 2. This dataset will serve as the basis for the linear and logistic regression models in Tasks 1 and 2.

In [8]:

```python
import pandas as pd
import numpy as np

# Seed for reproducibility
np.random.seed(42)

# Generating dataset
n_samples = 1000
n_features = 99
X = np.random.rand(n_samples, n_features) * 100  # Feature matrix
y_linear = X.dot(np.random.rand(n_features)) + np.random.rand(n_samples) * 50  # Continuous target variable
y_logistic = (np.random.rand(n_samples) > 0.5).astype(int)  # Binary target variable

# Creating a DataFrame
df_linear = pd.DataFrame(X, columns=[f'Feature_{i}' for i in range(1, n_features + 1)])
df_linear['Target'] = y_linear

df_logistic = pd.DataFrame(X, columns=[f'Feature_{i}' for i in range(1, n_features + 1)])
df_logistic['Target'] = y_logistic

# Saving datasets to CSV
linear_csv_path = "linear_dataset.csv"
logistic_csv_path = "logistic_dataset.csv"

df_linear.to_csv(linear_csv_path, index=False)
df_logistic.to_csv(logistic_csv_path, index=False)

linear_csv_path, logistic_csv_path
```

Out[8]:

```
('linear_dataset.csv', 'logistic_dataset.csv')
```

Task 1: Linear Regression Model

Linear regression is a statistical method that models the relationship between a dependent variable and one or more independent variables by fitting a linear equation to observed data. The basic form of the equation of a straight line is y = mx + c, where:

- y is the dependent variable,
- x is the independent variable,
- m is the slope of the line,
- c is the y-intercept.

In the context of multiple linear regression, where there are multiple independent variables, the equation is generalized to y = beta0 + beta1*x1 + beta2*x2 + ... + betan*xn + epsilon, with beta values representing the coefficients, and epsilon the error term.

Practical Example with Data:

For the practical example, we will use the generated dataset with 99 features as independent variables and a continuous target variable to demonstrate how to build a linear regression model using Python's scikit-learn library.

In [10]:

```python
import pandas as pd

# Load the dataset
df = pd.read_csv('linear_dataset.csv')
X = df.drop('Target', axis=1)
y = df['Target']
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and fit the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Predictions
predictions = model.predict(X_test)
from sklearn.metrics import mean_squared_error, r2_score

mse = mean_squared_error(y_test, predictions)
r2 = r2_score(y_test, predictions)

print(f'MSE: {mse}')
print(f'R^2: {r2}')
```

```
MSE: 232.67699313065947
R^2: 0.9929239263657009
```

Task 2: Logistic Regression Model Explanation and Example

Logistic regression is a statistical model that, in its basic form, uses a logistic function to model a binary dependent variable. Unlike linear regression, which predicts a continuous outcome, logistic regression predicts the probability of an outcome occurring, such as yes/no, true/false, success/failure.

The logistic function, also known as the sigmoid function, ensures that the output value always falls between 0 and 1. This characteristic makes logistic regression particularly well-suited for models where the outcome is binary.

The equation for logistic regression is given by the logit function: logit(p) = ln(p / (1 - p)) = beta0 + beta1*x1 + beta2*x2 + ... + betan*xn, where:

- p is the probability of the presence of the characteristic of interest,
- beta0, beta1, ..., betan are the regression coefficients.

In [11]:

```python
# Load the logistic regression dataset
df_logistic = pd.read_csv('logistic_dataset.csv')
X_logistic = df_logistic.drop('Target', axis=1)
y_logistic = df_logistic['Target']
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

# Splitting the dataset into training and testing sets
X_train_logistic, X_test_logistic, y_train_logistic, y_test_logistic = train_test_split(X_logistic, y_logistic, test_size=0.2, random_state=42)

# Initializing and training the logistic regression model
logistic_model = LogisticRegression(max_iter=1000)
logistic_model.fit(X_train_logistic, y_train_logistic)

# Making predictions
logistic_predictions = logistic_model.predict(X_test_logistic)
from sklearn.metrics import accuracy_score, confusion_matrix

accuracy = accuracy_score(y_test_logistic, logistic_predictions)
conf_matrix = confusion_matrix(y_test_logistic, logistic_predictions)

print(f'Accuracy: {accuracy}')
print(f'Confusion Matrix: \n{conf_matrix}')
```

```
Accuracy: 0.525
Confusion Matrix:
[[53 45]
 [50 52]]
```

In [ ]: