UGBA88 Lab L

April 30, 2020

1 Lab L: More Linear Regression

Address each data retrieval and analysis section prompt with python code. Answer each discussion section prompt with a few thoughtful sentences.

1.1 Setup

[64]:
```
# Import some useful functions
from numpy import *
from numpy.random import *
from datascience import *

# Import more useful functions for linear regression
from statsmodels.formula.api import *

# Define some useful functions
def correlation(array_1, array_2):
    return corrcoef(array_1, array_2).item(1)

# Customize look of graphics
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
%matplotlib inline

# Force display of all values
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

# Handle some obnoxious warning messages
```

```python
import warnings
warnings.filterwarnings("ignore")
```

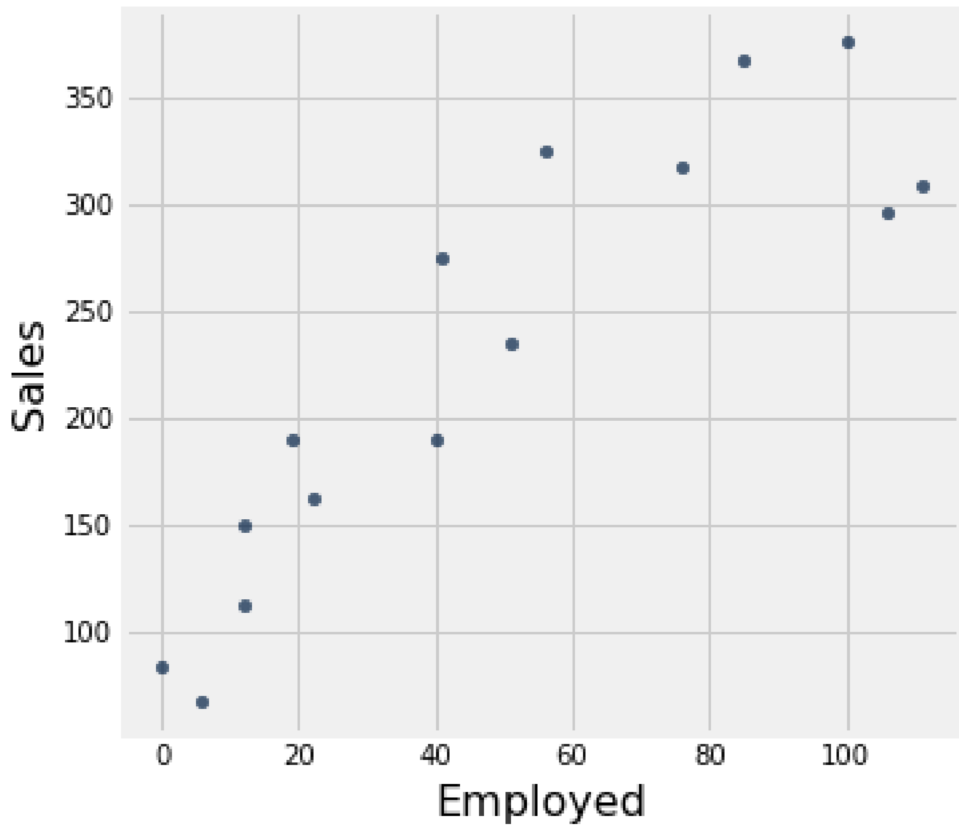
1.2 Sales Experience

Reynolds Products is assessing the effectiveness of its new sales staff, which is hired contigent on sales performance during an intial 90-day probation period. Management suspects that average daily sales for an individual salesperson increases with experience during and beyond the probation period.

1.2.1 Retrieve Data

[65]:
```python
# Retrieve a dataset from file 'Reynolds.csv'.  Show the first few observations.
# Visualize the dataset as a scatterplot.
data = Table().read_table('Reynolds.csv')
data
data.scatter("Employed")
```

[65]:
```
Employed | Sales
41       | 275
106      | 296
76       | 317
100      | 376
22       | 162
12       | 150
85       | 367
111      | 308
40       | 189
51       | 235
… (5 rows omitted)
```



1.2.2 Analysis

One-Piece Model

[66]:
```
# Build a linear regression model to predict a salesperson's average sales␣
↪based on number of days employed.
# Show the model goodness of fit (R^2).
# Show the model parameters.
# Show the model residuals.
model = ols('Sales ~ Employed', data).fit()
model.rsquared
model.params
model.resid
```

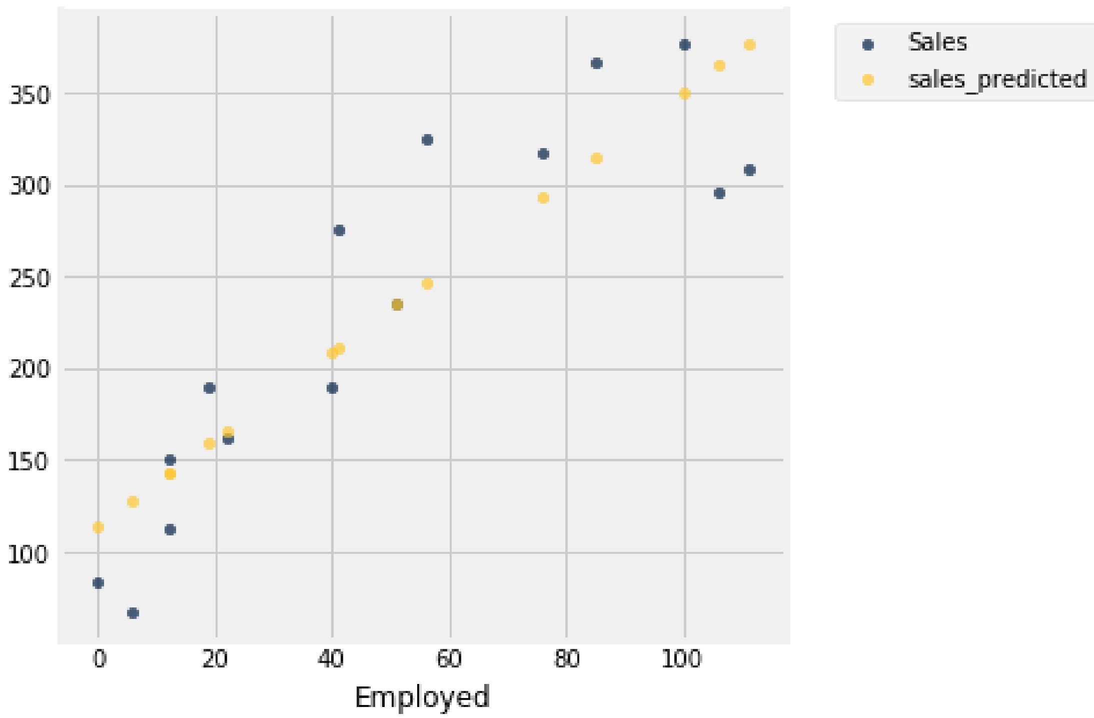
[66]:
```
0.7901387919147588
```

[66]:
```
Intercept    113.745287
Employed       2.367464
dtype: float64
```

[66]:
```
0     64.188704
1    -68.696431
2     23.327477
3     25.508350
4     -3.829487
5      7.845149
6     52.020305
7    -68.533749
8    -19.443832
9      0.514068
10   -30.745287
11   -30.154851
12   -60.950069
13    78.676750
14    30.272904
dtype: float64
```

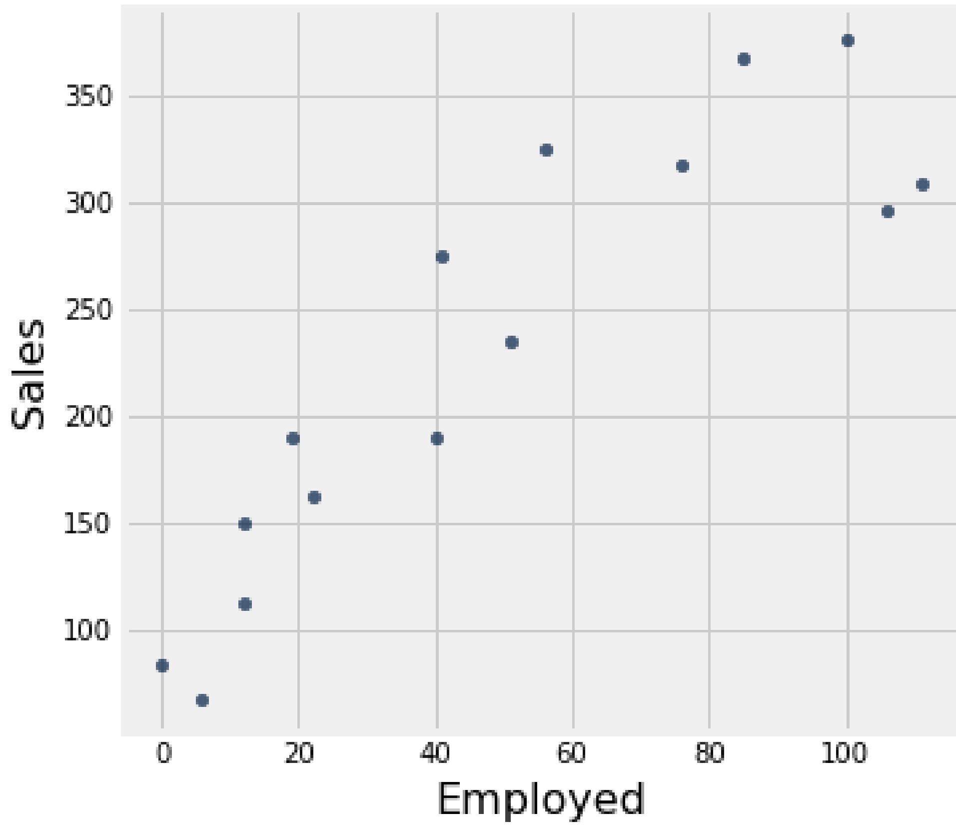
[67]:
```python
# Add a variable to the dataset for predicted sales.
# Show the RMSE calculated based on the dataset.
# Visualize the model performance as a scatterplot of
# average sales and predicted average sales vs. number of days employed.
data = data.with_column('sales_predicted', model.predict(data))
RMSE = sqrt(mean(model.resid**2))
RMSE
data.scatter('Employed')
```

[67]:
```
45.14254460064975
```



Two-Piece Model

[68]:
```
# Reset dataset to include average sales and number of days employed variables␣
↪only.
# Visualize the dataset as a scatterplot.
data = data.select('Employed','Sales')
data.scatter('Employed')
```



[69]:
```
# Set the breakpoint to 90.  This is the predictor variable value at which an␣
↪apparent discontinuity occurs.
breakpoint = 90
breakpoint
```

[69]: 90

[70]:
```
# Filter the dataset to include only sales associated with number of days␣
↪employed
# less than or equal to the breakpoint.
# Show the resulting filtered dataset.
data1 = data.where('Employed', are.below_or_equal_to(breakpoint))
data1.show()
```

<IPython.core.display.HTML object>

[71]:
```
# Build a linear regression model to predict a salesperson's average sales␣
↪based on number of days employed
# (use the filtered dataset).
# Show the model goodness of fit (R^2).
```

```python
# Show the model parameters.
# Show the model residuals.
model1 = ols("Sales ~ Employed", data1).fit()
model1.rsquared
model1.params
model1.resid
```

[71]: 0.8962746859309657

[71]:
```
Intercept    88.302977
Employed      3.360391
dtype: float64
```

[71]:
```
0      48.920986
1     -26.692703
2      -0.231582
3      21.372330
4      -6.936224
5     -33.718622
6     -24.682925
7      -5.302977
8     -16.627670
9     -41.465324
10     48.515119
11     36.849592
dtype: float64
```

[72]:
```python
# Filter the (unfiltered) data to include only sales associated with number of days employed
# greater than the breakpoint.
# Show the resulting filtered dataset.
data2 = data.where("Employed", are.above(breakpoint))
data2
```

[72]:
```
Employed | Sales
106      | 296
100      | 376
111      | 308
```

[73]:
```python
# Build a linear regression model to predict a salesperson's average sales based on number of days employed
# (use the filtered dataset).
# Show the model goodness of fit (R^2).
# Show the model parameters.
# Show the model residuals.
model2 = ols("Sales ~ Employed", data2).fit()
model2.rsquared
```

```
model2.params
model2.resid
```

[73]: 0.6711798230422877

[73]:
```
Intercept    1004.791209
Employed       -6.417582
dtype: float64
```

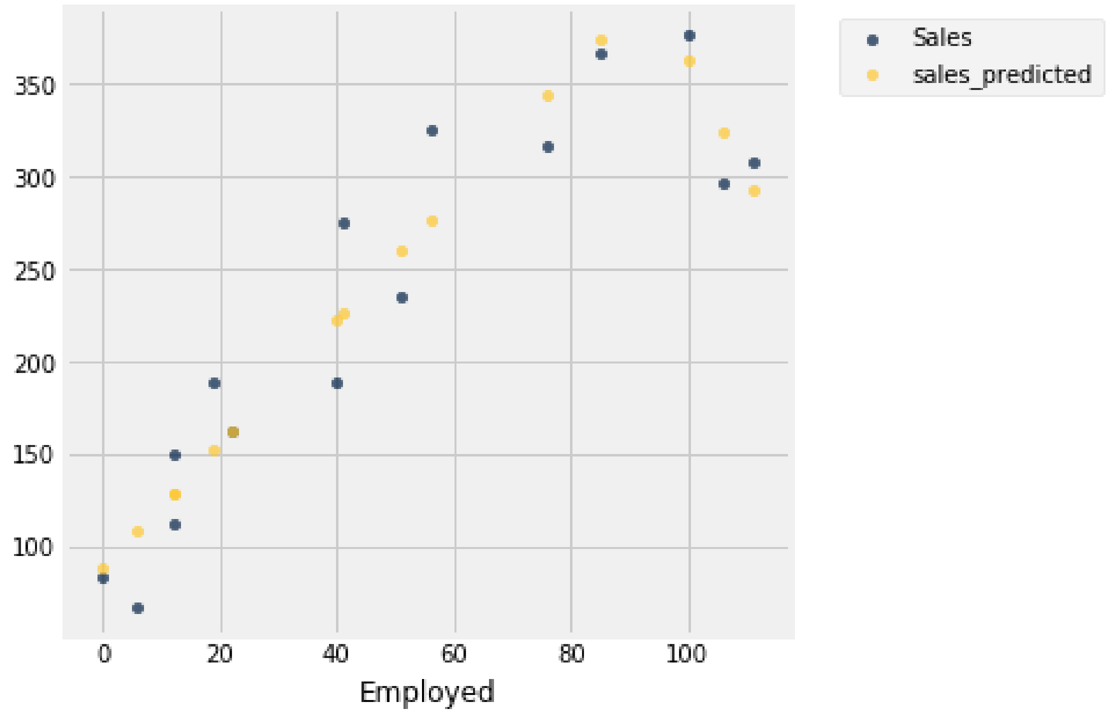
[73]:
```
0   -28.527473
1    12.967033
2    15.560440
dtype: float64
```

[74]:
```
# Build a dataset that recombines both filtered datasets.
# Add a variable to the dataset for predicted sales.
# Add a variable to the dataset for residuals.
# Show the resulting dataset.
data1 = data1.with_columns('sales_predicted', model1.predict(data1),␣
 ↪'residual', model1.resid)
data2 = data2.with_columns('sales_predicted', model2.predict(data2),␣
 ↪'residual', model2.resid)
data = data1.with_rows(data2.rows)
data.show()
```

```
<IPython.core.display.HTML object>
```

[75]:
```
# Show the RMSE calculated based on the dataset.
# Visualize the performance of the 2-piece model as a scatterplot of average␣
 ↪sales and predicted average sales
# vs. number of days employed
RMSE = sqrt(mean(data.column('residual')**2))
RMSE
data.select('Employed','Sales','sales_predicted').scatter('Employed')
```

[75]: 28.65492171823



Piecewise Model

[76]:
```
# Reset dataset to include average sales and number of days employed variables␣
↪only.
# Visualize the dataset as a scatterplot.
data = data.select('Employed','Sales')
data
```

[76]:
```
Employed | Sales
41       | 275
76       | 317
22       | 162
12       | 150
85       | 367
40       | 189
51       | 235
0        | 83
12       | 112
6        | 67
… (5 rows omitted)
```

[77]:
```
# Set the breakpoint to 90.  This is the predictor variable value at which an␣
↪apparent discontinuity occurs.
```

```
# Add a variable to the datset for switching (0 means number of days employed␣
 ↪is greater than breakpoint, 1 means otherwise).
# Add a variable to the dataset for adjustment, like this: (number of days␣
 ↪employed - breakpoint) * switch
# Show the resulting dataset.
data = data.with_column('switch', (data.column('Employed') > breakpoint).
 ↪astype(int))
data = data.with_column('adjust', (data.column('Employed')-breakpoint)*data.
 ↪column('switch'))
data.show()
```

```
<IPython.core.display.HTML object>
```

[78]:
```
# Build a linear regression model to predict a salesperson's average sales␣
 ↪based on number of days employed and an adjustment.
# Show the model goodness of fit (R^2).
# Show the model parameters.
# Show the model residuals.
model = ols('Sales ~ Employed + adjust', data).fit()
model.rsquared
model.params
model.resid
```

[78]:
```
0.9135462367549996
```

[78]:
```
Intercept    87.217242
Employed      3.409432
adjust       -7.872553
dtype: float64
```

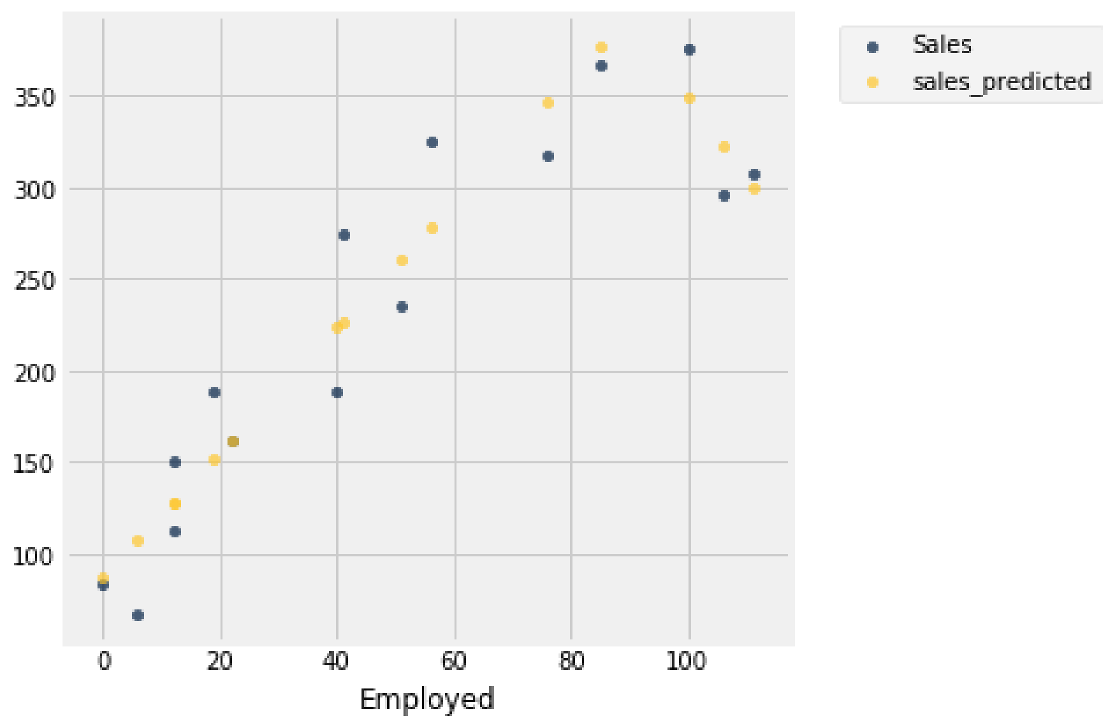
[78]:
```
0     47.996047
1    -29.334073
2     -0.224746
3     21.869574
4    -10.018961
5    -34.594521
6    -26.098273
7     -4.217242
8    -16.130426
9    -40.673834
10    46.854567
11    37.003550
12   -26.656180
13    26.565092
14     7.659426
dtype: float64
```

[79]:
```
# Add a variable to the dataset for predicted sales.
# Show the resulting dataset.
data = data.with_column('sales_predicted', model.predict(data))
data.show()
```

```
<IPython.core.display.HTML object>
```

[80]:
```
# Show the RMSE calculated based on the dataset.
# Visualize the performance of the model as a scatterplot of average sales and␣
↪predicted average sales
# vs. number of days employed.
RMSE = sqrt(mean(model.resid**2))
RMSE
data.select('Employed','Sales','sales_predicted').scatter('Employed')
```

[80]:
```
28.974229963184897
```



[81]:
```
# Predict the average sales of a salesperson that has been employed for 90 days.
model.predict(Table().with_columns('Employed', 90, 'adjust', (90-breakpoint)*0))
model.predict(Table().with_columns('Employed', 90, 'adjust', (90-breakpoint)*1))
```

[81]:
```
0    394.06612
dtype: float64
```

[81]:
```
0    394.06612
dtype: float64
```

1.2.3 Discussion

Assume that you are the manager of new hire sales for Reynolds Products. What do you think about the effectiveness of the probation period for new new salespeople?

