```
# 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 # generating random
y = 0.5 * X**2 + 1.5* X + 2 + np.random.randn(100,1)
# quadraic equation used y = 0.5 x^2 + 1.5x + 2 + outliers
```

```
X
```

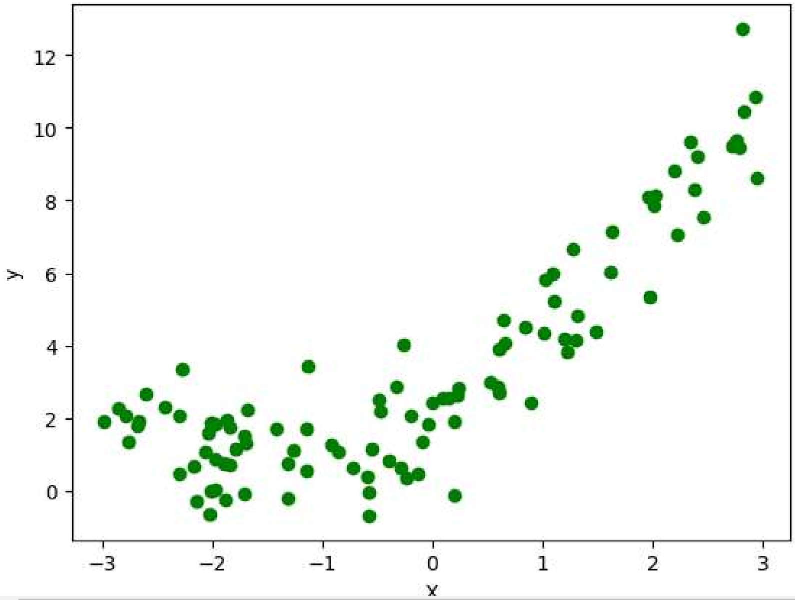
```
array([[-0.59292462],
       [-1.79295045],
       [ 0.2248603 ],
       [ 0.61214201],
       [ 2.03063131],
       [-1.89533084],
       [-1.71275517],
       [ 1.48531796],
       [ 1.10121956],
       [-0.13487131],
       [ 2.81520144],
       [ 0.64714572],
       [-0.03324061],
       [ 0.20415731],
       [-0.72286107],
       [-0.48431858],
       [-2.76704128],
       [ 1.97209441],
       [-0.28486864],
       [-1.31041575],
       [-1.14949744],
       [-1.97797078],
       [-2.02646536],
       [-1.96993044],
       [ 2.79373755],
       [-2.68173795],
       [-1.41608462],
       [-1.12751127],
       [-0.46935666],
       [-0.91368337],
       [ 2.83370467],
       [ 1.32019684],
       [-0.57655283],
       [ 0.59570938],
       [ 2.01127855],
       [ 1.09382655],
       [-2.14891444],
       [ 0.89789618],
       [ 2.38898945],
       [ 0.00696339],
       [ 0.08727929],
       [ 1.22356802],
       [-1.8661622 ],
       [-0.85375039],
       [-2.1647682 ],
       [-1.84362146],
       [ 2.41532717],
       [ 2.22245151],
       [ 2.20084075],
       [-1.26281989],
       [-2.66652324],
       [ 0.52458588],
       [ 2.93999691],
       [-0.57770845],
       [-0.32887007],
       [ 1.01011978],
       [ 1.62568912],
       [ 0.14591495],
```

```
y
```

```
array([[ 4.07310425e-01],
       [ 1.14877367e+00],
       [ 2.61929642e+00],
       [ 2.69599220e+00],
       [ 8.14311881e+00],
       [ 7.33611668e-01],
       [-7.97440852e-02],
       [ 4.39170454e+00],
       [ 5.23397827e+00],
       [ 4.72003214e-01],
       [ 1.27370489e+01],
       [ 4.68760338e+00],
       [ 1.81926993e+00],
       [-1.24074212e-01],
       [ 6.43394199e-01],
       [ 2.49627531e+00],
       [ 1.35833651e+00],
       [ 5.34354046e+00],
       [ 6.17387901e-01],
       [-2.06131988e-01],
       [ 5.50611882e-01],
       [ 1.34782547e-02],
       [-6.68760043e-01],
       [ 1.83011878e+00],
       [ 9.45678451e+00],
       [ 1.78706697e+00],
       [ 1.72353982e+00],
       [ 3.42527992e+00],
       [ 2.19854502e+00],
       [ 1.25333569e+00],
       [ 1.04469763e+01],
       [ 4.80844875e+00],
       [-4.00460690e-02],
       [ 2.87078874e+00],
       [ 7.84616144e+00],
       [ 5.96765899e+00],
       [-2.94833790e-01],
       [ 2.42302766e+00],
       [ 8.30420289e+00],
       [ 2.44059701e+00],
       [ 2.55440056e+00],
       [ 3.82327191e+00],
       [ 1.94273110e+00],
       [ 1.06983404e+00],
       [ 6.70712571e-01],
       [ 1.75935614e+00],
       [ 9.23722154e+00],
       [ 7.07028170e+00],
       [ 8.82131150e+00],
       [ 1.10311151e+00],
       [ 1.90889207e+00],
       [ 3.00495736e+00],
       [ 1.08504872e+01],
       [-6.86985564e-01],
       [ 2.86651818e+00],
       [ 4.32404742e+00],
       [ 6.00594008e+00],
       [ 2.56380192e+00],
```

```
plt.scatter(X,y, color = 'g')
plt.xlabel('X')
plt.ylabel('y')
```

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




```
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)
```

```
X_train.shape
```

```
(80, 1)
```

```
## Lets implement Simple Linear Regression
from sklearn.linear_model import LinearRegression
regression_1 = LinearRegression()
regression_1.fit(X_train, y_train)
```

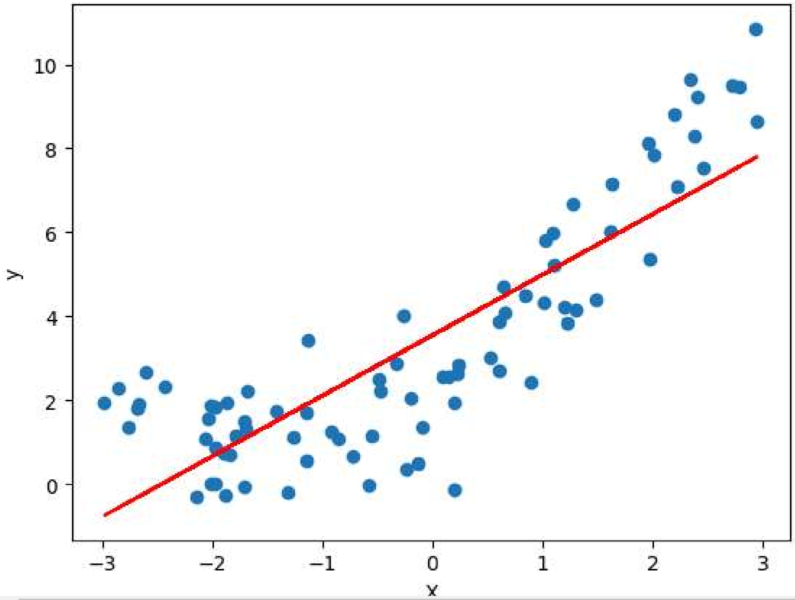
```
▾ LinearRegression
LinearRegression()
```

```
from sklearn.metrics import r2_score
score = r2_score(y_test, regression_1.predict(X_test))
print(score)
# Less accuracy as the line is a straight line
```

```
0.6584266126705804
```

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

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




```
#Lets apply Ploynomial Transformation
# h0(x) = Bo + B1 x1 + B2 x1^2
from sklearn.preprocessing import PolynomialFeatures
```

```
poly = PolynomialFeatures(degree = 2, include_bias = True)
# bias = True,so below is considered
# h0(x) = Bo * 1 + B1 x1 + B2 x1^2
X_train_poly = poly.fit_transform(X_train) # fit makes sure that the main data validation is train data no test refrence is used
X_test_poly = poly.transform(X_test) # apply the technique in test data
```

```
X_train_poly
# 1 + Bo X1 + B1 X1^2
```

```
array([[ 1.00000000e+00,  1.01011978e+00,  1.02034196e+00],
       [ 1.00000000e+00, -2.01692196e+00,  4.06797418e+00],
       [ 1.00000000e+00, -1.41608462e+00,  2.00529566e+00],
       [ 1.00000000e+00, -1.86616220e+00,  3.48256136e+00],
       [ 1.00000000e+00, -2.32558013e-01,  5.40832293e-02],
       [ 1.00000000e+00, -4.84318579e-01,  2.34564486e-01],
       [ 1.00000000e+00,  8.72792880e-02,  7.61767412e-03],
       [ 1.00000000e+00, -2.01491306e+00,  4.05987463e+00],
       [ 1.00000000e+00, -1.34871312e-01,  1.81902708e-02],
       [ 1.00000000e+00,  1.95521121e+00,  3.82285088e+00],
       [ 1.00000000e+00,  6.47145725e-01,  4.18797589e-01],
       [ 1.00000000e+00,  2.22245151e+00,  4.93929072e+00],
       [ 1.00000000e+00,  2.72143352e+00,  7.40620039e+00],
       [ 1.00000000e+00, -4.69356664e-01,  2.20295678e-01],
       [ 1.00000000e+00,  6.60960022e-01,  4.36868150e-01],
       [ 1.00000000e+00, -1.89533084e+00,  3.59227898e+00],
       [ 1.00000000e+00,  6.10106467e-01,  3.72229901e-01],
       [ 1.00000000e+00,  2.46449968e+00,  6.07375869e+00],
       [ 1.00000000e+00,  1.09382655e+00,  1.19645651e+00],
       [ 1.00000000e+00, -2.76704128e+00,  7.65651746e+00],
       [ 1.00000000e+00, -1.26281989e+00,  1.59471406e+00],
       [ 1.00000000e+00,  2.01127855e+00,  4.04524140e+00],
       [ 1.00000000e+00,  1.48531796e+00,  2.20616944e+00],
       [ 1.00000000e+00, -1.84020320e+00,  3.38634781e+00],
       [ 1.00000000e+00, -1.12751127e+00,  1.27128167e+00],
       [ 1.00000000e+00, -1.31041575e+00,  1.71718944e+00],
       [ 1.00000000e+00, -2.85202844e+00,  8.13406624e+00],
       [ 1.00000000e+00, -2.68173795e+00,  7.19171844e+00],
       [ 1.00000000e+00, -1.88157473e+00,  3.54032348e+00],
       [ 1.00000000e+00,  2.04157307e-01,  4.16802058e-02],
       [ 1.00000000e+00,  2.79373755e+00,  7.80496948e+00],
       [ 1.00000000e+00,  6.12142014e-01,  3.74717845e-01],
       [ 1.00000000e+00,  1.97209441e+00,  3.88915638e+00],
```

```
[ 1.00000000e+00,  2.38898945e+00,  5.70727061e+00],
[ 1.00000000e+00,  1.10121956e+00,  1.21268453e+00],
[ 1.00000000e+00, -2.97901940e+00,  8.87455657e+00],
[ 1.00000000e+00, -1.71275517e+00,  2.93353028e+00],
[ 1.00000000e+00, -2.03696153e+00,  4.14921229e+00],
[ 1.00000000e+00, -2.14891444e+00,  4.61783327e+00],
[ 1.00000000e+00,  1.19248461e+00,  1.42201954e+00],
[ 1.00000000e+00,  1.62568912e+00,  2.64286511e+00],
[ 1.00000000e+00,  1.30804689e+00,  1.71098666e+00],
[ 1.00000000e+00, -3.28870073e-01,  1.08155525e-01],
[ 1.00000000e+00, -8.53750395e-01,  7.28889737e-01],
[ 1.00000000e+00, -2.66652324e+00,  7.11034620e+00],
[ 1.00000000e+00, -1.70738959e+00,  2.91517919e+00],
[ 1.00000000e+00,  2.41532717e+00,  5.83380535e+00],
[ 1.00000000e+00,  2.01394616e-01,  4.05597915e-02],
[ 1.00000000e+00, -1.14439464e+00,  1.30963909e+00],
[ 1.00000000e+00,  1.62865242e+00,  2.65250870e+00],
[ 1.00000000e+00, -1.97281759e-01,  3.89200925e-02],
[ 1.00000000e+00,  1.22356802e+00,  1.49711870e+00],
[ 1.00000000e+00, -1.97192235e+00,  3.88847776e+00],
[ 1.00000000e+00,  2.20084075e+00,  4.84370001e+00],
[ 1.00000000e+00,  2.95316835e+00,  8.72120332e+00],
[ 1.00000000e+00,  1.45914954e-01,  2.12911739e-02],
[ 1.00000000e+00, -2.55707274e-01,  6.53862101e-02],
[ 1.00000000e+00, -5.76552827e-01,  3.32413162e-01],
```

```
from sklearn.metrics import r2_score
regression_2 = LinearRegression()
regression_2.fit(X_train_poly, y_train)
y_pred = regression_2.predict(X_test_poly)
score = r2_score(y_test, y_pred)
print(score)
```

```
0.9053141545351959
```

```
print(regression_2.coef_)
```

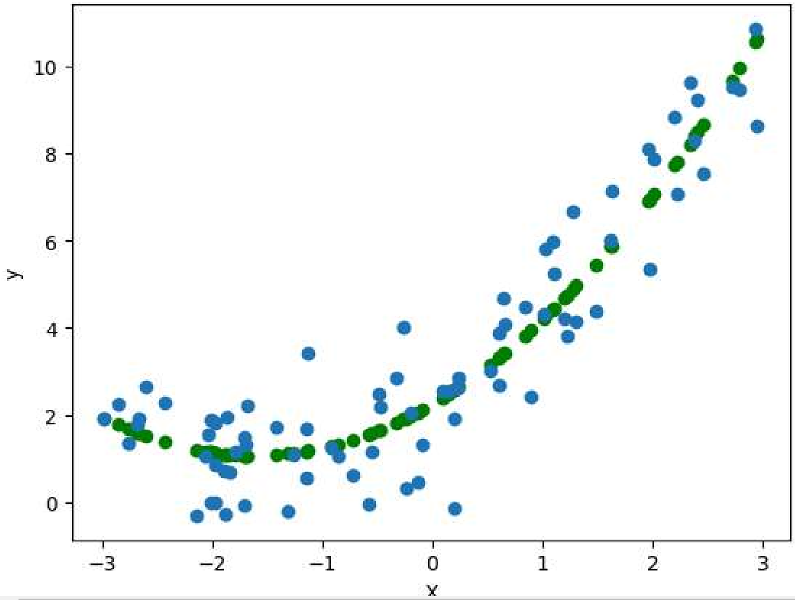
```
[[0.         1.47284521 0.45771528]]
```

```
print(regression_2.intercept_)
```

```
[2.25797166]
```

```
plt.scatter(X_train,regression_2.predict(X_train_poly), color = 'g')
plt.scatter(X_train, y_train)
plt.xlabel('X')
plt.ylabel('y')
```

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



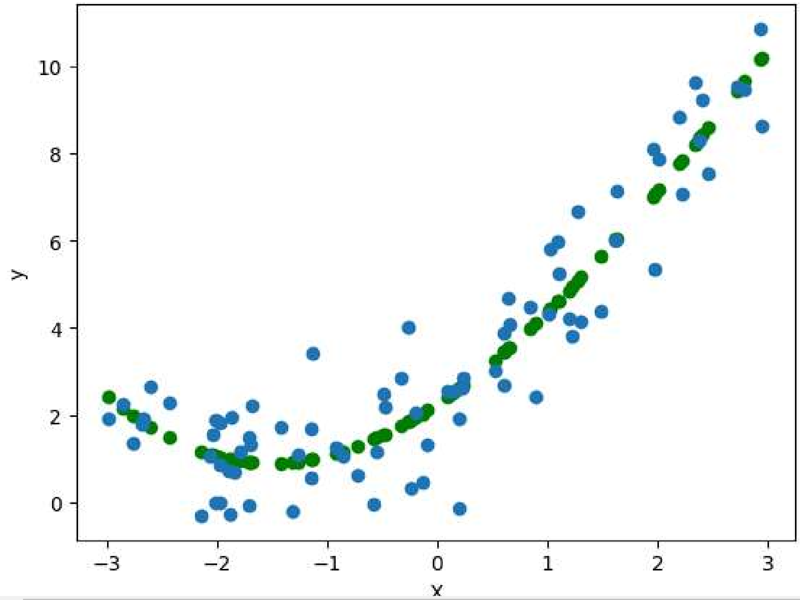

```
# For degree =3
poly3 = PolynomialFeatures(degree = 3, include_bias = True)
# bias = True,so below is considered
# h0(x) = Bo * 1 + B1 x1 + B2 x1^2
X_train_poly3 = poly3.fit_transform(X_train) # fit makes sure that the main data validation is train data no test refrence is use
X_test_poly3 = poly3.transform(X_test) # apply the technique in test data
```

```
from sklearn.metrics import r2_score
regression_3 = LinearRegression()
regression_3.fit(X_train_poly3, y_train)
y_pred = regression_2.predict(X_test_poly3)
score = r2_score(y_test, y_pred)
print(score)
```

```
0.9032885005131008
```

```
plt.scatter(X_train,regression_3.predict(X_train_poly3), color = 'g')
plt.scatter(X_train, y_train)
plt.xlabel('X')
plt.ylabel('y')
```

Text(0, 0.5, 'y')



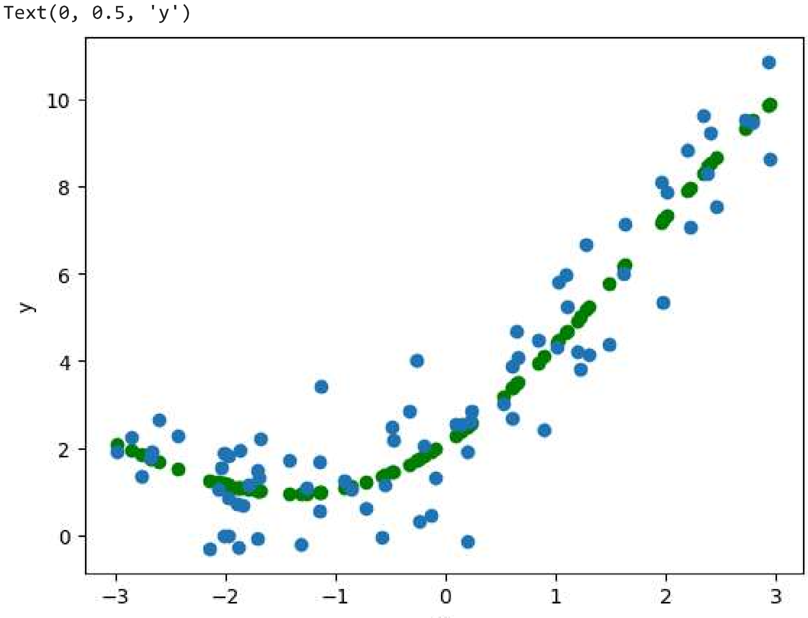

```
# For degree = 4
poly4 = PolynomialFeatures(degree = 4, include_bias = True)
# bias = True,so below is considered
# h0(x) = Bo * 1 + B1 x1 + B2 x1^2
X_train_poly4 = poly4.fit_transform(X_train) # fit makes sure that the main data validation is train data no test refrence is use
X_test_poly4 = poly4.transform(X_test) # apply the technique in test data

regression_4 = LinearRegression()
regression_4.fit(X_train_poly4, y_train)
y_pred = regression_4.predict(X_test_poly4)
score = r2_score(y_test, y_pred)
print(score)
```

0.902787330810482

```
plt.scatter(X_train,regression_4.predict(X_train_poly4), color = 'g')
plt.scatter(X_train, y_train)
plt.xlabel('X')
plt.ylabel('y')
```

Text(0, 0.5, 'y')



```
# 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
```

```
array([[ 1.00000000e+00, -3.00000000e+00,  9.00000000e+00],
       [ 1.00000000e+00, -2.96984925e+00,  8.82000455e+00],
       [ 1.00000000e+00, -2.93969849e+00,  8.64182723e+00],
       [ 1.00000000e+00, -2.90954774e+00,  8.46546804e+00],
       [ 1.00000000e+00, -2.87939698e+00,  8.29092700e+00],
       [ 1.00000000e+00, -2.84924623e+00,  8.11820409e+00],
       [ 1.00000000e+00, -2.81909548e+00,  7.94729931e+00],
       [ 1.00000000e+00, -2.78894472e+00,  7.77821267e+00],
       [ 1.00000000e+00, -2.75879397e+00,  7.61094417e+00],
       [ 1.00000000e+00, -2.72864322e+00,  7.44549380e+00],
       [ 1.00000000e+00, -2.69849246e+00,  7.28186157e+00],
       [ 1.00000000e+00, -2.66834171e+00,  7.12004747e+00],
       [ 1.00000000e+00, -2.63819095e+00,  6.96005151e+00],
       [ 1.00000000e+00, -2.60804020e+00,  6.80187369e+00],
       [ 1.00000000e+00, -2.57788945e+00,  6.64551400e+00],
       [ 1.00000000e+00, -2.54773869e+00,  6.49097245e+00],
       [ 1.00000000e+00, -2.51758794e+00,  6.33824903e+00],
       [ 1.00000000e+00, -2.48743719e+00,  6.18734375e+00],
       [ 1.00000000e+00, -2.45728643e+00,  6.03825661e+00],
       [ 1.00000000e+00, -2.42713568e+00,  5.89098760e+00],
       [ 1.00000000e+00, -2.39698492e+00,  5.74553673e+00],
       [ 1.00000000e+00, -2.36683417e+00,  5.60190399e+00],
       [ 1.00000000e+00, -2.33668342e+00,  5.46008939e+00],
       [ 1.00000000e+00, -2.30653266e+00,  5.32009293e+00],
       [ 1.00000000e+00, -2.27638191e+00,  5.18191460e+00],
       [ 1.00000000e+00, -2.24623116e+00,  5.04555441e+00],
       [ 1.00000000e+00, -2.21608040e+00,  4.91101235e+00],
       [ 1.00000000e+00, -2.18592965e+00,  4.77828843e+00],
       [ 1.00000000e+00, -2.15577889e+00,  4.64738264e+00],
       [ 1.00000000e+00, -2.12562814e+00,  4.51829499e+00],
       [ 1.00000000e+00, -2.09547739e+00,  4.39102548e+00],
       [ 1.00000000e+00, -2.06532663e+00,  4.26557410e+00],
       [ 1.00000000e+00, -2.03517588e+00,  4.14194086e+00],
       [ 1.00000000e+00, -2.00502513e+00,  4.02012575e+00],
       [ 1.00000000e+00, -1.97487437e+00,  3.90012878e+00],
       [ 1.00000000e+00, -1.94472362e+00,  3.78194995e+00],
       [ 1.00000000e+00, -1.91457286e+00,  3.66558925e+00],
```