Question 1

Load the data using pandas

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os

%matplotlib inline
```

Read and split data into train(1-3), test(1-3)

In [2]:
```
os.chdir('C:/Users/gilmo/OneDrive/Documents/EEL 5840')

df=pd.read_csv('beerfoam2a.csv')

df.drop(df.columns[2:5],axis=1,inplace=True)

df1 = df.iloc[0:15]
df1_train = df.iloc[0:12]  #12 samples
df1_test = df.iloc[12:15]  #3 samples

df2 = df.iloc[15:30]
df2_train = df.iloc[15:27]
df2_test = df.iloc[27:30]

df3 = df.iloc[30:]
df3_train = df.iloc[30:42]
df3_test = df.iloc[42:]

#True curves
x1_true = df1.foamTime.to_numpy()
t1_true = df1.foamHt.to_numpy()

x2_true = df2.foamTime.to_numpy()
t2_true = df2.foamHt.to_numpy()

x3_true = df3.foamTime.to_numpy()
t3_true = df3.foamHt.to_numpy()
```

Generate X1-X3 training sets

In [3]:
```
M=4
x1_train = df1_train.foamTime.to_numpy()
t1_train = df1_train.foamHt.to_numpy()
X1_train = np.array([x1_train**i for i in range(M+1)]).T

x2_train = df2_train.foamTime.to_numpy()
t2_train = df2_train.foamHt.to_numpy()
X2_train = np.array([x2_train**i for i in range(M+1)]).T
```

```
x3_train = df3_train.foamTime.to_numpy()
t3_train = df3_train.foamHt.to_numpy()
X3_train = np.array([x3_train**i for i in range(M+1)]).T
```

Generate X1-X3 test sets

In [4]:

```
M= 4

x1_test = df1_test.foamTime.to_numpy()
t1_test = df1_test.foamHt.to_numpy()
X1_test = np.array([x1_test**i for i in range(M+1)]).T

x2_test = df2_test.foamTime.to_numpy()
t2_test = df2_test.foamHt.to_numpy()
X2_test = np.array([x2_test**i for i in range(M+1)]).T

x3_test = df3_test.foamTime.to_numpy()
t3_test = df3_test.foamHt.to_numpy()
X3_test = np.array([x3_test**i for i in range(M+1)]).T
```

1. Build and train a polynomial regression model for each beer brand with model order 4

In [5]:

```
def PolynomialRegression(x,t,M):
    '''Fit a polynomial of order M to the data input data x and desire values t'''

    # Feature Matrix X
    X=np.array([x**i for i in range(M+1)]).T #create feature set phi(xi) of polynomials
                                            #the underlying data can be modeled by an

    # Coefficients w
    w = np.linalg.inv(X.T@X)@X.T@t  #inv(X.T@X)@X.T is the pseudoinverse of X

    # Model prediction, y = Xw
    y=X@w

    return w,y
```

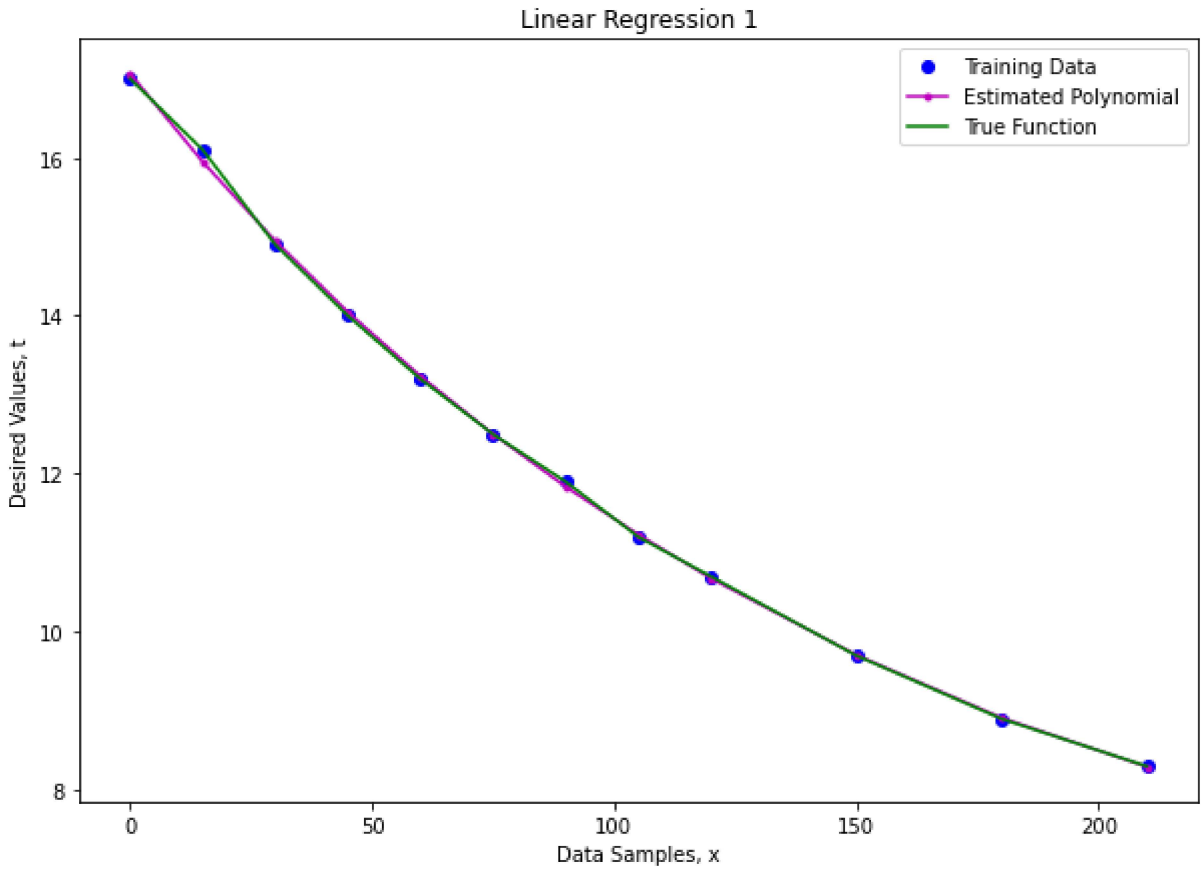
Train polynomial regression models (1-3)

Model 1 (Brand 1)

In [6]:

```
M = 4 #select model order
w1, y1_train = PolynomialRegression(x1_train,t1_train,M)

plt.figure(figsize=(10,7))
plt.plot(x1_train,t1_train,'bo', label='Training Data')
plt.plot(x1_train,y1_train,'.-m', label = 'Estimated Polynomial')
plt.plot(x1_train,t1_train,'g', label = 'True Function')
plt.legend()
plt.xlabel('Data Samples, x')
```

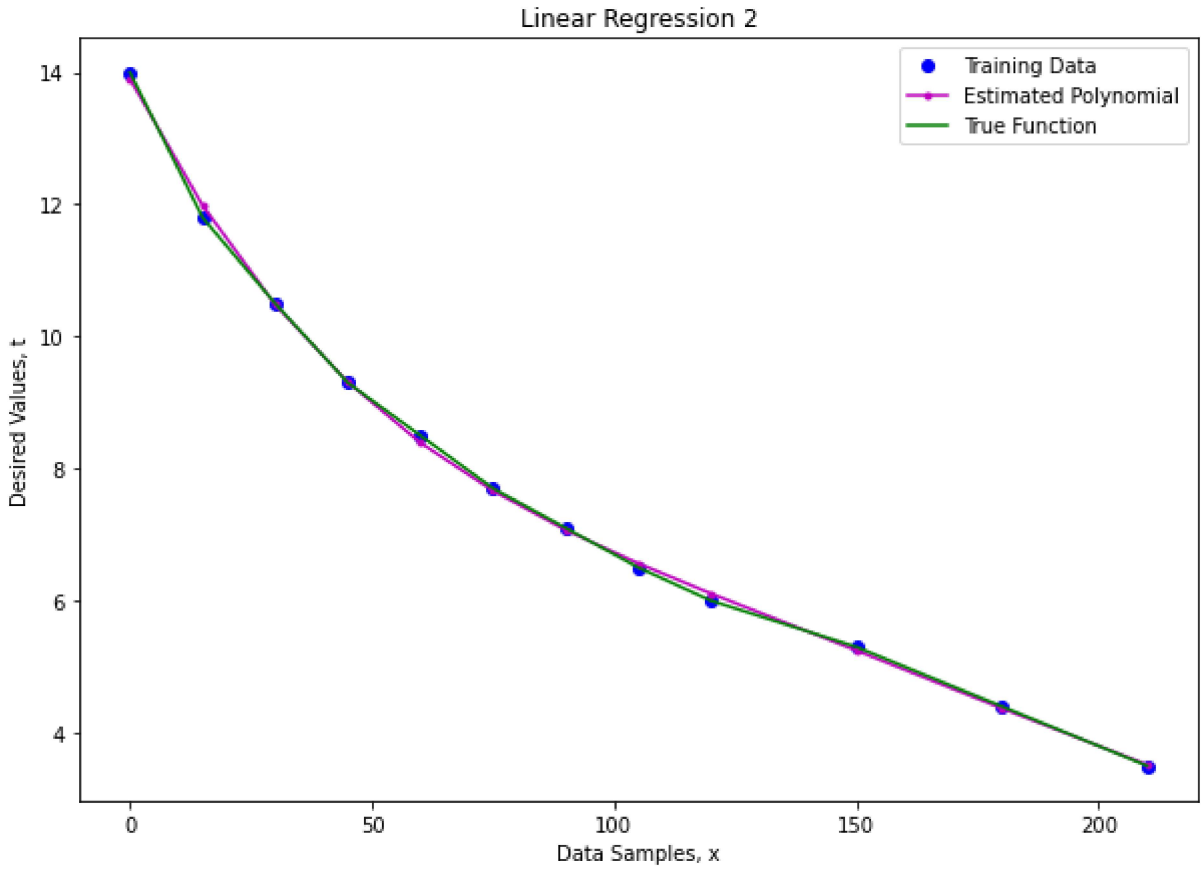
```
plt.ylabel('Desired Values, t');
plt.title('Linear Regression 1');
```



Model 2 (Brand 2)

In [7]:

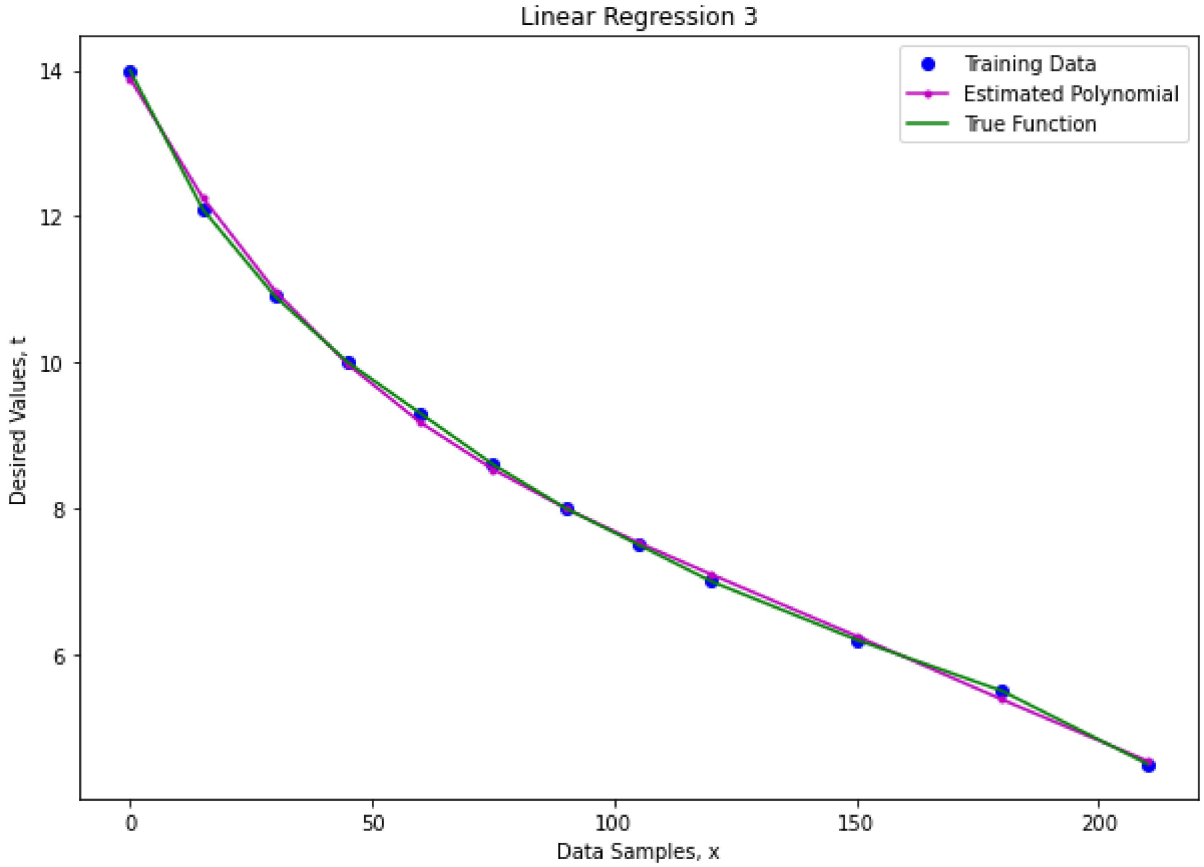
```
M = 4 #select model order
w2, y2_train = PolynomialRegression(x2_train,t2_train,M)

plt.figure(figsize=(10,7))
plt.plot(x2_train,t2_train,'bo', label='Training Data')
plt.plot(x2_train,y2_train,'.-m', label = 'Estimated Polynomial')
plt.plot(x2_train,t2_train,'g', label = 'True Function')
plt.legend()
plt.xlabel('Data Samples, x')
plt.ylabel('Desired Values, t');
plt.title('Linear Regression 2');
```



Model 3 (Brand 3)

In [8]:

```
M = 4 #select model order
w3, y3_train = PolynomialRegression(x3_train,t3_train,M)

plt.figure(figsize=(10,7))
plt.plot(x3_train,t3_train,'bo', label='Training Data')
plt.plot(x3_train,y3_train,'.-m', label = 'Estimated Polynomial')
plt.plot(x3_train,t3_train,'g', label = 'True Function')
plt.legend()
plt.xlabel('Data Samples, x')
plt.ylabel('Desired Values, t');
plt.title('Linear Regression 3');
```



Polynomial regression models (1-3)

Polynomial regression with regularization

In [9]:
```
def PolynomialRegression_reg(x,t,M,lam):
    # Compute feature matrix X with polynomial features
    X = np.array([x**m for m in range(M+1)]).T
    # Compute the solution for the parameters w
    w = np.linalg.inv(X.T@X + lam*np.eye(M+1))@X.T@t
    # Compute model prediction
    y = X@w
    return w, y
```

In [10]:
```
def PolynomialRegression_test(x,M,w):
    # Feature matrix for test set
    X = np.array([x**m for m in range(M+1)]).T

    # Prediction for test set
    y = X@w

    return y
```

Test Polynomial Regression w/ Regurlarization (Brand 1)

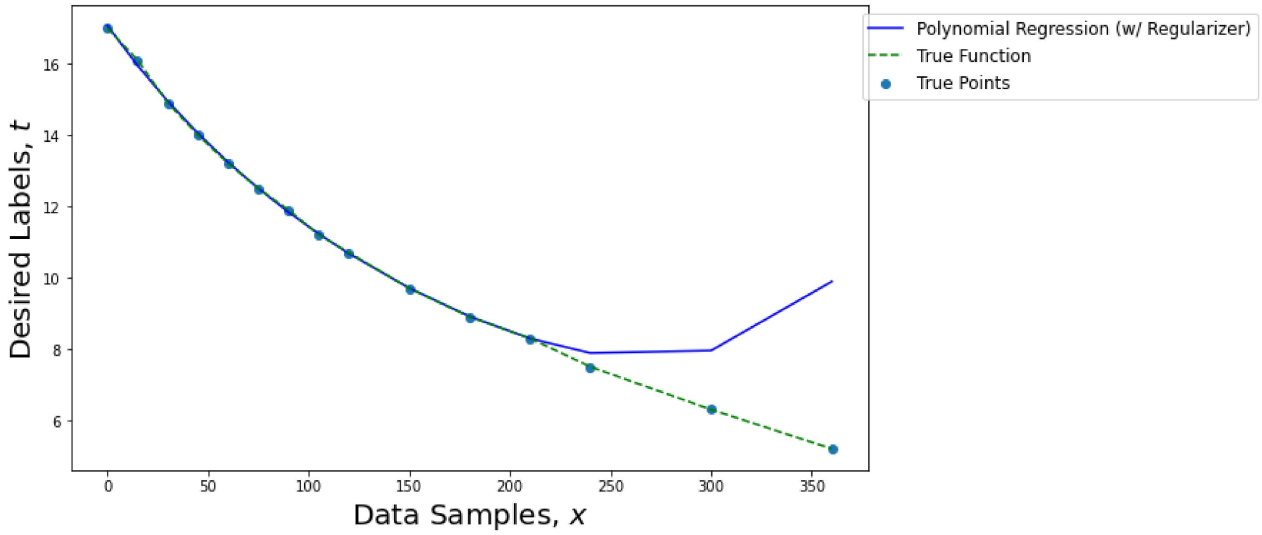
In [11]:
```
M = 4
test1_errors =[]
```

