Day42_Multiple_Linear_Regression

July 11, 2025

Multiple Linear Regression (MLR)

This notebook contains **step-by-step code, explanation, and interpretation** for building and evaluating a **Multiple Linear Regression (MLR)** model using Python.

It is designed for **teaching and learning purposes**, and also simulates a real-world business scenario where a company wants to identify **which department or factor most strongly impacts promotions** — to decide where to invest their time and resources.

Learning Objectives

- Understand how **Multiple Linear Regression (MLR)** works behind the scenes
- Learn to **preprocess data**: encoding, cleaning, reshaping
- Apply and interpret **linear regression model**
- Use **OLS (Ordinary Least Squares)** to get detailed statistics
- Perform **Backward Elimination** using p-values
- Understand concepts like:
  - **Bias and Variance**
  - **Intercepts and Coefficients**
  - **Adjusted R^2 vs R^2**
  - **T-Test & p-values**
  - **Feature Elimination**
  - Basic idea of **API (Application Programming Interface)**

Tools & Libraries Used

- **Python**
- **NumPy** – numerical operations
- **Pandas** – data manipulation
- **Matplotlib** – visualization
- **scikit-learn** – machine learning
- **statsmodels** – statistical modeling

Final Goal:

To help a company answer:

> "**Which department or variable impacts promotion the most, so we can confidently invest in it?**"

1 Import Libraries

[1]:
```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

2 Load the Dataset

[2]:
```python
df = pd.read_csv(r'C:\Users\Lenovo\Downloads\Investment.csv')
df.head()
```

[2]:

	DigitalMarketing	Promotion	Research	State	Profit
0	165349.20	136897.80	471784.10	Hyderabad	192261.83
1	162597.70	151377.59	443898.53	Bangalore	191792.06
2	153441.51	101145.55	407934.54	Chennai	191050.39
3	144372.41	118671.85	383199.62	Hyderabad	182901.99
4	142107.34	91391.77	366168.42	Chennai	166187.94

3 Understand the Data

Let's view the available columns and understand the structure of the dataset.

[4]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 5 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   DigitalMarketing  50 non-null     float64
 1   Promotion         50 non-null     float64
 2   Research          50 non-null     float64
 3   State             50 non-null     object
 4   Profit            50 non-null     float64
dtypes: float64(4), object(1)
memory usage: 2.1+ KB
```

[5]:
```python
df.columns #View Columns
```

[5]:
```
Index(['DigitalMarketing', 'Promotion', 'Research', 'State', 'Profit'],
dtype='object')
```

[6]:
```python
df.describe()
```

[6]:

	DigitalMarketing	Promotion	Research	Profit
count	50.000000	50.000000	50.000000	50.000000
mean	73721.615600	121344.639600	211025.097800	112012.639200

```
std        45902.256482   28017.802755  122290.310726   40306.180338
min            0.000000   51283.140000       0.000000   14681.400000
25%        39936.370000  103730.875000  129300.132500   90138.902500
50%        73051.080000  122699.795000  212716.240000  107978.190000
75%       101602.800000  144842.180000  299469.085000  139765.977500
max       165349.200000  182645.560000  471784.100000  192261.830000
```

[7]:
```python
df.isnull().sum()
```

[7]:
```
DigitalMarketing    0
Promotion           0
Research            0
State               0
Profit              0
dtype: int64
```

4 Define Independent & Dependent Variables

[8]:
```python
X = df.iloc[:, :-1]  # All columns except last
y = df.iloc[:, -4]   # Target variable (you can change it based on dataset)
```

5 Encode Categorical Data (if any)

[12]:
```python
X = pd.get_dummies(X, dtype=int)
print(X)
```

```
    DigitalMarketing  Promotion   Research  State_Bangalore  State_Chennai  \
0          165349.20  136897.80  471784.10                0              0
1          162597.70  151377.59  443898.53                1              0
2          153441.51  101145.55  407934.54                0              1
3          144372.41  118671.85  383199.62                0              0
4          142107.34   91391.77  366168.42                0              1
5          131876.90   99814.71  362861.36                0              0
6          134615.46  147198.87  127716.82                1              0
7          130298.13  145530.06  323876.68                0              1
8          120542.52  148718.95  311613.29                0              0
9          123334.88  108679.17  304981.62                1              0
10         101913.08  110594.11  229160.95                0              1
11         100671.96   91790.61  249744.55                1              0
12          93863.75  127320.38  249839.44                0              1
13          91992.39  135495.07  252664.93                1              0
14         119943.24  156547.42  256512.92                0              1
15         114523.61  122616.84  261776.23                0              0
16          78013.11  121597.55  264346.06                1              0
17          94657.16  145077.58  282574.31                0              0
18          91749.16  114175.79  294919.57                0              1
```

19	86419.70	153514.11	0.00	0	0
20	76253.86	113867.30	298664.47	1	0
21	78389.47	153773.43	299737.29	0	0
22	73994.56	122782.75	303319.26	0	1
23	67532.53	105751.03	304768.73	0	1
24	77044.01	99281.34	140574.81	0	0
25	64664.71	139553.16	137962.62	1	0
26	75328.87	144135.98	134050.07	0	1
27	72107.60	127864.55	353183.81	0	0
28	66051.52	182645.56	118148.20	0	1
29	65605.48	153032.06	107138.38	0	0
30	61994.48	115641.28	91131.24	0	1
31	61136.38	152701.92	88218.23	0	0
32	63408.86	129219.61	46085.25	1	0
33	55493.95	103057.49	214634.81	0	1
34	46426.07	157693.92	210797.67	1	0
35	46014.02	85047.44	205517.64	0	0
36	28663.76	127056.21	201126.82	0	1
37	44069.95	51283.14	197029.42	1	0
38	20229.59	65947.93	185265.10	0	0
39	38558.51	82982.09	174999.30	1	0
40	28754.33	118546.05	172795.67	1	0
41	27892.92	84710.77	164470.71	0	1
42	23640.93	96189.63	148001.11	1	0
43	15505.73	127382.30	35534.17	0	0
44	22177.74	154806.14	28334.72	1	0
45	1000.23	124153.04	1903.93	0	0
46	1315.46	115816.21	297114.46	0	1
47	0.00	135426.92	0.00	1	0
48	542.05	51743.15	0.00	0	0
49	0.00	116983.80	45173.06	1	0

	State_Hyderabad
0	1
1	0
2	0
3	1
4	0
5	1
6	0
7	0
8	1
9	0
10	0
11	0
12	0
13	0
14	0

```
15    1
16    0
17    1
18    0
19    1
20    0
21    1
22    0
23    0
24    1
25    0
26    0
27    1
28    0
29    1
30    0
31    1
32    0
33    0
34    0
35    1
36    0
37    0
38    1
39    0
40    0
41    0
42    0
43    1
44    0
45    1
46    0
47    0
48    1
49    0
```

6 Split the Dataset

[13]:
```python
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=0)
```

7 Train the Model

[14]:
```
from sklearn.linear_model import LinearRegression

regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

[14]:
```
LinearRegression()
```

8 Predict and Compare

[15]:
```
y_pred = regressor.predict(X_test)
print(y_pred)
```

```
[182645.56  91790.61 110594.11  84710.77 101145.55 127864.55  65947.93
 152701.92 122782.75  91391.77]
```

[16]:
```
comparison = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
print(comparison.head())
```

```
       Actual  Predicted
28  182645.56  182645.56
11   91790.61   91790.61
10  110594.11  110594.11
41   84710.77   84710.77
2   101145.55  101145.55
```

9 Model Evaluation

[17]:
```
bias = regressor.score(X_train, y_train)
variance = regressor.score(X_test, y_test)

print("Bias (Train Accuracy):", bias)
print("Variance (Test Accuracy):", variance)
```

```
Bias (Train Accuracy): 1.0
Variance (Test Accuracy): 1.0
```

10 Model Parameters

[19]:
```
m_slope = regressor.coef_
print("Slope/Co-efficients (b ):", m_slope)
```

```
Slope/Co-efficients (b ): [-3.58815785e-16  1.00000000e+00 -3.89677049e-17
-1.53880995e-13
  3.15122730e-13 -1.61241735e-13]
```

[20]:
```python
c_intercept = regressor.intercept_
print("Intercept (b ):", c_intercept)
```

```
Intercept (b ): 8.731149137020111e-11
```

11 Add Constant for OLS

OLS (Ordinary Least Squares) from `statsmodels` requires a constant column to represent the intercept.

12 Add Constant Column

[21]:
```python
X = np.append(arr=np.ones((X.shape[0], 1)).astype(int), values=X, axis=1)
```

13 Backward Elimination (Full Features)

[23]:
```python
import statsmodels.api as sm

X_opt = X[:, [0, 1, 2, 3, 4, 5]]  # Example: adjust as per your column count
regressor_OLS = sm.OLS(endog=y, exog=X_opt).fit()
regressor_OLS.summary()
```

[23]:

Dep. Variable:	Promotion	**R-squared:**	1.000
Model:	OLS	**Adj. R-squared:**	1.000
Method:	Least Squares	**F-statistic:**	7.500e+29
Date:	Fri, 11 Jul 2025	**Prob (F-statistic):**	0.00
Time:	11:07:08	**Log-Likelihood:**	1082.9
No. Observations:	50	**AIC:**	-2154.
Df Residuals:	44	**BIC:**	-2142.
Df Model:	5		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P> |t|**	**[0.025**	**0.975]**
const	-3.638e-11	7.46e-11	-0.488	0.628	-1.87e-10	1.14e-10
x1	-1.943e-16	4.98e-16	-0.390	0.698	-1.2e-15	8.09e-16
x2	1.0000	5.6e-16	1.78e+15	0.000	1.000	1.000
x3	3.469e-16	1.84e-16	1.886	0.066	-2.37e-17	7.18e-16
x4	2.183e-11	3.49e-11	0.625	0.535	-4.86e-11	9.22e-11
x5	7.276e-12	3.58e-11	0.203	0.840	-6.49e-11	7.95e-11

Omnibus:	0.163	**Durbin-Watson:**	0.214
Prob(Omnibus):	0.922	**Jarque-Bera (JB):**	0.041
Skew:	0.066	**Prob(JB):**	0.980
Kurtosis:	2.953	**Cond. No.**	1.47e+06

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The condition number is large, 1.47e+06. This might indicate that there are strong multicollinearity or other numerical problems.

OLS Regression Results – Full Explanation

Context:

You ran the following code:

```
X_opt = X[:, [0, 1, 2, 3, 4, 5]]
regressor_OLS = sm.OLS(endog=y, exog=X_opt).fit()
regressor_OLS.summary()
```

Part-by-Part Explanation of the Output

Header Summary

Term	Meaning
Dep. Variable	`Promotion`: The dependent variable (target/output)
Model	`OLS` = Ordinary Least Squares Regression
Method	Least Squares method used to estimate coefficients
No. Observations	50 data points (rows in dataset)
Df Residuals	44 = 50 - 6 → total observations minus number of model coefficients
Df Model	5 → You used 5 predictors (x1 to x5)
Covariance Type	Non-robust standard errors used

Model Quality Metrics

Metric	Value	Meaning
R-squared	`1.000`	Model explains **100%** of variance in target. This usually indicates **overfitting** or multicollinearity.
Adj. R-squared	`1.000`	Adjusted for number of predictors. Still 1.000 → very high!
F-statistic	`7.5e+29`	Very high → model is statistically significant
Prob (F-statistic)	`0.00`	p-value for F-test is 0 → the model overall is significant
AIC / BIC	AIC: `-2154`, BIC: `-2142`	Lower values indicate a better model (used for comparing models)

Coefficient Table (Main Focus)

Term	Coef	Std Err	t	P >	Significance
const	-3.638e-11	7.46e-11	-0.488	0.628	Not significant
x1	-1.943e-16	4.98e-16	-0.390	0.698	Not significant
x2	1.0000	5.6e-16	1.78e+15	0.000	Highly significant
x3	3.469e-16	1.84e-16	1.886	0.066	Borderline
x4	2.183e-11	3.49e-11	0.625	0.535	Not significant

Term	Coef	Std Err	t	P >	Significance
x5	7.276e-12	3.58e-11	0.203	0.840	Not significant

How to Interpret This?

- **Coef (Coefficient)**: The amount of change in the target variable for 1 unit change in that predictor.
- **P > |t| (p-value)**: Tells you whether the variable is statistically significant.
  - If **p < 0.05**, the variable is **significant** → Keep it.
  - If **p 0.05**, the variable is **not significant** → You can remove it (Backward Elimination).
- **t** and **Std Err**: Used internally to compute the p-value.
- Based on this table:
  - Keep only `x2`
  - Remove variables like `x1`, `x4`, `x5`

Statistical Tests

Metric	Value	Meaning
Omnibus / JB	These test if residuals are normally distributed (good if p > 0.05)	
Durbin-Watson	`0.214` → indicates **strong positive autocorrelation** (bad!)	
Skew / Kurtosis	`Skew  0`, `Kurtosis  3` → residuals are normally distributed	
Cond. No.	`1.47e+06` → very high → indicates **multicollinearity risk** (bad)	

What to Do Next? → Backward Elimination

Based on this summary:

- Remove the variable with **highest p-value > 0.05** → `x5` (`0.840`)
- Rerun OLS without it
- Repeat the process until all p-values < 0.05

OLS Regression Results

We started with 5 independent variables (x1 to x5) and added a constant term.

Summary Highlights:

- **R-squared = 1.000** → Model fits data perfectly (possible overfitting)
- **Only x2 is statistically significant (p < 0.05)**
- **x1, x4, x5 have high p-values → should be removed**
- **Durbin-Watson = 0.214** → Indicates autocorrelation (not ideal)
- **Condition Number = 1.47e+06** → Multicollinearity suspected

Next Step: Perform **Backward Elimination**: - Remove variable with highest p-value (`x5`) - Rerun the model - Continue until all p-values < 0.05

14 Remove Feature with Highest p-value

Keep repeating this by removing the feature with the highest p-value above 0.05 until all remaining features are significant.

[24]:
```
X_opt = X[:, [0, 1, 2, 3, 4]]   # Removed 5th feature
regressor_OLS = sm.OLS(endog=y, exog=X_opt).fit()
regressor_OLS.summary()
```

[24]:

Dep. Variable:	Promotion	**R-squared:**	1.000
Model:	OLS	**Adj. R-squared:**	1.000
Method:	Least Squares	**F-statistic:**	1.217e+30
Date:	Fri, 11 Jul 2025	**Prob (F-statistic):**	0.00
Time:	11:22:39	**Log-Likelihood:**	1088.9
No. Observations:	50	**AIC:**	-2168.
Df Residuals:	45	**BIC:**	-2158.
Df Model:	4		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	-1.164e-10	6.47e-11	-1.798	0.079	-2.47e-10	1.4e-11
x1	-1.943e-16	4.35e-16	-0.447	0.657	-1.07e-15	6.82e-16
x2	1.0000	4.91e-16	2.04e+15	0.000	1.000	1.000
x3	-3.886e-16	1.59e-16	-2.442	0.019	-7.09e-16	-6.81e-17
x4	-7.276e-12	2.69e-11	-0.270	0.788	-6.15e-11	4.7e-11

Omnibus:	1.887	**Durbin-Watson:**	0.742
Prob(Omnibus):	0.389	**Jarque-Bera (JB):**	1.425
Skew:	-0.208	**Prob(JB):**	0.491
Kurtosis:	2.285	**Cond. No.**	1.44e+06

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.44e+06. This might indicate that there are
strong multicollinearity or other numerical problems.

15 Final Model after Elimination

[25]:
```
X_opt = X[:, [0, 2]]   # Final selected features
regressor_OLS = sm.OLS(endog=y, exog=X_opt).fit()
regressor_OLS.summary()
```

[25]:

Dep. Variable:	Promotion	**R-squared:**	1.000
Model:	OLS	**Adj. R-squared:**	1.000
Method:	Least Squares	**F-statistic:**	4.132e+31
Date:	Fri, 11 Jul 2025	**Prob (F-statistic):**	0.00
Time:	11:23:02	**Log-Likelihood:**	1140.7
No. Observations:	50	**AIC:**	-2277.
Df Residuals:	48	**BIC:**	-2274.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P> |t|	[0.025	0.975]
const	-5.457e-11	1.94e-11	-2.818	0.007	-9.35e-11	-1.56e-11
x1	1.0000	1.56e-16	6.43e+15	0.000	1.000	1.000

Omnibus:	50.890	**Durbin-Watson:**	0.057
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	632.888
Skew:	-2.078	**Prob(JB):**	3.72e-138
Kurtosis:	19.927	**Cond. No.**	5.59e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.59e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

16 Concept Highlights

- **API** (Application Programming Interface): Connects front-end to back-end.
- **OLS**: A statistical method to fit linear regression.
- **p-value**: Helps determine if a feature is statistically significant ($p < 0.05$ is good).
- **T-test**: Performed on sample data to test hypothesis.
- **Backward Elimination**: A feature selection method based on p-values.
- **Adjusted $R^2 > R^2$**: Indicates a better, more reliable model when adding/removing variables.

17 Final Decision: Which Department to Focus On?

Based on statistical analysis using **Backward Elimination in OLS**, the model retained only **one important feature (x1)**, which:

- Has a **p-value = 0.000** → highly significant
- Has **coefficient = 1.0** → 1 unit increase in x1 increases promotion by 1 unit
- Explains **100% of the variation** in the Promotion outcome ($R^2 = 1.000$)

This means the department or factor represented by `x1` is **most directly responsible** for driving promotions.

18 Recommendation to Company:

Focus your time, budget, and resources on the department represented by **x1** is **Department_Marketing** — this is the best area to invest in for maximizing promotions and growth.