

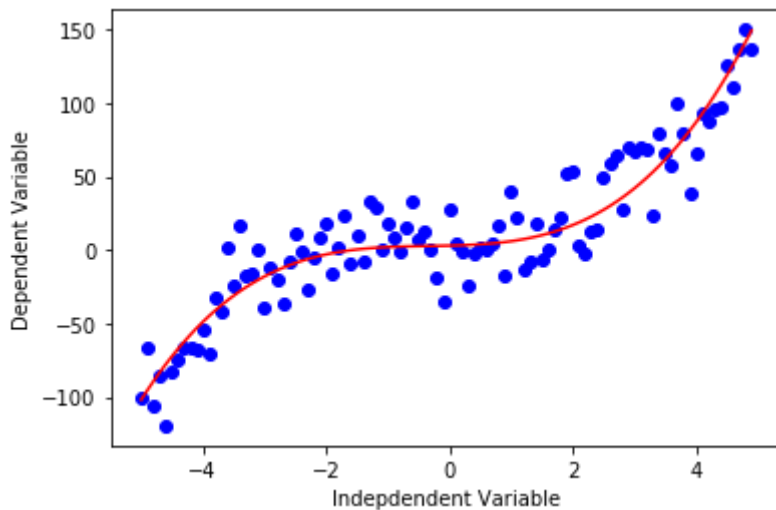




In [3]:

```
x = np.arange(-5.0, 5.0, 0.1)

##You can adjust the slope and intercept to verify the changes in the graph
y = 1*(x**3) + 1*(x**2) + 1*x + 3
y_noise = 20 * np.random.normal(size=x.size)
ydata = y + y_noise
plt.plot(x, ydata, 'bo')
plt.plot(x, y, 'r')
plt.ylabel('Dependent Variable')
plt.xlabel('Independent Variable')
plt.show()
```



As you can see, this function has  $x^3$  and  $x^2$  as independent variables. Also, the graphic of this function is not a straight line over the 2D plane. So this is a non-linear function.

Some other types of non-linear functions are:

## Quadratic

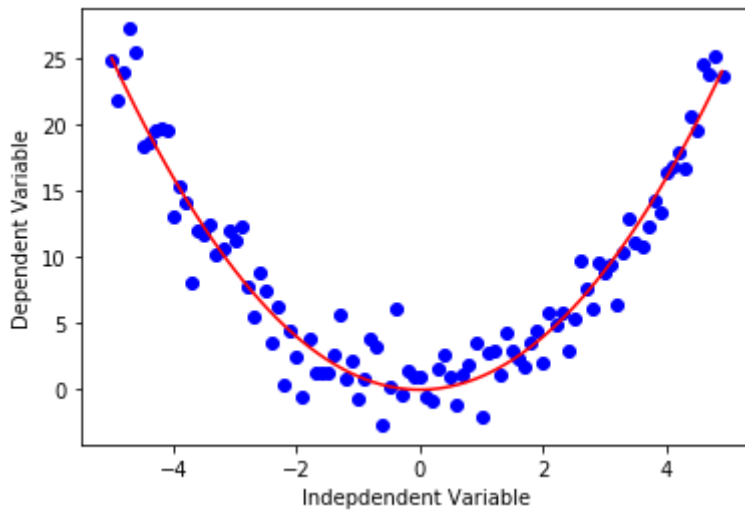
$$Y = X^2$$

In [4]:

```
x = np.arange(-5.0, 5.0, 0.1)

##You can adjust the slope and intercept to verify the changes in the graph

y = np.power(x,2)
y_noise = 2 * np.random.normal(size=x.size)
ydata = y + y_noise
plt.plot(x, ydata, 'bo')
plt.plot(x,y, 'r')
plt.ylabel('Dependent Variable')
plt.xlabel('Independent Variable')
plt.show()
```



## Exponential

An exponential function with base  $c$  is defined by  $Y = a + b c^X$  where  $b \neq 0$ ,  $c > 0$ ,  $c \neq 1$ , and  $x$  is any real number. The base,  $c$ , is constant and the exponent,  $x$ , is a variable.

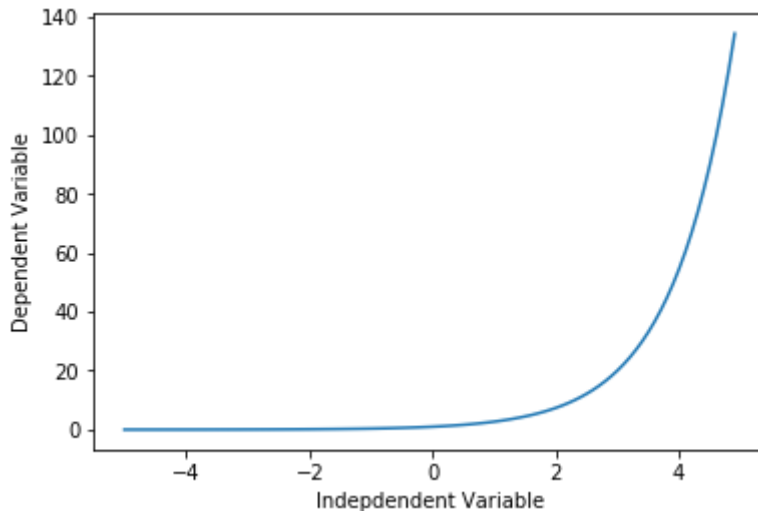
In [5]:

```
X = np.arange(-5.0, 5.0, 0.1)

##You can adjust the slope and intercept to verify the changes in the graph

Y= np.exp(X)

plt.plot(X,Y)
plt.ylabel('Dependent Variable')
plt.xlabel('Independent Variable')
plt.show()
```



## Logarithmic

The response  $y$  is a results of applying logarithmic map from input  $x$ 's to output variable  $y$ . It is one of the simplest form of **log()**: i.e.  $y = \log(x)$

Please consider that instead of  $x$ , we can use  $X$ , which can be polynomial representation of the  $x$ 's. In general form it would be written as

$$y = \log(X)$$

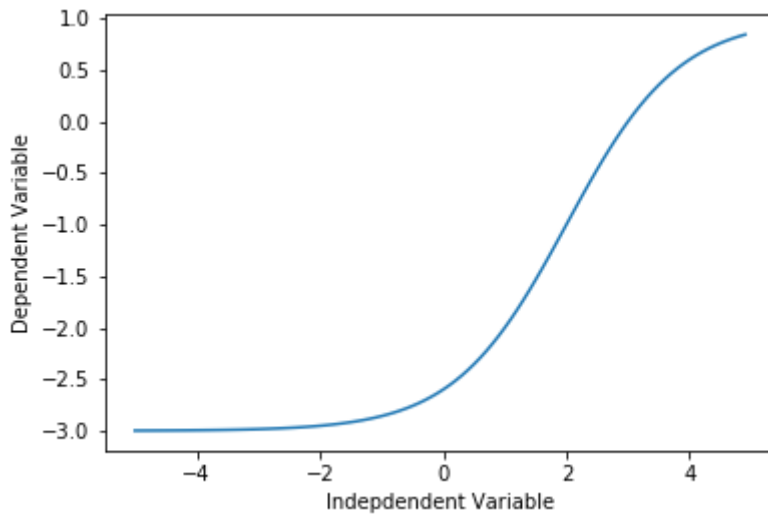


In [7]:

```
X = np.arange(-5.0, 5.0, 0.1)

Y = 1-4/(1+np.power(3, X-2))

plt.plot(X,Y)
plt.ylabel('Dependent Variable')
plt.xlabel('Independent Variable')
plt.show()
```



## Non-Linear Regression example

For an example, we're going to try and fit a non-linear model to the datapoints corresponding to China's GDP from 1960 to 2014. We download a dataset with two columns, the first, a year between 1960 and 2014, the second, China's corresponding annual gross domestic income in US dollars for that year.

In [8]:

```
import numpy as np
import pandas as pd

#downloading dataset
!wget -nv -O china_gdp.csv https://s3-api.us-gio.objectstorage.softlayer.net/cf-cou
rses-data/CognitiveClass/ML0101ENV3/labs/china_gdp.csv

df = pd.read_csv("china_gdp.csv")
df.head(10)
```

```
2020-01-30 19:25:08 URL:https://s3-api.us-gio.objectstorage.softlayer.n
et/cf-courses-data/CognitiveClass/ML0101ENV3/labs/china_gdp.csv [1218/1
218] -> "china_gdp.csv" [1]
```

Out[8]:

	Year	Value
0	1960	5.918412e+10
1	1961	4.955705e+10
2	1962	4.668518e+10
3	1963	5.009730e+10
4	1964	5.906225e+10
5	1965	6.970915e+10
6	1966	7.587943e+10
7	1967	7.205703e+10
8	1968	6.999350e+10
9	1969	7.871882e+10

**Did you know?** When it comes to Machine Learning, you will likely be working with large datasets. As a business, where can you host your data? IBM is offering a unique opportunity for businesses, with 10 Tb of IBM Cloud Object Storage: [Sign up now for free \(http://cocl.us/ML0101EN-IBM-Offer-CC\)](http://cocl.us/ML0101EN-IBM-Offer-CC).

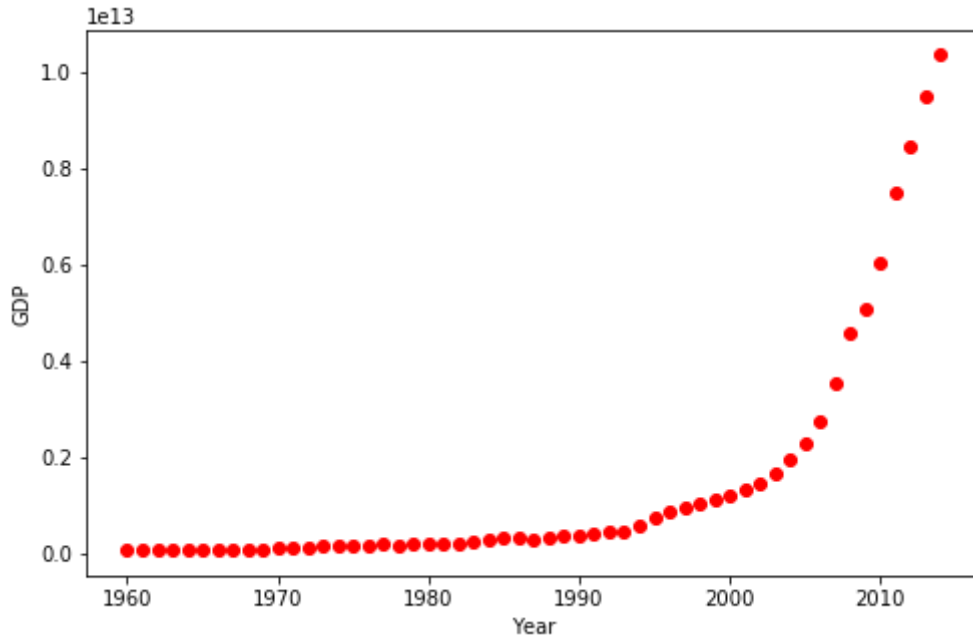
## Plotting the Dataset

This is what the datapoints look like. It kind of looks like an either logistic or exponential function. The growth starts off slow, then from 2005 on forward, the growth is very significant. And finally, it decelerate slightly in the 2010s.



In [9]:

```
plt.figure(figsize=(8,5))
x_data, y_data = (df["Year"].values, df["Value"].values)
plt.plot(x_data, y_data, 'ro')
plt.ylabel('GDP')
plt.xlabel('Year')
plt.show()
```



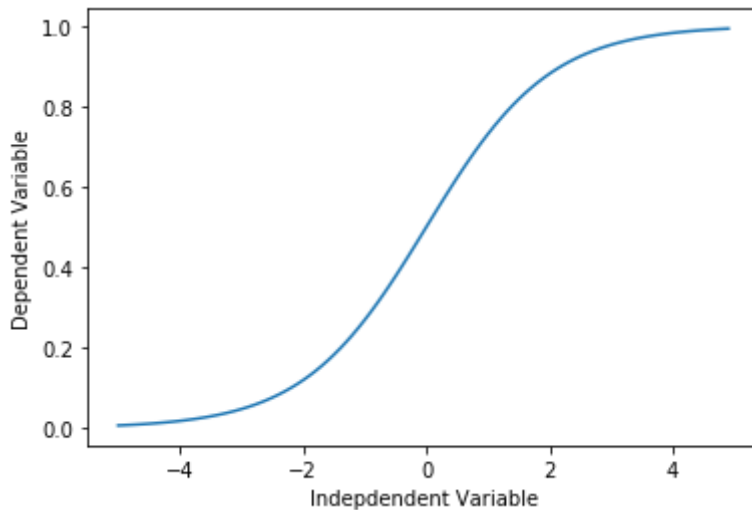
## Choosing a model

From an initial look at the plot, we determine that the logistic function could be a good approximation, since it has the property of starting with a slow growth, increasing growth in the middle, and then decreasing again at the end; as illustrated below:

In [10]:

```
X = np.arange(-5.0, 5.0, 0.1)
Y = 1.0 / (1.0 + np.exp(-X))

plt.plot(X,Y)
plt.ylabel('Dependent Variable')
plt.xlabel('Independent Variable')
plt.show()
```



The formula for the logistic function is the following:

$$\hat{Y} = \frac{1}{1 + e^{\beta_1(X - \beta_2)}}$$

$\beta_1$ : Controls the curve's steepness,

$\beta_2$ : Slides the curve on the x-axis.

## Building The Model

Now, let's build our regression model and initialize its parameters.

In [11]:

```
def sigmoid(x, Beta_1, Beta_2):
    y = 1 / (1 + np.exp(-Beta_1*(x-Beta_2)))
    return y
```





In [19]:

```
# split data into train/test
msk = np.random.rand(len(df)) < 0.8
train_x = xdata[msk]
test_x = xdata[~msk]
train_y = ydata[msk]
test_y = ydata[~msk]

# build the model using train set
popt, pcov = curve_fit(sigmoid, train_x, train_y)

# predict using test set
y_hat = sigmoid(test_x, *popt)

# evaluation
print("Mean absolute error: %.2f" % np.mean(np.absolute(y_hat - test_y)))
print("Residual sum of squares (MSE): %.2f" % np.mean((y_hat - test_y) ** 2))
from sklearn.metrics import r2_score
print("R2-score: %.2f" % r2_score(y_hat , test_y) )
```

Mean absolute error: 0.03  
Residual sum of squares (MSE): 0.00  
R2-score: 0.89

Double-click **here** for the solution.

## Want to learn more?

IBM SPSS Modeler is a comprehensive analytics platform that has many machine learning algorithms. It has been designed to bring predictive intelligence to decisions made by individuals, by groups, by systems – by your enterprise as a whole. A free trial is available through this course, available here: [SPSS Modeler \(http://cocl.us/ML0101EN-SPSSModeler\)](http://cocl.us/ML0101EN-SPSSModeler)

Also, you can use Watson Studio to run these notebooks faster with bigger datasets. Watson Studio is IBM's leading cloud solution for data scientists, built by data scientists. With Jupyter notebooks, RStudio, Apache Spark and popular libraries pre-packaged in the cloud, Watson Studio enables data scientists to collaborate on their projects without having to install anything. Join the fast-growing community of Watson Studio users today with a free account at [Watson Studio \(https://cocl.us/ML0101EN\\_DSX\)](https://cocl.us/ML0101EN_DSX)

## Thanks for completing this lesson!

**Author:** [Saeed Aghabozorgi \(https://ca.linkedin.com/in/saeedaghabozorgi\)](https://ca.linkedin.com/in/saeedaghabozorgi)

[Saeed Aghabozorgi \(https://ca.linkedin.com/in/saeedaghabozorgi\)](https://ca.linkedin.com/in/saeedaghabozorgi), PhD is a Data Scientist in IBM with a track record of developing enterprise level applications that substantially increases clients' ability to turn data into actionable knowledge. He is a researcher in data mining field and expert in developing advanced analytic methods like machine learning and statistical modelling on large datasets.

---

Copyright © 2018 [Cognitive Class \(https://cocl.us/DX0108EN\\_CC\)](https://cocl.us/DX0108EN_CC). This notebook and its source code are released under the terms of the [MIT License \(https://bigdatauniversity.com/mit-license/\)](https://bigdatauniversity.com/mit-license/).