Lab 2 - Modelling and Model Error

Introduction:

This Lab acts an introduction to the implementation and use of Models for predicting the behavior of a system. This includes all that is entailed in the successful implementation and use of a Model. The first step covered in this process is identifying the independent and dependent variables in the data so that the data can be correctly interpreted, visualized, and design decisions about the model can begin to be made. Next, the model is chosen and described so that an understanding can be reached on what the inputs and outputs are, and what parameters there are to adjust the output. Following this, the chosen model is converted into code so that it can accept the data set as input, and parameters are entered so that the resulting output is a prediction for the data set that is used. Following this, the model's output is visualized and compared to the expected output of the data set so that at a glance the effectiveness of the predictive ability of the model can be understood. Lastly, the error is calculated between the model's output and the data set so that a quantitative value can be convey the effectiveness of the model.

Author: Nigel Nelson

Questions:

1. *Describe the relationship between the number of independent variables, dependent variables, and model parameters. If there is no dependence between them, please state this and discuss why this might be or what implications it has.*

   In terms of the Gaussian model, there is a linear correlation between the number of independent variables, dependent variables, and model parameters. The reason for this is that for the Gaussian formula, if the independent variable is a column vector, then there will be sigma and mu as parameters, both as scalars, and the dependent variable will be a column vector the same size as the independent variable. However, if the independent variables increases to be an mxn matrix, then the sigma parameter will remain a scalar, but mu will increase in size to be an mxn matrix, and the dependent variable will also grow to be an mxn matrix. As for the Linear Regression model, there is a similar relation between the number of independent variables, dependent variables, and model parameters. The difference is that if the independent variable is an mxn matrix in size, then the parameter epsilon will remain a scalar, but beta will grow in size to be an nxq matrix. Lastly, this results in the dependent variable growing in size to be an mxq matrix. To generalize this across models, it can be expected have two parameters, one scalar parameter that translates the resulting dependent variables up or down in value, and another parameter that will grow in relation to the shape of the input, that is responsible for scaling the independent variables. Lastly, there is the dependent variable that will commonly result in one classification or continuous value per observation in the independent variable. However, these are just common generalizations and

by no means the rule when it comes to models. Models take all forms and the way their variables and parameters interact vary greatly and that is why the model familiarity step is so important in this lab, but also anytime a new model is being used.

2. *For this lab you were given datasets that had paired independent and dependent variables (supervised data).*
   A. *What would your model do if you gave it an independent variable value not from this dataset?*

   It would calculate a dependent variable as output based on the independence variable and the model parameters that were chosen. Essentially, it would treat it as any other value from the dataset, however, the output may be useless as this input value could be outside the spectrum of values that the model is prepared to predict.
   B. *Do you think the resulting output would be correct?*

   It would depend on the input value in comparison to the range of values in the dataset. If the value was within the dataset's range of values it's likely the output would be close to correct, but if the value was outset the range used in the dataset, it's likely the model wouldn't know how to correctly handle a significantly smaller or larger value, and its output would be an inaccurate value.
   C. *How can you be sure?*

   With the model only serving the purpose of a predictor, it could not be relied upon for a ground truth. So the only way to tell for certain if the output is correct, is to either find labeled data for that input value, or test that input value in the real world.

3. *You used mean absolute error to quantify the difference between your given data and model predictions. Lookup (either online or from your textbook) another metric used to quantify error.*
   A. Compare and contrast this new metric with mean absolute error

   The other error metric found was MSE, or Mean Squared Error. MSE is similar to MAE in that the both act as a means to quantify and average the amount of error across all the observations. Both error metrics are directionless as well, in MAE's case its due to the fact that it takes the absolute value of the amount of error, but in MSE's case it squares this difference to get a directionless result. The last similarity is that both MSE and MAE are both negatively oriented scores, meaning that the lower the value the less error. The biggest difference between the two comes from the fact that MSE squares the difference in error between predicted and expected, this results in higher amounts of error being penalized even greater compared to MAE. In addition, MAE outputs values that are the same units as the input, whereas MSE does not have this intuitive benefit.
   B. *Discuss what you think the advantages/disadvantages might be between MAE and your other metric. Hint: What shape do different error functions have as you change model parameters to get predictions that are closer to your observed data?*

   Depending on the use case, an advantage of MSE could be that it penalizes outliers in the data greater than MAE. This is due to the fact the error is squared, resulting in differences in error being penalized exponentially higher as error increases, whereas MAE penalizes linearly as error increases. This could be a good feature to have if outliers are an especially undesirable trait for a given data set. An advantage of MAE is that it outputs a value in the same units as the input data. This is a benefit as it is much more intuitive to understand how much error a model is producing when a common unit is used.

4. *We had you plot multiple figures for experiment 2.*
   A. *Can you think of a way to plot 2 independent variables and the dependent variable on the same plot?*

   One way that this could be accomplished is by using a 3D plot. This would be done by plotting the two independent variables on the X and Y axis, and then the dependent variable on the Z axis.

   B. *What about a way to visualize 3 independent variables and the dependent variable on the same plot? 4 and 1?*

   A way to visualize 3 independent variables and one dependent variable would be to use a 3D plot with a color bar. This would be done by plotting the independent variables on the X, Y, and Z axis's, and then using a color bar to visualize the effects on the dependent variable. Adding another independent variable for a total of 4 could be accomplished by changing the size of the data point marker, with a key included in the plot to be able to convert sizes to values for the 4th independent variable.

   C. *What about a way to visualize a dataset of 100 features and 1 dependent variable on the same plot?*

   Theoretically it would be possible to visualize a dataset of 100 features to 1 dependent variable. This could be done by finding a unique way to change a data point marker for each independent variable. However, as more and more features are added to the plot, more and more noise is created and it becomes much more difficult to quickly understand the data and how it all relates together.

   D. *With these plots in mind, describe how the error metric can help.*

   An error metric can greatly improve interpretability of data when several features are involved. The reason for this is that with more features, it becomes increasingly difficult to convey how these individual features effect a model's dependent variable in comparison to the expected output. Introducing an error metric allows for a single quantitative measure to be used in order to understand the amount of error the model is producing compared to the expected output.

