Linear Regression - Complete Lecture (Theory + Practical)

📘 Introduction to Linear Regression

Linear Regression is a statistical technique in Machine Learning used for predictive modeling. It models the relationship between a dependent variable (target) and one or more independent variables (features) using a straight line.

The equation for simple linear regression is: $y = \beta_0 + \beta_1 x + \varepsilon$

- **y** is the dependent variable
- **x** is the independent variable
- β_0 is the intercept
- β_1 is the slope/coefficient
- ε is the error term

In practice, we use training data to learn the best values of β_0 and β_1 that minimize the error between actual and predicted y.

```
# Step 1: Import Required Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

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

```
# Step 2: Load and Explore Dataset
data = load_diabetes()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = data.target
df.head()
```

	age	sex	bmi	bp	s1	s2	s3	s4	
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592	0.0
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493	-0.0
2	0.085299	0.050680	0.044451	-0.005670	-0.045599	-0.034194	-0.032356	-0.002592	0.0
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309	0.0
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592	-0.0

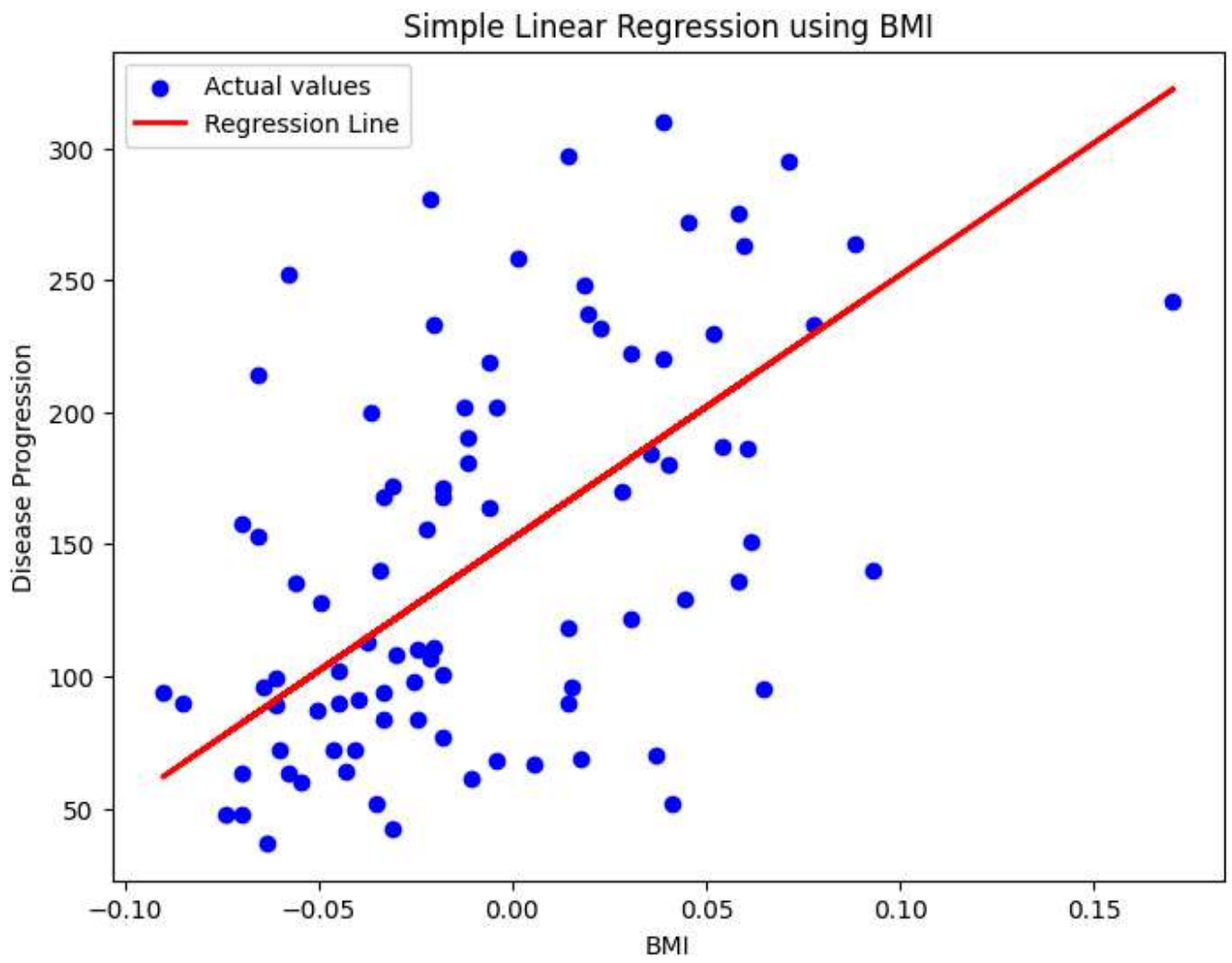
Next steps: Generate code with df | View recommended plots | New interactive sheet

```
# Step 3: Simple Linear Regression (Single Feature)
X = df[['bmi']]
y = df['target']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
model.intercept_, model.coef_
```

```
(np.float64(152.00335421448167), array([998.57768914]))
```

```
# Step 4: Visualize Regression Line
plt.figure(figsize=(8, 6))
plt.scatter(X_test, y_test, color='blue', label='Actual values')
plt.plot(X_test, y_pred, color='red', linewidth=2, label='Regression Line')
plt.xlabel('BMI')
plt.ylabel('Disease Progression')
plt.title('Simple Linear Regression using BMI')
plt.legend()
plt.show()
```




```
# Step 5: Evaluate the Model
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)
mse, rmse, r2
```

```
(4061.8259284949268, np.float64(63.73245584860925), 0.23335039815872138)
```

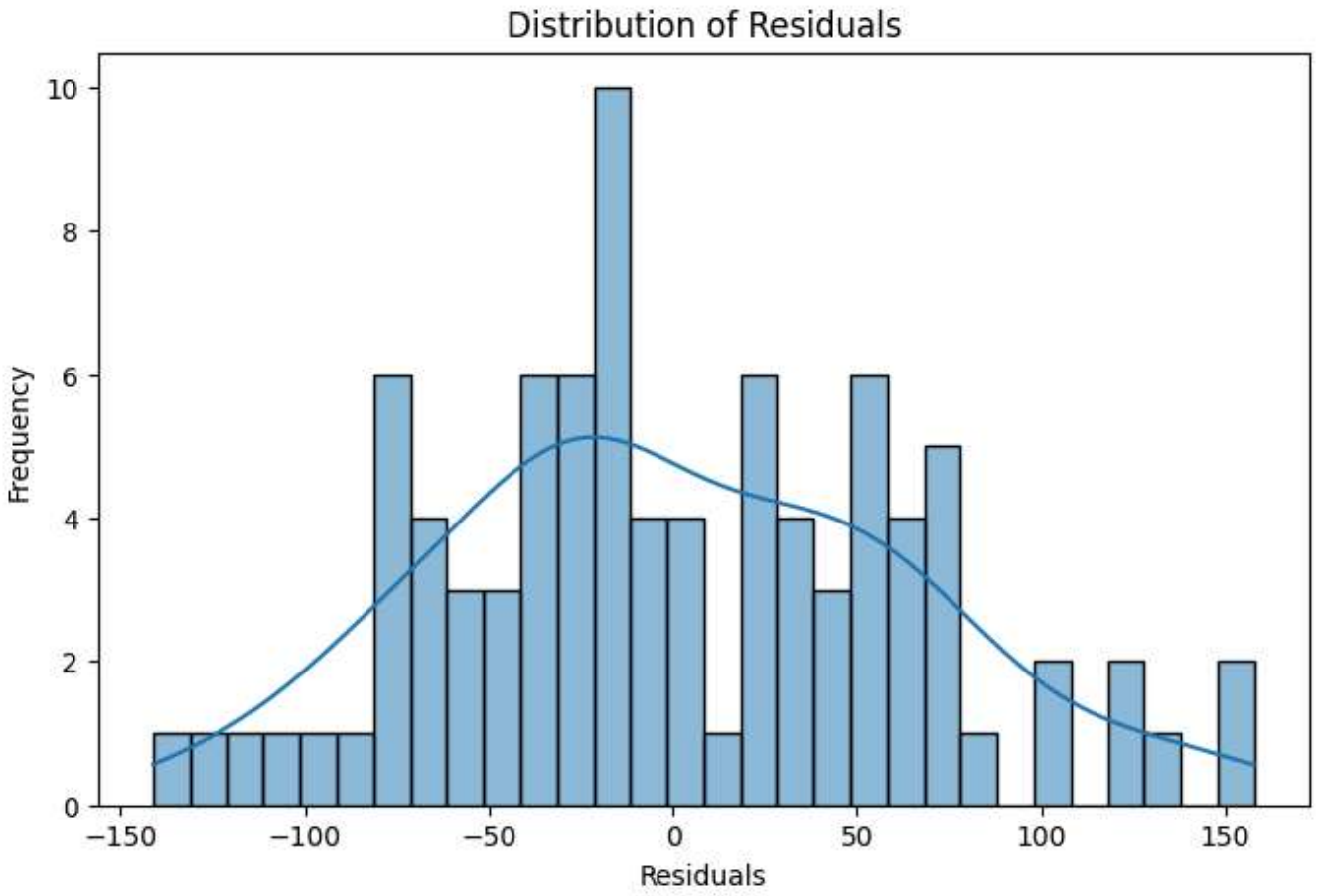
```
# Step 6: Multiple Linear Regression (All Features)
X_multi = df.drop('target', axis=1)
y_multi = df['target']
X_train_m, X_test_m, y_train_m, y_test_m = train_test_split(X_multi, y_multi, test_size=0.2,
model_multi = LinearRegression()
model_multi.fit(X_train_m, y_train_m)
y_pred_m = model_multi.predict(X_test_m)
mean_squared_error(y_test_m, y_pred_m), np.sqrt(mean_squared_error(y_test_m, y_pred_m)), r2_
```

```
(2900.193628493482, np.float64(53.85344583676593), 0.4526027629719195)
```

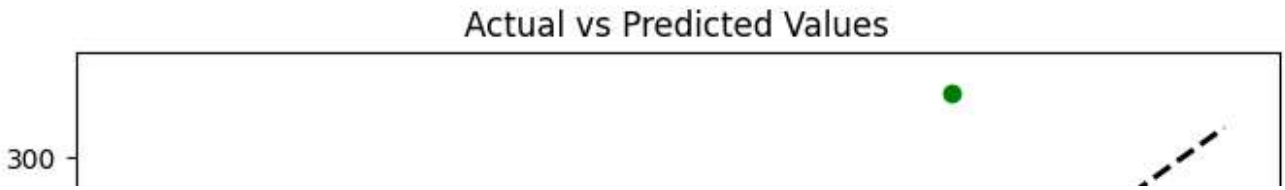
```
# Step 7: Coefficients and Feature Importance
coeff_df = pd.DataFrame(model_multi.coef_, X_multi.columns, columns=['Coefficient'])
coeff_df.sort_values(by='Coefficient', ascending=False)
```

	Coefficient
s5	736.198859
bmi	542.428759
s2	518.062277
bp	347.703844
s4	275.317902
s3	163.419983
s6	48.670657
age	37.904021
sex	-241.964362
s1	-931.488846

```
# Step 8: Residual Analysis
residuals = y_test - y_pred
plt.figure(figsize=(8, 5))
sns.histplot(residuals, kde=True, bins=30)
plt.title('Distribution of Residuals')
plt.xlabel('Residuals')
plt.ylabel('Frequency')
plt.show()
```




```
# Step 9: Actual vs Predicted Values Plot
plt.figure(figsize=(8,6))
plt.scatter(y_test, y_pred, color='green')
plt.xlabel('Actual Target Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs Predicted Values')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=2)
plt.show()
```


Actual vs Predicted Values
300