Question 1

Load the data using pandas

In [1]:
```
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
```

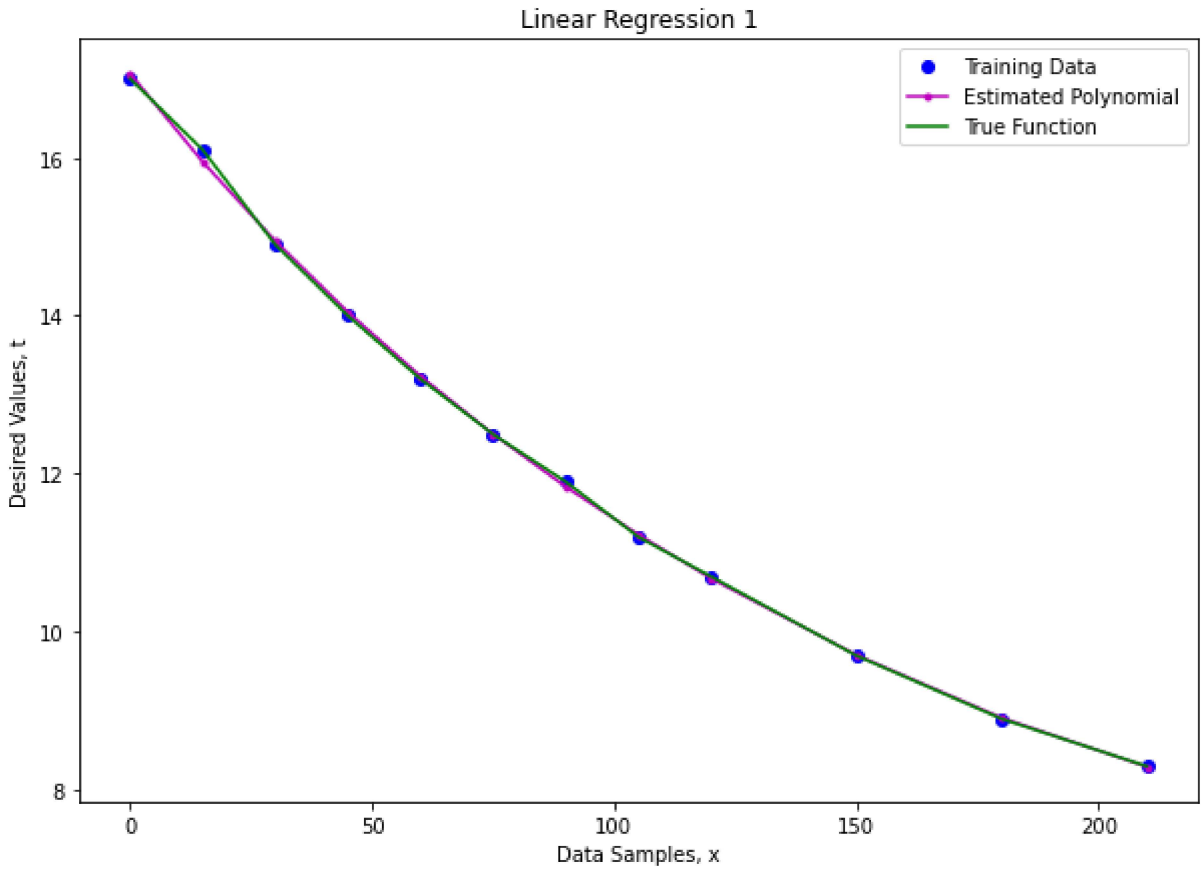
Train polynomial regression models (1-3)

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')
```

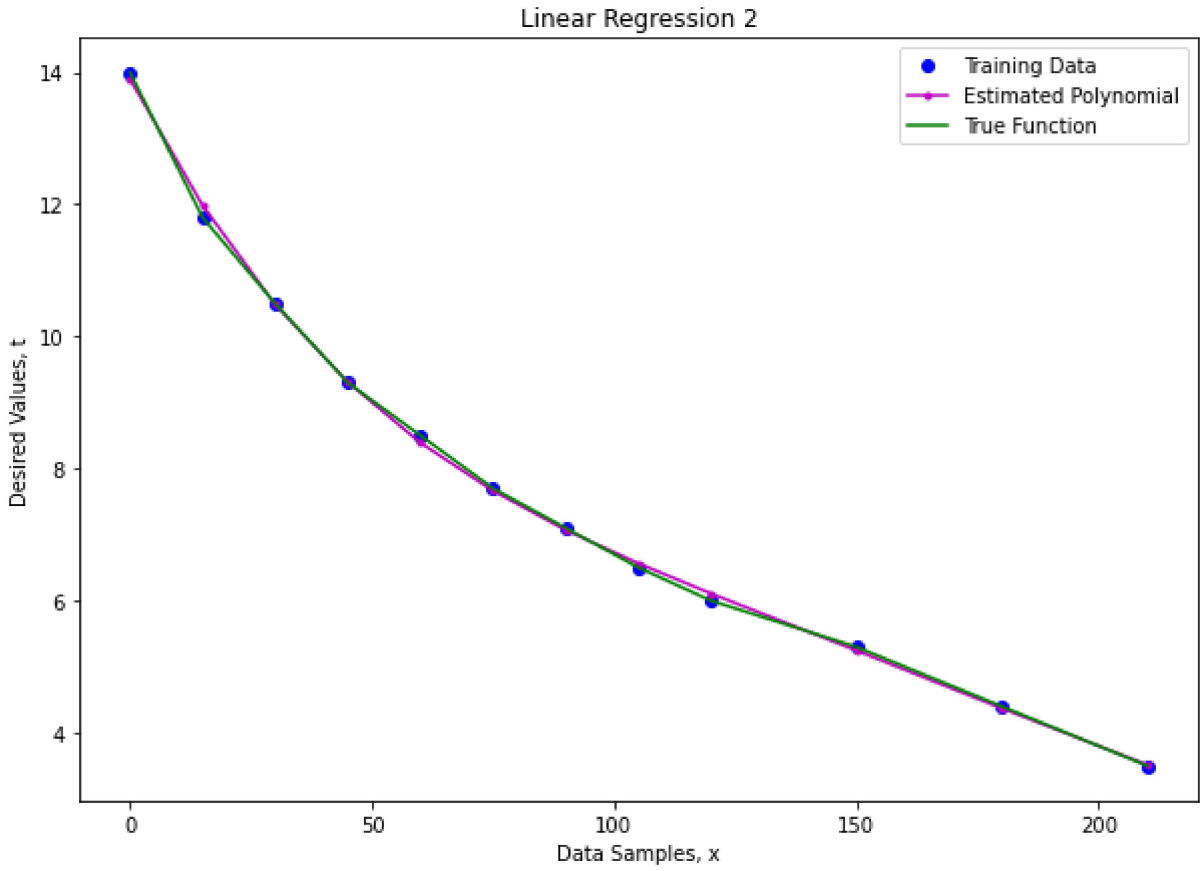
```
plt.ylabel('Desired Values, t');
plt.title('Linear Regression 1');
```



Model 2 (Brand 2)

In [7]:

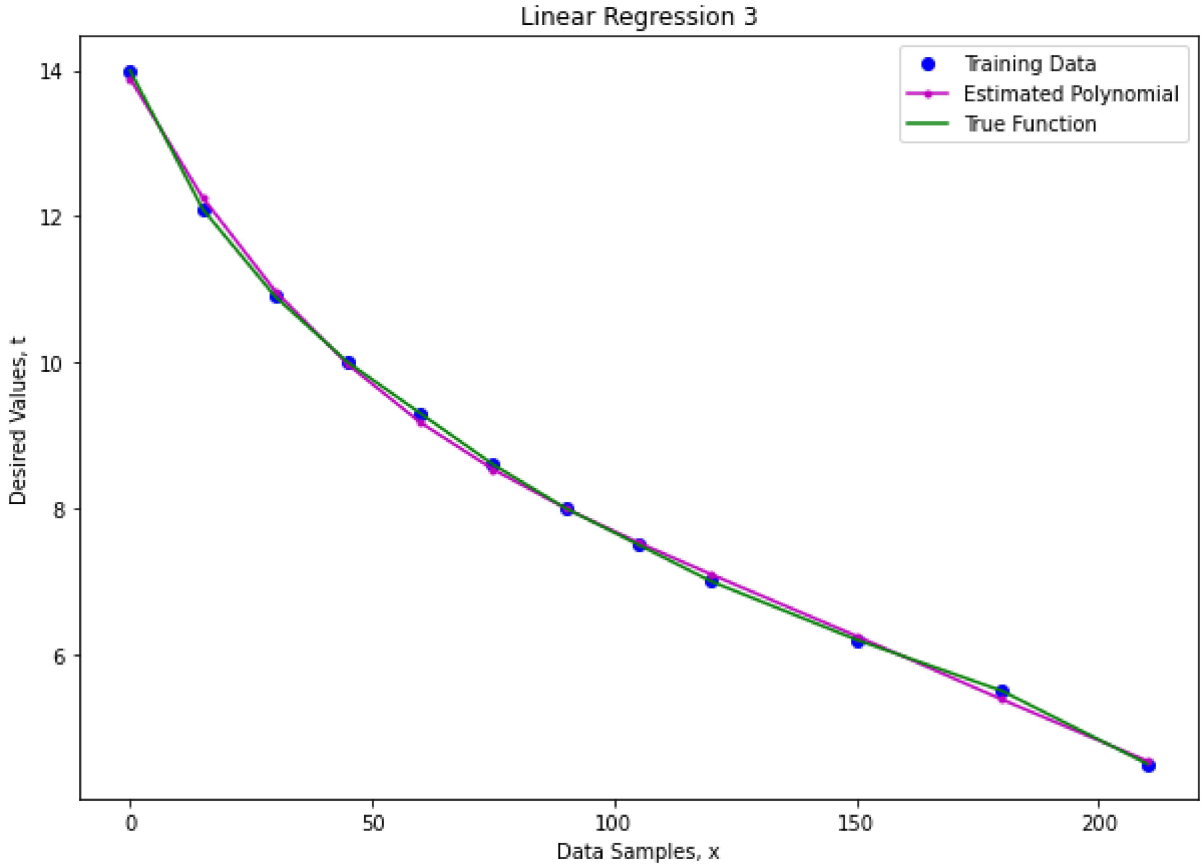
```
M = 4 #select model order
w2, y2_train = PolynomialRegression(x2_train,t2_train,M)

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 [35]:
```
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 [36]:
```
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)

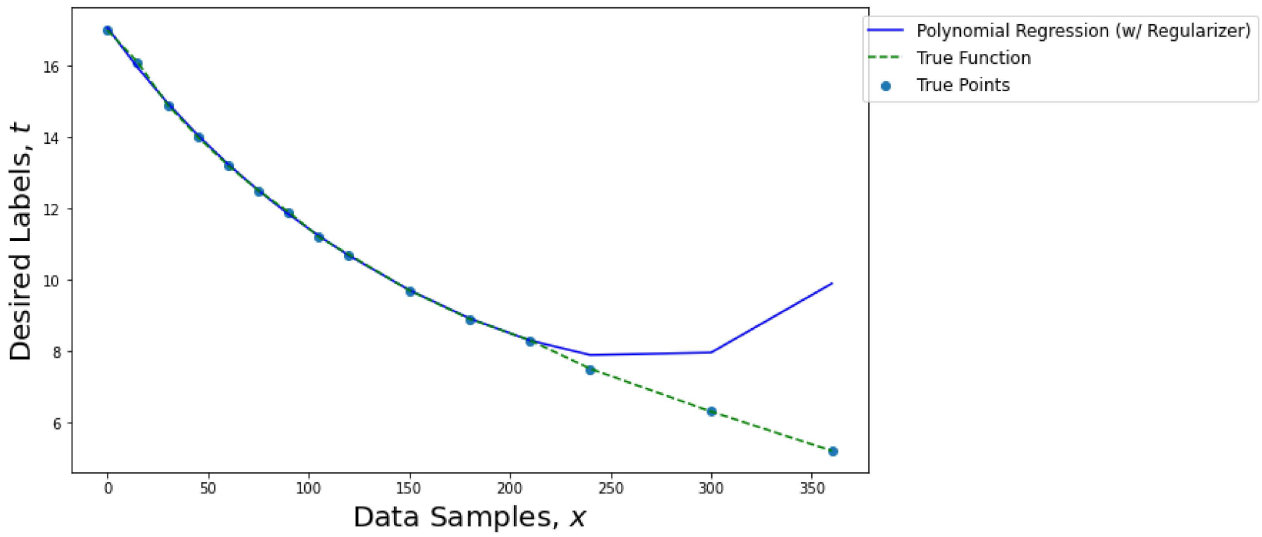
In [37]:
```
M = 4
test1_errors =[]
```

```
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 [38]:

```
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 [39]:

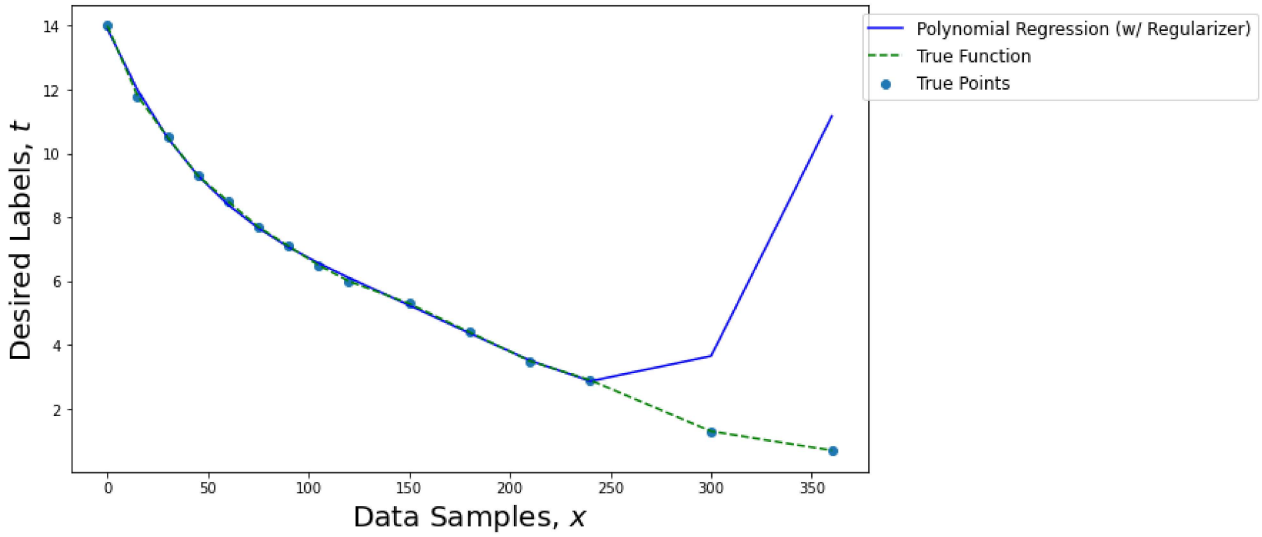
```
M = 4
test2_errors =[]
lams = [0.0]
lambdas = [[i] for i in lams]
for lams in lambdas:
    w2reg, y2reg = PolynomialRegression_reg(x2_train,t2_train,M,lams)
    test2_errors.append(0.5*np.sum((t2_train-y2reg)**2))
l2 = lambdas[np.argmin(test2_errors)]
print('Best Lambda:',l2)
```

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

In [40]:

```
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)

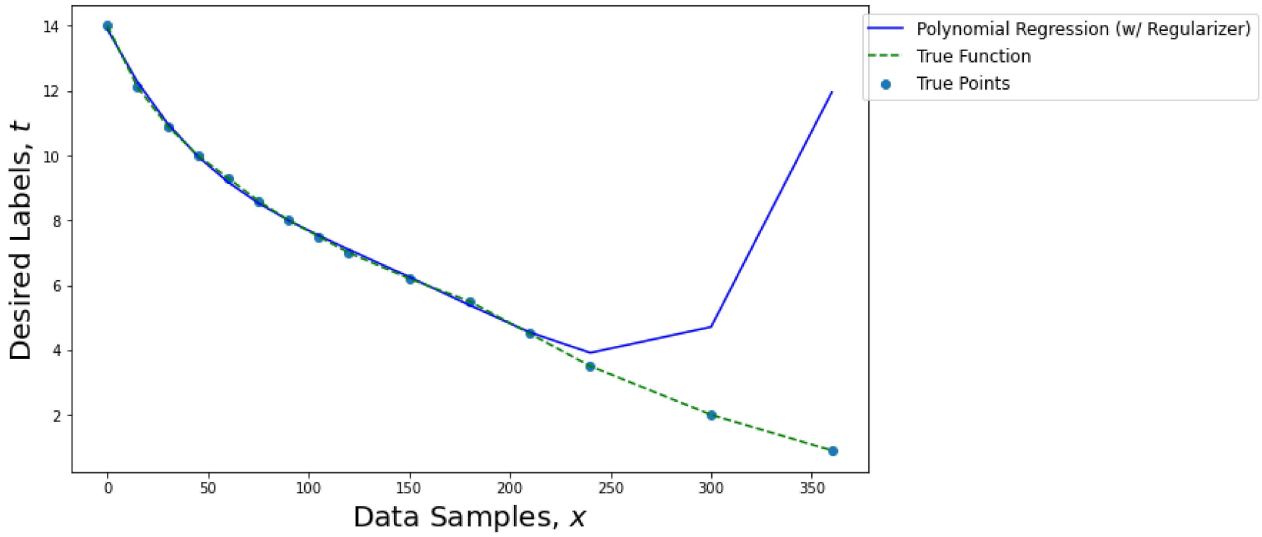
In [41]:

```
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 [42]:

```
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

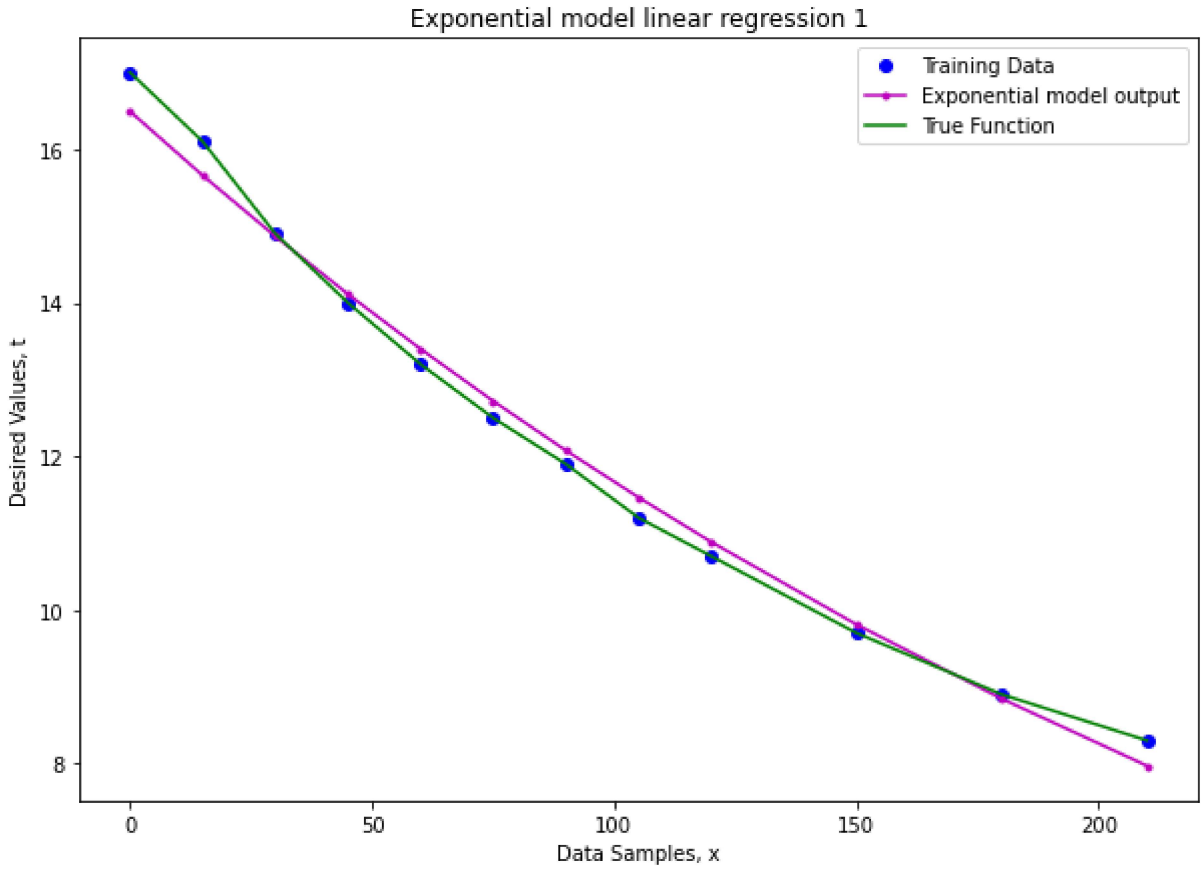
In [43]:

```
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 [44]:

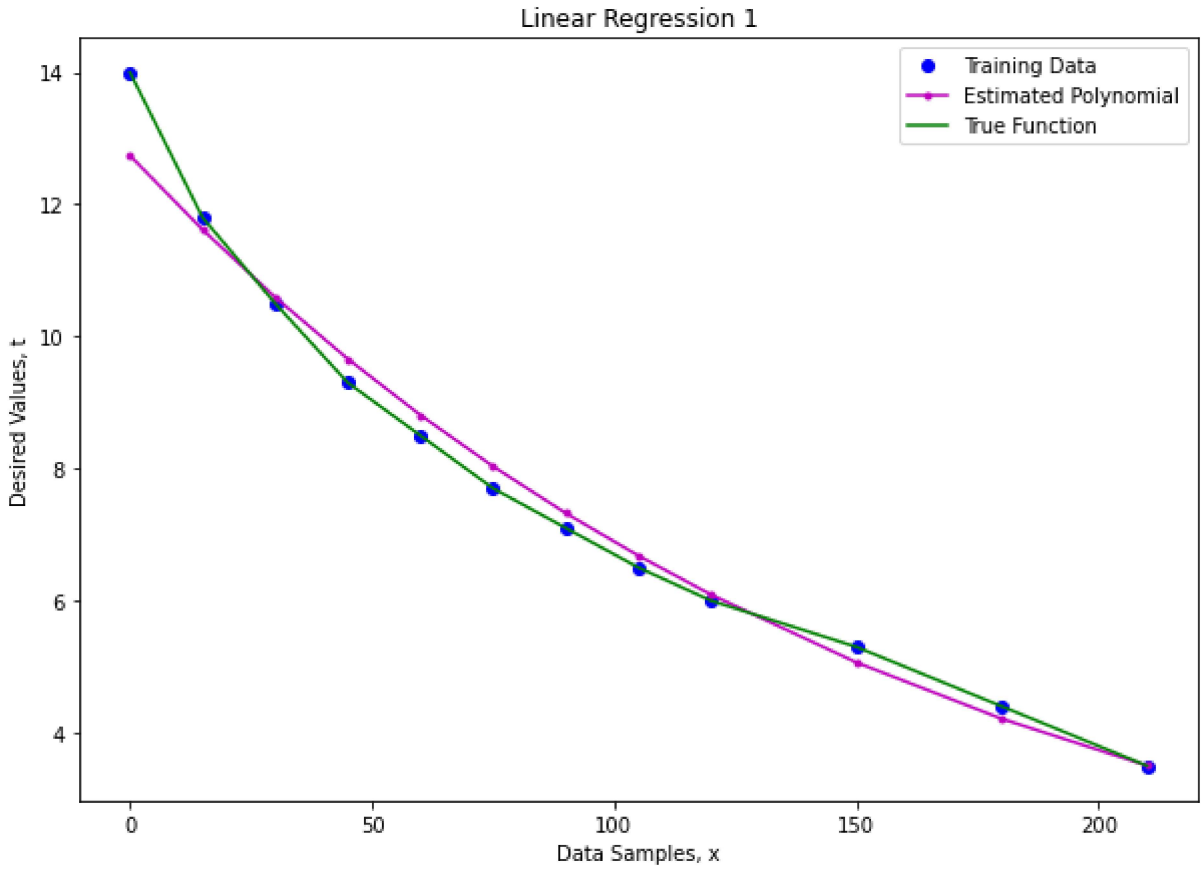
```
w1,y1 = LinRegression_Exp(x1_train,t1_train)

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 [45]:

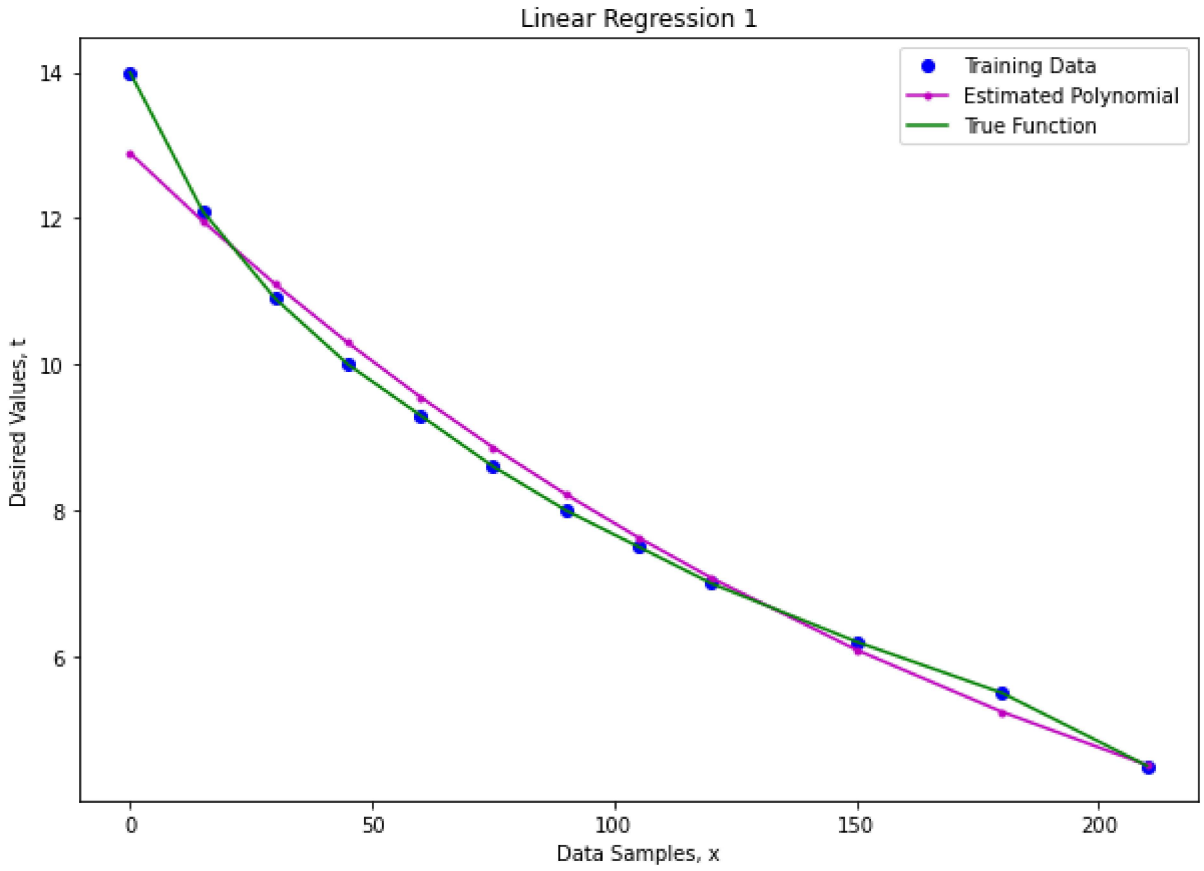
```
w2,y2 = LinRegression_Exp(x2_train,t2_train)

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 [46]:

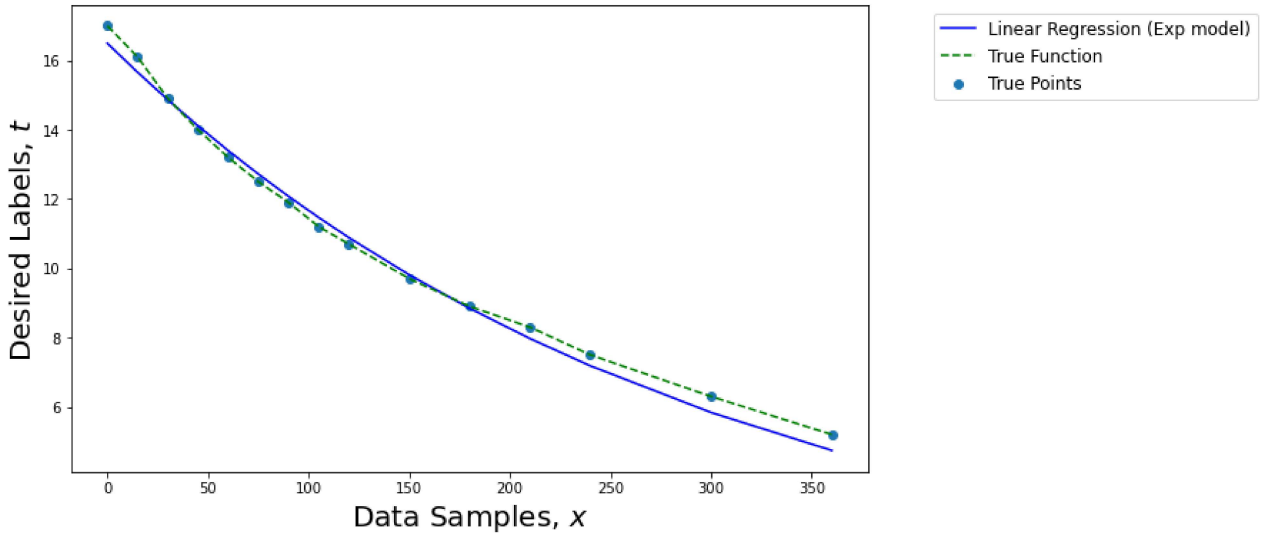
```
w3,y3 = LinRegression_Exp(x3_train,t3_train)

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 [47]:
```
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
```

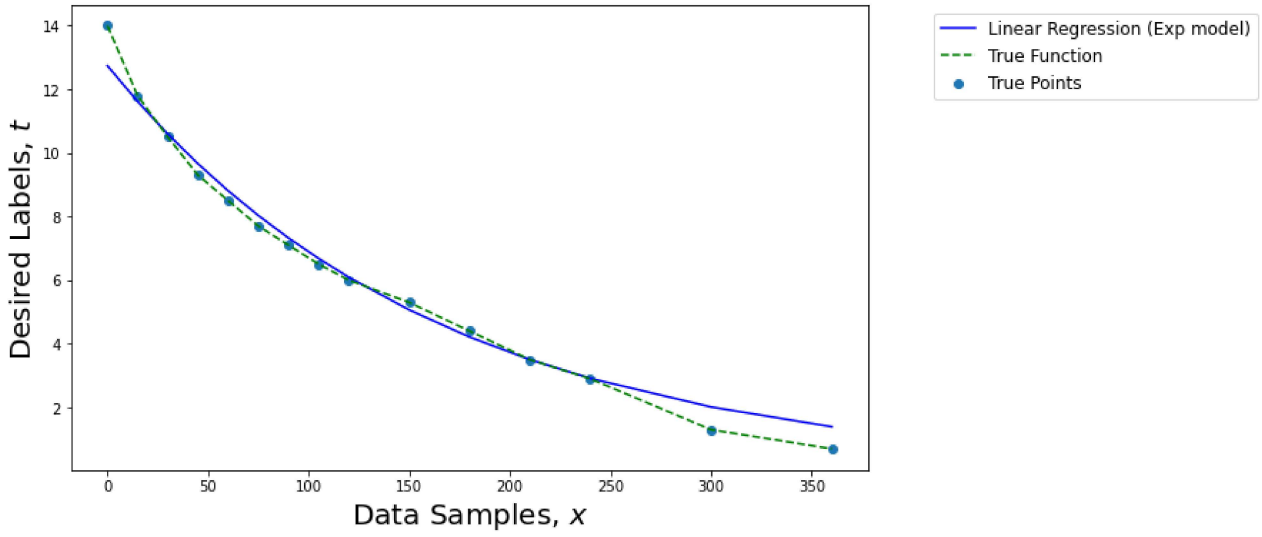
In [48]:
```
y1_pred_exp = LinRegressionExp_test(x1_true,w1)

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 [49]:

```
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)
```

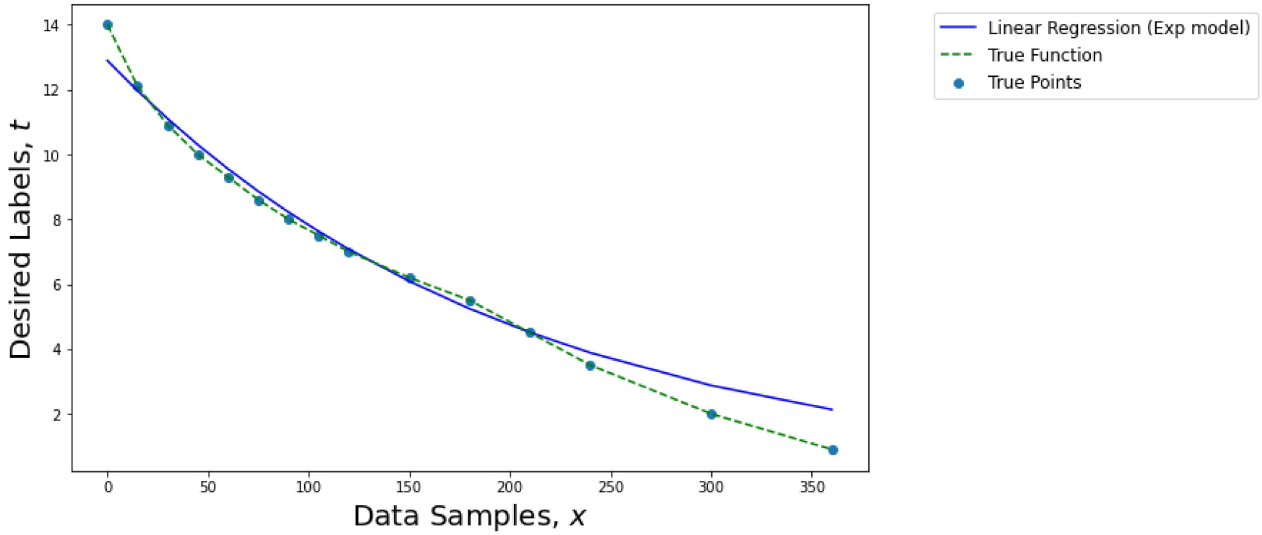


TEST Exp Model 3 (Brand 3)

In [50]:

```
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)
```



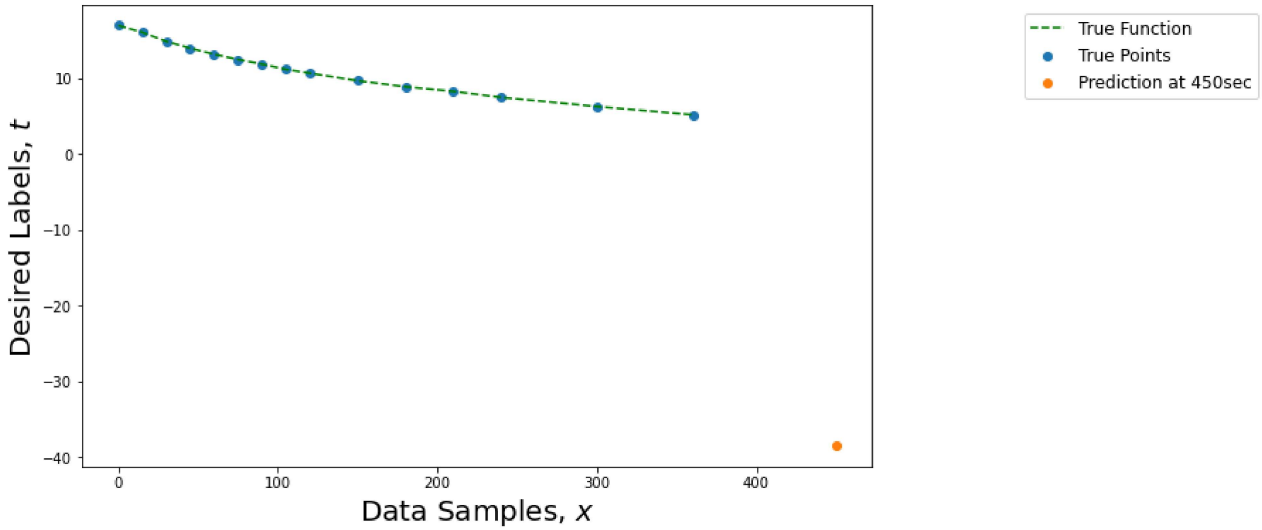
3. Predict height at 450sec

Polynomial Regression Model 1 (Brand 1)

In [51]:

```
#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);

print("Prediction, Polynomial Model 1:", y1_pred_reg)
```

```
Prediction, Polynomial Model 1: [-38.46463715]
```

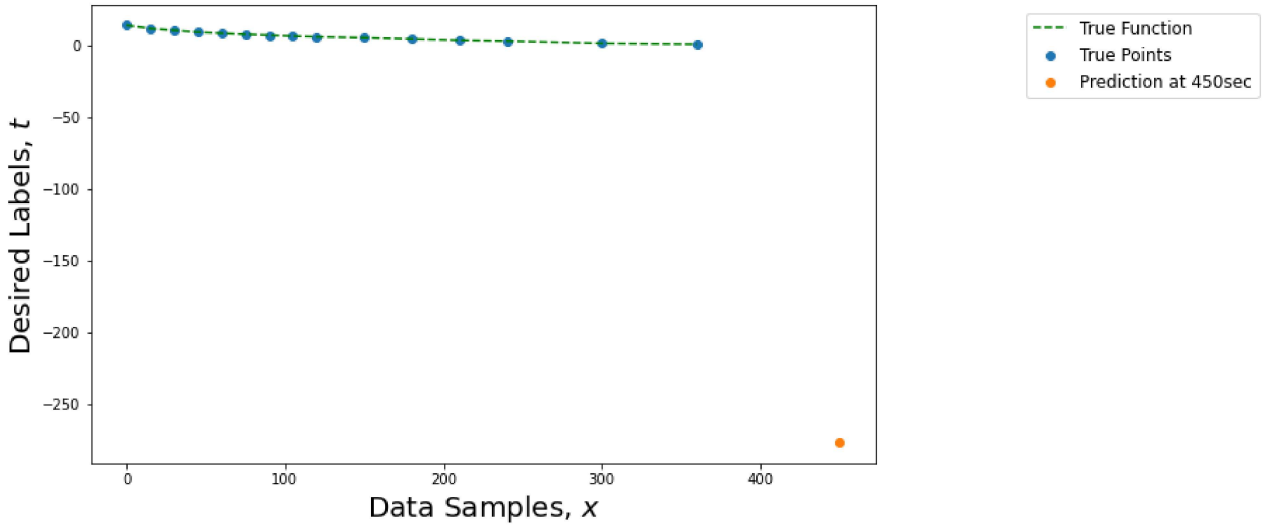


Polynomial Regression Model 2 (Brand 2)

In [52]:

```
#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);

print("Prediction, Polynomial Model 2:", y2_pred_reg)
```

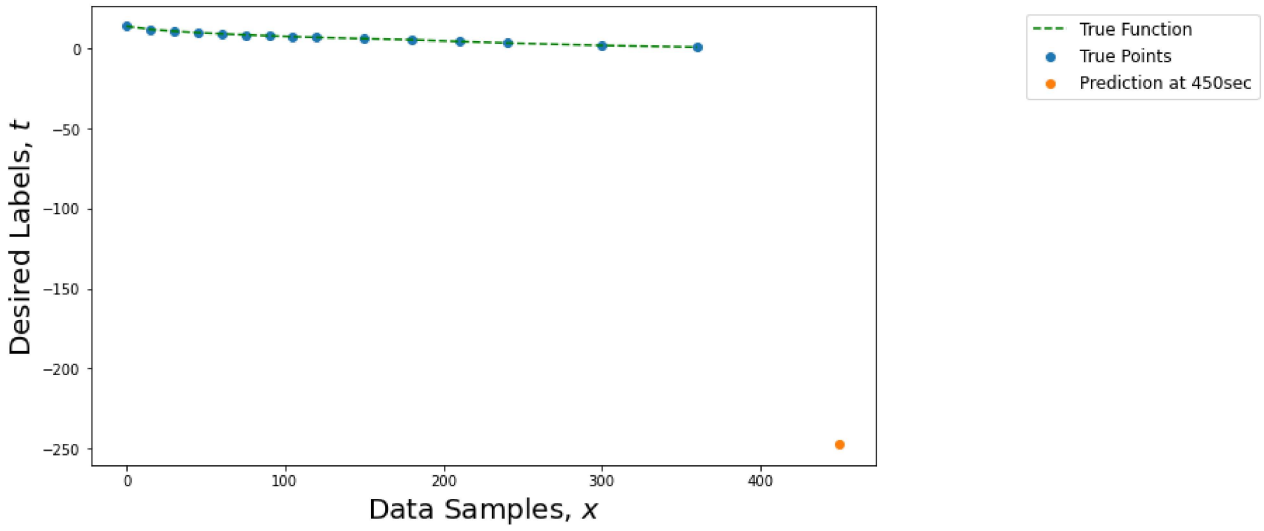
```
Prediction, Polynomial Model 2: [-276.7978397]
```



Polynomial Regression Model 3 (Brand 3)

In [53]:
```
#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);

print("Prediction, Polynomial Model 3:", y3_pred_reg)
```

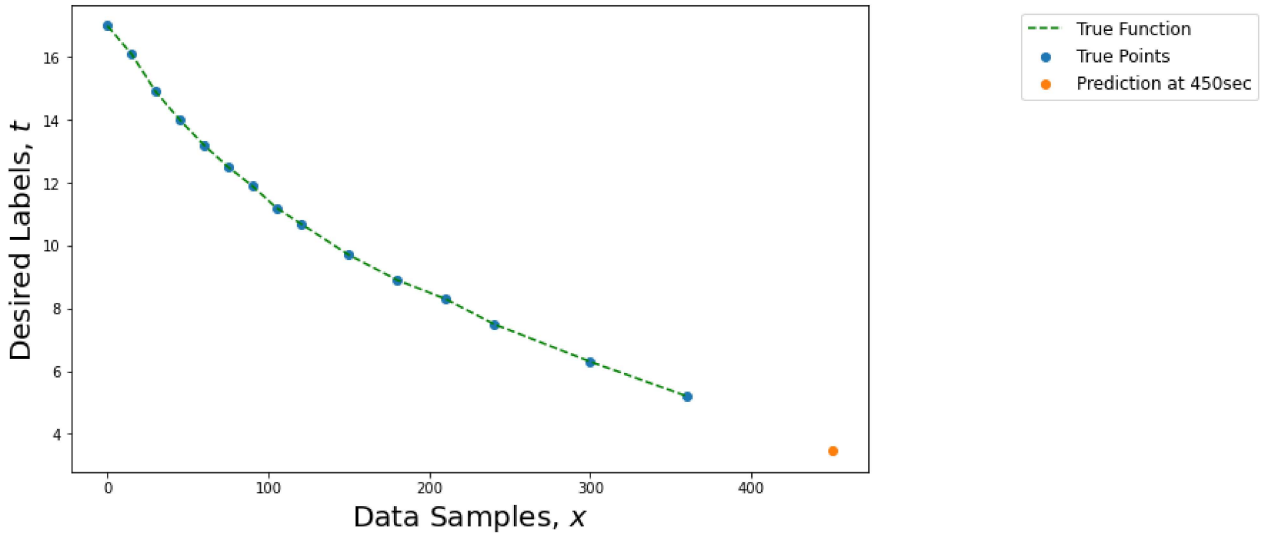
```
Prediction, Polynomial Model 3: [-247.2972587]
```



Exp Model 1 (Brand 1)

In [54]:
```
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);

print("Prediction, Exp Model 1:", y1reg)
```

```
Prediction, Exp Model 1: [3.47216008]
```

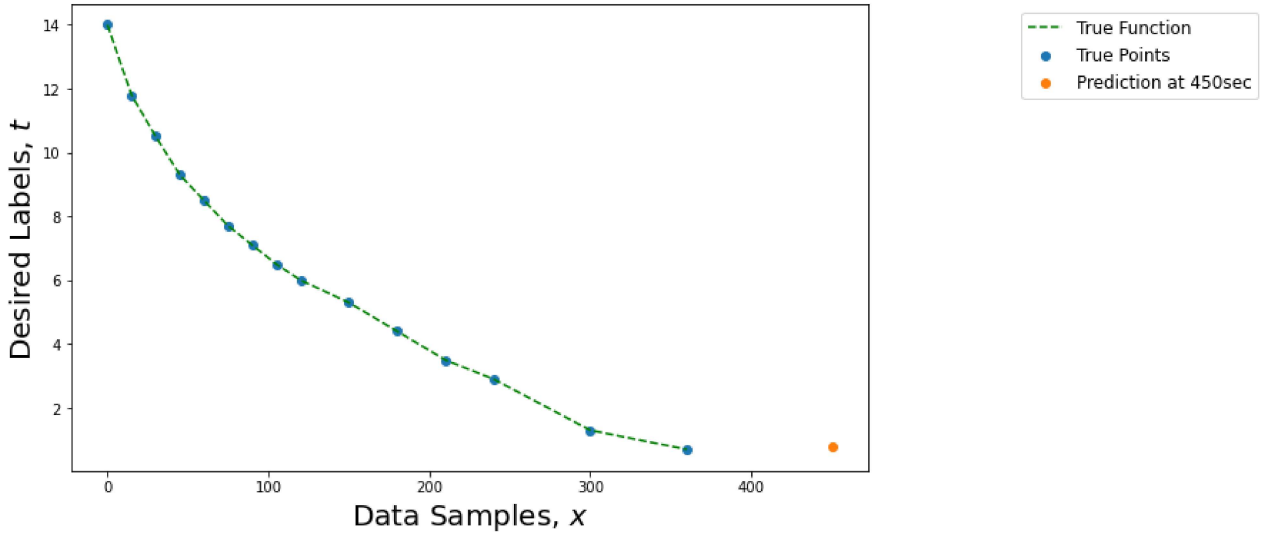


Exp Model 2 (Brand 2)

In [55]:

```
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);

print("Prediction, Exp Model 2:", y2reg)
```

```
Prediction, Exp Model 2: [0.80063714]
```



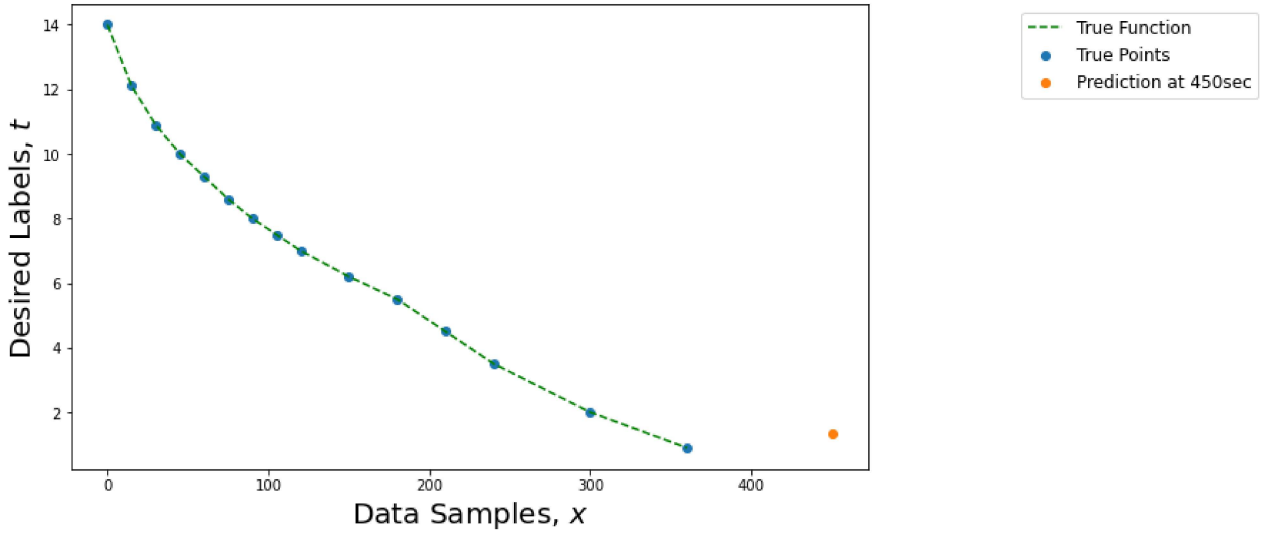
Exp Model 3 (Brand 3)

In [56]:

```
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);

print("Prediction, Exp Model 3:", y3reg)
```

```
Prediction, Exp Model 3: [1.35740203]
```



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

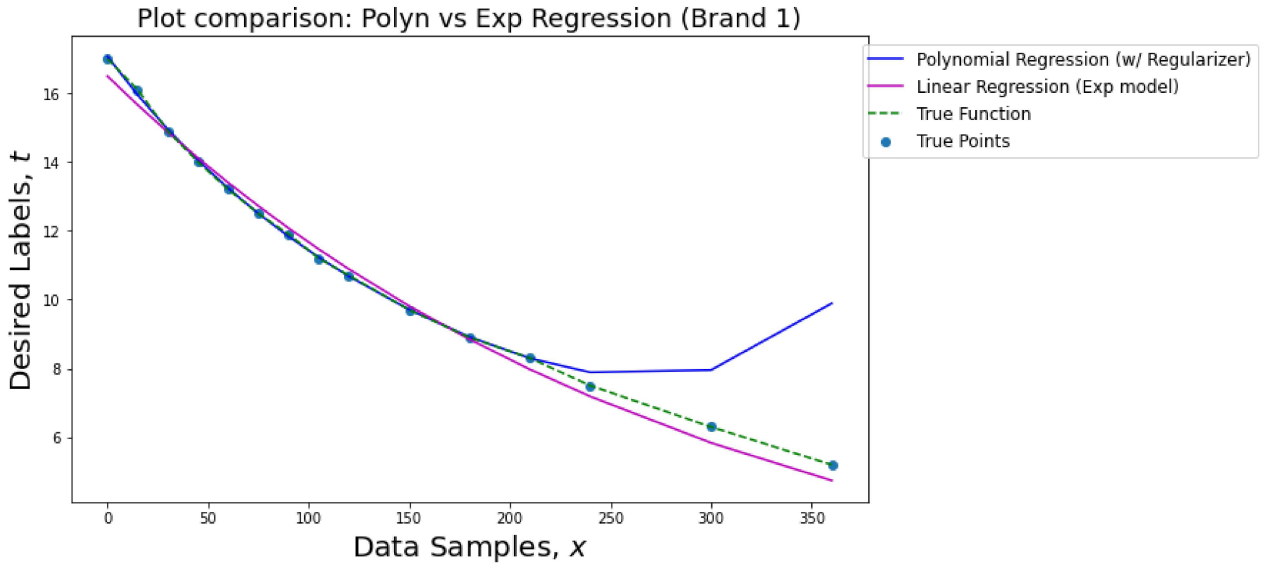
Plot (qualitative) comparison (paragraph below the three plots)

Polynomial Model 1 vs Exponential Model 1

In [57]:
```
y1_pred = PolynomialRegression_test(x1_true, M, w1reg)
y1_pred_exp = LinRegressionExp_test(x1_true,w1)
```

In [58]:
```
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[58]:
```
Text(0.5, 1.0, 'Plot comparison: Polyn vs Exp Regression (Brand 1)')
```

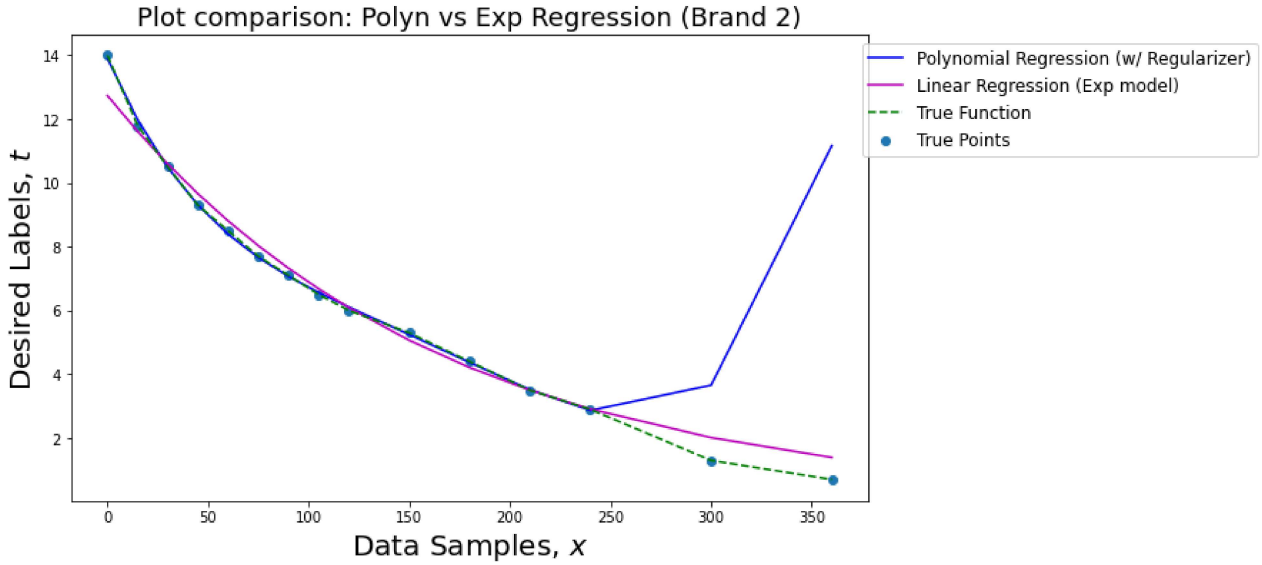


Polynomial Model 2 vs Exponential Model 2

In [59]:
```
y2_pred = PolynomialRegression_test(x2_true, M, w2reg)
y2_pred_exp = LinRegressionExp_test(x2_true,w2)
```

In [60]:
```
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[60]:
```
Text(0.5, 1.0, 'Plot comparison: Polyn vs Exp Regression (Brand 2)')
```

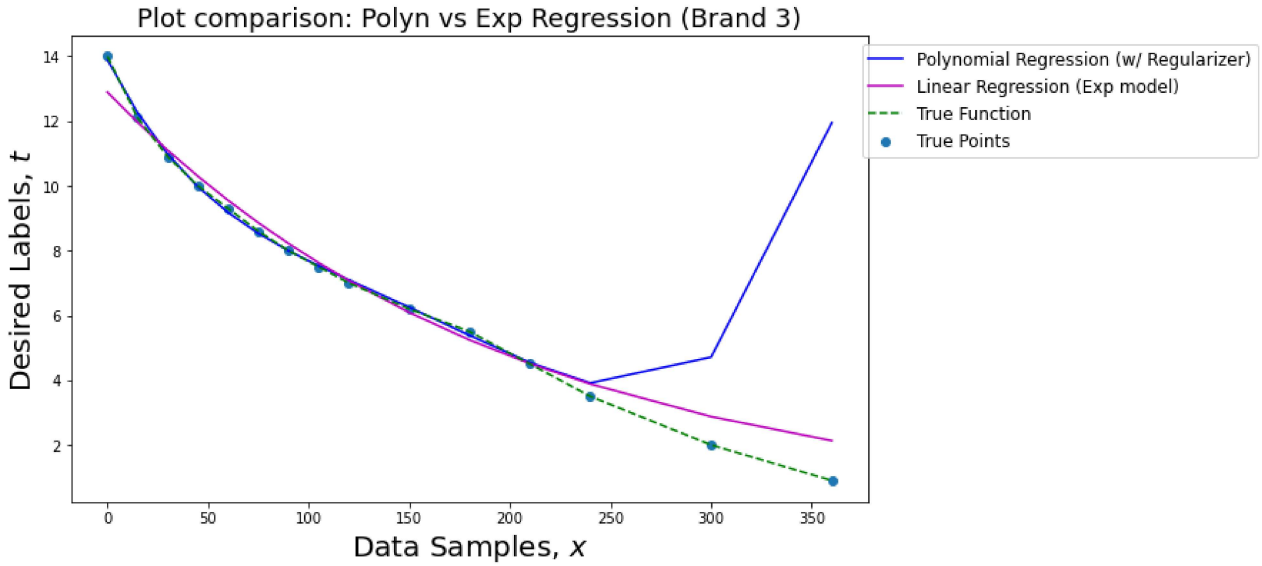


Polynomial Model 3 vs Exponential Model 3

In [61]:
```
y3_pred = PolynomialRegression_test(x3_true, M, w3reg)
y3_pred_exp = LinRegressionExp_test(x3_true,w3)
```

In [62]:
```
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[62]:
```
Text(0.5, 1.0, 'Plot comparison: Polyn vs Exp Regression (Brand 3)')
```



Qualitative comparison of three models (polynomial vs exponential)

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 polynomial model with its low training error and high test error exhibits high variance. Given that we're trainig on data that comes from a suspected exponentially decaying curve, the polynomial model may lack the number of training data to learn such a curve, whereas the exponential model is inherently set up for success in fitting and predicting on data arising from an exponential curve. The 4th order polynomial model could benefit from use of a regularizer to reduce its overfitting/high variance.

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 [63]:
```
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 [64]:
```
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 [65]:
```
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 [66]:
```
print('Average of MSE scores (Exponential Models):',(1/3)*(mse1_exp + mse2_exp + mse3_e
```

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

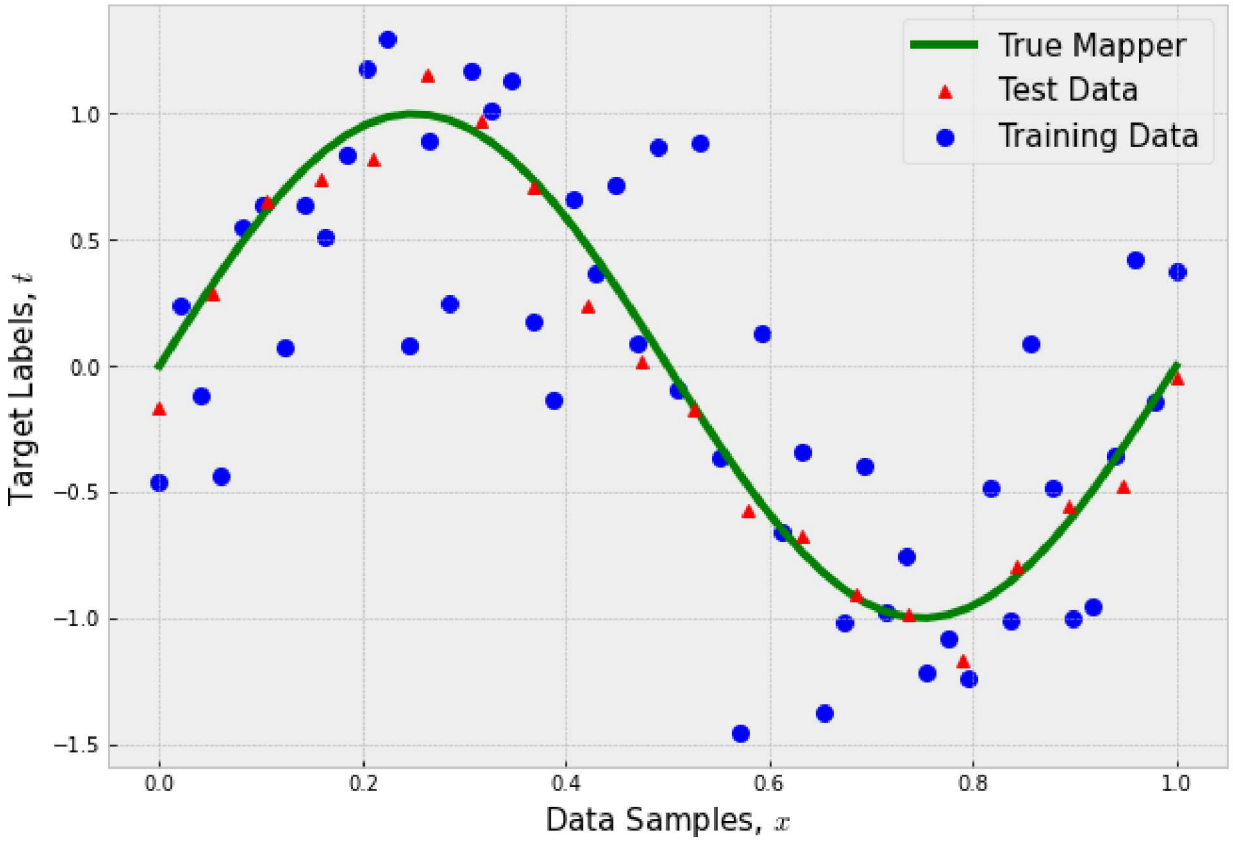
The best average MSE score goes to the class of exponential regression models. Regarding the prediction accuracy, prior knowledge of the exponential fit of $H(t) = H_0 e^{-\lambda t}$ may imply that the exponential model will provide a better prediction for future value. The exponential model prediction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

Question 2

In [67]:
```
%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 [68]:
```
# 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 [69]:

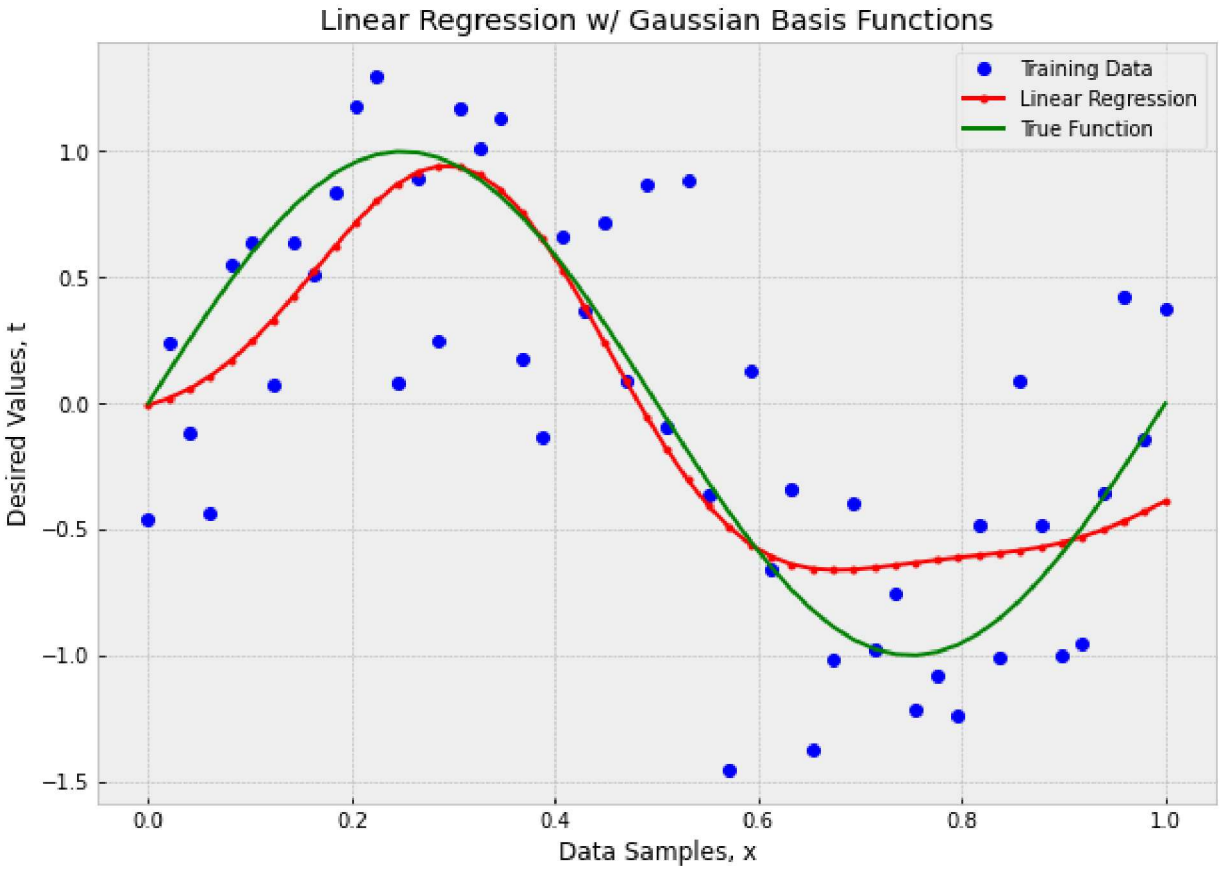
```
def LinearRegression_Gaussian(x,t,M,sigma,mu):
    '''Fit a Gaussian model with range of mu to the data input data x and desire values

    # 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 [70]:

```
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

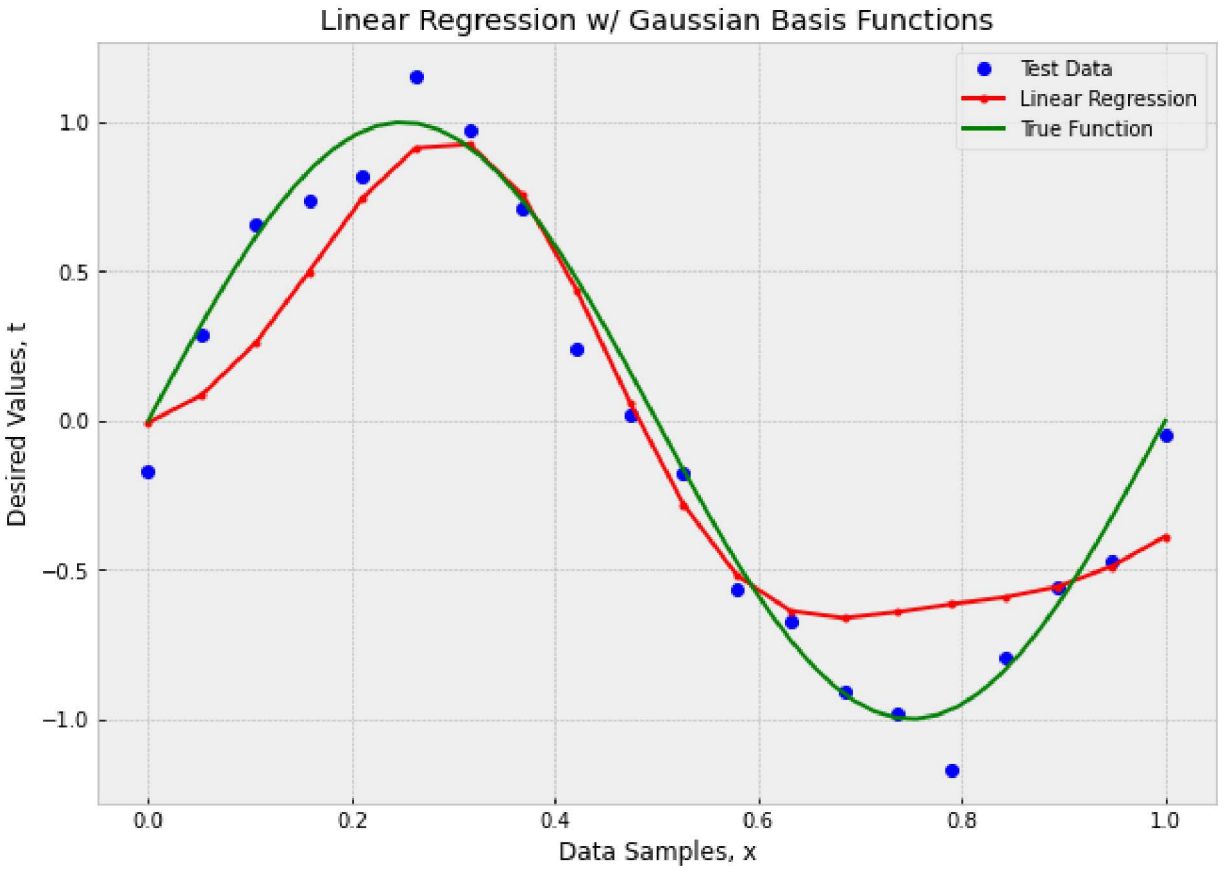
In [71]:

```
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 [72]:

```
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]. K-fold cross-validation divides a dataset into k equal sections. Within those sections a revolving part of the data is used for training and revolving part for validation, so that the training data and validation data do not ever coincide. Each validation fold iterates over the parameters of interest, and the average of validation scores for each set of parameters is calculated. We 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 [73]:
```
from sklearn.model_selection import KFold

k = 4 # number of folds

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

In [74]:
```
# 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 [75]:
```
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

            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 =  0.5209921083004455
MSE Validation =  0.4512117253114576

Fold  2
MSE Training =  0.5209921083004455
MSE Validation =  0.8329030019656434

Fold  3
MSE Training =  0.5209921083004455
MSE Validation =  0.5415051731621399

Fold  4
MSE Training =  0.5209921083004455
MSE Validation =  0.2381709902061907

Average Performance in Training =  0.5209921083004455
Average Performance in Validation =  0.5159477226613579
_____________________________________________________________
M Value =  1
Sigma Value =  0.3
Mus= [0.1 0.2 0.3 0.4]

Fold  1
MSE Training =  0.5306376966155748
MSE Validation =  0.4285864782729699

Fold  2
MSE Training =  0.5306376966155748
MSE Validation =  0.639130619534273

Fold  3
MSE Training =  0.5306376966155748
MSE Validation =  0.510392778283579

Fold  4
MSE Training =  0.5306376966155748
MSE Validation =  0.5439041016568028

Average Performance in Training =  0.5306376966155748
Average Performance in Validation =  0.5305034944369061
_____________________________________________________________
M Value =  1
Sigma Value =  0.3
Mus= [0.2 0.3 0.4 0.5]

Fold  1
MSE Training =  0.5427958273616742
MSE Validation =  0.76717776918285
```

```
Fold  2
MSE Training =  0.5427958273616742
MSE Validation =  0.4345430670607985

Fold  3
MSE Training =  0.5427958273616742
MSE Validation =  0.5039460963153913

Fold  4
MSE Training =  0.5427958273616742
MSE Validation =  0.45583894509429856

Average Performance in Training =  0.5427958273616742
Average Performance in Validation =  0.5403764694133346
```

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

Fold  1
MSE Training =  0.5549533949475138
MSE Validation =  0.7654964796100144

Fold  2
MSE Training =  0.5549533949475138
MSE Validation =  0.37926381827636174

Fold  3
MSE Training =  0.5549533949475138
MSE Validation =  0.6444463495503174

Fold  4
MSE Training =  0.5549533949475138
MSE Validation =  0.4277024733540828

Average Performance in Training =  0.5549533949475138
Average Performance in Validation =  0.5542272801976941
```

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

Fold  1
MSE Training =  0.5466873347805428
MSE Validation =  0.570544535299342

Fold  2
MSE Training =  0.5466873347805428
MSE Validation =  0.7281932241349961

Fold  3
MSE Training =  0.5466873347805428
MSE Validation =  0.531524053622501

Fold  4
MSE Training =  0.5466873347805428
MSE Validation =  0.33937393524256093

Average Performance in Training =  0.5466873347805428
```

```
Average Performance in Validation =  0.54240893707485
--------------------------------------------------------------------------------
M Value =  1
Sigma Value =  0.4
Mus= [0.1 0.2 0.3 0.4]

Fold  1
MSE Training =  0.5530861802003132
MSE Validation =  0.25256258897567435

Fold  2
MSE Training =  0.5530861802003132
MSE Validation =  0.6635057626960007

Fold  3
MSE Training =  0.5530861802003132
MSE Validation =  0.3192846497546739

Fold  4
MSE Training =  0.5530861802003132
MSE Validation =  0.992833720102316

Average Performance in Training =  0.5530861802003132
Average Performance in Validation =  0.5570466803821662
--------------------------------------------------------------------------------
M Value =  1
Sigma Value =  0.4
Mus= [0.2 0.3 0.4 0.5]

Fold  1
MSE Training =  0.558854737549328
MSE Validation =  0.3869737185675193

Fold  2
MSE Training =  0.558854737549328
MSE Validation =  0.7179579649020914

Fold  3
MSE Training =  0.558854737549328
MSE Validation =  0.5620385026932496

Fold  4
MSE Training =  0.558854737549328
MSE Validation =  0.5695135800035387

Average Performance in Training =  0.558854737549328
Average Performance in Validation =  0.5591209415415997
--------------------------------------------------------------------------------
M Value =  1
Sigma Value =  0.4
Mus= [0.3 0.4 0.5 0.6]

Fold  1
MSE Training =  0.5632714660600581
MSE Validation =  0.6173490115422117

Fold  2
MSE Training =  0.5632714660600581
MSE Validation =  0.6076578473235327
```

```
Fold  3
MSE Training =  0.5632714660600581
MSE Validation =  0.7406592432977886

Fold  4
MSE Training =  0.5632714660600581
MSE Validation =  0.2792144348478969

Average Performance in Training =  0.5632714660600581
Average Performance in Validation =  0.5612201342528574
```

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

Fold  1
MSE Training =  0.558183947211581
MSE Validation =  0.7584671946101046

Fold  2
MSE Training =  0.558183947211581
MSE Validation =  0.5098761367630147

Fold  3
MSE Training =  0.558183947211581
MSE Validation =  0.4579229726896112

Fold  4
MSE Training =  0.558183947211581
MSE Validation =  0.4938048650377642

Average Performance in Training =  0.558183947211581
Average Performance in Validation =  0.5550177922751237
```

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

Fold  1
MSE Training =  0.5612907238696498
MSE Validation =  0.4605494907357207

Fold  2
MSE Training =  0.5612907238696498
MSE Validation =  0.8602613815484166

Fold  3
MSE Training =  0.5612907238696498
MSE Validation =  0.3902105676414878

Fold  4
MSE Training =  0.5612907238696498
MSE Validation =  0.5176223368409044

Average Performance in Training =  0.5612907238696498
Average Performance in Validation =  0.5571609441916323
```

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

```
Fold  1
MSE Training =  0.5637339717021149
MSE Validation =  0.4744877763658806

Fold  2
MSE Training =  0.5637339717021149
MSE Validation =  0.5353901639565428

Fold  3
MSE Training =  0.5637339717021149
MSE Validation =  0.4378326425815675

Fold  4
MSE Training =  0.5637339717021149
MSE Validation =  0.8170244708279529

Average Performance in Training =  0.5637339717021149
Average Performance in Validation =  0.5661837634329859
```

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

Fold  1
MSE Training =  0.5653372073025832
MSE Validation =  0.8507782990951612

Fold  2
MSE Training =  0.5653372073025832
MSE Validation =  0.4344266985717779

Fold  3
MSE Training =  0.5653372073025832
MSE Validation =  0.5177916283227191

Fold  4
MSE Training =  0.5653372073025832
MSE Validation =  0.445474654632194

Average Performance in Training =  0.5653372073025832
Average Performance in Validation =  0.5621178201554631
```

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

Fold  1
MSE Training =  0.564698394718472
MSE Validation =  0.4274948657157864

Fold  2
MSE Training =  0.564698394718472
MSE Validation =  0.47655221704733874

Fold  3
MSE Training =  0.564698394718472
MSE Validation =  0.6470413911652685

Fold  4
```

```
MSE Training =  0.564698394718472
MSE Validation =  0.726484247168313

Average Performance in Training =  0.564698394718472
Average Performance in Validation =  0.5693931802741766
```

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

Fold  1
MSE Training =  0.5653787702337926
MSE Validation =  0.3716460144321766

Fold  2
MSE Training =  0.5653787702337926
MSE Validation =  0.5694169573621044

Fold  3
MSE Training =  0.5653787702337926
MSE Validation =  0.6516588173420576

Fold  4
MSE Training =  0.5653787702337926
MSE Validation =  0.6846011725216066

Average Performance in Training =  0.5653787702337926
Average Performance in Validation =  0.5693307404144863
```

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

Fold  1
MSE Training =  0.5658054792344569
MSE Validation =  0.42400284006569405

Fold  2
MSE Training =  0.5658054792344569
MSE Validation =  0.9448441756155846

Fold  3
MSE Training =  0.5658054792344569
MSE Validation =  0.42729277243359437

Fold  4
MSE Training =  0.5658054792344569
MSE Validation =  0.4473124573885901

Average Performance in Training =  0.5658054792344569
Average Performance in Validation =  0.5608630613758658
```

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

Fold  1
MSE Training =  0.5659637739161966
MSE Validation =  0.5236908676432387
```

```
Fold  2
MSE Training =  0.5659637739161966
MSE Validation =  0.3340840708933724

Fold  3
MSE Training =  0.5659637739161966
MSE Validation =  0.8348910001525502

Fold  4
MSE Training =  0.5659637739161966
MSE Validation =  0.594035207750274

Average Performance in Training =  0.5659637739161966
Average Performance in Validation =  0.5716752866098589
```

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

Fold  1
MSE Training =  0.473440505636947
MSE Validation =  0.5057614452319253

Fold  2
MSE Training =  0.473440505636947
MSE Validation =  0.2772692522163459

Fold  3
MSE Training =  0.473440505636947
MSE Validation =  0.6027946731679951

Fold  4
MSE Training =  0.473440505636947
MSE Validation =  0.5215908447503236

Average Performance in Training =  0.473440505636947
Average Performance in Validation =  0.4768540538416475
```

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

Fold  1
MSE Training =  0.4398924114606817
MSE Validation =  0.40755602851147216

Fold  2
MSE Training =  0.4398924114606817
MSE Validation =  0.5921458748706889

Fold  3
MSE Training =  0.4398924114606817
MSE Validation =  0.3623840267822342

Fold  4
MSE Training =  0.4398924114606817
MSE Validation =  0.3874906256399317

Average Performance in Training =  0.4398924114606817
Average Performance in Validation =  0.43739413895108176
```

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

Fold  1
MSE Training =  0.4022215492527587
MSE Validation =  0.32536962111770146

Fold  2
MSE Training =  0.4022215492527587
MSE Validation =  0.3485644805551034

Fold  3
MSE Training =  0.4022215492527587
MSE Validation =  0.4578392690874536

Fold  4
MSE Training =  0.4022215492527587
MSE Validation =  0.4879885759868357

Average Performance in Training =  0.4022215492527587
Average Performance in Validation =  0.4049404866867735
```

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

Fold  1
MSE Training =  0.3677315767539983
MSE Validation =  0.3833622161725408

Fold  2
MSE Training =  0.3677315767539983
MSE Validation =  0.3641984629343469

Fold  3
MSE Training =  0.3677315767539983
MSE Validation =  0.377332630489031

Fold  4
MSE Training =  0.3677315767539983
MSE Validation =  0.345024870286834

Average Performance in Training =  0.3677315767539983
Average Performance in Validation =  0.36747954497068813
```

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

Fold  1
MSE Training =  0.4287852209898322
MSE Validation =  0.3377233870415476

Fold  2
MSE Training =  0.4287852209898322
MSE Validation =  0.4975071967341708

Fold  3
```

```
MSE Training =  0.4287852209898322
MSE Validation =  0.4919082655369475

Fold  4
MSE Training =  0.4287852209898322
MSE Validation =  0.3898636894969918

Average Performance in Training =  0.4287852209898322
Average Performance in Validation =  0.4292506347024144
```

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

Fold  1
MSE Training =  0.4099027043045315
MSE Validation =  0.3034749857318131

Fold  2
MSE Training =  0.4099027043045315
MSE Validation =  0.3165010587415127

Fold  3
MSE Training =  0.4099027043045315
MSE Validation =  0.5317817791873586

Fold  4
MSE Training =  0.4099027043045315
MSE Validation =  0.5045054405687533

Average Performance in Training =  0.4099027043045315
Average Performance in Validation =  0.4140658160573595
```

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

Fold  1
MSE Training =  0.39320643052031995
MSE Validation =  0.4447785173157875

Fold  2
MSE Training =  0.39320643052031995
MSE Validation =  0.2606466868383211

Fold  3
MSE Training =  0.39320643052031995
MSE Validation =  0.2429105102575065

Fold  4
MSE Training =  0.39320643052031995
MSE Validation =  0.6312389790768754

Average Performance in Training =  0.39320643052031995
Average Performance in Validation =  0.39489367337212267
```

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

```
Fold  1
MSE Training =  0.37995077380716213
MSE Validation =  0.23657225843652666

Fold  2
MSE Training =  0.37995077380716213
MSE Validation =  0.5472987798778196

Fold  3
MSE Training =  0.37995077380716213
MSE Validation =  0.2734077572539187

Fold  4
MSE Training =  0.37995077380716213
MSE Validation =  0.4605268421020483

Average Performance in Training =  0.37995077380716213
Average Performance in Validation =  0.3794514094175783
```

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

Fold  1
MSE Training =  0.4118682261300172
MSE Validation =  0.3316746973821172

Fold  2
MSE Training =  0.4118682261300172
MSE Validation =  0.2860289612979597

Fold  3
MSE Training =  0.4118682261300172
MSE Validation =  0.33092891747437947

Fold  4
MSE Training =  0.4118682261300172
MSE Validation =  0.7160097278306093

Average Performance in Training =  0.4118682261300172
Average Performance in Validation =  0.41616057599626644
```

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

Fold  1
MSE Training =  0.40253410292183117
MSE Validation =  0.46582574554823586

Fold  2
MSE Training =  0.40253410292183117
MSE Validation =  0.35728653453152304

Fold  3
MSE Training =  0.40253410292183117
MSE Validation =  0.4940191194638155

Fold  4
MSE Training =  0.40253410292183117
```

```
MSE Validation =  0.2915013392907425

Average Performance in Training =  0.40253410292183117
Average Performance in Validation =  0.4021581847085792
```

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

Fold  1
MSE Training =  0.39468326296998496
MSE Validation =  0.5365366613251107

Fold  2
MSE Training =  0.39468326296998496
MSE Validation =  0.5400313152544152

Fold  3
MSE Training =  0.39468326296998496
MSE Validation =  0.281147510383257

Fold  4
MSE Training =  0.39468326296998496
MSE Validation =  0.1970841106971943

Average Performance in Training =  0.39468326296998496
Average Performance in Validation =  0.38869989941499433
```

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

Fold  1
MSE Training =  0.3885569295943879
MSE Validation =  0.4658385778584045

Fold  2
MSE Training =  0.3885569295943879
MSE Validation =  0.4923545590365271

Fold  3
MSE Training =  0.3885569295943879
MSE Validation =  0.3153699717190666

Fold  4
MSE Training =  0.3885569295943879
MSE Validation =  0.265574669954707

Average Performance in Training =  0.3885569295943879
Average Performance in Validation =  0.3847844446421763
```

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

Fold  1
MSE Training =  0.4050524527858103
MSE Validation =  0.32721501899323907

Fold  2
```

```
MSE Training =  0.4050524527858103
MSE Validation =  0.5003820369109471

Fold  3
MSE Training =  0.4050524527858103
MSE Validation =  0.5011444569151555

Fold  4
MSE Training =  0.4050524527858103
MSE Validation =  0.2900106191295193

Average Performance in Training =  0.4050524527858103
Average Performance in Validation =  0.40468803298721523
```

```
M Value =  2
Sigma Value =  0.7
Mus= [0.1 0.2 0.3 0.4]

Fold  1
MSE Training =  0.4020709660342217
MSE Validation =  0.34071458059035326

Fold  2
MSE Training =  0.4020709660342217
MSE Validation =  0.46404665351593494

Fold  3
MSE Training =  0.4020709660342217
MSE Validation =  0.45838367373678396

Fold  4
MSE Training =  0.4020709660342217
MSE Validation =  0.34508734779066125

Average Performance in Training =  0.4020709660342217
Average Performance in Validation =  0.40205806390843335
```

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

Fold  1
MSE Training =  0.39955564245950903
MSE Validation =  0.2035214372427251

Fold  2
MSE Training =  0.39955564245950903
MSE Validation =  0.40257958439954045

Fold  3
MSE Training =  0.39955564245950903
MSE Validation =  0.26402553506310905

Fold  4
MSE Training =  0.39955564245950903
MSE Validation =  0.7441802017390574

Average Performance in Training =  0.39955564245950903
Average Performance in Validation =  0.40357668961110804
```

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

Fold  1
MSE Training =  0.3975251603289381
MSE Validation =  0.5211072524205079

Fold  2
MSE Training =  0.3975251603289381
MSE Validation =  0.2762674753112957

Fold  3
MSE Training =  0.3975251603289381
MSE Validation =  0.3264762720192248

Fold  4
MSE Training =  0.3975251603289381
MSE Validation =  0.46605594097523

Average Performance in Training =  0.3975251603289381
Average Performance in Validation =  0.3974767351815646
```

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

Fold  1
MSE Training =  0.2513620423457392
MSE Validation =  0.3178049758000619

Fold  2
MSE Training =  0.2513620423457392
MSE Validation =  0.2638622220770053

Fold  3
MSE Training =  0.2513620423457392
MSE Validation =  0.20329318165497204

Fold  4
MSE Training =  0.2513620423457392
MSE Validation =  0.21390919708545167

Average Performance in Training =  0.2513620423457392
Average Performance in Validation =  0.24971739415437272
```

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

Fold  1
MSE Training =  0.2744483331459012
MSE Validation =  0.4086493629866249

Fold  2
MSE Training =  0.2744483331459012
MSE Validation =  0.24401435526824142

Fold  3
MSE Training =  0.2744483331459012
```

```
MSE Validation =  0.13250606905244072

Fold  4
MSE Training =  0.2744483331459012
MSE Validation =  0.3039762909460425

Average Performance in Training =  0.2744483331459012
Average Performance in Validation =  0.2722865195633374
```

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

Fold  1
MSE Training =  0.300698176869777
MSE Validation =  0.2506555462490173

Fold  2
MSE Training =  0.300698176869777
MSE Validation =  0.2456385690224408

Fold  3
MSE Training =  0.300698176869777
MSE Validation =  0.2375209150818457

Fold  4
MSE Training =  0.300698176869777
MSE Validation =  0.4777361969981455

Average Performance in Training =  0.300698176869777
Average Performance in Validation =  0.3028878068378623
```

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

Fold  1
MSE Training =  0.3230953937397377
MSE Validation =  0.32435071103570934

Fold  2
MSE Training =  0.3230953937397377
MSE Validation =  0.45793440376080474

Fold  3
MSE Training =  0.3230953937397377
MSE Validation =  0.3429978839645587

Fold  4
MSE Training =  0.3230953937397377
MSE Validation =  0.15575738225479127

Average Performance in Training =  0.3230953937397377
Average Performance in Validation =  0.32026009525396604
```

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

Fold  1
```

```
MSE Training =  0.32132277942437204
MSE Validation =  0.6018757172498671

Fold  2
MSE Training =  0.32132277942437204
MSE Validation =  0.3001592357798044

Fold  3
MSE Training =  0.32132277942437204
MSE Validation =  0.12644120365051

Fold  4
MSE Training =  0.32132277942437204
MSE Validation =  0.2351991781688959

Average Performance in Training =  0.32132277942437204
Average Performance in Validation =  0.31591883371226936
```

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

Fold  1
MSE Training =  0.3370671701905416
MSE Validation =  0.327233819009897

Fold  2
MSE Training =  0.3370671701905416
MSE Validation =  0.34230291644441213

Fold  3
MSE Training =  0.3370671701905416
MSE Validation =  0.29603288752204526

Fold  4
MSE Training =  0.3370671701905416
MSE Validation =  0.38308219152970974

Average Performance in Training =  0.3370671701905416
Average Performance in Validation =  0.33716295362651605
```

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

Fold  1
MSE Training =  0.35050942309734495
MSE Validation =  0.23133645534135058

Fold  2
MSE Training =  0.35050942309734495
MSE Validation =  0.20908516957772877

Fold  3
MSE Training =  0.35050942309734495
MSE Validation =  0.5431653406299418

Fold  4
MSE Training =  0.35050942309734495
MSE Validation =  0.44016716194665956
```

```
Average Performance in Training =  0.35050942309734495
Average Performance in Validation =  0.35593853187392016
```

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

Fold  1
MSE Training =  0.3605394255911446
MSE Validation =  0.43646596825445394

Fold  2
MSE Training =  0.3605394255911446
MSE Validation =  0.3329779436072512

Fold  3
MSE Training =  0.3605394255911446
MSE Validation =  0.3392418036792499

Fold  4
MSE Training =  0.3605394255911446
MSE Validation =  0.3294415651003388

Average Performance in Training =  0.3605394255911446
Average Performance in Validation =  0.3595318201603235
```

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

Fold  1
MSE Training =  0.3574005757644304
MSE Validation =  0.21359209234292273

Fold  2
MSE Training =  0.3574005757644304
MSE Validation =  0.39635437931806067

Fold  3
MSE Training =  0.3574005757644304
MSE Validation =  0.3084293122039557

Fold  4
MSE Training =  0.3574005757644304
MSE Validation =  0.5199644091817724

Average Performance in Training =  0.3574005757644304
Average Performance in Validation =  0.3595850482616779
```

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

Fold  1
MSE Training =  0.36572106668352544
MSE Validation =  0.21983371234474283

Fold  2
MSE Training =  0.36572106668352544
```

```
MSE Validation =  0.34778617413267654

Fold  3
MSE Training =  0.36572106668352544
MSE Validation =  0.4897864900774335

Fold  4
MSE Training =  0.36572106668352544
MSE Validation =  0.4191297440867183

Average Performance in Training =  0.36572106668352544
Average Performance in Validation =  0.3691340301603928
```

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

Fold  1
MSE Training =  0.3725359540394524
MSE Validation =  0.39232774363478434

Fold  2
MSE Training =  0.3725359540394524
MSE Validation =  0.33159609295060943

Fold  3
MSE Training =  0.3725359540394524
MSE Validation =  0.29636828691718703

Fold  4
MSE Training =  0.3725359540394524
MSE Validation =  0.4716140319463548

Average Performance in Training =  0.3725359540394524
Average Performance in Validation =  0.3729765388622339
```

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

Fold  1
MSE Training =  0.37761639192911395
MSE Validation =  0.39139124455610985

Fold  2
MSE Training =  0.37761639192911395
MSE Validation =  0.6119078880806639

Fold  3
MSE Training =  0.37761639192911395
MSE Validation =  0.30640598719943674

Fold  4
MSE Training =  0.37761639192911395
MSE Validation =  0.18008825214869995

Average Performance in Training =  0.37761639192911395
Average Performance in Validation =  0.3724483429962276
```

```
M Value =  3
```

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

Fold  1
MSE Training =  0.3848616529806377
MSE Validation =  0.4483310246106109

Fold  2
MSE Training =  0.3848616529806377
MSE Validation =  0.6001679585229392

Fold  3
MSE Training =  0.3848616529806377
MSE Validation =  0.1419982250467938

Fold  4
MSE Training =  0.3848616529806377
MSE Validation =  0.3257180973111837

Average Performance in Training =  0.3848616529806377
Average Performance in Validation =  0.3790538263728819
```

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

Fold  1
MSE Training =  0.3877389075252378
MSE Validation =  0.2751340453814083

Fold  2
MSE Training =  0.3877389075252378
MSE Validation =  0.31516481475669905

Fold  3
MSE Training =  0.3877389075252378
MSE Validation =  0.45849720646304687

Fold  4
MSE Training =  0.3877389075252378
MSE Validation =  0.517591143075828

Average Performance in Training =  0.3877389075252378
Average Performance in Validation =  0.39159680241924555
```

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

Fold  1
MSE Training =  0.3901274776169607
MSE Validation =  0.4427876242473803

Fold  2
MSE Training =  0.3901274776169607
MSE Validation =  0.19231466769037853

Fold  3
MSE Training =  0.3901274776169607
MSE Validation =  0.4371884042168152
```

```
Fold  4
MSE Training =  0.3901274776169607
MSE Validation =  0.5003152695879493

Average Performance in Training =  0.3901274776169607
Average Performance in Validation =  0.3931514914356308
```

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

Fold  1
MSE Training =  0.3920063118374788
MSE Validation =  0.20662735053382278

Fold  2
MSE Training =  0.3920063118374788
MSE Validation =  0.47181127375805815

Fold  3
MSE Training =  0.3920063118374788
MSE Validation =  0.33536543399125135

Fold  4
MSE Training =  0.3920063118374788
MSE Validation =  0.5630190223487056

Average Performance in Training =  0.3920063118374788
Average Performance in Validation =  0.39420577015795943
```

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

Fold  1
MSE Training =  0.2260156538822793
MSE Validation =  0.3880112676883215

Fold  2
MSE Training =  0.2260156538822793
MSE Validation =  0.17944051471384798

Fold  3
MSE Training =  0.2260156538822793
MSE Validation =  0.13953996255428283

Fold  4
MSE Training =  0.2260156538822793
MSE Validation =  0.18745249768619732

Average Performance in Training =  0.2260156538822793
Average Performance in Validation =  0.2236110606606624
```

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

Fold  1
MSE Training =  0.2154056315534069
```

```
MSE Validation =  0.20736465362005538

Fold  2
MSE Training =  0.2154056315534069
MSE Validation =  0.14969547662834432

Fold  3
MSE Training =  0.2154056315534069
MSE Validation =  0.21737086869969435

Fold  4
MSE Training =  0.2154056315534069
MSE Validation =  0.29333745500373465

Average Performance in Training =  0.2154056315534069
Average Performance in Validation =  0.21694211348795717
```

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

Fold  1
MSE Training =  0.20401181091156734
MSE Validation =  0.2542157842994251

Fold  2
MSE Training =  0.20401181091156734
MSE Validation =  0.21845980176781546

Fold  3
MSE Training =  0.20401181091156734
MSE Validation =  0.18118882733602704

Fold  4
MSE Training =  0.20401181091156734
MSE Validation =  0.15679516655599282

Average Performance in Training =  0.20401181091156734
Average Performance in Validation =  0.2026648949898151
```

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

Fold  1
MSE Training =  0.19467817929404987
MSE Validation =  0.10323354919386347

Fold  2
MSE Training =  0.19467817929404987
MSE Validation =  0.17354704688212438

Fold  3
MSE Training =  0.19467817929404987
MSE Validation =  0.3304747516233874

Fold  4
MSE Training =  0.19467817929404987
MSE Validation =  0.18083868301950012
```

```
Average Performance in Training =  0.19467817929404987
Average Performance in Validation =  0.19702350767971885
```

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

Fold  1
MSE Training =  0.2131120257025402
MSE Validation =  0.19495197026197275

Fold  2
MSE Training =  0.2131120257025402
MSE Validation =  0.20156198830354377

Fold  3
MSE Training =  0.2131120257025402
MSE Validation =  0.16961252050761244

Fold  4
MSE Training =  0.2131120257025402
MSE Validation =  0.2887974648069956

Average Performance in Training =  0.2131120257025402
Average Performance in Validation =  0.21373098597003115
```

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

Fold  1
MSE Training =  0.20505494395275672
MSE Validation =  0.2503637287275276

Fold  2
MSE Training =  0.20505494395275672
MSE Validation =  0.16979416086699034

Fold  3
MSE Training =  0.20505494395275672
MSE Validation =  0.1753668358730118

Fold  4
MSE Training =  0.20505494395275672
MSE Validation =  0.22385771686941358

Average Performance in Training =  0.20505494395275672
Average Performance in Validation =  0.20484561058423584
```

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

Fold  1
MSE Training =  0.19810331990368216
MSE Validation =  0.1437569127576879

Fold  2
MSE Training =  0.19810331990368216
MSE Validation =  0.2530454419164301
```

```
Fold  3
MSE Training =  0.19810331990368216
MSE Validation =  0.16571647537731285

Fold  4
MSE Training =  0.19810331990368216
MSE Validation =  0.2298448066577351

Average Performance in Training =  0.19810331990368216
Average Performance in Validation =  0.19809090917729147
```

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

Fold  1
MSE Training =  0.1927950130806129
MSE Validation =  0.3155963064615388

Fold  2
MSE Training =  0.1927950130806129
MSE Validation =  0.12214362117982133

Fold  3
MSE Training =  0.1927950130806129
MSE Validation =  0.13500100809841908

Fold  4
MSE Training =  0.1927950130806129
MSE Validation =  0.19409329145932783

Average Performance in Training =  0.1927950130806129
Average Performance in Validation =  0.19170855679977677
```

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

Fold  1
MSE Training =  0.20455378636276453
MSE Validation =  0.2402229199548594

Fold  2
MSE Training =  0.20455378636276453
MSE Validation =  0.13424330771545426

Fold  3
MSE Training =  0.20455378636276453
MSE Validation =  0.2742178976452491

Fold  4
MSE Training =  0.20455378636276453
MSE Validation =  0.17241779889009665

Average Performance in Training =  0.20455378636276453
Average Performance in Validation =  0.20527548105141488
```

```
M Value =  4
Sigma Value =  0.5
```

```
Mus= [0.1 0.2 0.3 0.4]

Fold  1
MSE Training =  0.19974439068286387
MSE Validation =  0.21017729382403114

Fold  2
MSE Training =  0.19974439068286387
MSE Validation =  0.244830477614444

Fold  3
MSE Training =  0.19974439068286387
MSE Validation =  0.15384730353785453

Fold  4
MSE Training =  0.19974439068286387
MSE Validation =  0.18549590524906345

Average Performance in Training =  0.19974439068286387
Average Performance in Validation =  0.1985877450563483
```

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

Fold  1
MSE Training =  0.19568207089834805
MSE Validation =  0.1384295465606406

Fold  2
MSE Training =  0.19568207089834805
MSE Validation =  0.28444277931821177

Fold  3
MSE Training =  0.19568207089834805
MSE Validation =  0.2244100933608967

Fold  4
MSE Training =  0.19568207089834805
MSE Validation =  0.13282018234679674

Average Performance in Training =  0.19568207089834805
Average Performance in Validation =  0.19502565039663644
```

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

Fold  1
MSE Training =  0.19249174211221656
MSE Validation =  0.2804560924354131

Fold  2
MSE Training =  0.19249174211221656
MSE Validation =  0.14492863376951573

Fold  3
MSE Training =  0.19249174211221656
MSE Validation =  0.1272387214048941
```

```
Fold  4
MSE Training =  0.19249174211221656
MSE Validation =  0.2139767506740022

Average Performance in Training =  0.19249174211221656
Average Performance in Validation =  0.1916500495709563
```

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

Fold  1
MSE Training =  0.1978273408246996
MSE Validation =  0.2950778146933693

Fold  2
MSE Training =  0.1978273408246996
MSE Validation =  0.19179379725425755

Fold  3
MSE Training =  0.1978273408246996
MSE Validation =  0.1942707257602834

Fold  4
MSE Training =  0.1978273408246996
MSE Validation =  0.10256561473270254

Average Performance in Training =  0.1978273408246996
Average Performance in Validation =  0.1959269881101532
```

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

Fold  1
MSE Training =  0.195828545037609
MSE Validation =  0.19478099512943753

Fold  2
MSE Training =  0.195828545037609
MSE Validation =  0.1301238788443907

Fold  3
MSE Training =  0.195828545037609
MSE Validation =  0.2735038064577588

Fold  4
MSE Training =  0.195828545037609
MSE Validation =  0.19046818439396462

Average Performance in Training =  0.195828545037609
Average Performance in Validation =  0.1972192162063879
```

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

Fold  1
MSE Training =  0.194086704353046
MSE Validation =  0.12425178073207938
```

```
Fold  2
MSE Training =  0.194086704353046
MSE Validation =  0.1981561263280801

Fold  3
MSE Training =  0.194086704353046
MSE Validation =  0.19773979198864022

Fold  4
MSE Training =  0.194086704353046
MSE Validation =  0.26167957683387855

Average Performance in Training =  0.194086704353046
Average Performance in Validation =  0.19545681897066955
```

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

Fold  1
MSE Training =  0.19261515290155573
MSE Validation =  0.2701655176514721

Fold  2
MSE Training =  0.19261515290155573
MSE Validation =  0.10173490061932257

Fold  3
MSE Training =  0.19261515290155573
MSE Validation =  0.18882476518621863

Fold  4
MSE Training =  0.19261515290155573
MSE Validation =  0.21084625211023586

Average Performance in Training =  0.19261515290155573
Average Performance in Validation =  0.1928928588918123
```

In [ ]:

In [ ]: