Homework 2 and 3

Homework 2 is due on Monday, February 3.
Homework 3 is due on Monday, February 10.

Written Part

Problem 1: Consider an $m \times n$ matrix X, an m-component vector y, and an n-component vector θ. Suppose the matrix X is tall and skinny, that is, suppose $m > n$. We saw in class the the gradient of the MSE cost function is proportional to $X^T X\theta - X^T y = X^T(X\theta - y)$. This quantity can be computed two different ways:

Way 1:

1. Compute the matrix product $X^T X$
2. Compute the matrix-vector product $(X^T X)\theta$
3. Compute the matrix-vector product $X^T y$
4. Subtract the vector $X^T y$ from the vector $(X^T X)\theta$

Way 2:

1. Compute the matrix-vector product $X\theta$
2. Subtract the vector y from the vector $X\theta$
3. Compute the matrix-vector product $X^T(X\theta - y)$

What would be the most efficient way in terms of operations (multiplications, additions, and subtractions) of computing it?

Answer 1

Way 2 would be the more efficient way to solve the MSE because it only consists of 2 matrix-vector multiplications and 1 vector subtraction.

Problem 2 (for M462 students): Consider the function $f(x) = x^T Mx$, where x is a vector, and M is an $n \times n$ matrix (possibly non-symmetric). Find the gradient of $f(x)$.

$$\frac{\partial f}{\partial x_i} = [0 \cdots 010 \cdots 0] M x + x^T M \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$\frac{\partial f}{\partial x_i} = [\text{row i of M}] x + x^T \begin{bmatrix} \text{col} \\ i \\ \text{of} \\ M \end{bmatrix}$$

$$\frac{\partial f}{\partial x_i} = M x + M^T x$$

Problem 2 (for M562 students): Consider two convex functions $f(x)$ and $g(x)$. Assume $g(x)$ is non-decresing. Show that the composite function $h(x) = g(f(x))$ is also convex.

Problem 3: Consider the MSE cost function $\mathrm{MSE}(\theta) = \|y - X\theta\|_2^2$. Find the second-order partial derivatives matrix (the *Hessian matrix*)

$$\begin{bmatrix} \frac{\partial^2 \mathrm{MSE}(\theta)}{\partial \theta_1^2} & \frac{\partial^2 \mathrm{MSE}(\theta)}{\partial \theta_1 \partial \theta_2} & \cdots & \frac{\partial^2 \mathrm{MSE}(\theta)}{\partial \theta_1 \partial \theta_n} \\ \frac{\partial^2 \mathrm{MSE}(\theta)}{\partial \theta_2 \partial \theta_1} & \frac{\partial^2 \mathrm{MSE}(\theta)}{\partial \theta_2^2} & \cdots & \frac{\partial^2 \mathrm{MSE}(\theta)}{\partial \theta_2 \partial \theta_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 \mathrm{MSE}(\theta)}{\partial \theta_n \partial \theta_1} & \frac{\partial^2 \mathrm{MSE}(\theta)}{\partial \theta_n \partial \theta_2} & \cdots & \frac{\partial^2 \mathrm{MSE}(\theta)}{\partial \theta_n^2} \end{bmatrix}$$

$$\frac{2}{m}(X^T(X\theta_i - y)(X\theta_j - y))$$

$$\begin{bmatrix} \frac{2}{m}(X^T(X\theta_1 - y)(X\theta_1 - y)) & \frac{2}{m}(X^T(X\theta_1 - y)(X\theta_2 - y)) & \cdots & \frac{2}{m}(X^T(X\theta_1 - y)(X\theta_n - y)) \\ \frac{2}{m}(X^T(X\theta_2 - y)(X\theta_1 - y)) & \frac{2}{m}(X^T(X\theta_2 - y)(X\theta_2 - y)) & \cdots & \frac{2}{m}(X^T(X\theta_2 - y)(X\theta_n - y)) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{2}{m}(X^T(X\theta_n - y)(X\theta_1 - y)) & \frac{2}{m}(X^T(X\theta_n - y)(X\theta_2 - y)) & \cdots & \frac{2}{m}(X^T(X\theta_n - y)(X\theta_n - y)) \end{bmatrix}$$

No Clue if ^^^ is correct

Programming Part

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

The Iris Dataset

The *iris* dataset contains a bunch of measurements for 150 flowers representing three species of iris (setosa, versicolor and virginica). For each flower, we have its petal length, petal width, sepal length, and sepal width, as well as its species.

In [2]:
```
url = 'https://raw.githubusercontent.com/um-perez-alvaro/lin-regress/master/ir
is.data'
iris_data = pd.read_csv(url, names=['sepal length','sepal width','petal lengt
h','petal width','species'])
iris_data.head(5) #first 5 rows
```

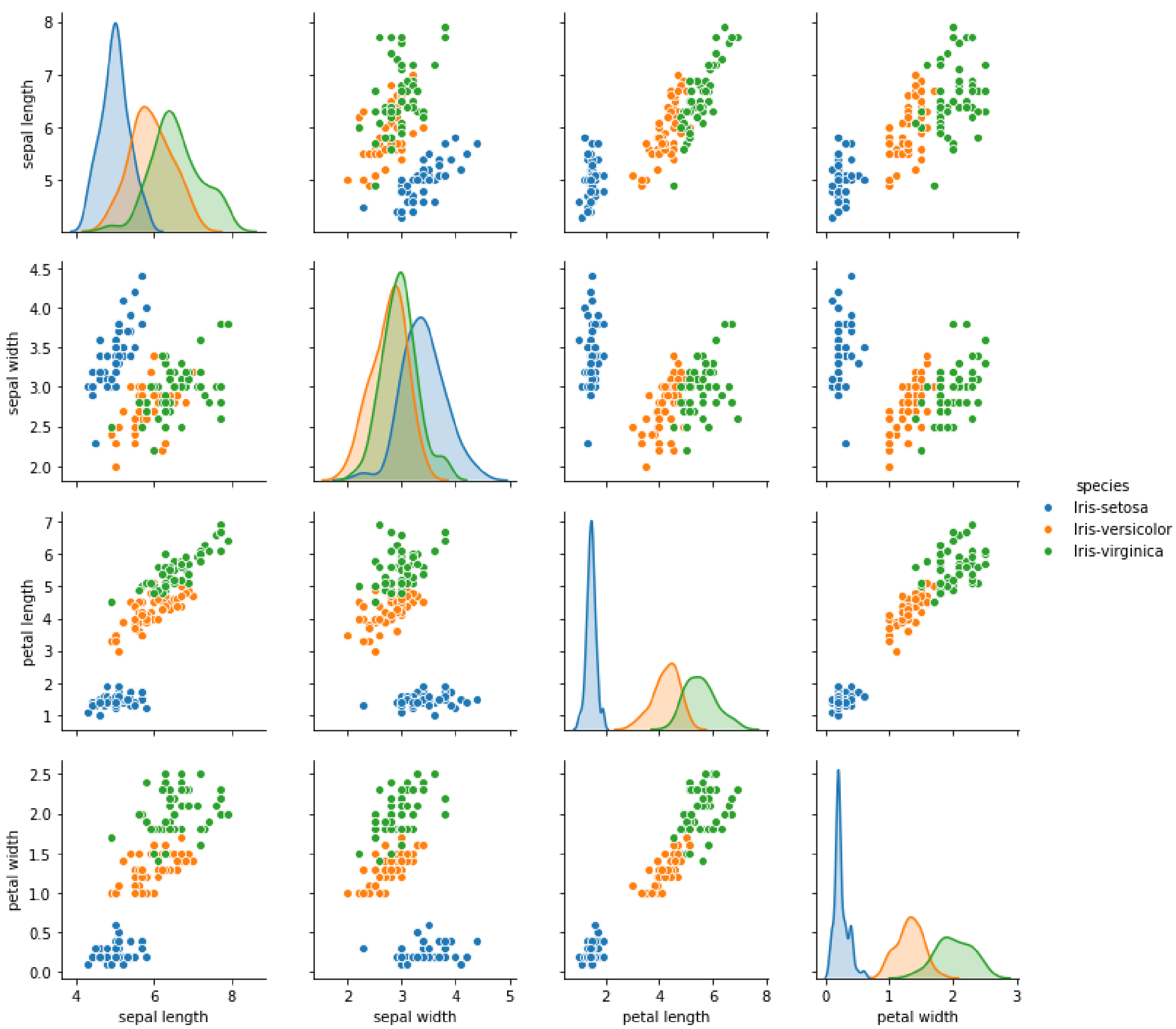
Out[2]:

	sepal length	sepal width	petal length	petal width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Let us look at the scatterplots for each of the six pairs of measurements.

In [3]:
```
sns.pairplot(data=iris_data,hue='species')
```

Out[3]: <seaborn.axisgrid.PairGrid at 0x196d000a488>



Assignments

Part 1: Add to the iris_data dataframe a new column called *target*. For each flower, set

$$\text{target} = \begin{cases} 1 & \text{if species} = \text{setosa}, \\ 0 & \text{if species} \neq \text{setosa}. \end{cases}$$

Hint: the easiest way to do is by using pandas.Series.map (see https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.map.html (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.map.html))

In [4]:
```
rule = {'Iris-setosa':1, 'Iris-versicolor':0, 'Iris-virginica':0}
iris_data['target'] = iris_data['species'].map(rule)
iris_data
```

Out[4]:

	sepal length	sepal width	petal length	petal width	species	target
0	5.1	3.5	1.4	0.2	Iris-setosa	1
1	4.9	3.0	1.4	0.2	Iris-setosa	1
2	4.7	3.2	1.3	0.2	Iris-setosa	1
3	4.6	3.1	1.5	0.2	Iris-setosa	1
4	5.0	3.6	1.4	0.2	Iris-setosa	1
...	...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica	0
146	6.3	2.5	5.0	1.9	Iris-virginica	0
147	6.5	3.0	5.2	2.0	Iris-virginica	0
148	6.2	3.4	5.4	2.3	Iris-virginica	0
149	5.9	3.0	5.1	1.8	Iris-virginica	0

150 rows × 6 columns

Part 2: Use Gradient Descent to train a linear model for predicting the target values.

Gradient descent is used after part 3.

In [9]:
```
y = iris_data['target'].to_numpy()
features = ['sepal length', 'sepal width', 'petal length', 'petal width']
x = iris_data[features].to_numpy()
x = iris_data.iloc[:,0:4]
x.head(5)
X = np.c_[np.ones(len(x)),x]
```

Part 3: Write a function for predicting whether the species of an iris flower is setosa or non-setosa. Your function must use the linear model from part 2 and follow the classification rule:

1. if the predicted target value is larger than or equal to 0.5, then the species is setosa.
2. if the predicted target value is less than 0.5, then the species is not setosa.

How many non-setosa iris flowers are correctly classified as non-setosa?

all of them.

How many non-setosa iris flowers are misclassified as setosa?

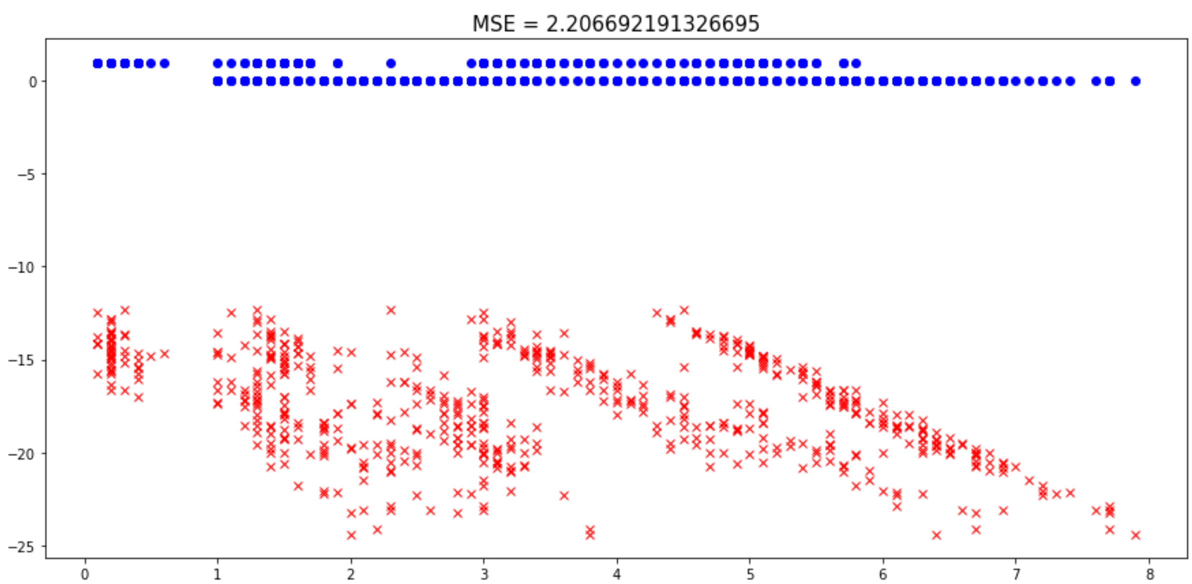
none.

In [266]:
```
def check_predicted(X,theta, y, setToLow, setToHigh, cutOff):
    y_pre = X@theta
    y_pre_post = y_pre.copy()
    y_a = 0
    y_b = 0
    for i in range(len(y_pre)):
        if y_pre[i][0] >= cutOff:
            y_pre_post[i][0] = setToHigh
        elif y_pre[i][0] < cutOff:
            y_pre_post[i][0] = setToLow
        if y_pre_post[i][0] == y[i]:
            if y[i] == 0:
                y_b = y_b + 1
            if y[i] == 1:
                y_a = y_a + 1
    return [y_pre, y_pre_post, str(y_a) + " were correctly 1\n" + str(y_b) + "
were correctly 0\n" + str((y_a + y_b) / len(y) * 100) + "% were correct "]
```

Gradient Descent

In [267]:
```
s = 0.01 #learning rate
```

In [281]:
```
'Initialization'
theta = np.random.randn(5,1)
y.shape = (150,1)

"MSE"
MSE = np.linalg.norm(X@theta-y)/m

"plot the data and the linear model"
plt.figure(figsize=(15,7))
plt.plot(x,y,'bo')

x_plot = np.linspace(0, 8, 150) #plot interpolation line

y_plot = theta[0][0]+theta[1][0]*x_plot+theta[2][0]*x_plot**2+theta[3][0]*x_pl
ot**3

y_pre = X@theta
plt.plot(x,y_pre,'rx')
plt.title('MSE = '+str(MSE),fontsize=15)

i = 0 # for the number of interation
```

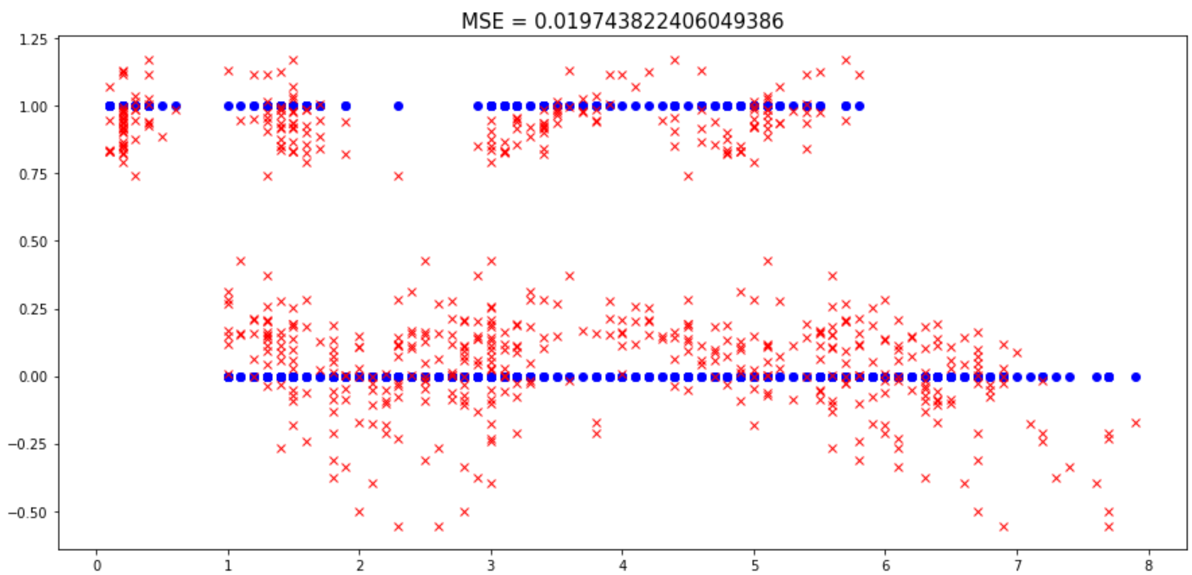


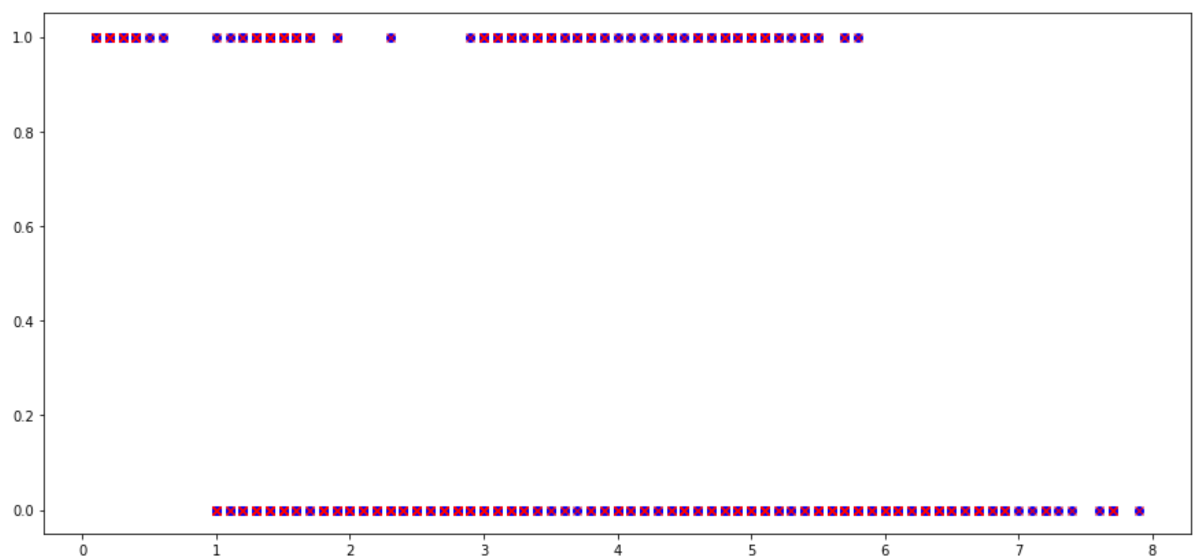
In [282]:
```
'Gradient Descent Step'
while i < 400: # number of iterations
    gradient = (2/m)*X.T@(X@theta-y)
    theta = theta - s*gradient
    i = i+1

"MSE"
MSE = np.linalg.norm(X@theta-y)/m

x_plot = np.linspace(0, 8, 150) #plot interpolation line
y_plot = theta[0][0]+theta[1][0]*x_plot+theta[2][0]*x_plot**2+theta[3][0]*x_pl
ot**3
```

In [283]:
```
ret = check_predicted(X,theta, y, 0, 1, 0.5)

print(ret[2])

plt.figure(figsize=(15,7))
plt.plot(x,y,'bo')
plt.plot(x,ret[0],'rx')
plt.title('MSE = '+str(MSE),fontsize=15)

plt.figure(figsize=(15,7))
plt.plot(x,y,'bo')
plt.plot(x,ret[1],'rx')
```

```
50 were correctly 1
100 were correctly 0
100.0% were correct
```

Out[283]:
```
[<matplotlib.lines.Line2D at 0x196f3e11fc8>,
 <matplotlib.lines.Line2D at 0x196f3c36708>,
 <matplotlib.lines.Line2D at 0x196f3e17048>,
 <matplotlib.lines.Line2D at 0x196f3e172c8>]
```





Stochastic Gradient Descent Function

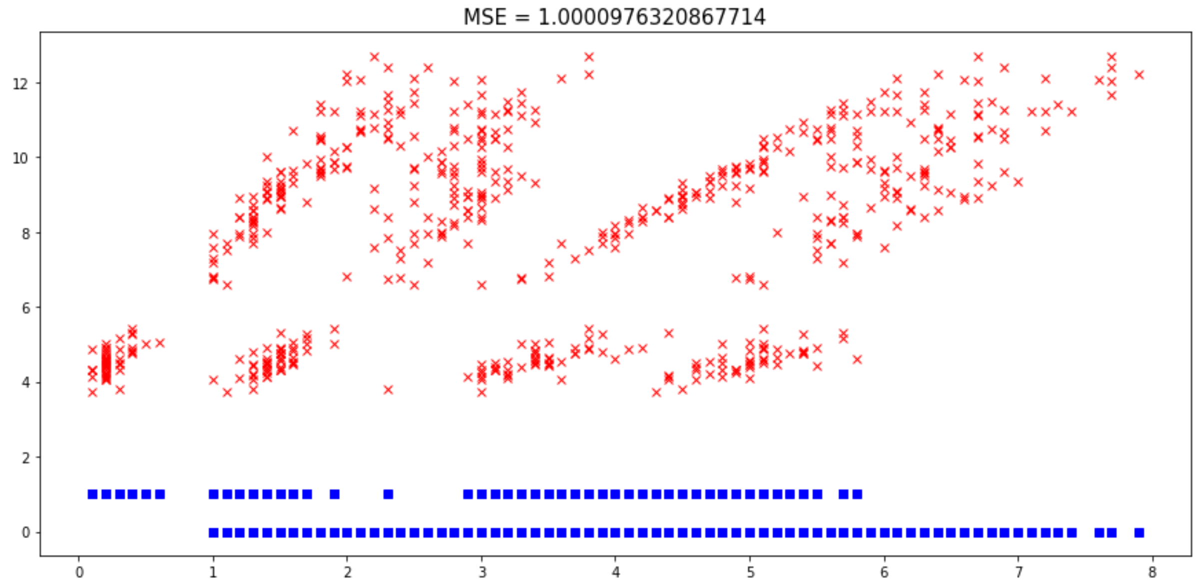
Also has a tone of errors, i can't fix them...

In [284]:
```
def linregression_SGD(X,y,n_epochs):
    m,n = X.shape
    theta = np.random.randn(n,1) #random initialization

    for epoch in range(n_epochs):
        for i in range(m):
            random_row = np.random.randint(m)
#             xi = X.iloc[random_row] #ith row;
#             xi = xi[None,:] #keep xi as a row vector
            xi = X[None,random_row] #keep xi as a row vector
            yi = y[random_row]
            gradient = 2*xi.T@(xi@theta-yi)
            s = learning_function(epoch*m+i)
            theta = theta - s*gradient
    return theta
```

In [285]:
```
#learning function hyperparameters
def learning_function(t):
    return s0/(t+s1)
```

In [286]:
```
'testing linregression_SGD'
n_epochs = 200
s0,s1 = 0,1
theta = linregression_SGD(X,y,n_epochs)

"MSE"
MSE = np.linalg.norm(X@theta-y)/m

plt.figure(figsize=(15,7))
plt.plot(x,y,'bs')

x_plot = np.linspace(0, 8, 150) #plot interpolation line
y_plot = theta[0]+theta[1]*x_plot+theta[2]*x_plot**2+theta[3]*x_plot**3

y_pre = X@theta
plt.plot(x,y_pre,'rx')
plt.title('MSE = '+str(MSE),fontsize=15)
```

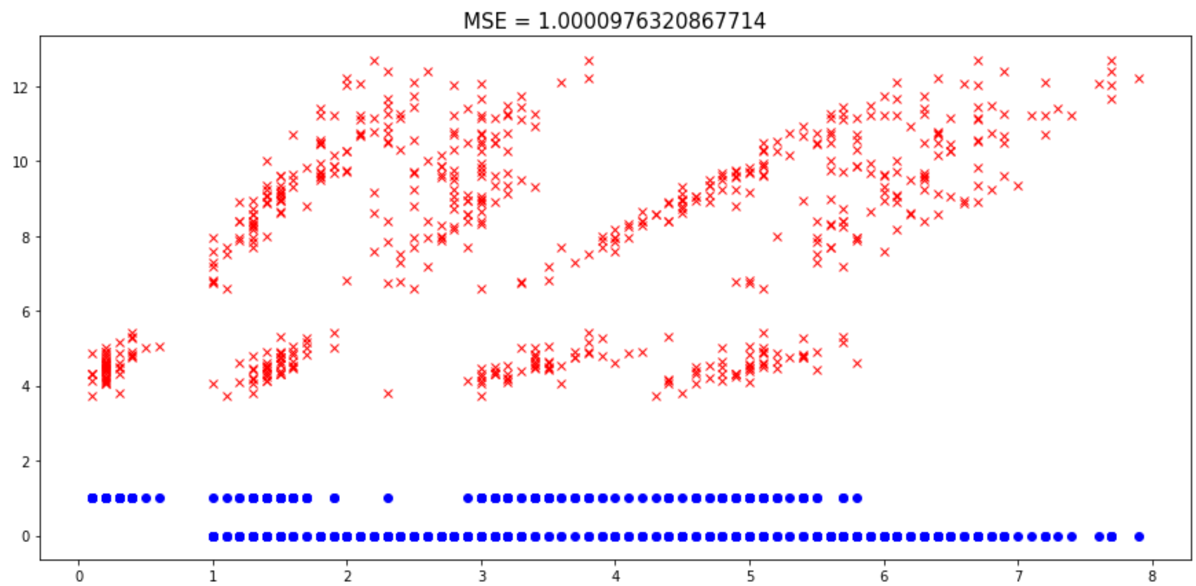
Out[286]:
```
Text(0.5, 1.0, 'MSE = 1.0000976320867714')
```

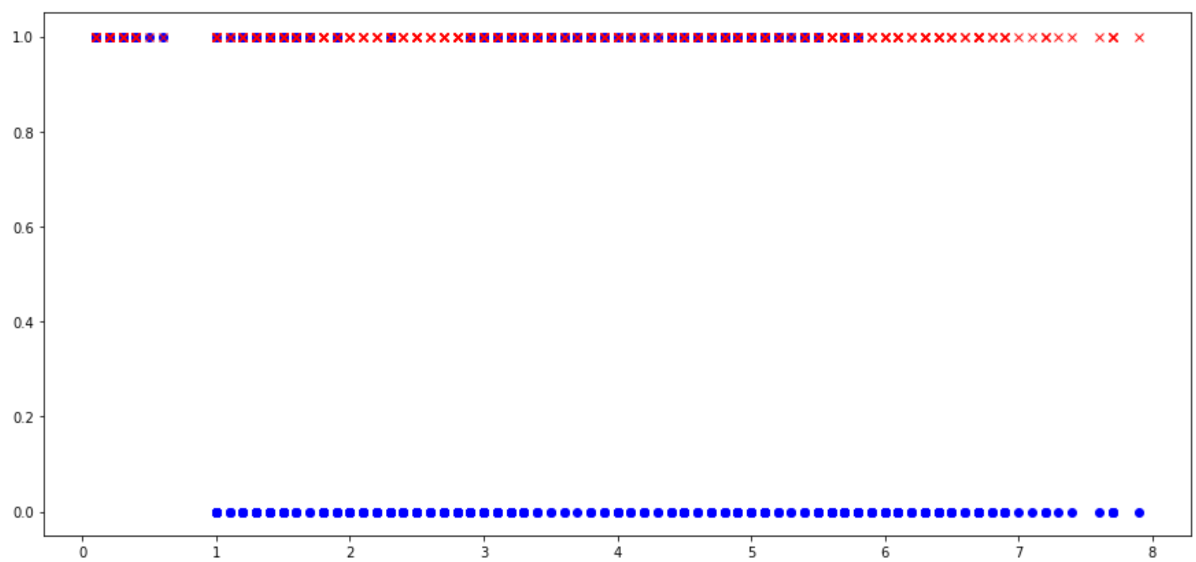


In [287]:
```
ret = check_predicted(X,theta, y, 0, 1, 0.5)

print(ret[2])

plt.figure(figsize=(15,7))
plt.plot(x,y,'bo')
plt.plot(x,ret[0],'rx')
plt.title('MSE = '+str(MSE),fontsize=15)

plt.figure(figsize=(15,7))
plt.plot(x,y,'bo')
plt.plot(x,ret[1],'rx')
```

```
50 were correctly 1
0 were correctly 0
33.33333333333333% were correct
```

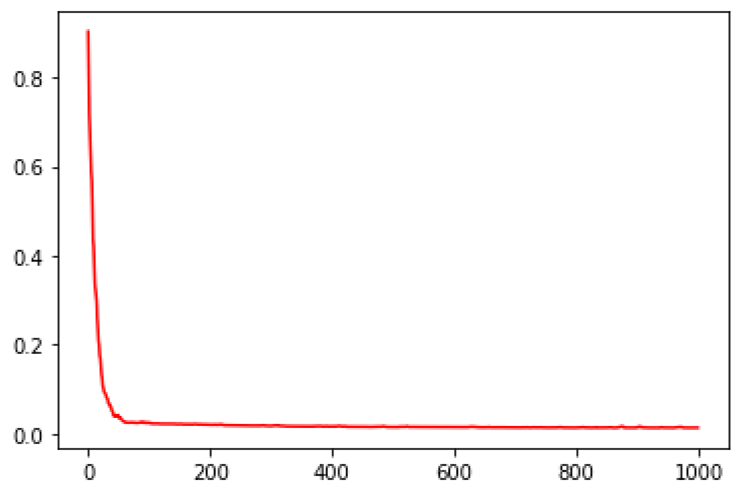
Out[287]:
```
[<matplotlib.lines.Line2D at 0x196f3f5d888>,
 <matplotlib.lines.Line2D at 0x196f3f4fbc8>,
 <matplotlib.lines.Line2D at 0x196f3fa3448>,
 <matplotlib.lines.Line2D at 0x196f3fa36c8>]
```





In [302]:
```
'Stochastic Gradient Descent (SGD)'
m = 150
n_iterations = 1000 #number of SGD steps
theta = np.random.rand(5,1) #random initialization
MSE = np.zeros((n_iterations,1)) #we will compute the MSE function after each
 SGD step
s = 0.1 #learning rate

for i in range(n_iterations):
    random_row = np.random.randint(m) #pick a random integer in [0,m-1]
#     xi = X.iloc[random_row] #ith row;
#     xi = xi[None,:] #keep xi as a row vector
    xi = X[None,random_row] #keep xi as a row vector
    yi = y[random_row]
    gradient = (2/m)*xi.T@(xi@theta-yi)
    theta = theta - s*gradient
    MSE[i] = np.linalg.norm(y-X@theta)/m

plt.plot(MSE,'r-')
```

Out[302]: [<matplotlib.lines.Line2D at 0x196f869da88>]



In [303]:
```
ret = check_predicted(X,theta, y, 0, 1, 0.5)

print(ret[2])

plt.figure(figsize=(15,7))
plt.plot(x,y,'bo')
plt.plot(x,ret[0],'rx')
# plt.title('MSE = '+str(MSE),fontsize=15)

plt.figure(figsize=(15,7))
plt.plot(x,y,'bo')
plt.plot(x,ret[1],'rx')
```

```
50 were correctly 1
100 were correctly 0
100.0% were correct
```

Out[303]:
```
[<matplotlib.lines.Line2D at 0x196f77b8dc8>,
 <matplotlib.lines.Line2D at 0x196f772d608>,
 <matplotlib.lines.Line2D at 0x196f5834608>,
 <matplotlib.lines.Line2D at 0x196f58346c8>]
```