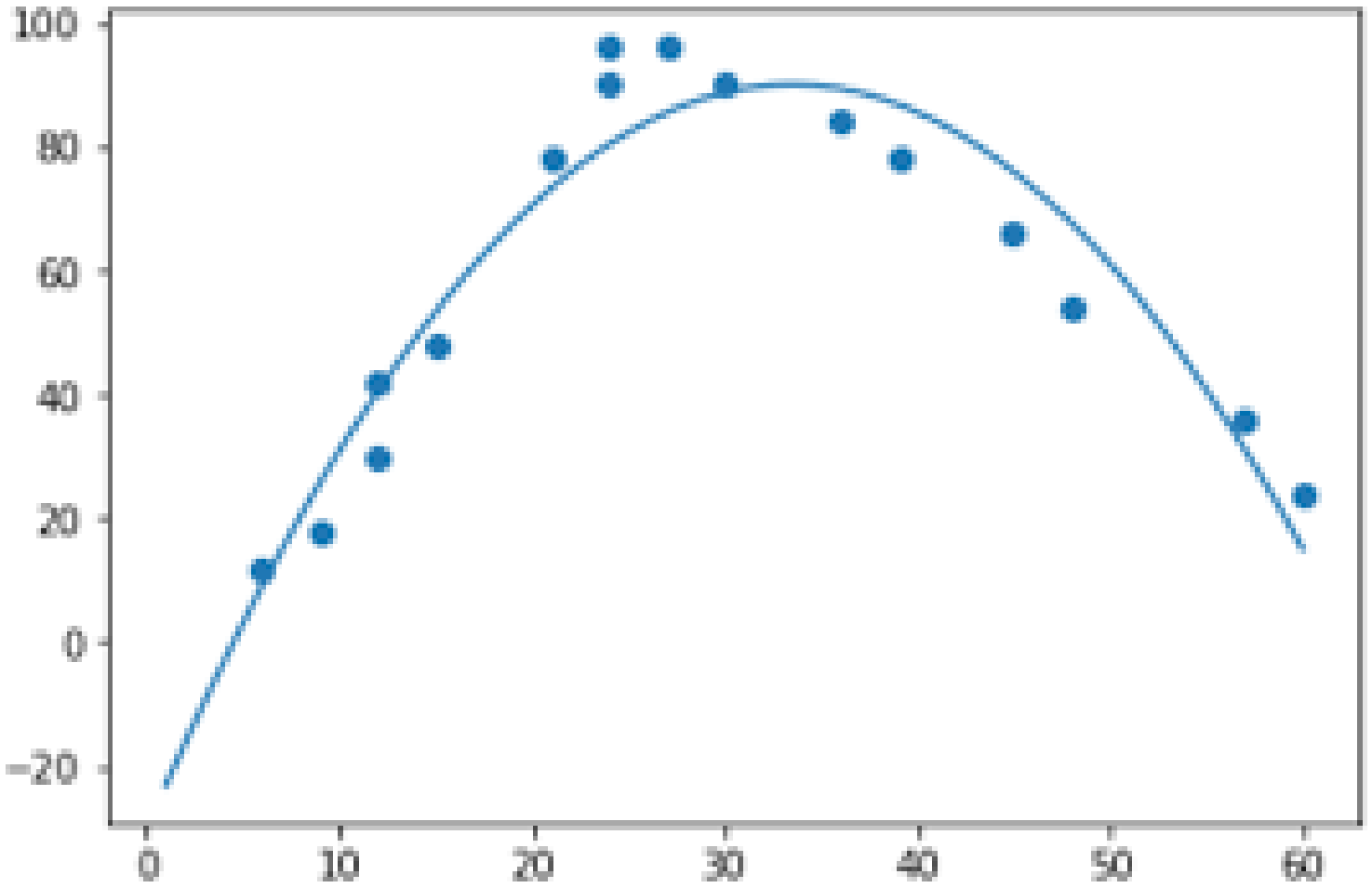
Polynomial Regression-Hồi quy bội

Nhóm trình bày: Nhóm12
Vũ Quang Minh-18110150
Đỗ Trung Hậu-18110091

Ngày 18 tháng 12 năm 2020

- **1) Polynomial Regression**
- **2) Thuật toán Polynomial Regression**
- **3) Ví dụ Polynomial Regresion**

1) Polynomial Regresion



1) Polynomial Regresion

Mô hình thống kê bội(Polynomial Regression) có một biến ngẫu nhiên Y và tập các biến x,x^2,x^3,...,x^n là phương trình có dạng:

$$Y(w,x) = w_0 + w_1 x + w_2 x^2 + ... + w_n x^n + \epsilon$$

Nếu đặt $x_1 = x, x_2 = x^2, .., x_n = x^n$ ta có:

$$Y(w,x) = w_0 + w_1 x_1 + w_2 x_2 + ... + w_n x_n + \epsilon$$

Phương trình còn có dạng:

$$Y(w,x) = w_0 + w_1 x_1 + w_2 x_2 + w_{12} x_1 x_2 + w_4 x_1^2 + \epsilon$$

Đặt $x_3 = x_1 x_2, w_3 = w_{12}, x_4 = x_1^2$

$$Y(w,x) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + \epsilon$$

- Với $w_0, w_1, w_2, ..., w_n$ là các tham số chưa biết còn gọi là hệ số hồi quy
- $x_1, x_2, ..., x_n$ là các biến độc lập, không ngẫu nhiên
- ϵ là thành phần sai số,ϵ được giả sử tuân theo phân phối chuẩn với E(ϵ)=0 và Var(ϵ)=σ^2

Xét $y_1, y_2, .., y_n$ là n giá trị quan trắc độc lập Y. Khi đó mỗi y_i được viết dưới dạng:

$$y_i = w_0 + w_1 x_1 + w_2 x_{i2} + ... + w_k x_{ik} + \epsilon_i \qquad i = 1, .., n$$

Với x_{ij} là các biến độc lập thứ j của quan trắc thứ i, i=1,2,..n và các sai số ϵ_i độc lập tương tự như trong mô hình hồi quy tuyến tính.
Đặt $x_0 = 1$, định nghĩa các ma trận sau:

$$\mathbf{Y} = \begin{pmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ y_n \end{pmatrix} \quad \mathbf{X} = \begin{pmatrix} 1 & x_{11} & x_{12} & ... & x_{1k} \\ 1 & x_{21} & x_{22} & ... & x_{2k} \\ . & & & & \\ . & & & & \\ . & & & & \\ 1 & x_{n1} & x_{n2} & ... & x_{nk} \end{pmatrix}$$

$$\mathbf{W} = \begin{pmatrix} w_0 \\ w_1 \\ . \\ . \\ . \\ w_n \end{pmatrix} \quad \epsilon = \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ . \\ . \\ . \\ \epsilon_n \end{pmatrix}$$

Mô hình hồi quy bội(Polynomial Regression) có dạng ma trận như sau:

$$y = XW + \epsilon \quad (1)$$

Tổng bình phương thặng dư trong mô hình hồi quy bội được định nghĩa như sau:

$$L = \sum_1^n \epsilon_i = \epsilon^T \epsilon = (y - XW)^T (y - XW)$$

Vecto ước lượng bình phương bé nhất $\widehat{W}$ mà làm L đạt giá trị nhỏ nhất khi:

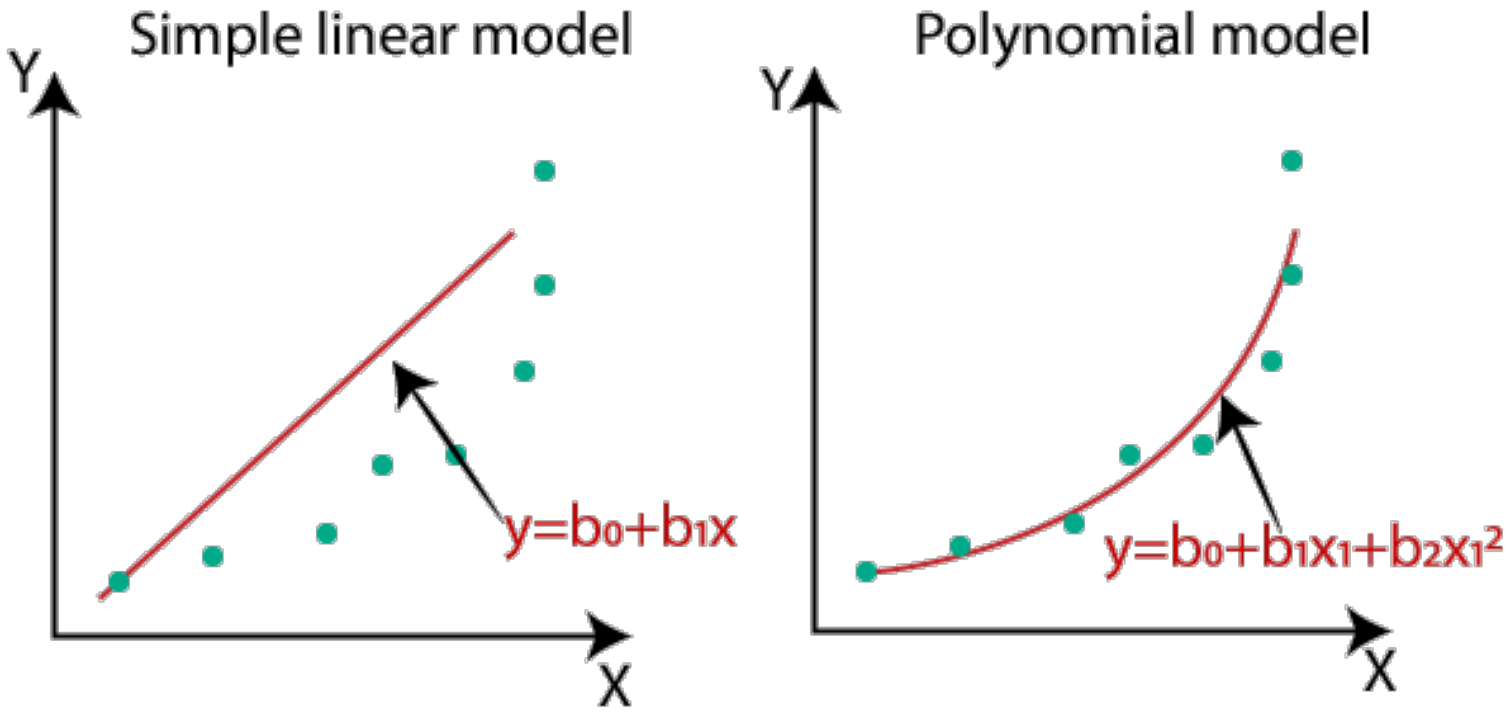
$$\frac{d(L,X,W)}{d(W)} = 0 \quad (2)$$

Giải (2) ta được $\widehat{W} = (x^T X)^{-1} X^T y$
Với $\widehat{W} = [w_0, w_1, .., w_n]^T$, mô hình hồi quy ước lượng có dạng:

$$\widehat{y_i} = \widehat{w_0} + \sum_1^k \widehat{W_j} X_{ij}$$

Biểu diễn dạng ma trận $\widehat{y} = X\widehat{W}$

Để minh họa cho đường $\widehat{y} = X\widehat{W}$.



