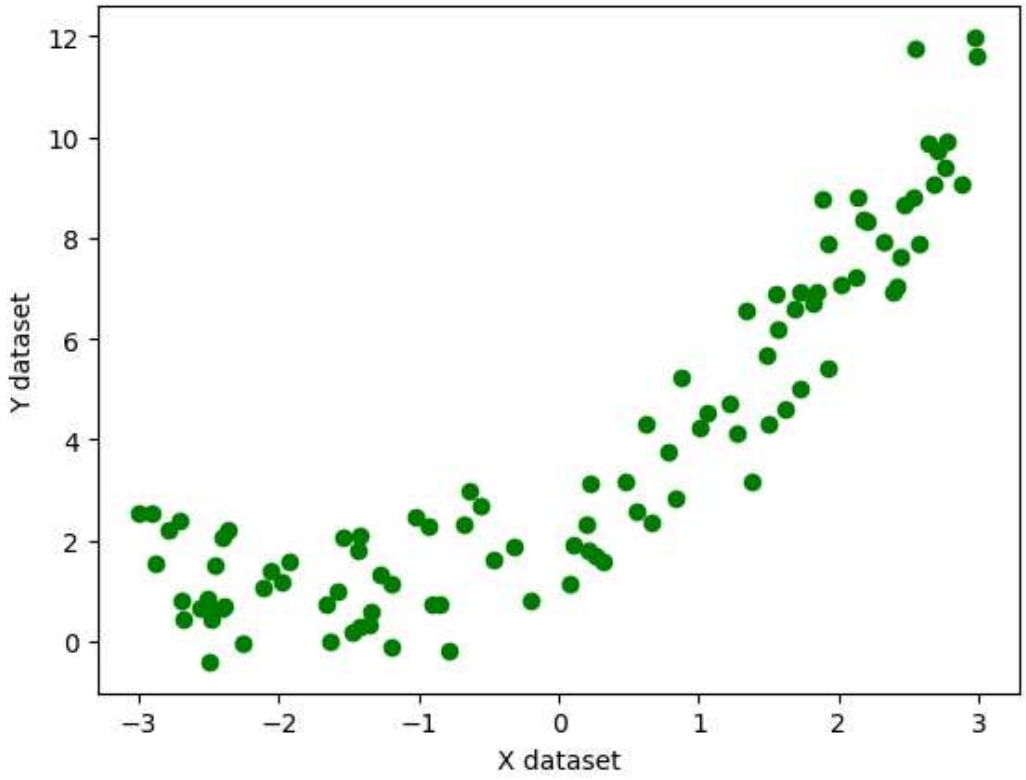
```
##import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline


x = 6 * np.random.rand(100, 1) - 3
y = 0.5 * x ** 2 + 1.5*x + 2 + np.random.randn(100, 1)
#quaratic equation used y = 0.5x^2+1.5x+2+outliers
plt.scatter(x, y, color='g')
plt.xlabel('X dataset')
plt.ylabel('Y dataset')
```

```
Text(0, 0.5, 'Y dataset')
```




```
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=42)


## Lets implement Simple Linear Regression
from sklearn.linear_model import LinearRegression
regression_1=LinearRegression()


regression_1.fit(X_train,y_train)
```

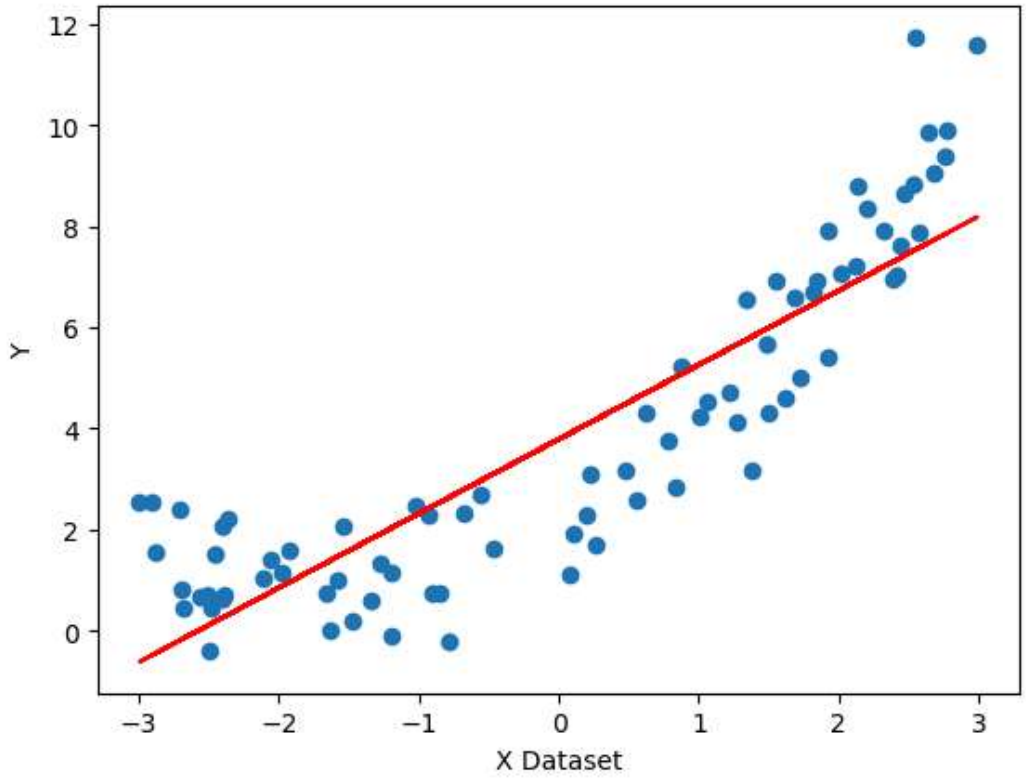
```
▾ LinearRegression
LinearRegression()
```

```
from sklearn.metrics import r2_score
score=r2_score(y_test,regression_1.predict(X_test))
print(score)
```

```
0.7575288451600102
```

```
## Lets visualize this model
plt.plot(X_train,regression_1.predict(X_train),color='r')
plt.scatter(X_train,y_train)
plt.xlabel("X Dataset")
plt.ylabel("Y")
```

```
Text(0, 0.5, 'Y')
```




```
#lets apply polynomial transformation
from sklearn.preprocessing import PolynomialFeatures
```

```
poly = PolynomialFeatures(degree = 2, include_bias= True)
X_train_poly = poly.fit_transform(X_train)
X_test_poly = poly.transform(X_test)
```

```
X_train_poly
```

```
[ 1.00000000e+00, -2.47863674e+00,  6.14364009e+00],
[ 1.00000000e+00, -2.40159535e+00,  5.76766022e+00],
[ 1.00000000e+00,  1.61683055e+00,  2.61414103e+00],
[ 1.00000000e+00,  6.20580023e-01,  3.85119565e-01],
[ 1.00000000e+00,  1.92645097e+00,  3.71121332e+00],
[ 1.00000000e+00,  1.93776984e-01,  3.75495197e-02],
[ 1.00000000e+00,  1.71947308e+00,  2.95658768e+00],
[ 1.00000000e+00, -1.63054862e+00,  2.65868881e+00],
[ 1.00000000e+00,  2.53904905e+00,  6.44677007e+00],
[ 1.00000000e+00, -7.89619526e-01,  6.23498996e-01],
[ 1.00000000e+00, -2.90958668e+00,  8.46569462e+00],
[ 1.00000000e+00, -2.39501478e+00,  5.73609582e+00],
[ 1.00000000e+00,  2.59205511e-01,  6.71874972e-02],
[ 1.00000000e+00,  2.56960207e+00,  6.60285479e+00],
[ 1.00000000e+00, -1.27251226e+00,  1.61928745e+00],
[ 1.00000000e+00,  2.77744400e+00,  7.71419519e+00],
[ 1.00000000e+00, -5.65380388e-01,  3.19654983e-01],
[ 1.00000000e+00,  1.55145117e+00,  2.40700073e+00],
[ 1.00000000e+00,  1.48943673e+00,  2.21842176e+00],
[ 1.00000000e+00, -6.82697089e-01,  4.66075315e-01],
[ 1.00000000e+00, -2.55776909e+00,  6.54218270e+00],
[ 1.00000000e+00, -1.46915692e+00,  2.15842206e+00],
[ 1.00000000e+00, -2.70921014e+00,  7.33981957e+00],
[ 1.00000000e+00,  1.34596503e+00,  1.81162187e+00],
[ 1.00000000e+00,  2.75382730e+00,  7.58356481e+00],
[ 1.00000000e+00,  2.99043232e+00,  8.94268547e+00],
[ 1.00000000e+00,  4.71577553e-01,  2.22385389e-01],
[ 1.00000000e+00, -2.05727154e+00,  4.23236621e+00],
[ 1.00000000e+00,  1.49282216e+00,  2.22851801e+00],
[ 1.00000000e+00,  1.68863081e+00,  2.85147401e+00],
[ 1.00000000e+00,  2.64218304e+00,  6.98113119e+00],
[ 1.00000000e+00, -9.01559469e-01,  8.12809476e-01],
[ 1.00000000e+00,  1.06446488e+00,  1.13308548e+00],
[ 1.00000000e+00,  2.38913694e+00,  5.70797532e+00],
[ 1.00000000e+00, -2.11741303e+00,  4.48343794e+00],
[ 1.00000000e+00,  8.38630919e-01,  7.03301818e-01],
[ 1.00000000e+00,  5.50614238e-01,  3.03176039e-01],
[ 1.00000000e+00, -1.57895332e+00,  2.49309358e+00],
[ 1.00000000e+00,  2.32241848e+00,  5.39362761e+00],
[ 1.00000000e+00,  2.12868785e+00,  4.53131196e+00],
[ 1.00000000e+00, -2.88459328e+00,  8.32087841e+00],
[ 1.00000000e+00,  2.68351920e+00,  7.20127528e+00],
[ 1.00000000e+00, -1.33990733e+00,  1.79535166e+00],
[ 1.00000000e+00, -2.99566012e+00,  8.97397958e+00],
[ 1.00000000e+00, -2.49961589e+00,  6.24807961e+00],
[ 1.00000000e+00,  2.44377161e+00,  5.97201970e+00],
[ 1.00000000e+00, -1.02888834e+00,  1.05861121e+00],
[ 1.00000000e+00,  7.84368114e-01,  6.15233339e-01],
[ 1.00000000e+00, -1.66147660e+00,  2.76050450e+00],
[ 1.00000000e+00,  1.37340123e+00,  1.88623094e+00]])
```

```
X_test_poly
```

```
array([[ 1.        ,  2.87510048,  8.26620277],
       [ 1.        , -0.64166612,  0.41173541],
       [ 1.        , -1.41865601,  2.01258487],
       [ 1.        ,  2.97791379,  8.86797054],
       [ 1.        , -2.79326832,  7.80234789],
       [ 1.        , -0.32071276,  0.10285667],
       [ 1.        , -2.25822829,  5.09959502],
```

```
[ 1.        ,  1.72595817,  2.97893159],
[ 1.        , -1.42013575,  2.01678554],
[ 1.        ,  0.21617792,  0.04673289],
[ 1.        ,  2.17113836,  4.71384178],
[ 1.        ,  1.88533361,  3.55448284],
[ 1.        ,  2.709747  ,  7.34272882],
[ 1.        , -2.51673805,  6.33397042],
[ 1.        , -0.19390947,  0.03760088],
[ 1.        , -1.43484479,  2.05877958],
[ 1.        , -1.36156631,  1.85386282],
[ 1.        ,  0.65662407,  0.43115517],
[ 1.        ,  0.32350461,  0.10465523],
[ 1.        ,  1.56086517,  2.43630009]])
```

```
from sklearn.metrics import r2_score


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

```
▾ LinearRegression
LinearRegression()
```

```
y_pred = regression.predict(X_test_poly)


score = r2_score(y_test, y_pred)
print(score)
```

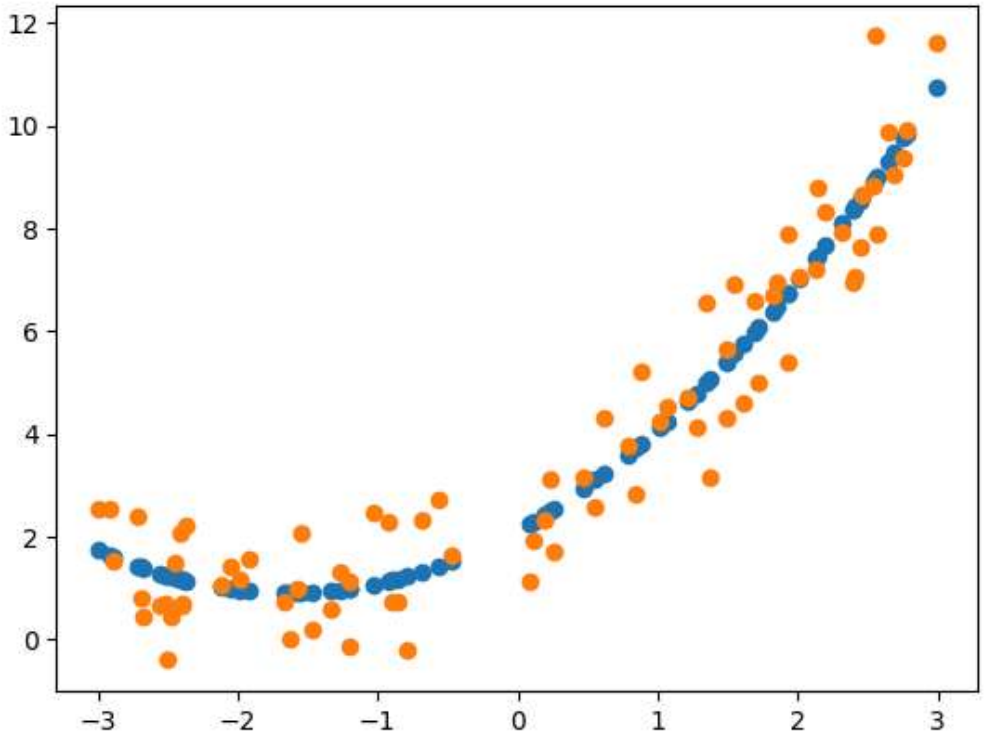
```
0.9240959265691082
```

```
print(regression.coef_)
```

```
[[0.         1.5044552  0.45845049]]
```

```
plt.scatter(X_train,regression.predict(X_train_poly))
plt.scatter(X_train,y_train)
```

```
<matplotlib.collections.PathCollection at 0x7f08c7ebb520>
```




```
poly=PolynomialFeatures(degree=3,include_bias=True)
X_train_poly=poly.fit_transform(X_train)
X_test_poly=poly.transform(X_test)
```

```
X_train_poly
```

```
[ 1.00000000e+00, -2.11741303e+00,  4.48343794e+00,
 -9.49328992e+00],
[ 1.00000000e+00,  8.38630919e-01,  7.03301818e-01,
  5.89810650e-01],
[ 1.00000000e+00,  5.50614238e-01,  3.03176039e-01,
  1.66933044e-01],
[ 1.00000000e+00, -1.57895332e+00,  2.49309358e+00,
 -3.93647839e+00],
[ 1.00000000e+00,  2.32241848e+00,  5.39362761e+00,
  1.25262604e+01],
[ 1.00000000e+00,  2.12868785e+00,  4.53131196e+00,
  9.64574870e+00],
[ 1.00000000e+00, -2.88459328e+00,  8.32087841e+00,
 -2.40023500e+01],
[ 1.00000000e+00,  2.68351920e+00,  7.20127528e+00,
  1.93247604e+01],
[ 1.00000000e+00, -1.33990733e+00,  1.79535166e+00,
 -2.40560486e+00],
[ 1.00000000e+00, -2.99566012e+00,  8.97397958e+00,
 -2.68829928e+01],
[ 1.00000000e+00, -2.49961589e+00,  6.24807961e+00,
 -1.56177991e+01],
[ 1.00000000e+00,  2.44377161e+00,  5.97201970e+00,
  1.45942522e+01],
[ 1.00000000e+00, -1.02888834e+00,  1.05861121e+00,
 -1.08919273e+00],
[ 1.00000000e+00,  7.84368114e-01,  6.15233339e-01,
  4.82569414e-01],
[ 1.00000000e+00, -1.66147660e+00,  2.76050450e+00,
 -4.58651364e+00],
[ 1.00000000e+00,  1.37340123e+00,  1.88623094e+00,
  2.59055190e+00]])
```

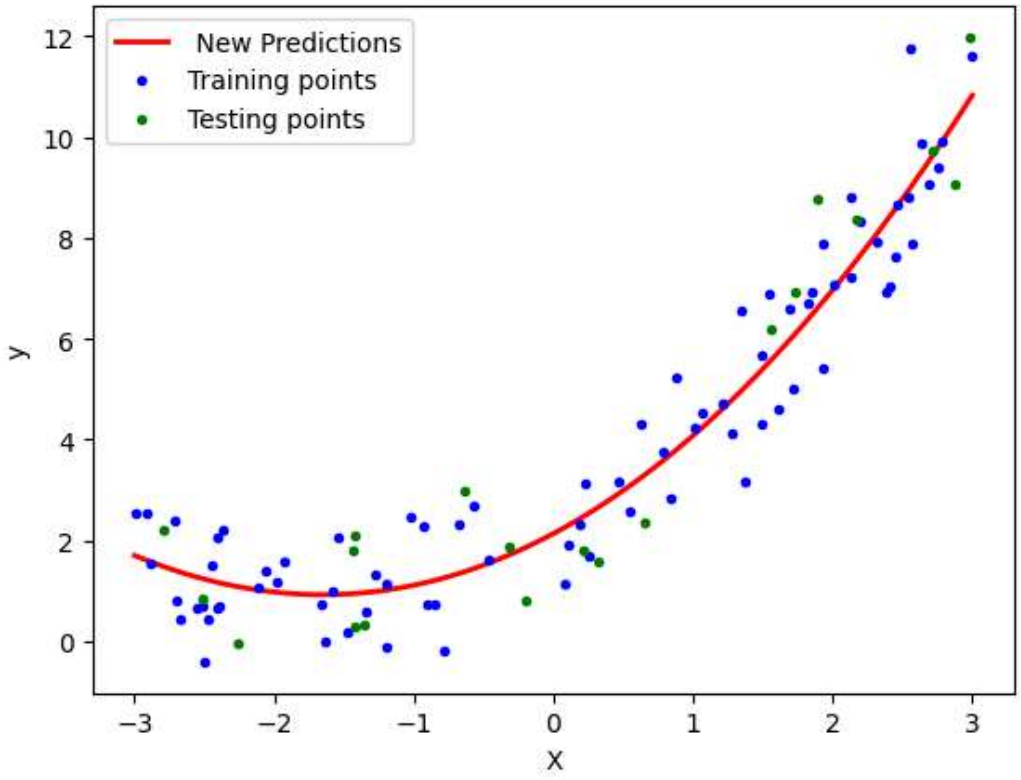
```
from sklearn.metrics import r2_score
regression = LinearRegression()
regression.fit(X_train_poly, y_train)
y_pred = regression.predict(X_test_poly)
score=r2_score(y_test,y_pred)
print(score)
```

```
0.9240410443625215
```

```
#3 Prediction of new data set
X_new = np.linspace(-3, 3, 200).reshape(200, 1)
X_new_poly = poly.transform(X_new)
X_new_poly
```

```
[ 1.00000000e+00,  2.33668342e+00,  5.46008939e+00,
  1.27585003e+01],
[ 1.00000000e+00,  2.36683417e+00,  5.60190399e+00,
  1.32587778e+01],
[ 1.00000000e+00,  2.39698492e+00,  5.74553673e+00,
  1.37719649e+01],
[ 1.00000000e+00,  2.42713568e+00,  5.89098760e+00,
  1.42982262e+01],
[ 1.00000000e+00,  2.45728643e+00,  6.03825661e+00,
  1.48377260e+01],
[ 1.00000000e+00,  2.48743719e+00,  6.18734375e+00,
  1.53906289e+01],
[ 1.00000000e+00,  2.51758794e+00,  6.33824903e+00,
  1.59570993e+01],
[ 1.00000000e+00,  2.54773869e+00,  6.49097245e+00,
  1.65373017e+01],
[ 1.00000000e+00,  2.57788945e+00,  6.64551400e+00,
  1.71314004e+01],
[ 1.00000000e+00,  2.60804020e+00,  6.80187369e+00,
  1.77395600e+01],
[ 1.00000000e+00,  2.63819095e+00,  6.96005151e+00,
  1.83619449e+01],
[ 1.00000000e+00,  2.66834171e+00,  7.12004747e+00,
  1.89987196e+01],
[ 1.00000000e+00,  2.69849246e+00,  7.28186157e+00,
  1.96500486e+01],
[ 1.00000000e+00,  2.72864322e+00,  7.44549380e+00,
  2.03160961e+01],
[ 1.00000000e+00,  2.75879397e+00,  7.61094417e+00,
  2.09970269e+01],
[ 1.00000000e+00,  2.78894472e+00,  7.77821267e+00,
  2.16930052e+01],
[ 1.00000000e+00,  2.81909548e+00,  7.94729931e+00,
  2.24041955e+01],
[ 1.00000000e+00,  2.84924623e+00,  8.11820409e+00,
  2.31307624e+01],
[ 1.00000000e+00,  2.87939698e+00,  8.29092700e+00,
  2.38728702e+01],
[ 1.00000000e+00,  2.90954774e+00,  8.46546804e+00,
  2.46306834e+01],
[ 1.00000000e+00,  2.93969849e+00,  8.64182723e+00,
  2.54043665e+01],
[ 1.00000000e+00,  2.96984925e+00,  8.82000455e+00,
  2.61940839e+01],
[ 1.00000000e+00,  3.00000000e+00,  9.00000000e+00,
  2.70000000e+01]])
```

```
y_new = regression.predict(X_new_poly)
plt.plot(X_new, y_new, "r-", linewidth=2, label=" New Predictions")
plt.plot(X_train, y_train, "b.",label='Training points')
plt.plot(X_test, y_test, "g.",label='Testing points')
plt.xlabel("X")
plt.ylabel("y")
plt.legend()
plt.show()
```



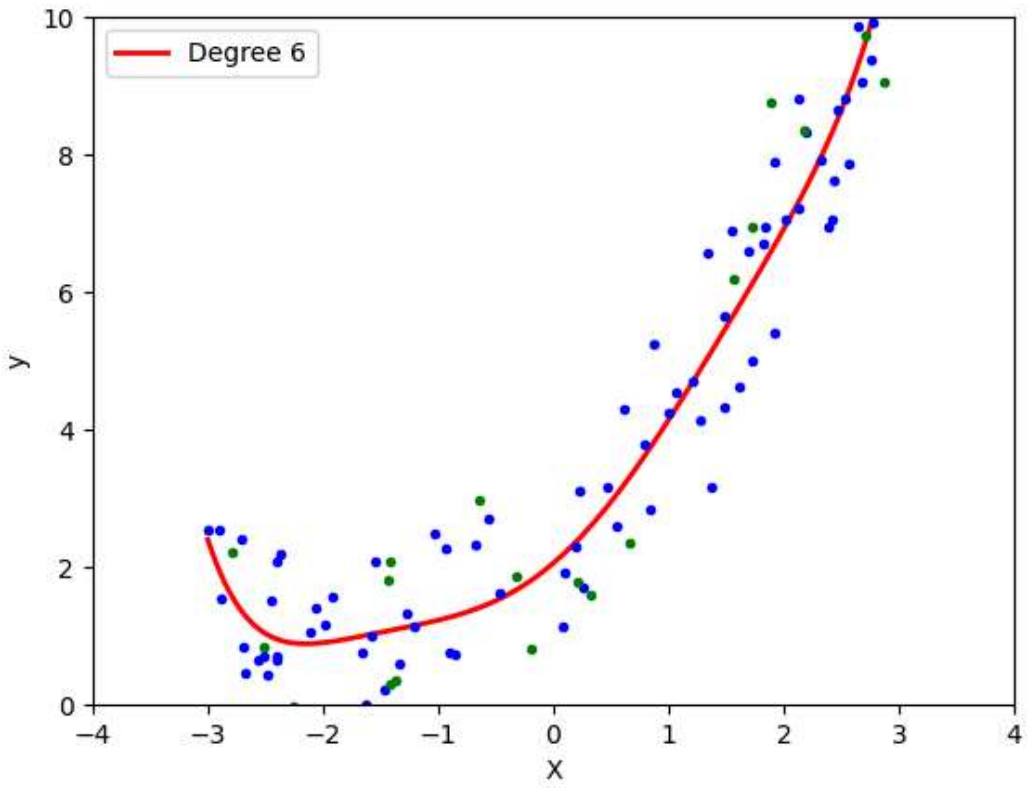

```
#pipeline concept


from sklearn.pipeline import Pipeline


def poly_regression(degree):
  X_new = np.linspace(-3,3,200).reshape(200, 1)

  poly_features = PolynomialFeatures(degree=degree, include_bias=True)
  lin_reg = LinearRegression()
  poly_regression=Pipeline([
      ("poly_features", poly_features),
      ("lin_reg", lin_reg)
  ])
  poly_regression.fit(X_train,y_train) ## ploynomial and fit of linear reression
  y_pred_new=poly_regression.predict(X_new)
  #plotting prediction line
  plt.plot(X_new, y_pred_new,'r', label="Degree " + str(degree), linewidth=2)
  plt.plot(X_train, y_train, "b.", linewidth=3)
  plt.plot(X_test, y_test, "g.", linewidth=3)
  plt.legend(loc="upper left")
  plt.xlabel("X")
  plt.ylabel("y")
  plt.axis([-4,4, 0, 10])
  plt.show()


poly_regression(6)
```


Degree 6
y
X
10
8
6
4
2
0
−4
−3
−2
−1
0
1
2
3
4