What does your analysis tell you about how to make business decisions?

The probational period seems effective during the probational period itself, as sales are shown to climb throughout this timeframe. However, immediately following the end of the probational period is a decline in the number of average sales, indicating a defectiveness within the system. I would try to redesign the probational period or the atmosphere of the workplace for established hires that would encourage workers to keep improving and working at a high capacity beyond the probational period.

This tells me that when I make business decisions, I can design models to pinpoint problems/inefficiencies within the systems I have implemented, as well as access the success of my business. '

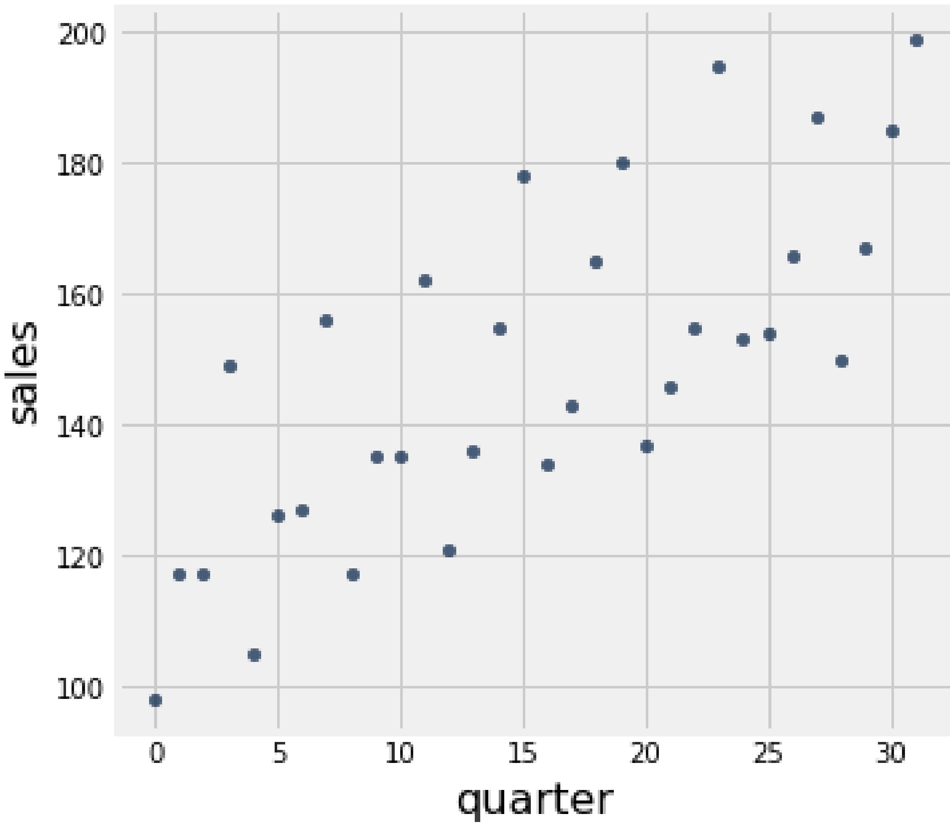
1.3 Smartphones

Investors in a smartphone retailer wants to assess the company's growth potential.

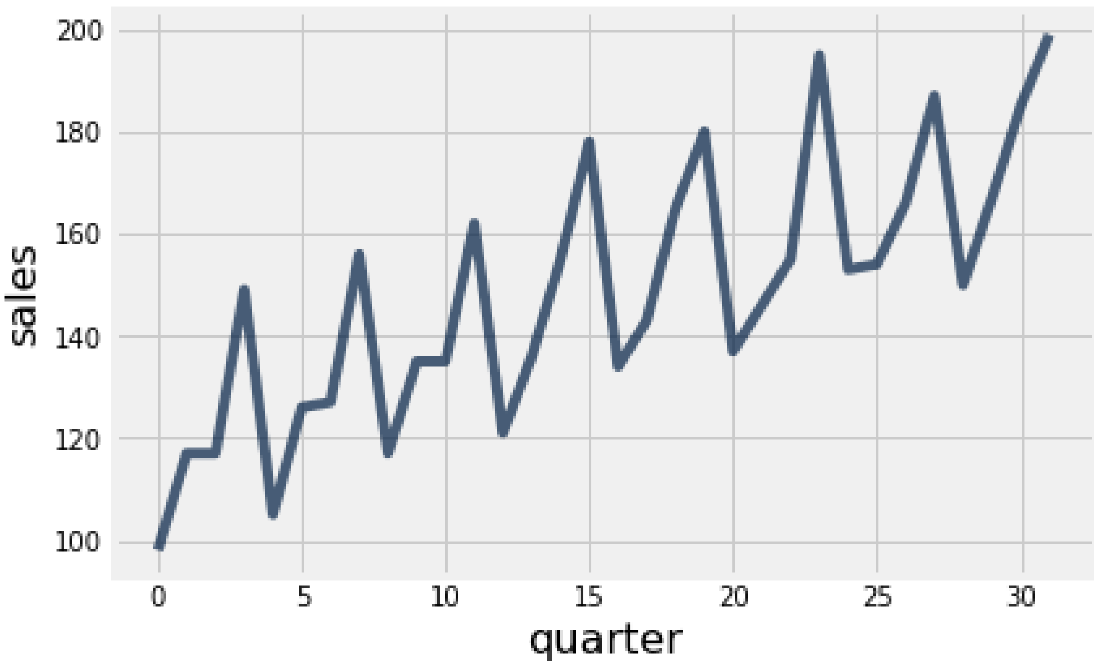
1.3.1 Retrieve Data

[82]:
```
# Retrieve a dataset from file 'Smartphones.csv'.  Show the first few␣
↪observations.
# Visualize the dataset as a scatterplot.
data = Table().read_table('Smartphones.csv')
data
data.scatter('quarter')
```

[82]:
```
quarter | sales
0       | 98
1       | 117
2       | 117
3       | 149
4       | 105
5       | 126
6       | 127
7       | 156
8       | 117
9       | 135
… (22 rows omitted)
```



[83]:
```
# Visualize the dataset as a lineplot.
data.plot('quarter')
```



1.3.2 Analysis

Model Accounts for Trend

[84]:
```
# Build a linear regression model to predict sales based on quarter.
# Show the model goodness of fit (R^2).
# Show the model parameters.
# Show the model residuals.
model = ols('sales ~ quarter', data).fit()
model.rsquared
model.params
model.resid
```

[84]:
```
0.5816018712562949
```

[84]:
```
Intercept    115.852273
quarter        2.102273
dtype: float64
```

[84]:
```
0    -17.852273
1     -0.954545
2     -3.056818
3     26.840909
4    -19.261364
5     -0.363636
```

```
6     -1.465909
7     25.431818
8    -15.670455
9      0.227273
10    -1.875000
11    23.022727
12   -20.079545
13    -7.181818
14     9.715909
15    30.613636
16   -15.488636
17    -8.590909
18    11.306818
19    24.204545
20   -20.897727
21   -14.000000
22    -7.102273
23    30.795455
24   -13.306818
25   -14.409091
26    -4.511364
27    14.386364
28   -24.715909
29    -9.818182
30     6.079545
31    17.977273
dtype: float64
```

[85]:

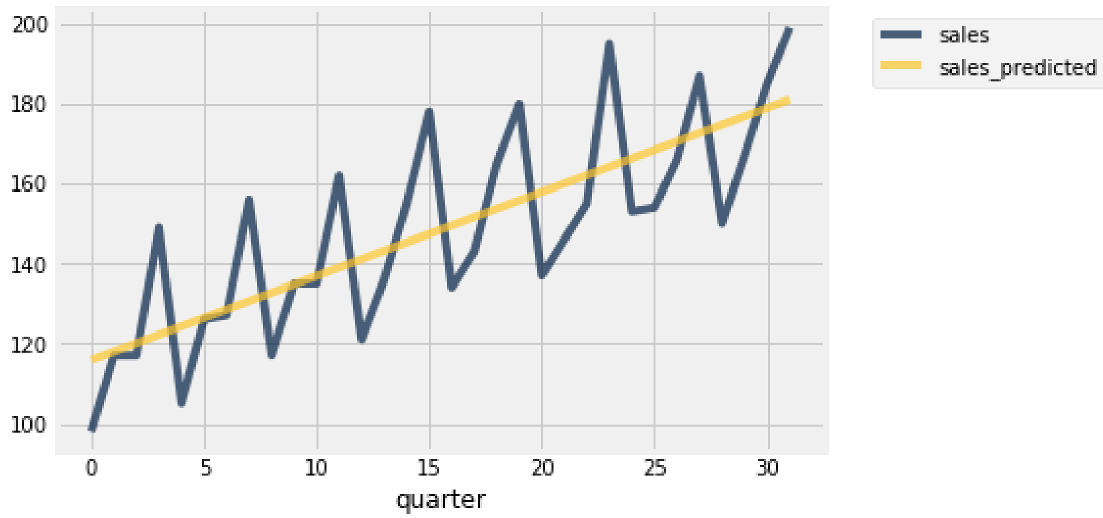
```
# Add a variable to the dataset for predicted sales.
# Show the resulting dataset.
data = data.with_column('sales_predicted', model.predict(data))
data
```

[85]:

```
quarter | sales | sales_predicted
0       | 98    | 115.852
1       | 117   | 117.955
2       | 117   | 120.057
3       | 149   | 122.159
4       | 105   | 124.261
5       | 126   | 126.364
6       | 127   | 128.466
7       | 156   | 130.568
8       | 117   | 132.67
9       | 135   | 134.773
… (22 rows omitted)
```

[86]:
```
# Show the RMSE calculated based on the dataset.
# Visualize the performance of the model as a scatterplot of sales and␣
 ↪predicted sales vs. quarter.
RMSE = sqrt(mean(model.resid**2))
RMSE
data.plot('quarter')
```

[86]: 16.46335030937175



Model Accounts for Trend & Seasonality

[87]:
```
# Reset dataset to include sales and quarter variables only.  SHow the first␣
 ↪few observations.
data = data.select('quarter','sales')
data
```

[87]:

quarter	sales
0	98
1	117
2	117
3	149
4	105
5	126
6	127
7	156
8	117
9	135

… (22 rows omitted)

[88]:
```
# Add a variable to the dataset for quarter name (1 means 1st quarter, etc.).
# Show the first few observations of teh resulting dataset.
# You can use the mod function for modulo arithmetic, e.g., mod(data.
↪column('quarter'),4)+1.
data = data.with_column('quarter_name', mod(data.column('quarter'),4)+1)
data
```

[88]:
```
quarter | sales | quarter_name
0       | 98    | 1
1       | 117   | 2
2       | 117   | 3
3       | 149   | 4
4       | 105   | 1
5       | 126   | 2
6       | 127   | 3
7       | 156   | 4
8       | 117   | 1
9       | 135   | 2
… (22 rows omitted)
```

[89]:
```
# Add dummy variables to the dataset for quarter name.  Show the first few␣
↪observations.
data = data.with_columns('q1', (data.column('quarter_name')==1).astype(int),␣
↪'q2', (data.column('quarter_name')==2).astype(int), 'q3', (data.
↪column('quarter_name')==3).astype(int), 'q4', (data.
↪column('quarter_name')==4).astype(int))
data
```

[89]:
```
quarter | sales | quarter_name | q1   | q2   | q3   | q4
0       | 98    | 1            | 1    | 0    | 0    | 0
1       | 117   | 2            | 0    | 1    | 0    | 0
2       | 117   | 3            | 0    | 0    | 1    | 0
3       | 149   | 4            | 0    | 0    | 0    | 1
4       | 105   | 1            | 1    | 0    | 0    | 0
5       | 126   | 2            | 0    | 1    | 0    | 0
6       | 127   | 3            | 0    | 0    | 1    | 0
7       | 156   | 4            | 0    | 0    | 0    | 1
8       | 117   | 1            | 1    | 0    | 0    | 0
9       | 135   | 2            | 0    | 1    | 0    | 0
… (22 rows omitted)
```

[90]:
```
# Build a linear regression model to predict sales based on quarter and quarter␣
↪name
# (for quarter name, use only the dummy variables that you need).
# Show the model goodness of fit (R^2).
# Show the model parameters.
# Show the model residuals.
```

```python
model = ols('sales ~ quarter + q1 + q2 + q3 + q4', data).fit()
model.rsquared
model.params
model.resid
```

```
[90]: 0.9617204753089558
```

```
[90]: Intercept    95.186310
      quarter       1.900298
      q1            5.084524
      q2           16.809226
      q3           25.033929
      q4           48.258631
      dtype: float64
```

```
[90]: 0     -2.270833
      1      3.104167
      2     -7.020833
      3     -0.145833
      4     -2.872024
      5      4.502976
      6     -4.622024
      7     -0.747024
      8      1.526786
      9      5.901786
      10    -4.223214
      11    -2.348214
      12    -2.074405
      13    -0.699405
      14     8.175595
      15     6.050595
      16     3.324405
      17    -1.300595
      18    10.574405
      19     0.449405
      20    -1.276786
      21    -5.901786
      22    -7.026786
      23     7.848214
      24     7.122024
      25    -5.502976
      26    -3.627976
      27    -7.752976
      28    -3.479167
      29    -0.104167
      30     7.770833
      31    -3.354167
```

```
dtype: float64
```

[91]:
```
# Add variable to the dataset for predicted sales.
data = data.with_column('sales_predicted', model.predict(data))
data
```

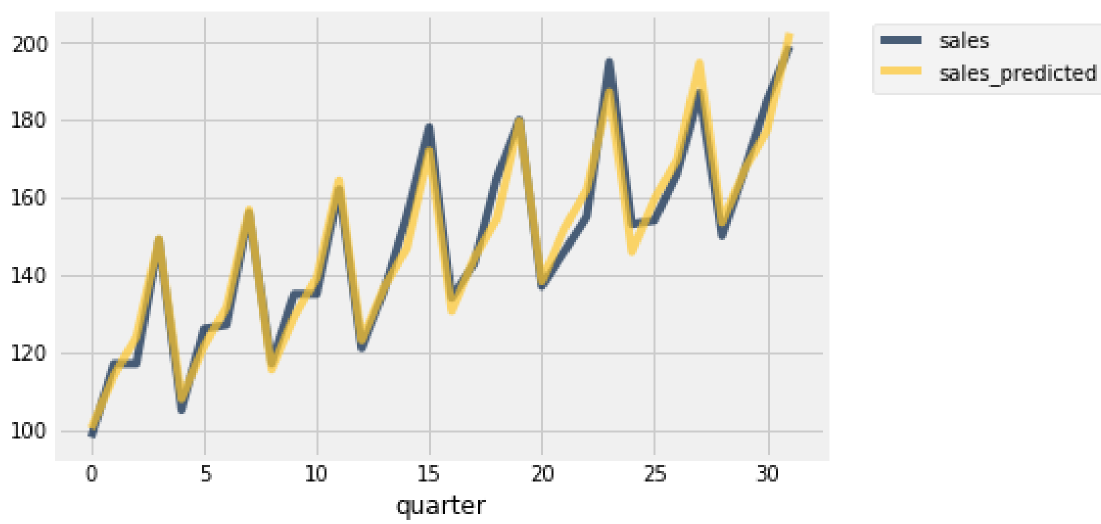
[91]:

quarter	sales	quarter_name	q1	q2	q3	q4	sales_predicted
0	98	1	1	0	0	0	100.271
1	117	2	0	1	0	0	113.896
2	117	3	0	0	1	0	124.021
3	149	4	0	0	0	1	149.146
4	105	1	1	0	0	0	107.872
5	126	2	0	1	0	0	121.497
6	127	3	0	0	1	0	131.622
7	156	4	0	0	0	1	156.747
8	117	1	1	0	0	0	115.473
9	135	2	0	1	0	0	129.098

… (22 rows omitted)

[92]:
```
# Show the RMSE calculated based on the dataset.
# Visualize the performance of the model as a scatterplot of sales and␣
↪predicted sales vs. quarter.
RMSE = sqrt(mean(model.resid**2))
RMSE
data.select('quarter','sales','sales_predicted').plot('quarter')
```

[92]: 4.979739456992081



1.3.3 Discussion

Assume that you are the director of financial planning and analysis for the smartphone company. How would you estimate next year's sales? How confident would you be about your estimate?

Assume that you are an investor in the smartphine company. How connfident are you about the company's growth potential?

What does your analysis tell you about how to make business decisions?

Based off the analysis and the patterns in the data, I would estimate that overall, we would see an increase in sales, with a sharp increase during q1, tapered down during q2, another increase during q3, and then a sharp decrease of sales during q4, and so on an so forth in a cyclical pattern.

If I were an investor, I would be fairly confident in the company's growth potential as they have demonstrated a positive growth trend overall, and their sales patterns seem to consistently follow the aforementioned pattern. Both these things make me fairly confident in the company's growth potential.

My analysis tells me that I should make business decisions based on accurate models that are designed to suit the data the best; like in this section, where we created a model based off the data and then refined it according to quarters to obtain the most accurate insights.

1.4 Coffee Shop

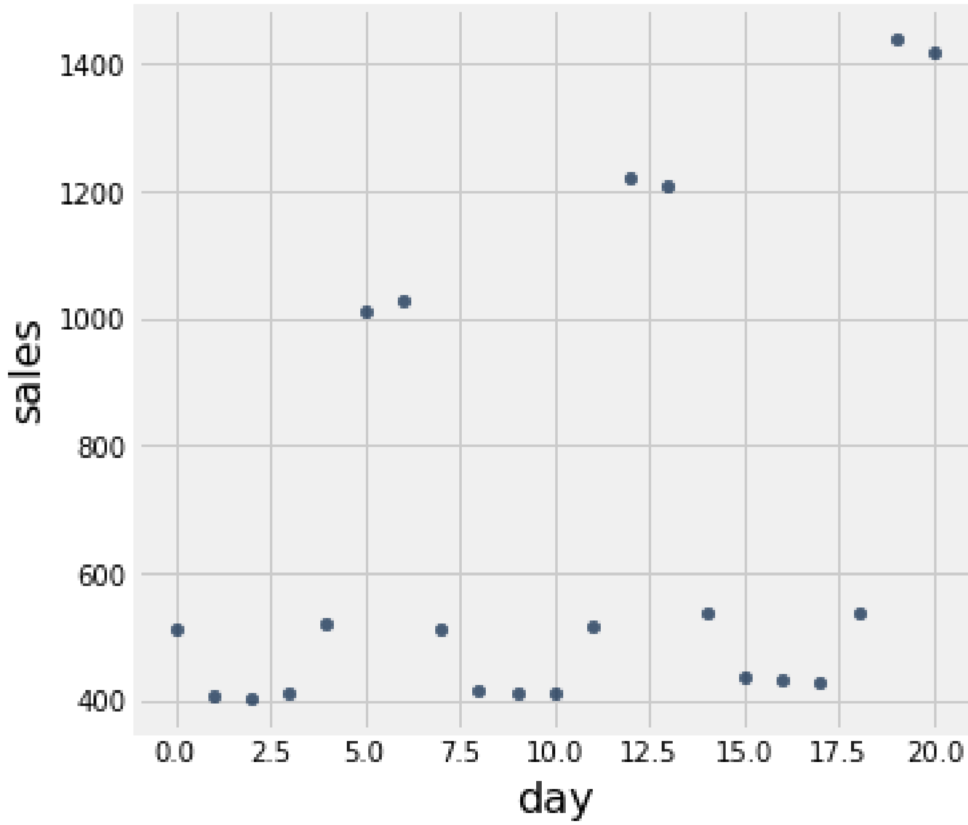
A newly opened coffee shop has tracked daily sales for 3 weeks. It seems that sales is affected by both trend and seasonality. Sales on certain days of the week always seem higher than on others, and both non-weekend and weekend sales are increasing but at different rates.

1.4.1 Retrieve Data

[96]:
```
# Retrieve a dataset from file 'Coffee_Shop.csv'.  Show the first few␣
↪observations.
# Visualize the dataset as a scatterplot.
# Note: day 0 is a Monday.
data = Table.read_table('Coffee_Shop.csv')
data
data.scatter('day')
```

[96]:
```
day  | sales
0    | 510
1    | 405
2    | 403
3    | 410
4    | 520
5    | 1010
6    | 1030
7    | 512
```

```
8     | 415
9     | 413
… (11 rows omitted)
```



1.4.2 Analysis

Multiple Linear Regression Model

[100]:
```
# Add a variable to the dataset for day of week (0 through 6, where 0 means␣
↪Monday, 1 means Tuesday, etc.)
# Add dummy variables to the dataset for day of week.
# Add a variable to the dataset for weekend (1 means Saturday or Sunday, 0␣
↪means not Saturday or Sunday).
# Show the resulting first few observations.
data = data.with_column('day_of_week', mod(data.column('day'),7)+1)
```

```
data = data.with_columns('mon', (data.column('day_of_week')==1).astype(int),␣
↪'tue', (data.column('day_of_week')==2).astype(int), 'wed', (data.
↪column('day_of_week')==3).astype(int), 'thu', (data.
↪column('day_of_week')==4).astype(int), 'fri', (data.
↪column('day_of_week')==5).astype(int), 'sat', (data.
↪column('day_of_week')==6).astype(int), 'sun', (data.
↪column('day_of_week')==7).astype(int))
data = data.with_column('weekend', (data.column('day_of_week')==6).
↪astype(int)+(data.column('day_of_week')==7).astype(int))
data
```

[100]:
```
day  | sales | day_of_week | mon  | tue  | wed  | thu  | fri  | sat  | sun  |
weekend
0    | 510   | 1           | 1    | 0    | 0    | 0    | 0    | 0    | 0    | 0
1    | 405   | 2           | 0    | 1    | 0    | 0    | 0    | 0    | 0    | 0
2    | 403   | 3           | 0    | 0    | 1    | 0    | 0    | 0    | 0    | 0
3    | 410   | 4           | 0    | 0    | 0    | 1    | 0    | 0    | 0    | 0
4    | 520   | 5           | 0    | 0    | 0    | 0    | 1    | 0    | 0    | 0
5    | 1010  | 6           | 0    | 0    | 0    | 0    | 0    | 1    | 0    | 1
6    | 1030  | 7           | 0    | 0    | 0    | 0    | 0    | 0    | 1    | 1
7    | 512   | 1           | 1    | 0    | 0    | 0    | 0    | 0    | 0    | 0
8    | 415   | 2           | 0    | 1    | 0    | 0    | 0    | 0    | 0    | 0
9    | 413   | 3           | 0    | 0    | 1    | 0    | 0    | 0    | 0    | 0
… (11 rows omitted)
```

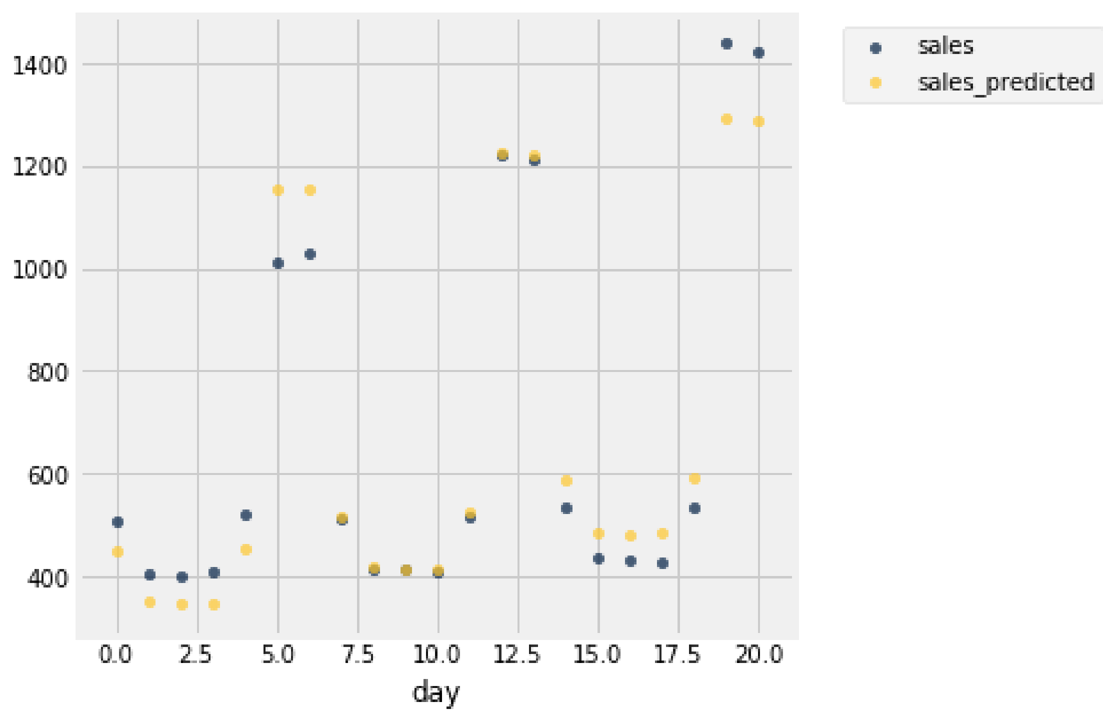
[114]:
```
# Build a linear regression model to predict sales based on day and day of week
# (for day of week, use only the dummy variables that you need).
# Show the model goodness of fit (R^2).
# Add variable to the dataset for predicted sales and show the first few␣
↪observations.
# Show the RMSE calculated based on the data.
# Visualize the model performance as a scatterplot of sales and predicted sales␣
↪vs. day.
model = ols('sales ~ day + mon + tue + wed + thu + fri + sat + sun', data).fit()
model.rsquared
data = data.with_column('sales_predicted', model.predict(data))
data
RMSE = sqrt(mean(model.resid**2))
RMSE
data.select('day', 'sales', 'sales_predicted').scatter('day')
```

[114]: 0.9603280537765004

[114]:
```
day  | sales | day_of_week | mon  | tue  | wed  | thu  | fri  | sat  | sun  |
weekend | sales_predicted | residual
0    | 510   | 1           | 1    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 451.786         | 3.1
```

```
1     | 405    | 2            | 0     | 1     | 0     | 0     | 0     | 0     | 0     | 0
| 352.119          | -2.23333
2     | 403    | 3            | 0     | 0     | 1     | 0     | 0     | 0     | 0     | 0
| 348.452          | -0.566667
3     | 410    | 4            | 0     | 0     | 0     | 1     | 0     | 0     | 0     | 0
| 349.452          | 5.43333
4     | 520    | 5            | 0     | 0     | 0     | 0     | 1     | 0     | 0     | 0
| 456.786          | 8.1
7     | 512    | 1            | 1     | 0     | 0     | 0     | 0     | 0     | 0     | 0
| 519              | -7
8     | 415    | 2            | 0     | 1     | 0     | 0     | 0     | 0     | 0     | 0
| 419.333          | -4.33333
9     | 413    | 3            | 0     | 0     | 1     | 0     | 0     | 0     | 0     | 0
| 415.667          | -2.66667
10    | 410    | 4            | 0     | 0     | 0     | 1     | 0     | 0     | 0     | 0
| 416.667          | -6.66667
11    | 517    | 5            | 0     | 0     | 0     | 0     | 1     | 0     | 0     | 0
| 524              | -7
… (11 rows omitted)
```

[114]: 71.45828746949203



2-Piece Multiple Linear Regression Model

[115]:
```
# Filter the dataset to include only NON-WEEKEND days.
# Build a linear regression model to predict sales based on day and day of week
# (for day of week, use only the dummy variables that you need).
# Show the model goodness of fit (R^2).
# Add variable to the (non-weekend) dataset for predicted sales and show the␣
↪first few observations.
data1 = data.where('weekend', 0).drop("sales_predicted")
model1 = ols('sales ~ day + mon + tue + wed + thu + fri', data1).fit()
model1.rsquared
data1 = data1.with_column("sales_predicted", model1.predict(data1))
data1
```

[115]: 0.9914697928111246

[115]:
```
day  | sales | day_of_week | mon  | tue  | wed  | thu  | fri  | sat  | sun  |
weekend | residual  | sales_predicted
0    | 510   | 1           | 1    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 3.1       | 506.9
1    | 405   | 2           | 0    | 1    | 0    | 0    | 0    | 0    | 0    | 0
| -2.23333  | 407.233
2    | 403   | 3           | 0    | 0    | 1    | 0    | 0    | 0    | 0    | 0
| -0.566667 | 403.567
3    | 410   | 4           | 0    | 0    | 0    | 1    | 0    | 0    | 0    | 0
| 5.43333   | 404.567
4    | 520   | 5           | 0    | 0    | 0    | 0    | 1    | 0    | 0    | 0
| 8.1       | 511.9
7    | 512   | 1           | 1    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| -7        | 519
8    | 415   | 2           | 0    | 1    | 0    | 0    | 0    | 0    | 0    | 0
| -4.33333  | 419.333
9    | 413   | 3           | 0    | 0    | 1    | 0    | 0    | 0    | 0    | 0
| -2.66667  | 415.667
10   | 410   | 4           | 0    | 0    | 0    | 1    | 0    | 0    | 0    | 0
| -6.66667  | 416.667
11   | 517   | 5           | 0    | 0    | 0    | 0    | 1    | 0    | 0    | 0
| -7        | 524
… (5 rows omitted)
```

[116]:
```
# Filter the dataset to include only WEEKEND days.
# Build a linear regression model to predict sales based on day and day of week
# (for day of week, use only the dummy variables that you need).
# Show the model goodness of fit (R^2).
# Add variable to the (weekend) dataset for predicted sales and show the first␣
↪few observations.
data2 = data.where('weekend', 1).drop("sales_predicted")
model2 = ols('sales ~ day + sat + sun', data2).fit()
model2.rsquared
```

```
data2 = data2.with_column("sales_predicted", model2.predict(data2))
data2
```

[116]: 0.9966406481572967

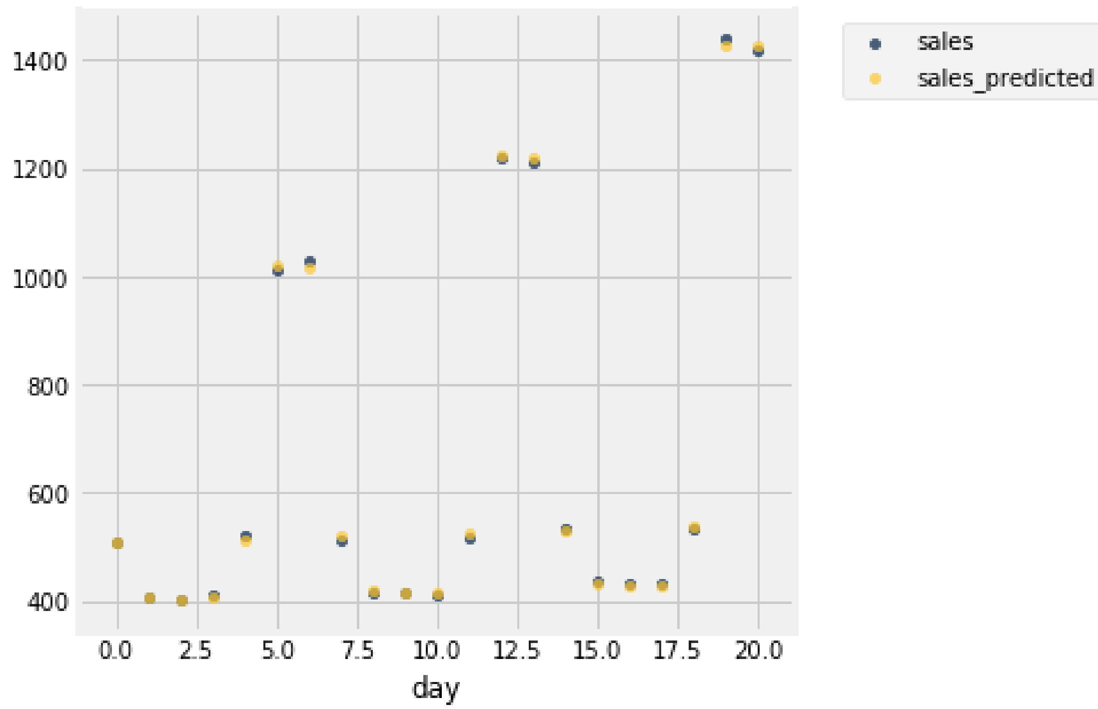
[116]:
```
day   | sales | day_of_week | mon   | tue   | wed   | thu   | fri   | sat   | sun   |
weekend | residual | sales_predicted
5     | 1010  | 6           | 0     | 0     | 0     | 0     | 0     | 1     | 0     | 1
| -8.33333 | 1018.33
6     | 1030  | 7           | 0     | 0     | 0     | 0     | 0     | 0     | 1     | 1
| 15       | 1015
12    | 1220  | 6           | 0     | 0     | 0     | 0     | 0     | 1     | 0     | 1
| -3.33333 | 1223.33
13    | 1210  | 7           | 0     | 0     | 0     | 0     | 0     | 0     | 1     | 1
| -10      | 1220
19    | 1440  | 6           | 0     | 0     | 0     | 0     | 0     | 1     | 0     | 1
| 11.6667  | 1428.33
20    | 1420  | 7           | 0     | 0     | 0     | 0     | 0     | 0     | 1     | 1
| -5       | 1425
```

[117]:
```
# Build a dataset that combines the non-weekend dataset and weekend dataset.
# Add a variable to the (combined) dataset for the residual and show the first␣
↪few observations.
data1 = data1.with_columns('sales_predicted', model1.predict(data1),␣
↪'residual', model1.resid)
data2 = data2.with_columns('sales_predicted', model2.predict(data2),␣
↪'residual', model2.resid)
data = data1.with_rows(data2.rows)
data.show()
```

```
<IPython.core.display.HTML object>
```

[118]:
```
# Show the RMSE calculated based on the dataset.
# Visualize the model performance as a scatterplot of sales and predicted sales␣
↪vs. day.
RMSE = sqrt(mean(model.resid**2))
RMSE
data.select('day', 'sales', 'sales_predicted').scatter('day')
```

[118]: 71.45828746949203



1.4.3 Discussion

Assume that you are the coffee shop owner. How would you estimate next week's sales? How confident would you be of your estimate?

What does your analysis tell you about how to make business decisions?

I would estimate my next weeks sales to be comparable to the pattern established by the data analysis above: where the bookends of the weekday period have higher sales compared to the rest of the group, the other days of the weekday being roughly equal, and have a huge boost of sales on the weekends. The data shows an overall upwards trend regarding weekend sales. I would be fairly confident of my estimate as the data shows a clear trend, and the model we established shows high accuracy.

My analysis tells me that I should design my models around the constraints/layout of the data in a way that makes the most sense. Here, we sorted the data according to the days of the week, and then further differentiated by weekday vs. weekend, which makes sense looking at the raw data. This allows us to obtain the most accurate results with our model and analysis.

 Document revised April 19, 2020

[ ]: