In [1]:
```
"""
ECGR 5105 - Intro to Machine Learning
Homework 5, Part 1
Phillip Harmon
""";
```

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import torch
```

In [3]:
```
#Normalization Functions
def normalize(x, xmax, xmin):
    return (x - xmin) / (xmax - xmin)

def denormalize(x, xmax, xmin):
    return (x * (xmax - xmin)) + xmin
```

In [4]:
```
#Define the linear model
def model_linear(x, w):
    return w[1] * x + w[0]

#Define the nonlinear model
def model_quadratic(x, w):
    return w[2] * x * x + w[1] * x + w[0]

#Define the loss function
def cost(y_p, y):
    square_error = (y_p - y)**2
    return square_error.mean()

#Define Forward Pass Function
def forward_pass(x, y, params, model=model_linear, enable_grad=True):
    with torch.set_grad_enabled(enable_grad):
        loss = cost( model(x, params) , y)
    return loss
```

In [5]:
```
#Define the Training Loop
def train_loop(params, x_t, y_t, x_v, y_v, model, epochs=5000, learn_rate=1e-2
    training_loss = []
    validation_loss = []
    for epoch in range(1, epochs + 1):
        if params.grad is not None:
            params.grad.zero_()

        loss_t = forward_pass(
            x = x_t,
            y = y_t,
            params = params,
            model = model,
            enable_grad = True)

        loss_v = forward_pass(
            x = x_v,
            y = y_v,
            params = params,
            model = model,
            enable_grad = False)

        training_loss.append(float(loss_t))
        validation_loss.append(float(loss_v))

        loss_t.backward()

        with torch.no_grad():
            params -= learn_rate * params.grad

        if epoch <= 3 or epoch % 500 == 0:
            print('Epoch {} | Training Loss = {} | Validation Loss = {}'.forma

    return params, training_loss, validation_loss
```

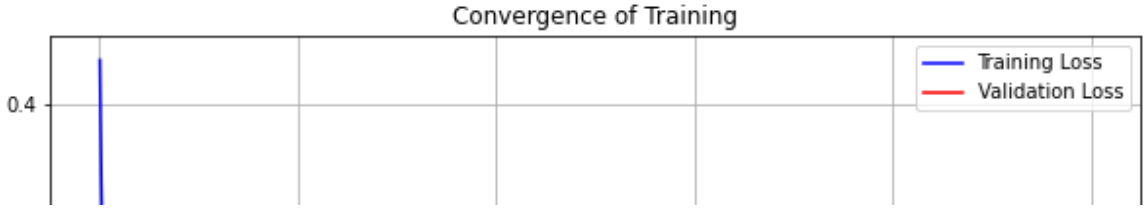
In [6]:
```
#helper for plotting visualization of training data
def training_visual(loss_t, loss_v):
    plt.rcParams["figure.figsize"] = (10,5)
    plt.grid()
    plt.xlabel('Epochs')
    plt.ylabel('MSE Loss')
    plt.title('Convergence of Training')
    plt.plot(range(1,len(loss_t) + 1),loss_t, color='blue', label='Training Lo
    plt.plot(range(1,len(loss_t) + 1),loss_v, color='red', label='Validation L
    plt.legend()
    #plt.ylim([0.0,0.25])
    plt.show()
    print("Final Training Loss = {} | Final Validation Loss = {}".format(loss_
```

In [7]:
```
#helper for plotting visualization of training data
def plot_results(x, y, params, model, x_label='X', y_label='Y', title='Y vs. X
    x_n = normalize(x, x.max(0,keepdim=True)[0], x.min(0,keepdim=True)[0])
    y_n = normalize(y, y.max(0,keepdim=True)[0], y.min(0,keepdim=True)[0])
    lin_x = torch.tensor(np.arange(min(x_n), max(x_n), (max(x_n) - min(x_n)) /
    lin_y = model(lin_x, params).detach()
    y_p = model(x_n, params)

    lin_x = denormalize(lin_x, x.max(0,keepdim=True)[0], x.min(0,keepdim=True)
    lin_y = denormalize(lin_y, y.max(0,keepdim=True)[0], y.min(0,keepdim=True)
    y_p = denormalize(y_p, y.max(0,keepdim=True)[0], y.min(0,keepdim=True)[0])

    print("Model MSE Loss for whole dataset = {}".format(cost(model(x_n,params

    plt.rcParams["figure.figsize"] = (10,5)
    plt.grid()
    plt.xlabel(x_label)
    plt.ylabel(y_label)
    plt.title(title)
    plt.scatter(x, y, color='red', label='Ground Truths')
    plt.scatter(x, y_p, color='blue', label='Predictions', marker="x")
    plt.plot(lin_x, lin_y, color='cyan', label='Model Curve', ls='--')
    plt.legend()
    plt.show()
```

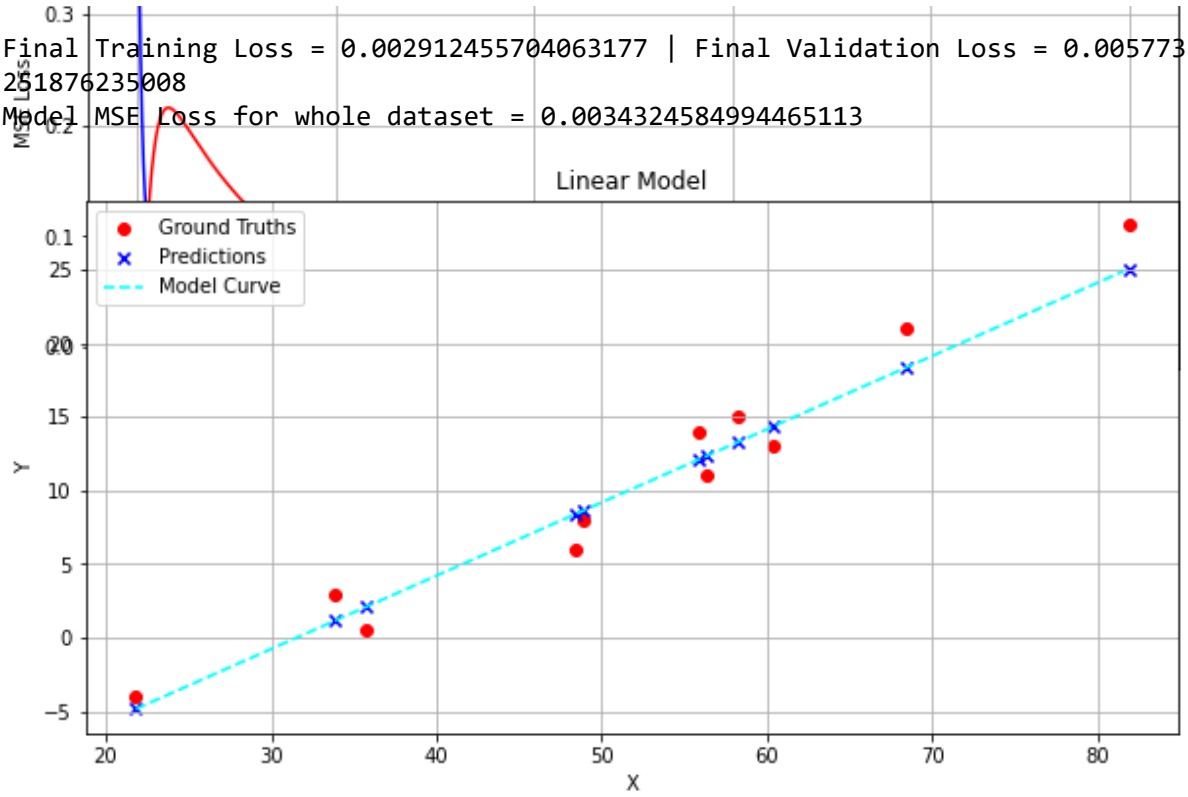
In [8]:
```
#Prepare the inputs
#measurement
xarr = [35.7, 55.9, 58.2, 81.9, 56.3, 48.9, 33.9, 21.8, 48.4, 60.4, 68.4]
x_raw = torch.tensor(xarr)
#celcius
yarr = [0.5, 14.0, 15.0, 28.0, 11.0, 8.0, 3.0, -4.0, 6.0, 13.0, 21.0]
y_raw = torch.tensor(yarr)

#Cleaning the inputs
x = normalize(x_raw, x_raw.max(0,keepdim=True)[0], x_raw.min(0,keepdim=True)[0
y = normalize(y_raw, y_raw.max(0,keepdim=True)[0], y_raw.min(0,keepdim=True)[0

#Train/Test Split
validation_percent = 0.2
split = int(validation_percent * x.shape[0])
shuffle_index = torch.randperm(x.shape[0])
index_t = shuffle_index[:-split]
index_v = shuffle_index[-split:]
x_t = x[index_t]
y_t = y[index_t]
x_v = x[index_v]
y_v = y[index_v]

#Define Constructs
epochs = 5000
```

In [9]:
```
"""Linear Model, 5000 epochs, LR=1e-2"""
print("Linear Model, Learning Rate = {}".format(1e-2))
param, loss_t, loss_v = train_loop(
    params = torch.tensor([1.0, 0.0], requires_grad=True),
    x_t = x_t,
    y_t = y_t,
    x_v = x_v,
    y_v = y_v,
    epochs = epochs,
    learn_rate = 1e-2,
    model = model_linear);

training_visual(loss_t, loss_v)
param.requires_grad = False
plot_results(
    x = x_raw,
    y = y_raw,
    params = param,
    model = model_linear,
    title = "Linear Model"
    )
```

```
Linear Model, Learning Rate = 0.01
Epoch 1 | Training Loss = 0.4392632246017456 | Validation Loss = 0.0825195312
5
Epoch 2 | Training Loss = 0.42202311754226685 | Validation Loss = 0.076636403
799057
Epoch 3 | Training Loss = 0.40559130907058716 | Validation Loss = 0.071388415
99225998
Epoch 500 | Training Loss = 0.03325355052947998 | Validation Loss = 0.1374309
8080158234
Epoch 1000 | Training Loss = 0.01530188787728548 | Validation Loss = 0.068599
82758760452
Epoch 1500 | Training Loss = 0.007968182675540447 | Validation Loss = 0.03705
5011838674545
Epoch 2000 | Training Loss = 0.004972160793840885 | Validation Loss = 0.02197
9261189699173
Epoch 2500 | Training Loss = 0.0037482124753296375 | Validation Loss = 0.0144
21489089727402
Epoch 3000 | Training Loss = 0.00324819702655077 | Validation Loss = 0.010439
775884151459
Epoch 3500 | Training Loss = 0.003043925855308771 | Validation Loss = 0.00824
1587318480015
Epoch 4000 | Training Loss = 0.0029604758601635695 | Validation Loss = 0.0069
78275254368782
Epoch 4500 | Training Loss = 0.0029263850301504135 | Validation Loss = 0.0062
28701677173376
Epoch 5000 | Training Loss = 0.002912455704063177 | Validation Loss = 0.00577
3251876235008
```

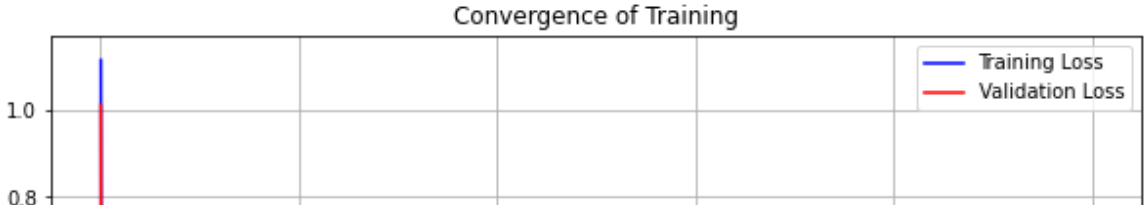


```
Final Training Loss = 0.002912455704063177 | Final Validation Loss = 0.005773
251876235008
Model MSE Loss for whole dataset = 0.0034324584994465113
```

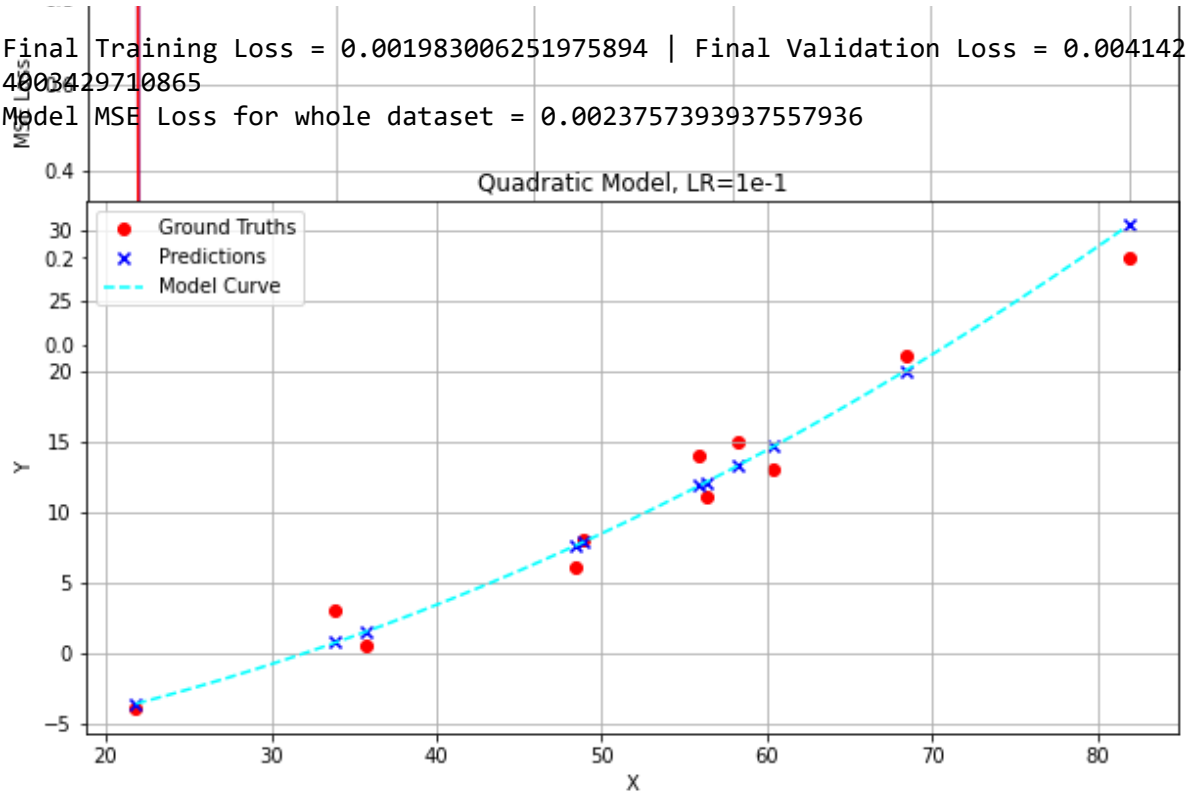


In [10]:
```
"""Quadratic Model, 5000 epochs, LR=1e-1"""
print("Quadratic Model, Learning Rate = {}".format(1e-1))
param, loss_t, loss_v = train_loop(
    params = torch.tensor([1.0, 1.0, 0.0], requires_grad=True),
    x_t = x_t,
    y_t = y_t,
    x_v = x_v,
    y_v = y_v,
    epochs = epochs,
    learn_rate = 1e-1,
    model = model_quadratic);

training_visual(loss_t, loss_v)
param.requires_grad = False
plot_results(
    x = x_raw,
    y = y_raw,
    params = param,
    model = model_quadratic,
    title = "Quadratic Model, LR=1e-1"
    )
```

```
Quadratic Model, Learning Rate = 0.1
Epoch 1 | Training Loss = 1.1159790754318237 | Validation Loss = 1.0119781494
140625
Epoch 2 | Training Loss = 0.6310301423072815 | Validation Loss = 0.4731969833
3740234
Epoch 3 | Training Loss = 0.3597252368927002 | Validation Loss = 0.2056051194
6678162
Epoch 500 | Training Loss = 0.0025247116573154926 | Validation Loss = 0.00195
8972541615367
Epoch 1000 | Training Loss = 0.002356366254389286 | Validation Loss = 0.00144
51019233092666
Epoch 1500 | Training Loss = 0.002238175133243203 | Validation Loss = 0.00140
01831877976656
Epoch 2000 | Training Loss = 0.002155187539756298 | Validation Loss = 0.00162
36853552982211
Epoch 2500 | Training Loss = 0.002096919110044837 | Validation Loss = 0.00199
43180959671736
Epoch 3000 | Training Loss = 0.00205600680783391 | Validation Loss = 0.002433
623420074582
Epoch 3500 | Training Loss = 0.002027279930189252 | Validation Loss = 0.00289
2129123210907
Epoch 4000 | Training Loss = 0.0020071109756827354 | Validation Loss = 0.0033
397795632481575
Epoch 4500 | Training Loss = 0.0019929492846131325 | Validation Loss = 0.0037
59450279176235
Epoch 5000 | Training Loss = 0.001983006251975894 | Validation Loss = 0.00414
24003429710865
```

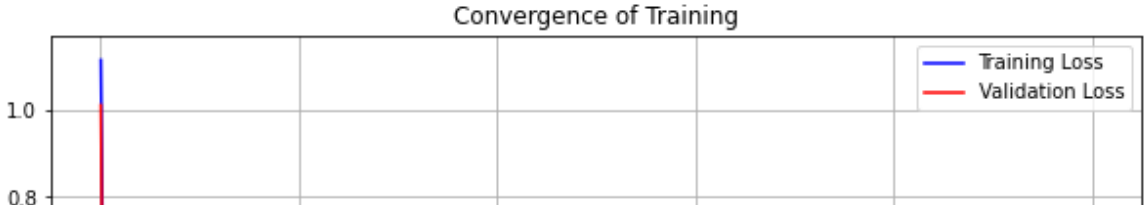


```
Final Training Loss = 0.001983006251975894 | Final Validation Loss = 0.004142
4003429710865
Model MSE Loss for whole dataset = 0.0023757393937557936
```

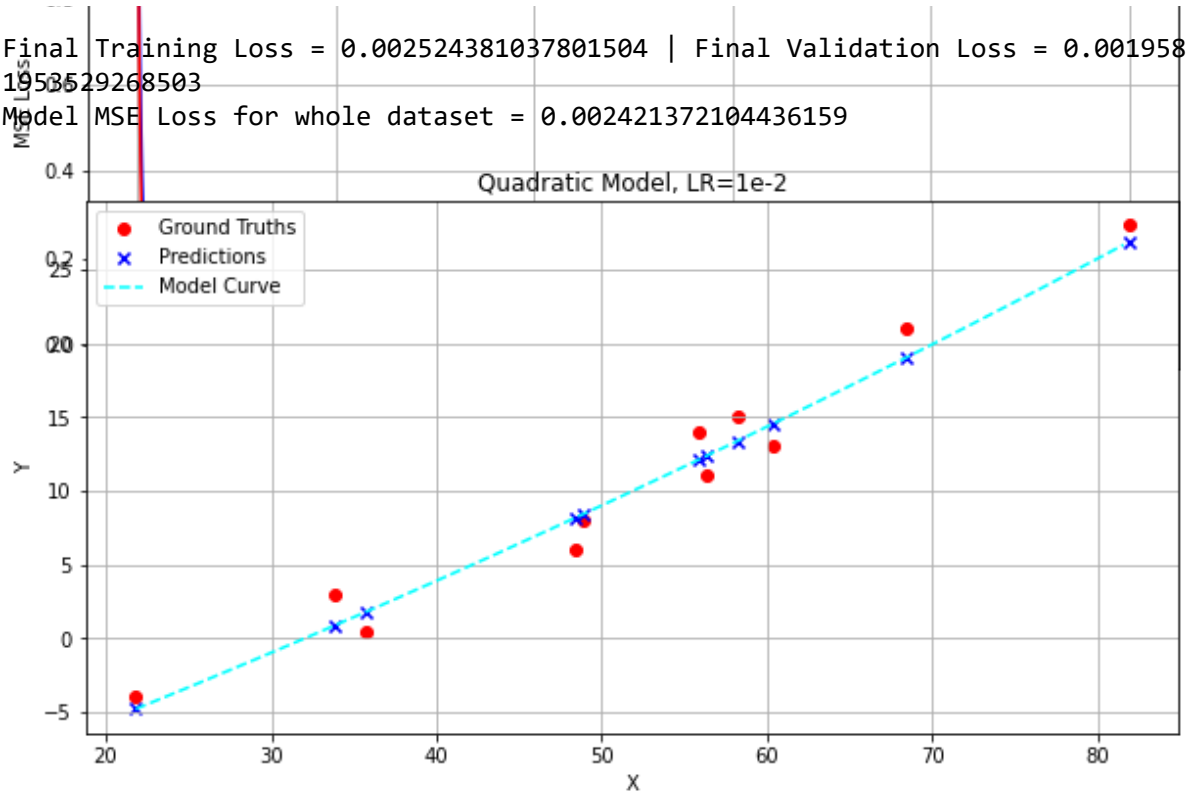


In [11]:
```
"""Quadratic Model, 5000 epochs, LR=1e-2"""
print("Quadratic Model, Learning Rate = {}".format(1e-2))
param, loss_t, loss_v = train_loop(
    params = torch.tensor([1.0, 1.0, 0.0], requires_grad=True),
    x_t = x_t,
    y_t = y_t,
    x_v = x_v,
    y_v = y_v,
    epochs = epochs,
    learn_rate = 1e-2,
    model = model_quadratic);

training_visual(loss_t, loss_v)
param.requires_grad = False
plot_results(
    x = x_raw,
    y = y_raw,
    params = param,
    model = model_quadratic,
    title = "Quadratic Model, LR=1e-2"
    )
```

```
Quadratic Model, Learning Rate = 0.01
Epoch 1 | Training Loss = 1.1159790754318237 | Validation Loss = 1.0119781494
140625
Epoch 2 | Training Loss = 1.0611895322799683 | Validation Loss = 0.9488234519
958496
Epoch 3 | Training Loss = 1.0091272592544556 | Validation Loss = 0.8892593979
83551
Epoch 500 | Training Loss = 0.006314016878604889 | Validation Loss = 0.032410
189509391785
Epoch 1000 | Training Loss = 0.0036123048048466444 | Validation Loss = 0.0131
32939115166664
Epoch 1500 | Training Loss = 0.0029110193718224764 | Validation Loss = 0.0066
75686687231064
Epoch 2000 | Training Loss = 0.0027154150884598494 | Validation Loss = 0.0042
31832455843687
Epoch 2500 | Training Loss = 0.0026481228414922953 | Validation Loss = 0.0031
79206047207117
Epoch 3000 | Training Loss = 0.002613841090351343 | Validation Loss = 0.00266
32335502654314
Epoch 3500 | Training Loss = 0.0025884974747896194 | Validation Loss = 0.0023
75128213316202
Epoch 4000 | Training Loss = 0.002565988339483738 | Validation Loss = 0.00219
1972453147173
Epoch 4500 | Training Loss = 0.002544754883274436 | Validation Loss = 0.00206
09640050679445
Epoch 5000 | Training Loss = 0.002524381037801504 | Validation Loss = 0.00195
81953529268503
```

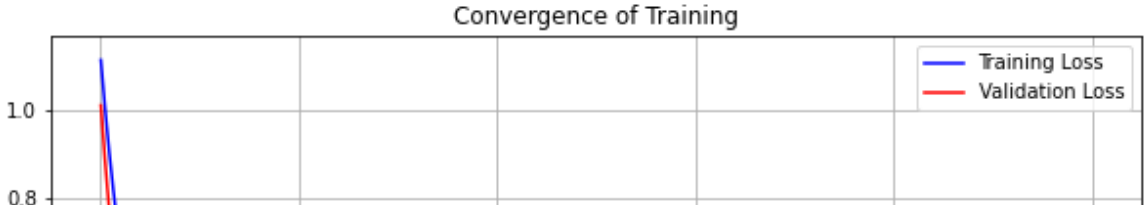


```
Final Training Loss = 0.002524381037801504 | Final Validation Loss = 0.001958
1953529268503
Model MSE Loss for whole dataset = 0.002421372104436159
```

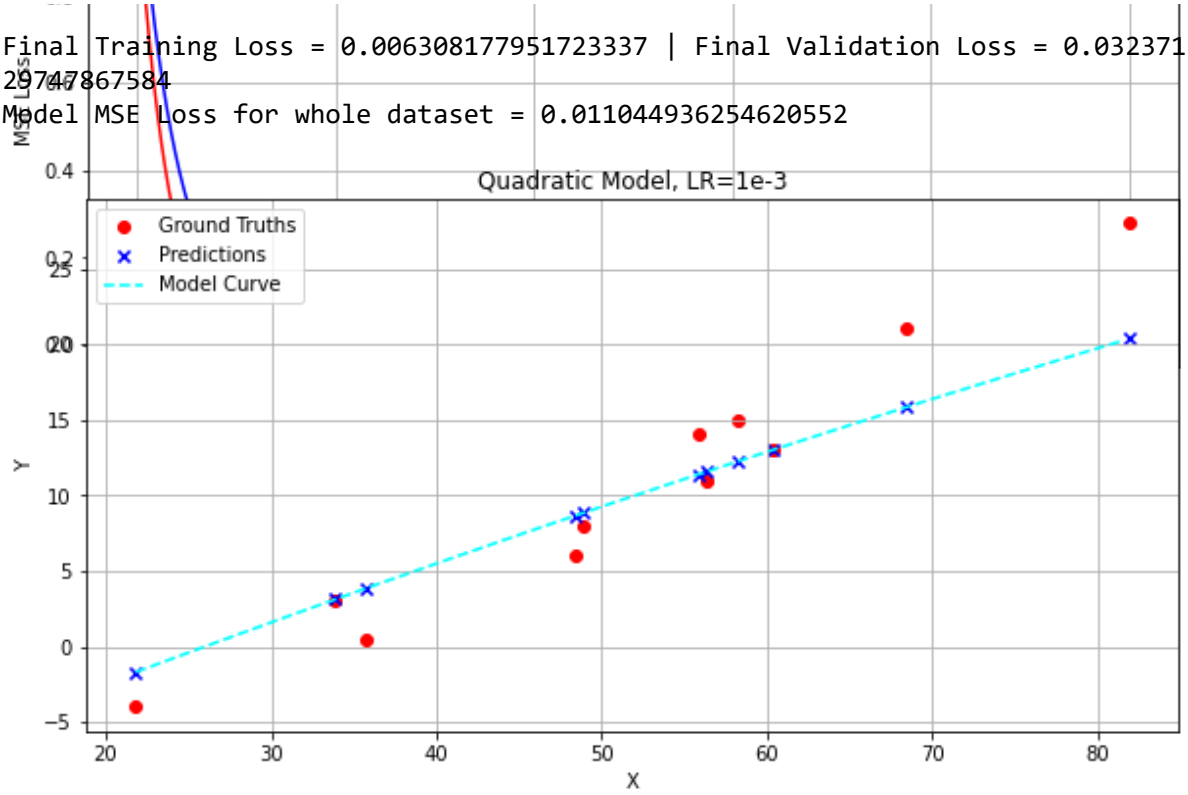


In [12]:
```
"""Quadratic Model, 5000 epochs, LR=1e-3"""
print("Quadratic Model, Learning Rate = {}".format(1e-3))
param, loss_t, loss_v = train_loop(
    params = torch.tensor([1.0, 1.0, 0.0], requires_grad=True),
    x_t = x_t,
    y_t = y_t,
    x_v = x_v,
    y_v = y_v,
    epochs = epochs,
    learn_rate = 1e-3,
    model = model_quadratic);

training_visual(loss_t, loss_v)
param.requires_grad = False
plot_results(
    x = x_raw,
    y = y_raw,
    params = param,
    model = model_quadratic,
    title = "Quadratic Model, LR=1e-3"
    )
```

```
Quadratic Model, Learning Rate = 0.001
Epoch 1 | Training Loss = 1.1159790754318237 | Validation Loss = 1.0119781494
140625
Epoch 2 | Training Loss = 1.1104371547698975 | Validation Loss = 1.0055698156
356812
Epoch 3 | Training Loss = 1.104923129081726 | Validation Loss = 0.99919855594
63501
Epoch 500 | Training Loss = 0.10345166176557541 | Validation Loss = 0.0208549
01522397995
Epoch 1000 | Training Loss = 0.020564958453178406 | Validation Loss = 0.03404
11551296711
Epoch 1500 | Training Loss = 0.012678220868110657 | Validation Loss = 0.05485
355854034424
Epoch 2000 | Training Loss = 0.010952466167509556 | Validation Loss = 0.05727
146938443184
Epoch 2500 | Training Loss = 0.009859978221356869 | Validation Loss = 0.05365
3694689273834
Epoch 3000 | Training Loss = 0.008940544910728931 | Validation Loss = 0.04879
271239042282
Epoch 3500 | Training Loss = 0.008141514845192432 | Validation Loss = 0.04403
477907180786
Epoch 4000 | Training Loss = 0.007445048075169325 | Validation Loss = 0.03969
774395227432
Epoch 4500 | Training Loss = 0.006837758701294661 | Validation Loss = 0.03582
017868757248
Epoch 5000 | Training Loss = 0.006308177951723337 | Validation Loss = 0.03237
129747867584
```

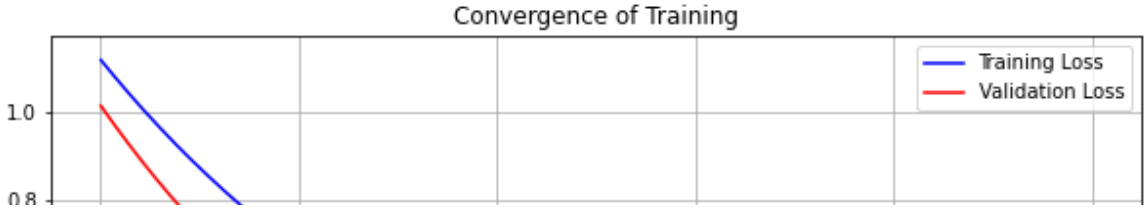


```
Final Training Loss = 0.006308177951723337 | Final Validation Loss = 0.032371
29747867584
Model MSE Loss for whole dataset = 0.011044936254620552
```

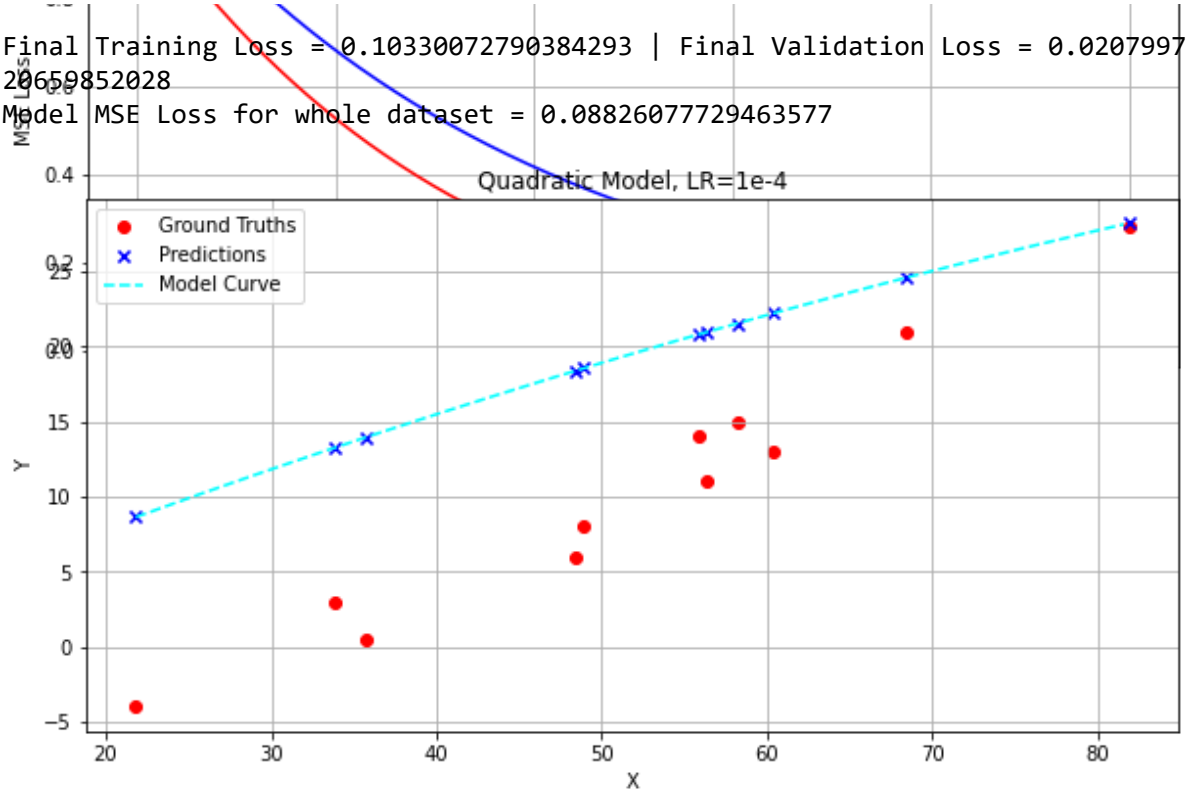


In [13]:
```
"""Quadratic Model, 5000 epochs, LR=1e-4"""
print("Quadratic Model, Learning Rate = {}".format(1e-4))
param, loss_t, loss_v = train_loop(
    params = torch.tensor([1.0, 1.0, 0.0], requires_grad=True),
    x_t = x_t,
    y_t = y_t,
    x_v = x_v,
    y_v = y_v,
    epochs = epochs,
    learn_rate = 1e-4,
    model = model_quadratic);

training_visual(loss_t, loss_v)
param.requires_grad = False
plot_results(
    x = x_raw,
    y = y_raw,
    params = param,
    model = model_quadratic,
    title = "Quadratic Model, LR=1e-4"
    )
```

```
Quadratic Model, Learning Rate = 0.0001
Epoch 1 | Training Loss = 1.1159790754318237 | Validation Loss = 1.0119781494
140625
Epoch 2 | Training Loss = 1.1154241561889648 | Validation Loss = 1.0113363265
99121
Epoch 3 | Training Loss = 1.1148697137832642 | Validation Loss = 1.0106948614
120483
Epoch 500 | Training Loss = 0.8711451888084412 | Validation Loss = 0.73380398
75030518
Epoch 1000 | Training Loss = 0.6804319620132446 | Validation Loss = 0.5259684
324264526
Epoch 1500 | Training Loss = 0.5322036147117615 | Validation Loss = 0.3721737
861633301
Epoch 2000 | Training Loss = 0.416988343000412 | Validation Loss = 0.25939500
33187866
Epoch 2500 | Training Loss = 0.32742494344711304 | Validation Loss = 0.177631
49738311768
Epoch 3000 | Training Loss = 0.2577933967113495 | Validation Loss = 0.1192180
9613704681
Epoch 3500 | Training Loss = 0.20365063846111298 | Validation Loss = 0.078292
8392291069
Epoch 4000 | Training Loss = 0.1615433543920517 | Validation Loss = 0.0503813
58712911606
Epoch 4500 | Training Loss = 0.12878841161727905 | Validation Loss = 0.032078
199088573456
Epoch 5000 | Training Loss = 0.10330072790384293 | Validation Loss = 0.020799
720659852028
```



```
Final Training Loss = 0.10330072790384293 | Final Validation Loss = 0.0207997
20659852028
Model MSE Loss for whole dataset = 0.08826077729463577
```



In [ ]:

In [1]:
```
"""
ECGR 5105 - Intro to Machine Learning
Homework 5, Part 2
Phillip Harmon
""";
```

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import torch
```

In [3]:
```
#Normalization Functions
def normalize(x, xmax, xmin):
    return (x - xmin) / (xmax - xmin)

def denormalize(x, xmax, xmin):
    return (x * (xmax - xmin)) + xmin
```

In [4]:
```
#Define the linear model
def model_linear(x, w):
    values = torch.column_stack((x, torch.ones(x.size(dim=0))))
    return (w * values).sum(1)

#Define the loss function
def cost(y_p, y):
    square_error = (y_p - y)**2
    return square_error.mean()

#Define Forward Pass Function
def forward_pass(x, y, params, model=model_linear, enable_grad=True):
    with torch.set_grad_enabled(enable_grad):
        loss = cost( model(x, params) , y)
    return loss
```

In [5]:
```
#helper for plotting visualization of training data
def training_visual(loss_t, loss_v, model, params, x, y):
    plt.rcParams["figure.figsize"] = (10,5)
    plt.grid()
    plt.xlabel('Epochs')
    plt.ylabel('MSE Loss')
    plt.title('Convergence of Training')
    plt.plot(range(1,len(loss_t) + 1),loss_t, color='blue', label='Training Lo
    plt.plot(range(1,len(loss_t) + 1),loss_v, color='red', label='Validation L
    plt.legend()
    plt.ylim([0.0,0.25])
    plt.show()
    print("Final Training Loss = {} | Final Validation Loss = {}".format(loss_

    x_n = normalize(x, x.max(0,keepdim=True)[0], x.min(0,keepdim=True)[0])
    y_n = normalize(y, y.max(0,keepdim=True)[0], y.min(0,keepdim=True)[0])
    print("Model MSE Loss for whole dataset = {}".format(cost(model(x_n,params
```

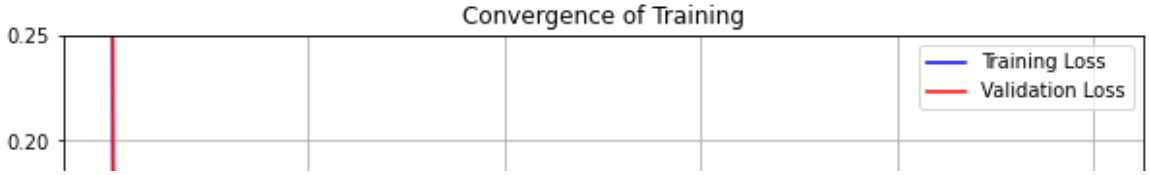
In [6]:
```
#Define the Training Loop
def train_loop(params, x_t, y_t, x_v, y_v, model, epochs=5000, learn_rate=1e-2
    training_loss = []
    validation_loss = []
    for epoch in range(1, epochs + 1):
        if params.grad is not None:
            params.grad.zero_()

        loss_t = forward_pass(
            x = x_t,
            y = y_t,
            params = params,
            model = model,
            enable_grad = True)

        loss_v = forward_pass(
            x = x_v,
            y = y_v,
            params = params,
            model = model,
            enable_grad = False)

        training_loss.append(float(loss_t))
        validation_loss.append(float(loss_v))

        loss_t.backward()

        with torch.no_grad():
            params -= learn_rate * params.grad

        if epoch <= 3 or epoch % 500 == 0:
            print('Epoch {} | Training Loss = {} | Validation Loss = {}'.forma

    return params, training_loss, validation_loss
```

In [7]:
```
#Define the Training Loop with optimizer
def train_loop_optim(params, x_t, y_t, x_v, y_v, model, optimizer, epochs=5000
    training_loss = []
    validation_loss = []
    for epoch in range(1, epochs + 1):

        loss_t = forward_pass(
            x = x_t,
            y = y_t,
            params = params,
            model = model,
            enable_grad = True)

        loss_v = forward_pass(
            x = x_v,
            y = y_v,
            params = params,
            model = model,
            enable_grad = False)

        training_loss.append(float(loss_t))
        validation_loss.append(float(loss_v))

        optimizer.zero_grad()
        loss_t.backward()
        optimizer.step()

        if epoch <= 3 or epoch % 500 == 0:
            print('Epoch {} | Training Loss = {} | Validation Loss = {}'.forma

    return params, training_loss, validation_loss
```

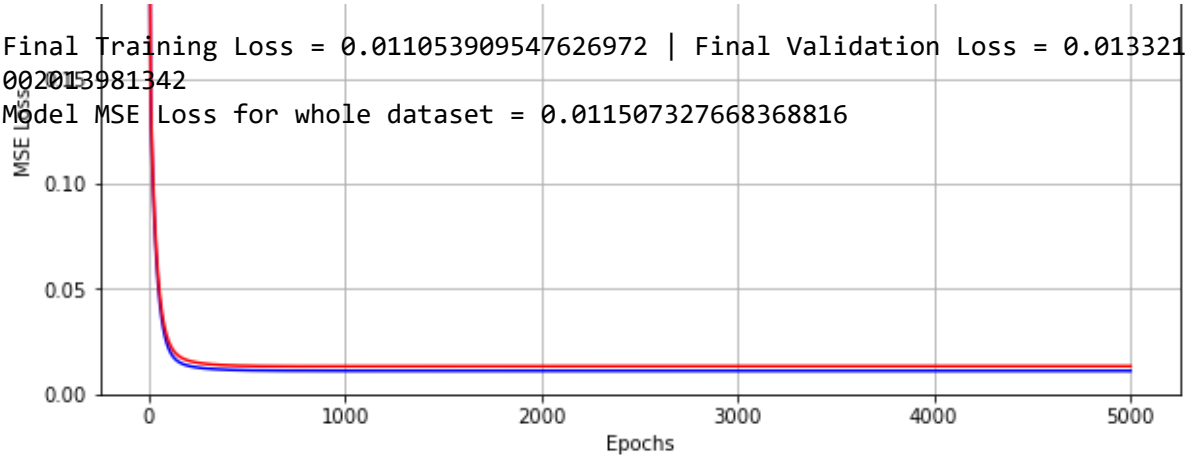
In [8]:
```
#Prepare the inputs

#Read in the CSV into a dataframe
csvData = pd.read_csv("./Housing.csv")
csvCols = len(csvData.columns)
csvRows = len(csvData)

#Collect Data
dataLabels = ['price','area','bedrooms','bathrooms','stories','parking']
data = csvData[dataLabels]

y_raw = data.pop('price').values
x_raw = data.values

y_raw = torch.from_numpy(y_raw)
x_raw = torch.from_numpy(x_raw)

#Cleaning the inputs
x = normalize(x_raw, x_raw.max(0,keepdim=True)[0], x_raw.min(0,keepdim=True)[0
y = normalize(y_raw, y_raw.max(0,keepdim=True)[0], y_raw.min(0,keepdim=True)[0

#Train/Test Split
validation_percent = 0.2
split = int(validation_percent * x.shape[0])
shuffle_index = torch.randperm(x.shape[0])
index_t = shuffle_index[:-split]
index_v = shuffle_index[-split:]
x_t = x[index_t]
y_t = y[index_t]
x_v = x[index_v]
y_v = y[index_v]

#Define Constructs
epochs = 5000
```

In [9]:
```
"""LR=1e-1"""
print("Learning Rate = {}".format(1e-1))
param = torch.column_stack((torch.ones(1,x.size(dim=1)), torch.zeros(1)))[0]
param.requires_grad = True
param, loss_t, loss_v = train_loop(
    params = param,
    x_t = x_t,
    y_t = y_t,
    x_v = x_v,
    y_v = y_v,
    epochs = epochs,
    learn_rate = 1e-1,
    model = model_linear);

param.requires_grad = False
training_visual(
    loss_t = loss_t,
    loss_v = loss_v,
    model = model_linear,
    params = param,
    x = x_raw,
    y = y_raw)
```

```
Learning Rate = 0.1
Epoch 1 | Training Loss = 1.2239868640899658 | Validation Loss = 1.2471128702
163696
Epoch 2 | Training Loss = 0.7253420352935791 | Validation Loss = 0.7413965463
638306
Epoch 3 | Training Loss = 0.4595836102962494 | Validation Loss = 0.4711650907
9933167
Epoch 500 | Training Loss = 0.011338168755173683 | Validation Loss = 0.013433
423824608326
Epoch 1000 | Training Loss = 0.01107100211083889 | Validation Loss = 0.013250
169344246387
Epoch 1500 | Training Loss = 0.011055151000618935 | Validation Loss = 0.01329
437643289566
Epoch 2000 | Training Loss = 0.011054002679884434 | Validation Loss = 0.01331
313606351614
Epoch 2500 | Training Loss = 0.011053916066884995 | Validation Loss = 0.01331
8805955350399
Epoch 3000 | Training Loss = 0.011053909547626972 | Validation Loss = 0.01332
040410488844
Epoch 3500 | Training Loss = 0.011053909547626972 | Validation Loss = 0.01332
0842757821083
Epoch 4000 | Training Loss = 0.011053908616304398 | Validation Loss = 0.01332
0968486368656
Epoch 4500 | Training Loss = 0.011053909547626972 | Validation Loss = 0.01332
099549472332
Epoch 5000 | Training Loss = 0.011053909547626972 | Validation Loss = 0.01332
1002013981342
```

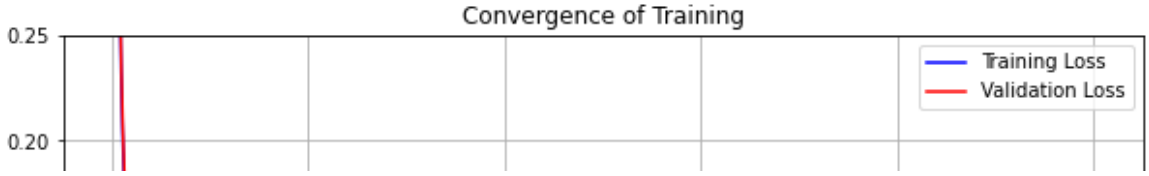


```
Final Training Loss = 0.011053909547626972 | Final Validation Loss = 0.013321
002013981342
Model MSE Loss for whole dataset = 0.011507327668368816
```

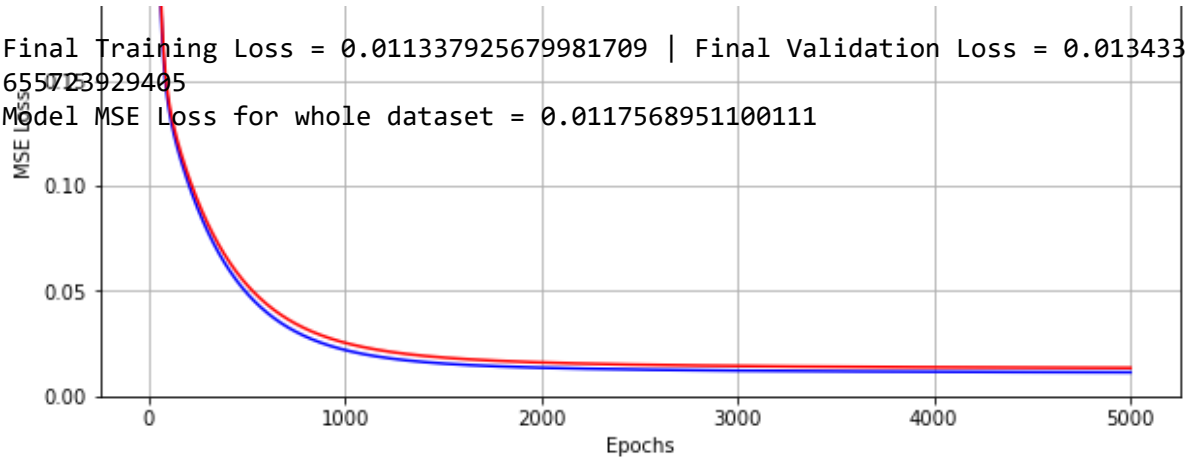


In [10]:
```
"""LR=1e-2"""
print("Learning Rate = {}".format(1e-2))
param = torch.column_stack((torch.ones(1,x.size(dim=1)), torch.zeros(1)))[0]
param.requires_grad = True
param, loss_t, loss_v = train_loop(
    params = param,
    x_t = x_t,
    y_t = y_t,
    x_v = x_v,
    y_v = y_v,
    epochs = epochs,
    learn_rate = 1e-2,
    model = model_linear);

param.requires_grad = False
training_visual(
    loss_t = loss_t,
    loss_v = loss_v,
    model = model_linear,
    params = param,
    x = x,
    y = y)
```

```
Learning Rate = 0.01
Epoch 1 | Training Loss = 1.2239868640899658 | Validation Loss = 1.2471128702
163696
Epoch 2 | Training Loss = 1.1670957803726196 | Validation Loss = 1.1894673109
054565
Epoch 3 | Training Loss = 1.1132428646087646 | Validation Loss = 1.1348906755
447388
Epoch 500 | Training Loss = 0.04896058142185211 | Validation Loss = 0.0529560
03695726395
Epoch 1000 | Training Loss = 0.021924735978245735 | Validation Loss = 0.02535
1444259285927
Epoch 1500 | Training Loss = 0.015495523810386658 | Validation Loss = 0.01833
866350352764
Epoch 2000 | Training Loss = 0.013500482775270939 | Validation Loss = 0.01599
494181573391
Epoch 2500 | Training Loss = 0.0126235606148839 | Validation Loss = 0.0149284
13555026054
Epoch 3000 | Training Loss = 0.012125825509428978 | Validation Loss = 0.01432
8173361718655
Epoch 3500 | Training Loss = 0.011806707829236984 | Validation Loss = 0.01395
3054323792458
Epoch 4000 | Training Loss = 0.011591452173888683 | Validation Loss = 0.01370
7597739994526
Epoch 4500 | Training Loss = 0.011442586779594421 | Validation Loss = 0.01354
3758541345596
Epoch 5000 | Training Loss = 0.011337925679981709 | Validation Loss = 0.01343
3655723929405
```

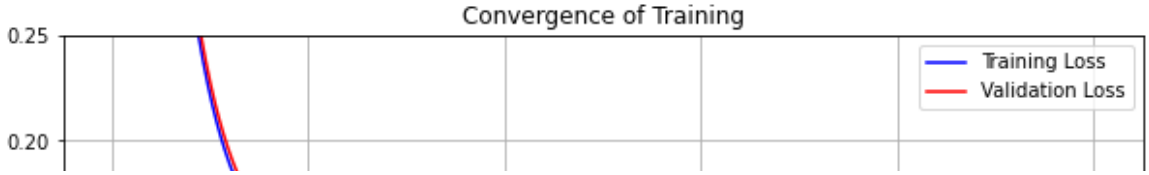


```
Final Training Loss = 0.011337925679981709 | Final Validation Loss = 0.013433
6557723929405
Model MSE Loss for whole dataset = 0.0117568951100111
```

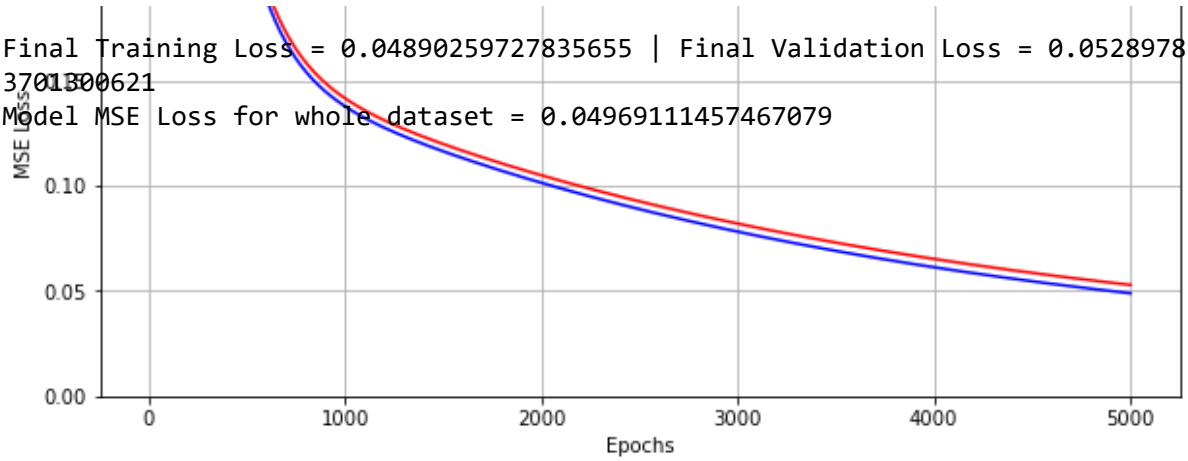


In [11]:
```
"""LR=1e-3"""
print("Learning Rate = {}".format(1e-3))
param = torch.column_stack((torch.ones(1,x.size(dim=1)), torch.zeros(1)))[0]
param.requires_grad = True
param, loss_t, loss_v = train_loop(
    params = param,
    x_t = x_t,
    y_t = y_t,
    x_v = x_v,
    y_v = y_v,
    epochs = epochs,
    learn_rate = 1e-3,
    model = model_linear);

param.requires_grad = False
training_visual(
    loss_t = loss_t,
    loss_v = loss_v,
    model = model_linear,
    params = param,
    x = x,
    y = y)
```

```
Learning Rate = 0.001
Epoch 1 | Training Loss = 1.2239868640899658 | Validation Loss = 1.2471128702
163696
Epoch 2 | Training Loss = 1.2182273864746094 | Validation Loss = 1.2412774562
835693
Epoch 3 | Training Loss = 1.2124992609024048 | Validation Loss = 1.2354736328
125
Epoch 500 | Training Loss = 0.22195826470851898 | Validation Loss = 0.2284213
751554489
Epoch 1000 | Training Loss = 0.1378132402896881 | Validation Loss = 0.1416181
4749240875
Epoch 1500 | Training Loss = 0.11642010509967804 | Validation Loss = 0.119888
45467567444
Epoch 2000 | Training Loss = 0.10145088285207748 | Validation Loss = 0.105018
61572265625
Epoch 2500 | Training Loss = 0.08889636397361755 | Validation Loss = 0.092615
79811573029
Epoch 3000 | Training Loss = 0.07819823920726776 | Validation Loss = 0.082041
94158315659
Epoch 3500 | Training Loss = 0.06906675547361374 | Validation Loss = 0.072995
67759037018
Epoch 4000 | Training Loss = 0.06126762181520462 | Validation Loss = 0.065246
23930454254
Epoch 4500 | Training Loss = 0.05460244044661522 | Validation Loss = 0.058601
23783349991
Epoch 5000 | Training Loss = 0.04890259727835655 | Validation Loss = 0.052897
83701300621
```

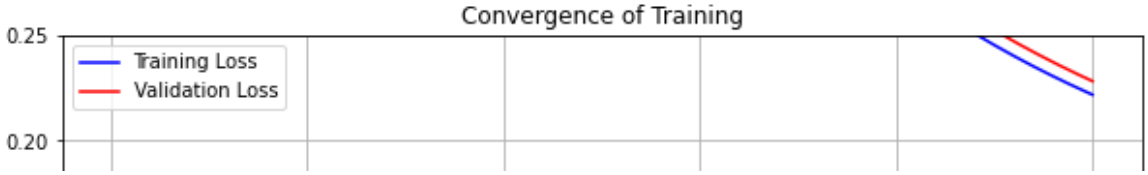


```
Final Training Loss = 0.04890259727835655 | Final Validation Loss = 0.0528978
3701300621
Model MSE Loss for whole dataset = 0.04969111457467079
```

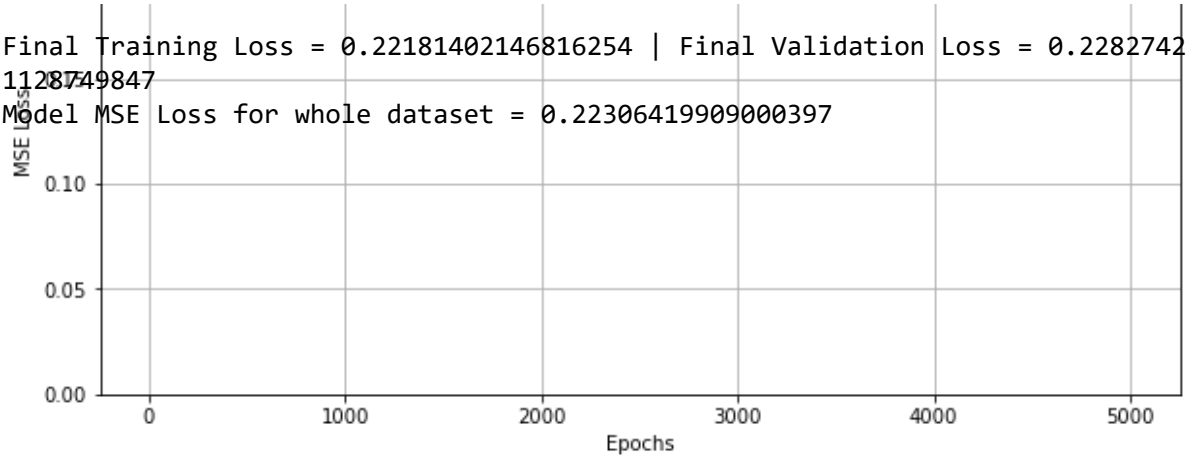


In [12]:
```
"""LR=1e-4"""
print("Learning Rate = {}".format(1e-4))
param = torch.column_stack((torch.ones(1,x.size(dim=1)), torch.zeros(1)))[0]
param.requires_grad = True
param, loss_t, loss_v = train_loop(
    params = param,
    x_t = x_t,
    y_t = y_t,
    x_v = x_v,
    y_v = y_v,
    epochs = epochs,
    learn_rate = 1e-4,
    model = model_linear);

param.requires_grad = False
training_visual(
    loss_t = loss_t,
    loss_v = loss_v,
    model = model_linear,
    params = param,
    x = x,
    y = y)
```

```
Learning Rate = 0.0001
Epoch 1 | Training Loss = 1.2239868640899658 | Validation Loss = 1.2471128702
163696
Epoch 2 | Training Loss = 1.2234102487564087 | Validation Loss = 1.2465283870
697021
Epoch 3 | Training Loss = 1.2228338718414307 | Validation Loss = 1.2459446191
78772
Epoch 500 | Training Loss = 0.9717539548873901 | Validation Loss = 0.99144971
37069702
Epoch 1000 | Training Loss = 0.7787889838218689 | Validation Loss = 0.7956739
068031311
Epoch 1500 | Training Loss = 0.6313768029212952 | Validation Loss = 0.6459552
645683289
Epoch 2000 | Training Loss = 0.5186406970024109 | Validation Loss = 0.5313202
738761902
Epoch 2500 | Training Loss = 0.43230369687080383 | Validation Loss = 0.443414
8371219635
Epoch 3000 | Training Loss = 0.3660649359226227 | Validation Loss = 0.3758769
3333625793
Epoch 3500 | Training Loss = 0.3151301443576813 | Validation Loss = 0.3238632
082939148
Epoch 4000 | Training Loss = 0.27585095167160034 | Validation Loss = 0.283685
92262268066
Epoch 4500 | Training Loss = 0.24545010924339294 | Validation Loss = 0.252535
99882125854
Epoch 5000 | Training Loss = 0.22181402146816254 | Validation Loss = 0.228274
21128749847
```

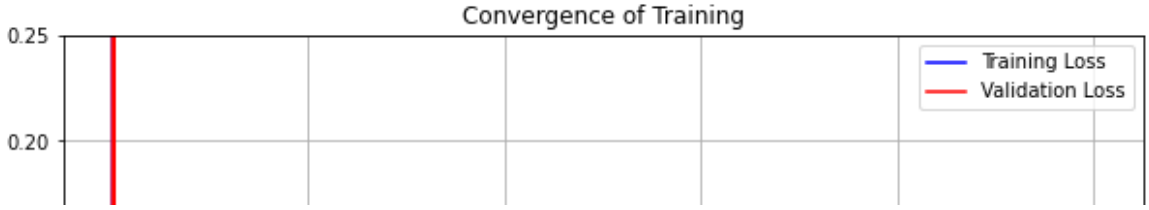


```
Final Training Loss = 0.22181402146816254 | Final Validation Loss = 0.2282742
1128749847
Model MSE Loss for whole dataset = 0.22306419909000397
```

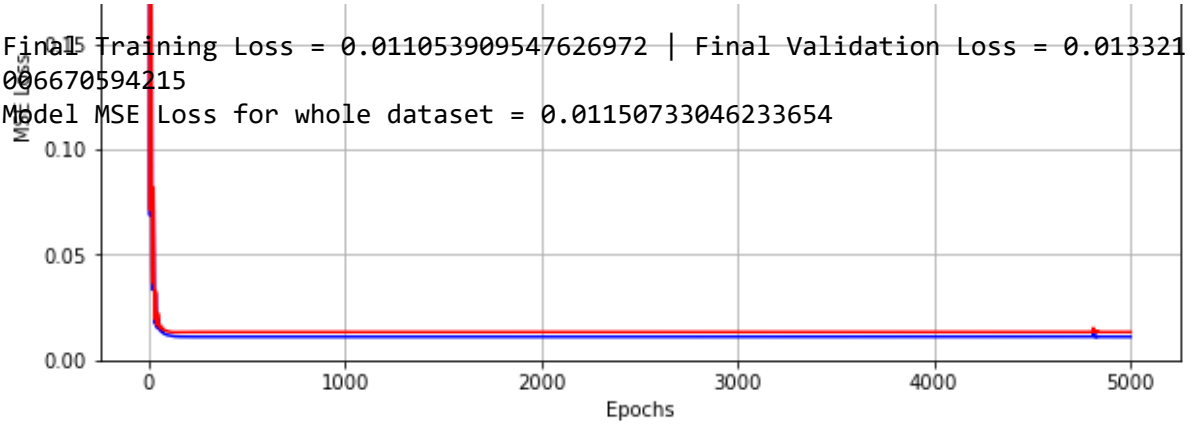


In [13]:
```
"""Adam Optimizer, LR=1e-1"""
param = torch.column_stack((torch.ones(1,x.size(dim=1)), torch.zeros(1)))[0]
param.requires_grad = True
optimizer = torch.optim.Adam([param], lr=1e-1)
param, loss_t, loss_v = train_loop_optim(
    params = param,
    x_t = x_t,
    y_t = y_t,
    x_v = x_v,
    y_v = y_v,
    epochs = epochs,
    optimizer = optimizer,
    model = model_linear);

param.requires_grad = False
training_visual(
    loss_t = loss_t,
    loss_v = loss_v,
    model = model_linear,
    params = param,
    x = x_raw,
    y = y_raw)
```

```
Epoch 1 | Training Loss = 1.2239868640899658 | Validation Loss = 1.2471128702
163696
Epoch 2 | Training Loss = 0.7801061272621155 | Validation Loss = 0.7957643270
492554
Epoch 3 | Training Loss = 0.4473039209842682 | Validation Loss = 0.4572403132
915497
Epoch 500 | Training Loss = 0.011053908616304398 | Validation Loss = 0.013321
008533239365
Epoch 1000 | Training Loss = 0.011053909547626972 | Validation Loss = 0.01332
100946456194
Epoch 1500 | Training Loss = 0.011053909547626972 | Validation Loss = 0.01332
100946456194
Epoch 2000 | Training Loss = 0.011053909547626972 | Validation Loss = 0.01332
100946456194
Epoch 2500 | Training Loss = 0.011053909547626972 | Validation Loss = 0.01332
100946456194
Epoch 3000 | Training Loss = 0.011053909547626972 | Validation Loss = 0.01332
1010395884514
Epoch 3500 | Training Loss = 0.011053908616304398 | Validation Loss = 0.01332
100946456194
Epoch 4000 | Training Loss = 0.011053909547626972 | Validation Loss = 0.01332
1010395884514
Epoch 4500 | Training Loss = 0.011053908616304398 | Validation Loss = 0.01332
100946456194
Epoch 5000 | Training Loss = 0.011053909547626972 | Validation Loss = 0.01332
1006670594215
```

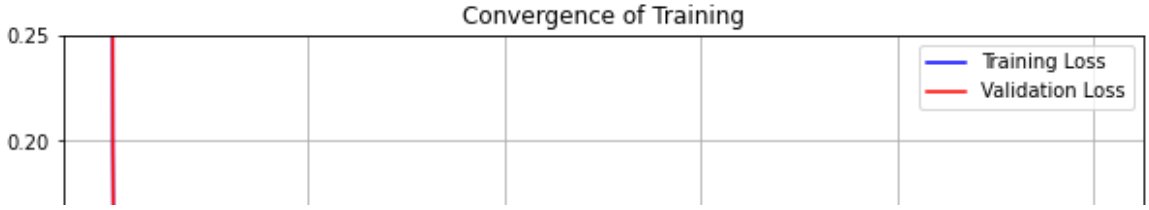


```
Final Training Loss = 0.011053909547626972 | Final Validation Loss = 0.013321
006670594215
Model MSE Loss for whole dataset = 0.01150733046233654
```



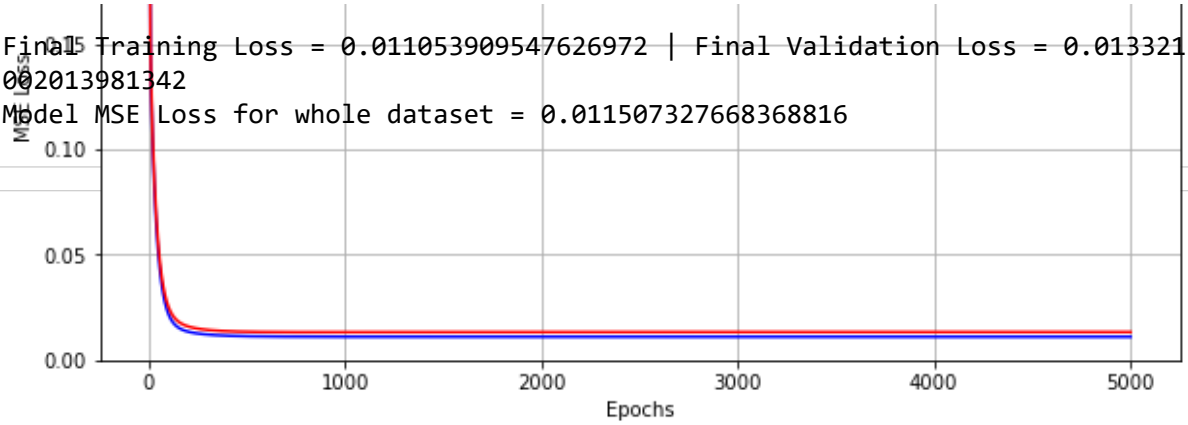
In [14]:
```
"""SGD Optimizer, LR=1e-1"""
param = torch.column_stack((torch.ones(1,x.size(dim=1)), torch.zeros(1)))[0]
param.requires_grad = True
optimizer = torch.optim.SGD([param], lr=1e-1)
param, loss_t, loss_v = train_loop_optim(
    params = param,
    x_t = x_t,
    y_t = y_t,
    x_v = x_v,
    y_v = y_v,
    epochs = epochs,
    optimizer = optimizer,
    model = model_linear);

param.requires_grad = False
training_visual(
    loss_t = loss_t,
    loss_v = loss_v,
    model = model_linear,
    params = param,
    x = x_raw,
    y = y_raw)
```

```
Epoch 1 | Training Loss = 1.2239868640899658 | Validation Loss = 1.2471128702
163696
Epoch 2 | Training Loss = 0.7253420352935791 | Validation Loss = 0.7413965463
638306
Epoch 3 | Training Loss = 0.4595836102962494 | Validation Loss = 0.4711650907
9933167
Epoch 500 | Training Loss = 0.011338168755173683 | Validation Loss = 0.013433
423824608326
Epoch 1000 | Training Loss = 0.01107100211083889 | Validation Loss = 0.013250
169344246387
Epoch 1500 | Training Loss = 0.011055151000618935 | Validation Loss = 0.01329
437643289566
Epoch 2000 | Training Loss = 0.011054002679884434 | Validation Loss = 0.01331
313606351614
Epoch 2500 | Training Loss = 0.011053916066884995 | Validation Loss = 0.01331
8805955350399
Epoch 3000 | Training Loss = 0.011053909547626972 | Validation Loss = 0.01332
040410488844
Epoch 3500 | Training Loss = 0.011053909547626972 | Validation Loss = 0.01332
0842757821083
Epoch 4000 | Training Loss = 0.011053908616304398 | Validation Loss = 0.01332
0968486368656
Epoch 4500 | Training Loss = 0.011053909547626972 | Validation Loss = 0.01332
099549472332
Epoch 5000 | Training Loss = 0.011053909547626972 | Validation Loss = 0.01332
1002013981342
```



```
Final Training Loss = 0.011053909547626972 | Final Validation Loss = 0.013321
002013981342
Model MSE Loss for whole dataset = 0.011507327668368816
```

In [ ]:



In [1]:
```
"""
ECGR 5105 - Intro to Machine Learning
Homework 5, Part 2
Phillip Harmon
""";
```

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import torch
from torch import optim, nn
from collections import OrderedDict
```

In [3]:
```
#Normalization Functions
def normalize(x, xmax, xmin):
    return (x - xmin) / (xmax - xmin)

def denormalize(x, xmax, xmin):
    return (x * (xmax - xmin)) + xmin
```

In [4]:
```
#helper for plotting visualization of training data
def training_visual(loss_t, loss_v, model, loss_function, x, y):
    cost_function = loss_function()
    plt.rcParams["figure.figsize"] = (10,5)
    plt.grid()
    plt.xlabel('Epochs')
    plt.ylabel('MSE Loss')
    plt.title('Convergence of Training')
    plt.plot(range(1,len(loss_t) + 1),loss_t, color='blue', label='Training Lo
    plt.plot(range(1,len(loss_t) + 1),loss_v, color='red', label='Validation L
    plt.legend()
    plt.ylim([0.0,0.25])
    plt.show()
    print("Final Training Loss = {} | Final Validation Loss = {}".format(loss_

    x_n = normalize(x, x.max(0,keepdim=True)[0], x.min(0,keepdim=True)[0])
    y_n = normalize(y, y.max(0,keepdim=True)[0], y.min(0,keepdim=True)[0])
    print("Model MSE Loss for whole dataset = {}".format(cost_function(model(x
```

In [5]:
```
#Training Loop Function
def training_loop(x_t, y_t, x_v, y_v, model, loss_function, optimizer, epochs)
    training_loss = []
    validation_loss = []

    cost_function = loss_function()

    for epoch in range(1, epochs + 1):

        loss_t = cost_function( model(x_t), y_t)
        loss_v = cost_function( model(x_v), y_v)

        optimizer.zero_grad()
        loss_t.backward()
        optimizer.step()

        training_loss.append(float(loss_t))
        validation_loss.append(float(loss_v))

        if epoch <= 3 or epoch % 50 == 0:
            print('Epoch {} | Training Loss = {} | Validation Loss = {}'.forma

    return training_loss, validation_loss
```

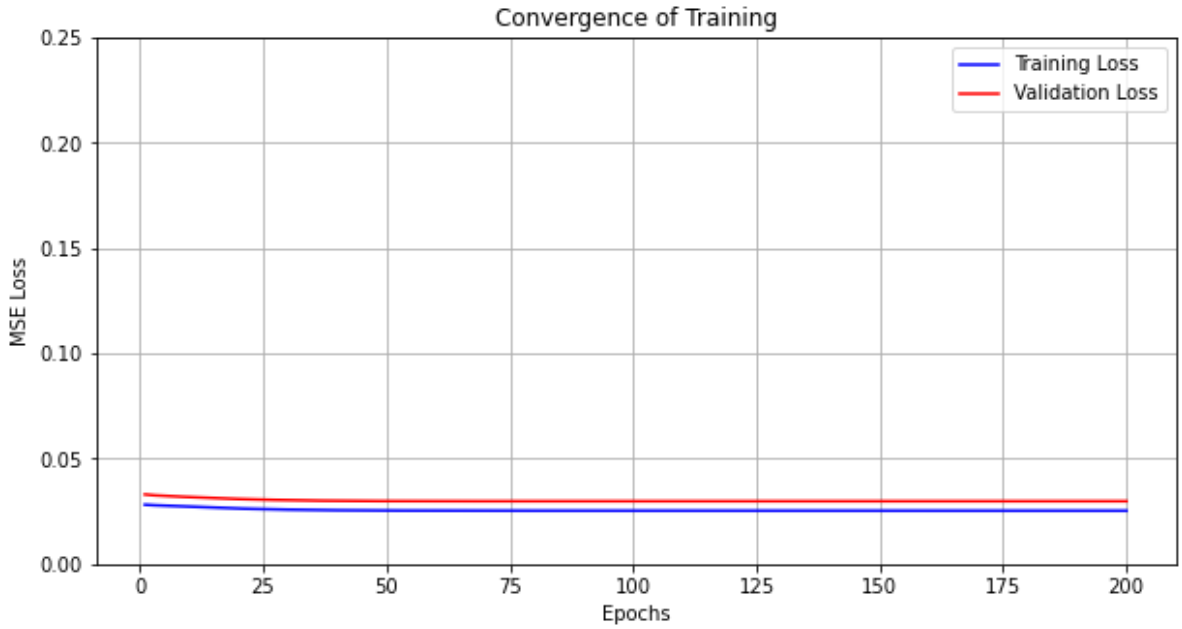
In [6]:
```
#Prepare the inputs

#Read in the CSV into a dataframe
csvData = pd.read_csv("./Housing.csv")
csvCols = len(csvData.columns)
csvRows = len(csvData)

#Collect Data
dataLabels = ['price','area','bedrooms','bathrooms','stories','parking']
data = csvData[dataLabels]

y_raw = data.pop('price').values
x_raw = data.values

y_raw = torch.from_numpy(y_raw)
x_raw = torch.from_numpy(x_raw)

#Cleaning the inputs
x = normalize(x_raw, x_raw.max(0,keepdim=True)[0], x_raw.min(0,keepdim=True)[0
y = normalize(y_raw, y_raw.max(0,keepdim=True)[0], y_raw.min(0,keepdim=True)[0

#Train/Test Split
validation_percent = 0.2
split = int(validation_percent * x.shape[0])
shuffle_index = torch.randperm(x.shape[0])
index_t = shuffle_index[:-split]
index_v = shuffle_index[-split:]
x_t = x[index_t]
y_t = y[index_t]
x_v = x[index_v]
y_v = y[index_v]
```

In [7]:
```
operation = "Adam Optimizer, 1-layer net, LR=1e-3"
#Define Constructs
epochs = 200
learn_rate = 1e-3
neural_net = nn.Sequential(OrderedDict([
    ('Layer_1_Model', nn.Linear(5,8)),
    ('Layer_1_Activation', nn.Tanh()),
    ('Output_Model', nn.Linear(8,1))
    ]))
optimizer = optim.Adam(neural_net.parameters(), lr=learn_rate)
```

In [8]:
```
%%time
print(operation)
#Perform the Training
loss_t, loss_v = training_loop(
    epochs = epochs,
    optimizer = optimizer,
    model = neural_net,
    loss_function = nn.MSELoss,
    x_t = x_t,
    x_v = x_v,
    y_t = y_t,
    y_v = y_v)
```

```
Adam Optimizer, 1-layer net, LR=1e-3
Epoch 1 | Training Loss = 0.02815861813724041 | Validation Loss = 0.033026494
08578873
Epoch 2 | Training Loss = 0.027996810153126717 | Validation Loss = 0.03276772
424578667
Epoch 3 | Training Loss = 0.02789880894124508 | Validation Loss = 0.032603956
75897598
Epoch 50 | Training Loss = 0.025389045476913452 | Validation Loss = 0.0298800
7850944996
Epoch 100 | Training Loss = 0.02530243806540966 | Validation Loss = 0.0298248
95784258842
Epoch 150 | Training Loss = 0.025298453867435455 | Validation Loss = 0.029826
37658715248

G:\GprogramFiles\Conda\lib\site-packages\torch\nn\modules\loss.py:536: UserWa
rning: Using a target size (torch.Size([436])) that is different to the input
size (torch.Size([436, 1])). This will likely lead to incorrect results due t
o broadcasting. Please ensure they have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
G:\GprogramFiles\Conda\lib\site-packages\torch\nn\modules\loss.py:536: UserWa
rning: Using a target size (torch.Size([109])) that is different to the input
size (torch.Size([109, 1])). This will likely lead to incorrect results due t
o broadcasting. Please ensure they have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)


Epoch 200 | Training Loss = 0.025295691564679146 | Validation Loss = 0.029823
73721897602
Wall time: 252 ms
```

In [9]:
```
training_visual(
    loss_t = loss_t,
    loss_v = loss_v,
    model = neural_net,
    loss_function = nn.MSELoss,
    x = x_raw,
    y = y_raw)
```



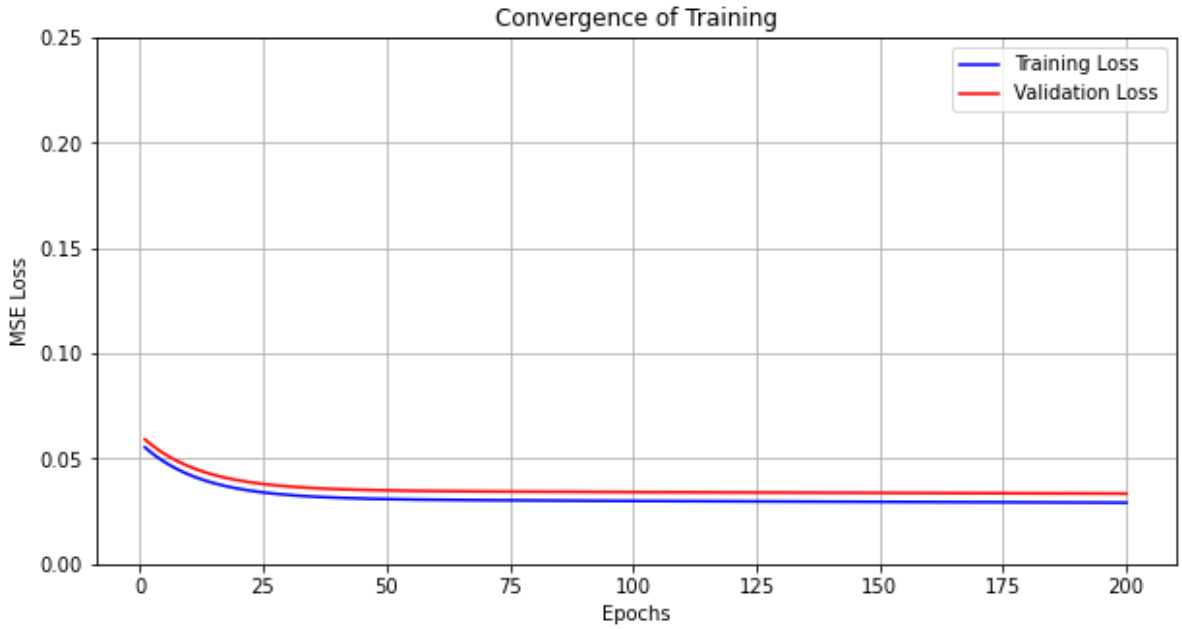

```
Final Training Loss = 0.025295691564679146 | Final Validation Loss = 0.029823
73721897602
Model MSE Loss for whole dataset = 0.02620089240372181

G:\GprogramFiles\Conda\lib\site-packages\torch\nn\modules\loss.py:536: UserWa
rning: Using a target size (torch.Size([545])) that is different to the input
size (torch.Size([545, 1])). This will likely lead to incorrect results due t
o broadcasting. Please ensure they have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
```

In [10]:
```
operation = "SGD Optimizer, 1-layer net, LR=1e-3"
#Define Constructs
epochs = 200
learn_rate = 1e-2
neural_net = nn.Sequential(OrderedDict([
    ('Layer_1_Model', nn.Linear(5,8)),
    ('Layer_1_Activation', nn.Tanh()),
    ('Output_Model', nn.Linear(8,1))
    ]))
optimizer = optim.SGD(neural_net.parameters(), lr=learn_rate)
```

In [11]:
```
%%time
print(operation)
#Perform the Training
loss_t, loss_v = training_loop(
    epochs = epochs,
    optimizer = optimizer,
    model = neural_net,
    loss_function = nn.MSELoss,
    x_t = x_t,
    x_v = x_v,
    y_t = y_t,
    y_v = y_v)
```

```
SGD Optimizer, 1-layer net, LR=1e-3
Epoch 1 | Training Loss = 0.05540982633829117 | Validation Loss = 0.059190109
37213898
Epoch 2 | Training Loss = 0.053433191031217575 | Validation Loss = 0.05722943
693399429
Epoch 3 | Training Loss = 0.05161477252840996 | Validation Loss = 0.055426158
010959625
Epoch 50 | Training Loss = 0.030822787433862686 | Validation Loss = 0.0349380
37395477295
Epoch 100 | Training Loss = 0.029939431697130203 | Validation Loss = 0.034144
54683661461
Epoch 150 | Training Loss = 0.02950666844844818 | Validation Loss = 0.0337643
064558506
Epoch 200 | Training Loss = 0.02912852168083191 | Validation Loss = 0.0334276
7059803009
Wall time: 163 ms
```

In [12]:
```
training_visual(
    loss_t = loss_t,
    loss_v = loss_v,
    model = neural_net,
    loss_function = nn.MSELoss,
    x = x_raw,
    y = y_raw)
```




```
Final Training Loss = 0.02912852168083191 | Final Validation Loss = 0.0334276
7059803009
Model MSE Loss for whole dataset = 0.029983295127749443
```

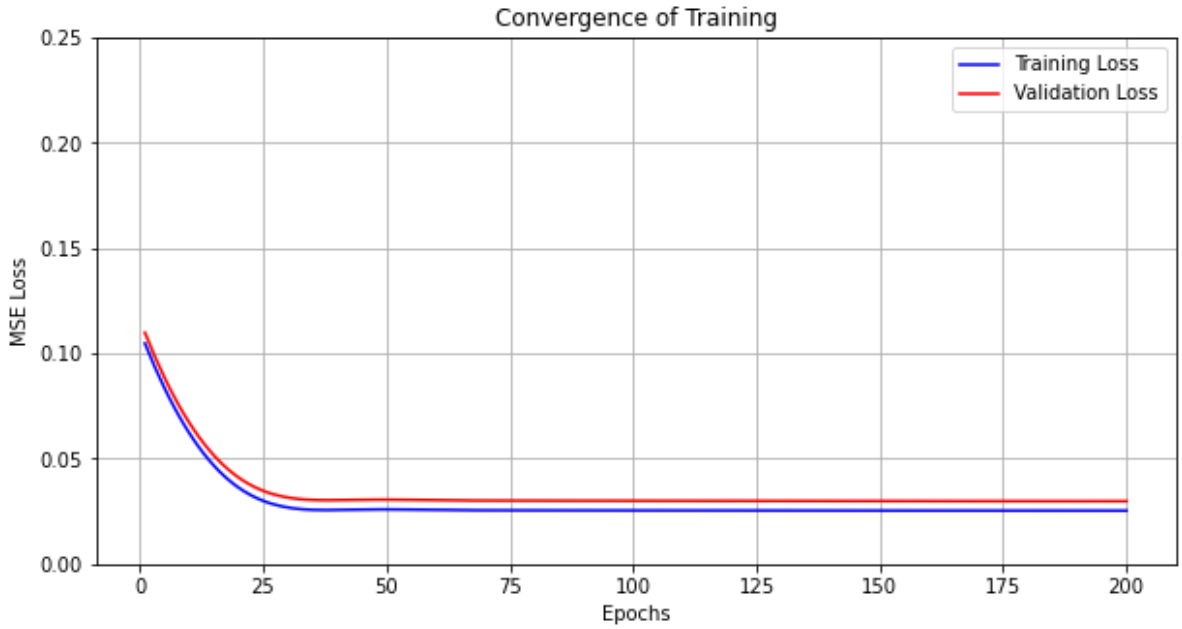
In [13]:
```
#Adam is Better!
#Let's try it for a bigger neural net!
```

In [14]:
```
operation = "Adam Optimizer, 3-layer net [in->8->13->5->out], LR=1e-3"
#Define Constructs
epochs = 200
learn_rate = 1e-3
neural_net = nn.Sequential(OrderedDict([
    ('Layer_1_Model', nn.Linear(5,8)),
    ('Layer_1_Activation', nn.Tanh()),
    ('Layer_2_Model', nn.Linear(8,13)),
    ('Layer_2_Activation', nn.Tanh()),
    ('Layer_3_Model', nn.Linear(13,5)),
    ('Layer_3_Activation', nn.Tanh()),
    ('Output_Model', nn.Linear(5,1))
    ]))
optimizer = optim.Adam(neural_net.parameters(), lr=learn_rate)
```

In [15]:
```
%%time
print(operation)
#Perform the Training
loss_t, loss_v = training_loop(
    epochs = epochs,
    optimizer = optimizer,
    model = neural_net,
    loss_function = nn.MSELoss,
    x_t = x_t,
    x_v = x_v,
    y_t = y_t,
    y_v = y_v)
```

```
Adam Optimizer, 3-layer net [in->8->13->5->out], LR=1e-3
Epoch 1 | Training Loss = 0.10475734621286392 | Validation Loss = 0.109851643
4431076
Epoch 2 | Training Loss = 0.09903859347105026 | Validation Loss = 0.104110471
9042778
Epoch 3 | Training Loss = 0.09356650710105896 | Validation Loss = 0.098616868
25752258
Epoch 50 | Training Loss = 0.025937208905816078 | Validation Loss = 0.0305798
0164885521
Epoch 100 | Training Loss = 0.02541847713291645 | Validation Loss = 0.0299924
96594786644
Epoch 150 | Training Loss = 0.025350719690322876 | Validation Loss = 0.029882
799834012985
Epoch 200 | Training Loss = 0.025321977213025093 | Validation Loss = 0.029842
05074608326
Wall time: 317 ms
```

In [16]:
```
training_visual(
    loss_t = loss_t,
    loss_v = loss_v,
    model = neural_net,
    loss_function = nn.MSELoss,
    x = x_raw,
    y = y_raw)
```



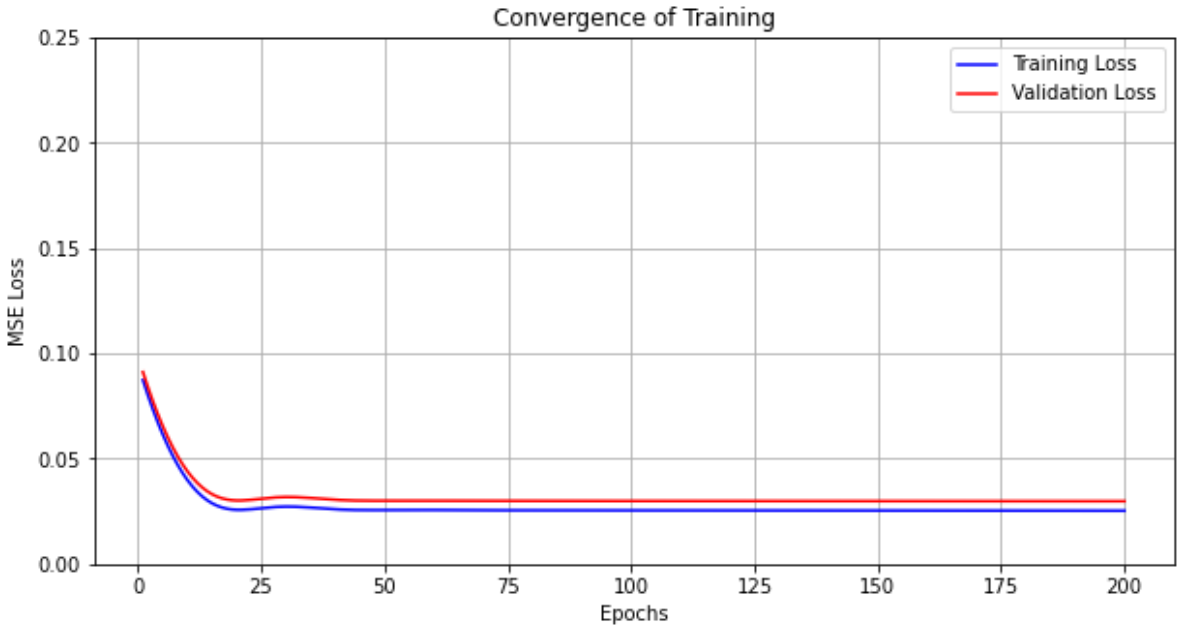

```
Final Training Loss = 0.025321977213025093 | Final Validation Loss = 0.029842
05074608326
Model MSE Loss for whole dataset = 0.026226604357361794
```

In [17]:
```
operation = "Adam Optimizer, 3-layer net [in->13->21->5->out], LR=1e-3"
#Define Constructs
epochs = 200
learn_rate = 1e-3
neural_net = nn.Sequential(OrderedDict([
    ('Layer_1_Model', nn.Linear(5,13)),
    ('Layer_1_Activation', nn.Tanh()),
    ('Layer_2_Model', nn.Linear(13,21)),
    ('Layer_2_Activation', nn.Tanh()),
    ('Layer_3_Model', nn.Linear(21,5)),
    ('Layer_3_Activation', nn.Tanh()),
    ('Output_Model', nn.Linear(5,1))
    ]))
optimizer = optim.Adam(neural_net.parameters(), lr=learn_rate)
```

In [18]:
```
%%time
print(operation)
#Perform the Training
loss_t, loss_v = training_loop(
    epochs = epochs,
    optimizer = optimizer,
    model = neural_net,
    loss_function = nn.MSELoss,
    x_t = x_t,
    x_v = x_v,
    y_t = y_t,
    y_v = y_v)
```

```
Adam Optimizer, 3-layer net [in->13->21->5->out], LR=1e-3
Epoch 1 | Training Loss = 0.08736804872751236 | Validation Loss = 0.091150604
18844223
Epoch 2 | Training Loss = 0.08022507280111313 | Validation Loss = 0.084061607
71846771
Epoch 3 | Training Loss = 0.07354738563299179 | Validation Loss = 0.077434360
98098755
Epoch 50 | Training Loss = 0.025625130161643028 | Validation Loss = 0.0300666
7084991932
Epoch 100 | Training Loss = 0.0254366397857666 | Validation Loss = 0.02992679
923772812
Epoch 150 | Training Loss = 0.025343503803014755 | Validation Loss = 0.029858
145862817764
Epoch 200 | Training Loss = 0.025298304855823517 | Validation Loss = 0.029821
33999466896
Wall time: 333 ms
```

In [19]:
```
training_visual(
    loss_t = loss_t,
    loss_v = loss_v,
    model = neural_net,
    loss_function = nn.MSELoss,
    x = x_raw,
    y = y_raw)
```




```
Final Training Loss = 0.025298304855823517 | Final Validation Loss = 0.029821
33999466896
Model MSE Loss for whole dataset = 0.026202790439128876
```

In [ ]: