VELLORE INSTITUTE OF TECHNOLOGY
CSE4020 Machine Learning
Lab Assessment - 3

17BCE0581

SATYAM SINGH CHAUHAN

Difference between Linear and Polynomial Regression

Importing the Required Libraries

- **matplotlib.pyplot**
- **numpy**

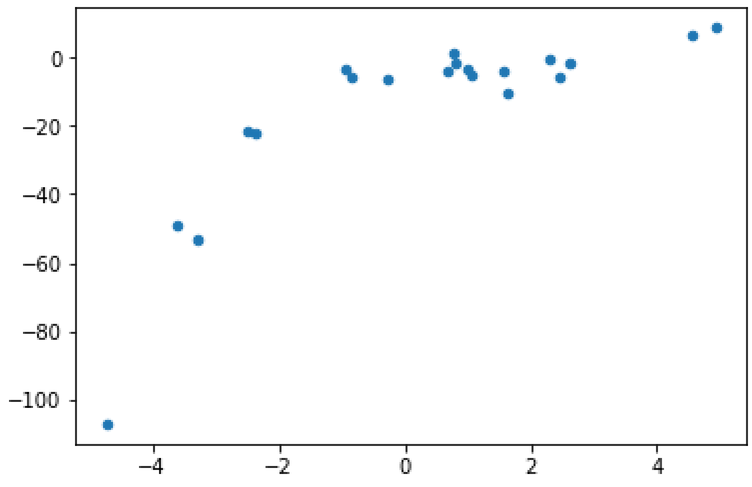
In [26]:

```
import matplotlib.pyplot as plt
import numpy as np
```

Creating X & Y Variables of DataSet and then Plotting it.

In [27]:

```
np.random.seed(0)
x=2-3*np.random.normal(0,1,20)
y=x-2*(x**2)+0.5*(x**3)+np.random.normal(-3,3,20)
plt.scatter(x,y,s=20)
plt.show()
```



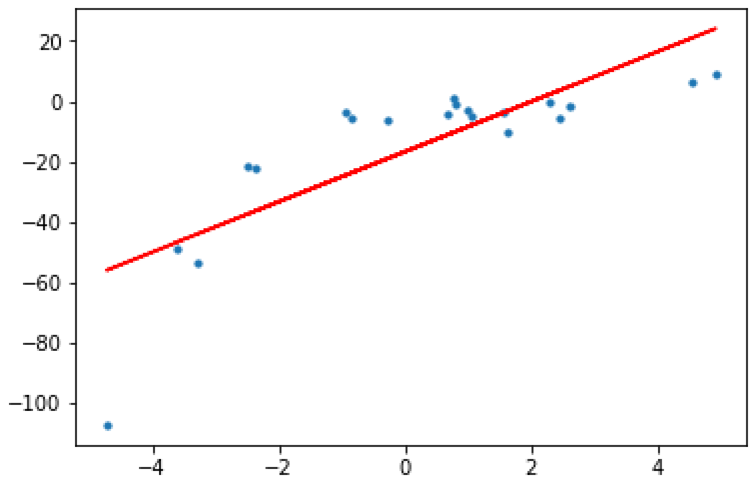
In [28]:

```
x=x[:,np.newaxis]
y=y[:,np.newaxis]
```

Creating Linear Regression Model
And Plotting The obtained result

In [29]:

```
from sklearn.linear_model import LinearRegression
model=LinearRegression()
model.fit(x,y)
y_pred=model.predict(x)
plt.scatter(x,y,s=10)
plt.plot(x,y_pred,color='r')
plt.show()
```



Importing The Library r2_score From sklearn.metrics
Calculating The r2 Score For Linear Regression Model

In [31]:

```
from sklearn.metrics import r2_score
r2 =r2_score(y,y_pred)
print("r2_score: ",r2)
```

```
r2_score:  0.6386750054827146
```

Importing The Library PloynomialFeatures From sklearn.preprocessing
Creating Polynomial Regression Model of degree 2

In [32]:

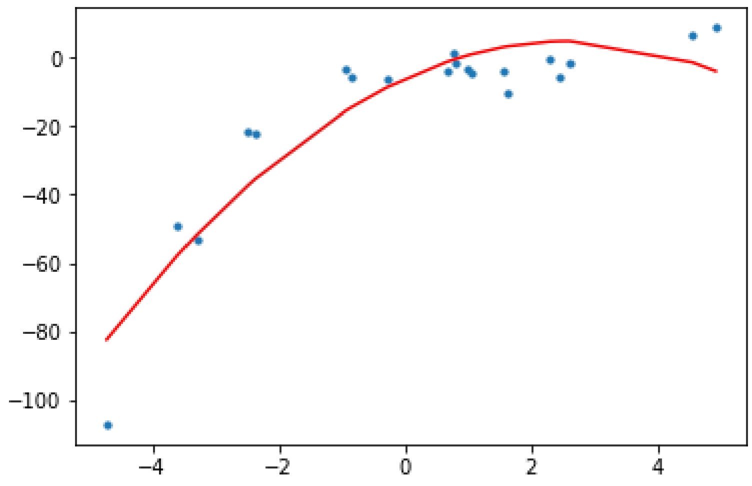
```
from sklearn.preprocessing import PolynomialFeatures
poly_features = PolynomialFeatures(degree=2)
x_poly=poly_features.fit_transform(x)
model=LinearRegression()
model.fit(x_poly,y)
y_poly_pred = model.predict(x_poly)
r2 =r2_score(y,y_poly_pred)
print("r2_score: ",r2)
```

```
r2_score:  0.8537647164420812
```

Plotting Graph For Polynomial Regression Model Of Degree 2

In [33]:

```
plt.scatter(x,y,s=10)
import operator
sort_axis=operator.itemgetter(0)
sorted_zip=sorted(zip(x,y_poly_pred),key=sort_axis)
x,y_poly_pred=zip(*sorted_zip)
plt.plot(x,y_poly_pred,color='r')
plt.show()
```



Creating Polynomial Regression Model of degree 3

In [34]:

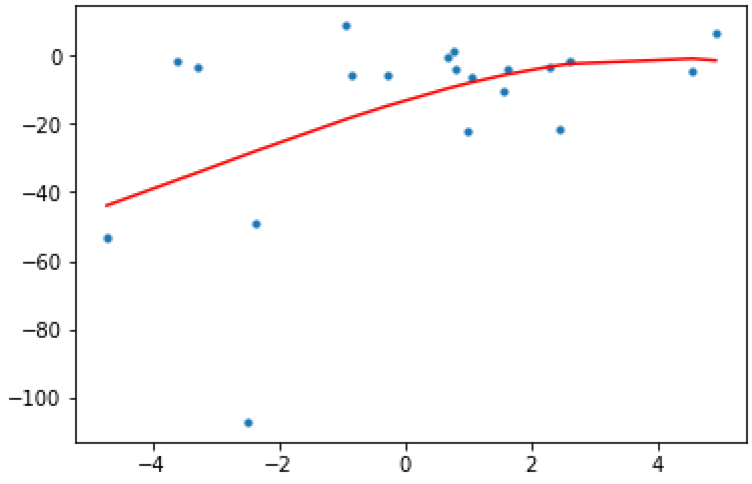
```
from sklearn.preprocessing import PolynomialFeatures
poly_features = PolynomialFeatures(degree=3)
x_poly=poly_features.fit_transform(x)
model=LinearRegression()
model.fit(x_poly,y)
y_poly_pred = model.predict(x_poly)
r2 =r2_score(y,y_poly_pred)
print("r2_score: ",r2)
```

```
r2_score:  0.23225482806073827
```

Plotting Graph For Polynomial Regression Model Of Degree 3

In [35]:

```
plt.scatter(x,y,s=10)
import operator
sort_axis=operator.itemgetter(0)
sorted_zip=sorted(zip(x,y_poly_pred),key=sort_axis)
x,y_poly_pred=zip(*sorted_zip)
plt.plot(x,y_poly_pred,color='r')
plt.show()
```



Creating Polynomial Regression Model of degree 4

In [36]:

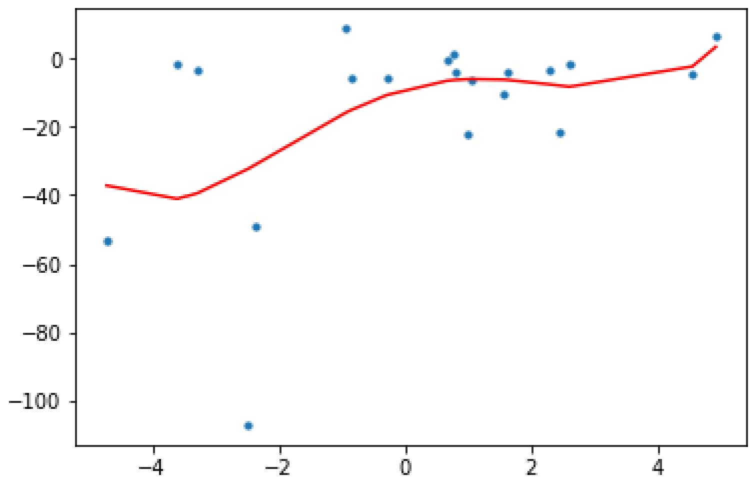
```
from sklearn.preprocessing import PolynomialFeatures
poly_features = PolynomialFeatures(degree=4)
x_poly=poly_features.fit_transform(x)
model=LinearRegression()
model.fit(x_poly,y)
y_poly_pred = model.predict(x_poly)
r2 =r2_score(y,y_poly_pred)
print("r2_score: ",r2)
```

```
r2_score:  0.25292987536181566
```

Plotting Graph For Polynomial Regression Model Of Degree 4

In [37]:

```
plt.scatter(x,y,s=10)
import operator
sort_axis=operator.itemgetter(0)
sorted_zip=sorted(zip(x,y_poly_pred),key=sort_axis)
x,y_poly_pred=zip(*sorted_zip)
plt.plot(x,y_poly_pred,color='r')
plt.show()
```



Creating Polynomial Regression Model of degree 5

In [38]:

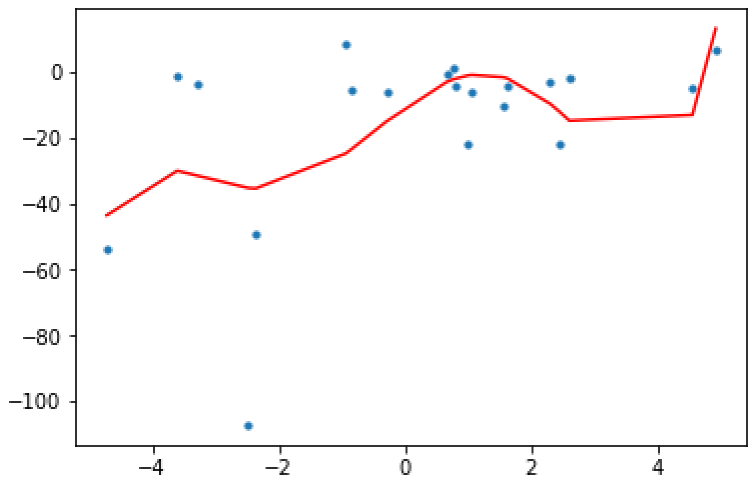
```
from sklearn.preprocessing import PolynomialFeatures
poly_features = PolynomialFeatures(degree=5)
x_poly=poly_features.fit_transform(x)
model=LinearRegression()
model.fit(x_poly,y)
y_poly_pred = model.predict(x_poly)
r2 =r2_score(y,y_poly_pred)
print("r2_score: ",r2)
```

```
r2_score:  0.3129204624156172
```

Plotting Graph For Polynomial Regression Model Of Degree 5

In [39]:

```
plt.scatter(x,y,s=10)
import operator
sort_axis=operator.itemgetter(0)
sorted_zip=sorted(zip(x,y_poly_pred),key=sort_axis)
x,y_poly_pred=zip(*sorted_zip)
plt.plot(x,y_poly_pred,color='r')
plt.show()
```



Conclusion

When we compare all the reult we got. And see the r2 Score. We found out that among all the Polynomial Regression Model 2, 3, 4 and 5, the Poplynomial Regression Model of Degree 2 is found to be Best. Which says that it is best in terms of Accuracy.

In [ ]:

```

```