5. *How well did the model parameter sets you chose for experiment 2 perform? With the methods and tools that you have at your disposal, can you think of a more structured way to find a good set of parameters rather than picking a set and checking? Hint: It is relatively inexpensive to evaluate these models. How could you use a loop, or set of loops, to find a set of model parameters that have low error?*

   Overall, the model parameters that I selected for experiment 2 did not perform very well. Parameter set E and F had MAE scores of ~1.64 and ~2.94, respectively. This means that on average, these models were off on their sales value predictions by at least an entire sale unit. These parameters were selected by hand, by simply plugging in values and seeing what seemed to decrease the MAE, however, a more structured approach would likely produce better results. One way to go about this would be to use a series of loops. Where each independent variable would have its own loop where it would be determined if increasing or decreasing decreased the MAE, and from there would iterate the value in that direction until a decrease in MAE was detected. Each independent variable would go through its looping procedure, and then an all-encompassing loop would be used to iterate that series of independent variable loops until the benefit seen by the model fell below a given threshold

Experiment 1: Gaussian Distribution

Imports:

In [1]:
```
import numpy as np
import scipy
import scipy.stats as stats
import matplotlib.pyplot as plt
from sklearn.metrics import mean_absolute_error
import math
```

Part I - Data Familiarity

- **Importing the Gaussian Distribution Data:**

In [2]:
```
data = np.loadtxt(open("gaussdist.csv", "rb"), delimiter=",")
data[:5]
```

Out[2]:
```
array([[6.99000000e+00, 1.56842355e-01],
       [8.90000000e+00, 7.89692304e-03],
       [9.58000000e+00, 1.55088416e-03],
       [5.46000000e+00, 3.18990459e-01],
       [1.38000000e+00, 1.39635489e-03]])
```

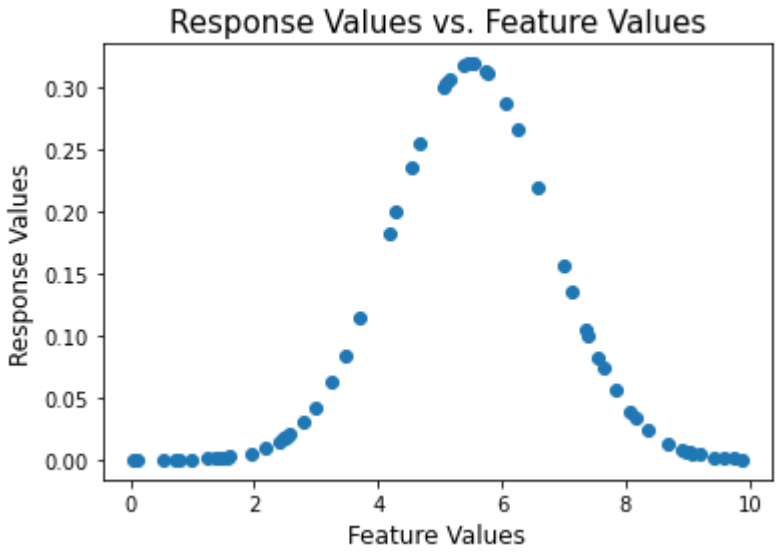
- **Identifying response variable and feature variable:**
  - Based from the Gaussian Distribution data that is loaded in the above cell, it is clear that the second column is not monotonic, which is a characteristic of the output for a Gaussian Distribution. As such, the first column is the feature variable values, and the second column is the response variable values.

In [3]:
```
feature_vars = data[:, 0]
response_vars = data[:, 1]
```

- **Plotting the Response Values vs. the Feature Values**

In [4]:
```
plt.scatter(feature_vars, response_vars)
plt.title("Response Values vs. Feature Values", fontsize=15)
plt.xlabel("Feature Values", fontsize=12)
plt.ylabel("Response Values", fontsize=12)
```

Out[4]: Text(0, 0.5, 'Response Values')



Part II - Model Familiarity

- For Experiment 1, the Gaussian Distribution function is model that is being used. In the space below this formula is written out mathematically, and key model components are identified.

  The Gaussian Distribution function:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

  - **Indepenent variable: x**
  - **Dependent variable: f(x)**
  - **Model parameters:** μ, σ

Part III - Model Implementation

- In the below cell, the Gaussian Distribution function is converted into a method so that model parameters can be tested, and the resulting dependent variables can be analyzed.

In [5]:
```
def gaussian_model(x, mu=1, sigma=1):
    left_side = 1 / (sigma*(math.sqrt(2  * math.pi)))
    right_side = math.e ** (-.5*((x - mu)/sigma)**2)
    return left_side * right_side
```

Part IV - Model Output and Visualization

- Now with the model implemented, sets of variables must be defined to explore the effect that different sets of parameters have on the resulting dependent variables. Found below are the sets of parameters that the model will run on.

	μ	σ
Parameter Set a	1	0.75
Parameter Set b	1	1.25
Parameter Set c	5	1.25
Parameter Set d	6	1.25

- **Running the model on the specified parameter sets**

In [6]:
```
param_a_response = gaussian_model(feature_vars, 1, 0.75)
param_b_response = gaussian_model(feature_vars, 1, 1.25)
param_c_response = gaussian_model(feature_vars, 5, 1.25)
param_d_response = gaussian_model(feature_vars, 6, 1.25)
```

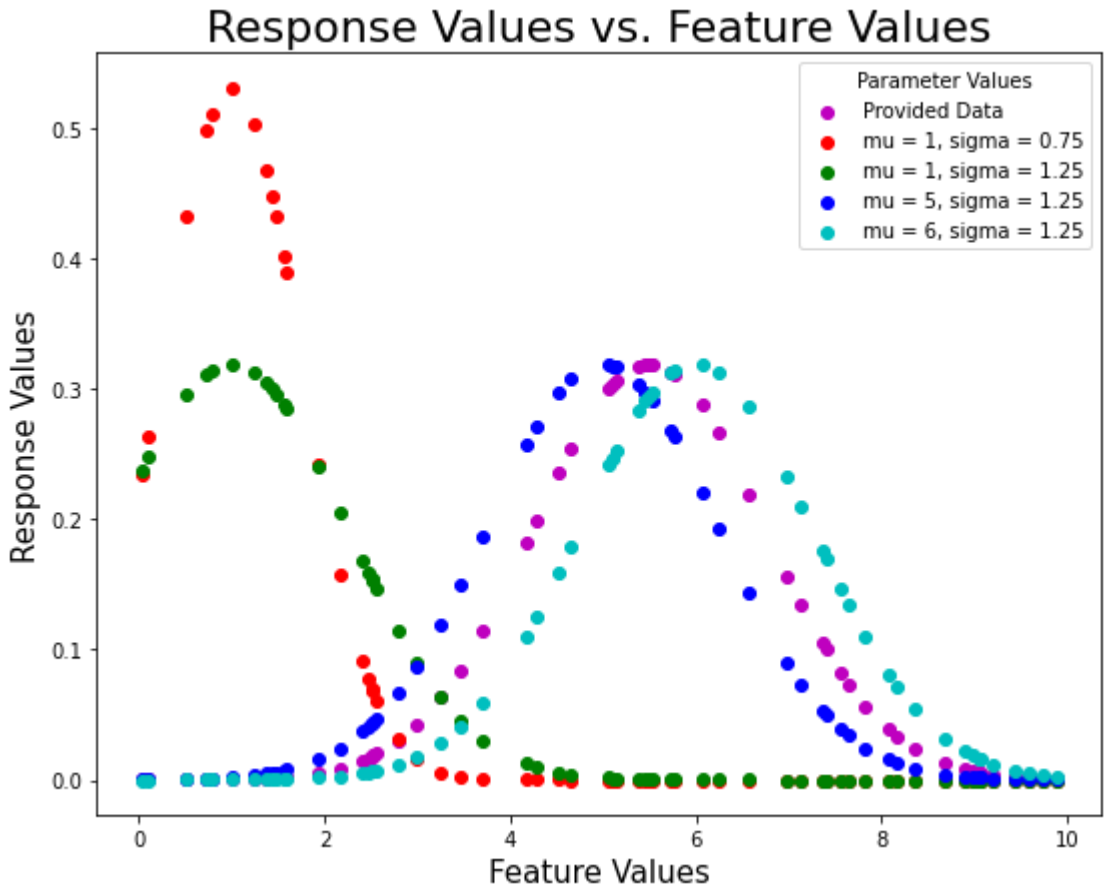
- **Plotting the model's response values and the given data vs. the feature values**

In [7]:
```
figure, axis = plt.subplots(figsize=(9,7))

axis.scatter(feature_vars, response_vars, label="Provided Data", c='m')
axis.scatter(feature_vars, param_a_response, label="mu = 1, sigma = 0.75", c=
axis.scatter(feature_vars, param_b_response, label="mu = 1, sigma = 1.25", c=
axis.scatter(feature_vars, param_c_response, label="mu = 5, sigma = 1.25", c=
axis.scatter(feature_vars, param_d_response, label="mu = 6, sigma = 1.25", c=

axis.set_xlabel("Feature Values", fontsize=15)
axis.set_ylabel("Response Values", fontsize=15)
axis.set_title("Response Values vs. Feature Values", fontsize=22)
plt.legend(title="Parameter Values")
```

Out[7]: <matplotlib.legend.Legend at 0x24dad318c10>



Part V – Error between Model and Data:

- In this section, the error of the model implemented above is quantified for each set of response values. The metric used to quantify this error is Mean Absolute Error, which is written out mathematically, and whose key variables are identified below.

$$MeanAbsoluteError = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

  - **Model Predictions:** y_i
  - **Dependent Variable(s) from dataset:** x_i

- **Calculating the Mean Absolute Error (MAE) for each set of response values:**

In [8]:
```
provided_set_mae = mean_absolute_error(response_vars, response_vars)
print("Provided set's MAE: " + str(provided_set_mae))
print()
set_a_mae = mean_absolute_error(param_a_response, response_vars)
print("Set A's MAE: " + str(set_a_mae))
set_b_mae = mean_absolute_error(param_b_response, response_vars)
print("Set B's MAE: " + str(set_b_mae))
set_c_mae = mean_absolute_error(param_c_response, response_vars)
print("Set C's MAE: " + str(set_c_mae))
set_d_mae = mean_absolute_error(param_d_response, response_vars)
print("Set D's MAE: " + str(set_d_mae))
```

```
Provided set's MAE: 0.0

Set A's MAE: 0.18937403173481707
Set B's MAE: 0.16922757281678635
Set C's MAE: 0.026490469979950228
Set D's MAE: 0.027470666663403025
```

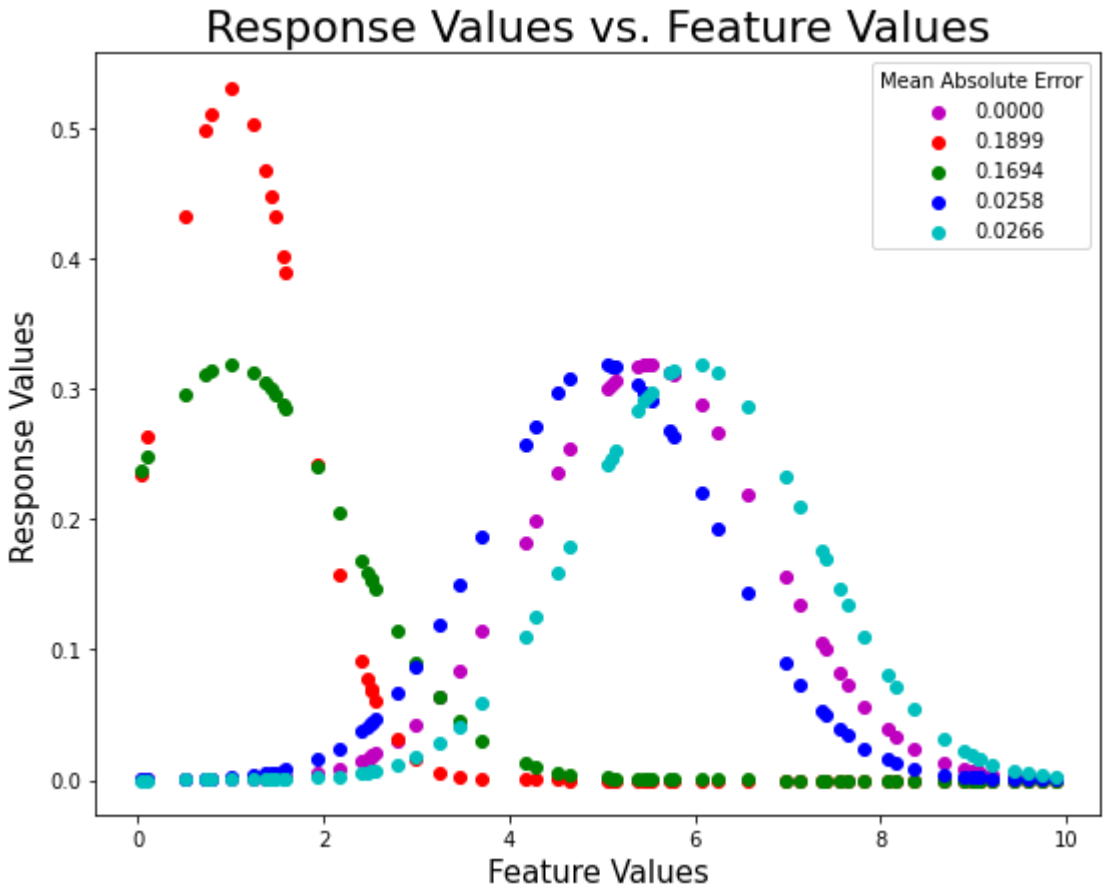
- **Plotting the model's response values and the given data vs. the feature values with MAE included in the figure legend:**

In [9]:
```
figure, axis = plt.subplots(figsize=(9,7))

axis.scatter(feature_vars, response_vars, label="0.0000", c='m')
axis.scatter(feature_vars, param_a_response, label="0.1899", c='r')
axis.scatter(feature_vars, param_b_response, label="0.1694", c='g')
axis.scatter(feature_vars, param_c_response, label="0.0258", c='b')
axis.scatter(feature_vars, param_d_response, label="0.0266", c='c')

axis.set_xlabel("Feature Values", fontsize=15)
axis.set_ylabel("Response Values", fontsize=15)
axis.set_title("Response Values vs. Feature Values", fontsize=22)
plt.legend(title="Mean Absolute Error")
```

Out[9]: <matplotlib.legend.Legend at 0x24dad4deb20>



Experiment 2: Multiple Linear Regression Model

Part I - Data Familiarity

- **Importing the Advertising Data:**

In [10]:
```
data_E2 = np.loadtxt(open("advertising.csv", "rb"), delimiter=",", skiprows=1
data_column_labels = np.loadtxt(open("advertising.csv", "rb"), delimiter=",",
print("Advertising data shape: " + str(data_E2.shape))
print("Advertising data column labels: " + str(data_column_labels))
```

```
Advertising data shape: (200, 6)
Advertising data column labels: [b'' b'Offset' b'TV' b'radio' b'newspaper'
b'sales']
```

- **Identifying response variable and feature variables:**
  - After viewing the labels for the Advertising data set, it is clear that the feature variables are the second, third, fourth, and fith columns, and the response variable is the final 'sales' column.

In [11]:
```
features = data_E2[:, 1:5]
offset_values = data_E2[:, 1]
tv_values = data_E2[:, 2]
radio_values = data_E2[:, 3]
newspaper_values = data_E2[:, 4]
sale_values = data_E2[:, 5]
```

- **Plotting the Response variable vs. the Feature variables**

In [12]:
```
figure, axis = plt.subplots(2,2,figsize=(9,7))

axis[0, 0].scatter(offset_values, sale_values)
axis[0, 0].set_xlabel("Offset Values", fontsize=11)
axis[0, 0].set_ylabel("Sales Values", fontsize=11)
axis[0, 0].set_title("Sale Values vs. Offset Values", fontsize=14)

axis[0, 1].scatter(tv_values, sale_values)
axis[0, 1].set_xlabel("TV Values", fontsize=11)
axis[0, 1].set_ylabel("Sales Values", fontsize=11)
axis[0, 1].set_title("Sale Values vs. TV Values", fontsize=14)

axis[1, 0].scatter(radio_values, sale_values)
axis[1, 0].set_xlabel("Radio Values", fontsize=11)
axis[1, 0].set_ylabel("Sales Values", fontsize=11)
axis[1, 0].set_title("Sale Values vs. Radio Values", fontsize=14)

axis[1, 1].scatter(newspaper_values, sale_values)
axis[1, 1].set_xlabel("Newspaper Values", fontsize=11)
axis[1, 1].set_ylabel("Sales Values", fontsize=11)
axis[1, 1].set_title("Sale Values vs. Newspaper Values", fontsize=14)
figure.tight_layout()
```

Part II - Model Familiarity

- For Experiment 2, the Multiple Linear Regression Function is model that is being used. In the space below this formula is written out mathematically, and key model components are identified.

  The Linear Regression function:

  $$\mathbf{y} = X\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

  - **Indepenent variable: X**
  - **Dependent variable: y**
  - **Model parameters:** β, ε
- To clarify, X will be an nxm matrix, where n is the number of observations in the data set, and m is the number of features, which in this data set is 4, Offset, Tv, radio, and newspaper.

Part III - Model Implementation

- In the below cell, the Multiple Linear Regression Function is converted into a method so that model parameters can be tested, and the resulting dependent variables can be analyzed.

In [13]:
```
def linear_regression(x, beta, epsilon):
    return np.dot(x, beta) + epsilon
```

Part IV - Model Output and Visualization

- Now with the model implemented, sets of variables must be defined to explore the effect that different sets of parameters have on the resulting dependent variables. Found below are the sets of parameters that the model will run on.

	β	ε
Parameter Set e	[0.01 , 0.05, 0.1, 0.04]	4

	β	ε
Parameter Set f	[0.05, 0.06, 0.3, 0.05]	-2

- **Running the model on the specified parameter sets**

In [14]:
```
param_e_response = linear_regression(features, np.array([0.01, .05, .1,.04]),
param_f_response = linear_regression(features, np.array([.05, .06, .3, .05]),
```

- **Plotting Parameter Set e's response values and the given data vs. the feature values**

In [15]:

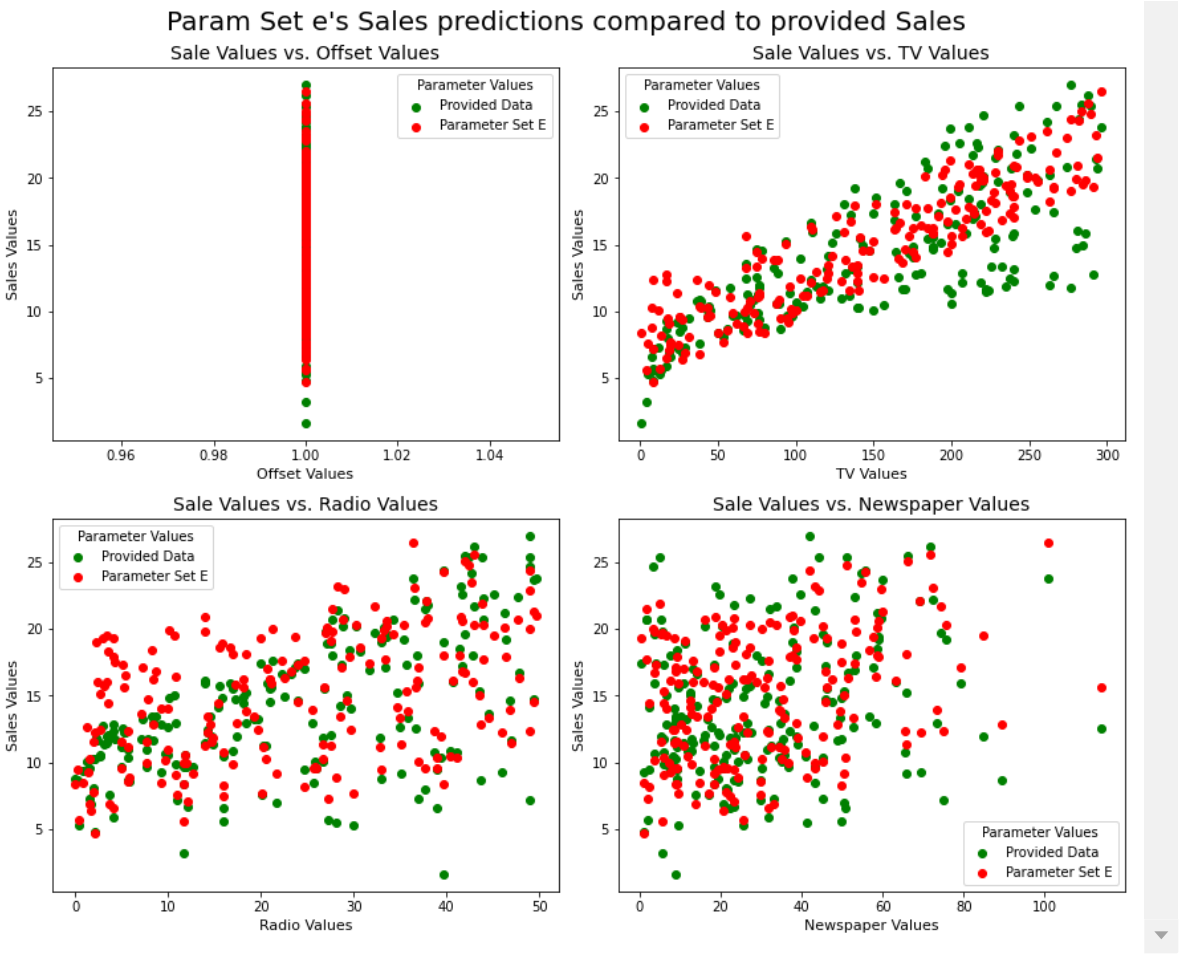
```
figure, axis = plt.subplots(2,2,figsize=(12,10))

axis[0, 0].scatter(offset_values, sale_values, label="Provided Data", c='g')
axis[0, 0].scatter(offset_values, param_e_response, label="Parameter Set E",
axis[0, 0].set_xlabel("Offset Values", fontsize=11)
axis[0, 0].set_ylabel("Sales Values", fontsize=11)
axis[0, 0].set_title("Sale Values vs. Offset Values", fontsize=14)
axis[0, 0].legend(title="Parameter Values")

axis[0, 1].scatter(tv_values, sale_values, label="Provided Data", c='g')
axis[0, 1].scatter(tv_values, param_e_response, label="Parameter Set E", c='r
axis[0, 1].set_xlabel("TV Values", fontsize=11)
axis[0, 1].set_ylabel("Sales Values", fontsize=11)
axis[0, 1].set_title("Sale Values vs. TV Values", fontsize=14)
axis[0, 1].legend(title="Parameter Values")

axis[1, 0].scatter(radio_values, sale_values, label="Provided Data", c='g')
axis[1, 0].scatter(radio_values, param_e_response, label="Parameter Set E", c
axis[1, 0].set_xlabel("Radio Values", fontsize=11)
axis[1, 0].set_ylabel("Sales Values", fontsize=11)
axis[1, 0].set_title("Sale Values vs. Radio Values", fontsize=14)
axis[1, 0].legend(title="Parameter Values")

axis[1, 1].scatter(newspaper_values, sale_values, label="Provided Data", c='g
axis[1, 1].scatter(newspaper_values, param_e_response, label="Parameter Set E
axis[1, 1].set_xlabel("Newspaper Values", fontsize=11)
axis[1, 1].set_ylabel("Sales Values", fontsize=11)
axis[1, 1].set_title("Sale Values vs. Newspaper Values", fontsize=14)
axis[1, 1].legend(title="Parameter Values")
figure.suptitle("Param Set e's Sales predictions compared to provided Sales",
figure.tight_layout()
```



- **Plotting Parameter Set f's response values and the given data vs. the feature values**

In [16]:

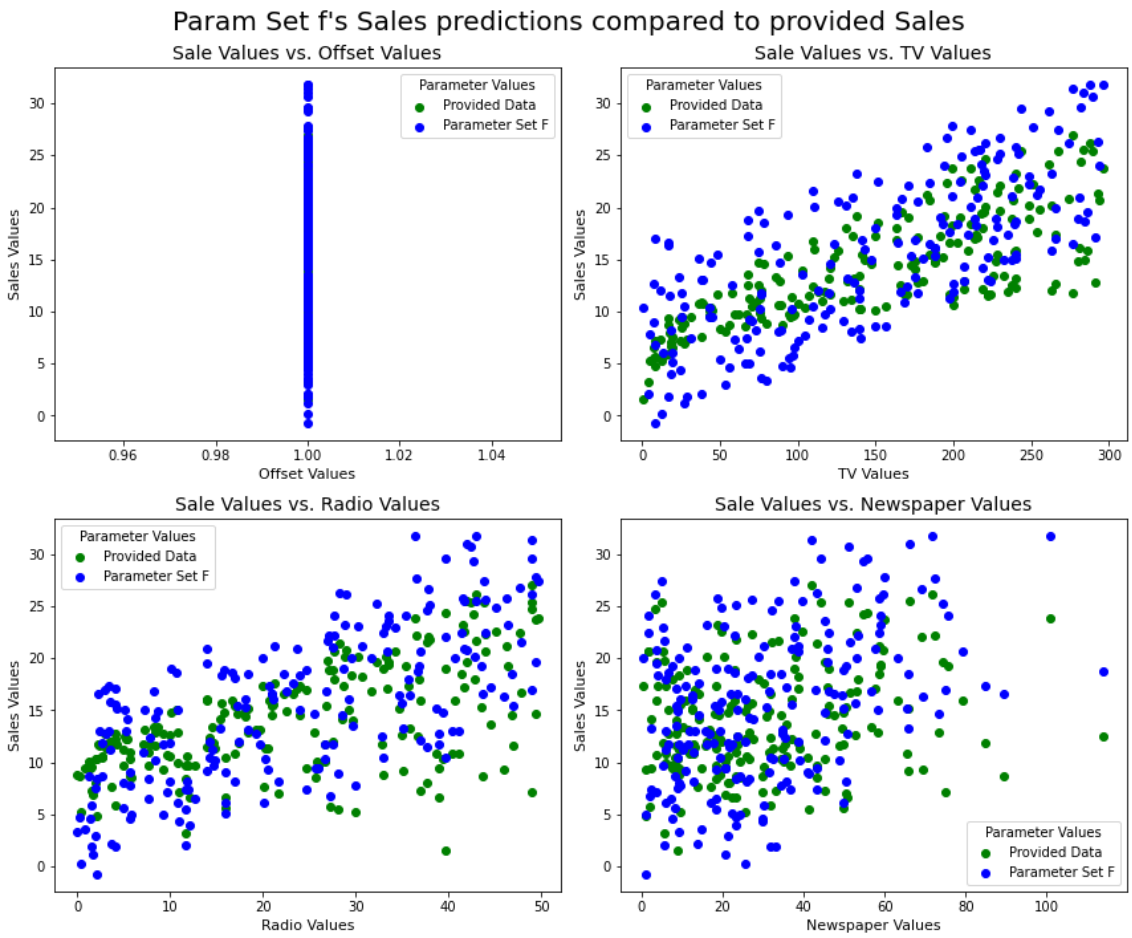
```
figure, axis = plt.subplots(2,2,figsize=(12,10))

axis[0, 0].scatter(offset_values, sale_values, label="Provided Data", c='g')
axis[0, 0].scatter(offset_values, param_f_response, label="Parameter Set F",
axis[0, 0].set_xlabel("Offset Values", fontsize=11)
axis[0, 0].set_ylabel("Sales Values", fontsize=11)
axis[0, 0].set_title("Sale Values vs. Offset Values", fontsize=14)
axis[0, 0].legend(title="Parameter Values")

axis[0, 1].scatter(tv_values, sale_values, label="Provided Data", c='g')
axis[0, 1].scatter(tv_values, param_f_response, label="Parameter Set F", c='b
axis[0, 1].set_xlabel("TV Values", fontsize=11)
axis[0, 1].set_ylabel("Sales Values", fontsize=11)
axis[0, 1].set_title("Sale Values vs. TV Values", fontsize=14)
axis[0, 1].legend(title="Parameter Values")

axis[1, 0].scatter(radio_values, sale_values, label="Provided Data", c='g')
axis[1, 0].scatter(radio_values, param_f_response, label="Parameter Set F", c
axis[1, 0].set_xlabel("Radio Values", fontsize=11)
axis[1, 0].set_ylabel("Sales Values", fontsize=11)
axis[1, 0].set_title("Sale Values vs. Radio Values", fontsize=14)
axis[1, 0].legend(title="Parameter Values")

axis[1, 1].scatter(newspaper_values, sale_values, label="Provided Data", c='g
axis[1, 1].scatter(newspaper_values, param_f_response, label="Parameter Set F
axis[1, 1].set_xlabel("Newspaper Values", fontsize=11)
axis[1, 1].set_ylabel("Sales Values", fontsize=11)
axis[1, 1].set_title("Sale Values vs. Newspaper Values", fontsize=14)
axis[1, 1].legend(title="Parameter Values")
figure.suptitle("Param Set f's Sales predictions compared to provided Sales",
figure.tight_layout()
```



Part V – Error between Model and Data:

- In this section, the error of the model implemented above is quantified for each set of response values. The metric used to quantify this error is Mean Absolute Error, which is written out mathematically, and whose key variables are identified below.

$$MeanAbsoluteError = \frac{\sum_{i=1}^{n} |e_i|}{n}$$

  - **Model Predictions:** y_i
  - **Dependent Variable(s) from dataset:** x_i

- **Calculating the Mean Absolute Error (MAE) for each set of response values:**

In [17]:
```
provided_set_mae = mean_absolute_error(response_vars, response_vars)
print("Provided set's MAE: " + str(provided_set_mae))
print()
set_a_mae = mean_absolute_error(param_e_response, sale_values)
print("Set E's MAE: " + str(set_a_mae))
set_b_mae = mean_absolute_error(param_f_response, sale_values)
print("Set F's MAE: " + str(set_b_mae))
```

```
Provided set's MAE: 0.0

Set E's MAE: 1.6408150000000001
Set F's MAE: 2.9387100000000004
```

- **Plotting Parameter Set e's response values and the given data vs. the feature values with MAE included in the figure legend:**

In [18]:

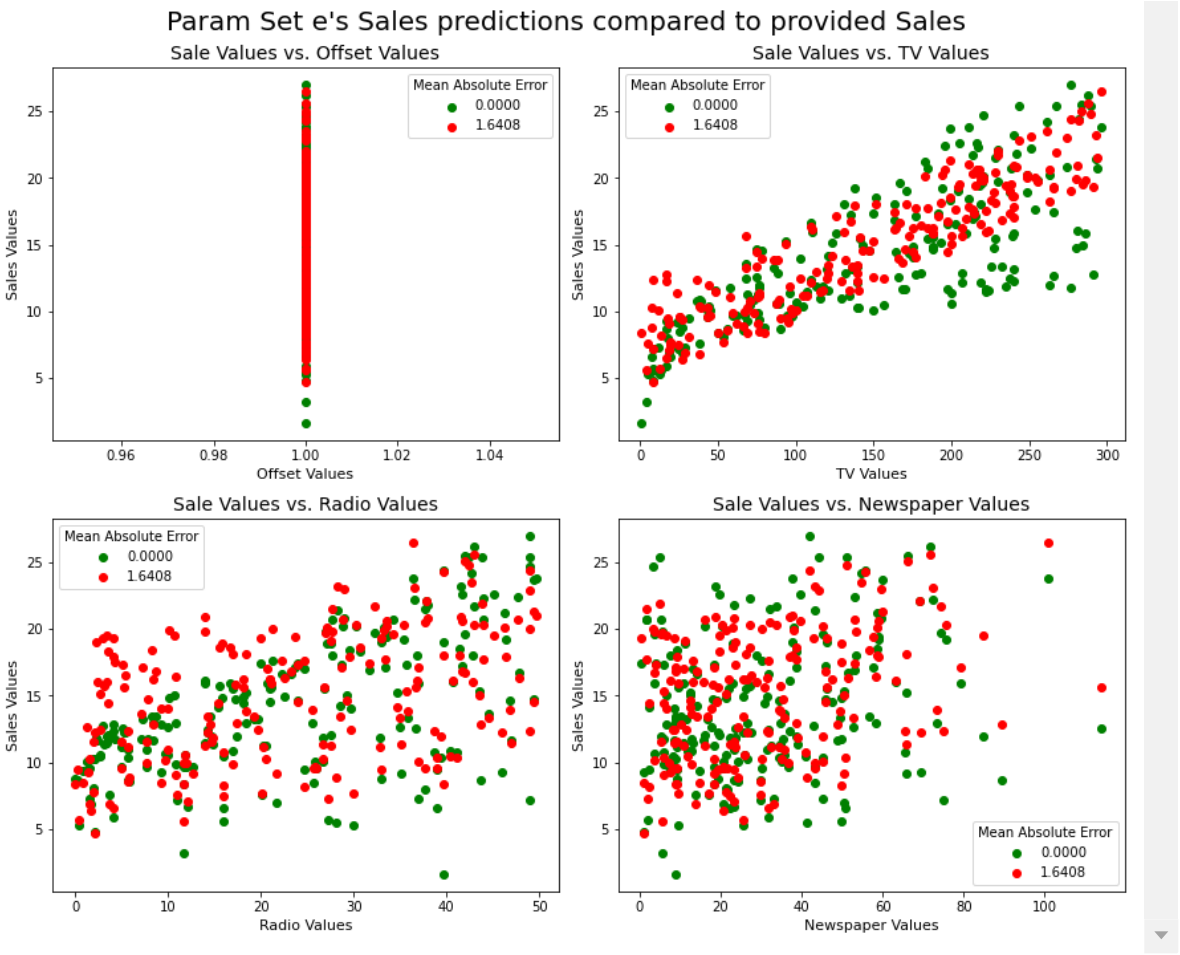
```
figure, axis = plt.subplots(2,2,figsize=(12,10))

axis[0, 0].scatter(offset_values, sale_values, label="0.0000", c='g')
axis[0, 0].scatter(offset_values, param_e_response, label="1.6408", c='r')
axis[0, 0].set_xlabel("Offset Values", fontsize=11)
axis[0, 0].set_ylabel("Sales Values", fontsize=11)
axis[0, 0].set_title("Sale Values vs. Offset Values", fontsize=14)
axis[0, 0].legend(title="Mean Absolute Error")

axis[0, 1].scatter(tv_values, sale_values, label="0.0000", c='g')
axis[0, 1].scatter(tv_values, param_e_response, label="1.6408", c='r')
axis[0, 1].set_xlabel("TV Values", fontsize=11)
axis[0, 1].set_ylabel("Sales Values", fontsize=11)
axis[0, 1].set_title("Sale Values vs. TV Values", fontsize=14)
axis[0, 1].legend(title="Mean Absolute Error")

axis[1, 0].scatter(radio_values, sale_values, label="0.0000", c='g')
axis[1, 0].scatter(radio_values, param_e_response, label="1.6408", c='r')
axis[1, 0].set_xlabel("Radio Values", fontsize=11)
axis[1, 0].set_ylabel("Sales Values", fontsize=11)
axis[1, 0].set_title("Sale Values vs. Radio Values", fontsize=14)
axis[1, 0].legend(title="Mean Absolute Error")

axis[1, 1].scatter(newspaper_values, sale_values, label="0.0000", c='g')
axis[1, 1].scatter(newspaper_values, param_e_response, label="1.6408", c='r')
axis[1, 1].set_xlabel("Newspaper Values", fontsize=11)
axis[1, 1].set_ylabel("Sales Values", fontsize=11)
axis[1, 1].set_title("Sale Values vs. Newspaper Values", fontsize=14)
axis[1, 1].legend(title="Mean Absolute Error")
figure.suptitle("Param Set e's Sales predictions compared to provided Sales",
figure.tight_layout()
```



- **Plotting Parameter Set f's response values and the given data vs. the feature values with MAE included in the figure legend:**

In [19]:

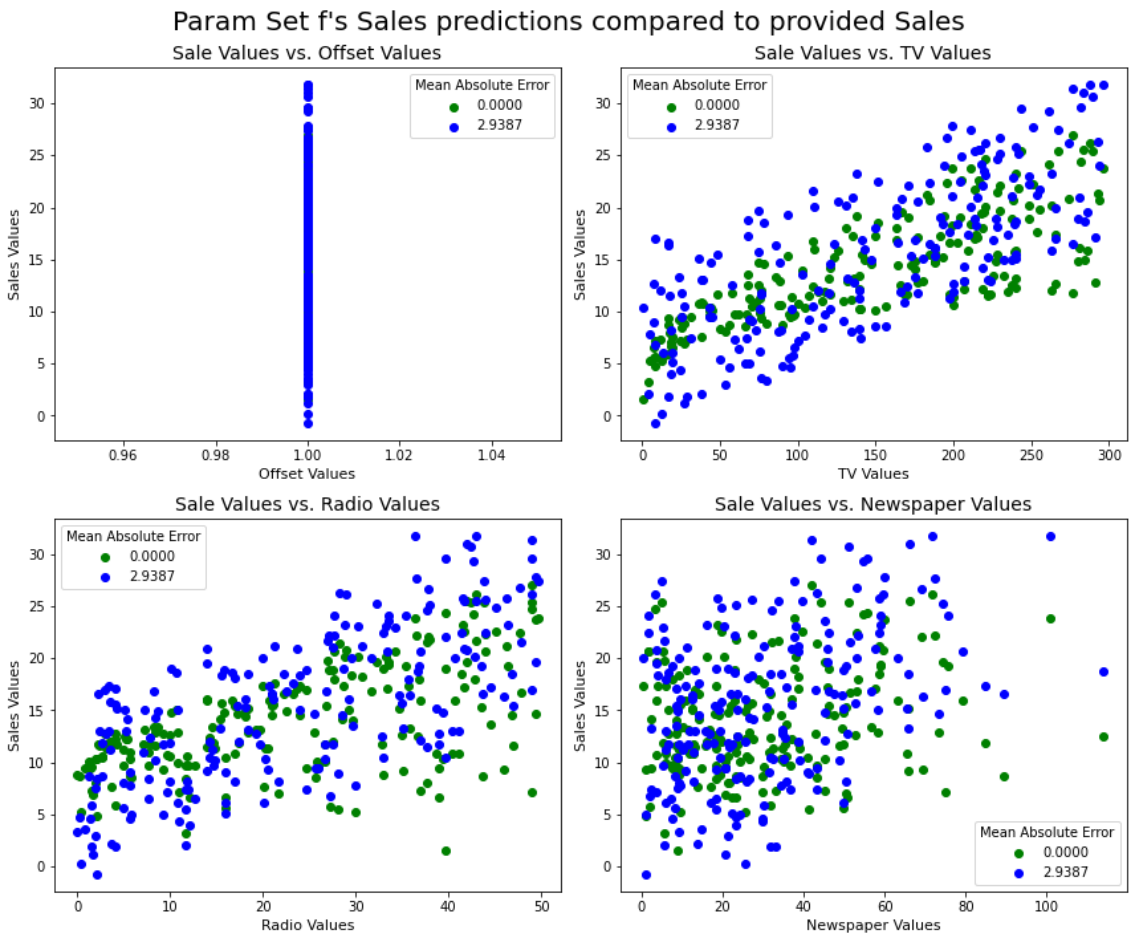
```
figure, axis = plt.subplots(2,2,figsize=(12,10))

axis[0, 0].scatter(offset_values, sale_values, label="0.0000", c='g')
axis[0, 0].scatter(offset_values, param_f_response, label="2.9387", c='b')
axis[0, 0].set_xlabel("Offset Values", fontsize=11)
axis[0, 0].set_ylabel("Sales Values", fontsize=11)
axis[0, 0].set_title("Sale Values vs. Offset Values", fontsize=14)
axis[0, 0].legend(title="Mean Absolute Error")

axis[0, 1].scatter(tv_values, sale_values, label="0.0000", c='g')
axis[0, 1].scatter(tv_values, param_f_response, label="2.9387", c='b')
axis[0, 1].set_xlabel("TV Values", fontsize=11)
axis[0, 1].set_ylabel("Sales Values", fontsize=11)
axis[0, 1].set_title("Sale Values vs. TV Values", fontsize=14)
axis[0, 1].legend(title="Mean Absolute Error")

axis[1, 0].scatter(radio_values, sale_values, label="0.0000", c='g')
axis[1, 0].scatter(radio_values, param_f_response, label="2.9387", c='b')
axis[1, 0].set_xlabel("Radio Values", fontsize=11)
axis[1, 0].set_ylabel("Sales Values", fontsize=11)
axis[1, 0].set_title("Sale Values vs. Radio Values", fontsize=14)
axis[1, 0].legend(title="Mean Absolute Error")

axis[1, 1].scatter(newspaper_values, sale_values, label="0.0000", c='g')
axis[1, 1].scatter(newspaper_values, param_f_response, label="2.9387", c='b')
axis[1, 1].set_xlabel("Newspaper Values", fontsize=11)
axis[1, 1].set_ylabel("Sales Values", fontsize=11)
axis[1, 1].set_title("Sale Values vs. Newspaper Values", fontsize=14)
axis[1, 1].legend(title="Mean Absolute Error")
figure.suptitle("Param Set f's Sales predictions compared to provided Sales",
figure.tight_layout()
```



Conclusion:

This Lab acted an introduction to the implementation and use of Models for predicting the behavior of a system. This included all that is entailed in the successful implementation and use of a Model. In two different experiments, the variables of different data sets were identified, a model was chosen and analyzed, this model was then run on various parameters, and finally the models outputs were visualized and analyzed for error. Through this process best practices for implementing a model were learned, and several other lessons were discovered as well. This includes the fact that altering model parameters by hand to decrease the error is a tedious process that is best left to automation. In addition, the importance of readability for future labs was engrained thanks to the number of steps required to go from importing data to visualizing a model's predictions based on that data. Lastly, the proper way to use and discuss error metrics was learned, making it much easier to understand the shortfalls of a model at a single glance.