```
# 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
```

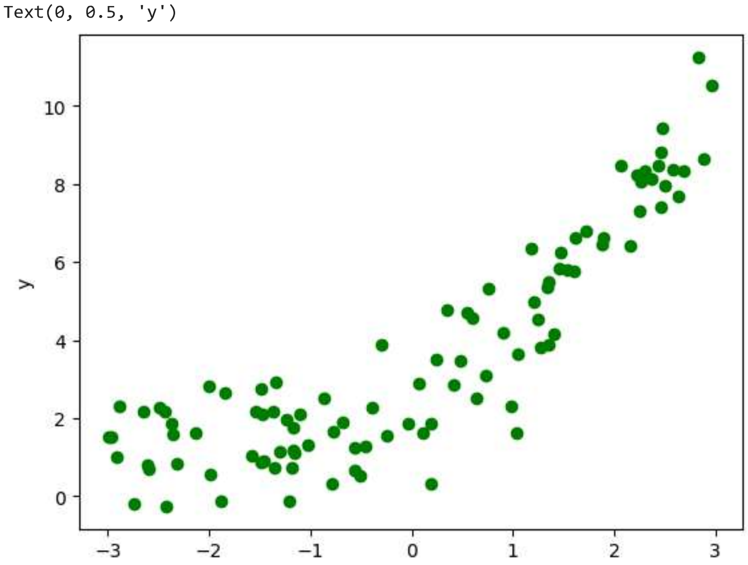
```
       [ 0.35494045],
       [ 1.59892478],
       [ 0.98077218],
       [ 0.11279075],
       [-1.33787593],
       [ 2.45827246],
       [-2.95960175],
       [ 0.90533741],
       [-2.49182981],
       [-0.7750623 ],
       [-2.3601102 ],
       [ 1.40181601],
       [-2.60567264],
       [ 2.825867  ],
       [ 0.07448628],
       [ 0.19013754],
       [ 1.61801902],
       [-0.78885902],
       [ 1.35071778],
       [-1.5822502 ],
       [-1.84211822],
       [-2.43825107],
       [ 2.05968257],
       [ 1.04408029],
       [ 1.89541579],
       [-1.48270101],
       [-0.55580853],
       [-0.56043315],
       [-2.59761692],
       [-1.47417372],
       [ 2.23010071],
       [-1.98251825],
       [-2.1317708 ],
       [ 2.26453949],
       [ 1.03138606],
       [ 1.4699336 ],
       [-2.4258232 ],
       [-1.17086409],
       [ 1.88054604],
       [ 2.15688848],
       [ 2.45632316],
       [ 2.96838235],
       [ 0.24949284],
       [-1.29567834],
       [-2.3238105 ],
       [-2.98553445],
       [-0.03068258],
       [ 1.25004397],
       [ 2.68627352],
       [ 0.60461731],
       [-1.18051507],
       [-1.45847585],
       [-1.14987708],
       [-1.37011553],
       [ 2.47028203],
       [-1.023296  ],
       [ 2.50247315],
       [-1.09702919]])
```

```
y
```

```
       [ 4.17969948],
       [ 2.27142537],
       [ 1.6448053 ],
       [ 1.60247106],
       [ 4.14358636],
       [ 0.81949879],
       [11.23878487],
       [ 2.90446672],
       [ 0.32228212],
       [ 6.62357954],
       [ 0.32394935],
       [ 5.48159556],
       [ 1.04109018],
       [ 2.64615801],
       [ 2.17314113],
       [ 8.46860419],
       [ 3.64960727],
       [ 6.61338358],
       [ 0.88771445],
       [ 1.25139284],
       [ 0.66516108],
       [ 0.70905381],
       [ 2.1051586 ],
       [ 8.21384737],
       [ 0.57070955],
       [ 1.62416024],
       [ 8.03904891],
       [ 1.61666109],
       [ 6.25214016],
       [-0.25511155],
       [ 1.75897703],
       [ 6.45273422],
       [ 6.41438069],
       [ 7.41737111],
       [10.52892286],
       [ 3.50819154],
       [ 1.14729421],
       [ 0.83620258],
       [ 1.5310346 ],
       [ 1.85879419],
       [ 4.53687863],
       [ 8.31283793],
       [ 4.58076515],
       [ 0.73490872],
       [ 0.91825945],
       [ 1.10706921],
       [ 2.17726851],
       [ 9.40658024],
       [ 1.32923981],
       [ 7.93758429],
       [ 2.09560507]])
```

```
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)
```

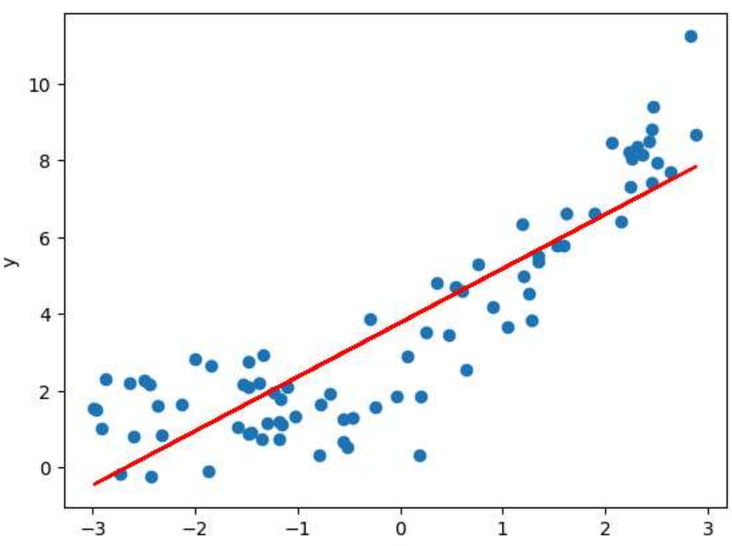
```
▾ LinearRegression
LinearRegression()
```

```
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
```

```
0.6782008705031266
```

```
# 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
# hθ(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
# hθ(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
```

```
       [ 1.00000000e+00,  2.30501387e+00,  5.31308894e+00],
       [ 1.00000000e+00, -1.84211822e+00,  3.39339953e+00],
       [ 1.00000000e+00,  2.42892762e+00,  5.89968937e+00],
       [ 1.00000000e+00, -1.23162043e+00,  1.51688887e+00],
       [ 1.00000000e+00,  6.46298571e-01,  4.17701842e-01],
       [ 1.00000000e+00, -2.45957799e-01,  6.04952389e-02],
       [ 1.00000000e+00, -2.88013935e+00,  8.29520265e+00],
       [ 1.00000000e+00, -2.91087782e+00,  8.47320971e+00],
       [ 1.00000000e+00, -2.42582320e+00,  5.88461820e+00],
       [ 1.00000000e+00, -1.99904976e+00,  3.99619993e+00],
       [ 1.00000000e+00,  2.05968257e+00,  4.24229229e+00],
       [ 1.00000000e+00,  7.54681176e-01,  5.69543677e-01],
       [ 1.00000000e+00,  1.25004397e+00,  1.56260993e+00],
       [ 1.00000000e+00,  7.44862815e-02,  5.54820613e-03],
       [ 1.00000000e+00, -1.09702919e+00,  1.20347305e+00],
       [ 1.00000000e+00, -2.60567264e+00,  6.78952988e+00],
       [ 1.00000000e+00,  1.59892478e+00,  2.55656047e+00],
       [ 1.00000000e+00, -2.49182981e+00,  6.20921579e+00],
       [ 1.00000000e+00, -1.48270101e+00,  2.19840229e+00],
       [ 1.00000000e+00, -1.33787593e+00,  1.78991200e+00],
       [ 1.00000000e+00, -5.55808528e-01,  3.08923119e-01],
       [ 1.00000000e+00, -1.58225020e+00,  2.50351570e+00],
       [ 1.00000000e+00, -1.02329600e+00,  1.04713470e+00],
       [ 1.00000000e+00, -1.17086409e+00,  1.37092271e+00],
       [ 1.00000000e+00,  2.88890384e+00,  8.34576538e+00],
       [ 1.00000000e+00,  1.61801902e+00,  2.61798556e+00],
       [ 1.00000000e+00, -2.95960175e+00,  8.75924254e+00],
       [ 1.00000000e+00,  2.50247315e+00,  6.26237185e+00],
       [ 1.00000000e+00,  1.90137540e-01,  3.61522842e-02],
       [ 1.00000000e+00,  2.26453949e+00,  5.12813909e+00],
       [ 1.00000000e+00, -1.48068398e+00,  2.19242504e+00],
       [ 1.00000000e+00, -1.14987708e+00,  1.32221731e+00],
       [ 1.00000000e+00, -7.88859020e-01,  6.22298554e-01],
       [ 1.00000000e+00, -2.43825107e+00,  5.94506830e+00],
       [ 1.00000000e+00,  2.49492837e-01,  6.22466757e-02],
       [ 1.00000000e+00,  1.18562196e+00,  1.40569944e+00],
       [ 1.00000000e+00, -2.93836967e-01,  8.63401630e-02],
       [ 1.00000000e+00, -6.80484043e-01,  4.63058532e-01],
       [ 1.00000000e+00, -2.36011020e+00,  5.57012015e+00],
       [ 1.00000000e+00, -5.15534987e-01,  2.65776323e-01],
       [ 1.00000000e+00,  2.63175147e+00,  6.92611580e+00],
       [ 1.00000000e+00,  1.53548179e+00,  2.35770433e+00],
       [ 1.00000000e+00, -2.98553445e+00,  8.91341593e+00],
       [ 1.00000000e+00,  6.04617307e-01,  3.65562088e-01],
       [ 1.00000000e+00, -2.13177080e+00,  4.54444674e+00],
       [ 1.00000000e+00, -2.32381050e+00,  5.40009526e+00],
       [ 1.00000000e+00,  2.45632316e+00,  6.03352349e+00],
       [ 1.00000000e+00,  1.34510118e+00,  1.80929719e+00],
       [ 1.00000000e+00,  1.35071778e+00,  1.82443851e+00],
       [ 1.00000000e+00, -1.47417372e+00,  2.17318815e+00],
       [ 1.00000000e+00,  1.97137798e-01,  3.88633115e-02],
       [ 1.00000000e+00, -1.18051507e+00,  1.39361584e+00],
       [ 1.00000000e+00, -7.75062301e-01,  6.00721571e-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)
```

```
0.8551468588505584
```

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

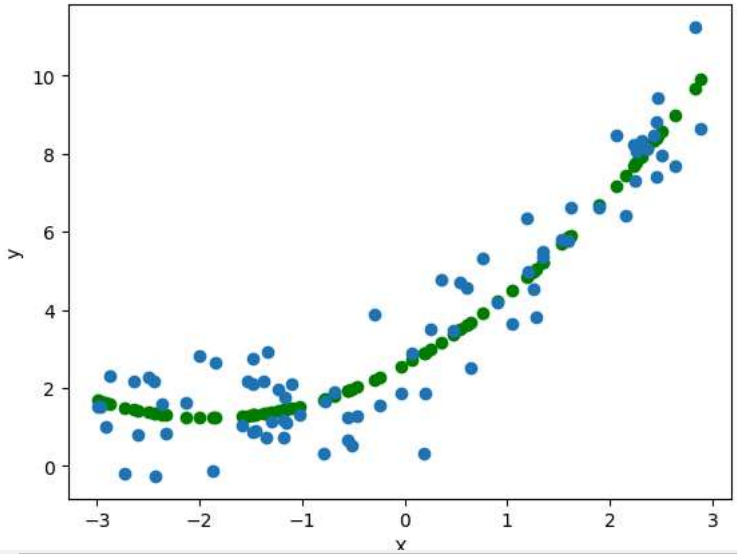
```
[[0.         1.43527917 0.37730751]]
```

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

```
[2.60211072]
```

```
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')



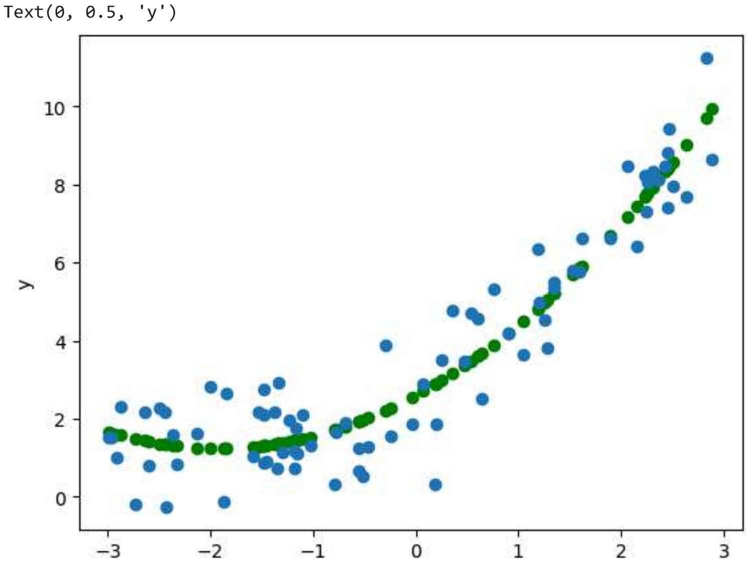
```
# For degree =3
poly3 = PolynomialFeatures(degree = 3, include_bias = True)
# bias = True,so below is considered
# h0(x) = Bo * 1 + B1 x1 + B2 x1^2
X_train_poly3 = poly3.fit_transform(X_train) # fit makes sure that the main data validation is train data no test refrence is used
X_test_poly3 = poly3.transform(X_test) # 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_3.predict(X_test_poly3)
score = r2_score(y_test, y_pred)
print(score)
```

0.8560655799538519

```
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
```

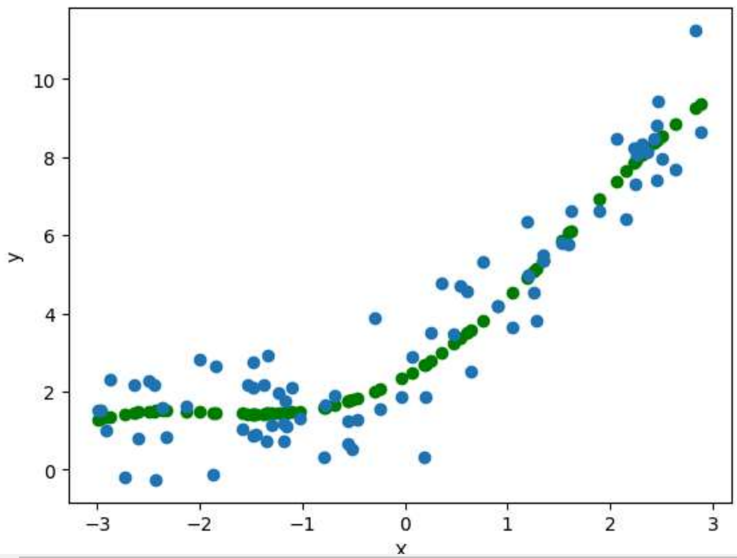
```
X_train_poly4 = poly4.fit_transform(X_train) # fit makes sure that the main data validation is train data no test refrence is used
X_test_poly4 = poly4.transform(X_test) # apply the technique in test data

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)
```

```
0.8489518106014626
```

```
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')
```

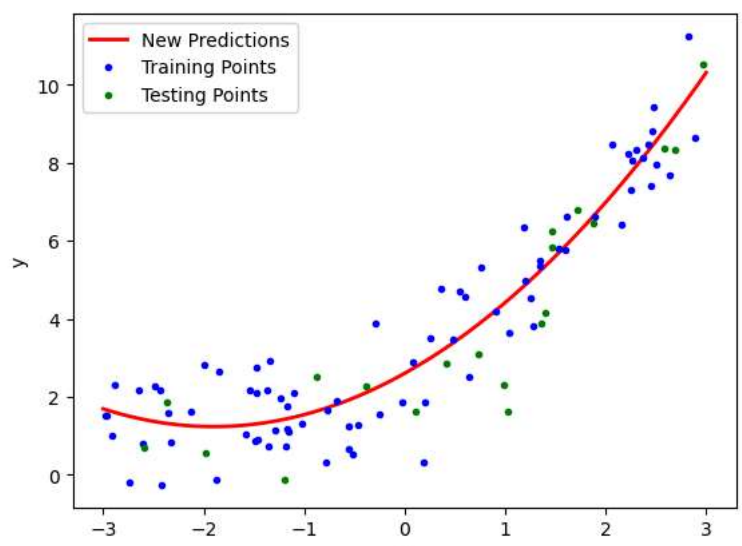



```
# Prediction of new data set
X_new = np.linspace(-3,3,200).reshape(200,1)
X_new_poly = poly.transform(X_new)
```

```
X_new_poly
```

```
[ 1.00000000e+00,  2.18592965e+00,  4.77828843e+00],
[ 1.00000000e+00,  2.21608040e+00,  4.91101235e+00],
[ 1.00000000e+00,  2.24623116e+00,  5.04555441e+00],
[ 1.00000000e+00,  2.27638191e+00,  5.18191460e+00],
[ 1.00000000e+00,  2.30653266e+00,  5.32009293e+00],
[ 1.00000000e+00,  2.33668342e+00,  5.46008939e+00],
[ 1.00000000e+00,  2.36683417e+00,  5.60190399e+00],
[ 1.00000000e+00,  2.39698492e+00,  5.74553673e+00],
[ 1.00000000e+00,  2.42713568e+00,  5.89098760e+00],
[ 1.00000000e+00,  2.45728643e+00,  6.03825661e+00],
[ 1.00000000e+00,  2.48743719e+00,  6.18734375e+00],
[ 1.00000000e+00,  2.51758794e+00,  6.33824903e+00],
[ 1.00000000e+00,  2.54773869e+00,  6.49097245e+00],
[ 1.00000000e+00,  2.57788945e+00,  6.64551400e+00],
[ 1.00000000e+00,  2.60804020e+00,  6.80187369e+00],
[ 1.00000000e+00,  2.63819095e+00,  6.96005151e+00],
[ 1.00000000e+00,  2.66834171e+00,  7.12004747e+00],
[ 1.00000000e+00,  2.69849246e+00,  7.28186157e+00],
[ 1.00000000e+00,  2.72864322e+00,  7.44549380e+00],
[ 1.00000000e+00,  2.75879397e+00,  7.61094417e+00],
[ 1.00000000e+00,  2.78894472e+00,  7.77821267e+00],
[ 1.00000000e+00,  2.81909548e+00,  7.94729931e+00],
[ 1.00000000e+00,  2.84924623e+00,  8.11820409e+00],
[ 1.00000000e+00,  2.87939698e+00,  8.29092700e+00],
[ 1.00000000e+00,  2.90954774e+00,  8.46546804e+00],
[ 1.00000000e+00,  2.93969849e+00,  8.64182723e+00],
[ 1.00000000e+00,  2.96984925e+00,  8.82000455e+00],
[ 1.00000000e+00,  3.00000000e+00,  9.00000000e+00]])
```

```
y_new = regression_2.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 Concepts

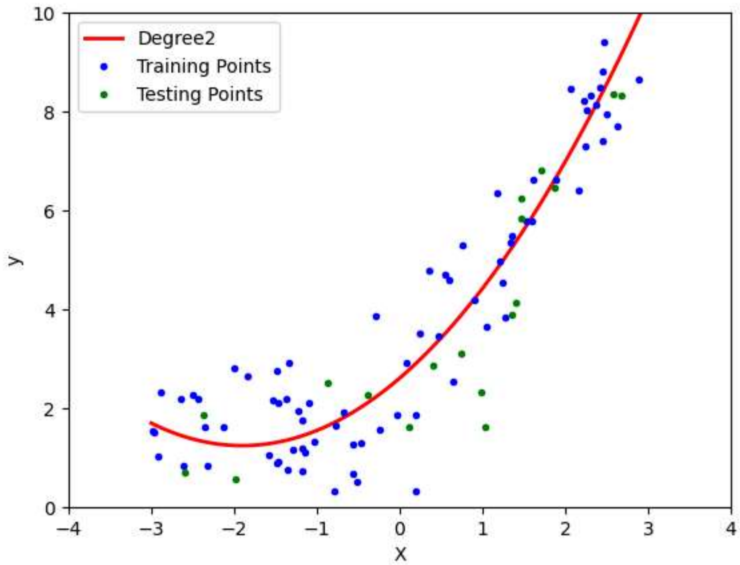
```
from sklearn.pipeline import Pipeline
```

Suggested code may be subject to a license |

```
def poly_regression(degree):
  X_new = np.linspace(-3,3,200).reshape(200,1)
  X_new_poly = poly.transform(X_new)
  poly_features = PolynomialFeatures(degree = degree, include_bias = True)
  lin_reg = LinearRegression()
  polynomial_regression = Pipeline([
      ("poly_features", poly_features),
      ("lin_reg", lin_reg)
  ])
  polynomial_regression.fit(X_train,y_train) ## Ploynomail and fit of linear regression
  y_pred_new = polynomial_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.', label = "Training Points")
  plt.plot(X_test, y_test, 'g.', label = "Testing Points")
  plt.xlabel('X')
  plt.ylabel('y')
  plt.legend(loc = "upper left")
  plt.axis([-4, 4, 0, 10])
  plt.show()
  y_pred = polynomial_regression.predict(X_test)
  score = r2_score(y_test, y_pred)
  print(score)
```

```
poly_regression(2)
```




```
0.8551468588505584
```

```
Start coding or generate with AI.
```

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.