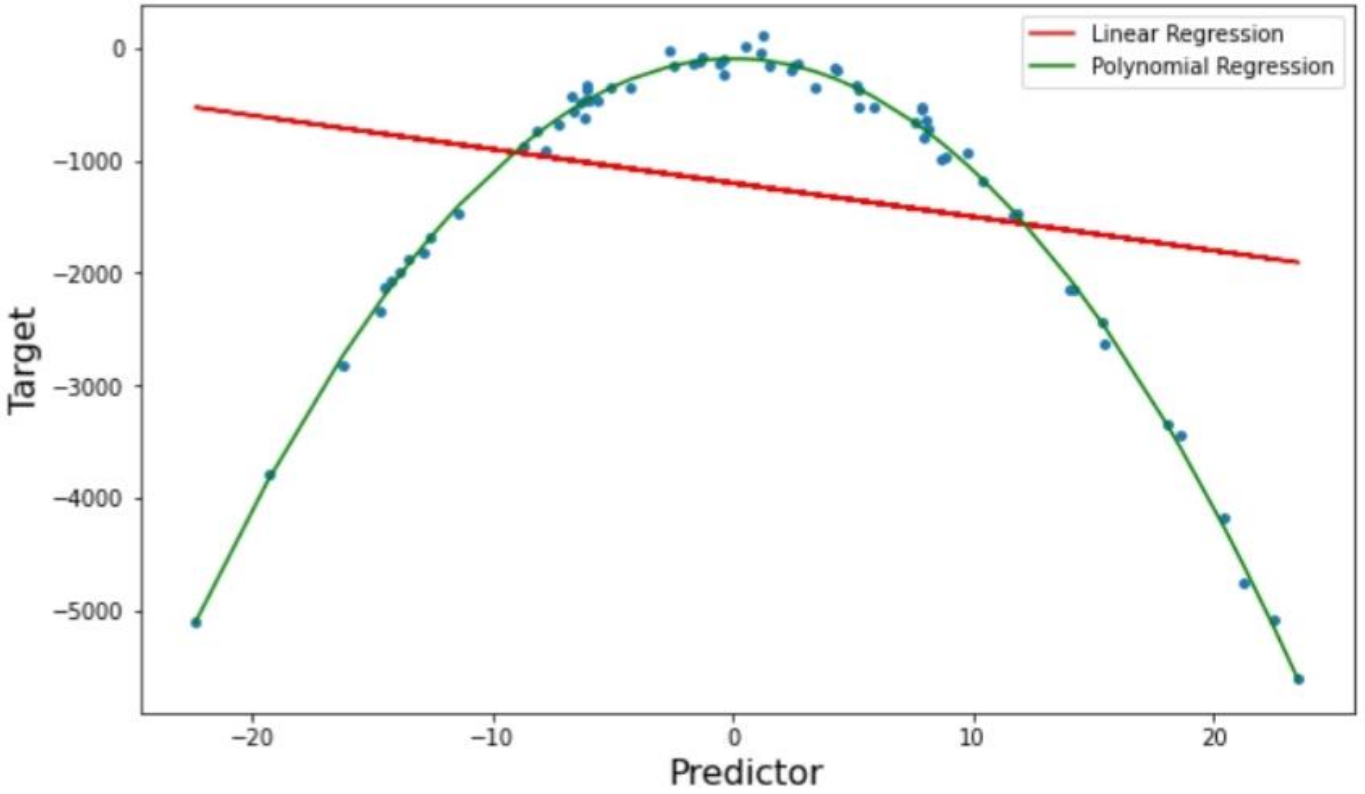
Machine Learning

- 1st Term, 2025/2026
- December 2025

- **Prof. Mohammed A. Al Ghamdi**

Nonlinear Regression



Nonlinear Regression:

- Nonlinear regression is a mathematical model that fits an equation to certain data using a generated line.
- It shows association using a curve, making it nonlinear in the parameter.

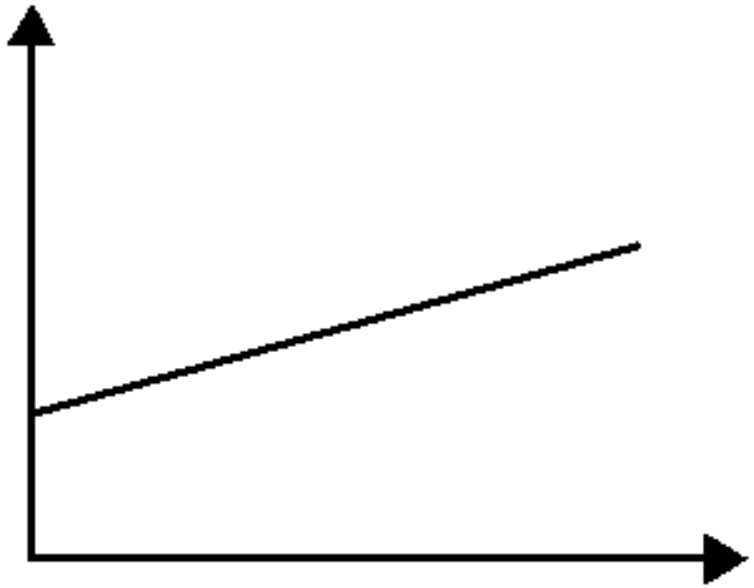
Nonlinear Regression :

$$Y_i = h\left[x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(m)}; \Theta_1, \Theta_2, \ldots, \Theta_p\right] + E_i$$

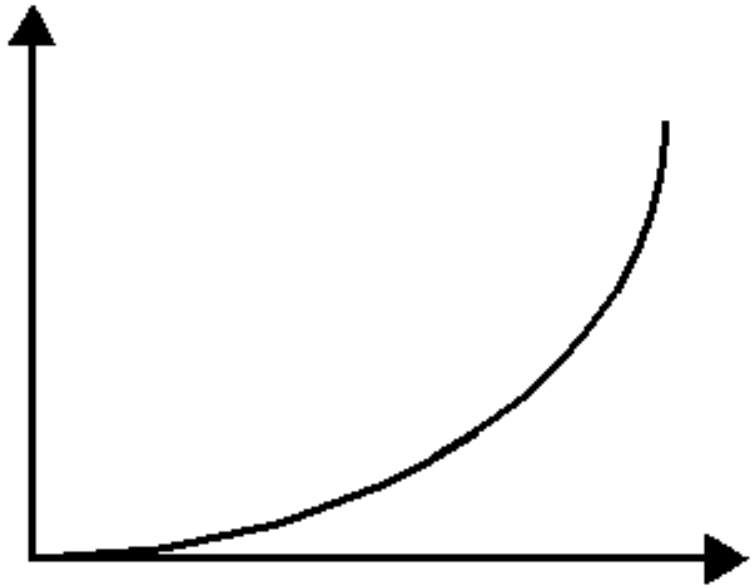
Where:

- Y_i is the responsive variable.
- h is the function.
- x is the input.
- Θ is the parameter to be estimated.

