schulzd_Lab5

October 19, 2020

1 Lab 5 - Modelling and Model Error

David Schulz

1.1 Introduction

In beginning of our discussions of optimization algorithms, it is important to appreciate the relationships between our model, the model's parameters, and the data sets we want to predict. For this lab we investigate two commonly used models and the data that these models attempt to replicate. This builds familiarity with the different parts of the model and shows shared themes and commonalities between models. When first using models, it can be helpful to identify their constituent parts. To help with this process, this lab will have the following format: 1. **Data Familiarity:** Identify and visualize the given data, including the data's independent and dependent variables. 2. **Model Familiarity:** Identify the model that we will use to simulate, or predict, the data given. 3. **Model Implementation:** Code the model such that it accepts inputs of independent variable(s) and tunable parameters so that it produces a prediction or dependent variable as output. 4. **Model Output and Visualization:** Visualize the model's output and plot the model's output against the given data. 5. **Error between Model and Data:** Calculate and summarize the difference, or error, between the model's output and the given data.

The two models I use are a gaussian distribution and a multi-dimensional linear model. Models take independent variables as input and produce dependent variables as their output. By adjusting the model parameters, we can change the relationship between the independent variable (feature) input and the dependent variable (response) output.

1.2 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.
    - There is no consistent relationship between anything because it depends on what the model is actually used for. There could be more than one dependent variable with any number of independent variables and any number of parameters. I think this is a good thing because it allows for a lot of flexibility to find the right model that works for a specific situation.
2. For this lab you were given datasets that had paired independent and dependent variables (supervised data).

- What would your model do if you gave it an independent variable value not from this dataset?
  - It would create a response value based on the given parameters.
- Do you think the resulting output would be correct?
  - It would be correct if the parameters are accurate.
- How can you be sure?
  - Use a method that makes sure the parameters are correct, make the prediction with the new value again, and calculate the error metric between the previously predicted value and the new one.

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.
   - Compare and contrast this new metric with mean absolute error.
     - The metric I looked up is mean squared error (MSE). MSE is almost the same as MAE, except instead of finding the absolute value of the difference between each predicted and true value, the difference is squared.
   - 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?
     - The MSE will be much more exaggerated the further away the predicted data is from the true data. MSE is exponential, while MAE is linear. With the exponential error metric, the approach to the correct parameters might be sped up if the parameters are adjusted based on the error.
4. Objectively speaking, it was a pain in the ass to plot that many figures for experiment 2.
   - Can you think of a way to plot 2 independent variables and the dependent variable on the same plot?
     - Using a contourf plot, the two independent variables would be the x and y axis and the dependent variable would be the z value at its x,y point.
   - What about 3 independent variables and the dependent variable on the same plot? 4 and 1?
     - Transform the solution space down into 2 dimensions.
   - What if you had a dataset of 100 features and 1 dependent variable?
     - Also transform the solution space down into 2 dimensions.
   - With these plots in mind, describe how the error metric can help.
     - A
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?
   - The first parameter set overestimated the responses and the second set underestimated them. A loop could be used to slightly adjust every parameter repeatedly as the MAE gets closer and closer to 0.

1.3 Experiment 1: Gaussian Distribution

1.3.1 Part 1 - Given Data

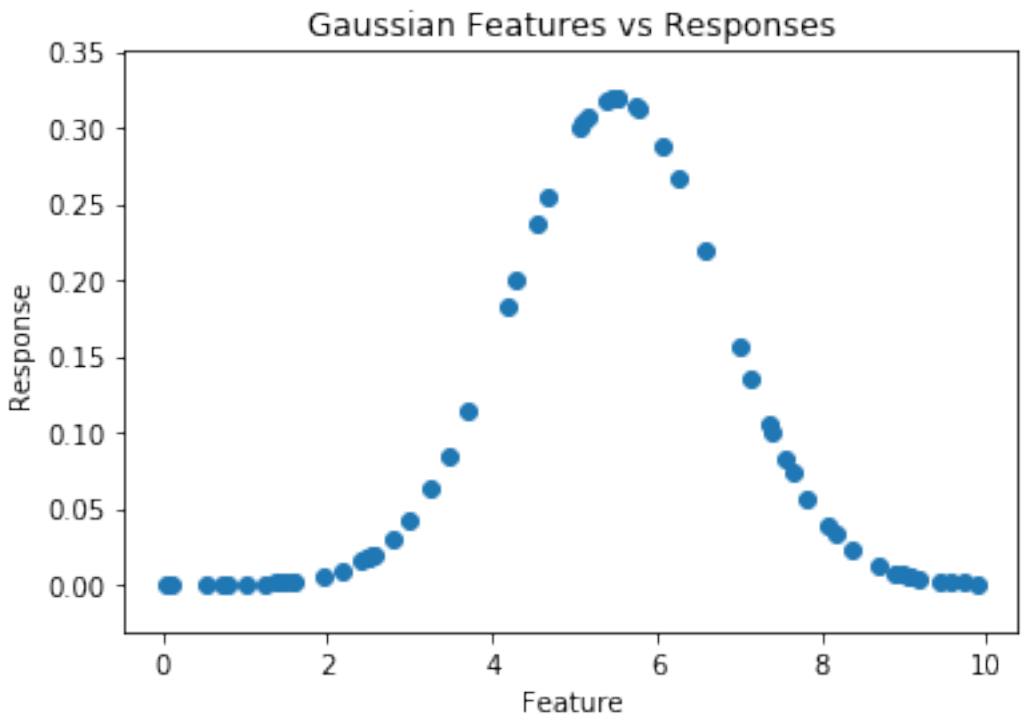
[1]:
```
import pandas as pd

gauss = pd.read_csv('gaussdist.csv', header=None)
print(gauss.head())
gauss = gauss.to_numpy()
```

```
      0         1
0  6.99  0.156842
1  8.90  0.007897
2  9.58  0.001551
3  5.46  0.318990
4  1.38  0.001396
```

Column 0 is the feature variables and column 1 is the response variables.

