# Linear Regression

Sami Uddin Shinwari

July 2024

## 1 Basics of Linear Regression

### 1.1 Objective

Linear regression aims to predict the value of the dependent variable ($Y$) based on the values of the independent variables ($X$).

### 1.2 Types

- **Simple Linear Regression:** Involves one independent variable.
- **Multiple Linear Regression:** Involves two or more independent variables.

## 2 The Linear Regression Equation

The general form of the linear regression equation is:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n + \epsilon$$

Where:

- $Y$ is the dependent variable.
- $\beta_0$ is the intercept.
- $\beta_1, \beta_2, \ldots, \beta_n$ are the coefficients (slopes) for the independent variables.
- $X_1, X_2, \ldots, X_n$ are the independent variables.
- $\epsilon$ is the error term (residual).

## 3 Diagram of Simple Linear Regression

In the figure 1:

- The blue dots represent the observed data points.
- The red line is the best-fitting line, which minimizes the sum of the squared differences between the observed and predicted values.
- The vertical lines (residuals) represent the difference between the actual data points and the predicted values on the regression line.

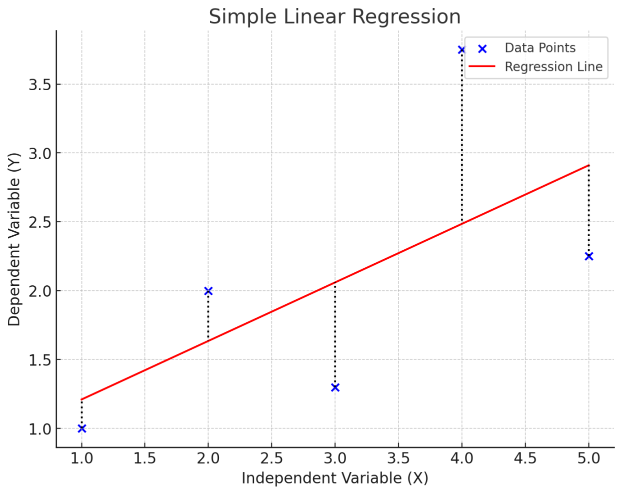

Figure 1: Simple Linear Regression

## 4 Assumptions of Linear Regression

1. **Linearity:** The relationship between the dependent and independent variables is linear.
2. **Independence:** Observations are independent of each other.
3. **Homoscedasticity:** The residuals (errors) have constant variance at every level of the independent variables.
4. **Normality:** The residuals are normally distributed.
5. **No multicollinearity:** Independent variables are not highly correlated with each other.

## 5 Estimating the Coefficients

### 5.1 Ordinary Least Squares (OLS)

The most common method for estimating the coefficients in a linear regression model is Ordinary Least Squares (OLS). This method minimizes the sum of the squared residuals:

$$\min \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2$$

Where:

- $Y_i$ is the actual value.
- $\hat{Y}_i$ is the predicted value.

## 6 Evaluating the Model

### 6.1 R-squared ($R^2$)

$R^2$ measures the proportion of variance in the dependent variable that is predictable from the independent variables. It ranges from 0 to 1.

$$R^2 = 1 - \frac{\sum (Y_i - \hat{Y}_i)^2}{\sum (Y_i - \bar{Y})^2}$$

Where:

- $\bar{Y}$ is the mean of the actual values.

### 6.2 Adjusted R-squared

Adjusted $R^2$ adjusts the $R^2$ value based on the number of predictors in the model.

### 6.3 Mean Squared Error (MSE)

MSE is the average of the squared differences between the actual and predicted values.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

### 6.4 Root Mean Squared Error (RMSE)

RMSE is the square root of MSE and provides a measure of the average error magnitude.

## 7 Interpretation of Coefficients

- **Intercept ($\beta_0$):** Represents the expected value of $Y$ when all $X$ variables are zero.
- **Slope ($\beta_1, \beta_2, \ldots, \beta_n$):** Represents the change in $Y$ for a one-unit change in $X$, holding all other variables constant.

## 8 Significance Testing

### 8.1 t-test

Used to determine if the coefficients are significantly different from zero.

$$t = \frac{\beta_i}{SE(\beta_i)}$$

Where $SE(\beta_i)$ is the standard error of the coefficient.

### 8.2 p-value

The probability that the observed results would occur by chance. A p-value less than 0.05 typically indicates statistical significance.

## 9 Residual Analysis

Analyzing residuals helps validate the assumptions of linear regression:

- **Residual Plot:** Should show no pattern.
- **QQ Plot:** Should show residuals are normally distributed.
- **Durbin-Watson Test:** Checks for autocorrelation in residuals.

## 10 Applications

Linear regression is used in various fields such as:

- **Economics:** Predicting economic indicators.
- **Finance:** Modeling stock prices.
- **Marketing:** Understanding the impact of advertising on sales.
- **Healthcare:** Predicting patient outcomes based on clinical measurements.

## 11 Limitations

- **Linearity Assumption:** Not all relationships are linear.
- **Outliers:** Can have a large influence on the model.
- **Multicollinearity:** High correlation between independent variables can distort results.
- **Extrapolation:** Predictions outside the range of the data can be unreliable.

## 12 Conclusion

Linear regression is a powerful tool for understanding relationships between variables and making predictions. Its simplicity and interpretability make it a popular choice for many applications, but it's important to validate its assumptions and be aware of its limitations.

## 13 DataFrame

| **Avg. Session Length** | **Time on App** | **Time on Website** | **Length of Membership** | **Yearly Amount Spent** |
|---|---|---|---|---|
| 34.497268 | 12.655651 | 39.577668 | 4.082621 | 587.951054 |
| 31.926272 | 11.109461 | 37.268959 | 2.664034 | 392.204933 |
| 33.000915 | 11.330278 | 37.110597 | 4.104543 | 487.547505 |
| 34.305557 | 13.717514 | 36.721283 | 3.120179 | 581.852344 |
| 33.330673 | 12.795189 | 37.536653 | 4.446308 | 599.406092 |

Table 1: Ecommerce Customers Data (Truncated)

## 14 Discussion of Features

The given DataFrame contains data from an ecommerce platform, including several features related to customer behavior and spending. For the purpose of performing linear regression to predict customer spending, we identify independent and dependent features as follows:

### 14.1 Independent Features

Independent features (also known as predictor variables) are those that we believe influence the dependent feature. In this case, the independent features include:

- **Avg. Session Length**: The average duration of a customer's session on the website.
- **Time on App**: The amount of time a customer spends using the company's mobile app.

- **Time on Website**: The amount of time a customer spends on the company's website.
- **Length of Membership**: The duration for which the customer has been a member of the ecommerce platform.

These features are selected as independent variables because they represent different aspects of customer interaction with the ecommerce platform, which are likely to impact their yearly spending.

### 14.2 Dependent Feature

The dependent feature (also known as the response variable) is the one we are trying to predict. In this case, the dependent feature is:

- **Yearly Amount Spent**: The total amount of money a customer spends in a year.

This feature is chosen as the dependent variable because the primary goal is understanding and predicting customer spending behavior based on their interactions with the e-commerce platform.

### 14.3 Rationale for Feature Selection

The selection of these independent features is based on the assumption that customer engagement metrics (session length, time on app, time on website) and customer loyalty (length of membership) are significant predictors of how much a customer spends annually. By analyzing the relationship between these features and the yearly amount spent, we can develop a model to forecast future spending and identify key factors that drive customer purchases.

This document explains the process of reading a CSV file containing e-commerce customer data, training a linear regression model, and evaluating its performance using Python.

## 15 Python Code

Listing 1: Linear Regression on Ecommerce Customers Data

```
import pandas as pd
import statsmodels.api as sm
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# Read the CSV file into a DataFrame
df = pd.read_csv('Ecommerce_Customers.csv')

# Print the DataFrame
print(df)

# Select the relevant columns
X = df[['Avg. Session Length', 'Time on App', 'Time on Website', 'Length
    of Membership']]
y = df['Yearly Amount Spent']

# Split the data 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)
```

```
# Train the model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")

# Display the coefficients
print("Coefficients:")
print(model.coef_)
print("Intercept:")
print(model.intercept_)
```

## 16 Explanation

- **Import Libraries:** Import necessary libraries including `pandas` for data manipulation, `statsmodels` for statistical models, `sklearn.metrics` for evaluation metrics, `sklearn.model_selection` for splitting data, and `sklearn.linear_model` for linear regression.
- **Read CSV File:** Read the CSV file into a pandas DataFrame.
- **Select Columns:** Select the relevant columns for the features (`X`) and the target variable (`y`).
- **Split Data:** Split the data into training and testing sets using `train_test_split`.
- **Train Model:** Train the linear regression model using the training data.
- **Make Predictions:** Make predictions on the testing data.
- **Evaluate Model:** Evaluate the model using Mean Squared Error (MSE) and R-squared ($R^2$).
- **Display Coefficients:** Print the coefficients and intercept of the model.

## 17 Deep Learning for Regression with Python

Listing 2: Deep Learning for Regression with Python

```
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score

# Read the CSV file into a DataFrame
```

```
df = pd.read_csv('Ecommerce_Customers.csv')

# Select the relevant columns
X = df[['Avg. Session Length', 'Time on App', 'Time on Website', 'Length of
    Membership']]
y = df['Yearly Amount Spent']

# Split the data 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)

# Standardize the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Build the neural network model
model = Sequential()
model.add(Dense(64, input_dim=X_train_scaled.shape[1], activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(1))  # Output layer

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
history = model.fit(X_train_scaled, y_train, epochs=100, batch_size=32,
    validation_split=0.2)

# Make predictions
y_pred = model.predict(X_test_scaled)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")

# Optional: Convert y_test and y_pred to 1D arrays for better readability
y_test_1d = y_test.values.flatten()
y_pred_1d = y_pred.flatten()

# Display first 10 actual vs predicted values
print("First 10 Actual vs Predicted values:")
for actual, predicted in zip(y_test_1d[:10], y_pred_1d[:10]):
    print(f"Actual: {actual:.2f}, Predicted: {predicted:.2f}")

# Plotting the training and validation loss
import matplotlib.pyplot as plt

plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```