polynomialregression

August 25, 2023

1 Polynomial Regression

- Polynomial Regression is an extended version of linear regression where polynomial terms are introduced. When the regression line/good fit line isn't linear anymore and doesn't fit the data well(or when the data is non-linear), we opt for polynomial regression.
- Polynomial regression is also called as polynomial linear regression as still the target variable is in linear relationship with the coefficients.

Polynomial slope intercept formulae, $\mathrm{Y{=}B0{+}B1X1{+}(B2X2)^{2+(B3X3)}3{+}(B4X4)^{4+......+(BNXN)}N}$

[ ]:

```
# Importing necessary libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[ ]:

```
# Onboarding data

from google.colab import files
rawdata=files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving Ice_cream selling data.csv to Ice_cream selling data (1).csv
```

[ ]:

```
# DataFrame

df=pd.read_csv('Ice_cream selling data (1).csv')
df
```

[ ]:

```
    Temperature (°C)  Ice Cream Sales (units)
0          -4.662263                41.842986
1          -4.316559                34.661120
2          -4.213985                39.383001
3          -3.949661                37.539845
4          -3.578554                32.284531
5          -3.455712                30.001138
6          -3.108440                22.635401
```

```
7          -3.081303                25.365022
8          -2.672461                19.226970
9          -2.652287                20.279679
10         -2.651498                13.275828
11         -2.288264                18.123991
12         -2.111870                11.218294
13         -1.818938                10.012868
14         -1.660348                12.615181
15         -1.326379                10.957731
16         -1.173123                 6.689123
17         -0.773330                 9.392969
18         -0.673753                 5.210163
19         -0.149635                 4.673643
20         -0.036156                 0.328626
21         -0.033895                 0.897603
22          0.008608                 3.165600
23          0.149245                 1.931416
24          0.688781                 2.576782
25          0.693599                 4.625689
26          0.874905                 0.789974
27          1.024181                 2.313806
28          1.240712                 1.292361
29          1.359813                 0.953115
30          1.740000                 3.782570
31          1.850552                 4.857988
32          1.999310                 8.943823
33          2.075101                 8.170735
34          2.318591                 7.412094
35          2.471946                10.336631
36          2.784836                15.996620
37          2.831760                12.568237
38          2.959932                21.342916
39          3.020874                20.114413
40          3.211366                22.839406
41          3.270044                16.983279
42          3.316073                25.142082
43          3.335932                26.104740
44          3.610778                28.912188
45          3.704057                17.843957
46          4.130868                34.530743
47          4.133534                27.698383
48          4.899032                41.514822
```

[ ]:

```
# Shallow copy

df_copy=df.copy()
```

2 Exploratory Data Analysis

```
[ ]: df.head()
```

```
[ ]:    Temperature (°C)  Ice Cream Sales (units)
    0         -4.662263                41.842986
    1         -4.316559                34.661120
    2         -4.213985                39.383001
    3         -3.949661                37.539845
    4         -3.578554                32.284531
```

```
[ ]: df.shape
```

```
[ ]: (49, 2)
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 49 entries, 0 to 48
Data columns (total 2 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Temperature (°C)         49 non-null     float64
 1   Ice Cream Sales (units)  49 non-null     float64
dtypes: float64(2)
memory usage: 912.0 bytes
```

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

```
[ ]:        Temperature (°C)  Ice Cream Sales (units)
    count         49.000000                49.000000
    mean           0.271755                15.905308
    std            2.697672                12.264682
    min           -4.662263                 0.328626
    25%           -2.111870                 4.857988
    50%            0.688781                12.615181
    75%            2.784836                25.142082
    max            4.899032                41.842986
```

```
[ ]: df.isna().sum()
```

```
[ ]: Temperature (°C)           0
     Ice Cream Sales (units)    0
     dtype: int64
```

```
[ ]: df.duplicated().sum()
[ ]: 0
```

```
[ ]: df.columns
```

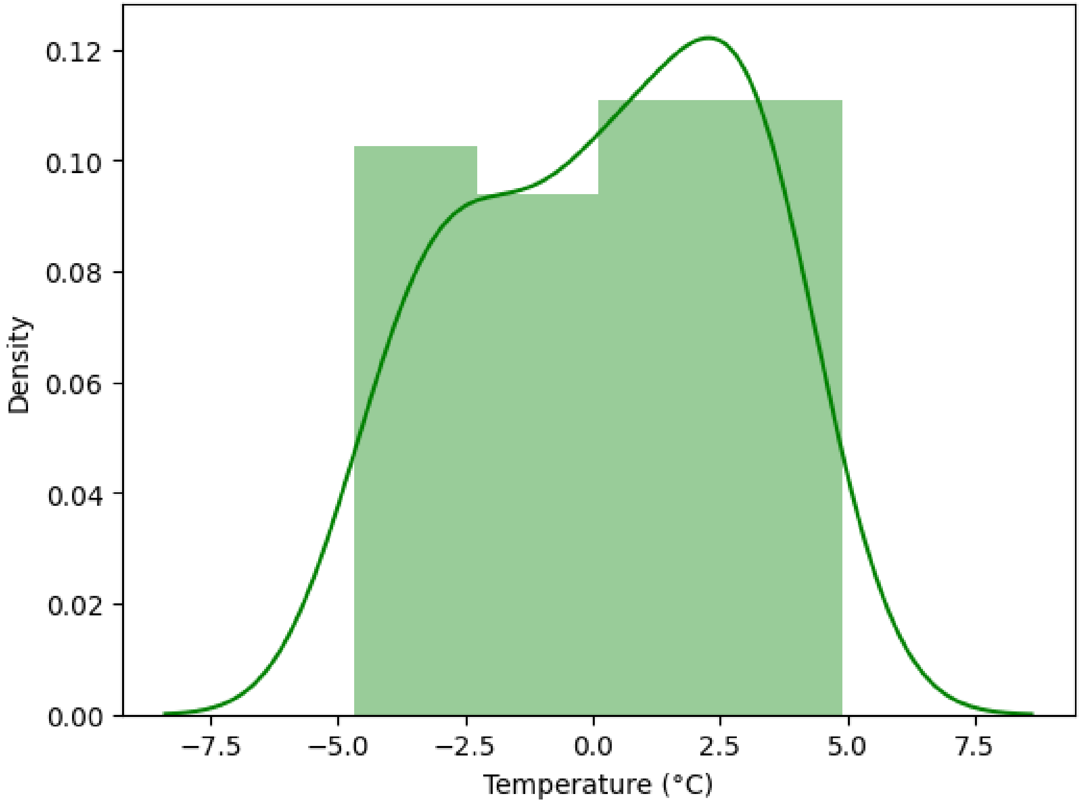
```
[ ]: Index(['Temperature (°C)', 'Ice Cream Sales (units)'], dtype='object')
```

3 Univariate Analysis

```
[ ]: sns.distplot(df['Temperature (°C)'],color='green')
```

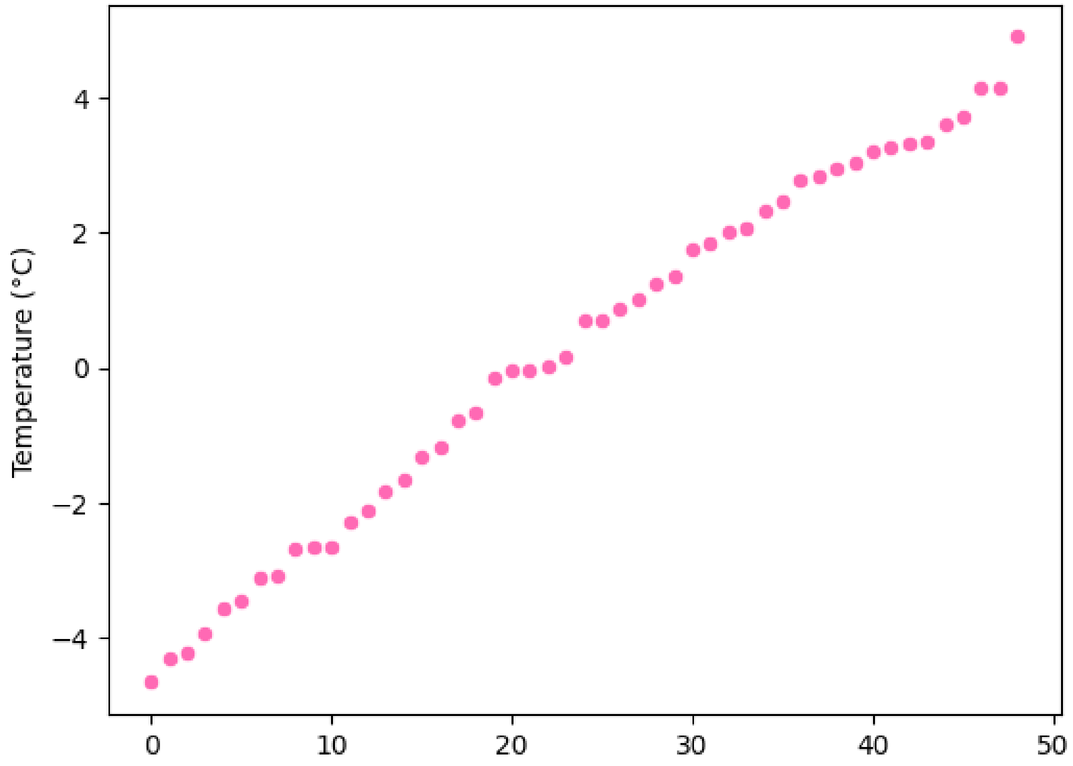
```
<ipython-input-64-3bd4a6f783fa>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['Temperature (°C)'],color='green')
```

```
[ ]: <Axes: xlabel='Temperature (°C)', ylabel='Density'>
```



[ ]:
```
sns.scatterplot(df['Temperature (°C)'],color='hotpink')
```

[ ]:
```
<Axes: ylabel='Temperature (°C)'>
```



4 Splitting independent and dependent variable

[ ]:
```
X=df.iloc[:,[0]]
Y=df.iloc[:,[-1]]
```

5 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=0.2,random_state=3)
```

6 Model Building

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

     lr=LinearRegression()
     lr.fit(x_train,y_train)
```

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

```
[ ]: y_predict=lr.predict(x_test)
```

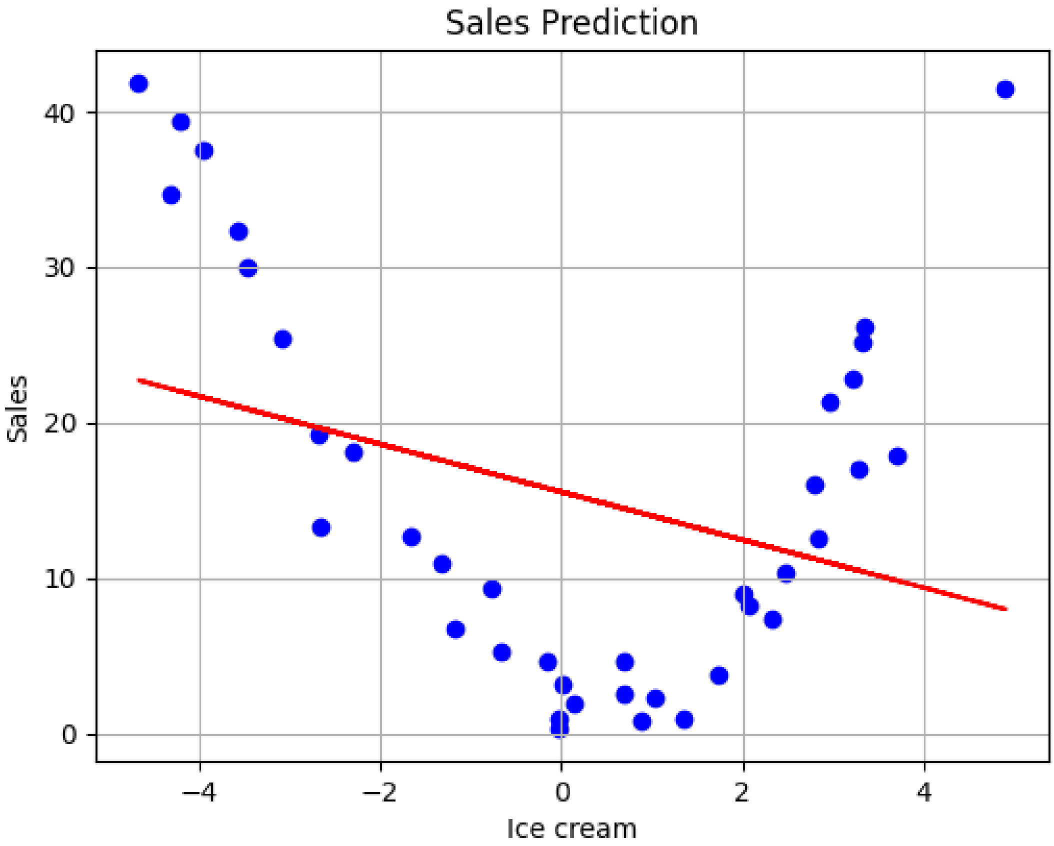
```
[ ]: y_predict
```

```
[ ]: array([[18.77660674],
            [10.87747649],
            [19.60829152],
            [ 9.16923147],
            [12.67856553],
            [13.61709043],
            [18.32579355],
            [ 9.16512885],
            [ 9.96963269],
            [20.31029702]])
```

```
[ ]: y_test
```

```
[ ]:     Ice Cream Sales (units)
     12                11.218294
     39                20.114413
     9                 20.279679
     46                34.530743
     31                 4.857988
     28                 1.292361
     13                10.012868
     47                27.698383
     44                28.912188
     6                 22.635401
```

```
[ ]: plt.scatter(x_train,y_train,color='blue')
     plt.plot(x_train,lr.predict(x_train),color='red')
     plt.title('Sales Prediction')
     plt.xlabel('Ice cream')
     plt.ylabel('Sales')
     plt.grid()
     plt.show()
```



[ ]:

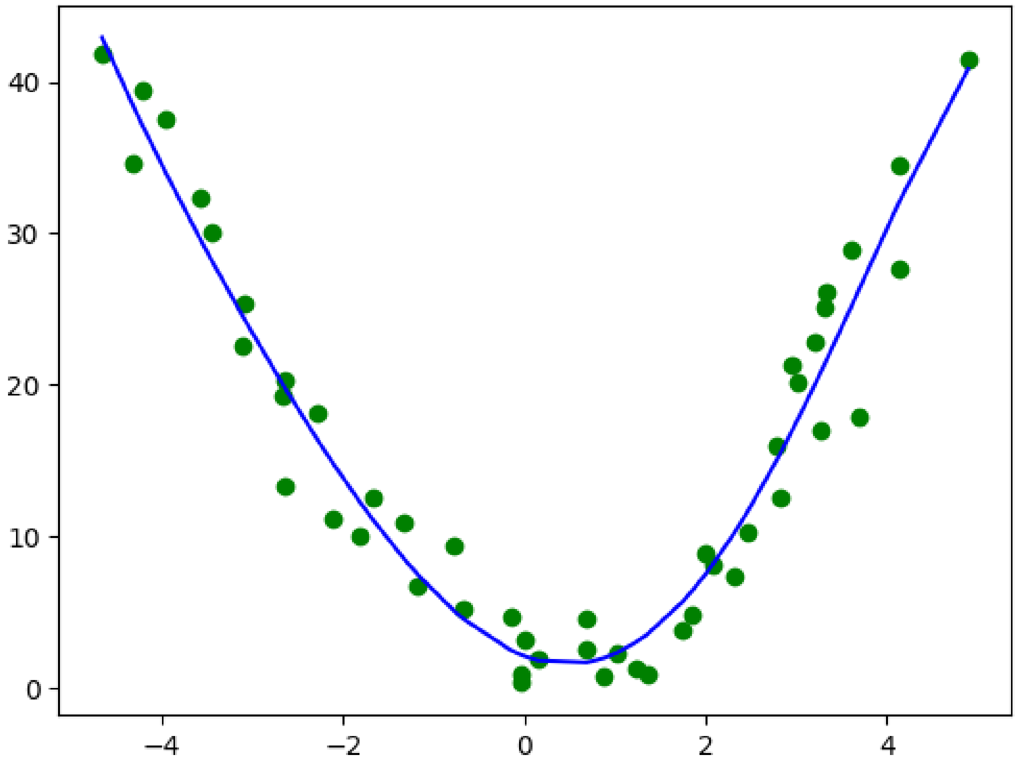
```
# Fitting Polynomial Regression

from sklearn.preprocessing import PolynomialFeatures

pf=PolynomialFeatures(degree=5)
X_poly=pf.fit_transform(X)
lrg=LinearRegression()
lrg.fit(X_poly,Y)
```

[ ]:

```
LinearRegression()
```

[ ]:

```
plt.scatter(X,Y,color='green')
plt.plot(X,lrg.predict(pf.fit_transform(X)),color='blue')
plt.show()
```


40
30
20
10
0
−4
−2
0
2
4