RunProblem

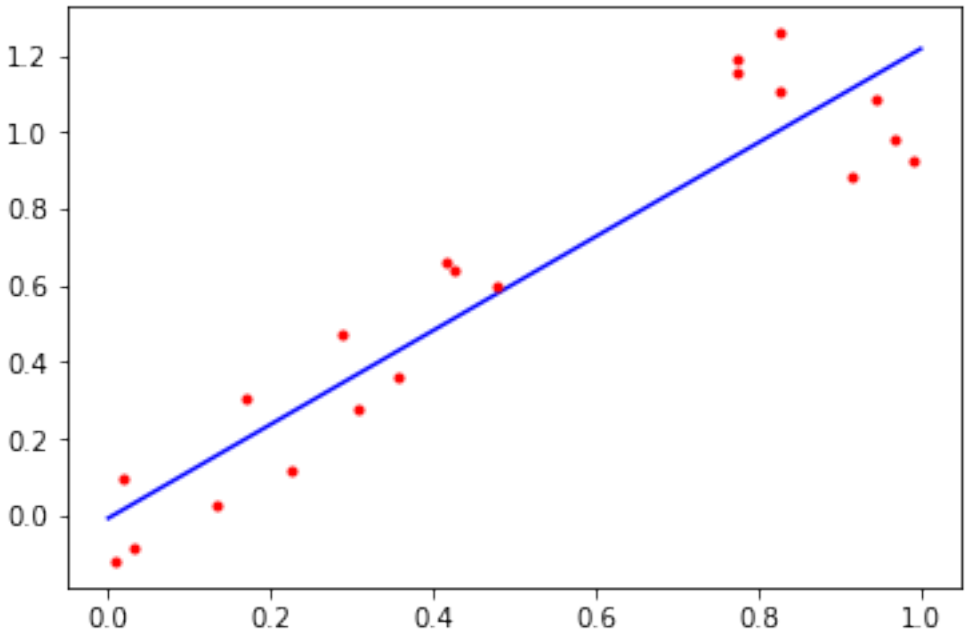
January 19, 2019

In [69]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas

# Function that creates the X matrix as defined for fitting our model
def create_X(x,deg):
    X = np.ones((len(x),deg+1))
    for i in range(1,deg+1):
        X[:,i] = x**i
    return X

# Function for predicting the response
def predict_y(x,beta):
    return np.dot(create_X(x,len(beta)-1),beta)

# Function for fitting the model
def fit_beta(df,deg):
    return np.linalg.lstsq(create_X(df.x,deg),df.y,rcond=None)[0]

# Function for computing the MSE
def mse(y,yPred):
    return np.mean((y-yPred)**2)

# Loading training, validation and test data
dfTrain = pandas.read_csv('Data_Train.csv')
dfVal = pandas.read_csv('Data_Val.csv')
dfTest = pandas.read_csv('Data_Test.csv')

############ TRAINING A MODEL

# Fitting model
deg = 1
X = create_X(dfTrain.x,deg)
beta = fit_beta(dfTrain,deg)

# Computing training error
yPredTrain = predict_y(dfTrain.x,beta)
err = mse(dfTrain.y,yPredTrain)
```

```
        print('Training Error = {:2.3}'.format(err))

        # Computing test error
        yPredTest = predict_y(dfTest.x,beta)
        err = mse(dfTest.y,yPredTest)
        print('Test Error = {:2.3}'.format(err))
```

```
Training Error = 0.0258
Test Error = 0.0154
```

In [71]:
```
############ PLOTTING FITTED MODEL

        # Plotting by taking 100 equally spaced points between 0 and 1
        x = np.linspace(0,1,100)
        y = predict_y(x,beta)
        plt.plot(x,y,'b-',dfTrain.x,dfTrain.y,'r.')
        plt.show()

        # # ## Plotting with original given data
        # y = predict_y(dfTrain.x,beta)
        # plt.plot(dfTrain.x,y,'b-',dfTrain.x,dfTrain.y,'r.')
        # plt.show()
```



In [36]:
```
############ HYPER-PARAMETER TUNING
```

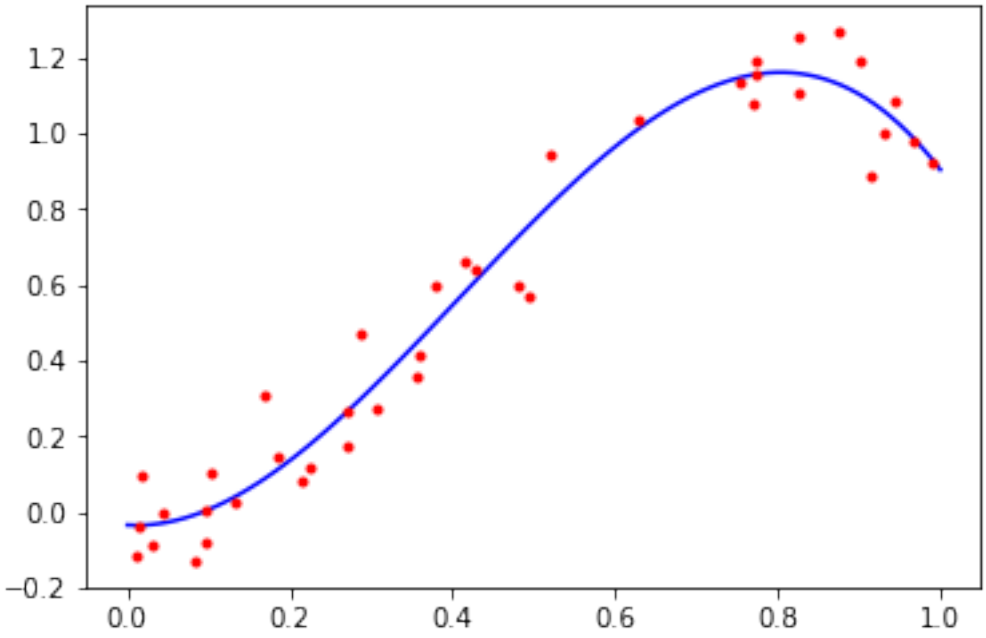
```
# Initializing range of degree values to be tested and errors
degRange = list(range(1,11))
errTrain = np.zeros(len(degRange))
errVal = np.zeros(len(degRange))

# Computing error as a function of degree
for deg in degRange:
    X = create_X(dfTrain.x,deg)
    beta = fit_beta(dfTrain,deg)

    # Computing training error
    yPredTrain = predict_y(dfTrain.x,beta)
    err = mse(dfTrain.y,yPredTrain)
    errTrain[deg-1] = err

    # Computing test error
    yPredVal = predict_y(dfVal.x,beta)
    err = mse(dfVal.y,yPredVal)
    errVal[deg-1] = err

# Plotting training and validation errors
plt.plot(degRange,errTrain,'b-',degRange,errVal,'r-')
plt.legend(('Training Error','Validation Error'))
plt.show()
```

In [73]: ############ TRAINING SELECTED MODEL

```
        # Concatenating data training and validation data frames
        frames = [dfTrain, dfVal]
        df = pandas.concat(frames)

        # Fit model using the optimal degree found in the previous cell
        # Based on the graph, the optimal degree found is 3
        degOpt = 3
        beta = fit_beta(df,degOpt)

        # Compute and print training and test errors
        # Computing training error
        yPredTrain = predict_y(df.x,beta)
        err = mse(df.y,yPredTrain)
        print('Train Error = {:2.3}'.format(err))

        # Computing test error
        yPredTest = predict_y(dfTest.x,beta)
        err = mse(dfTest.y,yPredTest)
        print('Test Error = {:2.3}'.format(err))
```

```
Train Error = 0.0087
Test Error = 0.0108
```

In [74]:
```
        ############ PLOTTING FITTED MODEL
        # # Plot the fitted model as in the second cell

        # ## Plotting with original given data
        # df["yPredicted"] = yPredTrain
        # print(df)
        # sorted_df = df.sort_values(by=["x"])
        # print(sorted_df)
        # plt.plot(sorted_df.x,sorted_df.yPredicted,'b-',df.x,df.y,'r.')
        # plt.show()


        # Plotting by taking 100 equally spaced points between 0 and 1
        x = np.linspace(0,1,100)
        y = predict_y(x,beta)
        plt.plot(x,y,'b-',df.x,df.y,'r.')
        plt.show()
```


1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0