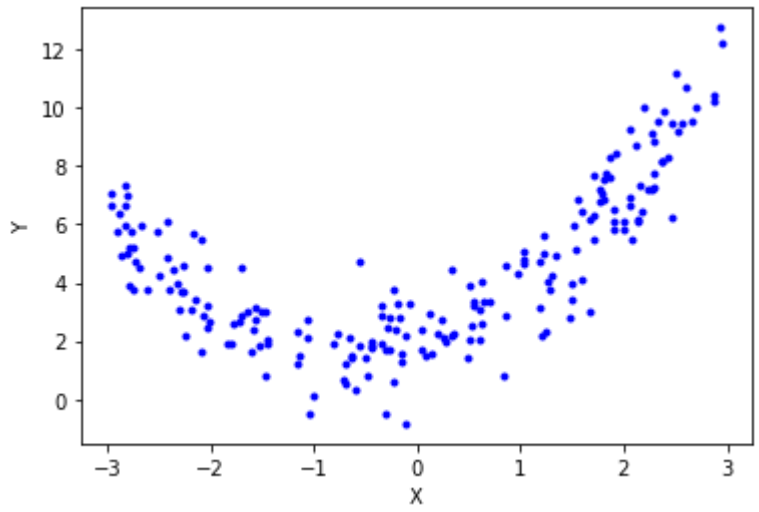
Polynomial Regression with One Variable

Step-1) import all the libraries

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import r2_score
```

Step-2) Create and visualize the data

In [3]:
```
X = 6 * np.random.rand(200, 1) - 3
y = 0.8 * X**2 + 0.9*X + 2 + np.random.randn(200, 1)
#equation used -> y = 0.8x^2 + 0.9x + 2
#visualize the data
plt.plot(X, y, 'b.')
plt.xlabel("X")
plt.ylabel("Y")
plt.show()
```



Step-3) split data in train and test set

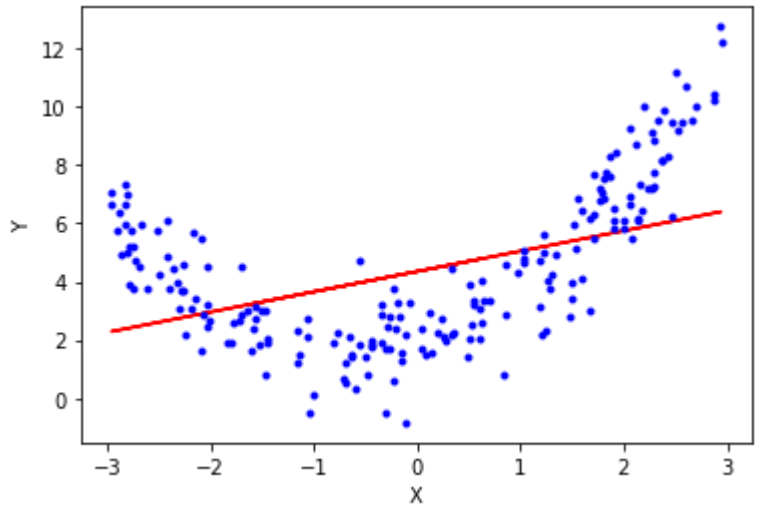
In [4]:
```
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_s
```

Step-4) Apply simple linear regression

In [6]:
```
lr = LinearRegression()
lr.fit(x_train, y_train)
y_pred = lr.predict(x_test)
print(r2_score(y_test, y_pred))
```

```
0.3084959616830809
```

In [7]:
```
plt.plot(x_train, lr.predict(x_train), color="r")
plt.plot(X, y, "b.")
plt.xlabel("X")
plt.ylabel("Y")
plt.show()
```



Step-5) Apply Polynomial Regression

Now we will convert the input to polynomial terms by using the degree as 2 because of the equation we have used, the intercept is 2. while dealing with real-world problems we choose degree by heat and trial method.

In [8]:
```
#applying polynomial regression degree 2
poly = PolynomialFeatures(degree=2, include_bias=True)
x_train_trans = poly.fit_transform(x_train)
x_test_trans = poly.transform(x_test)
#include bias parameter
lr = LinearRegression()
lr.fit(x_train_trans, y_train)
y_pred = lr.predict(x_test_trans)
print(r2_score(y_test, y_pred))
```

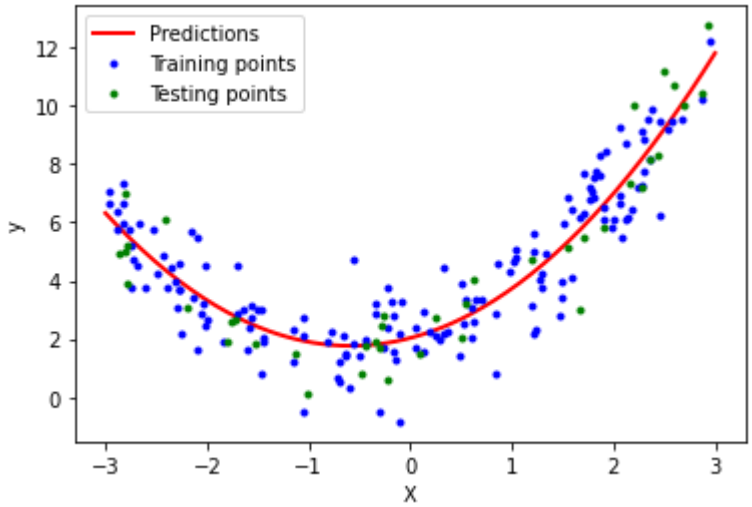
```
0.9000705296001988
```

In [ ]:
```
After converting to polynomial terms we fit the linear regression which is now wo
If you print x_train value and train transformed value you will see the 3 polynom
performing descent well and if you see the coefficients and intercept value. our
it has predicted 0.88 and intercept was 2 and it has given 1.9 which is very clos
can be said as a generalized model.
```

In [9]:
```
print(lr.coef_)
print(lr.intercept_)
```

```
[[0.         0.91428165 0.78049026]]
[2.03218397]
```

In [10]:
```
X_new = np.linspace(-3, 3, 200).reshape(200, 1)
X_new_poly = poly.transform(X_new)
y_new = lr.predict(X_new_poly)
plt.plot(X_new, y_new, "r-", linewidth=2, label="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()
```



In [ ]:
```
```