Scatter Plot

Learn how to create scatter plots comparing numerical features to sales.

Chapter Goals:

- Create a scatter plot to visualize the relationships between dataset features and weekly sales

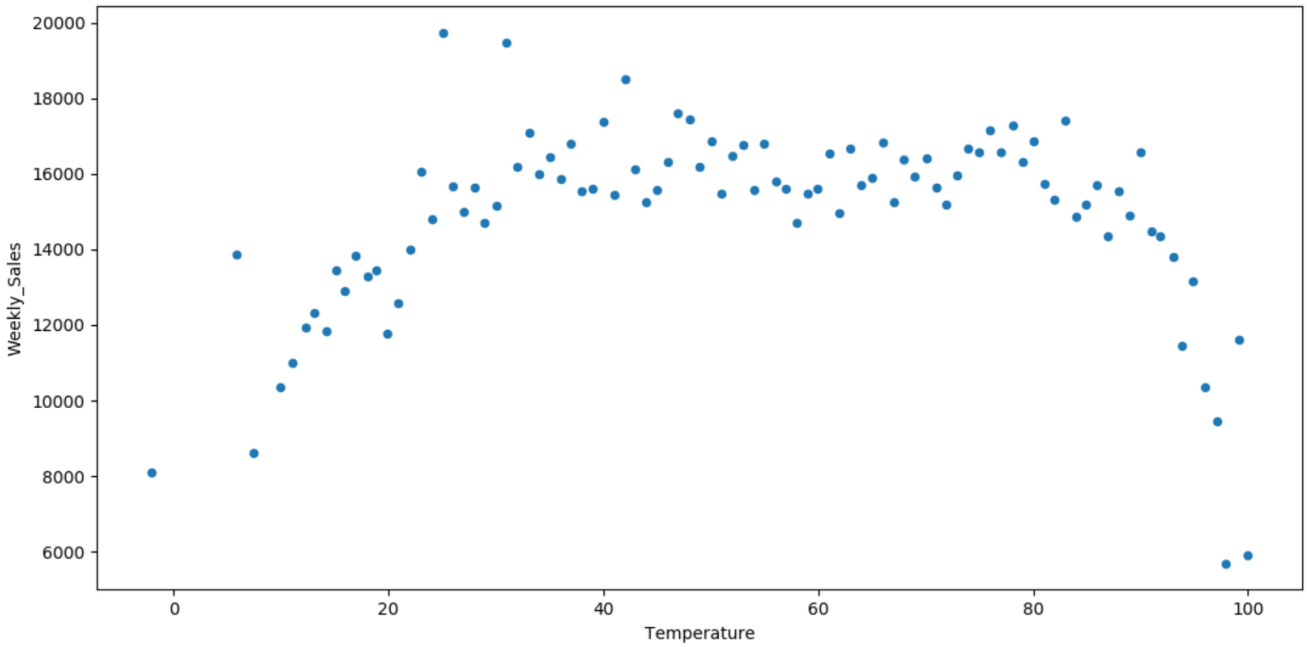
A. Plotting data

The most important part of interpreting data is being able to visualize trends and correlations between dataset features. We do this through data plots, which allows us to easily discover interesting patterns in the dataset and decide whether the dataset is feasible for training a machine learning model.

We create the data plots using the pyplot API of Matplotlib. One of the most common plots for data analysis is the 2-D scatter plot. It's used for plotting the relationship between a numeric dependent feature (Y-axis) and a numeric independent feature (X-axis). Since we're trying to predict weekly sales for stores, we'll make plots with the `'Weekly_Sales'` feature as the dependent feature.

```
import matplotlib.pyplot as plt

plot_df = final_dataset[['Weekly_Sales', 'Temperature']]
rounded_temp = plot_df['Temperature'].round(0)  # nearest integer
plot_df = plot_df.groupby(rounded_temp).mean()
plot_df.plot.scatter(x='Temperature', y='Weekly_Sales')
plt.show()
```



Creating a scatter plot with 'Temperature' as the independent feature and 'Weekly_Sales' as the dependent feature.

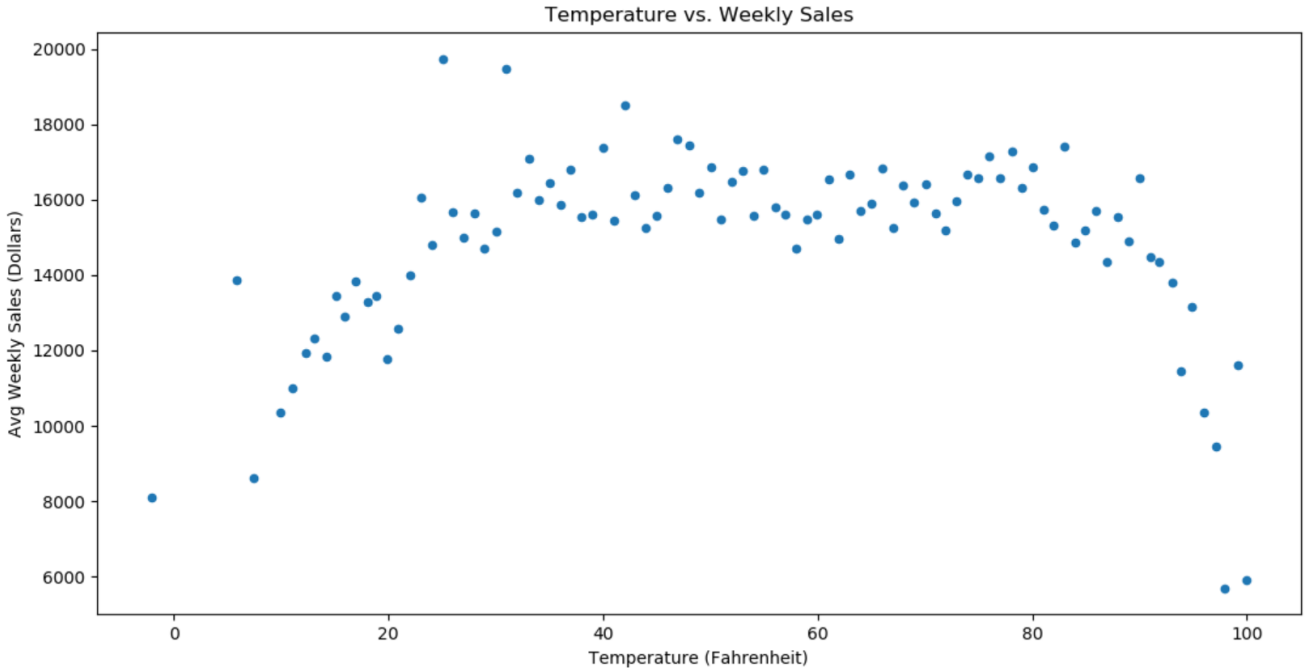
In the code, we rounded the temperature to the nearest integer and took the average weekly sales at each integer temperature value (degrees Fahrenheit).

We'll do more detailed analysis of the feature scatter plots in later chapters, but you can tell from the above scatter plot that sales are pretty consistent around $160,000 weekly, although they tend to drop at the high and low ends of the temperature spectrum.

For organizational purposes, and to present a more professional plot to the project supervisor, we should label and title the scatter plots appropriately.

```
import matplotlib.pyplot as plt

plot_df = final_dataset[['Weekly_Sales', 'Temperature']]
rounded_temp = plot_df['Temperature'].round(0)  # nearest integer
plot_df = plot_df.groupby(rounded_temp).mean()
plot_df.plot.scatter(x='Temperature', y='Weekly_Sales')
plt.title('Temperature vs. Weekly Sales')
plt.xlabel('Temperature (Fahrenheit)')
plt.ylabel('Avg Weekly Sales (Dollars)')
plt.show()
```



Labeling and titling the previous scatter plot. The plot_df variable contains the average weekly sales at each integer temperature value.