```
lams = [0.0]
lambdas = [[i] for i in lams]
for lams in lambdas:
    w1reg, y1reg = PolynomialRegression_reg(x1_train,t1_train,M,lams)
    test1_errors.append(0.5*np.sum((t1_train-y1reg)**2))
l1 = lambdas[np.argmin(test1_errors)]
print('Best Lambda:',l1)
```

```
Best Lambda: [0.0]
```

In [12]:

```
y1_pred = PolynomialRegression_test(x1_true, M, w1reg)

fig=plt.figure(figsize=(10,6))
plt.scatter(x1_true,t1_true, label='True Points')
plt.plot(x1_true, y1_pred, 'b',label = 'Polynomial Regression (w/ Regularizer)')
plt.plot(x1_true, t1_true,'--g', label = 'True Function')
plt.legend(bbox_to_anchor=(1.5, 1),fontsize=12,ncol=1)
plt.xlabel('Data Samples, $x$', fontsize=20)
plt.ylabel('Desired Labels, $t$', fontsize=20);

mse1_poly = 0.5*np.sum((t1_true-y1_pred)**2)
```



Test Polynomial Regression w/ Regurlarization (Brand 2)

In [13]:

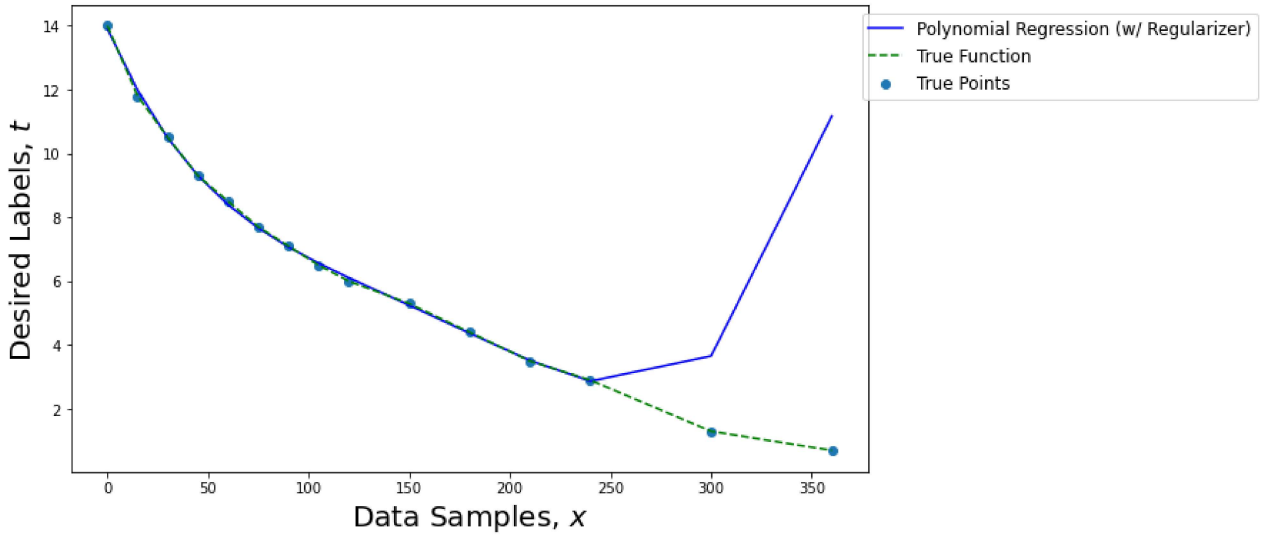
```
M = 4
test2_errors =[]
lams = [0.0]
lambdas = [[i] for i in lams]
for lams in lambdas:
    w2reg, y2reg = PolynomialRegression_reg(x2_train,t2_train,M,lams)
    test2_errors.append(0.5*np.sum((t2_train-y2reg)**2))
l2 = lambdas[np.argmin(test2_errors)]
print('Best Lambda:',l2)
```

```
Best Lambda: [0.0]
```

In [14]:

```
y2_pred = PolynomialRegression_test(x2_true, M, w2reg)

fig=plt.figure(figsize=(10,6))
```

```
plt.scatter(x2_true,t2_true, label='True Points')
plt.plot(x2_true, y2_pred, 'b',label = 'Polynomial Regression (w/ Regularizer)')
plt.plot(x2_true, t2_true,'--g', label = 'True Function')
plt.legend(bbox_to_anchor=(1.5, 1),fontsize=12,ncol=1)
plt.xlabel('Data Samples, $x$', fontsize=20)
plt.ylabel('Desired Labels, $t$', fontsize=20);

mse2_poly = 0.5*np.sum((t2_true-y2_pred)**2)
```



Test Polynomial Regression w/ Regurlarization (Brand 3)

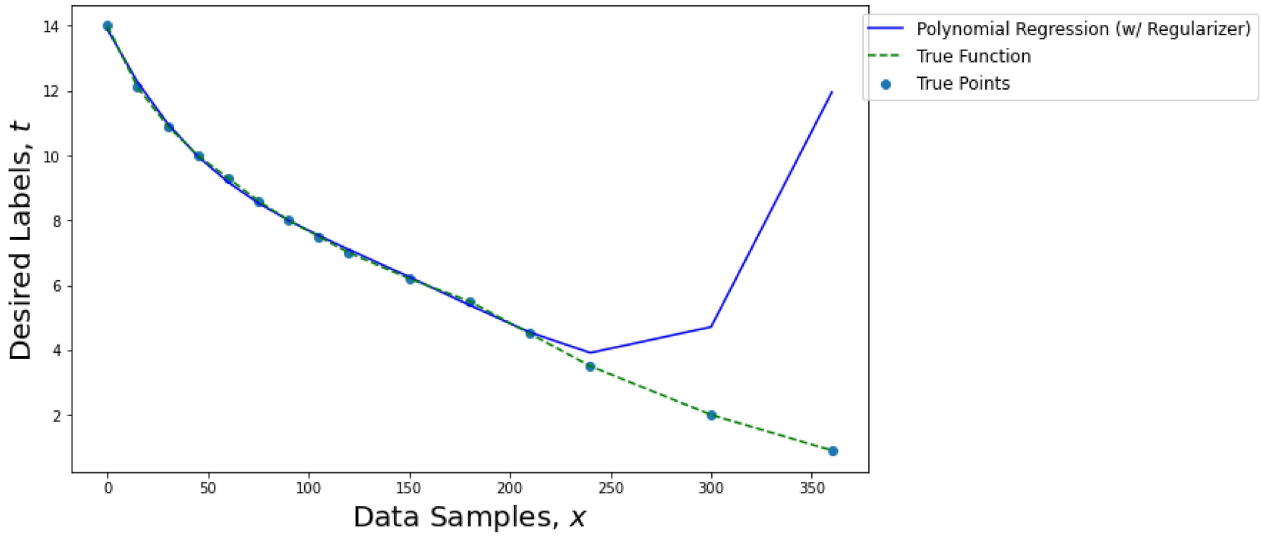
In [15]:

```
M = 4
test3_errors =[]
lams = [0.0]
lambdas = [[i] for i in lams]
for lams in lambdas:
    w3reg, y3reg = PolynomialRegression_reg(x3_train,t3_train,M,lams)
    test3_errors.append(0.5*np.sum((t3_train-y3reg)**2))
l3 = lambdas[np.argmin(test3_errors)]
print('Best Lambda:',l3)
```

```
Best Lambda: [0.0]
```

In [16]:

```
y3_pred = PolynomialRegression_test(x3_true, M, w3reg)

fig=plt.figure(figsize=(10,6))
plt.scatter(x3_true,t3_true, label='True Points')
plt.plot(x3_true, y3_pred, 'b',label = 'Polynomial Regression (w/ Regularizer)')
plt.plot(x3_true, t3_true,'--g', label = 'True Function')
plt.legend(bbox_to_anchor=(1.5, 1),fontsize=12,ncol=1)
plt.xlabel('Data Samples, $x$', fontsize=20)
plt.ylabel('Desired Labels, $t$', fontsize=20);


mse3_poly = 0.5*np.sum((t3_true-y3_pred)**2)
```

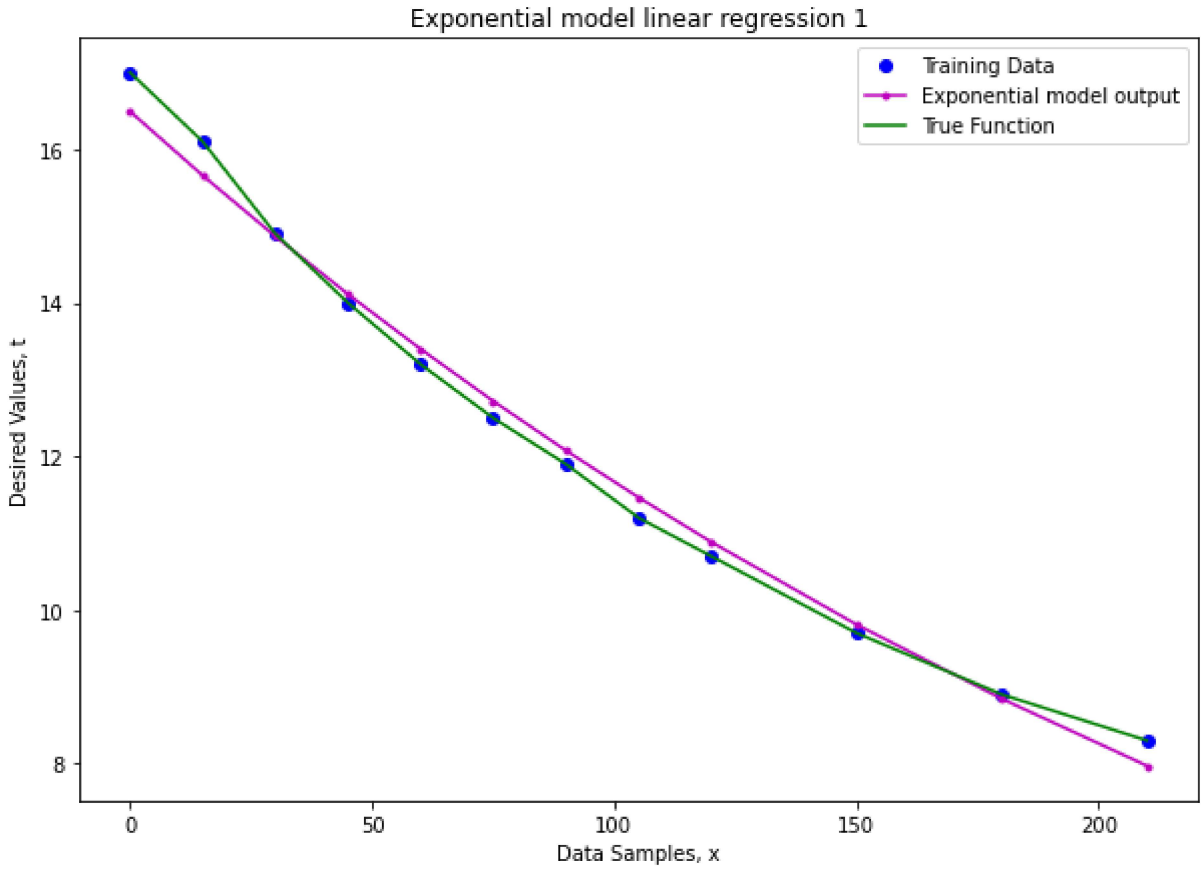


Define Exponential Model

In [17]:
```
def LinRegression_Exp(x,t):
    '''Fit a gaussian of order M to the data input data x and desire values t'''

    # Feature Matrix X
    ones = np.ones(len(x))
    X = np.array([ones, x]).T

    # Coefficients w
    w = np.linalg.inv(X.T@X)@X.T@np.log(t)

    # Model prediction, y = Xw
    y=np.exp(X@w)

    return w,y
```

Exp Model 1 (Brand 1)

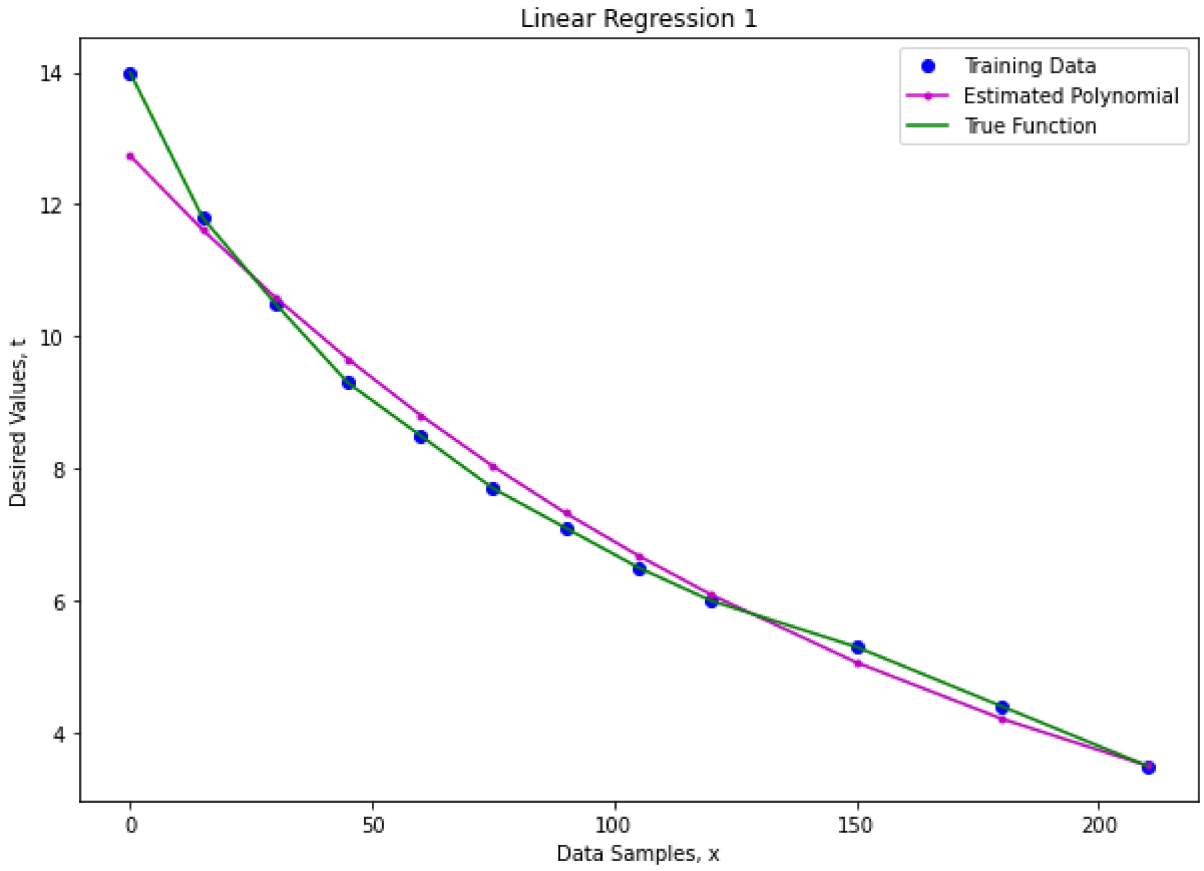
In [18]:
```
w1,y1 = LinRegression_Exp(x1_train,t1_train)

plt.figure(figsize=(10,7))
plt.plot(x1_train,t1_train,'bo', label='Training Data')
plt.plot(x1_train,y1,'.-m', label = 'Exponential model output')
plt.plot(x1_train,t1_train,'g', label = 'True Function')
plt.legend()
plt.xlabel('Data Samples, x')
plt.ylabel('Desired Values, t');
plt.title('Exponential model linear regression 1');
```



Exp Model 2 (Brand 2)

In [19]:

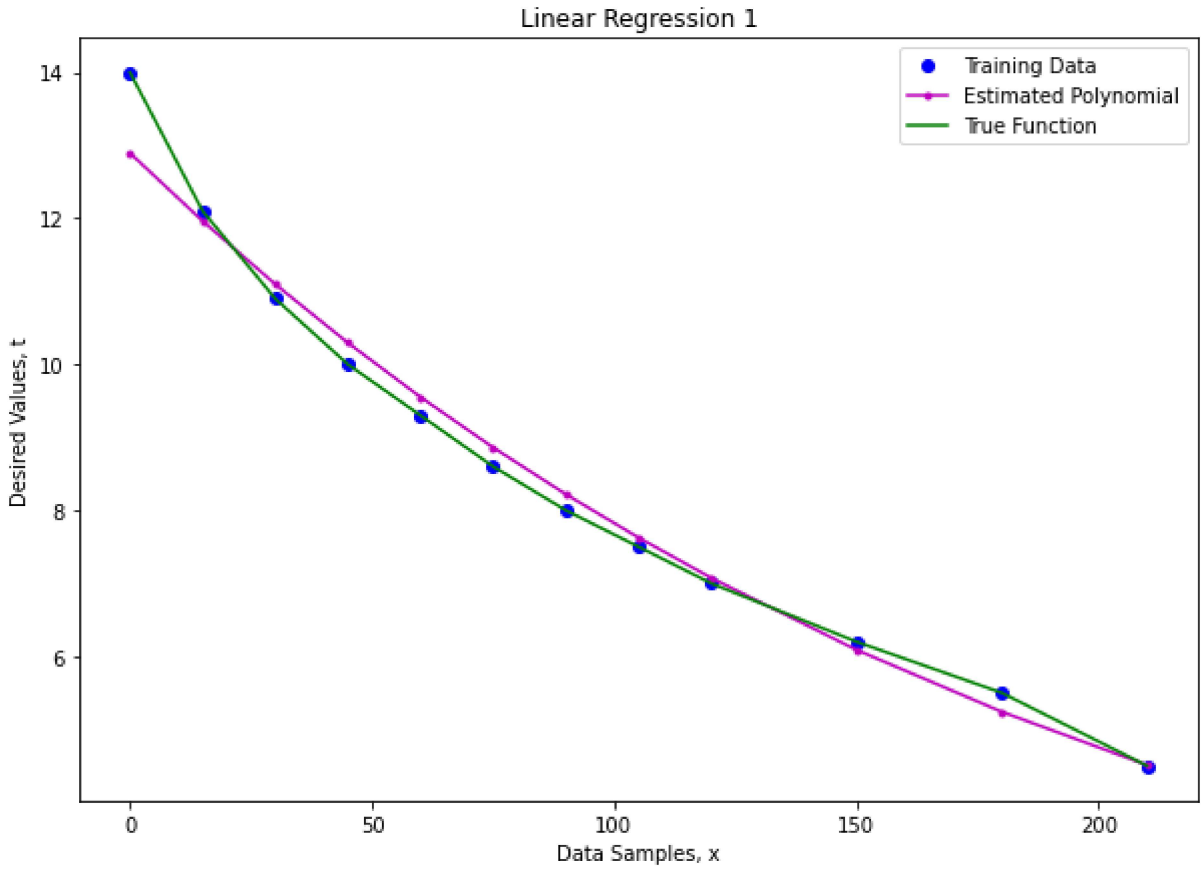
```
w2,y2 = LinRegression_Exp(x2_train,t2_train)

plt.figure(figsize=(10,7))
plt.plot(x2_train,t2_train,'bo', label='Training Data')
plt.plot(x2_train,y2,'.-m', label = 'Estimated Polynomial')
plt.plot(x2_train,t2_train,'g', label = 'True Function')
plt.legend()
plt.xlabel('Data Samples, x')
plt.ylabel('Desired Values, t');
plt.title('Linear Regression 1');
```



Exp Model 3 (Brand 3)

In [20]:

```
w3,y3 = LinRegression_Exp(x3_train,t3_train)

plt.figure(figsize=(10,7))
plt.plot(x3_train,t3_train,'bo', label='Training Data')
plt.plot(x3_train,y3,'.-m', label = 'Estimated Polynomial')
plt.plot(x3_train,t3_train,'g', label = 'True Function')
plt.legend()
plt.xlabel('Data Samples, x')
plt.ylabel('Desired Values, t');
plt.title('Linear Regression 1');
```



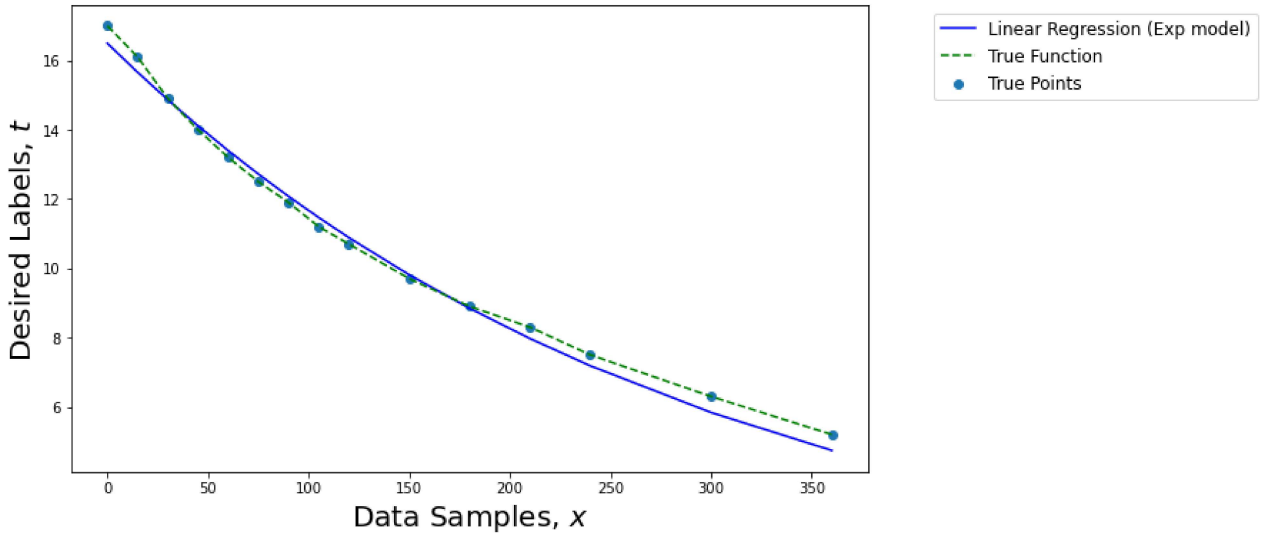
TEST Exp Model 1 (Brand 1)

In [21]:

```
def LinRegressionExp_test(x,w):

    # Feature Matrix X
    ones = np.ones(len(x))
    X = np.array([ones, x]).T

    # Model prediction, y = Xw
    y=np.exp(X@w)

    return y
```

In [22]:

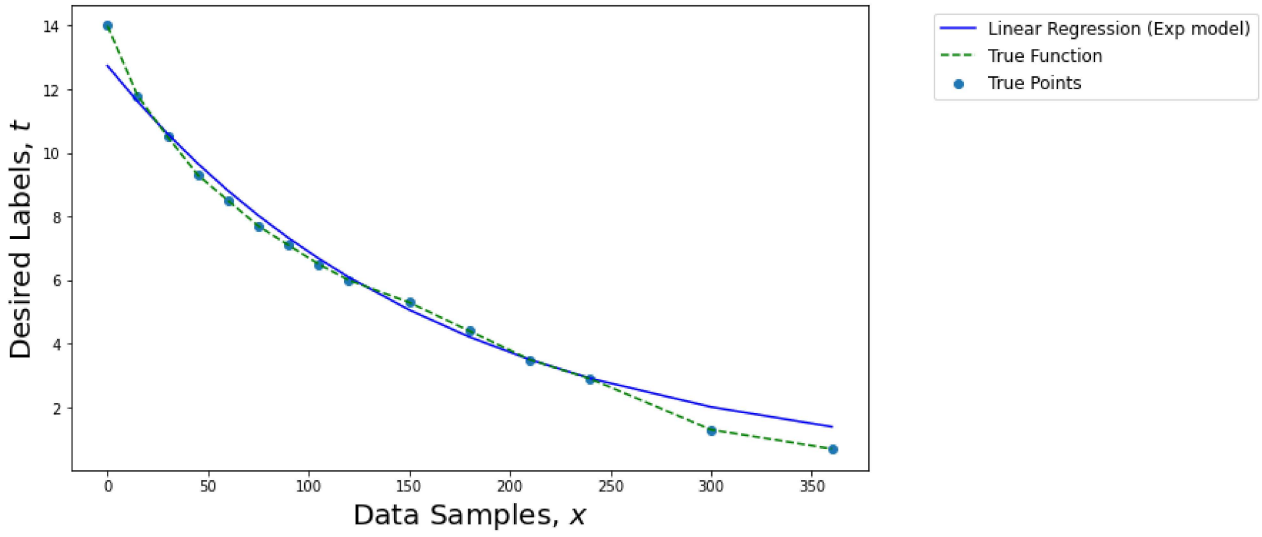
```
y1_pred_exp = LinRegressionExp_test(x1_true,w1)

fig=plt.figure(figsize=(10,6))
plt.scatter(x1_true,t1_true, label='True Points')
plt.plot(x1_true, y1_pred_exp, 'b',label = 'Linear Regression (Exp model)')
plt.plot(x1_true, t1_true,'--g', label = 'True Function')
plt.legend(bbox_to_anchor=(1.5, 1),fontsize=12,ncol=1)
plt.xlabel('Data Samples, $x$', fontsize=20)
plt.ylabel('Desired Labels, $t$', fontsize=20);

mse1_exp = 0.5*np.sum((t1_true-y1_pred_exp)**2)
```



TEST Exp Model 2 (Brand 2)

In [23]:

```
y2_pred_exp = LinRegressionExp_test(x2_true,w2)

fig=plt.figure(figsize=(10,6))
plt.scatter(x2_true,t2_true, label='True Points')
plt.plot(x2_true, y2_pred_exp, 'b',label = 'Linear Regression (Exp model)')
plt.plot(x2_true, t2_true,'--g', label = 'True Function')
plt.legend(bbox_to_anchor=(1.5, 1),fontsize=12,ncol=1)
plt.xlabel('Data Samples, $x$', fontsize=20)
plt.ylabel('Desired Labels, $t$', fontsize=20);

mse2_exp = 0.5*np.sum((t2_true-y2_pred_exp)**2)
```

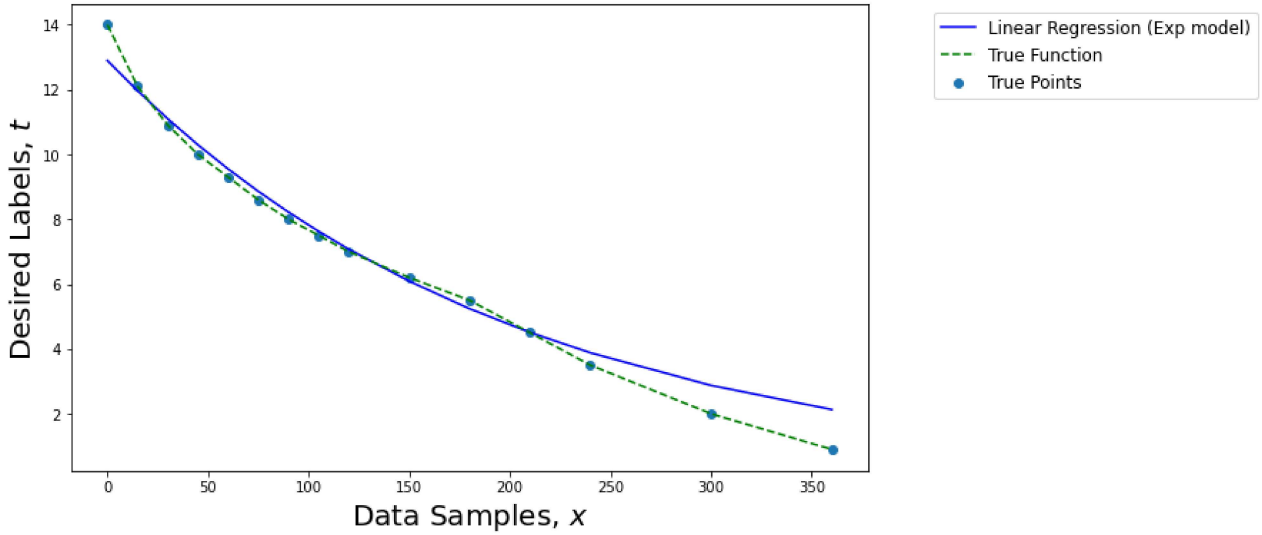


TEST Exp Model 3 (Brand 3)

In [24]:

```
y3_pred_exp = LinRegressionExp_test(x3_true,w3)

fig=plt.figure(figsize=(10,6))
plt.scatter(x3_true,t3_true, label='True Points')
plt.plot(x3_true, y3_pred_exp, 'b',label = 'Linear Regression (Exp model)')
plt.plot(x3_true, t3_true,'--g', label = 'True Function')
```

```
plt.legend(bbox_to_anchor=(1.5, 1),fontsize=12,ncol=1)
plt.xlabel('Data Samples, $x$', fontsize=20)
plt.ylabel('Desired Labels, $t$', fontsize=20);

mse3_exp = 0.5*np.sum((t3_true-y3_pred_exp)**2)
```

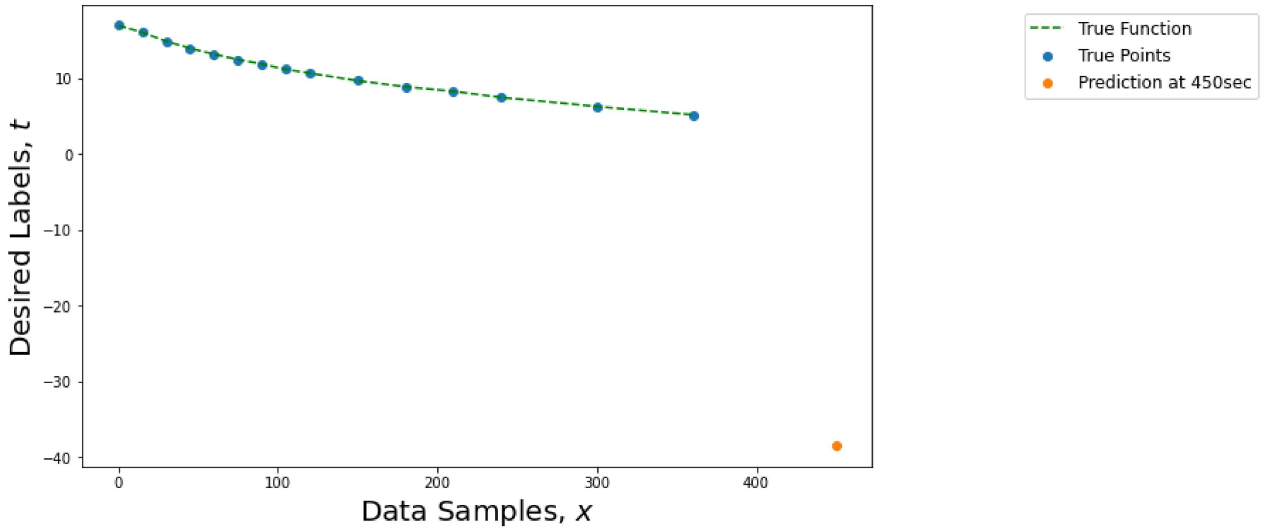


3. Predict height at 450sec

Polynomial Regression Model 1 (Brand 1)

In [25]:

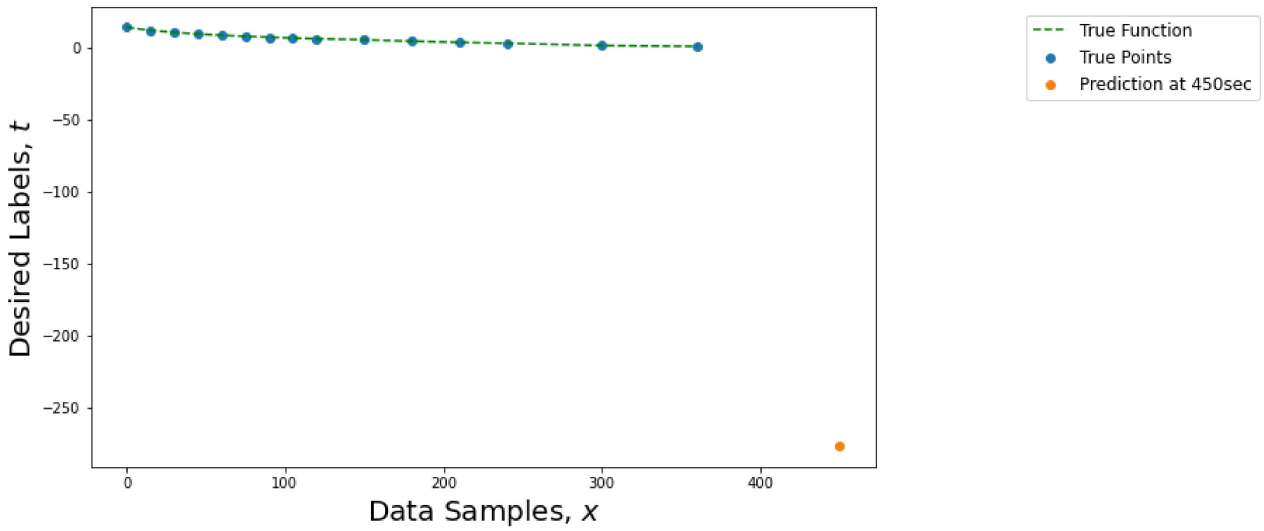
```
#PREDICT AT 450sec
# x1_new = np.linspace(0,450,15,dtype=int)

x1_new = np.array([450])
X1_pred = np.array([x1_new**m for m in range(M+1)]).T

#X1_new = np.array([x1_new**m for m in range(M+1)]).T
w1pred, y1reg = PolynomialRegression_reg(x1_train,t1_train,M,l1)
y1_pred_reg = X1_pred@w1pred

fig=plt.figure(figsize=(10,6))
plt.scatter(x1_true,t1_true, label='True Points')
plt.scatter(x1_new, y1_pred_reg,label = 'Prediction at 450sec')
plt.plot(x1_true,t1_true,'--g', label = 'True Function')
plt.legend(bbox_to_anchor=(1.5, 1),fontsize=12,ncol=1)
plt.xlabel('Data Samples, $x$', fontsize=20)
plt.ylabel('Desired Labels, $t$', fontsize=20);
```



Polynomial Regression Model 2 (Brand 2)

In [26]:

```
#PREDICT AT 450sec using Model 2

x2_new = np.array([450])
X2_pred = np.array([x2_new**m for m in range(M+1)]).T

#X1_new = np.array([x1_new**m for m in range(M+1)]).T
w2pred, y2reg = PolynomialRegression_reg(x2_train,t2_train,M,l2)
y2_pred_reg = X2_pred@w2pred

fig=plt.figure(figsize=(10,6))
plt.scatter(x2_true,t2_true, label='True Points')
plt.scatter(x2_new, y2_pred_reg,label = 'Prediction at 450sec')
plt.plot(x2_true,t2_true,'--g', label = 'True Function')
plt.legend(bbox_to_anchor=(1.5, 1),fontsize=12,ncol=1)
plt.xlabel('Data Samples, $x$', fontsize=20)
plt.ylabel('Desired Labels, $t$', fontsize=20);
```

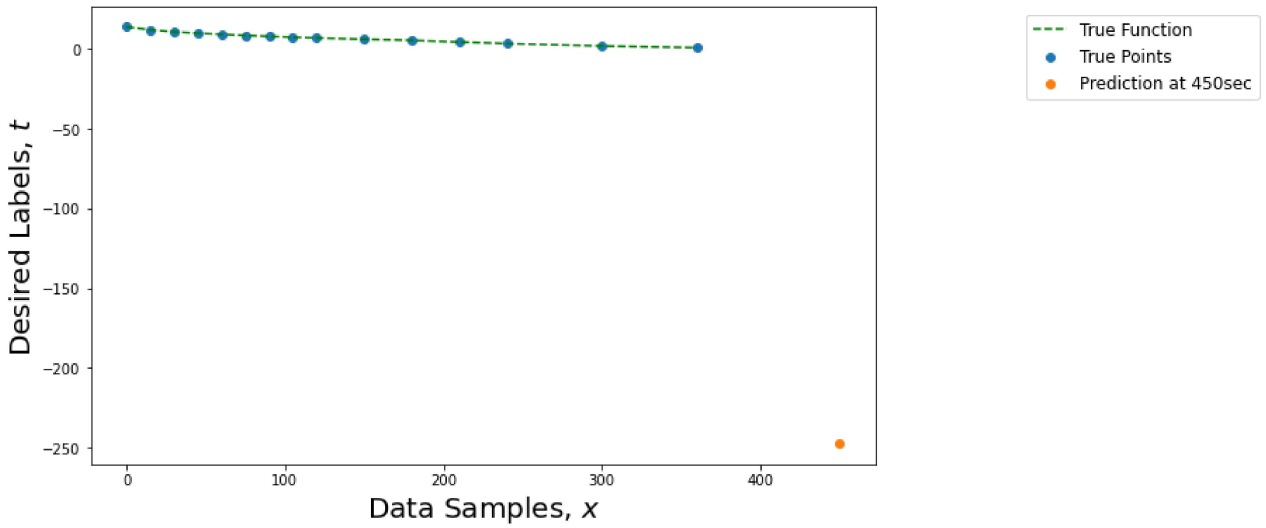


Polynomial Regression Model 3 (Brand 3)

In [27]:

```
#PREDICT AT 450sec
```

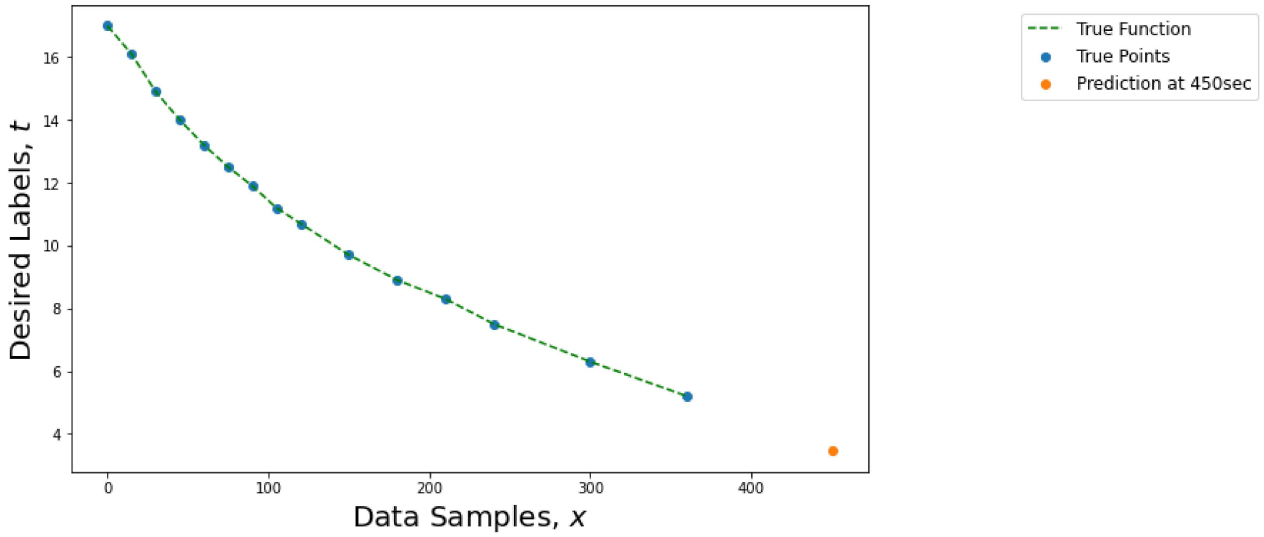
```
# x1_new = np.linspace(0,450,15,dtype=int)

x3_new = np.array([450])
X3_pred = np.array([x3_new**m for m in range(M+1)]).T

#X1_new = np.array([x1_new**m for m in range(M+1)]).T
w3pred, y3reg = PolynomialRegression_reg(x3_train,t3_train,M,l1)
y3_pred_reg = X3_pred@w3pred

fig=plt.figure(figsize=(10,6))
plt.scatter(x3_true,t3_true, label='True Points')
plt.scatter(x3_new, y3_pred_reg,label = 'Prediction at 450sec')
plt.plot(x3_true,t3_true,'--g', label = 'True Function')
plt.legend(bbox_to_anchor=(1.5, 1),fontsize=12,ncol=1)
plt.xlabel('Data Samples, $x$', fontsize=20)
plt.ylabel('Desired Labels, $t$', fontsize=20);
```



Exp Model 1 (Brand 1)

In [28]:

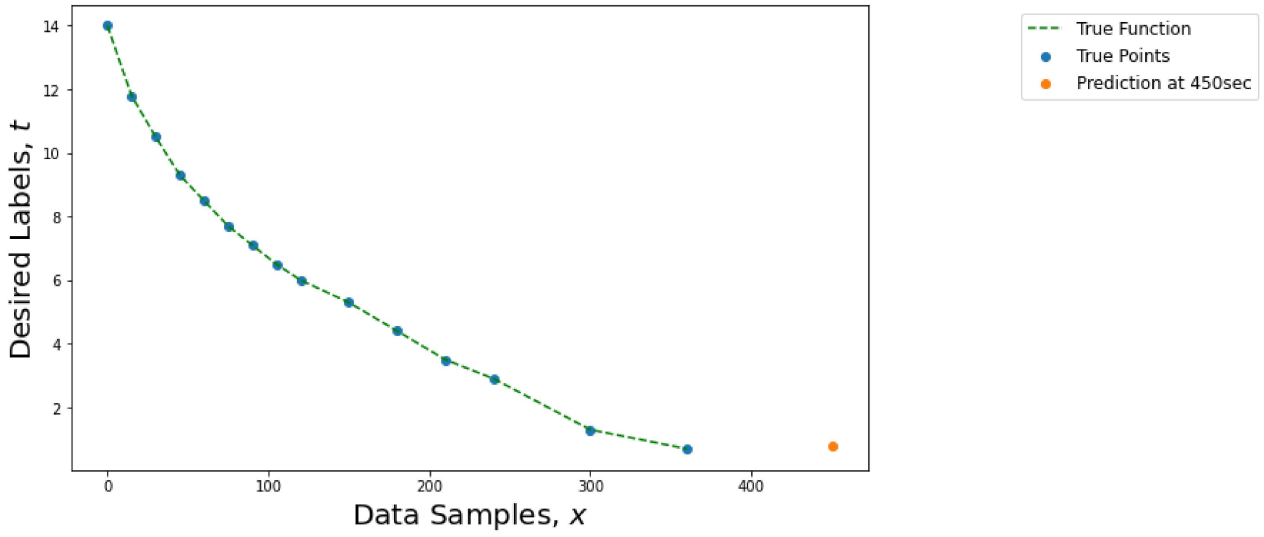
```
x1_new = np.array([450])

y1reg = LinRegressionExp_test(x1_new,w1)

fig=plt.figure(figsize=(10,6))
plt.scatter(x1_true,t1_true, label='True Points')
plt.scatter(x1_new, y1reg,label = 'Prediction at 450sec')
plt.plot(x1_true,t1_true,'--g', label = 'True Function')
plt.legend(bbox_to_anchor=(1.5, 1),fontsize=12,ncol=1)
plt.xlabel('Data Samples, $x$', fontsize=20)
plt.ylabel('Desired Labels, $t$', fontsize=20);
```



Exp Model 2 (Brand 2)

In [29]:

```
x2_new = np.array([450])

y2reg = LinRegressionExp_test(x2_new,w2)

fig=plt.figure(figsize=(10,6))
plt.scatter(x2_true,t2_true, label='True Points')
plt.scatter(x2_new, y2reg,label = 'Prediction at 450sec')
plt.plot(x2_true,t2_true,'--g', label = 'True Function')
plt.legend(bbox_to_anchor=(1.5, 1),fontsize=12,ncol=1)
plt.xlabel('Data Samples, $x$', fontsize=20)
plt.ylabel('Desired Labels, $t$', fontsize=20);
```

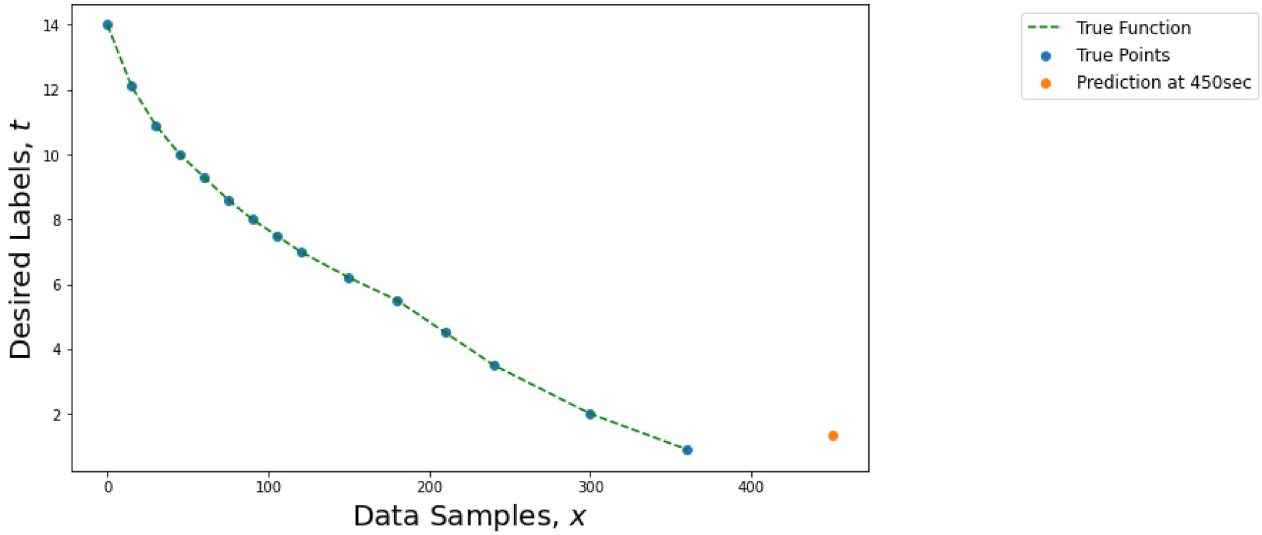


Exp Model 3 (Brand 3)

In [30]:

```
x3_new = np.array([450])

y3reg = LinRegressionExp_test(x3_new,w3)

fig=plt.figure(figsize=(10,6))
plt.scatter(x3_true,t3_true, label='True Points')
```

```
plt.scatter(x3_new, y3reg,label = 'Prediction at 450sec')
plt.plot(x3_true,t3_true,'--g', label = 'True Function')
plt.legend(bbox_to_anchor=(1.5, 1),fontsize=12,ncol=1)
plt.xlabel('Data Samples, $x$', fontsize=20)
plt.ylabel('Desired Labels, $t$', fontsize=20);
```



4. Compare both models using plots (qualitative measure) and select a measure to assess the goodness-of-fit (quantitative measure, e.g. MSE)

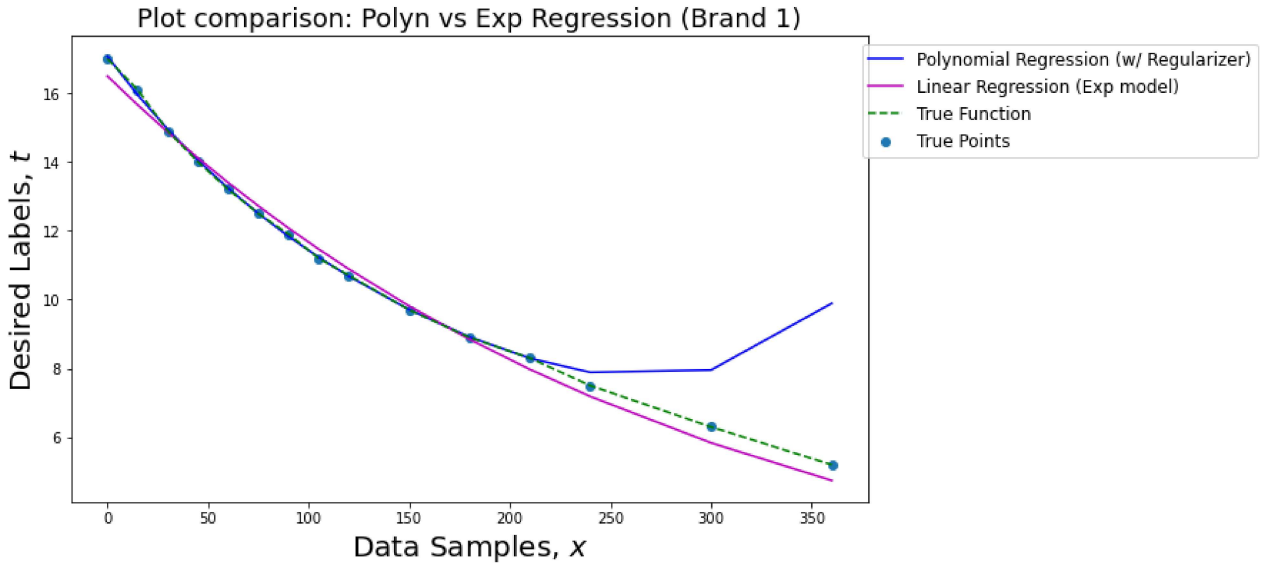
Plot (qualitative) comparison

Polynomial Model 1 vs Exponential Model 1

In [31]:
```
y1_pred = PolynomialRegression_test(x1_true, M, w1reg)
y1_pred_exp = LinRegressionExp_test(x1_true,w1)
```

In [32]:
```
fig=plt.figure(figsize=(10,6))
plt.scatter(x1_true,t1_true, label='True Points')
plt.plot(x1_true, y1_pred, 'b',label = 'Polynomial Regression (w/ Regularizer)')
plt.plot(x1_true, y1_pred_exp, 'm',label = 'Linear Regression (Exp model)')
plt.plot(x1_true, t1_true,'--g', label = 'True Function')
plt.legend(bbox_to_anchor=(1.5, 1),fontsize=12,ncol=1)
plt.xlabel('Data Samples, $x$', fontsize=20)
plt.ylabel('Desired Labels, $t$', fontsize=20);
plt.title('Plot comparison: Polyn vs Exp Regression (Brand 1)',fontsize=18)
```

Out[32]:
```
Text(0.5, 1.0, 'Plot comparison: Polyn vs Exp Regression (Brand 1)')
```



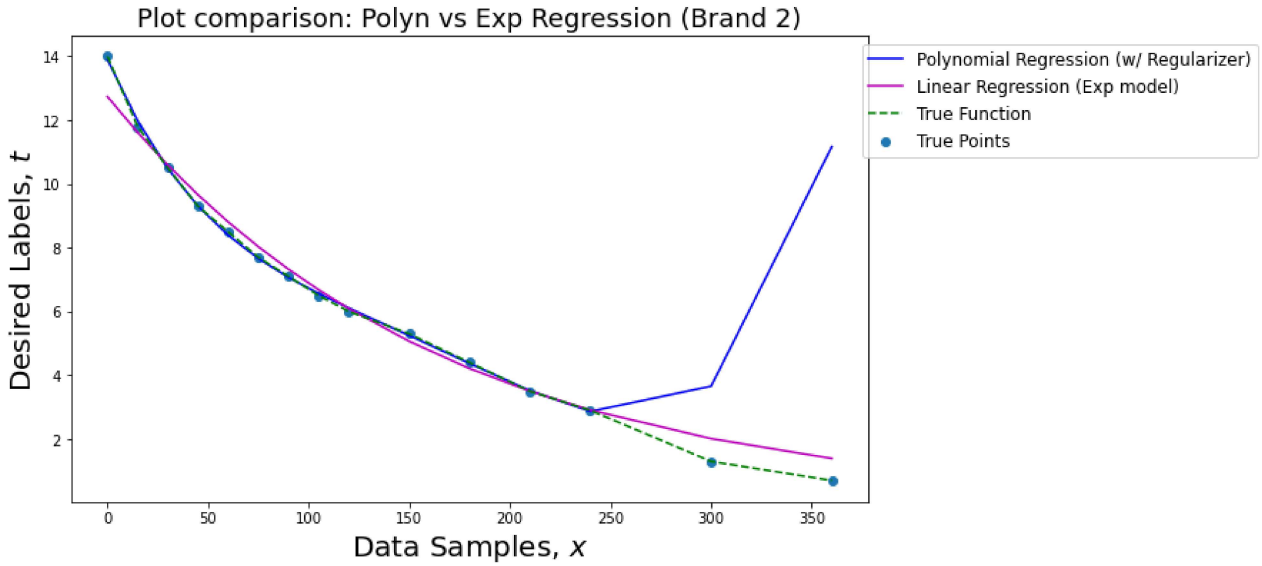
Comparison of models: For the given training set, the polynomial regression model fits the training data better than the exponential model -- nearly exactly. This is perhaps due to overfitting. Consequently, the polynomial model may fail to generalize to new data when compared with the exponential model. The 4th order polynomial model could benefit from use of a regularizer to reduce its overfitting/high variance

Polynomial Model 2 vs Exponential Model 2

In [33]:
```
y2_pred = PolynomialRegression_test(x2_true, M, w2reg)
y2_pred_exp = LinRegressionExp_test(x2_true,w2)
```

In [34]:
```
fig=plt.figure(figsize=(10,6))
plt.scatter(x2_true,t2_true, label='True Points')
plt.plot(x2_true, y2_pred, 'b',label = 'Polynomial Regression (w/ Regularizer)')
plt.plot(x2_true, y2_pred_exp, 'm',label = 'Linear Regression (Exp model)')
plt.plot(x2_true, t2_true,'--g', label = 'True Function')
plt.legend(bbox_to_anchor=(1.5, 1),fontsize=12,ncol=1)
plt.xlabel('Data Samples, $x$', fontsize=20)
plt.ylabel('Desired Labels, $t$', fontsize=20);
plt.title('Plot comparison: Polyn vs Exp Regression (Brand 2)',fontsize=18)
```

Out[34]:
```
Text(0.5, 1.0, 'Plot comparison: Polyn vs Exp Regression (Brand 2)')
```



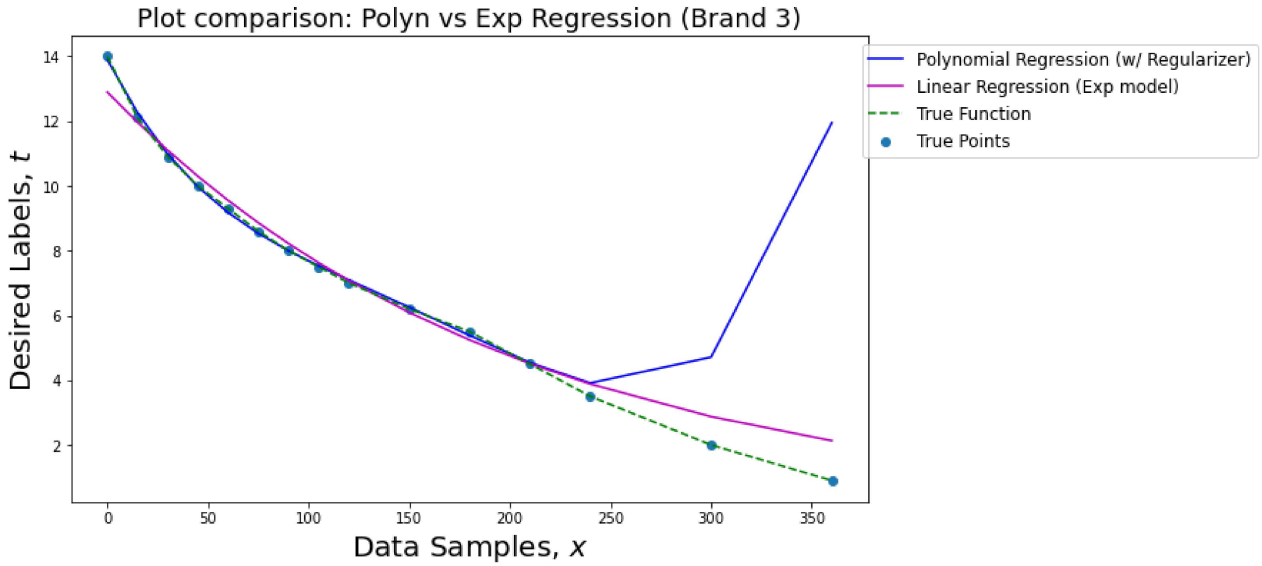
Comparison of models: For the given training set, the polynomial regression model (with regularizer) fits the training data better than the exponential model -- nearly exactly. This is perhaps due to overfitting. Consequently, the polynomial model may fail to generalize to new data when compared with the exponential model.

Polynomial Model 3 vs Exponential Model 3

In [35]:
```
y3_pred = PolynomialRegression_test(x3_true, M, w3reg)
y3_pred_exp = LinRegressionExp_test(x3_true,w3)
```

In [36]:
```
fig=plt.figure(figsize=(10,6))
plt.scatter(x3_true,t3_true, label='True Points')
plt.plot(x3_true, y3_pred, 'b',label = 'Polynomial Regression (w/ Regularizer)')
plt.plot(x3_true, y3_pred_exp, 'm',label = 'Linear Regression (Exp model)')
plt.plot(x3_true, t3_true,'--g', label = 'True Function')
plt.legend(bbox_to_anchor=(1.5, 1),fontsize=12,ncol=1)
plt.xlabel('Data Samples, $x$', fontsize=20)
plt.ylabel('Desired Labels, $t$', fontsize=20);
plt.title('Plot comparison: Polyn vs Exp Regression (Brand 3)',fontsize=18)
```

Out[36]:
```
Text(0.5, 1.0, 'Plot comparison: Polyn vs Exp Regression (Brand 3)')
```



Comparison of models: For the given training set, the polynomial regression model (with regularizer) fits the training data better than the exponential model -- nearly exactly. This is perhaps due to overfitting. Consequently, the polynomial model may fail to generalize to new data when compared with the exponential model.

Goodness of Fit (MSE Test): Use average of MSE scores for polynomial vs. exponential models

Using MSE scores obtained from testing polynomial regression models (1-3):

In [37]:
```
print('MSE for Polynomial Model 1:', mse1_poly)
print('MSE for Polynomial Model 2:', mse2_poly)
print('MSE for Polynomial Model 3:', mse3_poly)
```

```
MSE for Polynomial Model 1: 12.465959548470064
MSE for Polynomial Model 2: 57.622153258662784
MSE for Polynomial Model 3: 64.84278041407696
```

In [38]:
```
print('Average of MSE scores (Polynomial Models):',(1/3)*(mse1_poly + mse2_poly + mse3_
```

```
Average of MSE scores (Polynomial Models): 44.97696440706993
```

Using MSE scores obtained from testing exponential regression models (1-3):

In [39]:
```
print('MSE for Exponential Model 1:', mse1_exp)
print('MSE for Exponential Model 2:', mse2_exp)
print('MSE for Exponential Model 3:', mse3_exp)
```

```
MSE for Exponential Model 1: 0.6696559055050737
MSE for Exponential Model 2: 1.5689247747949917
MSE for Exponential Model 3: 2.034737996494934
```

In [40]:
```
print('Average of MSE scores (Exponential Models):',(1/3)*(mse1_exp + mse2_exp + mse3_e
```

```
Average of MSE scores (Exponential Models): 1.4244395589316663
```

The best average MSE score goes to the class of exponential regression models. Regarding the

prediction accuracy, prior knowledge of the exponential fit of $H(t) = H_0 e^{-\lambda t}$ may imply that the exponential model will provide a better prediction for future value. The exponential model prediction value is more consistent with this type of decay than the prediction values obtained with the polynomial model.

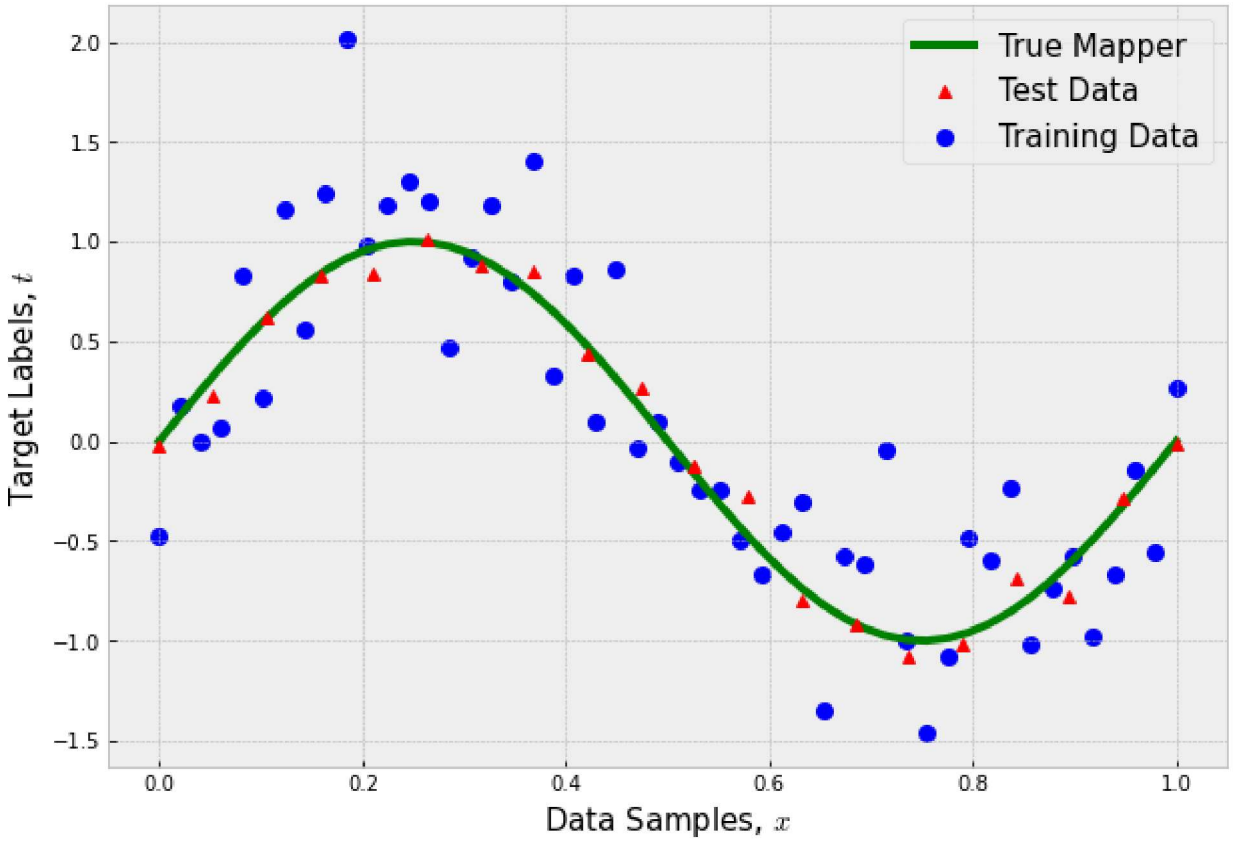
Question 2

In [4]:

```
%matplotlib inline
plt.style.use('bmh')

def NoisySinusoidalData(N, a, b, sigma):
    '''Generates N data points in the range [a,b) sampled from a sin(2*pi*x)
    with additive zero-mean Gaussian random noise with standard deviation sigma'''

    # N input samples, evenly spaced numbers between [a,b) incrementing by 1/N
    x = np.linspace(a,b,N)

    # draw N sampled from a univariate Gaussian distribution with mean 0, sigma standar
    noise = np.random.normal(0,sigma,N)

    # desired values, noisy sinusoidal
    t = np.sin(2*np.pi*x) + noise

    return x, t
```

In [5]:

```
# Generate input samples and desired values
N_train = 50 # number of data samples for training
N_test = 20 # number of data samples for test

a, b = [0,1] # data samples interval

sigma_train = 0.4 # standard deviation of the zero-mean Gaussian noise -- training data
sigma_test = 0.1 # standard deviation of the zero-mean Gaussian noise -- test data

x_train, t_train = NoisySinusoidalData(N_train, a, b, sigma_train) # Training Data - No
x_true, t_true = NoisySinusoidalData(N_train, a, b, 0) # True Sinusoidal - in practice,
x_test, t_test = NoisySinusoidalData(N_test, a, b, sigma_test) # Test Data - Noisy sinu

# Plotting
plt.figure(figsize=(10,7))
plt.scatter(x_train, t_train, c='b', linewidths=3, label = 'Training Data')
plt.plot(x_true, t_true, 'g', linewidth=4, label = 'True Mapper')
plt.plot(x_test, t_test, 'r^', label = 'Test Data')
plt.legend(fontsize=15)
plt.xlabel('Data Samples, $x$',size=15)
plt.ylabel('Target Labels, $t$',size=15);
```



Build a linear regression model with Gaussian basis functions

In [6]:

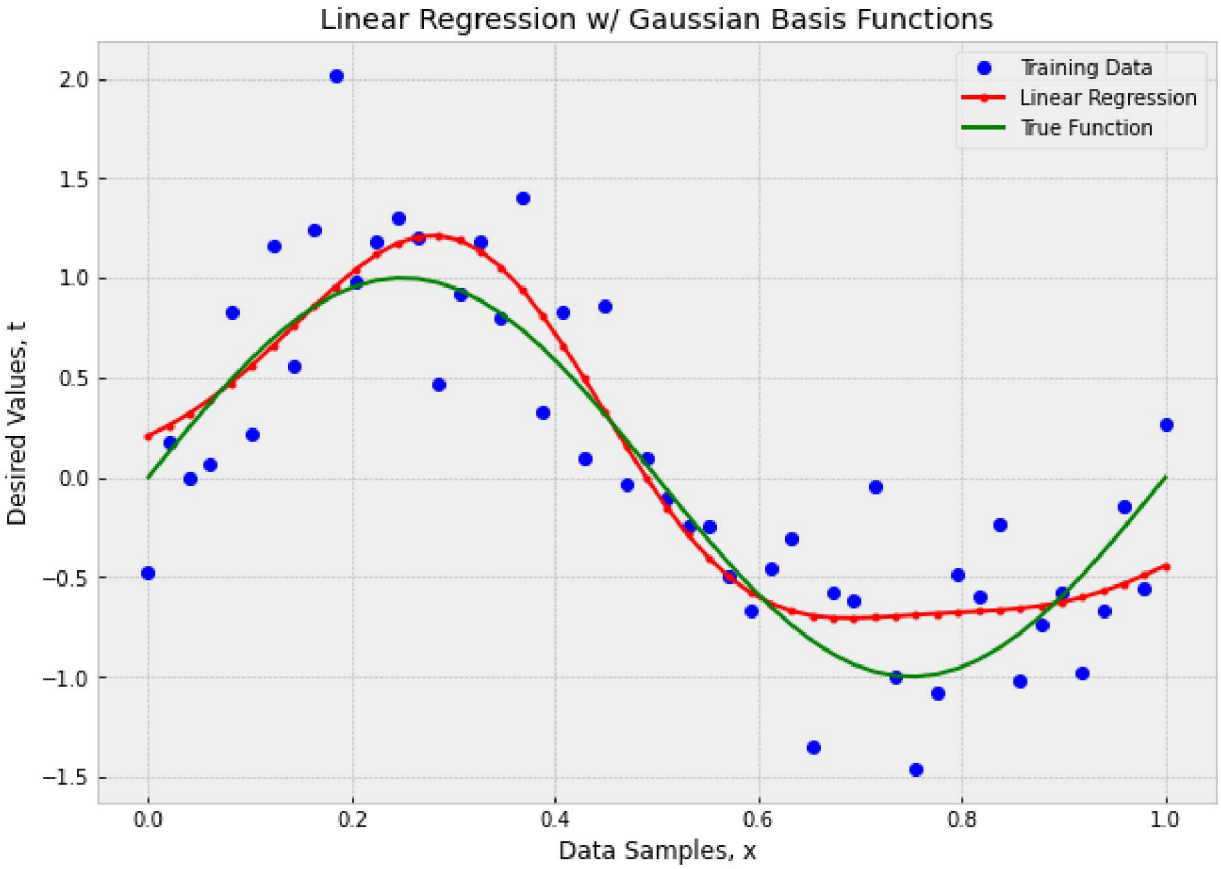
```
def LinearRegression_Gaussian(x,t,M,sigma,mu):
    '''Fit a Gaussian model with range of mu to the data input data x and desire values

    # Feature Matrix X
    X=np.array([np.exp((-1/((2*sigma)**2))*(x-mu[i])**2) for i in range(M)]).T

    # Coefficients w
    w = np.linalg.inv(X.T@X)@X.T@t  #inv(X.T@X)@X.T is the pseudoinverse of X

    # Model prediction, y = Xw
    y=X@w

    return w,y
```

Train Gaussian model on training set

In [7]:

```
M = 4 #select model order
mu = np.array([0.1, 0.3, 0.6, 0.9])
sigma = 0.1

# Find the parameters that fit the noisy sinusoidal
w, y_train = LinearRegression_Gaussian(x_train,t_train,M,sigma,mu)

plt.figure(figsize=(10,7))
plt.plot(x_train,t_train,'bo', label='Training Data')
```

```
plt.plot(x_train,y_train,'.-r', label = 'Linear Regression')
plt.plot(x_true,t_true,'g', label = 'True Function')
plt.legend()
plt.xlabel('Data Samples, x')
plt.ylabel('Desired Values, t');
plt.title('Linear Regression w/ Gaussian Basis Functions');
```



Make predictions using test set

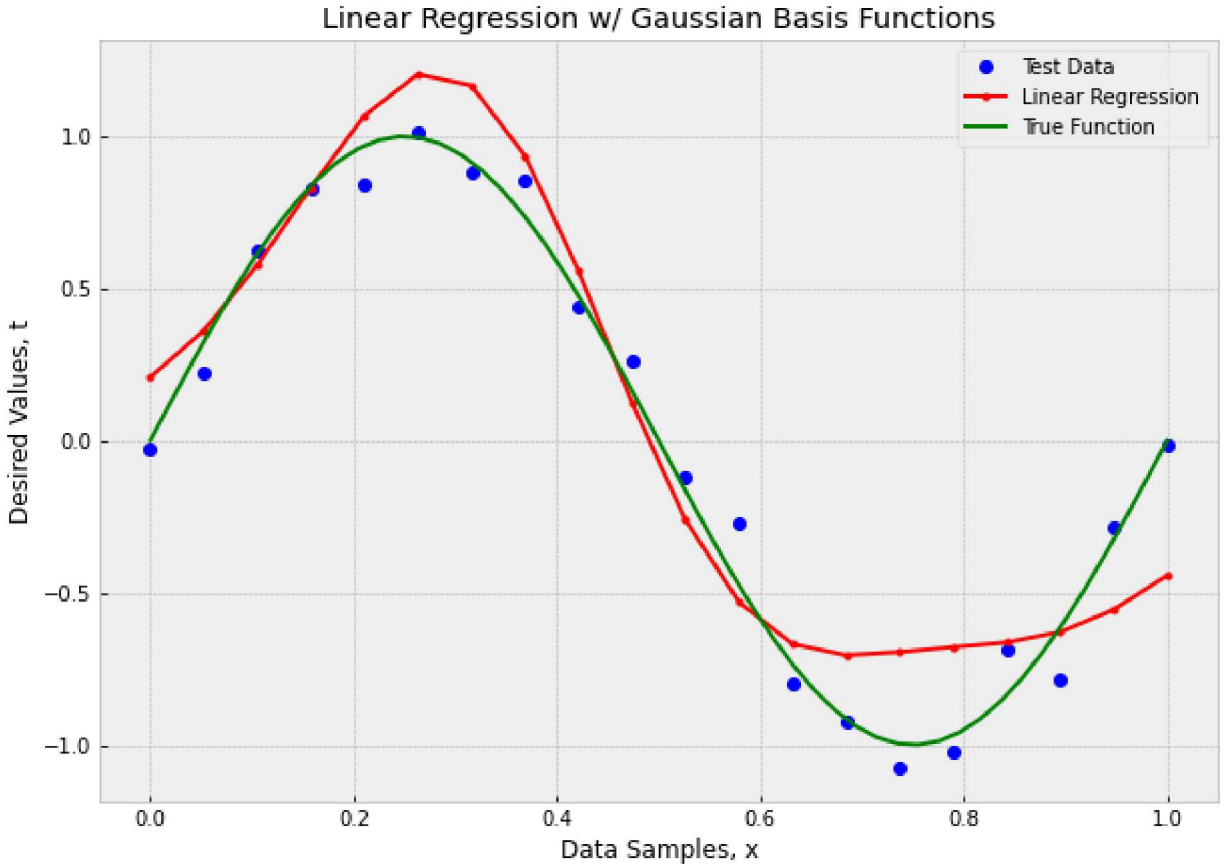
In [8]:

```
def LinearRegression_Gaussian_test(x, sigma, mu, w):
    '''Fit a Gaussian model with range mu and weights w to the data input data x and de

    # Feature Matrix X
    X=np.array([np.exp((-1/((2*sigma)**2))*(x-mu[i])**2) for i in range(M)]).T

    # Model prediction, y = Xw
    y=X@w

    return y
```

In [9]:

```
sigma = 0.1
mu = np.array([0.1, 0.3, 0.6, 0.9])
y_test = LinearRegression_Gaussian_test(x_test, sigma, mu, w)

plt.figure(figsize=(10,7))
plt.plot(x_test, t_test,'bo', label='Test Data')
plt.plot(x_test, y_test,'.-r', label = 'Linear Regression')
plt.plot(x_true, t_true,'g', label = 'True Function')
```

```
plt.legend()
plt.xlabel('Data Samples, x')
plt.ylabel('Desired Values, t');
plt.title('Linear Regression w/ Gaussian Basis Functions');
```



Finding μs, σ, and number of gaussians

After determining parameters for the basis function $e^{\frac{-(x_i-\mu_j)^2}{2\sigma^2}}$, namely μ and σ, optimal values for the parameters can be searched using k-fold cross-validation. Since the values t are observed on the interval y, [-1,1] and the domain of x is x, [0,1], we might begin looking at σs from (0,1] and μs from [0,1]. We could then locate the best average validation score (avg_perf_val) and fine-tune those parameters until a satisfactory score is obtained. We would use that combination of μ, σ, and number of gaussians.

In [10]:
```
from sklearn.model_selection import KFold

k = 4 # number of folds

kf = KFold(n_splits=k,shuffle=True)
```

In [11]:
```
# Values for number of gaussians, sigmas, and arrays of mu for inital parameter search.
gaussians = np.array([1,2,3,4]) # candidate #gaussians array
sigmas = np.array([.3, .4, .5, .7]) # candidate sigma array
mus = np.array([[0.0,0.1,0.2,0.3],[0.1,0.2,0.3,0.4],[0.2,0.3,0.4,0.5],[0.3,0.4,0.5,0.6]
```

In [23]:
```
len(avg_perf_val)
```

Out[23]:
```
256
```

In [12]:
```
avg_perf_val = []

for M in gaussians:
    for sig in sigmas:
        for mu in mus:
            print('M Value = ',M)
            print('Sigma Value = ',sig)
            print('Mus=', mu)
            # For each training/validation split
            f=1

            #initialize performance measures
            MSE_train_avg,MSE_val_avg = 0, 0

            for train_index, validation_index in kf.split(x_train):
                print('\nFold ',f)

                # Select training set using the indices found from kf.split
                x_train2, x_validation = x_train[train_index], x_train[validation_index

                # Select validation set using the indices found from kf.split
                t_train2, t_validation = t_train[train_index], t_train[validation_index

                # Training model with training set
                w, y_train = LinearRegression_Gaussian(x_train, t_train, M, sig, mu)

                # Evaluate trained model in validation set
                y_val = LinearRegression_Gaussian_test(x_validation, sig, mu, w)

                # Performance Measure
                MSE_train = np.mean((t_train-y_train)**2)
                MSE_val   = np.mean((t_validation-y_val)**2)

                # Average performance measure
                MSE_train_avg = MSE_train_avg+MSE_train
                MSE_val_avg = MSE_val_avg+MSE_val
                print('MSE Training = ', MSE_train)
                print('MSE Validation = ', MSE_val)
                f+=1
                avg_perf_val.append(MSE_val_avg/k)
            print('\nAverage Performance in Training = ', MSE_train_avg/k)
            print('Average Performance in Validation = ', MSE_val_avg/k)
            print('_____________________________________________________________')
```

```
M Value =  1
Sigma Value =  0.3
Mus= [0.  0.1 0.2 0.3]

Fold  1
MSE Training =  0.5246686750174497
MSE Validation =  0.6140683237649195

Fold  2
```

```
MSE Training =  0.5246686750174497
MSE Validation =  0.5172886079694347

Fold  3
MSE Training =  0.5246686750174497
MSE Validation =  0.5314432792030902

Fold  4
MSE Training =  0.5246686750174497
MSE Validation =  0.4290395239907332

Average Performance in Training =  0.5246686750174497
Average Performance in Validation =  0.5229599337320443
```

```
M Value =  1
Sigma Value =  0.3
Mus= [0.1 0.2 0.3 0.4]

Fold  1
MSE Training =  0.5456458154149655
MSE Validation =  0.49933111715934636

Fold  2
MSE Training =  0.5456458154149655
MSE Validation =  0.8401742084422585

Fold  3
MSE Training =  0.5456458154149655
MSE Validation =  0.31250316321799926

Fold  4
MSE Training =  0.5456458154149655
MSE Validation =  0.5098902982759521

Average Performance in Training =  0.5456458154149655
Average Performance in Validation =  0.5404746967738892
```

```
M Value =  1
Sigma Value =  0.3
Mus= [0.2 0.3 0.4 0.5]

Fold  1
MSE Training =  0.5735295539433191
MSE Validation =  0.5613553841471681

Fold  2
MSE Training =  0.5735295539433191
MSE Validation =  0.7934601505239401

Fold  3
MSE Training =  0.5735295539433191
MSE Validation =  0.5407571079819157

Fold  4
MSE Training =  0.5735295539433191
MSE Validation =  0.38123253755487974

Average Performance in Training =  0.5735295539433191
Average Performance in Validation =  0.5692012950519758
```

```
M Value =  1
Sigma Value =  0.3
Mus= [0.3 0.4 0.5 0.6]

Fold  1
MSE Training =  0.6042645225844371
MSE Validation =  0.5403469938911412

Fold  2
MSE Training =  0.6042645225844371
MSE Validation =  0.6644898901052163

Fold  3
MSE Training =  0.6042645225844371
MSE Validation =  0.5315750160020909

Fold  4
MSE Training =  0.6042645225844371
MSE Validation =  0.6809538704370097

Average Performance in Training =  0.6042645225844371
Average Performance in Validation =  0.6043414426088645
```

```
M Value =  1
Sigma Value =  0.4
Mus= [0.  0.1 0.2 0.3]

Fold  1
MSE Training =  0.5810486276470123
MSE Validation =  0.7086572489219605

Fold  2
MSE Training =  0.5810486276470123
MSE Validation =  0.5637715774547614

Fold  3
MSE Training =  0.5810486276470123
MSE Validation =  0.3118587170938104

Fold  4
MSE Training =  0.5810486276470123
MSE Validation =  0.7307126695272922

Average Performance in Training =  0.5810486276470123
Average Performance in Validation =  0.5787500532494562
```

```
M Value =  1
Sigma Value =  0.4
Mus= [0.1 0.2 0.3 0.4]

Fold  1
MSE Training =  0.5975088943515914
MSE Validation =  0.5298608410223615

Fold  2
MSE Training =  0.5975088943515914
MSE Validation =  0.62656613198813

Fold  3
MSE Training =  0.5975088943515914
```

```
MSE Validation =  0.2704940354662759

Fold  4
MSE Training =  0.5975088943515914
MSE Validation =  0.9663304702373221

Average Performance in Training =  0.5975088943515914
Average Performance in Validation =  0.5983128696785224
```

```
M Value =  1
Sigma Value =  0.4
Mus= [0.2 0.3 0.4 0.5]

Fold  1
MSE Training =  0.6140121054205979
MSE Validation =  0.41646981061526744

Fold  2
MSE Training =  0.6140121054205979
MSE Validation =  0.44134223370588543

Fold  3
MSE Training =  0.6140121054205979
MSE Validation =  0.638017650635981

Fold  4
MSE Training =  0.6140121054205979
MSE Validation =  0.9910697406019285

Average Performance in Training =  0.6140121054205979
Average Performance in Validation =  0.6217248588897656
```

```
M Value =  1
Sigma Value =  0.4
Mus= [0.3 0.4 0.5 0.6]

Fold  1
MSE Training =  0.6292100709419532
MSE Validation =  0.696531982686252

Fold  2
MSE Training =  0.6292100709419532
MSE Validation =  0.5411278154497511

Fold  3
MSE Training =  0.6292100709419532
MSE Validation =  0.6543814410483856

Fold  4
MSE Training =  0.6292100709419532
MSE Validation =  0.6265290732290828

Average Performance in Training =  0.6292100709419532
Average Performance in Validation =  0.6296425781033679
```

```
M Value =  1
Sigma Value =  0.5
Mus= [0.  0.1 0.2 0.3]

Fold  1
```

```
MSE Training =  0.6114798993681904
MSE Validation =  0.5240685025658579

Fold  2
MSE Training =  0.6114798993681904
MSE Validation =  1.0147244551736863

Fold  3
MSE Training =  0.6114798993681904
MSE Validation =  0.3678701069838093

Fold  4
MSE Training =  0.6114798993681904
MSE Validation =  0.5129371028324775

Average Performance in Training =  0.6114798993681904
Average Performance in Validation =  0.6049000418889577
```

```
M Value =  1
Sigma Value =  0.5
Mus= [0.1 0.2 0.3 0.4]

Fold  1
MSE Training =  0.6215026919182994
MSE Validation =  0.49623633097387015

Fold  2
MSE Training =  0.6215026919182994
MSE Validation =  0.755989587361401

Fold  3
MSE Training =  0.6215026919182994
MSE Validation =  0.7517852146433053

Fold  4
MSE Training =  0.6215026919182994
MSE Validation =  0.48123125681973217

Average Performance in Training =  0.6215026919182994
Average Performance in Validation =  0.6213105974495772
```

```
M Value =  1
Sigma Value =  0.5
Mus= [0.2 0.3 0.4 0.5]

Fold  1
MSE Training =  0.630801771841755
MSE Validation =  0.7880882000850927

Fold  2
MSE Training =  0.630801771841755
MSE Validation =  0.6821987702780297

Fold  3
MSE Training =  0.630801771841755
MSE Validation =  0.46543005618320205

Fold  4
MSE Training =  0.630801771841755
MSE Validation =  0.570099775264061
```

```
Average Performance in Training =  0.630801771841755
Average Performance in Validation =  0.6264542004525964
```

```
M Value =  1
Sigma Value =  0.5
Mus= [0.3 0.4 0.5 0.6]

Fold  1
MSE Training =  0.6389955167622411
MSE Validation =  0.6269587973604162

Fold  2
MSE Training =  0.6389955167622411
MSE Validation =  0.7691175223860468

Fold  3
MSE Training =  0.6389955167622411
MSE Validation =  0.33605553135579697

Fold  4
MSE Training =  0.6389955167622411
MSE Validation =  0.8140097754282064

Average Performance in Training =  0.6389955167622411
Average Performance in Validation =  0.6365354066326165
```

```
M Value =  1
Sigma Value =  0.7
Mus= [0.  0.1 0.2 0.3]

Fold  1
MSE Training =  0.6351507725515537
MSE Validation =  1.0155860424485599

Fold  2
MSE Training =  0.6351507725515537
MSE Validation =  0.40349099858347603

Fold  3
MSE Training =  0.6351507725515537
MSE Validation =  0.7128798409534637

Fold  4
MSE Training =  0.6351507725515537
MSE Validation =  0.39624825022663784

Average Performance in Training =  0.6351507725515537
Average Performance in Validation =  0.6320512830530344
```

```
M Value =  1
Sigma Value =  0.7
Mus= [0.1 0.2 0.3 0.4]

Fold  1
MSE Training =  0.6391547012000043
MSE Validation =  0.8753274852864071

Fold  2
MSE Training =  0.6391547012000043
```

```
MSE Validation =  0.5603884483721339

Fold  3
MSE Training =  0.6391547012000043
MSE Validation =  0.773763507640652

Fold  4
MSE Training =  0.6391547012000043
MSE Validation =  0.3340221525626132

Average Performance in Training =  0.6391547012000043
Average Performance in Validation =  0.6358753984654516
```

```
M Value =  1
Sigma Value =  0.7
Mus= [0.2 0.3 0.4 0.5]

Fold  1
MSE Training =  0.6427866446299724
MSE Validation =  0.8534197310415124

Fold  2
MSE Training =  0.6427866446299724
MSE Validation =  0.7479985061497586

Fold  3
MSE Training =  0.6427866446299724
MSE Validation =  0.32939462710775175

Fold  4
MSE Training =  0.6427866446299724
MSE Validation =  0.6140133018932562

Average Performance in Training =  0.6427866446299724
Average Performance in Validation =  0.6362065415480697
```

```
M Value =  1
Sigma Value =  0.7
Mus= [0.3 0.4 0.5 0.6]

Fold  1
MSE Training =  0.6460037773386795
MSE Validation =  0.419066858612532

Fold  2
MSE Training =  0.6460037773386795
MSE Validation =  0.5938933566430391

Fold  3
MSE Training =  0.6460037773386795
MSE Validation =  0.9698763012351281

Fold  4
MSE Training =  0.6460037773386795
MSE Validation =  0.6244325378158353

Average Performance in Training =  0.6460037773386795
Average Performance in Validation =  0.6518172635766337
```

```
M Value =  2
```

```
Sigma Value =  0.3
Mus= [0.  0.1 0.2 0.3]

Fold  1
MSE Training =  0.4521899969320076
MSE Validation =  0.4741089420160902

Fold  2
MSE Training =  0.4521899969320076
MSE Validation =  0.5275496091437407

Fold  3
MSE Training =  0.4521899969320076
MSE Validation =  0.3809120982132454

Fold  4
MSE Training =  0.4521899969320076
MSE Validation =  0.418082791913636

Average Performance in Training =  0.4521899969320076
Average Performance in Validation =  0.45016336032167814
```

```
M Value =  2
Sigma Value =  0.3
Mus= [0.1 0.2 0.3 0.4]

Fold  1
MSE Training =  0.40427453958821574
MSE Validation =  0.5040515727749465

Fold  2
MSE Training =  0.40427453958821574
MSE Validation =  0.45181316495214274

Fold  3
MSE Training =  0.40427453958821574
MSE Validation =  0.3673888223093202

Fold  4
MSE Training =  0.40427453958821574
MSE Validation =  0.2815682934372319

Average Performance in Training =  0.40427453958821574
Average Performance in Validation =  0.40120546336841034
```

```
M Value =  2
Sigma Value =  0.3
Mus= [0.2 0.3 0.4 0.5]

Fold  1
MSE Training =  0.3513033849036525
MSE Validation =  0.5018230854258858

Fold  2
MSE Training =  0.3513033849036525
MSE Validation =  0.40672829634010327

Fold  3
MSE Training =  0.3513033849036525
MSE Validation =  0.1325112447972385
```

```
Fold  4
MSE Training =  0.3513033849036525
MSE Validation =  0.3469888620548253

Average Performance in Training =  0.3513033849036525
Average Performance in Validation =  0.34701287215451326
```

```
M Value =  2
Sigma Value =  0.3
Mus= [0.3 0.4 0.5 0.6]

Fold  1
MSE Training =  0.30403235334609774
MSE Validation =  0.18973950951320057

Fold  2
MSE Training =  0.30403235334609774
MSE Validation =  0.3897692680165905

Fold  3
MSE Training =  0.30403235334609774
MSE Validation =  0.523178317798544

Fold  4
MSE Training =  0.30403235334609774
MSE Validation =  0.11582197881958954

Average Performance in Training =  0.30403235334609774
Average Performance in Validation =  0.30462726853698113
```

```
M Value =  2
Sigma Value =  0.4
Mus= [0.  0.1 0.2 0.3]

Fold  1
MSE Training =  0.3945439847045024
MSE Validation =  0.43283205219663645

Fold  2
MSE Training =  0.3945439847045024
MSE Validation =  0.3189543383053301

Fold  3
MSE Training =  0.3945439847045024
MSE Validation =  0.1513053592295702

Fold  4
MSE Training =  0.3945439847045024
MSE Validation =  0.6781926539953931

Average Performance in Training =  0.3945439847045024
Average Performance in Validation =  0.39532110093173245
```

```
M Value =  2
Sigma Value =  0.4
Mus= [0.1 0.2 0.3 0.4]

Fold  1
MSE Training =  0.3667992499759933
```

```
MSE Validation =  0.22999985856262384

Fold  2
MSE Training =  0.3667992499759933
MSE Validation =  0.14511497566793968

Fold  3
MSE Training =  0.3667992499759933
MSE Validation =  0.7505631903901652

Fold  4
MSE Training =  0.3667992499759933
MSE Validation =  0.37139261409336305

Average Performance in Training =  0.3667992499759933
Average Performance in Validation =  0.37426765967852293
```

```
M Value =  2
Sigma Value =  0.4
Mus= [0.2 0.3 0.4 0.5]

Fold  1
MSE Training =  0.34225355699470933
MSE Validation =  0.49642825924172773

Fold  2
MSE Training =  0.34225355699470933
MSE Validation =  0.3684864655269131

Fold  3
MSE Training =  0.34225355699470933
MSE Validation =  0.2042390109867748

Fold  4
MSE Training =  0.34225355699470933
MSE Validation =  0.2848265246584868

Average Performance in Training =  0.34225355699470933
Average Performance in Validation =  0.3384950651034756
```

```
M Value =  2
Sigma Value =  0.4
Mus= [0.3 0.4 0.5 0.6]

Fold  1
MSE Training =  0.322822970534961
MSE Validation =  0.19114401195726766

Fold  2
MSE Training =  0.322822970534961
MSE Validation =  0.23761150528840558

Fold  3
MSE Training =  0.322822970534961
MSE Validation =  0.5945054307997645

Fold  4
MSE Training =  0.322822970534961
MSE Validation =  0.2861051360797605
```

```
Average Performance in Training =  0.322822970534961
Average Performance in Validation =  0.32734152103129954
```

```
M Value =  2
Sigma Value =  0.5
Mus= [0.  0.1 0.2 0.3]

Fold  1
MSE Training =  0.3708438643768567
MSE Validation =  0.32548343723126616

Fold  2
MSE Training =  0.3708438643768567
MSE Validation =  0.2752454164644492

Fold  3
MSE Training =  0.3708438643768567
MSE Validation =  0.675067291363571

Fold  4
MSE Training =  0.3708438643768567
MSE Validation =  0.21932588536964048

Average Performance in Training =  0.3708438643768567
Average Performance in Validation =  0.37378050760723175
```

```
M Value =  2
Sigma Value =  0.5
Mus= [0.1 0.2 0.3 0.4]

Fold  1
MSE Training =  0.3565361023430354
MSE Validation =  0.39152619510036507

Fold  2
MSE Training =  0.3565361023430354
MSE Validation =  0.2895705663242325

Fold  3
MSE Training =  0.3565361023430354
MSE Validation =  0.4049819520195153

Fold  4
MSE Training =  0.3565361023430354
MSE Validation =  0.34273031619981853

Average Performance in Training =  0.3565361023430354
Average Performance in Validation =  0.3572022574109828
```

```
M Value =  2
Sigma Value =  0.5
Mus= [0.2 0.3 0.4 0.5]

Fold  1
MSE Training =  0.3444114159780673
MSE Validation =  0.3040291486912229

Fold  2
MSE Training =  0.3444114159780673
MSE Validation =  0.5638825794917002
```

```
Fold  3
MSE Training =  0.3444114159780673
MSE Validation =  0.2092061293535493

Fold  4
MSE Training =  0.3444114159780673
MSE Validation =  0.28560373169023096

Average Performance in Training =  0.3444114159780673
Average Performance in Validation =  0.3406803973066758
```

```
M Value =  2
Sigma Value =  0.5
Mus= [0.3 0.4 0.5 0.6]

Fold  1
MSE Training =  0.33486632069868366
MSE Validation =  0.1394021496105036

Fold  2
MSE Training =  0.33486632069868366
MSE Validation =  0.22033516481776855

Fold  3
MSE Training =  0.33486632069868366
MSE Validation =  0.6189036672418501

Fold  4
MSE Training =  0.33486632069868366
MSE Validation =  0.38665724503870363

Average Performance in Training =  0.33486632069868366
Average Performance in Validation =  0.34132455667720646
```

```
M Value =  2
Sigma Value =  0.7
Mus= [0.  0.1 0.2 0.3]

Fold  1
MSE Training =  0.35972607804572126
MSE Validation =  0.4132445821271362

Fold  2
MSE Training =  0.35972607804572126
MSE Validation =  0.4729165114108544

Fold  3
MSE Training =  0.35972607804572126
MSE Validation =  0.20658229832209093

Fold  4
MSE Training =  0.35972607804572126
MSE Validation =  0.3322685088689245

Average Performance in Training =  0.35972607804572126
Average Performance in Validation =  0.3562529751822515
```

```
M Value =  2
Sigma Value =  0.7
```

```
Mus= [0.1 0.2 0.3 0.4]

Fold  1
MSE Training =  0.35480390684483665
MSE Validation =  0.3239079700274093

Fold  2
MSE Training =  0.35480390684483665
MSE Validation =  0.3158542171077766

Fold  3
MSE Training =  0.35480390684483665
MSE Validation =  0.41306300049010725

Fold  4
MSE Training =  0.35480390684483665
MSE Validation =  0.3722109086335941

Average Performance in Training =  0.35480390684483665
Average Performance in Validation =  0.3562590240647218
```

```
M Value =  2
Sigma Value =  0.7
Mus= [0.2 0.3 0.4 0.5]

Fold  1
MSE Training =  0.35058705961152625
MSE Validation =  0.4326676562409596

Fold  2
MSE Training =  0.35058705961152625
MSE Validation =  0.21241318131038886

Fold  3
MSE Training =  0.35058705961152625
MSE Validation =  0.4035674403285345

Fold  4
MSE Training =  0.35058705961152625
MSE Validation =  0.358374400705531

Average Performance in Training =  0.35058705961152625
Average Performance in Validation =  0.35175566964635346
```

```
M Value =  2
Sigma Value =  0.7
Mus= [0.3 0.4 0.5 0.6]

Fold  1
MSE Training =  0.34711123036265307
MSE Validation =  0.29649248417191715

Fold  2
MSE Training =  0.34711123036265307
MSE Validation =  0.2813876949299225

Fold  3
MSE Training =  0.34711123036265307
MSE Validation =  0.42161253656454983
```

```
Fold  4
MSE Training =  0.34711123036265307
MSE Validation =  0.39864739591951165

Average Performance in Training =  0.34711123036265307
Average Performance in Validation =  0.3495350278964753
```

```
M Value =  3
Sigma Value =  0.3
Mus= [0.  0.1 0.2 0.3]

Fold  1
MSE Training =  0.16494341140488333
MSE Validation =  0.27562919517842777

Fold  2
MSE Training =  0.16494341140488333
MSE Validation =  0.13587895763037844

Fold  3
MSE Training =  0.16494341140488333
MSE Validation =  0.12297569770248958

Fold  4
MSE Training =  0.16494341140488333
MSE Validation =  0.11848801760831756

Average Performance in Training =  0.16494341140488333
Average Performance in Validation =  0.16324296702990332
```

```
M Value =  3
Sigma Value =  0.3
Mus= [0.1 0.2 0.3 0.4]

Fold  1
MSE Training =  0.19752278383751784
MSE Validation =  0.33262868496082276

Fold  2
MSE Training =  0.19752278383751784
MSE Validation =  0.10226661491503249

Fold  3
MSE Training =  0.19752278383751784
MSE Validation =  0.16491152369502524

Fold  4
MSE Training =  0.19752278383751784
MSE Validation =  0.18696350076245602

Average Performance in Training =  0.19752278383751784
Average Performance in Validation =  0.19669258108333412
```

```
M Value =  3
Sigma Value =  0.3
Mus= [0.2 0.3 0.4 0.5]

Fold  1
MSE Training =  0.23002864540894385
MSE Validation =  0.16041783407793436
```

```
Fold  2
MSE Training =  0.23002864540894385
MSE Validation =  0.29239258644670973

Fold  3
MSE Training =  0.23002864540894385
MSE Validation =  0.24112233198610575

Fold  4
MSE Training =  0.23002864540894385
MSE Validation =  0.22678573498279594

Average Performance in Training =  0.23002864540894385
Average Performance in Validation =  0.23017962187338645
```

```
M Value =  3
Sigma Value =  0.3
Mus= [0.3 0.4 0.5 0.6]

Fold  1
MSE Training =  0.25476157460586224
MSE Validation =  0.34653430927672046

Fold  2
MSE Training =  0.25476157460586224
MSE Validation =  0.26405812199464923

Fold  3
MSE Training =  0.25476157460586224
MSE Validation =  0.1836579113907727

Fold  4
MSE Training =  0.25476157460586224
MSE Validation =  0.21637351558966944

Average Performance in Training =  0.25476157460586224
Average Performance in Validation =  0.252655964562953
```

```
M Value =  3
Sigma Value =  0.4
Mus= [0.  0.1 0.2 0.3]

Fold  1
MSE Training =  0.2501089397244983
MSE Validation =  0.24077808731126768

Fold  2
MSE Training =  0.2501089397244983
MSE Validation =  0.24237392287028475

Fold  3
MSE Training =  0.2501089397244983
MSE Validation =  0.12686434148993378

Fold  4
MSE Training =  0.2501089397244983
MSE Validation =  0.39184156299879397

Average Performance in Training =  0.2501089397244983
```

```
Average Performance in Validation =  0.25046447866757005
```

```
M Value =  3
Sigma Value =  0.4
Mus= [0.1 0.2 0.3 0.4]

Fold  1
MSE Training =  0.2699570792422335
MSE Validation =  0.30974925660484137

Fold  2
MSE Training =  0.2699570792422335
MSE Validation =  0.4180399285038309

Fold  3
MSE Training =  0.2699570792422335
MSE Validation =  0.1480074410299713

Fold  4
MSE Training =  0.2699570792422335
MSE Validation =  0.18837543861160658

Average Performance in Training =  0.2699570792422335
Average Performance in Validation =  0.26604301618756254
```

```
M Value =  3
Sigma Value =  0.4
Mus= [0.2 0.3 0.4 0.5]

Fold  1
MSE Training =  0.28626128775023096
MSE Validation =  0.2728529773692937

Fold  2
MSE Training =  0.28626128775023096
MSE Validation =  0.3890186120834385

Fold  3
MSE Training =  0.28626128775023096
MSE Validation =  0.20131298779461448

Fold  4
MSE Training =  0.28626128775023096
MSE Validation =  0.2744148225908882

Average Performance in Training =  0.28626128775023096
Average Performance in Validation =  0.28439984995955875
```

```
M Value =  3
Sigma Value =  0.4
Mus= [0.3 0.4 0.5 0.6]

Fold  1
MSE Training =  0.29791886447493693
MSE Validation =  0.1823211630667078

Fold  2
MSE Training =  0.29791886447493693
MSE Validation =  0.31232849515139544
```

```
Fold  3
MSE Training =  0.29791886447493693
MSE Validation =  0.42772138687306444

Fold  4
MSE Training =  0.29791886447493693
MSE Validation =  0.27773675203622755

Average Performance in Training =  0.29791886447493693
Average Performance in Validation =  0.3000269492818488
```

```
M Value =  3
Sigma Value =  0.5
Mus= [0.  0.1 0.2 0.3]

Fold  1
MSE Training =  0.293727262160468
MSE Validation =  0.20790862594863962

Fold  2
MSE Training =  0.293727262160468
MSE Validation =  0.3399222446957232

Fold  3
MSE Training =  0.293727262160468
MSE Validation =  0.2584126900630575

Fold  4
MSE Training =  0.293727262160468
MSE Validation =  0.37196745907416623

Average Performance in Training =  0.293727262160468
Average Performance in Validation =  0.2945527549453966
```

```
M Value =  3
Sigma Value =  0.5
Mus= [0.1 0.2 0.3 0.4]

Fold  1
MSE Training =  0.3042355396380868
MSE Validation =  0.3014376768668675

Fold  2
MSE Training =  0.3042355396380868
MSE Validation =  0.25193606557941883

Fold  3
MSE Training =  0.3042355396380868
MSE Validation =  0.46771262359678184

Fold  4
MSE Training =  0.3042355396380868
MSE Validation =  0.20044723724510316

Average Performance in Training =  0.3042355396380868
Average Performance in Validation =  0.30538340082204285
```

```
M Value =  3
Sigma Value =  0.5
Mus= [0.2 0.3 0.4 0.5]
```

```
Fold  1
MSE Training =  0.3127168393786403
MSE Validation =  0.23066291844229897

Fold  2
MSE Training =  0.3127168393786403
MSE Validation =  0.3056893826010537

Fold  3
MSE Training =  0.3127168393786403
MSE Validation =  0.49933398140933005

Fold  4
MSE Training =  0.3127168393786403
MSE Validation =  0.2226045232047059

Average Performance in Training =  0.3127168393786403
Average Performance in Validation =  0.3145727014143471
```

```
M Value =  3
Sigma Value =  0.5
Mus= [0.3 0.4 0.5 0.6]

Fold  1
MSE Training =  0.3189589399386016
MSE Validation =  0.3963697123495213

Fold  2
MSE Training =  0.3189589399386016
MSE Validation =  0.41403249316310037

Fold  3
MSE Training =  0.3189589399386016
MSE Validation =  0.2400678586296503

Fold  4
MSE Training =  0.3189589399386016
MSE Validation =  0.21099200180918284

Average Performance in Training =  0.3189589399386016
Average Performance in Validation =  0.3153655164878637
```

```
M Value =  3
Sigma Value =  0.7
Mus= [0.  0.1 0.2 0.3]

Fold  1
MSE Training =  0.3280023787675633
MSE Validation =  0.13135260543900293

Fold  2
MSE Training =  0.3280023787675633
MSE Validation =  0.5045133565248519

Fold  3
MSE Training =  0.3280023787675633
MSE Validation =  0.3322990627708789

Fold  4
```

```
MSE Training =  0.3280023787675633
MSE Validation =  0.3455227232997921

Average Performance in Training =  0.3280023787675633
Average Performance in Validation =  0.3284219370086315
```

```
M Value =  3
Sigma Value =  0.7
Mus= [0.1 0.2 0.3 0.4]

Fold  1
MSE Training =  0.3317910431184704
MSE Validation =  0.42391892230126166

Fold  2
MSE Training =  0.3317910431184704
MSE Validation =  0.19954199515469864

Fold  3
MSE Training =  0.3317910431184704
MSE Validation =  0.389915788570302

Fold  4
MSE Training =  0.3317910431184704
MSE Validation =  0.3171308971793677

Average Performance in Training =  0.3317910431184704
Average Performance in Validation =  0.33262690080140755
```

```
M Value =  3
Sigma Value =  0.7
Mus= [0.2 0.3 0.4 0.5]

Fold  1
MSE Training =  0.3349554484528753
MSE Validation =  0.3612735167621391

Fold  2
MSE Training =  0.3349554484528753
MSE Validation =  0.14782593785641354

Fold  3
MSE Training =  0.3349554484528753
MSE Validation =  0.2256011803362217

Fold  4
MSE Training =  0.3349554484528753
MSE Validation =  0.6185221123806599

Average Performance in Training =  0.3349554484528753
Average Performance in Validation =  0.33830568683385853
```

```
M Value =  3
Sigma Value =  0.7
Mus= [0.3 0.4 0.5 0.6]

Fold  1
MSE Training =  0.33747834745423694
MSE Validation =  0.13315635813519042
```

```
Fold  2
MSE Training =  0.33747834745423694
MSE Validation =  0.4439225013277006

Fold  3
MSE Training =  0.33747834745423694
MSE Validation =  0.5258733371741741

Fold  4
MSE Training =  0.33747834745423694
MSE Validation =  0.2551176794670145

Average Performance in Training =  0.33747834745423694
Average Performance in Validation =  0.33951746902601987
```

```
M Value =  4
Sigma Value =  0.3
Mus= [0.  0.1 0.2 0.3]

Fold  1
MSE Training =  0.12321875457204834
MSE Validation =  0.125984639005893

Fold  2
MSE Training =  0.12321875457204834
MSE Validation =  0.09978904292467401

Fold  3
MSE Training =  0.12321875457204834
MSE Validation =  0.15237687559246468

Fold  4
MSE Training =  0.12321875457204834
MSE Validation =  0.11644644636628927

Average Performance in Training =  0.12321875457204834
Average Performance in Validation =  0.12364925097233025
```

```
M Value =  4
Sigma Value =  0.3
Mus= [0.1 0.2 0.3 0.4]

Fold  1
MSE Training =  0.12154840632815071
MSE Validation =  0.0888763885293329

Fold  2
MSE Training =  0.12154840632815071
MSE Validation =  0.15794287665541815

Fold  3
MSE Training =  0.12154840632815071
MSE Validation =  0.1341359853892385

Fold  4
MSE Training =  0.12154840632815071
MSE Validation =  0.10492817036124245

Average Performance in Training =  0.12154840632815071
Average Performance in Validation =  0.121470855233808
```

```
M Value =  4
Sigma Value =  0.3
Mus= [0.2 0.3 0.4 0.5]

Fold  1
MSE Training =  0.12321937647530992
MSE Validation =  0.16681815698744304

Fold  2
MSE Training =  0.12321937647530992
MSE Validation =  0.07460194239746609

Fold  3
MSE Training =  0.12321937647530992
MSE Validation =  0.1920503982898235

Fold  4
MSE Training =  0.12321937647530992
MSE Validation =  0.05982522935698283

Average Performance in Training =  0.12321937647530992
Average Performance in Validation =  0.12332393175792887
```

```
M Value =  4
Sigma Value =  0.3
Mus= [0.3 0.4 0.5 0.6]

Fold  1
MSE Training =  0.1295426692014117
MSE Validation =  0.13602838150395707

Fold  2
MSE Training =  0.1295426692014117
MSE Validation =  0.12125775107375505

Fold  3
MSE Training =  0.1295426692014117
MSE Validation =  0.06948135433103912

Fold  4
MSE Training =  0.1295426692014117
MSE Validation =  0.19155312371565478

Average Performance in Training =  0.1295426692014117
Average Performance in Validation =  0.1295801526561015
```

```
M Value =  4
Sigma Value =  0.4
Mus= [0.  0.1 0.2 0.3]

Fold  1
MSE Training =  0.12059127917305906
MSE Validation =  0.18401289695542064

Fold  2
MSE Training =  0.12059127917305906
MSE Validation =  0.06708660816225874

Fold  3
```

```
MSE Training =  0.12059127917305906
MSE Validation =  0.17873469288978594

Fold  4
MSE Training =  0.12059127917305906
MSE Validation =  0.051704506453807475

Average Performance in Training =  0.12059127917305906
Average Performance in Validation =  0.1203846761153182
```

```
M Value =  4
Sigma Value =  0.4
Mus= [0.1 0.2 0.3 0.4]

Fold  1
MSE Training =  0.12094871289872679
MSE Validation =  0.10348343125529777

Fold  2
MSE Training =  0.12094871289872679
MSE Validation =  0.1384753050159149

Fold  3
MSE Training =  0.12094871289872679
MSE Validation =  0.15836173584696775

Fold  4
MSE Training =  0.12094871289872679
MSE Validation =  0.08346927027058014

Average Performance in Training =  0.12094871289872679
Average Performance in Validation =  0.12094743559719015
```

```
M Value =  4
Sigma Value =  0.4
Mus= [0.2 0.3 0.4 0.5]

Fold  1
MSE Training =  0.12333362833256381
MSE Validation =  0.08413546769788714

Fold  2
MSE Training =  0.12333362833256381
MSE Validation =  0.20502418873014003

Fold  3
MSE Training =  0.12333362833256381
MSE Validation =  0.11579992926989051

Fold  4
MSE Training =  0.12333362833256381
MSE Validation =  0.08483389431876263

Average Performance in Training =  0.12333362833256381
Average Performance in Validation =  0.12244837000417008
```

```
M Value =  4
Sigma Value =  0.4
Mus= [0.3 0.4 0.5 0.6]
```

```
Fold  1
MSE Training =  0.12787767063129685
MSE Validation =  0.10921553833121028

Fold  2
MSE Training =  0.12787767063129685
MSE Validation =  0.08185131674953608

Fold  3
MSE Training =  0.12787767063129685
MSE Validation =  0.07840223031161768

Fold  4
MSE Training =  0.12787767063129685
MSE Validation =  0.247432304314644

Average Performance in Training =  0.12787767063129685
Average Performance in Validation =  0.129225347426752
```

```
M Value =  4
Sigma Value =  0.5
Mus= [0.  0.1 0.2 0.3]

Fold  1
MSE Training =  0.12078194807160539
MSE Validation =  0.13484629882064836

Fold  2
MSE Training =  0.12078194807160539
MSE Validation =  0.09910309867121002

Fold  3
MSE Training =  0.12078194807160539
MSE Validation =  0.15768108629445235

Fold  4
MSE Training =  0.12078194807160539
MSE Validation =  0.09213185005439017

Average Performance in Training =  0.12078194807160539
Average Performance in Validation =  0.12094058346017522
```

```
M Value =  4
Sigma Value =  0.5
Mus= [0.1 0.2 0.3 0.4]

Fold  1
MSE Training =  0.12208717940848461
MSE Validation =  0.1904604095971951

Fold  2
MSE Training =  0.12208717940848461
MSE Validation =  0.06786936364869794

Fold  3
MSE Training =  0.12208717940848461
MSE Validation =  0.08357111174119448

Fold  4
MSE Training =  0.12208717940848461
```

```
MSE Validation =  0.14526821477777394

Average Performance in Training =  0.12208717940848461
Average Performance in Validation =  0.12179227494121536
```

```
M Value =  4
Sigma Value =  0.5
Mus= [0.2 0.3 0.4 0.5]

Fold  1
MSE Training =  0.12439946387730974
MSE Validation =  0.0983434627161055

Fold  2
MSE Training =  0.12439946387730974
MSE Validation =  0.17860625965977778

Fold  3
MSE Training =  0.12439946387730974
MSE Validation =  0.1159994772991982

Fold  4
MSE Training =  0.12439946387730974
MSE Validation =  0.10230275628238555

Average Performance in Training =  0.12439946387730974
Average Performance in Validation =  0.12381298898936675
```

```
M Value =  4
Sigma Value =  0.5
Mus= [0.3 0.4 0.5 0.6]

Fold  1
MSE Training =  0.12772612012252615
MSE Validation =  0.11094130355575621

Fold  2
MSE Training =  0.12772612012252615
MSE Validation =  0.09908440557425435

Fold  3
MSE Training =  0.12772612012252615
MSE Validation =  0.10867936626657713

Fold  4
MSE Training =  0.12772612012252615
MSE Validation =  0.1959849493531037

Average Performance in Training =  0.12772612012252615
Average Performance in Validation =  0.12867250618742285
```

```
M Value =  4
Sigma Value =  0.7
Mus= [0.  0.1 0.2 0.3]

Fold  1
MSE Training =  0.12311300374988378
MSE Validation =  0.06393467781826237

Fold  2
```

```
MSE Training =  0.12311300374988378
MSE Validation =  0.15606514287355017

Fold  3
MSE Training =  0.12311300374988378
MSE Validation =  0.1223613378982868

Fold  4
MSE Training =  0.12311300374988378
MSE Validation =  0.15227637197676533

Average Performance in Training =  0.12311300374988378
Average Performance in Validation =  0.12365938264171616
```

```
M Value =  4
Sigma Value =  0.7
Mus= [0.1 0.2 0.3 0.4]

Fold  1
MSE Training =  0.1244475653923054
MSE Validation =  0.1002545146879405

Fold  2
MSE Training =  0.1244475653923054
MSE Validation =  0.08162595099171696

Fold  3
MSE Training =  0.1244475653923054
MSE Validation =  0.13377469999259203

Fold  4
MSE Training =  0.1244475653923054
MSE Validation =  0.18771965132238497

Average Performance in Training =  0.1244475653923054
Average Performance in Validation =  0.12584370424865862
```

```
M Value =  4
Sigma Value =  0.7
Mus= [0.2 0.3 0.4 0.5]

Fold  1
MSE Training =  0.1260765612154744
MSE Validation =  0.0439398256931056

Fold  2
MSE Training =  0.1260765612154744
MSE Validation =  0.09046128712524036

Fold  3
MSE Training =  0.1260765612154744
MSE Validation =  0.17234541759474087

Fold  4
MSE Training =  0.1260765612154744
MSE Validation =  0.2073723819165276

Average Performance in Training =  0.1260765612154744
Average Performance in Validation =  0.12852972808240362
```

```
M Value =  4
Sigma Value =  0.7
Mus= [0.3 0.4 0.5 0.6]

Fold  1
MSE Training =  0.12799684959472327
MSE Validation =  0.11438749578031288

Fold  2
MSE Training =  0.12799684959472327
MSE Validation =  0.09206025377718832

Fold  3
MSE Training =  0.12799684959472327
MSE Validation =  0.0617198933292436

Fold  4
MSE Training =  0.12799684959472327
MSE Validation =  0.2479485846281437

Average Performance in Training =  0.12799684959472327
Average Performance in Validation =  0.12902905687872213
```

In [25]:

```
print('Best avg performance in validation score:', np.min(avg_perf_val))
```

```
Best avg performance in validation score: 0.0109849564232764
```