Vecto sai số có dạng **e=y**-$\widehat{y}$

- Sai số trong mô hình hồi quy là $\epsilon \sim N(0,\sigma^2)$
- Ước lượng σ^2(MSE: Mean Squared Error):

$$\widehat{\sigma^2} = \frac{\sum_{i=1}^{n} e_i^2}{n-k-1} = \frac{SSE}{n-k-1} \qquad \text{(SSE: sum of Squared error)}$$

- E($\widehat{w}$)=w,$\widehat{w}$ là ước lượng không chật cho w.
- SST= SSR+SSE

(SST : $SumofSquaredTotal$, SSR : $SumofSquaredRegression$)

Với

$$\text{SST}=\sum_{i=1}^{n} (y_i - \bar{y})^2$$

và

$$\text{SSE}=\sum_{i=1}^{n} (y_i - \widehat{y})^2$$

suy ra

$$\text{SSR}=\sum_{i=1}^{n} (\widehat{y} - \bar{y})^2$$

Để đánh giá độ phù hợp trong Polynomal Regression ta sử dụng hệ số R-bình phương hay còn gọi là R-squared:

$$R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST}$$

Hệ số xác định hiệu chỉnh:

$$R^2 = 1 - \frac{SSE/(n-k-1)}{SST/(n-1)}$$

Sử dụng model Polynomial bằng Scikit-learn.

Các bước thực hiện:

1) Add a second degree polynomial term:

```
X = np.array([258.0, 270.0, 294.0,
              320.0, 342.0, 368.0,
              396.0, 446.0, 480.0, 586.0])\
             [:, np.newaxis]

y = np.array([236.4, 234.4, 252.8,
              298.6, 314.2, 342.2,
              360.8, 368.0, 391.2,
              390.8])
lr = LinearRegression()
pr = LinearRegression()
quadratic = PolynomialFeatures(degree=2)
X_quad = quadratic.fit_transform(X)
```

2. Fit a simple linear regression model for comparison:

```
# fit linear features
lr.fit(X, y)
X_fit = np.arange(250, 600, 10)[:, np.newaxis]
y_lin_fit = lr.predict(X_fit)
```

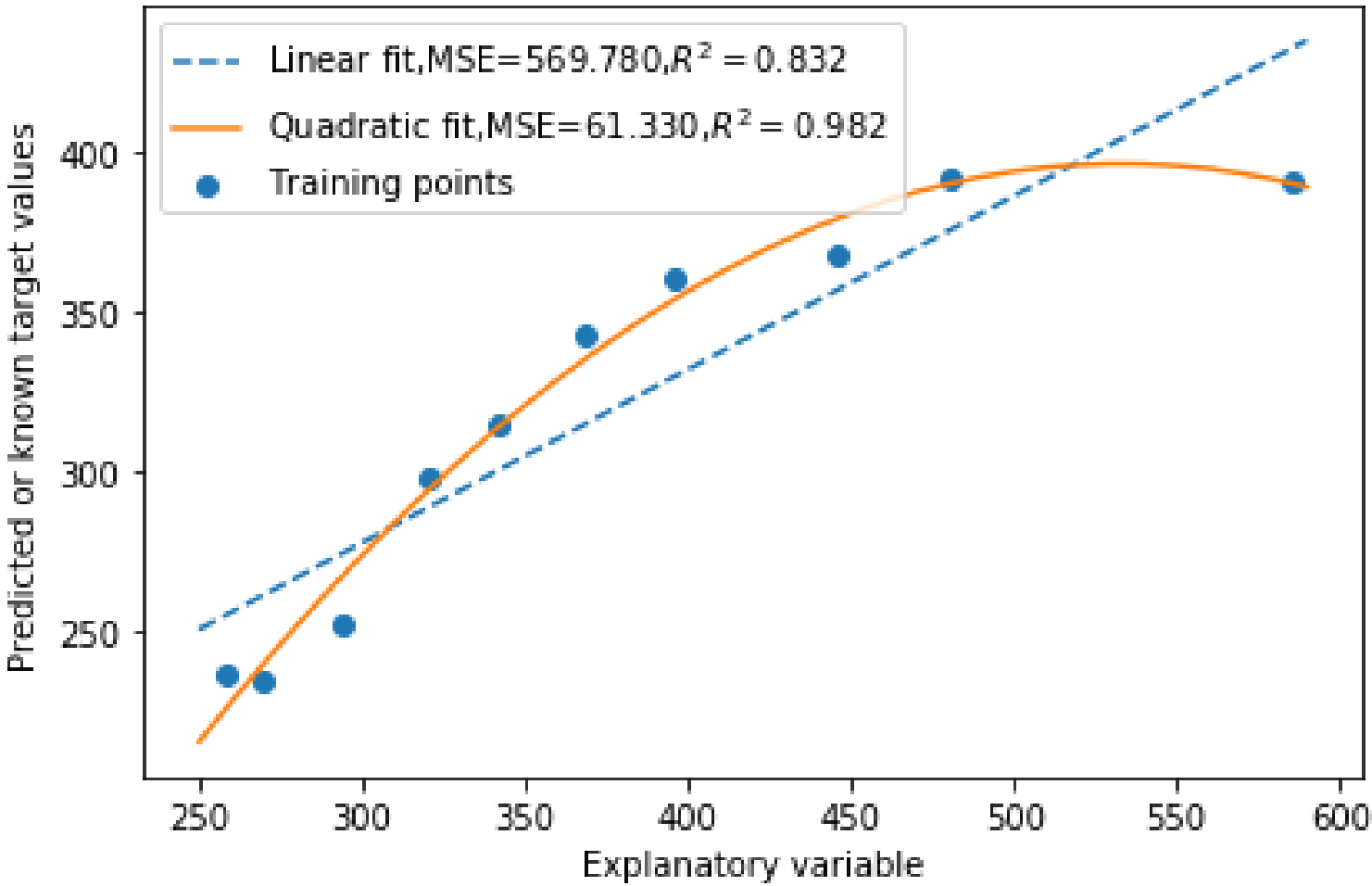
3. Fit a multiple regression model on the transformed features for polynomial regression:

```
# fit quadratic features
pr.fit(X_quad, y)
y_quad_fit = pr.predict(quadratic.fit_transform(X_fit))
```

4. Plot the results:

```
# plot results
plt.scatter(X, y, label='Training points')
plt.plot(X_fit, y_lin_fit, label='Linear
     fit,MSE=%.3f,$R^2=%.3f$'%(linear_MSE, linear_r2), linestyle='--')
plt.plot(X_fit, y_quad_fit, label='Quadratic
     fit,MSE=%.3f,$R^2=%.3f$'%(quad_MSE, quad_r2))
plt.xlabel('Explanatory variable')
plt.ylabel('Predicted or known target values')
plt.legend(loc='upper left')

plt.tight_layout()
#plt.savefig('images/10_11.png', dpi=300)
plt.show()
```

1) Polynomial Regresion



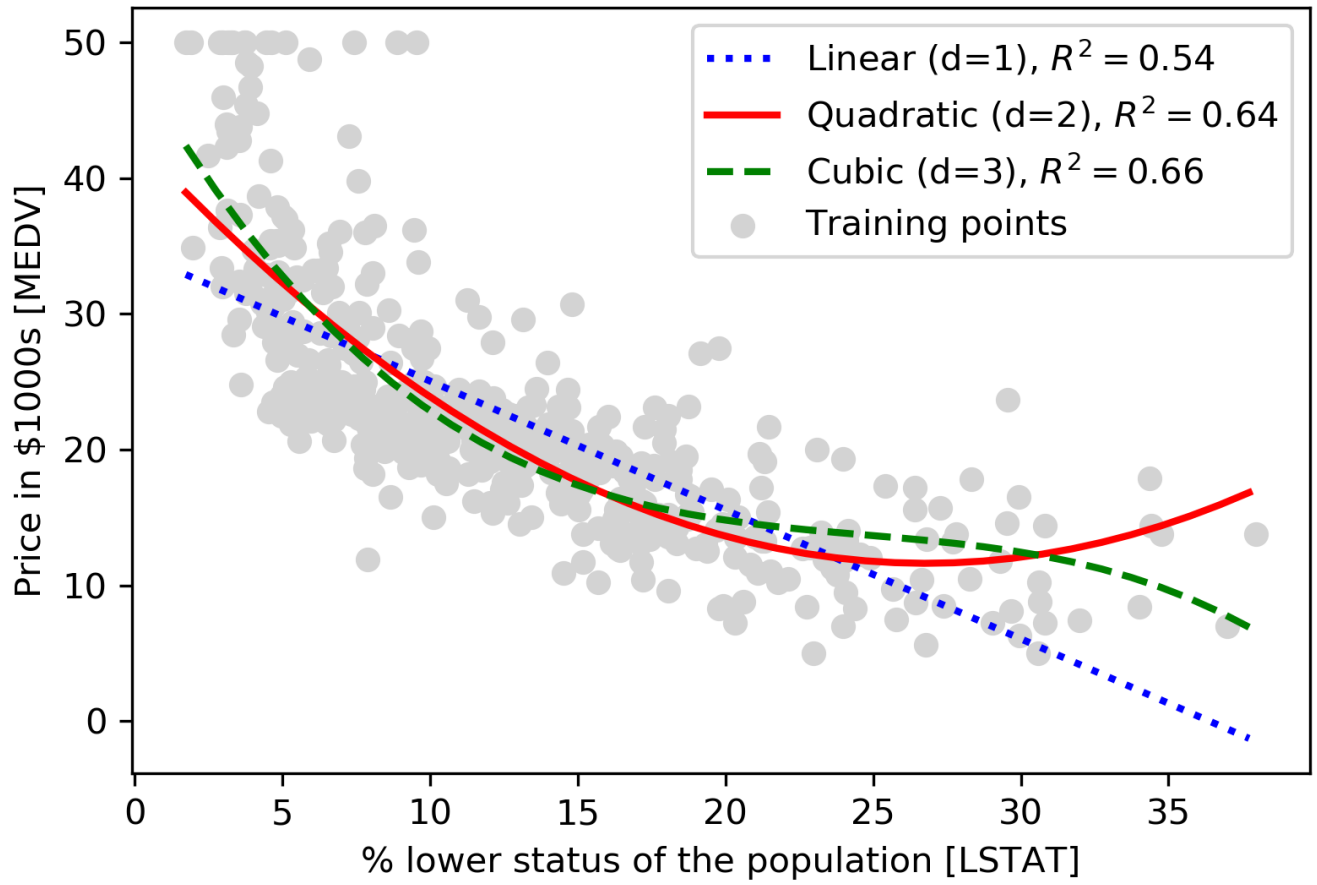
Modeling nonlinear relationships in the Housing dataset

Trong bộ dữ liệu Housing. Bằng cách thực thi đoạn code, chúng ta sẽ mô hình hóa mối quan hệ giữa giá nhà và LSTAT (trạng thái thấp hơn phần trăm của dân số) khi sử dụng hồi quy đa thức bậc hai, bậc ba và so sánh nó với sự phù hợp của hồi quy tuyến tính tuyến tính.

```
X = df[['LSTAT']].values
y = df['MEDV'].values

regr = LinearRegression()

# create quadratic features
quadratic = PolynomialFeatures(degree=2)
cubic = PolynomialFeatures(degree=3)
X_quad = quadratic.fit_transform(X)
X_cubic = cubic.fit_transform(X)

# fit features
X_fit = np.arange(X.min(), X.max(), 1)[:, np.newaxis]

regr = regr.fit(X, y)
y_lin_fit = regr.predict(X_fit)
linear_r2 = r2_score(y, regr.predict(X))

regr = regr.fit(X_quad, y)
y_quad_fit = regr.predict(quadratic.fit_transform(X_fit))
quadratic_r2 = r2_score(y, regr.predict(X_quad))
```

```
regr = regr.fit(X_cubic, y)
y_cubic_fit = regr.predict(cubic.fit_transform(X_fit))
cubic_r2 = r2_score(y, regr.predict(X_cubic))


# plot results
plt.scatter(X, y, label='Training points', color='lightgray')

plt.plot(X_fit, y_lin_fit,
         label='Linear (d=1), $R^2=%.2f$' % linear_r2,
         color='blue',
         lw=2,
         linestyle=':')

plt.plot(X_fit, y_quad_fit,
         label='Quadratic (d=2), $R^2=%.2f$' % quadratic_r2,
         color='red',
         lw=2,
         linestyle='-')

plt.plot(X_fit, y_cubic_fit,
         label='Cubic (d=3), $R^2=%.2f$' % cubic_r2,
         color='green',
         lw=2,
         linestyle='--')

plt.xlabel('% lower status of the population [LSTAT]')
```

```
plt.ylabel('Price in $1000s [MEDV]')
plt.legend(loc='upper right')

#plt.savefig('images/10_12.png', dpi=300)
plt.show()
```



Sau đây là một số hàm dùng trong thuật toán Polynomial regresion

```
   # Univariate Polynomial Regression

class PolynomailRegression() :
    def __init__(self, degree, learning_rate, iterations) :
        self.degree = degree
        self.learning_rate = learning_rate
        self.iterations = iterations
    # function to tranform X
    def transform(self, X) :
        # initialize X_transform
        X_transform = np.ones((self.m, 1))
        j = 0
        for j in range(self.degree + 1) :
            if j != 0 :
                x_pow = np.power( X, j )
                # append x_pow to X_transform
                X_transform = np.append(X_transform, x_pow.reshape(-1,
                     1), axis = 1)
        return X_transform
    # function to normalize X_tranform
    def normalize(self, X) :
        X[:, 1:] = (X[:, 1:] - np.mean(X[:, 1:], axis = 0 )) /
             np.std(X[:, 1:], axis = 0)
        return X
    # model training
```

```
def fit(self, X, Y) :
    self.X = X
    self.Y = Y
    self.m, self.n = self.X.shape
    # weight initialization
    self.W = np.zeros(self.degree + 1)
    # tranform X for polynomial h( x ) = w0 * x^0 + w1 * x^1 + w2 *
        x^2 + ........+ wn * x^n
    X_transform = self.transform(self.X)
    # normalize X_transform
    X_normalize = self.normalize(X_transform)
    # gradient descent learning
    for i in range(self.iterations) :
        h = self.predict( self.X )
        error = h - self.Y
        # update weights
        self.W = self.W - self.learning_rate * (1 / self.m) *
            np.dot(X_normalize.T, error)
    return self
# predict
def predict(self, X) :
    # tranform X for polynomial h( x ) = w0 * x^0 + w1 * x^1 + w2 *
        x^2 + ........+ wn * x^n
    X_transform = self.transform(X)
    X_normalize = self.normalize(X_transform)
    return np.dot(X_transform, self.W)
```

3) Ví dụ Polynomial Regression

Ví dụ: Có một công ty Nhân sự sắp tuyển dụng một ứng viên mới. Ứng viên đã nói với mức lương trước đây của anh ta là 160K mỗi năm, và nhân sự phải kiểm tra xem anh ta nói thật hay nói dối. Vì vậy, để xác định điều này, họ chỉ có một tập dữ liệu về công ty trước đây của anh ấy, trong đó mức lương của 10 position hàng đầu được đề cập với các level tương ứng. Bằng cách kiểm tra tập dữ liệu có sẵn, chúng tôi nhận thấy rằng có một mối quan hệ phi tuyến tính giữa các cấp Vị trí và mức lương. Mục tiêu là phải xây dựng một model *Bluffing detector regression*, để bộ phận nhân sự có thể thuê một ứng viên trung thực.

Từ đề bài chúng ta xây dựng 1 model Polynomial Regression gồm những bước sau:

- Data Pre-processing
- Build a Linear Regression model and fit it to the dataset
- Build a Polynomial Regression model and fit it to the dataset
- Visualize the result for Linear Regression and Polynomial Regression model.
- Predicting the output.

3) Ví dụ Polynomial Regression

•Data Pre-processing

Bước Data Pre-processiong sẽ vẫn giống như trong các model Regression , ngoại trừ một số thay đổi. Trong Polynomial Regression model, sẽ không sử dụng tính năng chia tỷ lệ và không chia tập dữ liệu của mình thành tập để training và test. Vì có hai lý do:

- Tập dữ liệu chứa rất ít thông tin không phù hợp để chia thành tập kiểm tra và tranning, nếu không model sẽ không thể tìm thấy mối tương quan giữa mức lương và Level.
- Trong model này, để dự đoán rất chính xác về mức lương thì model phải có đủ thông tin. Code bước Data pre-processing được thể hiện bên dưới:

```
# Importing libraries
# -------------------
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
# Importing the dataset
# ---------------------
dataset =
     pd.read_csv('/home/minhvu/Desktop/Py4DS/Position_Salaries.csv')
X = dataset.iloc[:, 1:2].values
y = dataset.iloc[:, 2].values
```

Position	Level(X-variable)	Salary(Y-Variable)
Business Analyst	1	45000
Junior Consultant	2	50000
Senior Consultant	3	60000
Manager	4	80000
Country Manager	5	110000
Region Manager	6	150000
Partner	7	200000
Senior Partner	8	300000
C-level	9	500000
CEO	10	1000000

3) Ví dụ Polynomial Regression

Như chúng ta có thể thấy trong đầu ra ở trên, có ba cột hiện diện (Position, Level và Salary). Nhưng chúng ta chỉ đang xét hai cột vì Position tương đương với các Level. Ở đây có thể dự đoán được mức level là 6.5 vì ứng viên đã có hơn 4 năm kinh nghiệm làm Region Manager, do đó thì ứng viên này phải ở giữa level 6 và 7.

- Building the Linear regression model

```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X, y)
y_lin_pred = lin_reg.predict(X)
```

- Building the Polynomial regression mode

```
# Fitting Polynomial Regression(degree=2) to the dataset
from sklearn.preprocessing import PolynomialFeatures
poly_2_reg = PolynomialFeatures(degree=2)
X_poly = poly_2_reg.fit_transform(X)
lin_reg_2 = LinearRegression()
lin_reg_2.fit(X_poly, y)
```

3) Ví dụ Polynomial Regression

Lệnh *poly.fit_tranform(X)* để đổi ma trận ban đầu thành một ma trận đặc trưng của đa thức và sau đó điều chỉnh nó với model Polynomial Regression giá trị tham số(the parameter value) với bậc đa thức bằng 2(ax^2+bx+c)

	0	1	2
0	1	1	1
1	1	2	4
2	1	3	9
3	1	4	16
4	1	5	25
5	1	6	36
6	1	7	49
7	1	8	64
8	1	9	81
9	1	10	100

Sau khi code, chúng ta sẽ nhận được một ma trận x_poly, để chọn và kiểm tra các biến sao cho hợp lí.

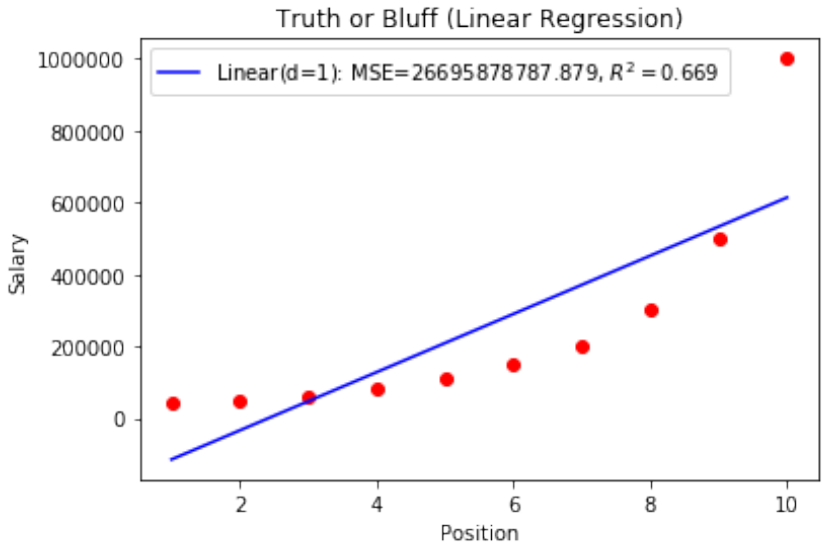
• Visualize the result for Linear Regression and Polynomial Regression model.

1) Visualizing the result for Linear regression

Đầu tiên chúng ta sẽ trực quan hóa kết quả bằng Linear regression model

```
# Visualising the Linear Regression results
plt.scatter(X, y, color='red')
plt.plot(X,
        lin_reg.predict(X),
        label='Linear(d=1): MSE=%.3f,
             $R^2=%.3f$'%(linear_MSE,linear_r2),
        color='blue')
plt.title('Truth or Bluff (Linear Regression)')
plt.legend(loc='upper left')
plt.xlabel('Position')
plt.ylabel('Salary')
plt.show()
```

3) Ví dụ Polynomial Regression



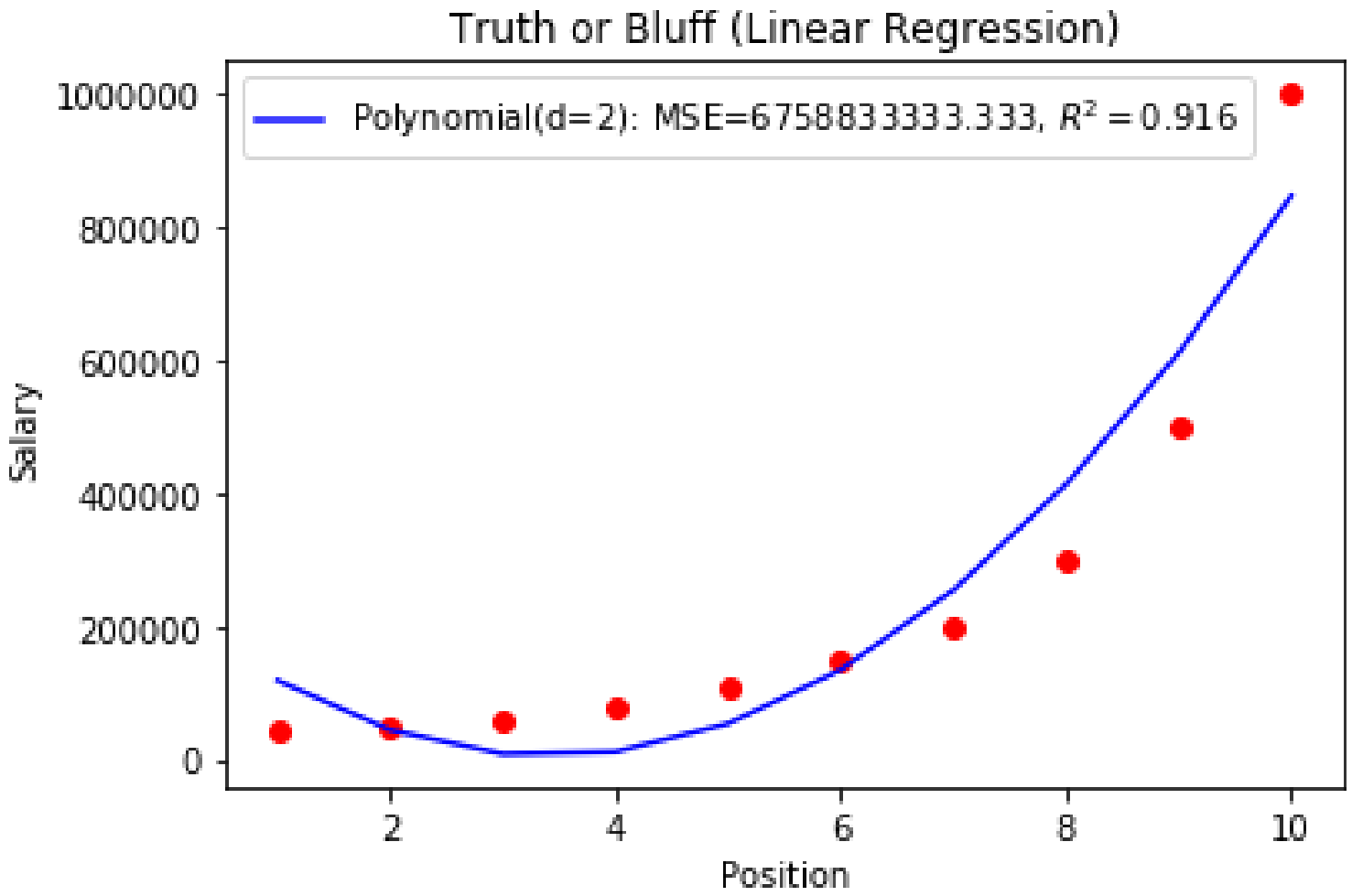
Trong ảnh trên chúng ta thấy được đường Regression(màu đỏ) và các giá trị thực tế(màu xanh). Nếu chúng ta coi đầu ra này để dự đoán giá trị của CEO, thì nó sẽ nằm ở mức lương 600k, nằm rất xa với giá trị thực. Vì thế cần một model khác để dự đoán.

2) Visualizing the result for Polynomial Regression Tiếp theo chúng ta sử dụng kết quả Polynomial regression để dự đoán. Sau đây là đoạn code của nó.

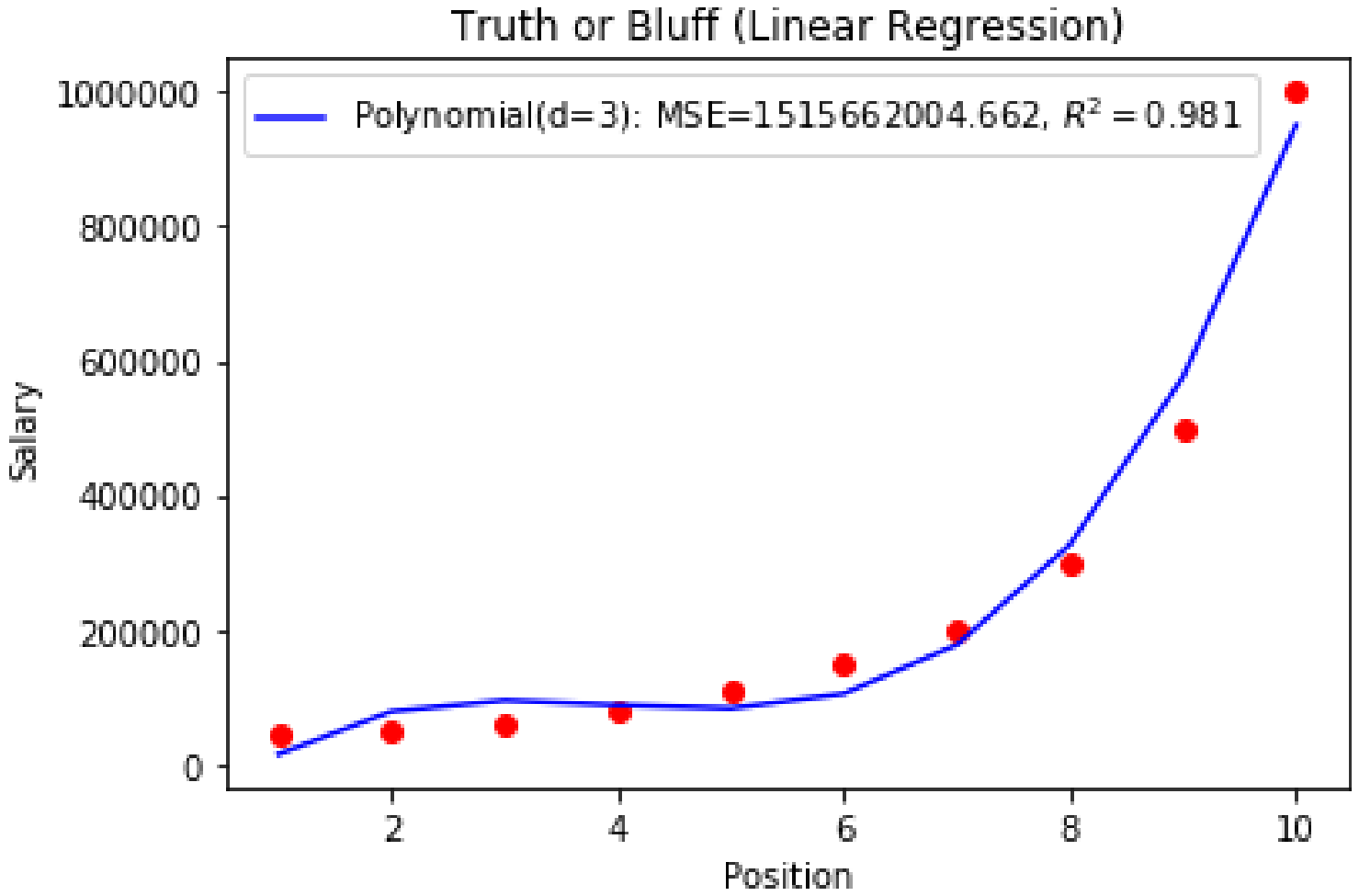
```
# Visualising the Polynomial Regression results
plt.scatter(X, y, color='red')
plt.plot(X,
         lin_reg_2.predict(poly_2_reg.fit_transform(X)),
         label='Polynomial(d=2): MSE=%.3f,
              $R^2=%.3f$'%(poly_2_MSE,poly_2_r2),
         color='blue')
plt.title('Truth or Bluff (Linear Regression)')
plt.legend(loc='upper left')
plt.xlabel('Position')
plt.ylabel('Salary')
plt.show()
```

Trong đoạn code trên, ta dùng *lin_reg.fit_transform(X)*, thay vì x_poly. Bởi vì chúng ta sẽ dùng Linear regressor để dự đoán một ma trận đặc trưng đa thức.

3) Ví dụ Polynomial Regression

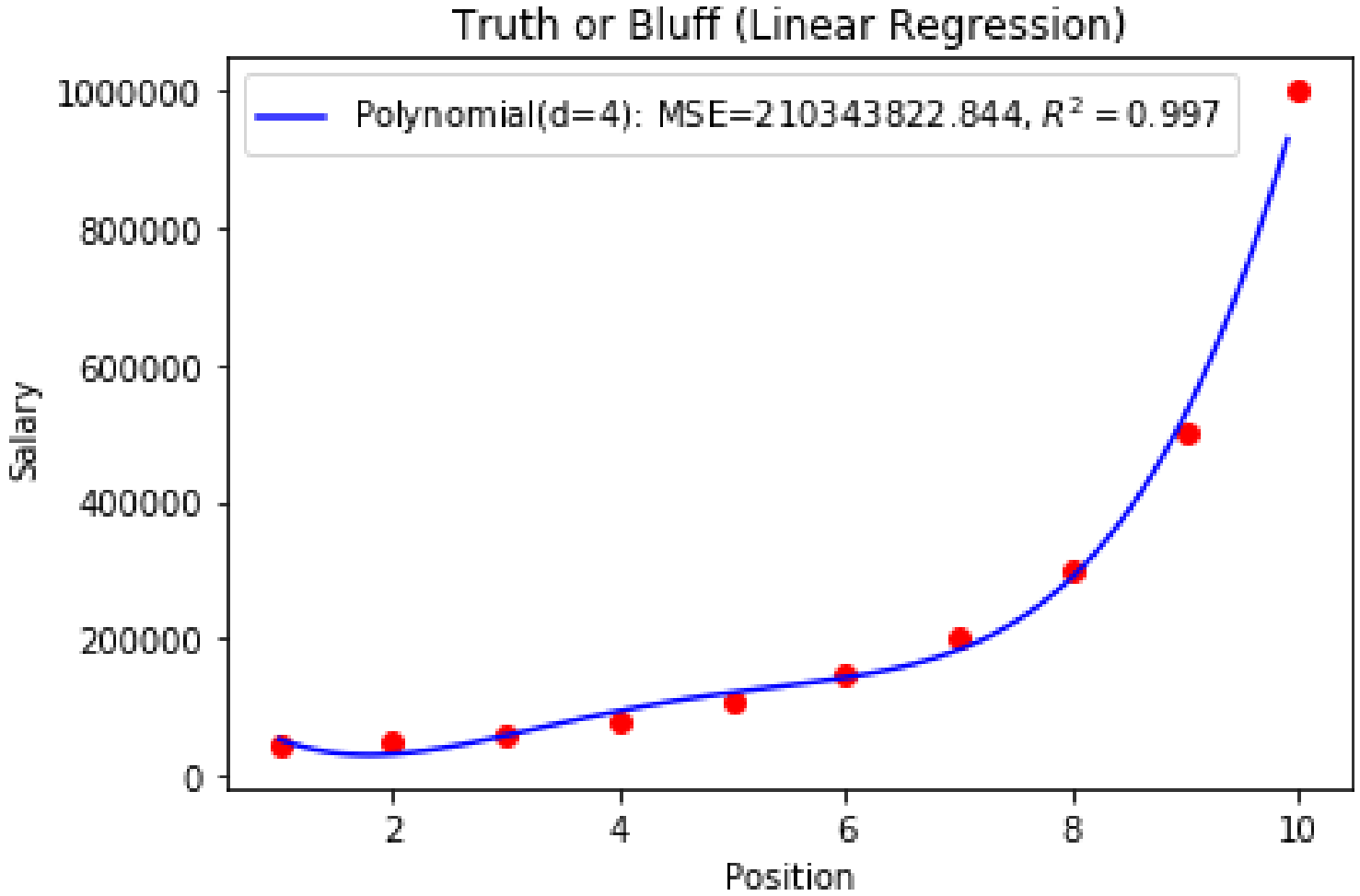


Nếu thay đổidegree=3 (bậc đa thức bằng 3), thì chúng ta sẽ có được một đường hồi quy chính xác hơn.



3) Ví dụ Polynomial Regression

Nếu tăng lên degree=4 thì càng chính xác hơn nữa. Do đó ta có thể nhận được kết quả chính xác hơn bằng cách tăng mức độ bậc của đa thức(tăng degree)



Bây giờ, chúng ta sẽ dự đoán đầu ra cuối cùng bằng cách sử dụng mô hình Polynomial Regression. Dưới đây là đoạn code.

```
poly_4_pred = lin_reg_4.predict(poly_4_reg.fit_transform([[6.5]]))
print(poly_4_pred)
```

Out:

```
[158862.45265153]
```

Chúng ta có thể thấy, đầu ra dự đoán cho Polynomial Regression là [158862.45265153], gần với giá trị thực, do đó chúng ta có thể nói rằng ứng viên này đang nói thật.