Module 04 – Project

SALES PREDICTION

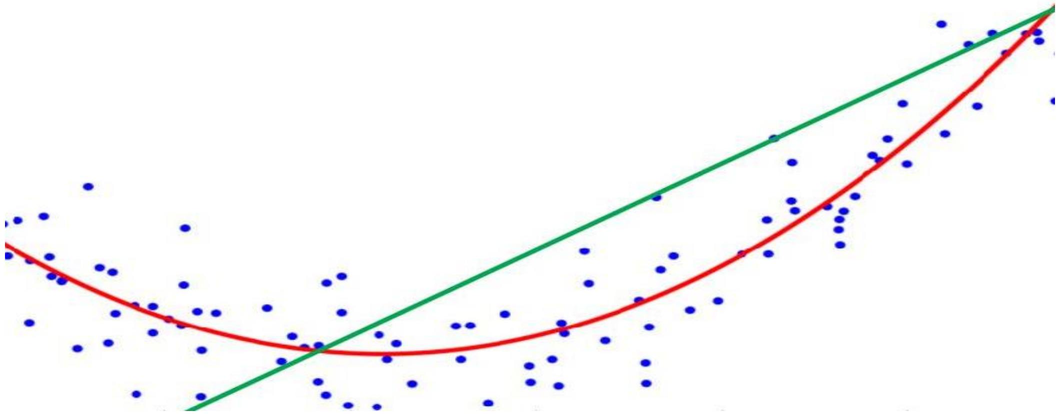
Nguyen Quoc Thai

Objectives

Regression

- Regression Task
- Linear Regression
- Non-linear Regression
- Using Sklearn library

	TV	Radio	Social Media	Influencer	Sales
0	16.0	6.566231	2.907983	Mega	54.732757
1	13.0	9.237765	2.409567	Mega	46.677897
2	41.0	15.886446	2.913410	Mega	150.177829
3	83.0	30.020028	6.922304	Mega	298.246340
4	15.0	8.437408	1.405998	Micro	56.594181



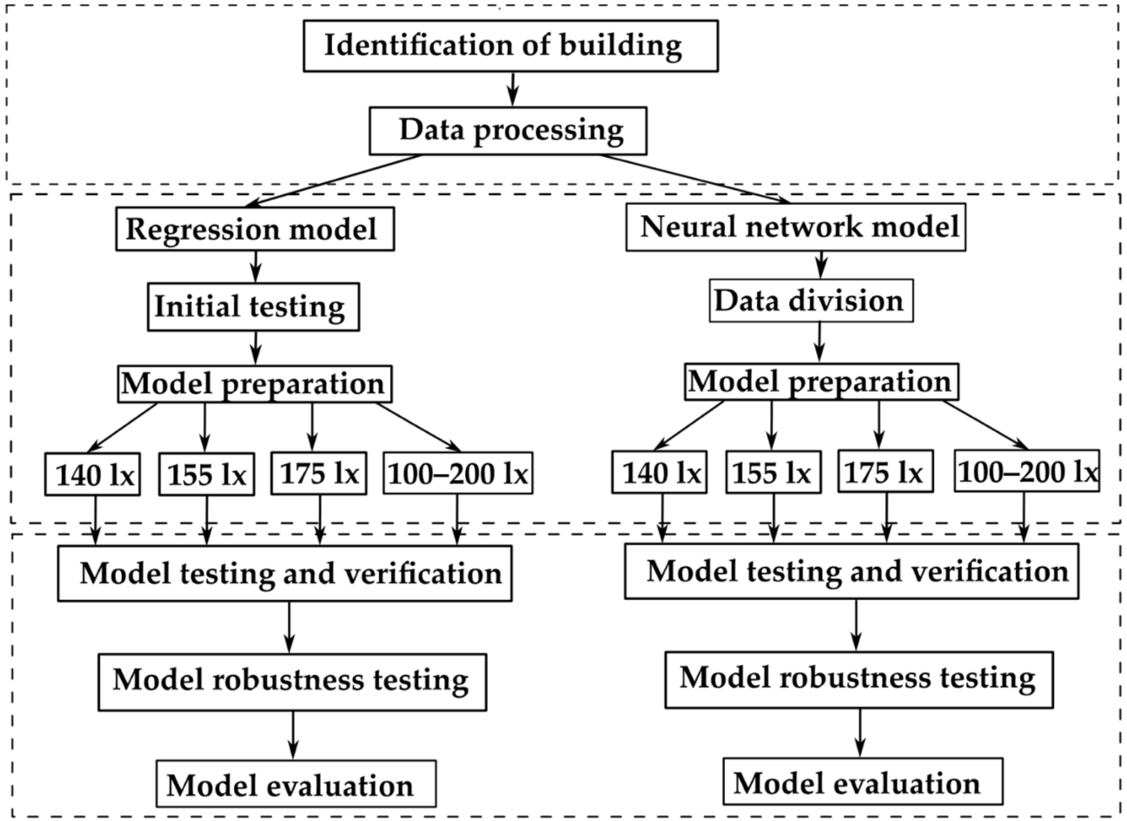
Sales Prediction

- Exploratory Data Analysis (EDA)
- Feature Scaling
- Modeling
- Evaluation
- Custom Polynomial Features

Research Ideas

Research ideas for polynomial regression applications

[A Comparative Analysis of Polynomial Regression and Artificial Neural Networks for Prediction of Lighting Consumption]

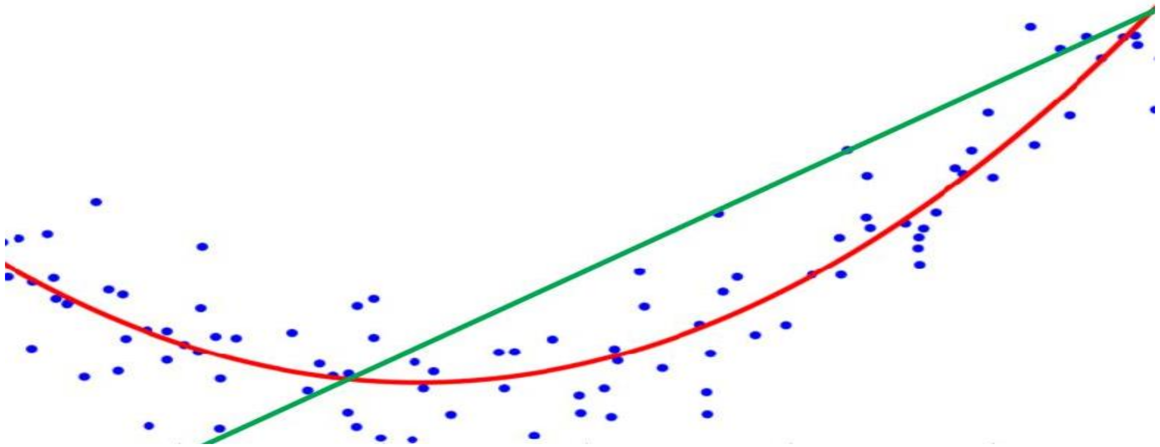


Các nhóm (Gồm các thành viên thuộc AIO) có lời giải sơ bộ, AD Vinh sẽ hướng dẫn tiếp để ra paper.

Outline

SECTION 1

Regression



SECTION 2

Sales Prediction

	TV	Radio	Social Media	Influencer	Sales
0	16.0	6.566231	2.907983	Mega	54.732757
1	13.0	9.237765	2.409567	Mega	46.677897
2	41.0	15.886446	2.913410	Mega	150.177829
3	83.0	30.020028	6.922304	Mega	298.246340
4	15.0	8.437408	1.405998	Micro	56.594181

Regression

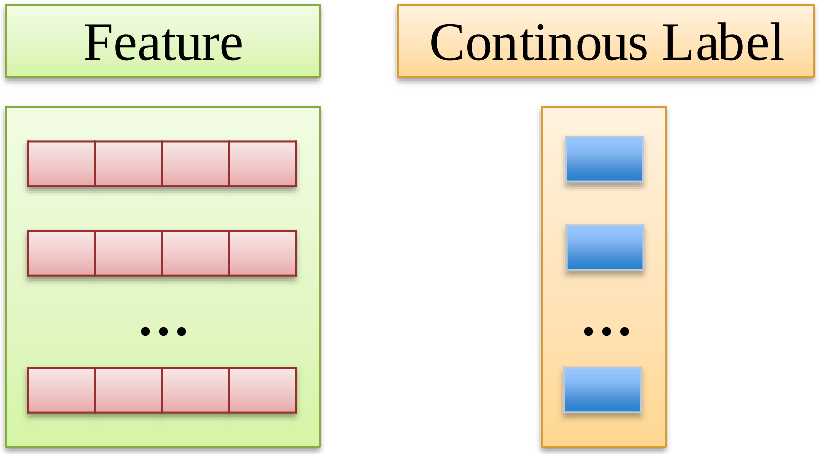
Regression Task

Regression

- Predict a continuous value based on the input variables

What will be the temperature tomorrow?

Training Data



Test Data

Regression

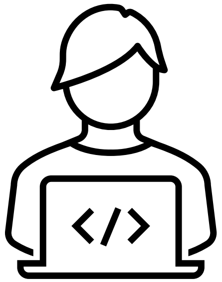
Regression Task

Data

Level	Salary
0	8
1	15
2	18
3	22
4	26
5	30
6	38
7	47

Level	Salary
3,5	?
10	?

Learning



Prediction

Regression

Linear Regression

Data

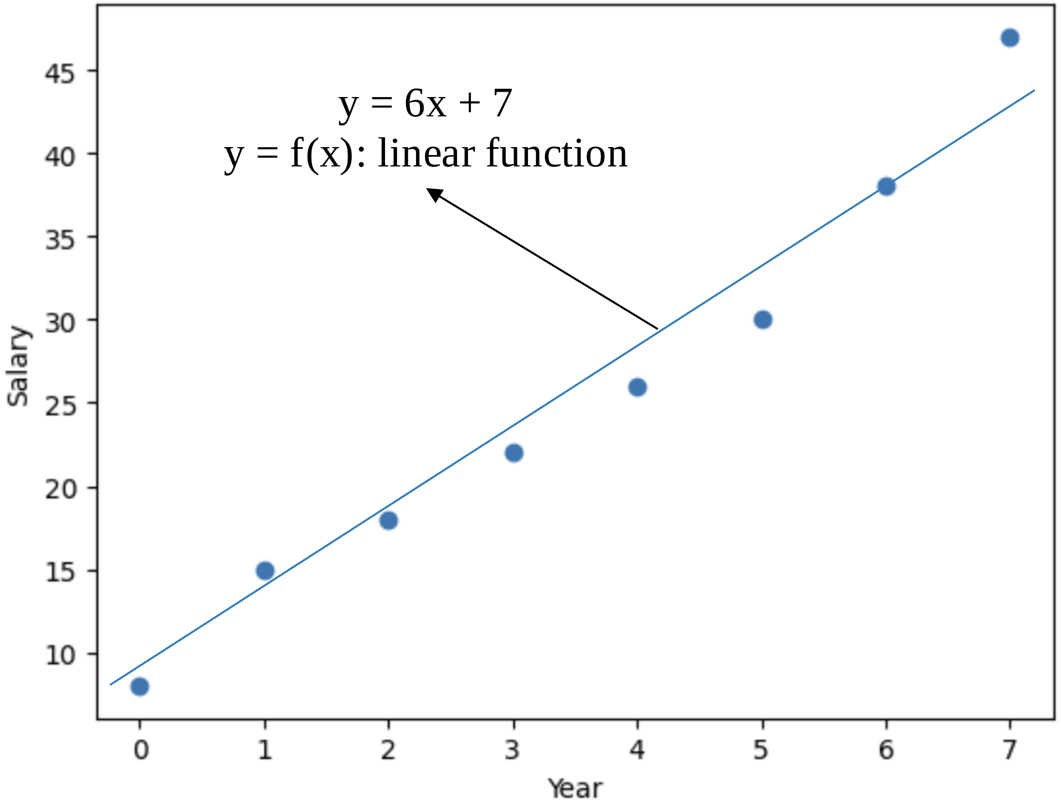
Level	Salary
0	8
1	15
2	18
3	22
4	26
5	30
6	38
7	47

Modeling

$$y = wx + b$$

Find w and b to fit the data

Visualization



AI VIET NAM
@aivietnam.edu.vn

Regression

Linear Regression using Gradient Descent

Data

Level	Salary
0	8
1	15
2	18
3	22
4	26
5	30
6	38
7	47

Inputs / Features

$$X = \begin{bmatrix} 1 & \varphi_1(1) & \dots & \varphi_{d-1}(1) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \varphi_1(N) & \dots & \varphi_{d-1}(N) \end{bmatrix}$$

Target

$$Y = \begin{bmatrix} y(1) \\ \vdots \\ y(N) \end{bmatrix}$$

Weight

$$\theta = \begin{bmatrix} \theta_0 \\ \vdots \\ \theta_{d-1} \end{bmatrix}$$

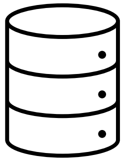
Predict

$$\hat{Y} = \begin{bmatrix} \theta_0 + \theta_1 * \varphi_1(1) + \cdots + \theta_{d-1}\varphi_{d-1}(1) \\ \vdots \\ \theta_0 + \theta_1 * \varphi_1(N) + \cdots + \theta_{d-1}\varphi_{d-1}(N) \end{bmatrix}$$

Regression

Linear Regression using Gradient Descent

Data



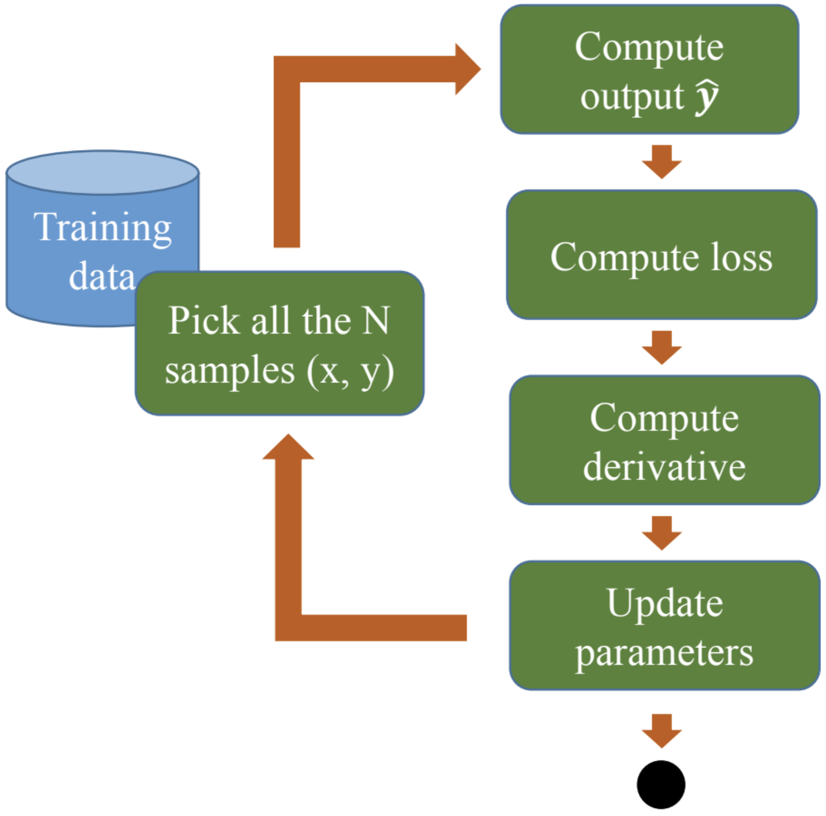
$$X = \begin{bmatrix} 1 & \varphi_1(1) & \dots & \varphi_{d-1}(1) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \varphi_1(N) & \dots & \varphi_{d-1}(N) \end{bmatrix}$$

$$Y = \begin{bmatrix} y(1) \\ \vdots \\ y(N) \end{bmatrix}$$

$$y = wx + b$$

$$\theta = \begin{bmatrix} b \\ w \end{bmatrix} \qquad \theta = \begin{bmatrix} \theta_0 \\ \vdots \\ \theta_{d-1} \end{bmatrix}$$

Learning



1) Pick all the N samples from training data
2) Compute output $\hat{\mathbf{y}}$

$$\hat{\boldsymbol{y}} = \boldsymbol{x\theta}$$

3) Compute loss

$$\boldsymbol{L}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = (\hat{\boldsymbol{y}} - \boldsymbol{y}) \odot (\hat{\boldsymbol{y}} - \boldsymbol{y})$$

4) Compute derivative

$$\boldsymbol{k} = 2(\hat{\boldsymbol{y}} - \boldsymbol{y})$$

$$L'_{\boldsymbol{\theta}} = \boldsymbol{x}^T \boldsymbol{k}$$

5) Update parameters

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \frac{L'_{\boldsymbol{\theta}}}{N}$$

η is learning rate

Regression

Limitation

Modeling

$$y = wx + b$$

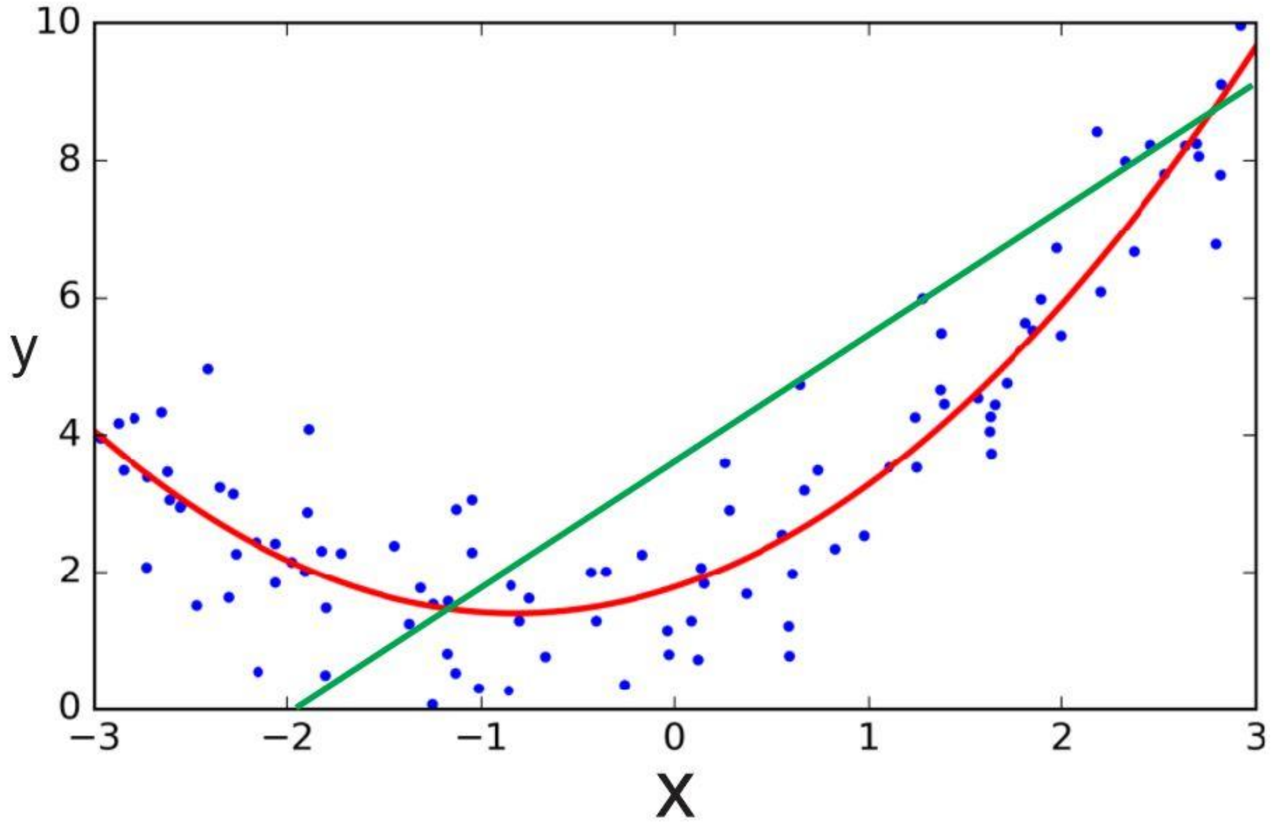
The **main disadvantage** of this technique is that **the model is linear in both the parameters and the features**. This is a very restrictive assumption, quite of the often **data exhibits behaviours that are nonlinear in the features**

Extend this approach to more flexible models…

Regression

Moving Beyond Linearity

Data Visualization



Linear function

$$\hat{y}(i) = \theta_0 + \theta_1 * \varphi(i)$$

Polynomial function

$$\hat{y}(i) = \theta_0 + \theta_1 * \varphi(i) + \theta_2 * \varphi(i)^2$$

Nonlinear regression estimates the ouput based on nonlinear function

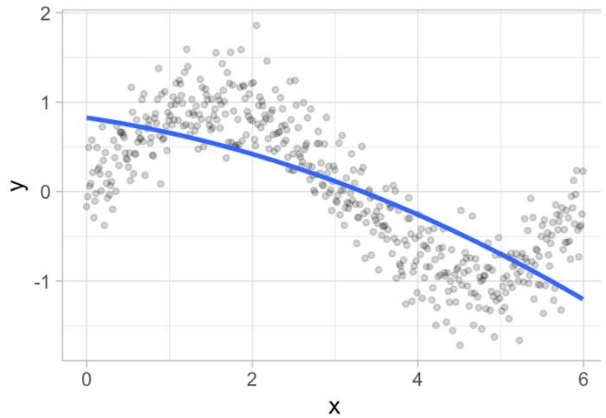
Notice that the prediction is **still linear in the parameters** but **nonlinear in the features**

AI VIET NAM
@aivietnam.edu.vn

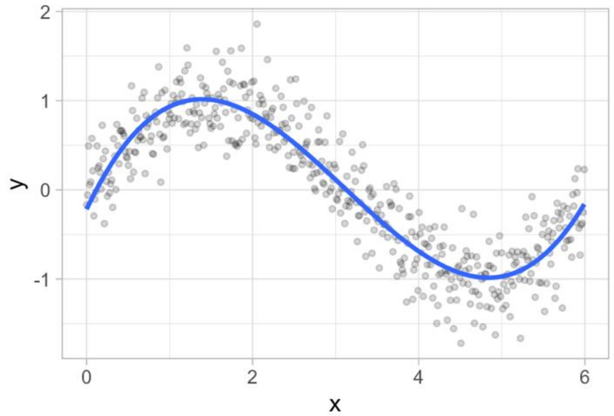
Regression

! Moving Beyond Linearity

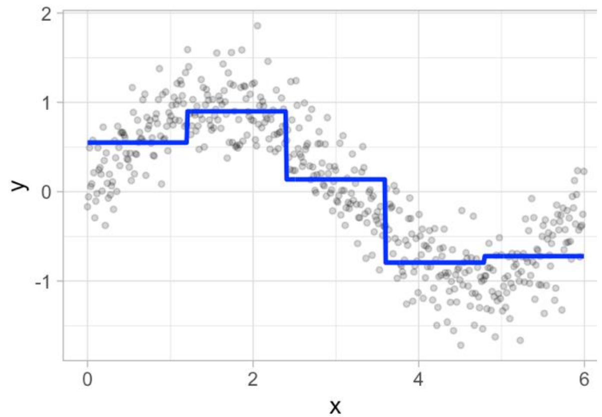
2-degree polynomial function



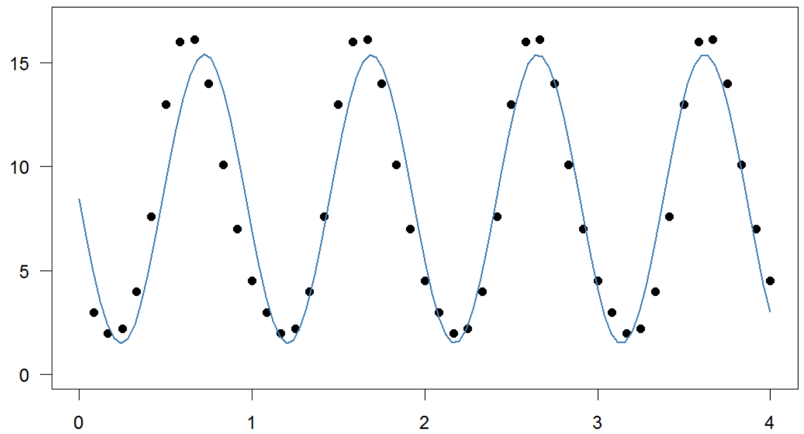
3-degree polynomial function



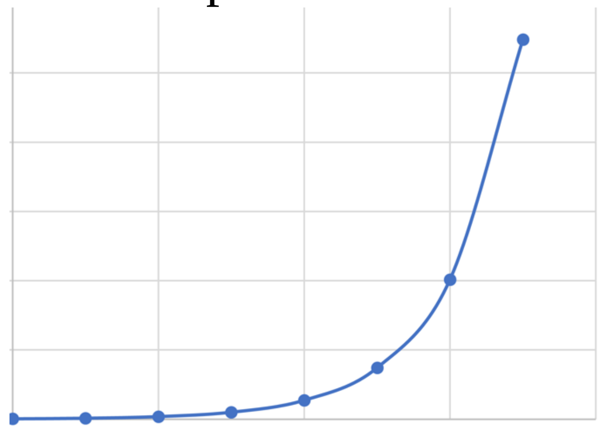
Step function



Sinusoidal



Exponetial function



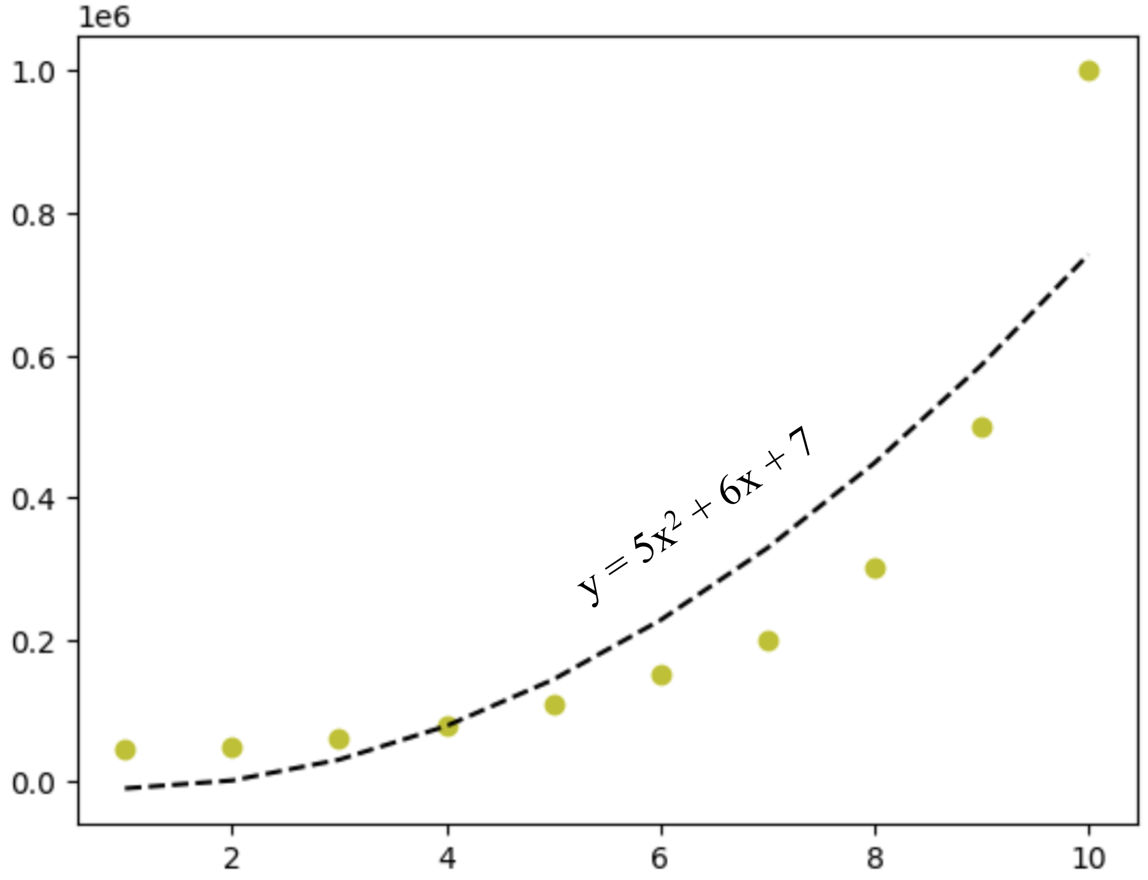
Regression

Polynomial Regression

2-degree polynomial function

$$\hat{y}(i) = \theta_0 * 1 + \theta_1 * \varphi(i) + \theta_2 * \varphi(i)^2$$

Find θ_0 , θ_1 , θ_2 to fit the data



Regression

Polynomial Features

2-degree polynomial function

$$\hat{y}(i) = \theta_0 * 1 + \theta_1 * \varphi(i) + \theta_2 * \varphi(i)^2$$

Create polynomial feature

$$\begin{bmatrix} 1 \\ \varphi(i) \\ \varphi(i)^2 \end{bmatrix} = \boldsymbol{\psi}(\boldsymbol{\varphi}(\boldsymbol{i}))$$

$\psi(\cdot)$is referred to as basis function and it can be seen as a funtion that transforms the input in some way (In this case its powers function)

Regression

Polynomial Features

2-degree polynomial function

$$\hat{y}(i) = \theta_0 * 1 + \theta_1 * \varphi(i) + \theta_2 * \varphi(i)^2$$

Create polynomial feature

Data

Level	Salary
0	45000
1	50000
2	60000
3	80000
4	110000
5	160000

$\boldsymbol{\psi(\varphi(i))}$

Input	1	$\varphi(i)$	$\varphi(i)^2$
0	1	0	0
1	1	1	1
2	1	2	4
3	1	3	9
4	1	4	16
5	1	5	25

Regression

Polynomial Features

2-degree polynomial function

$$\hat{y}(i) = \theta_0 * 1 + \theta_1 * \varphi(i) + \theta_2 * \varphi(i)^2$$

Features

1	$\varphi(i)$	$\varphi(i)^2$
1	0	0
1	1	1
1	2	4
1	3	9
1	4	16
1	5	25

Target

45000
50000
60000
80000
110000
160000

Regression

Polynomial Features

3-degree polynomial function

$$\hat{y}(i) = \theta_0 * 1 + \theta_1 * \varphi(i) + \theta_2 * \varphi(i)^2 + \theta_3 * \varphi(i)^3$$

Features

1	$\varphi(i)$	$\varphi(i)^2$	$\varphi(i)^3$
1	0	0	0
1	1	1	1
1	2	4	8
1	3	9	27
1	4	16	64
1	5	25	125

Target

45000
50000
60000
80000
110000
160000

Regression

Polynomial Features

Input

0
1
2
3
4
5

Features

1	$\varphi(\text{i})$	$\varphi(\text{i})^2$
1	0	0
1	1	1
1	2	4
1	3	9
1	4	16
1	5	25

Algorithm

```
def create_polynomial_features(X, degree=2):
    """Creates the polynomial features
    Args:
        X: A torch tensor for the data.
        degree: A intege for the degree of
        the generated polynomial function.
    """
    X_new = X
    for d in range(2, degree+1):
        X_new = np.c_[X_new, np.power(X, d)]
    return X_new
```

Regression

Polynomial Features

Input

0
1
2
3
4
5

Features

1	$\varphi(i)$	$\varphi(i)^2$
1	0	0
1	1	1
1	2	4
1	3	9
1	4	16
1	5	25

```
from sklearn.preprocessing import PolynomialFeatures
```

```
poly_features = PolynomialFeatures(degree=2)
```

```
X.to_frame()
```

```
(10, 1)
```

```
X_poly = poly_features.fit_transform(X.to_frame())
X_poly
```

```
array([[  1.,   1.,   1.],
       [  1.,   2.,   4.],
       [  1.,   3.,   9.],
       [  1.,   4.,  16.],
       [  1.,   5.,  25.],
       [  1.,   6.,  36.],
       [  1.,   7.,  49.],
       [  1.,   8.,  64.],
       [  1.,   9.,  81.],
       [  1.,  10., 100.]])
```

Regression

Vectorization

Data

Level	Salary
0	45000
1	50000
2	60000
3	80000
4	110000
5	160000

Inputs / Features with b-degree

$$X = \begin{bmatrix} 1 & \varphi_1(1) & \dots & \varphi_1(1)^{b} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \varphi_1(N) & \dots & \varphi_1(N)^{b} \end{bmatrix}$$

Target

$$Y = \begin{bmatrix} y(1) \\ \vdots \\ y(N) \end{bmatrix}$$

Weight

$$\theta = \begin{bmatrix} \theta_0 \\ \vdots \\ \theta_b \end{bmatrix}$$

Predict

$$\hat{Y} = \begin{bmatrix} \theta_0 + \theta_1 * \varphi_1(1) + \cdots + \theta_b \varphi_1(1)^{b} \\ \vdots \\ \theta_0 + \theta_1 * \varphi_1(N) + \cdots + \theta_b \varphi_1(N)^{b} \end{bmatrix}$$

Regression

Model

Nonlinear Regression Model

$$X = \begin{bmatrix} 1 & \varphi_1(1) & \cdots & \varphi_1(1)^b \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \varphi_1(N) & \cdots & \varphi_1(N)^b \end{bmatrix}$$

$$Y = \begin{bmatrix} y(1) \\ \vdots \\ y(N) \end{bmatrix} \qquad \theta = \begin{bmatrix} \theta_0 \\ \vdots \\ \theta_b \end{bmatrix}$$

$$\hat{Y} = \begin{bmatrix} \theta_0 + \theta_1 * \varphi_1(1) + \cdots + \theta_b \varphi_1(1)^b \\ \vdots \\ \theta_0 + \theta_1 * \varphi_1(N) + \cdots + \theta_b \varphi_1(N)^b \end{bmatrix}$$

Both models are linear in the parameters

$$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} (\hat{y} - y)^2$$

Using Gradient Decent

Linear Regression Model

$$X = \begin{bmatrix} 1 & \varphi_1(1) \\ \vdots & \vdots \\ 1 & \varphi_1(N) \end{bmatrix}$$

$$Y = \begin{bmatrix} y(1) \\ \vdots \\ y(N) \end{bmatrix} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}$$

$$\hat{Y} = \begin{bmatrix} \theta_0 + \theta_1 * \varphi_1(1) \\ \vdots \\ \theta_0 + \theta_1 * \varphi_1(N) \end{bmatrix}$$

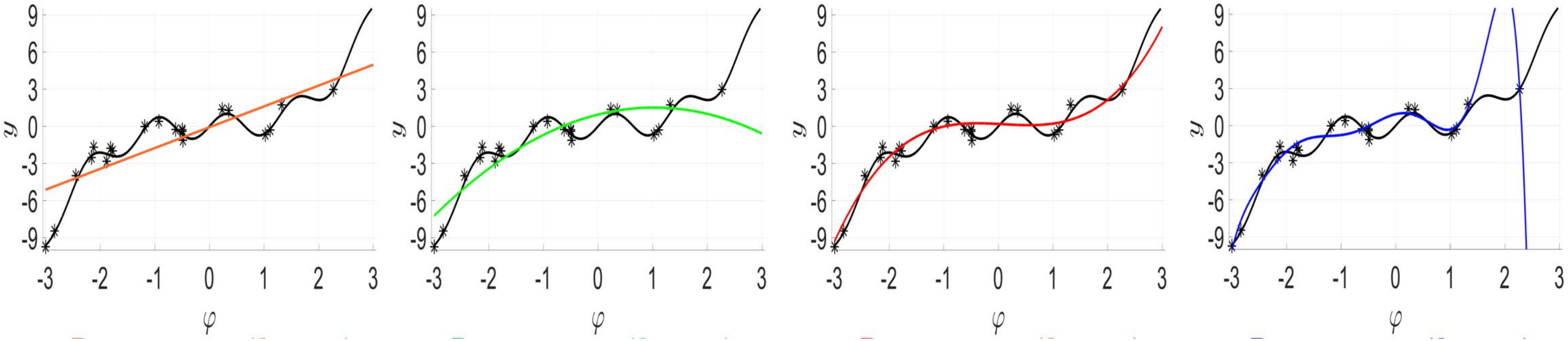
Regression

Degree Choice

b-degree polynomial function

$$\hat{y}(i) = \theta_0 * 1 + \theta_1 * \varphi(i) + \theta_2 * \varphi(i)^2 + \cdots + \theta_b * \varphi(i)^b$$

The choice of the degree of the polynomial if critical and depends on the dataset at hand



1-degree | **2-degree** | **3-degree** | **9-degree**

Too simple/not flexible enough | Just right | Overfitting

Regression

Degree Choice

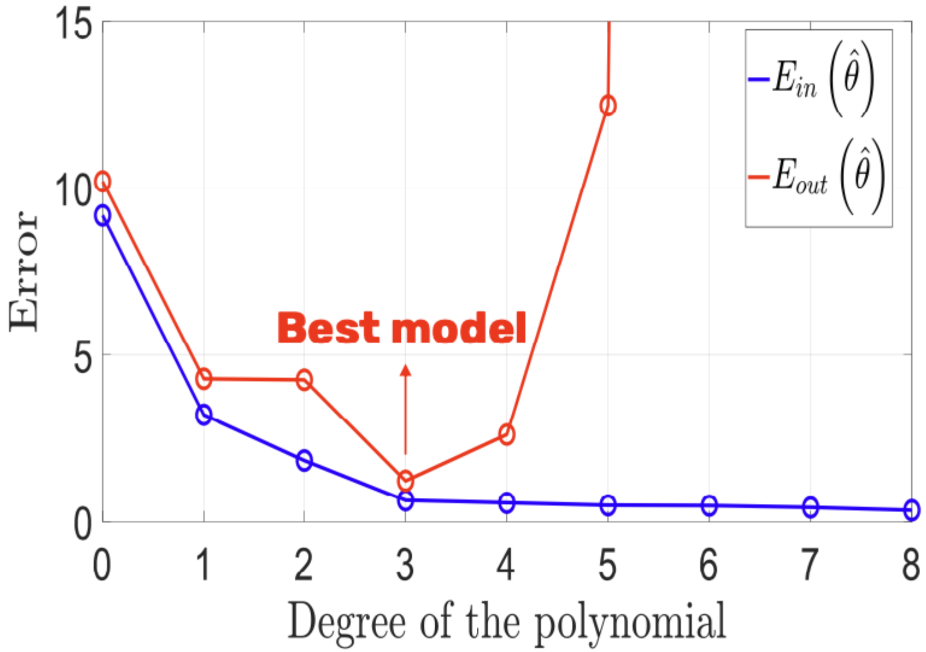
b-degree polynomial function

$$\hat{y}(i) = \theta_0 * 1 + \theta_1 * \varphi(i) + \theta_2 * \varphi(i)^2 + \cdots + \theta_b * \varphi(i)^b$$

The choice of the degree of the polynomial if critical and depends on the dataset at hand

Good method for choice of the degree:
K-fold cross-validation

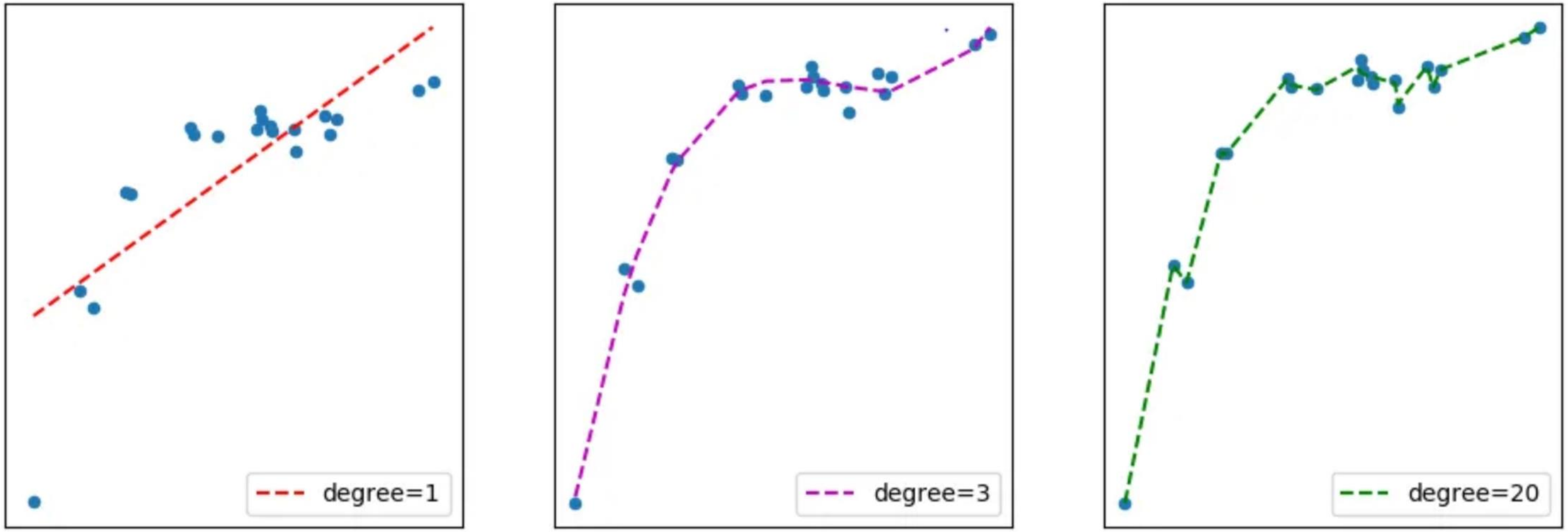
Choose the degree which has the lowest out-of-sample error



Regression

Disadvantages

Increasing the degree of the polynomial always results in a model that is more sensitive to stochastic noise (even if that degree is the best one obtained from validation), especially at the boundaries (where we often have less data).



QUIZ TIME

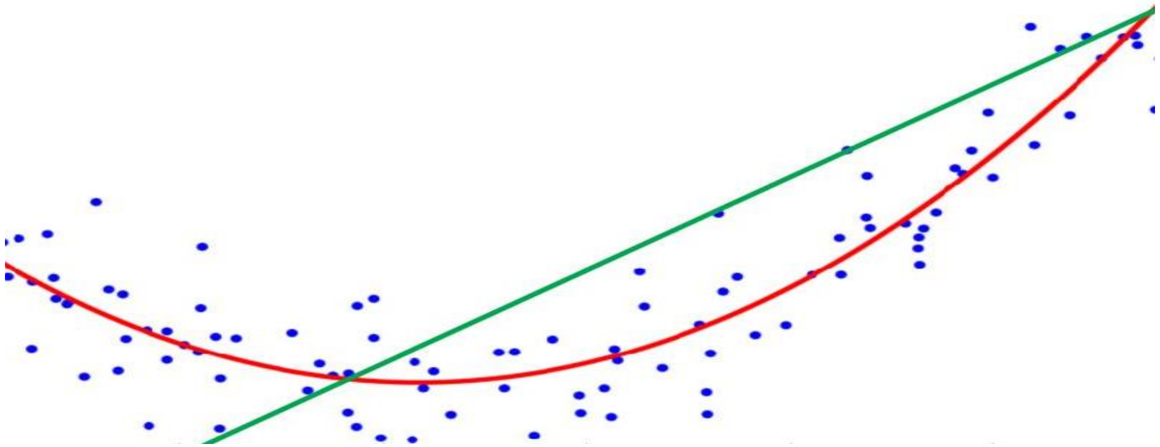
Outline

SECTION 1

Regression

SECTION 2

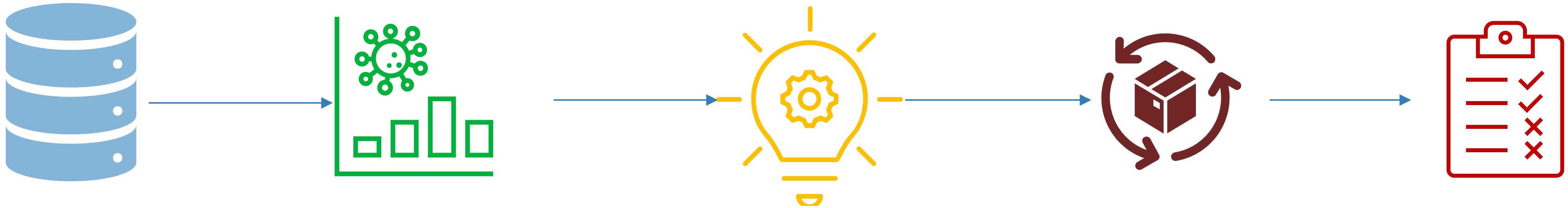
Sales Prediction



	TV	Radio	Social Media	Influencer	Sales
0	16.0	6.566231	2.907983	Mega	54.732757
1	13.0	9.237765	2.409567	Mega	46.677897
2	41.0	15.886446	2.913410	Mega	150.177829
3	83.0	30.020028	6.922304	Mega	298.246340
4	15.0	8.437408	1.405998	Micro	56.594181

Sales Prediction

! Dataset



Dataset

Preprocessing

Exploratory Data Analysis

Feature Scaling

Modeling

Evaluation

Sales Prediction

Dataset

```
import pandas as pd

df = pd.read_csv('./SalesPrediction.csv')
df
```

	TV	Radio	Social Media	Influencer	Sales
0	16.0	6.566231	2.907983	Mega	54.732757
1	13.0	9.237765	2.409567	Mega	46.677897
2	41.0	15.886446	2.913410	Mega	150.177829
3	83.0	30.020028	6.922304	Mega	298.246340
4	15.0	8.437408	1.405998	Micro	56.594181

Sales Prediction

Dataset

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4572 entries, 0 to 4571
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   TV            4562 non-null   float64
 1   Radio         4568 non-null   float64
 2   Social Media  4566 non-null   float64
 3   Influencer    4572 non-null   object
 4   Sales         4566 non-null   float64
dtypes: float64(4), object(1)
memory usage: 178.7+ KB
```

```
1 df.describe()
```

	TV	Radio	Social Media	Sales
count	4562.000000	4568.000000	4566.000000	4566.000000
mean	54.066857	18.160356	3.323956	192.466602
std	26.125054	9.676958	2.212670	93.133092
min	10.000000	0.000684	0.000031	31.199409
25%	32.000000	10.525957	1.527849	112.322882
50%	53.000000	17.859513	3.055565	189.231172
75%	77.000000	25.649730	4.807558	272.507922
max	100.000000	48.871161	13.981662	364.079751

Sales Prediction

Preprocessing – One hot encoding

```
df = pd.get_dummies(df)
df
```

Radio	Social Media	Sales	Influencer_Macro	Influencer_Mega	Influencer_Micro	Influencer_Nano
6.566231	2.907983	54.732757	False	True	False	False
9.237765	2.409567	46.677897	False	True	False	False
15.886446	2.913410	150.177829	False	True	False	False
30.020028	6.922304	298.246340	False	True	False	False
8.437408	1.405998	56.594181	False	False	True	False

Sales Prediction

Preprocessing – Handling Missing Values

```python
df.isnull().sum()
```

	0
TV	10
Radio	4
Social Media	6
Sales	6
Influencer_Macro	0
Influencer_Mega	0
Influencer_Micro	0
Influencer_Nano	0

dtype: int64

```python
df = df.fillna(0)
df.isnull().sum()
```

	0
TV	0
Radio	0
Social Media	0
Sales	0
Influencer_Macro	0
Influencer_Mega	0
Influencer_Micro	0
Influencer_Nano	0

dtype: int64

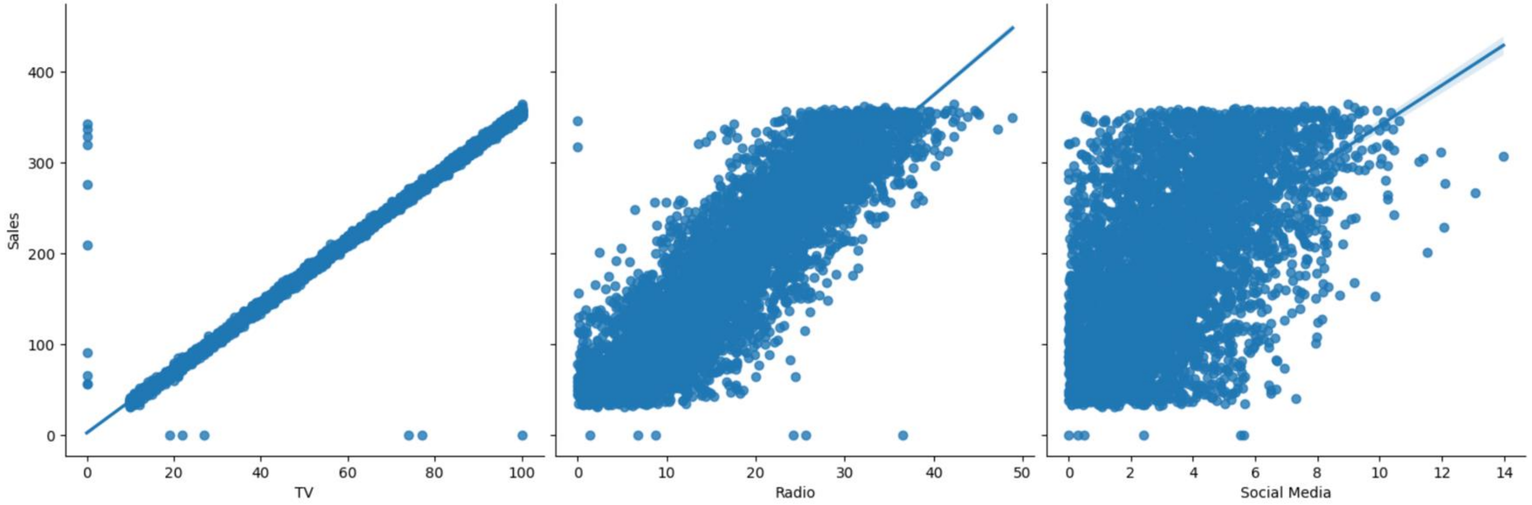
```python
df = df.fillna(df.mean())
df.isnull().sum()
```

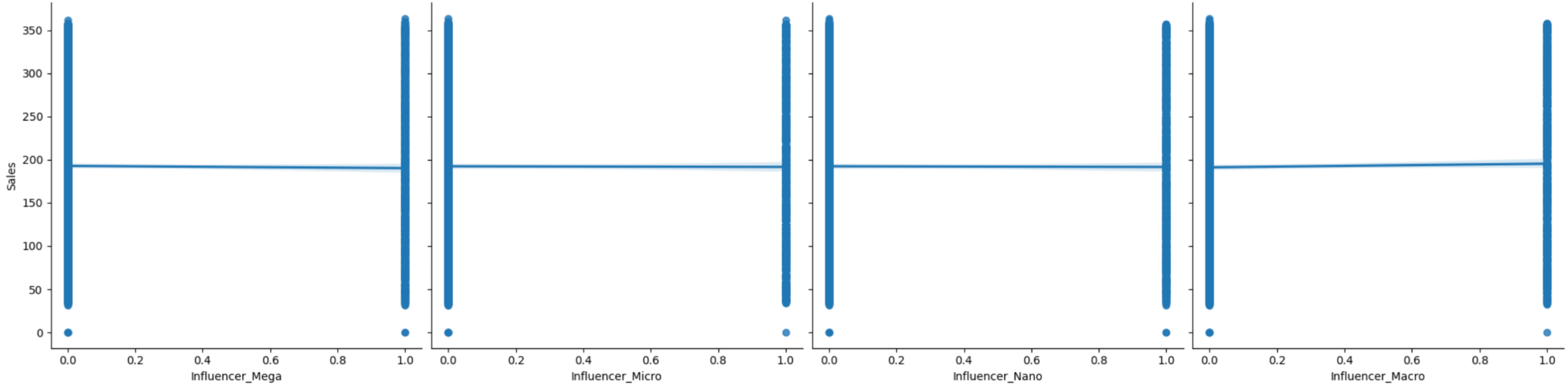
	0
TV	0
Radio	0
Social Media	0
Sales	0
Influencer_Macro	0
Influencer_Mega	0
Influencer_Micro	0
Influencer_Nano	0

dtype: int64

Sales Prediction

EDA



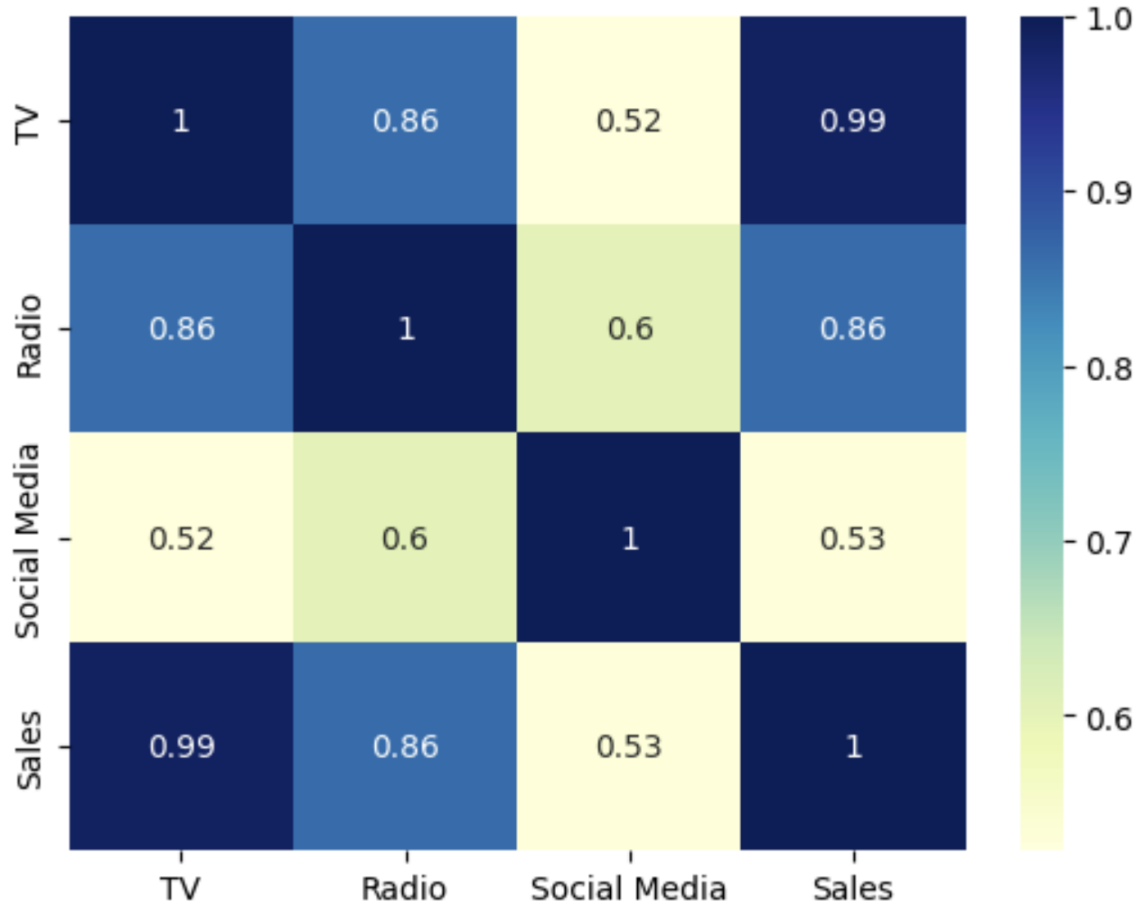


Sales Prediction

EDA

```
df[['TV', 'Radio', 'Social Media', 'Sales']].corr()
```

	TV	Radio	Social Media	Sales
TV	1.000000	0.860518	0.522565	0.988570
Radio	0.860518	1.000000	0.604450	0.863790
Social Media	0.522565	0.604450	1.000000	0.526777
Sales	0.988570	0.863790	0.526777	1.000000



Sales Prediction

Train Test Split

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.33,
    random_state=0
)
```

```
X_train.shape, X_test.shape
```

((3063, 7), (1509, 7))

```
y_train.shape, y_test.shape
```

((3063, 1), (1509, 1))

Sales Prediction

Feature Scaling

MaxAbsScaler

$$x_{new} = \frac{x}{x_{max}}$$

MinMaxScaler

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

StandardScaler

$$x_{new} = \frac{x - \mu}{\sigma}$$

```python
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_processed = scaler.fit_transform(X_train)
```

```python
scaler.mean_
```

```
array([53.9970617 , 18.22209011,  3.33487105,  0.24779628,  0.25138753,
        0.25008162,  0.25073457])
```

```python
scaler.scale_
```

```
array([26.24285095,  9.6336957 ,  2.21929717,  0.43173288,  0.43381083,
        0.43305981,  0.43343598])
```

```python
X_test_processed = scaler.transform(X_test)
```

```python
X_train_processed.shape
```

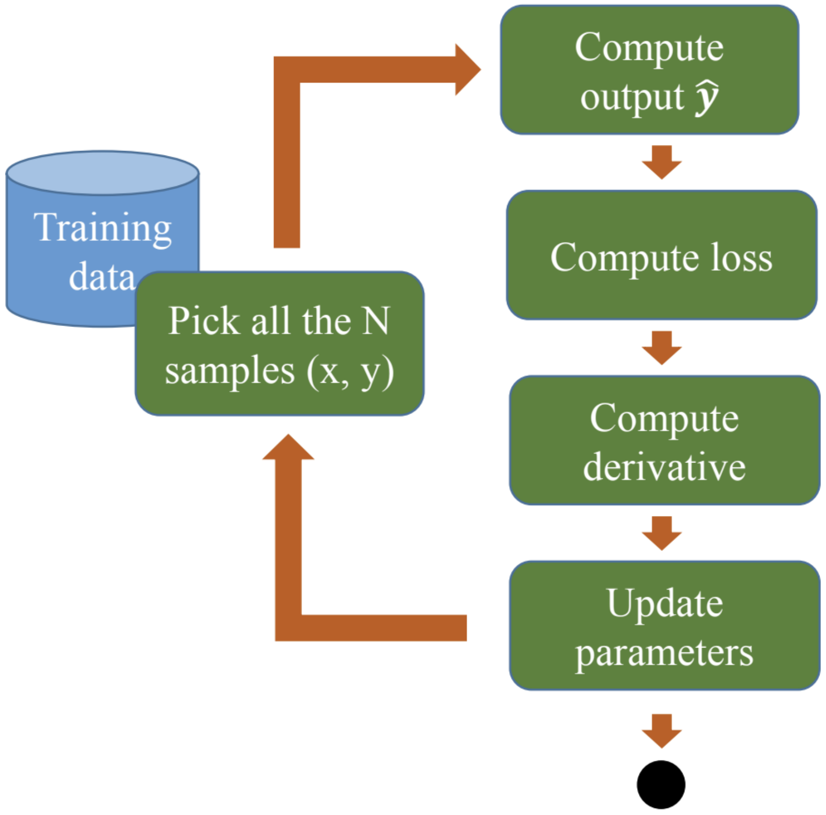
```
(3063, 7)
```

```python
X_test_processed.shape
```

```
(1509, 7)
```

Sales Prediction

Modeling



1) Pick all the N samples from training data
2) Compute output $\hat{\boldsymbol{y}}$

$$\hat{\boldsymbol{y}} = \boldsymbol{x}\boldsymbol{\theta}$$

3) Compute loss

$$\boldsymbol{L}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = (\hat{\boldsymbol{y}} - \boldsymbol{y}) \odot (\hat{\boldsymbol{y}} - \boldsymbol{y})$$

4) Compute derivative

$$\boldsymbol{k} = 2(\hat{\boldsymbol{y}} - \boldsymbol{y})$$

$$L'_{\boldsymbol{\theta}} = \boldsymbol{x}^T \boldsymbol{k}$$

5) Update parameters

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \frac{L'_{\boldsymbol{\theta}}}{N}$$

η is learning rate

Sales Prediction

Modeling

```python
from sklearn.linear_model import LinearRegression

linear_model = LinearRegression()
linear_model.fit(X_train_processed, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```python
preds = linear_model.predict(X_test_processed)
r2_score(y_test, preds)
```

```
0.9820910569999272
```

Sales Prediction

Polynominal Regression

```
from sklearn.preprocessing import PolynomialFeatures

poly_features = PolynomialFeatures(degree=2)
X_train_poly = poly_features.fit_transform(X_train_processed)
```

```
X_train_poly
```

```
array([[ 1.        ,  0.34306251, -0.39269809, ...,  2.99869452,
        -1.00174084,  0.33464052],
       [ 1.        , -0.19041611, -0.28821416, ...,  0.33347845,
         0.33405898,  0.33464052],
       [ 1.        , -0.41904981, -1.07312224, ...,  0.33347845,
        -0.99826219,  2.98828125],
       ...,
       [ 1.        , -1.6003239 , -1.72760008, ...,  0.33347845,
        -0.99826219,  2.98828125],
       [ 1.        , -0.57147227, -0.9126861 , ...,  2.99869452,
        -1.00174084,  0.33464052],
       [ 1.        , -1.25737336, -1.45632493, ...,  2.99869452,
        -1.00174084,  0.33464052]])
```

```
X_test_poly = poly_features.transform(X_test_processed)
```

Sales Prediction

Polynominal Regression

```
poly_model = LinearRegression()
poly_model.fit(X_train_poly, y_train)
```

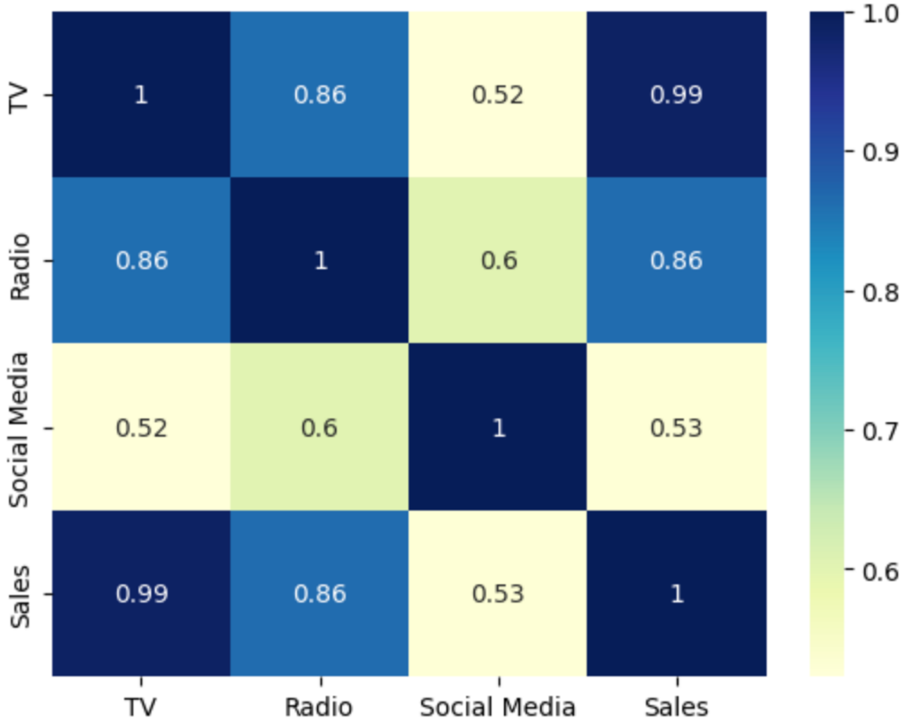
```
▾ LinearRegression
LinearRegression()
```

```
preds = poly_model.predict(X_test_poly)
r2_score(y_test, preds)
```

```
0.9785743009106321
```

Sales Prediction

Custom Polynominal Regression




```
X_train_processed[:, 2:3]
```

```
array([[-0.17117575],
       [-1.47454833],
       [-0.55726535],
       ...,
       [ 0.58703816],
       [-1.22457248],
       [-1.04684805]])
```

```
x_train_poly = create_polynomial_features(X_train_processed[:, 2:3], degree=2)
x_train_poly
```

```
array([[-0.17117575,  0.02930114],
       [-1.47454833,  2.17429276],
       [-0.55726535,  0.31054467],
       ...,
       [ 0.58703816,  0.3446138 ],
       [-1.22457248,  1.49957777],
       [-1.04684805,  1.09589085]])
```

Sales Prediction

Custom Polynominal Regression

```
x_test_poly = create_polynomial_features(X_test_processed[:, 2:3], degree=2)
X_test_poly = np.hstack((X_test_processed, x_test_poly[:, 1:]))
X_test_poly
```

```
array([[-0.34283858, -0.11361891, -0.84351677, ...,  1.73167391,
        -0.57848122,  0.71152054],
       [ 0.76222428,  1.17276695, -0.45136527, ..., -0.57747593,
         1.72866459,  0.20373061],
       [ 1.14328044,  1.04152694,  1.06570814, ..., -0.57747593,
        -0.57848122,  1.13573384],
       ...,
       [ 0.95275236,  1.11773825,  1.06866838, ..., -0.57747593,
        -0.57848122,  1.14205211],
       [-0.41904981, -0.32445517,  1.20063234, ..., -0.57747593,
        -0.57848122,  1.44151802],
       [-0.91442282, -1.25598448, -0.32806086, ..., -0.57747593,
        -0.57848122,  0.10762393]])
```

```
X_train_poly.shape, X_test_poly.shape
```

```
((3063, 8), (1509, 8))
```

```
poly_model = LinearRegression()
poly_model.fit(X_train_poly, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
preds = poly_model.predict(X_test_poly)
r2_score(y_test, preds)
```

```
0.9820873203866817
```

AI VIET NAM
@aivietnam.edu.vn

Sales Prediction

Results

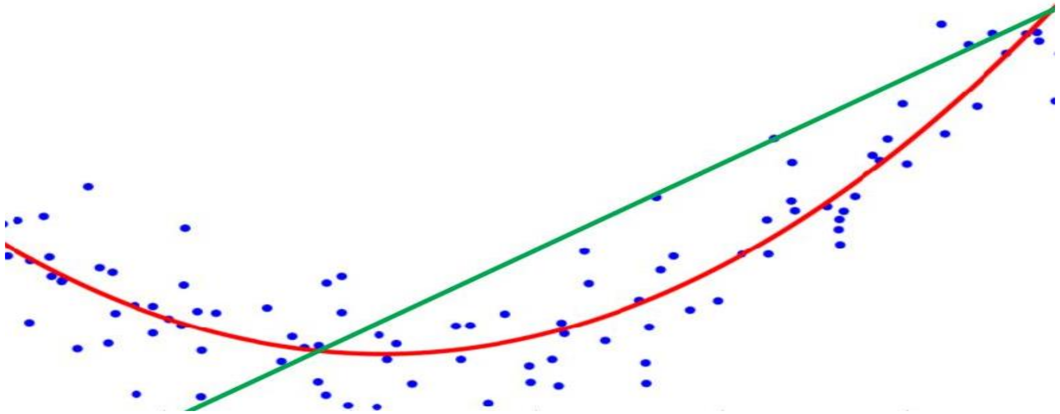
Model	R2
Linear Regression	0.9821
Polynominal Regression (2)	0.9786
Custom Polynominal Regression	0.9821

Summary

Regression

- ❖ Regression Task
- ❖ Linear Regression
- ❖ Non-linear Regression
- ❖ Using Sklearn library

	TV	Radio	Social Media	Influencer	Sales
0	16.0	6.566231	2.907983	Mega	54.732757
1	13.0	9.237765	2.409567	Mega	46.677897
2	41.0	15.886446	2.913410	Mega	150.177829
3	83.0	30.020028	6.922304	Mega	298.246340
4	15.0	8.437408	1.405998	Micro	56.594181



Sales Prediction

- ❖ Exploratory Data Analysis (EDA)
- ❖ Feature Scaling
- ❖ Modeling
- ❖ Evaluation
- ❖ Custom Polynomial Features

Thanks!

Any questions?