[2]:
```
import matplotlib.pyplot as plt

features = gauss[:, 0]
responses = gauss[:, 1]

plt.title('Gaussian Features vs Responses')
plt.xlabel('Feature')
plt.ylabel('Response')
plt.scatter(features, responses)
```

[2]:
```
<matplotlib.collections.PathCollection at 0x7f857d1cc490>
```



1.3.2 Part II - Model Familiarity

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

- Model Parameters: σ, μ
- Independent Variables: x
- Dependent Variables: $f(x)$

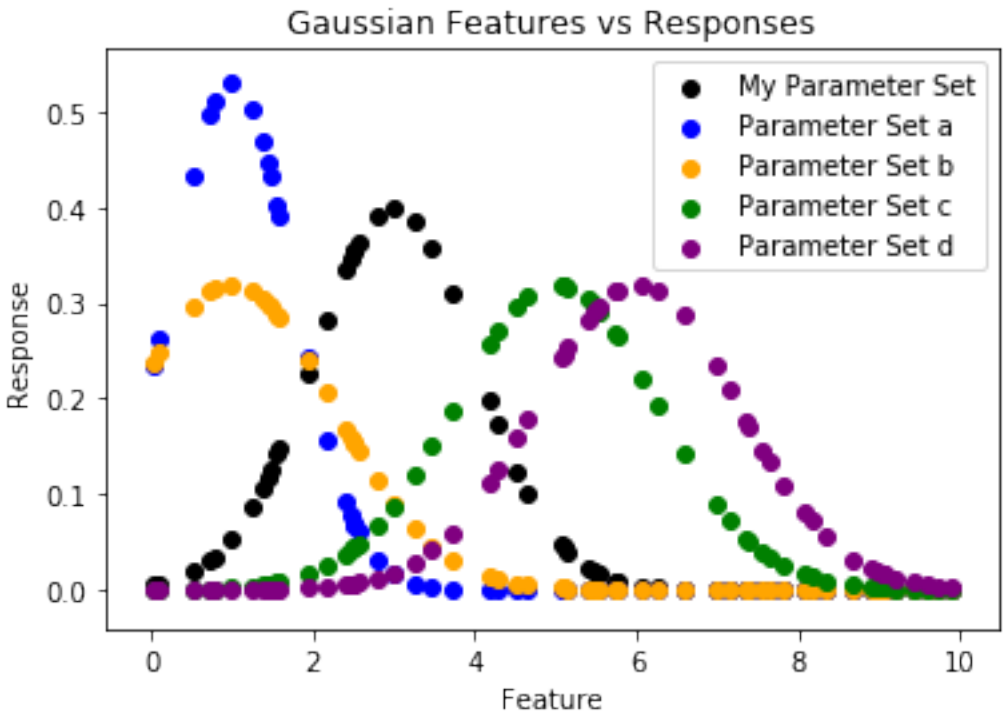
1.3.3 Part III - Model Implementation

[3]:
```
import numpy as np
import math

def gauss_model(X, mu, sig):
    return (1 / (sig * math.sqrt(2 * math.pi))) * np.exp(-1/2 * ((X - mu) /␣
 ↪sig)**2)

preds_1 = gauss_model(features, 3, 1)
```

1.3.4 Part IV - Model Output and Visualization

[4]:
```
preds_a = gauss_model(features, 1, 0.75)
preds_b = gauss_model(features, 1, 1.25)
preds_c = gauss_model(features, 5, 1.25)
preds_d = gauss_model(features, 6, 1.25)

plt.title('Gaussian Features vs Responses')
plt.xlabel('Feature')
plt.ylabel('Response')
plt.scatter(features, preds_1, c='black', label='My Parameter Set')
plt.scatter(features, preds_a, c='blue', label='Parameter Set a')
plt.scatter(features, preds_b, c='orange', label='Parameter Set b')
plt.scatter(features, preds_c, c='green', label='Parameter Set c')
plt.scatter(features, preds_d, c='purple', label='Parameter Set d')
plt.legend()
```

[4]:
```
<matplotlib.legend.Legend at 0x7f857c33ae50>
```



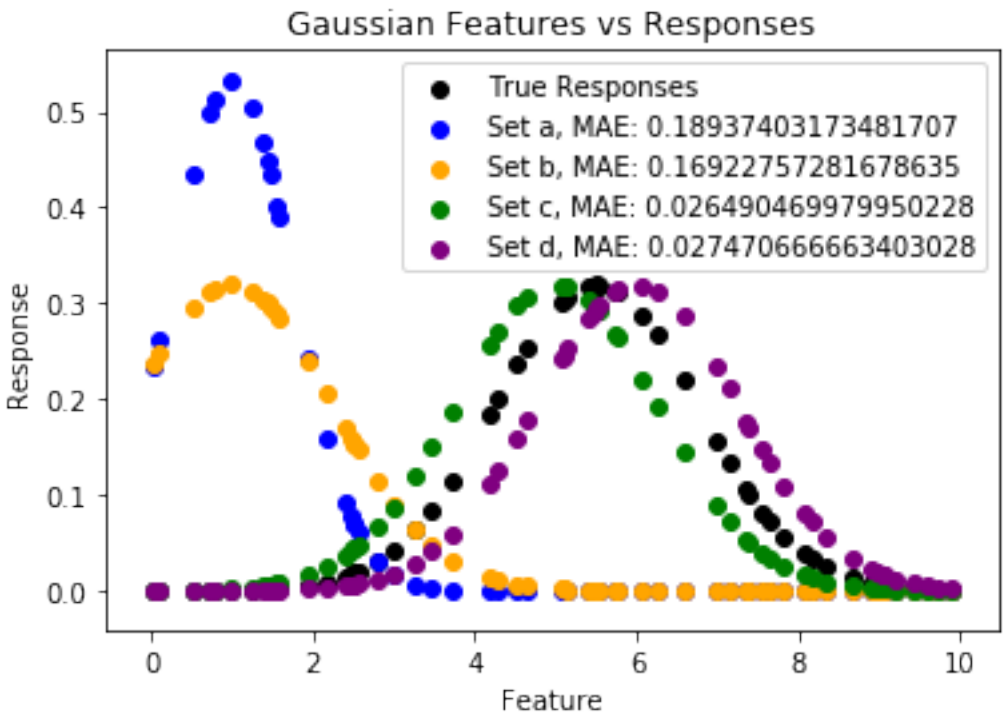
1.3.5 Part V - Error Between Model and Data

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

- Model Predictions: y
- Dependent Variables: x

[5]:
```
def calc_mae(preds, trues):
    return np.sum(np.absolute(preds - trues)) / len(trues)

a_mae = calc_mae(preds_a, responses)
b_mae = calc_mae(preds_b, responses)
c_mae = calc_mae(preds_c, responses)
d_mae = calc_mae(preds_d, responses)

plt.title('Gaussian Features vs Responses')
plt.xlabel('Feature')
plt.ylabel('Response')
plt.scatter(features, responses, c='black', label='True Responses')
plt.scatter(features, preds_a, c='blue', label='Set a, MAE: ' + str(a_mae))
plt.scatter(features, preds_b, c='orange', label='Set b, MAE: ' + str(b_mae))
plt.scatter(features, preds_c, c='green', label='Set c, MAE: ' + str(c_mae))
plt.scatter(features, preds_d, c='purple', label='Set d, MAE: ' + str(d_mae))
plt.legend()
```

[5]:
```
<matplotlib.legend.Legend at 0x7f857c261b90>
```



1.4 Experiment 2: Multiple Linear Regression Model

1.4.1 Part I - Given Data

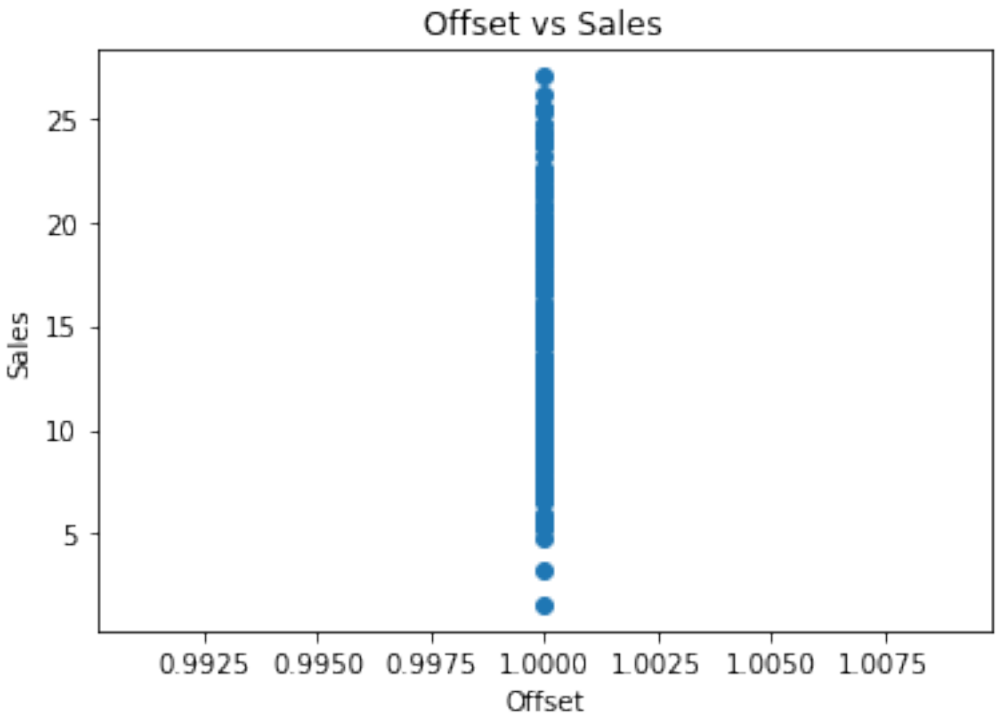
[6]:
```
advert = pd.read_csv('advertising.csv').drop('Unnamed: 0', axis=1)
print(advert.head())
advert = advert.to_numpy()
```

```
   Offset     TV  radio  newspaper  sales
0       1  230.1   37.8       69.2   22.1
1       1   44.5   39.3       45.1   10.4
2       1   17.2   45.9       69.3    9.3
3       1  151.5   41.3       58.5   18.5
4       1  180.8   10.8       58.4   12.9
```

The sales column is the dependent variable and the rest are the independent variables.

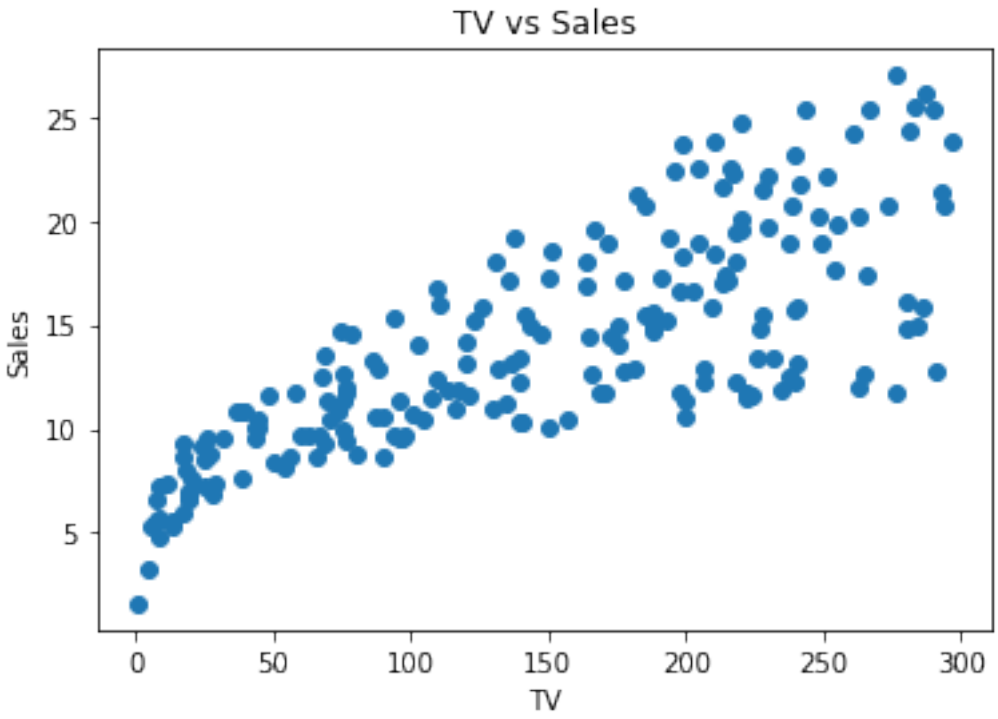
[7]:
```
features = advert[:, :4]
sales = advert[:, 4]

plt.title('Offset vs Sales')
plt.xlabel('Offset')
plt.ylabel('Sales')
plt.scatter(features[:, 0], sales)
```

[7]:
```
<matplotlib.collections.PathCollection at 0x7f857c1cb9d0>
```



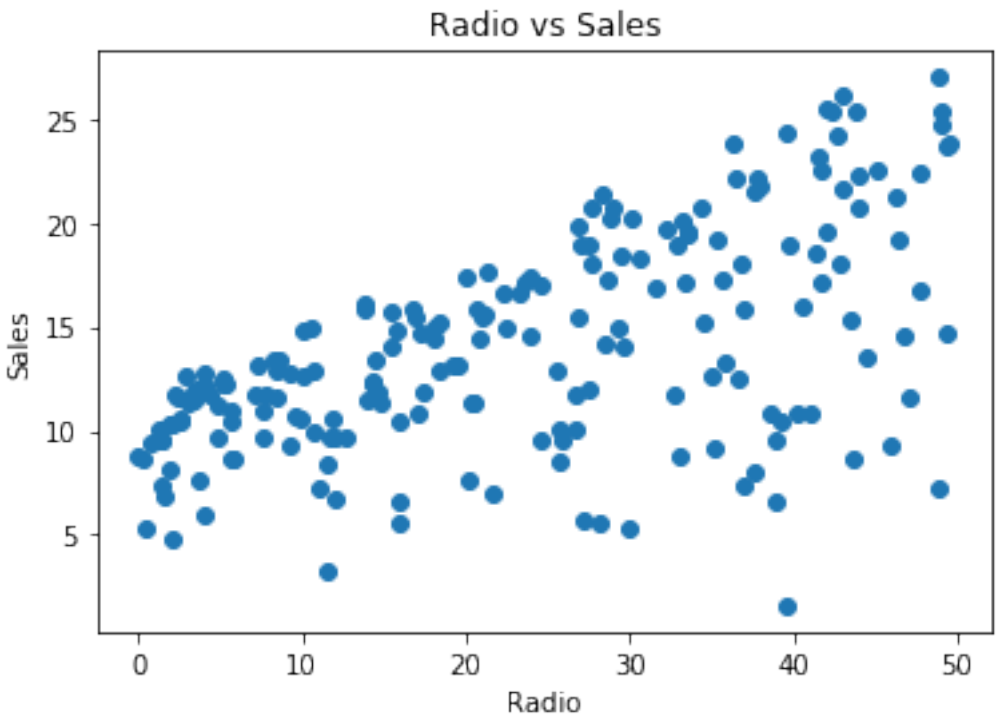
[8]:
```
plt.title('TV vs Sales')
plt.xlabel('TV')
plt.ylabel('Sales')
plt.scatter(features[:, 1], sales)
```

[8]: <matplotlib.collections.PathCollection at 0x7f857c1b0250>



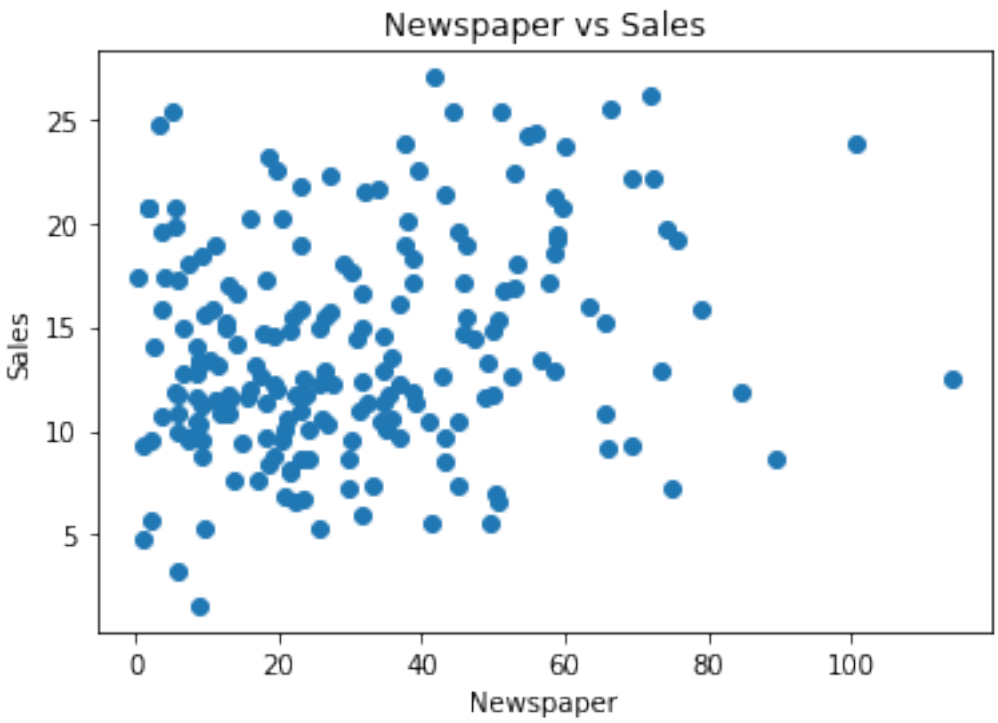
[9]:
```
plt.title('Radio vs Sales')
plt.xlabel('Radio')
plt.ylabel('Sales')
plt.scatter(features[:, 2], sales)
```

[9]: <matplotlib.collections.PathCollection at 0x7f857c13fc10>



[10]:
```
plt.title('Newspaper vs Sales')
plt.xlabel('Newspaper')
plt.ylabel('Sales')
plt.scatter(features[:, 3], sales)
```

[10]: <matplotlib.collections.PathCollection at 0x7f857c0ef7d0>



1.4.2 Part II - Model Familiarity

$$Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \beta_3 X_{i3} + \beta_4 X_{i4}$$

- Model Parameters: β_0, β_1, β_2, β_3, β_4
- Independent Variables: X
- Dependent Variables: Y

1.4.3 Part III - Model Implementation

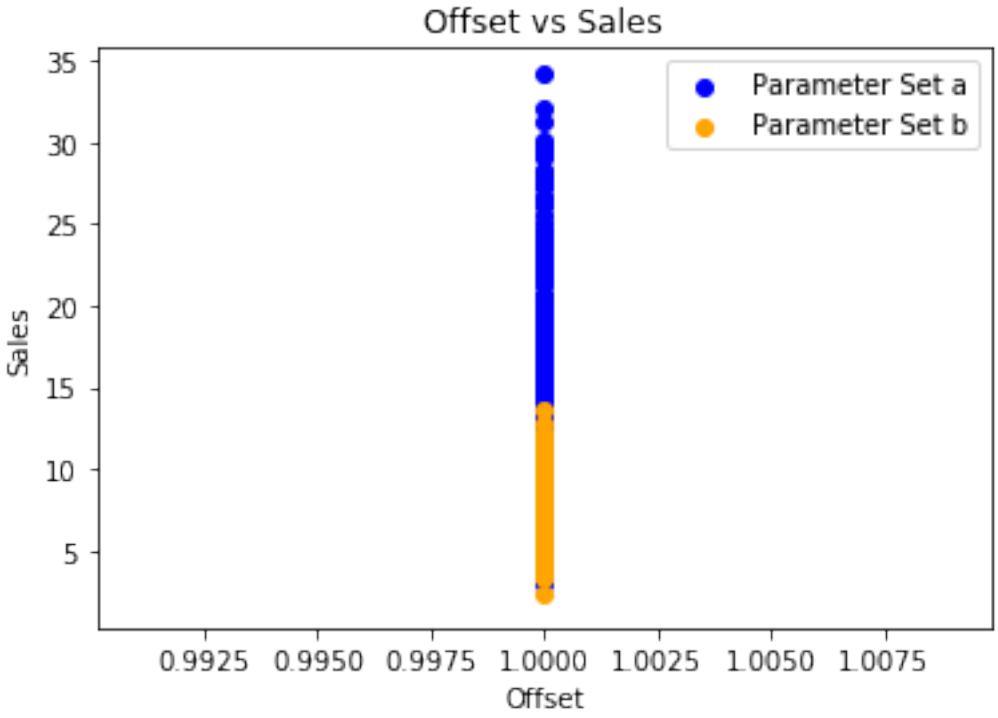
[11]:
```
def advert_model(X, weights):
    return weights[0] + (weights[1] * X[:, 0]) + (weights[2] * X[:, 1]) +␣
↪(weights[3] * X[:, 2]) + (weights[4] * X[:, 3])
```

1.4.4 Part IV - Model Output and Visualization

[12]:
```
weights_a = np.array([1, 1, 0.05, 0.2, 0.1])
preds_a = advert_model(features, weights_a)
weights_b = np.array([1, 1, 0.01, 0.1, 0.05])
preds_b = advert_model(features, weights_b)
```

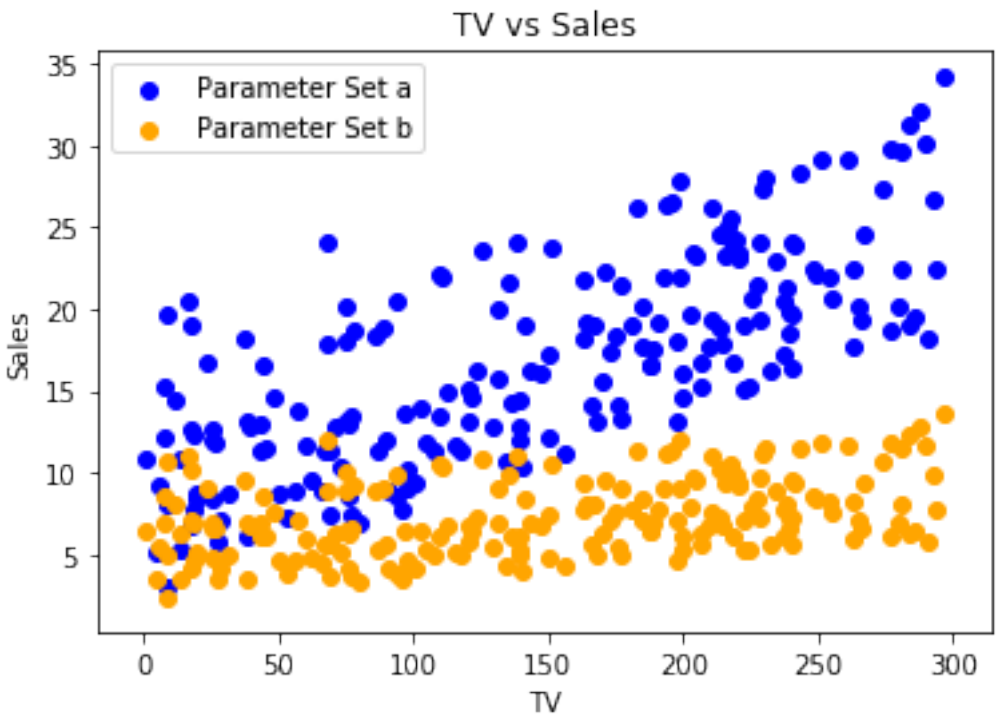
```
plt.title('Offset vs Sales')
plt.xlabel('Offset')
plt.ylabel('Sales')
plt.scatter(features[:, 0], preds_a, c='blue', label='Parameter Set a')
plt.scatter(features[:, 0], preds_b, c='orange', label='Parameter Set b')
plt.legend()
```

[12]: <matplotlib.legend.Legend at 0x7f857c05e710>



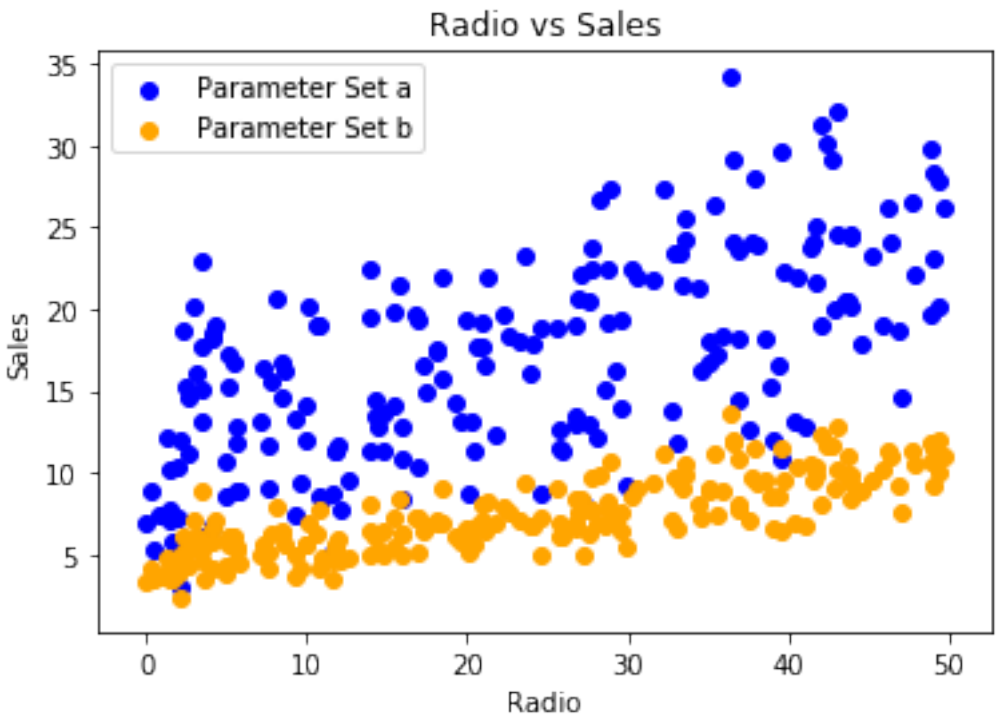
[13]:
```
plt.title('TV vs Sales')
plt.xlabel('TV')
plt.ylabel('Sales')
plt.scatter(features[:, 1], preds_a, c='blue', label='Parameter Set a')
plt.scatter(features[:, 1], preds_b, c='orange', label='Parameter Set b')
plt.legend()
```

[13]: <matplotlib.legend.Legend at 0x7f857bfd0fd0>



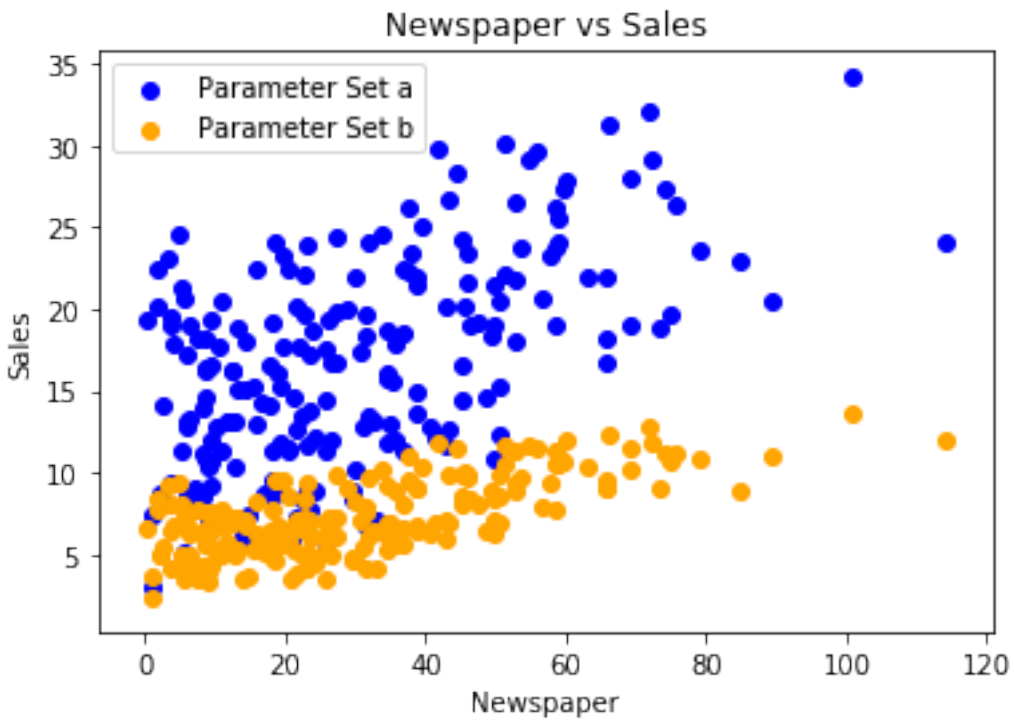
[14]:
```
plt.title('Radio vs Sales')
plt.xlabel('Radio')
plt.ylabel('Sales')
plt.scatter(features[:, 2], preds_a, c='blue', label='Parameter Set a')
plt.scatter(features[:, 2], preds_b, c='orange', label='Parameter Set b')
plt.legend()
```

[14]: <matplotlib.legend.Legend at 0x7f857bf4af10>



[15]:
```
plt.title('Newspaper vs Sales')
plt.xlabel('Newspaper')
plt.ylabel('Sales')
plt.scatter(features[:, 3], preds_a, c='blue', label='Parameter Set a')
plt.scatter(features[:, 3], preds_b, c='orange', label='Parameter Set b')
plt.legend()
```

[15]:
```
<matplotlib.legend.Legend at 0x7f857beccb10>
```



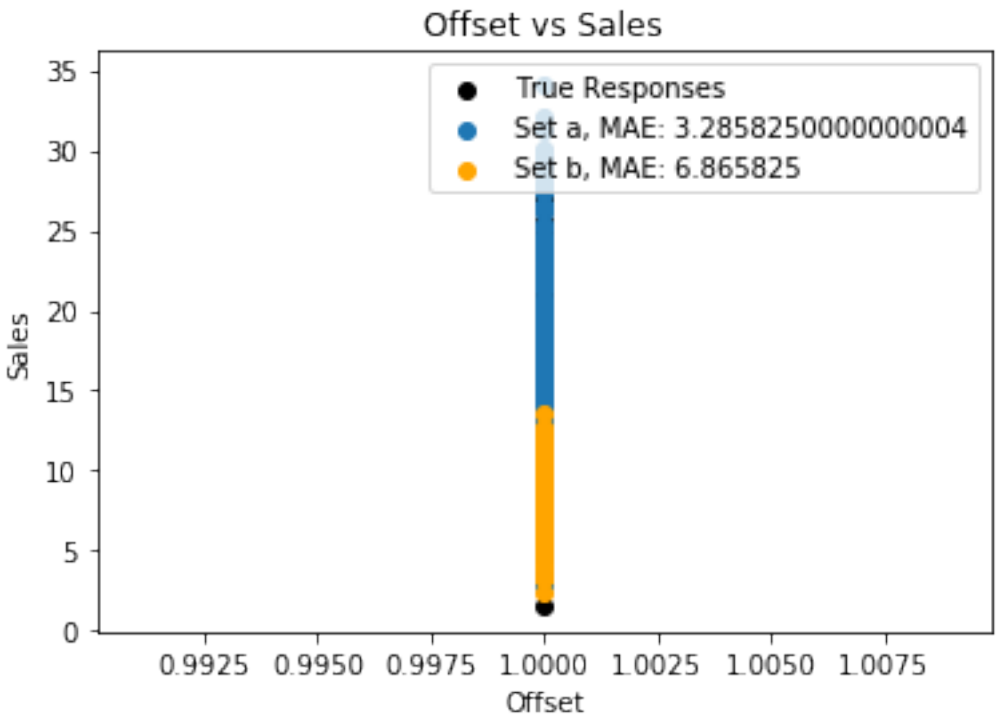
1.4.5 Part V - Error Between Model and Data

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

- Model Predictions: y
- Dependent Variables: x

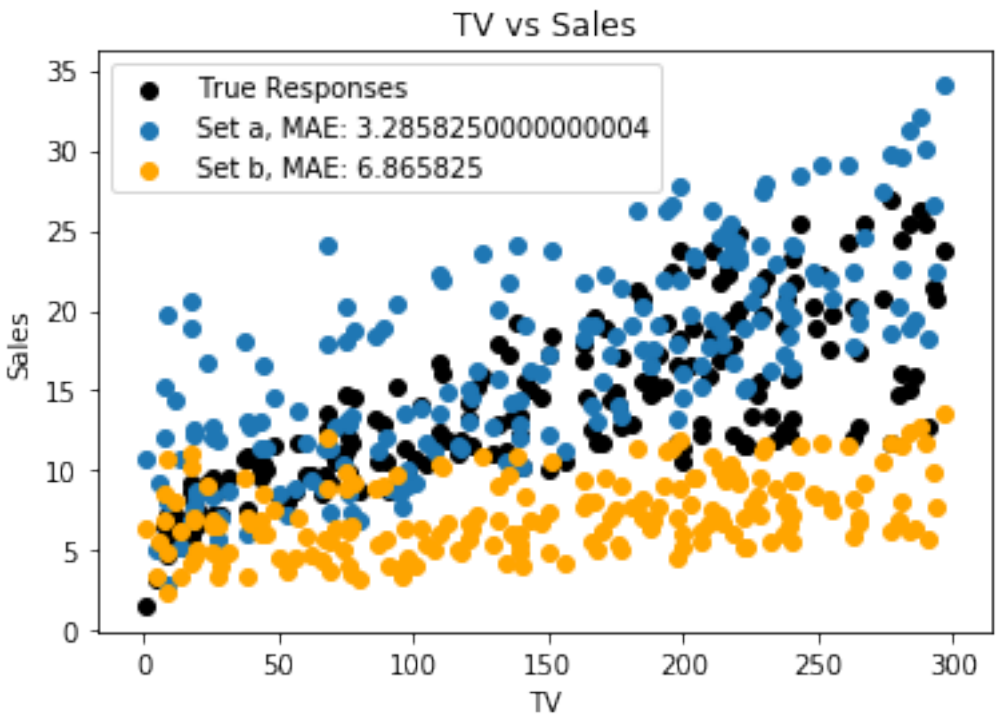
[16]:
```
def calc_mae(preds, trues):
    return np.sum(np.absolute(preds - trues)) / len(trues)

a_mae = calc_mae(preds_a, sales)
b_mae = calc_mae(preds_b, sales)

plt.title('Offset vs Sales')
plt.xlabel('Offset')
plt.ylabel('Sales')
plt.scatter(features[:, 0], sales, c='black', label='True Responses')
plt.scatter(features[:, 0], preds_a, label='Set a, MAE: ' + str(a_mae))
plt.scatter(features[:, 0], preds_b, c='orange', label='Set b, MAE: ' +␣
 ↪str(b_mae))
plt.legend()
```

[16]:
```
<matplotlib.legend.Legend at 0x7f857c2b9350>
```



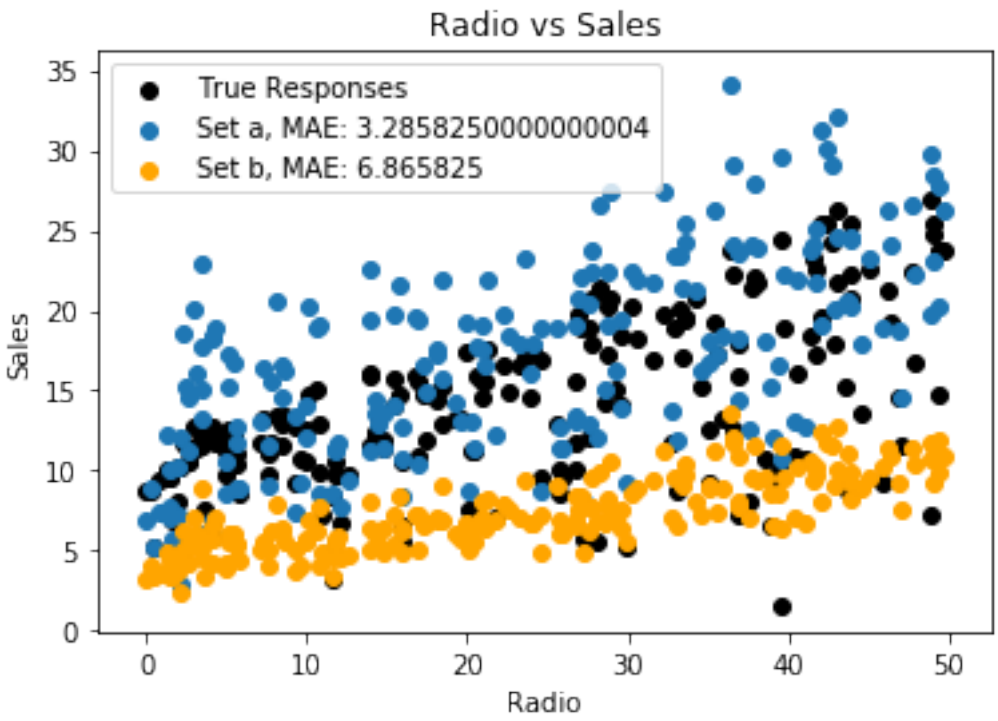
[17]:
```
plt.title('TV vs Sales')
plt.xlabel('TV')
plt.ylabel('Sales')
plt.scatter(features[:, 1], sales, c='black', label='True Responses')
plt.scatter(features[:, 1], preds_a, label='Set a, MAE: ' + str(a_mae))
plt.scatter(features[:, 1], preds_b, c='orange', label='Set b, MAE: ' +␣
 ↪str(b_mae))
plt.legend()
```

[17]:
```
<matplotlib.legend.Legend at 0x7f857be282d0>
```



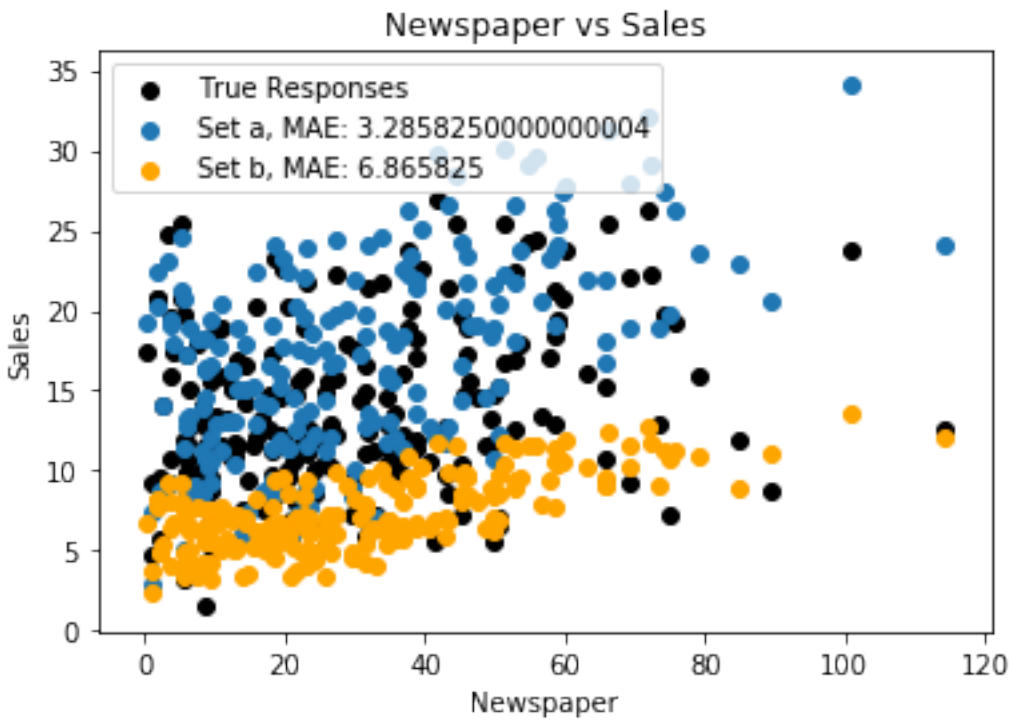
[18]:
```
plt.title('Radio vs Sales')
plt.xlabel('Radio')
plt.ylabel('Sales')
plt.scatter(features[:, 2], sales, c='black', label='True Responses')
plt.scatter(features[:, 2], preds_a, label='Set a, MAE: ' + str(a_mae))
plt.scatter(features[:, 2], preds_b, c='orange', label='Set b, MAE: ' +␣
 ↪str(b_mae))
plt.legend()
```

[18]:
```
<matplotlib.legend.Legend at 0x7f857bda2a50>
```



[19]:
```
plt.title('Newspaper vs Sales')
plt.xlabel('Newspaper')
plt.ylabel('Sales')
plt.scatter(features[:, 3], sales, c='black', label='True Responses')
plt.scatter(features[:, 3], preds_a, label='Set a, MAE: ' + str(a_mae))
plt.scatter(features[:, 3], preds_b, c='orange', label='Set b, MAE: ' +␣
 ↪str(b_mae))
plt.legend()
```

[19]:
```
<matplotlib.legend.Legend at 0x7f857bd35fd0>
```


Newspaper vs Sales
True Responses
Set a, MAE: 3.2858250000000004
Set b, MAE: 6.865825
Sales
Newspaper
0
5
10
15
20
25
30
35
0
20
40
60
80
100
120