In [25]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
```

In [26]:
```
m = 100 # we want 100 data samples
X = 6*np.random.rand(m,1)-3  # generate a random array of shape (m,
y = 0.5*X**2 + X + 2 + np.random.randn(m,1)  # equation that gives
```

In [30]:
```
X[:5]
```

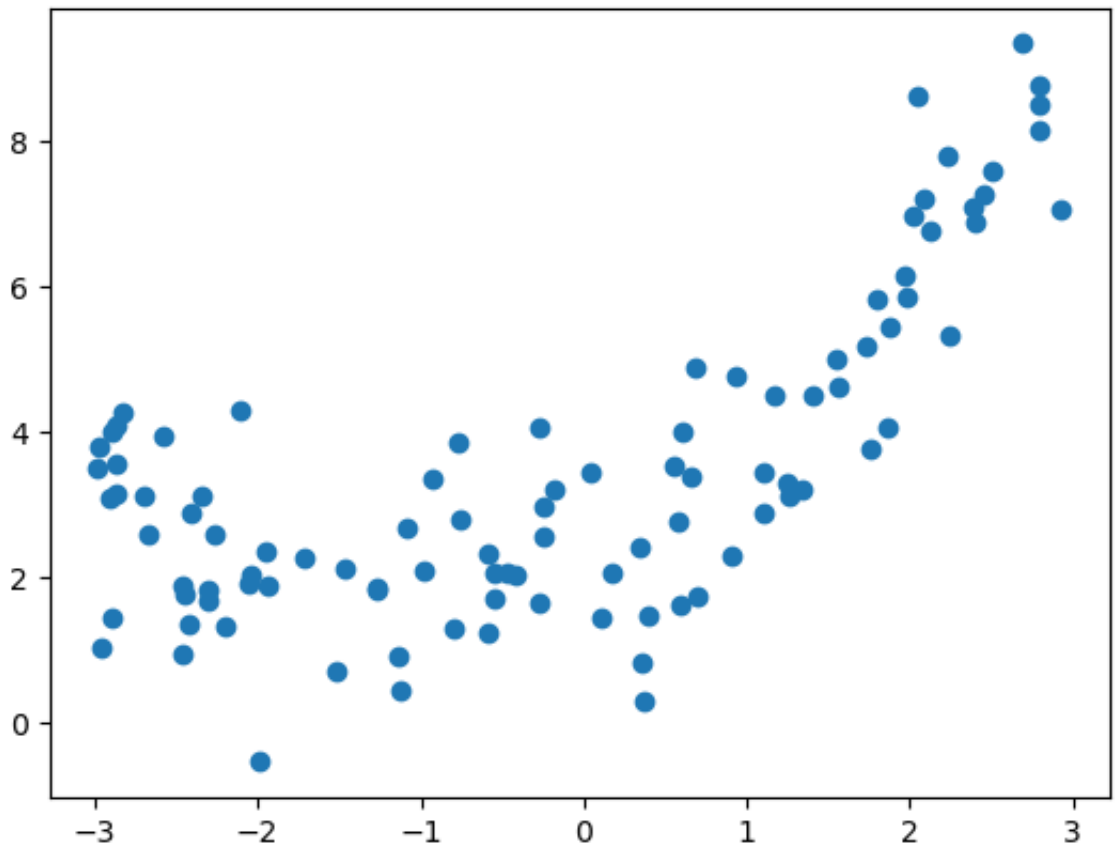
Out[30]:
```
array([[ 1.79505446],
       [ 0.58085429],
       [-0.26711278],
       [-2.96579082],
       [ 1.334207  ]])
```

In [31]:
```
y[:5]
```

Out[31]:
```
array([[5.81029918],
       [2.75675936],
       [1.64010361],
       [3.79117639],
       [3.19608564]])
```

In [32]:
```
plt.scatter(X,y)
```

Out[32]:
```
<matplotlib.collections.PathCollection at 0x139679390>
```



In [33]:
```
# now we try to build a regression model to simulate this non linea
# the original equation

poly_features = PolynomialFeatures(degree=2, include_bias = False)
X_poly = poly_features.fit_transform(X)
```

In [34]:
```
X_poly[0]  # 2 features : x, x^2.
```

Out[34]:
```
array([1.79505446, 3.22222051])
```

In [35]:
```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
```

In [36]:
```
lin_reg.fit(X_poly,y)
```

Out[36]:
```
▾ LinearRegression

LinearRegression()
```

In [38]:
```
lin_reg.intercept_, lin_reg.coef_ # parameters of our new model hyp
```

Out[38]:
```
(array([2.23949271]), array([[0.97139219, 0.43411957]]))
```

In [77]:
```
predictions = lin_reg.predict(X_poly)
predictions[:5]
```

Out[77]:
```
array([[5.38202357],
       [2.95019838],
       [2.01099555],
       [3.17702555],
       [4.30831083]])
```

FINAL ANS COMPARISON

In [79]:
```
y[:5]
```

Out[79]:
```
array([[5.81029918],
       [2.75675936],
       [1.64010361],
       [3.79117639],
       [3.19608564]])
```

In [80]:
```
predictions[:5]
```

Out[80]:
```
array([[5.38202357],
       [2.95019838],
       [2.01099555],
       [3.17702555],
       [4.30831083]])
```