Nonlinear vs Linear:



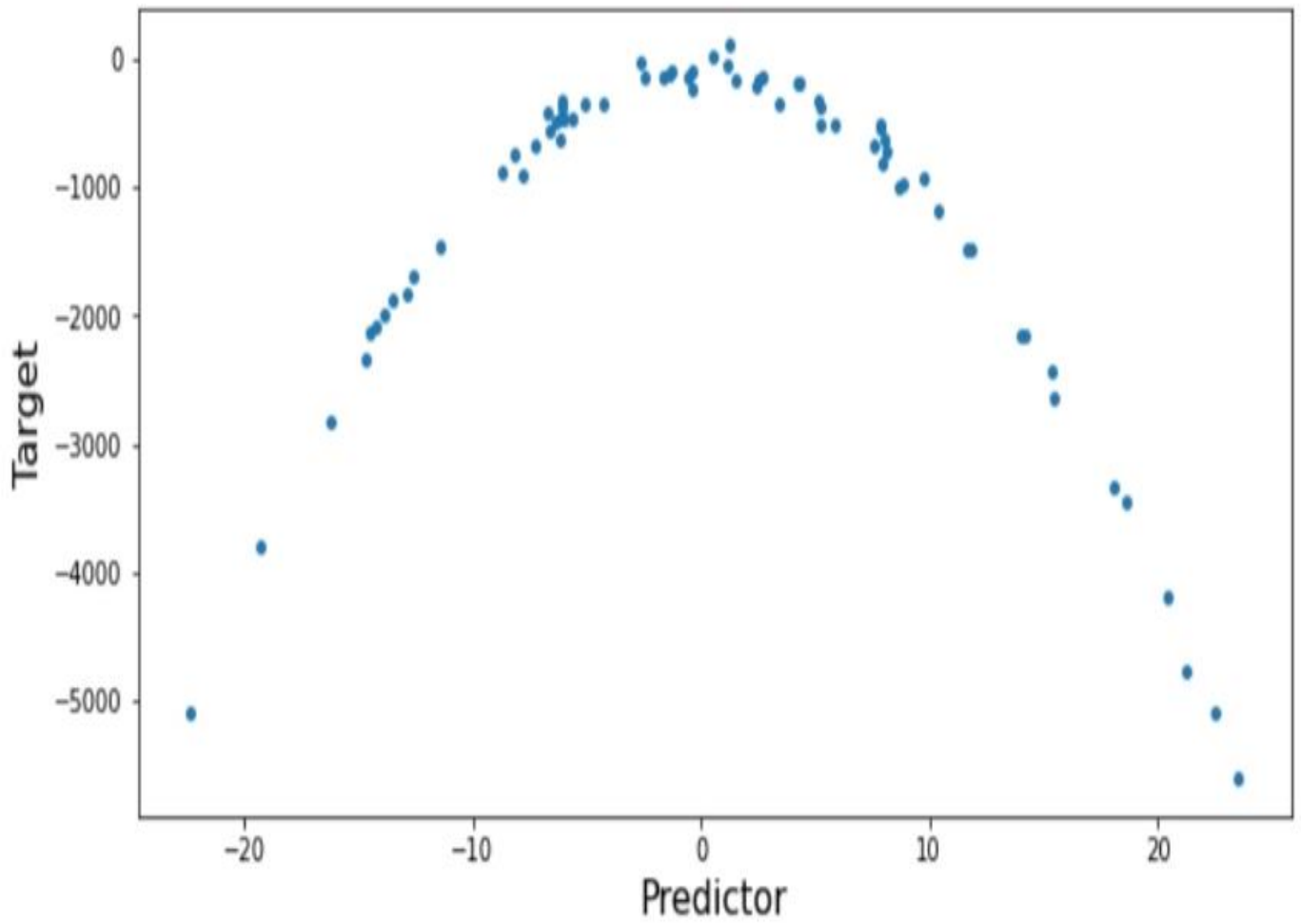
Linear relationship
Number of personnel vs. employee costs



Nonlinear relationship
Population growth over time

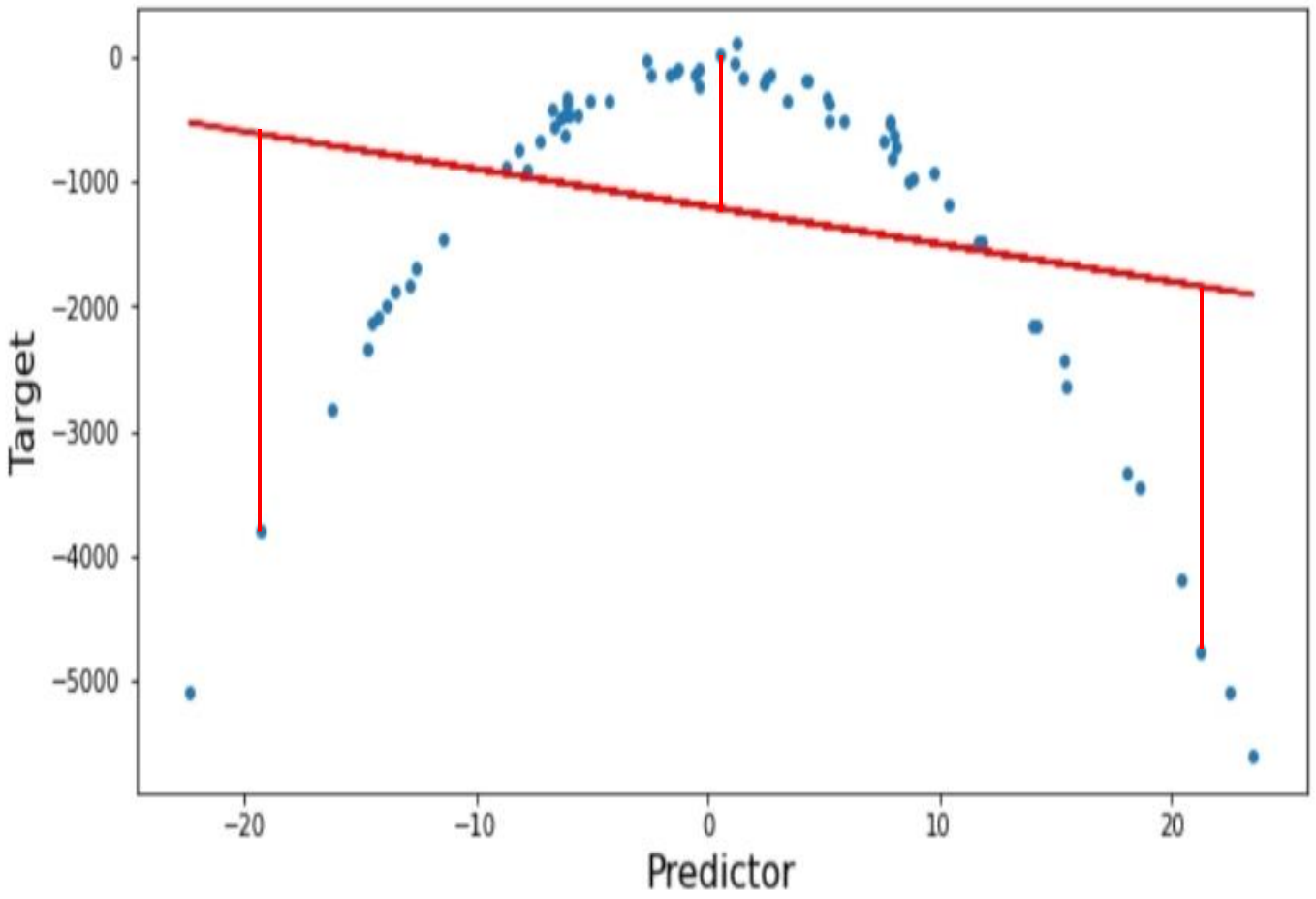
Nonlinear Example:

- In Linear Regression, the main step is to find the Best Fit Line.
- For this graph, the best fit line will be as follow.



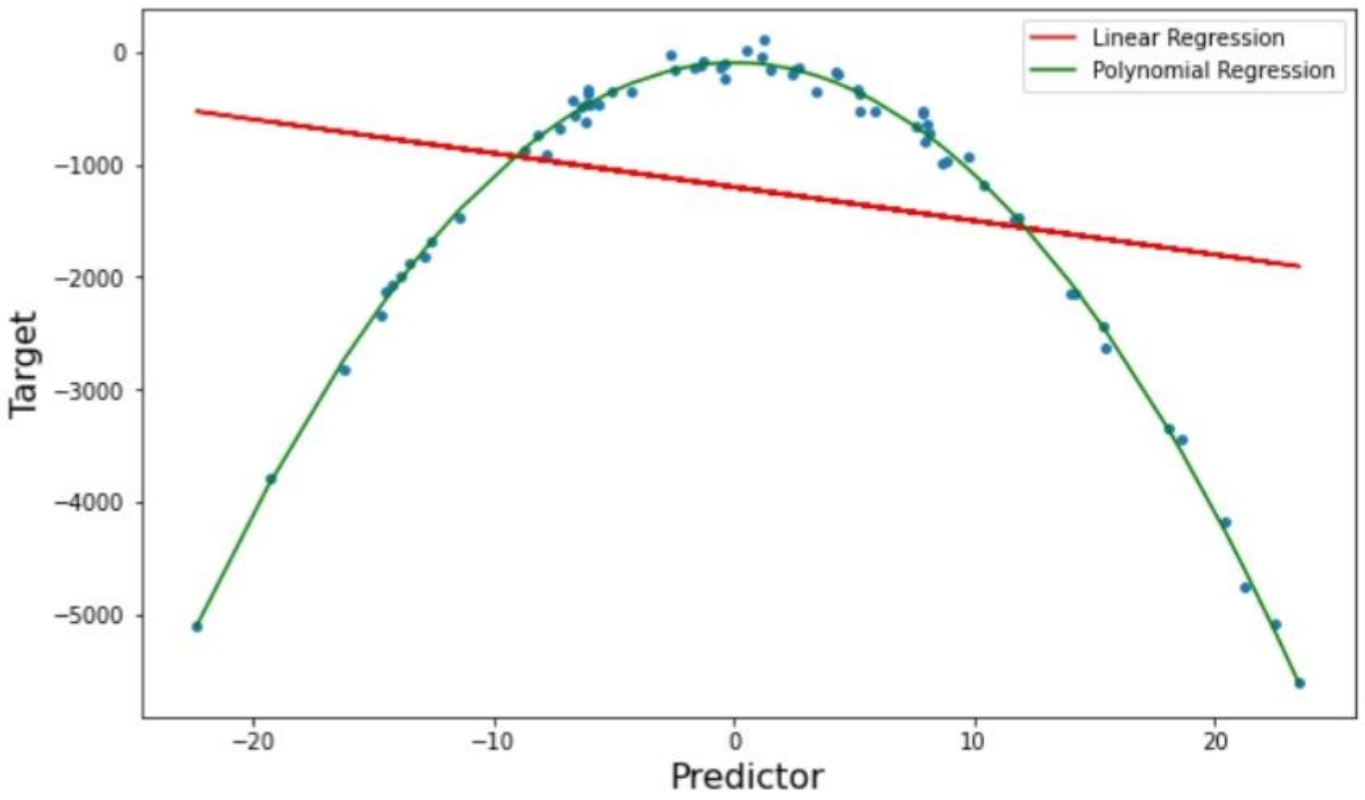
Nonlinear Example:

- The Best Fit Line represent the prediction model using normal linear equation.
- Cost function is high (represented by distance between actual value and predicted value) = High Bias.
- Increasing the Complexity to enhance the prediction model.
- How?



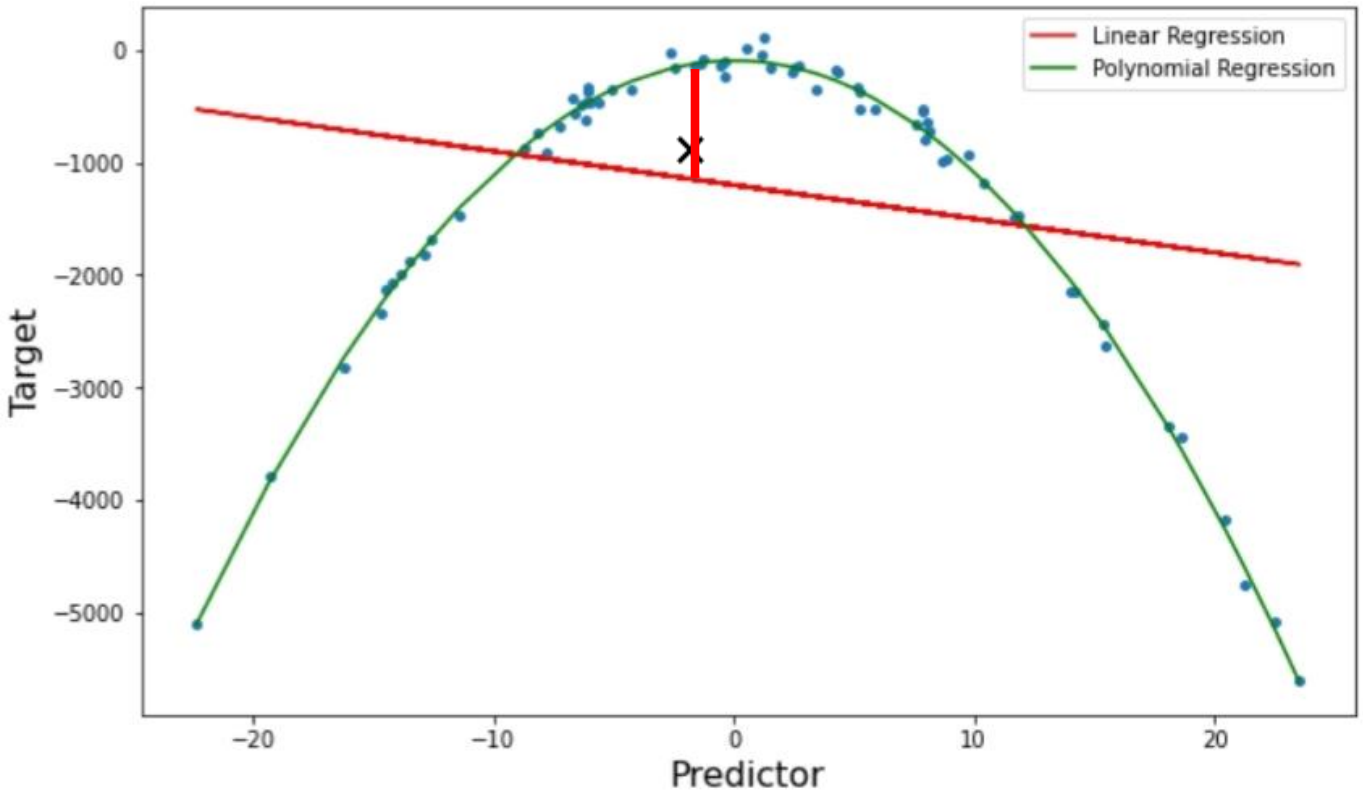
Nonlinear Example:

- Using Polynomial Function to detect the changes of data.
- No much difference between the actual value and predicted value.
- => Low Bias.



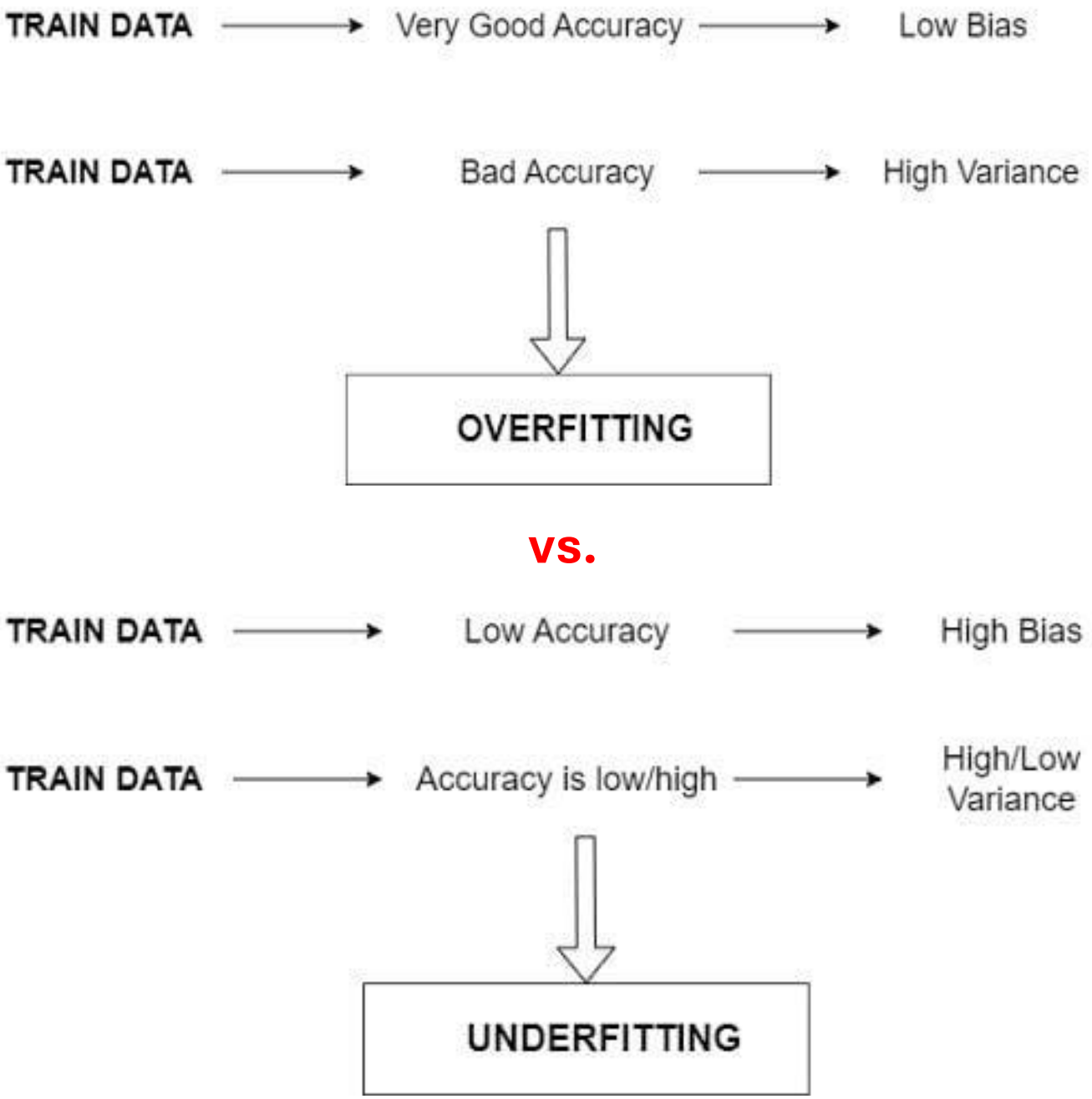
Nonlinear Example:

- High Variance may occur during the testing even within Polynomial function.
- Linear regression behaves healthy in this case!.
- Bias Variance Problem, nice bias and not variance.
- Underfitting vs Overfitting.



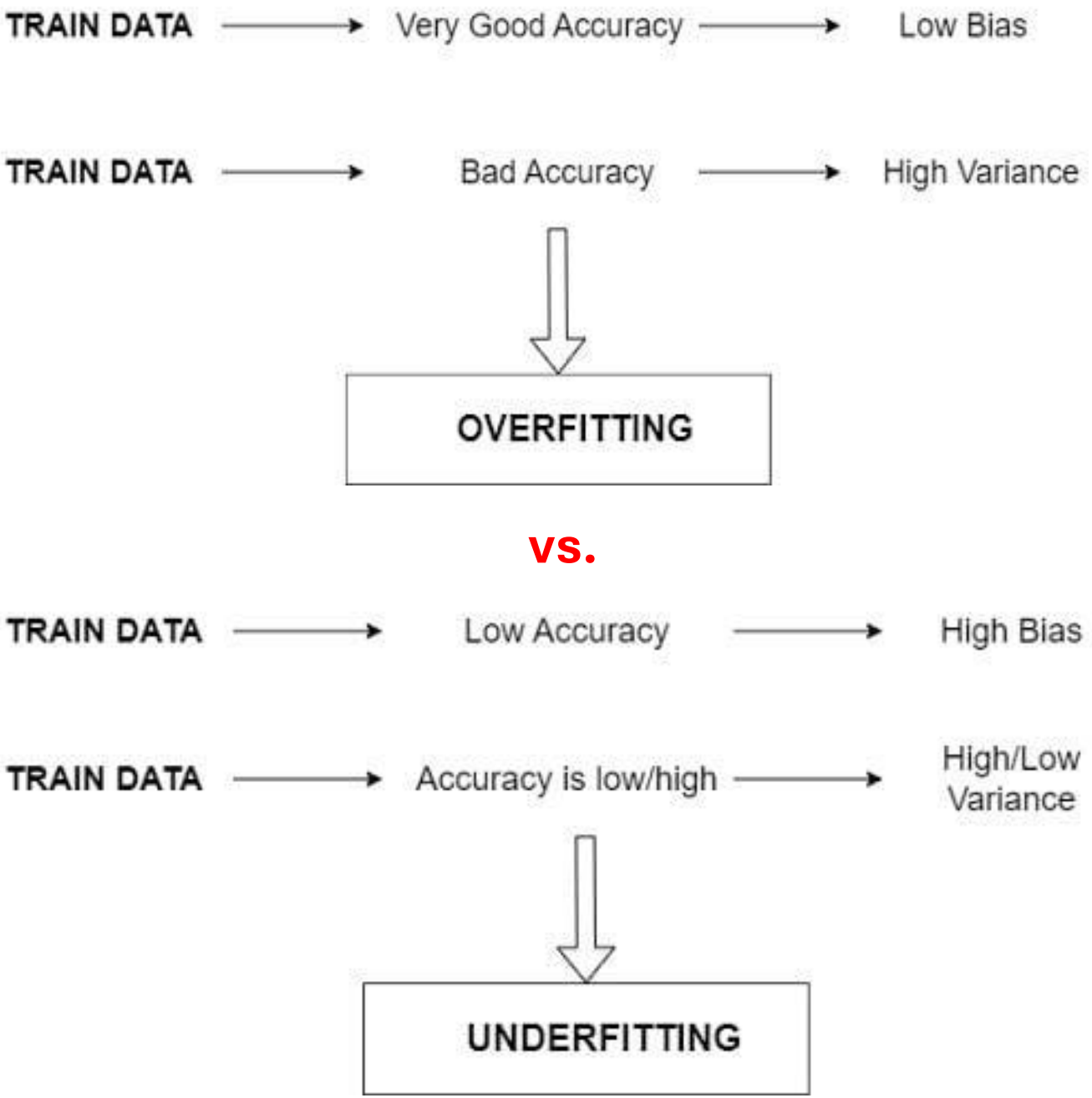
Nonlinear Example:

- The model performs well on training data, but it won't be able to predict accurate outcomes for new = Overfitting.
- The model will perform poorly on both the training and the test data.
- Both overfitting and underfitting are common issue and have impact the model's performance.



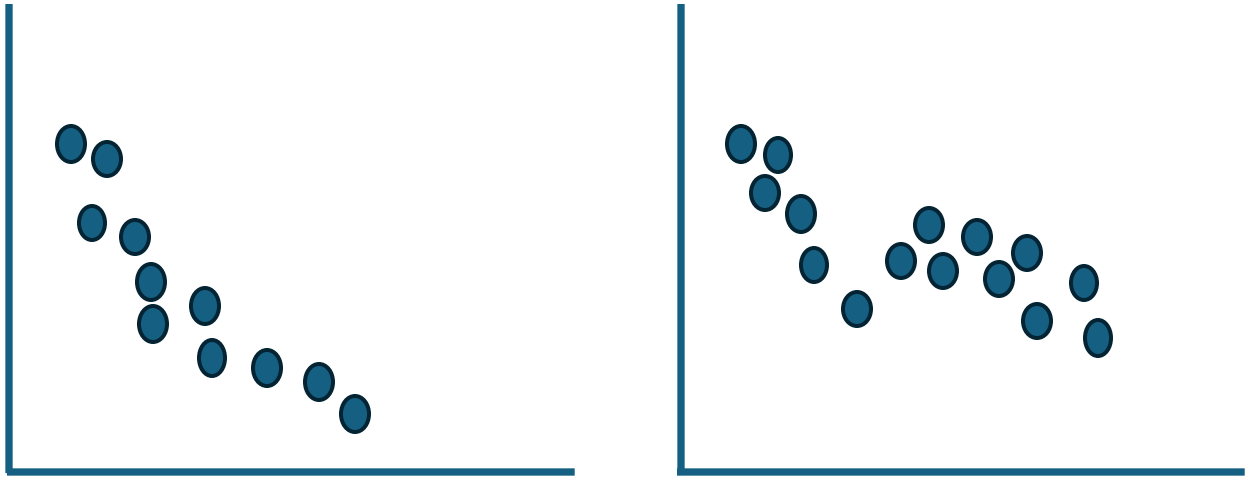
Nonlinear Example:

- It's important to choose an appropriate model for the given data set.
- Reducing the model complexity can help in overfitting case.
- Similarly, for underfitting, increasing the model complexity can improve results.
- Overfitting is caused by too many features in the data set, and underfitting is caused by too few features.



Nonlinear Regression

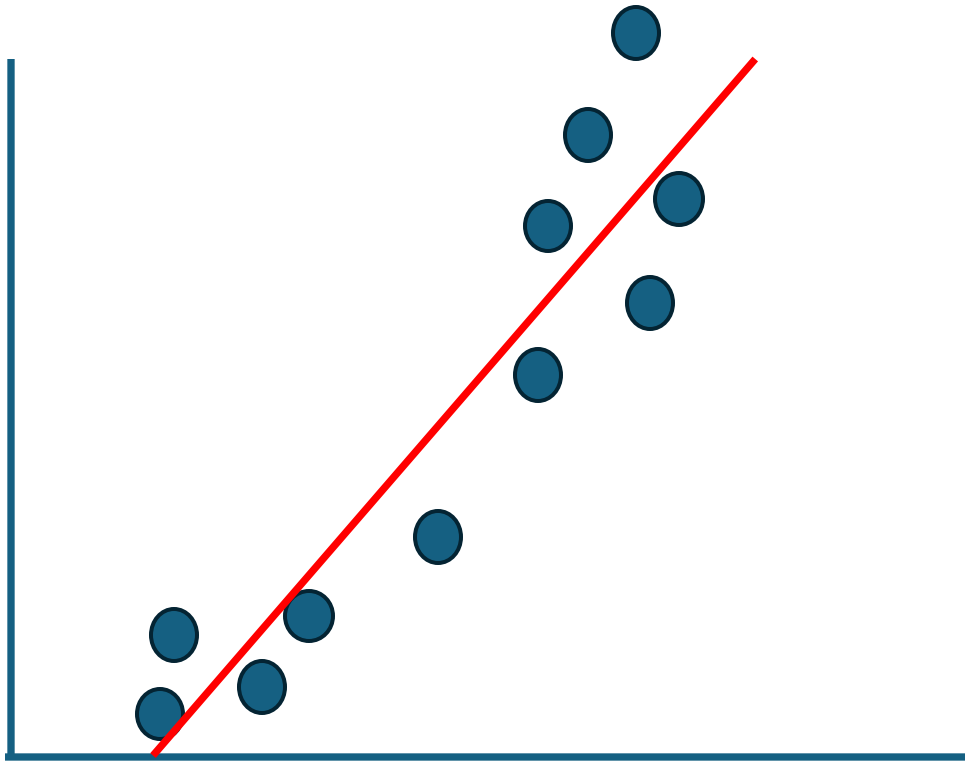
- In linear regression, the Linear equation is applied to find the Best Fit Line.
- As discussed, Linear regression doesn't fit always well for the data.



Nonlinear Regression

- The data is represented as a curve up model.
- Trying to apply Linear Regression method using (i.e.):

$$Y = 1.3X - 0.2$$

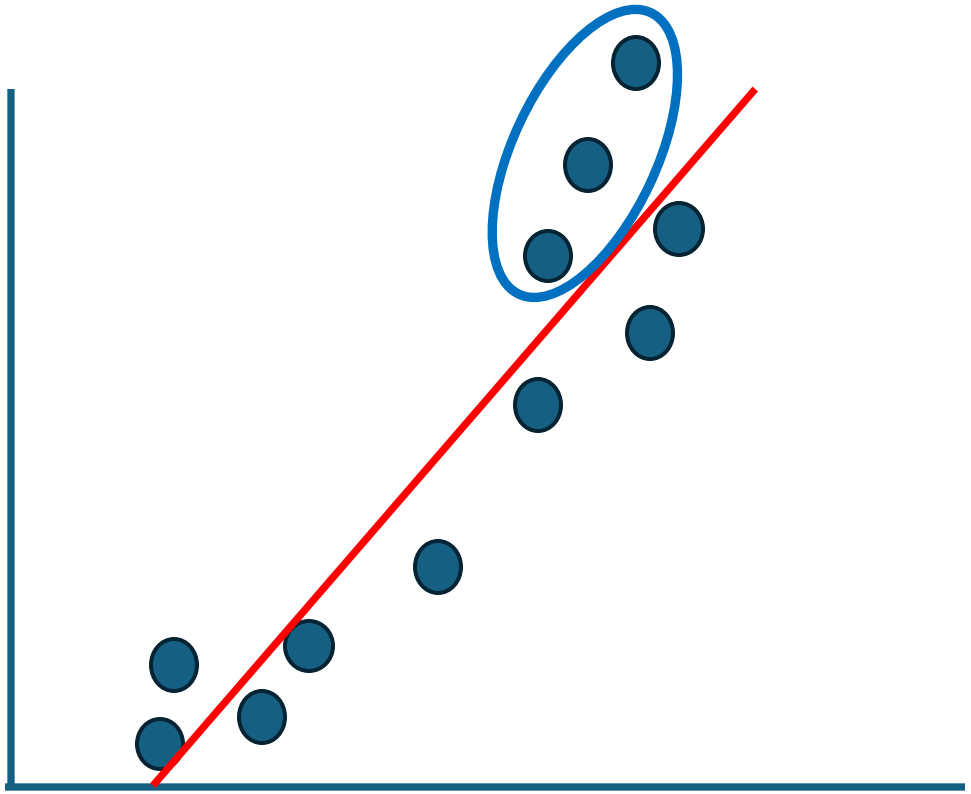


Nonlinear Regression

- The data is represented as a curve up model.
- Trying to apply Linear Regression method using (i.e.):

$$Y = 1.3X - 0.2$$

- The Linear regression method can’t catch the changes on data.
- So, the Nonlinear regression method may apply to deal in this case.

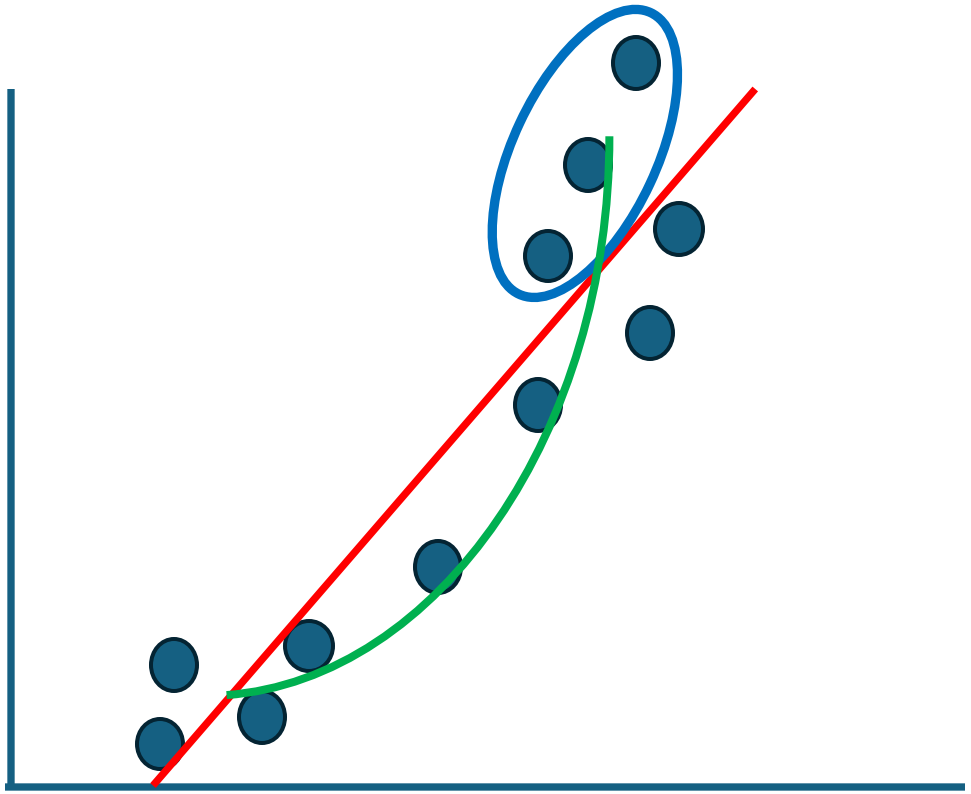


Polynomial Regression

- Polynomial regression method may fit in such case (i.e.).

$$Y = 2.26X^2 - 0.6X + 0.16$$

- The new regression (Quadratic function) is given better model.



Polynomial Regression

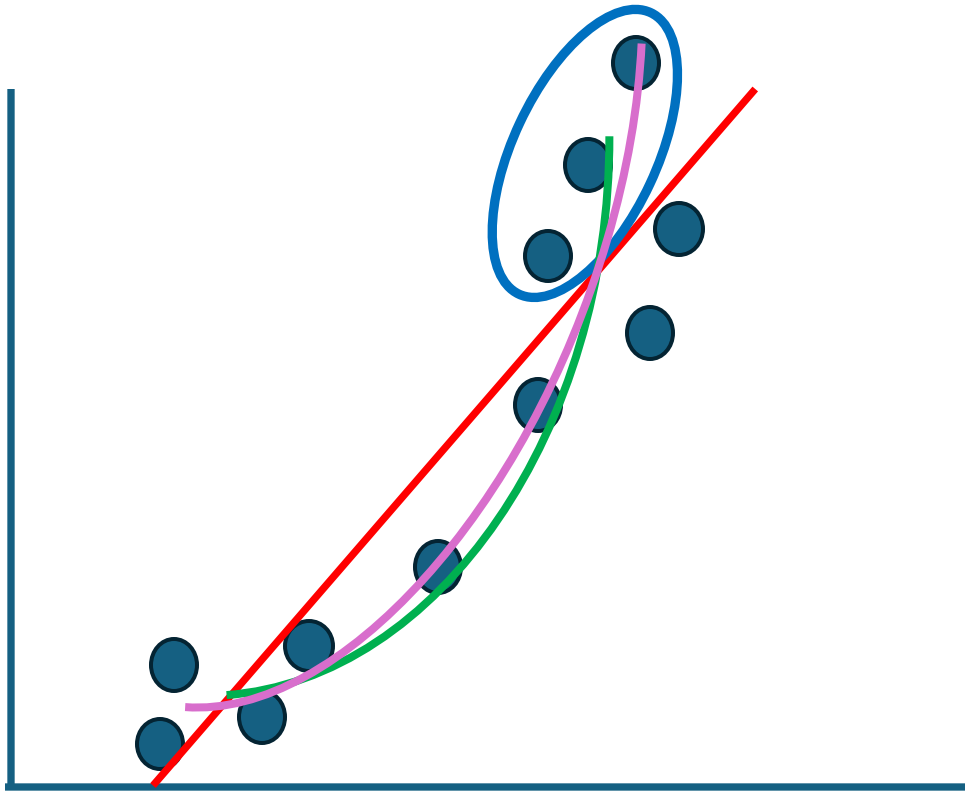
- Polynomial regression method may fit in such case (i.e.).

$$Y = 2.26X^2 - 0.6X + 0.16$$

- The new regression (Quadratic function) is given better model.
- Going further and check a new regression (The new regression (Cubic function) is given better model.

$$Y = 3.07X^3 - 1.98X^2 + 2.6X - 0.04$$

- Much better in catching the changes over the proposed data.
- The target is to find out the polynomial degree to fit nicely.



Example (Polynomial Regression vs. Linear Regression):

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

np.random.seed(1)
x_1 = np.absolute(np.random.randn(100, 1) * 10)
x_2 = np.absolute(np.random.randn(100, 1) * 30)
y = 2*x_1**2 + 3*x_1 + 2 + np.random.randn(100, 1)*20

df = pd.DataFrame({"x_1":x_1.reshape(100,), "x_2":x_2.reshape(100,), "y":y.reshape(100,)},
index=range(0,100))
X, y = df[["x_1", "x_2"]], df["y"]

poly = PolynomialFeatures(degree=2, include_bias=False)
poly_features = poly.fit_transform(X)
```

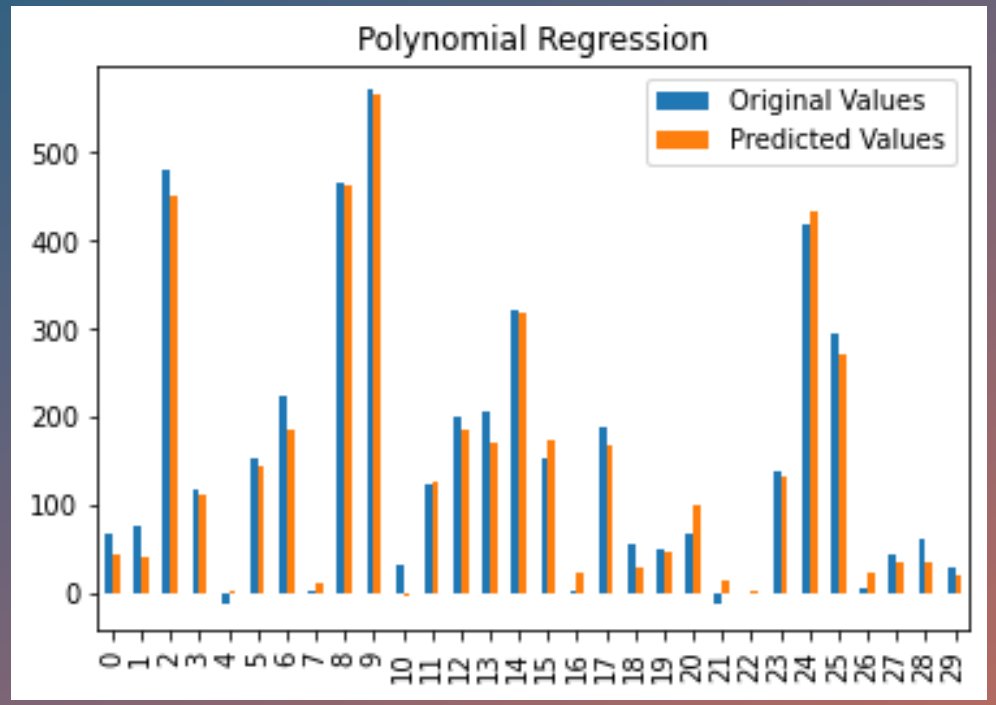
Example:

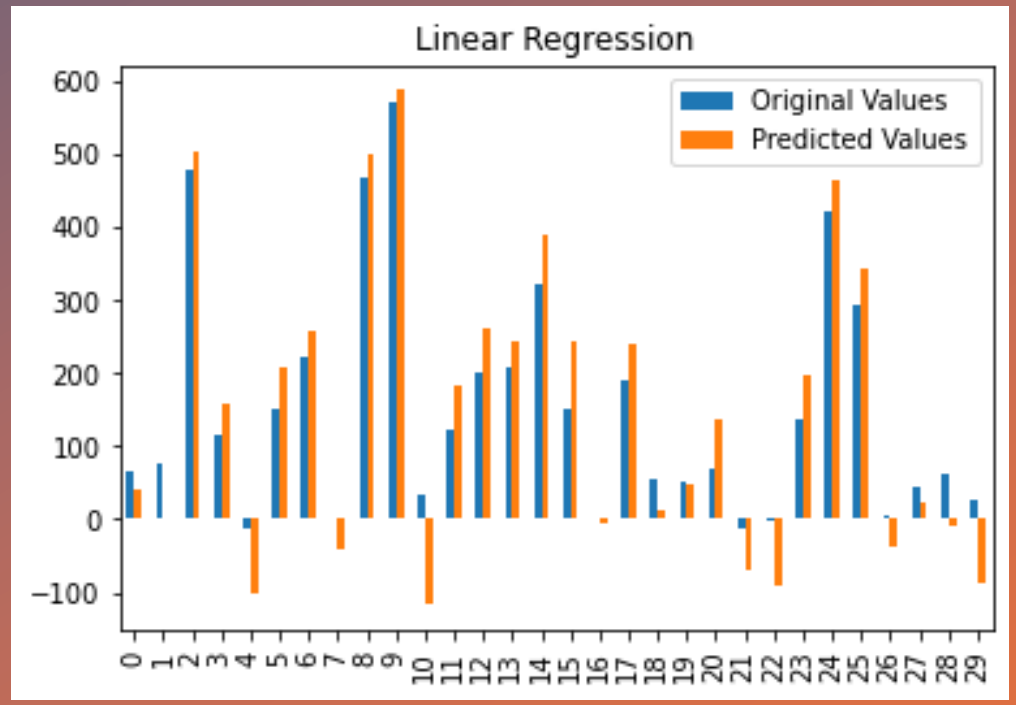
```
X_train, X_test, y_train, y_test = train_test_split(poly_features, y, test_size=0.3, random_state=42)
poly_reg_model = LinearRegression()
poly_reg_model.fit(X_train, y_train)
poly_reg_y_predicted = poly_reg_model.predict(X_test)
poly_reg_rmse = np.sqrt(mean_squared_error(y_test, poly_reg_y_predicted))
print('The RMSE for the Polynomial Regression Model is :', round(poly_reg_rmse,3))

my_list = list(y_test)
df2 = pd.DataFrame({
    'Original Values': my_list,
    'Predicted Values': poly_reg_y_predicted
})
df2.plot(y=["Original Values", "Predicted Values"], kind="bar", title = 'Polynomial Regression')
```

Example:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
lin_reg_model = LinearRegression()
lin_reg_model.fit(X_train, y_train)
lin_reg_y_predicted = lin_reg_model.predict(X_test)
lin_reg_rmse = np.sqrt(mean_squared_error(y_test, lin_reg_y_predicted))
print('The RMSE for the Linear Regression Model is :', round(lin_reg_rmse,3))

my_list = list(y_test)
df3 = pd.DataFrame({
    'Original Values': my_list,
    'Predicted Values': lin_reg_y_predicted
})
df3.plot(y=["Original Values", "Predicted Values"], kind="bar", title = 'Linear Regression')
```

Output:






```
The RMSE for the Polynomial Regression Model is : 20.938
The RMSE for the Linear Regression Model is : 62.302
```

Thanks