Analyse Customer Data and Recommend Products with PixieDust

Margriet Groenendijk - Developer Advocate
@MargrietGr

IBM Code London - 24 July 2018 - Skills Matter

https://ibm.biz/BdZCKW

CALL FOR CODE

- **Commit to the Cause**
  Get to know about “Call for Code”
  developer.ibm.com/callforcode
- **Push For Change**
  Register for the IBM Coder program
- **Weather Data**
  callforcode.weather.com

callforcode.org

ibm.biz/localcart

Python
Apache Spark
Jupyter notebooks
PixieDust
Watson Studio
Watson Machine Learning

Jupyter notebooks

Organize your:

- Notes
- Ideas
- Python code

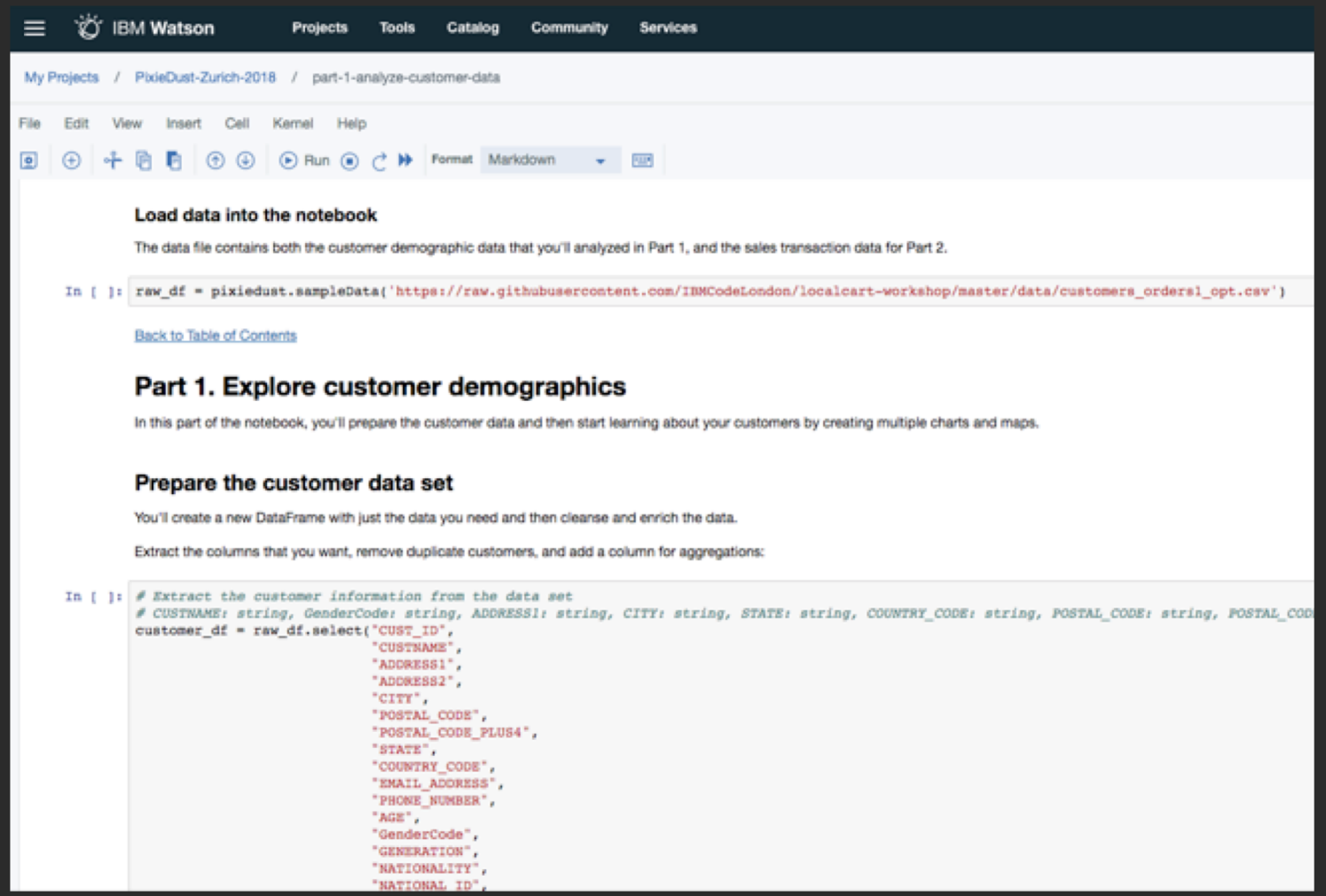
http://jupyter.org/

Install Python and Jupyter notebooks:

https://www.anaconda.com/distribution/

Or use the Cloud:

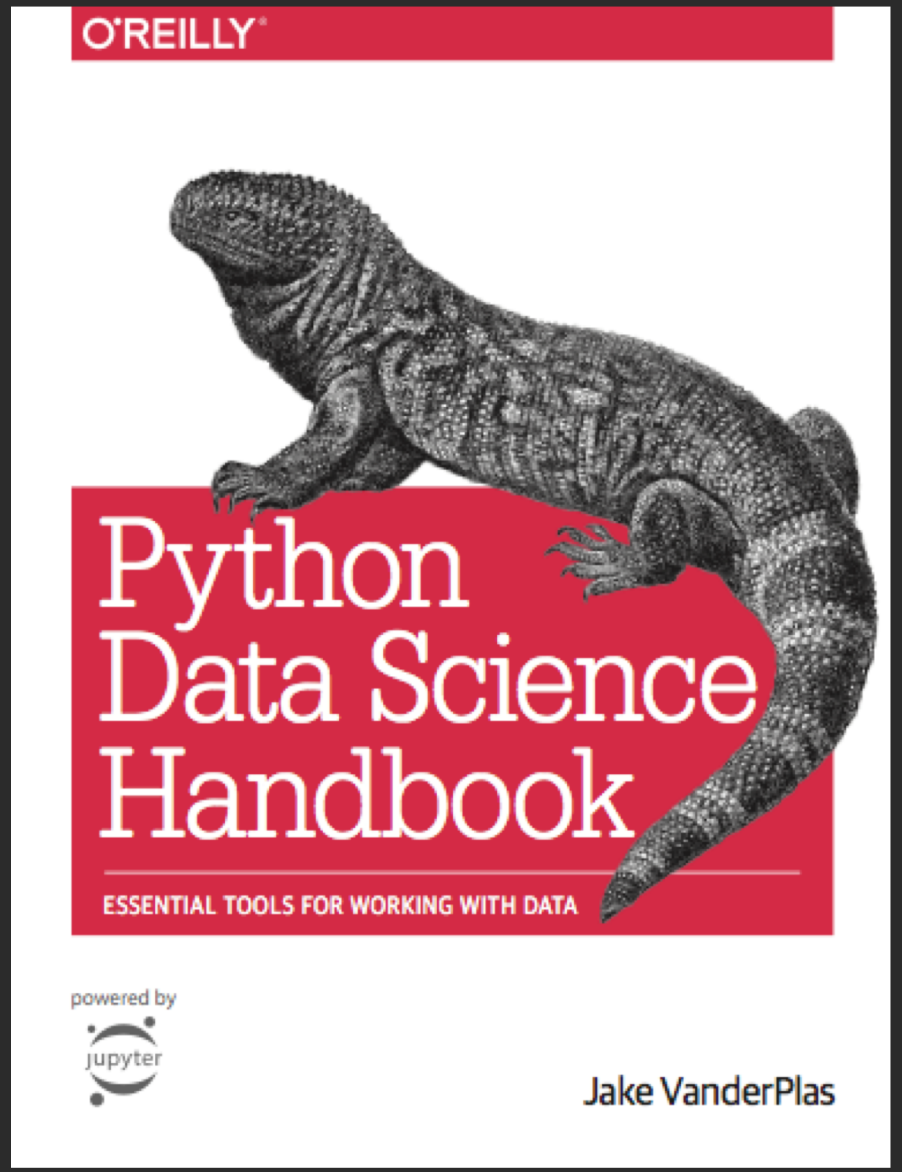
https://eu-gb.dataplatform.ibm.com/



Python

Used by many Data Scientists

https://jakevdp.github.io/PythonDataScienceHandbook/



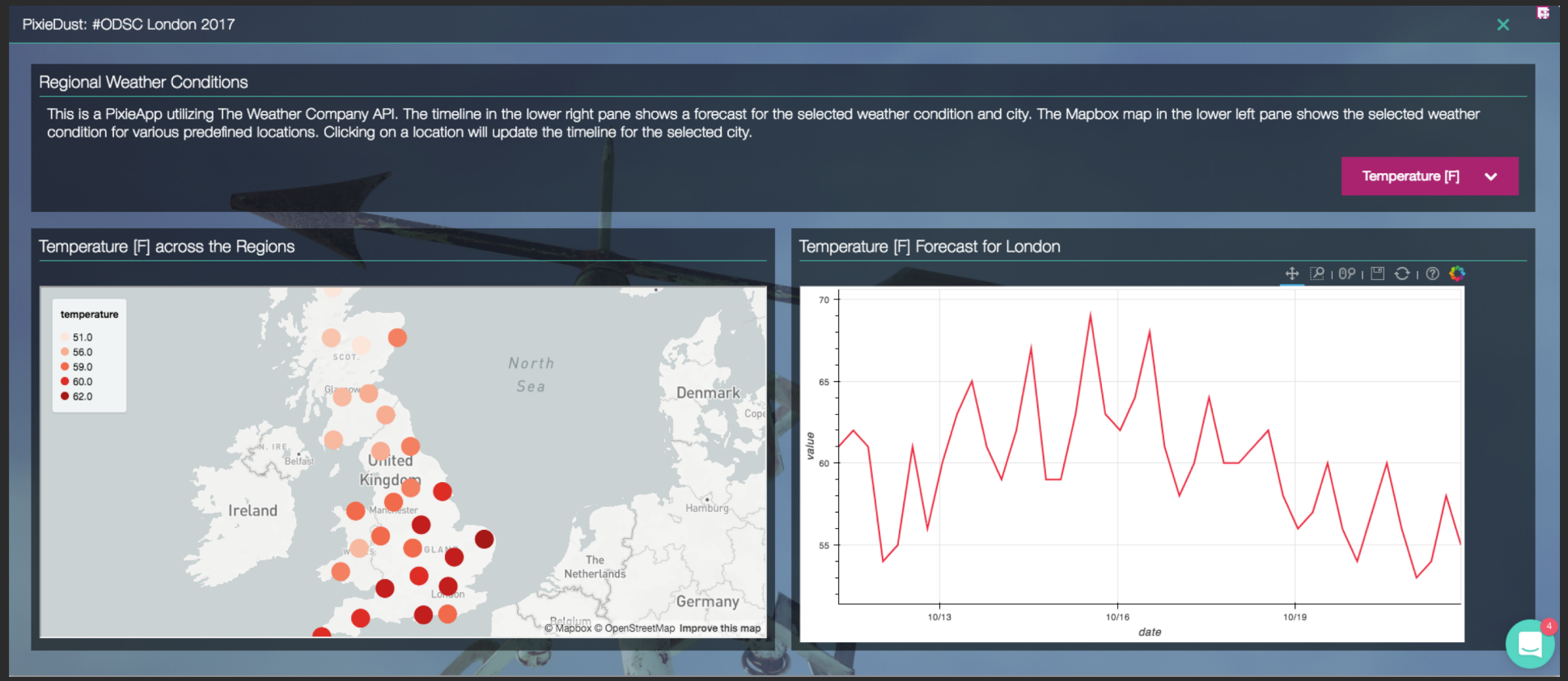
PixieDust

https://pixiedust.github.io/pixiedust/

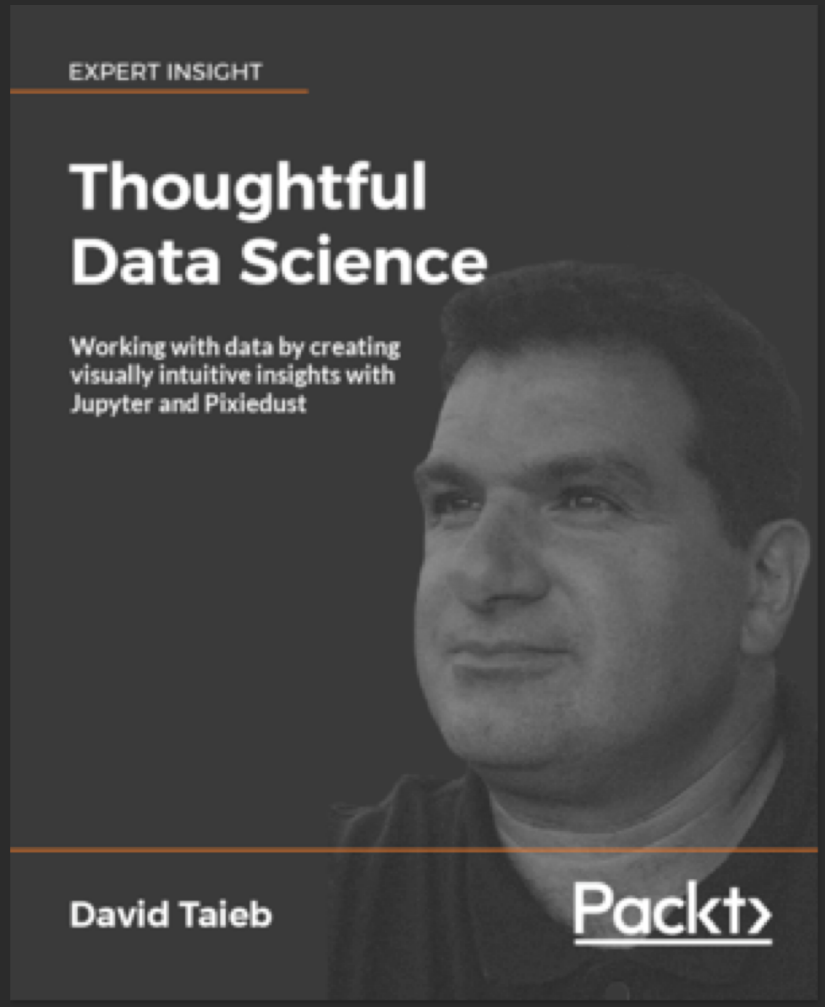
```
> display(df)
```

PixieApps

- HTML
- CSS
- Javascript
- Python
  - **@PixieApp**
  - **@route()**



https://www.packtpub.com/big-data-and-business-intelligence/thoughtful-data-science



Apache Spark - DataFrames

```
df = pixiedust.sampleData()

df = df.select()

df.filter()

df.join()
```

```
df.groupby().sum()

df.printSchema()

df.show()

df.cache()
```

Watson Studio

App in **IBM Cloud**

Sign up here: https://ibm.biz/BdZCKW

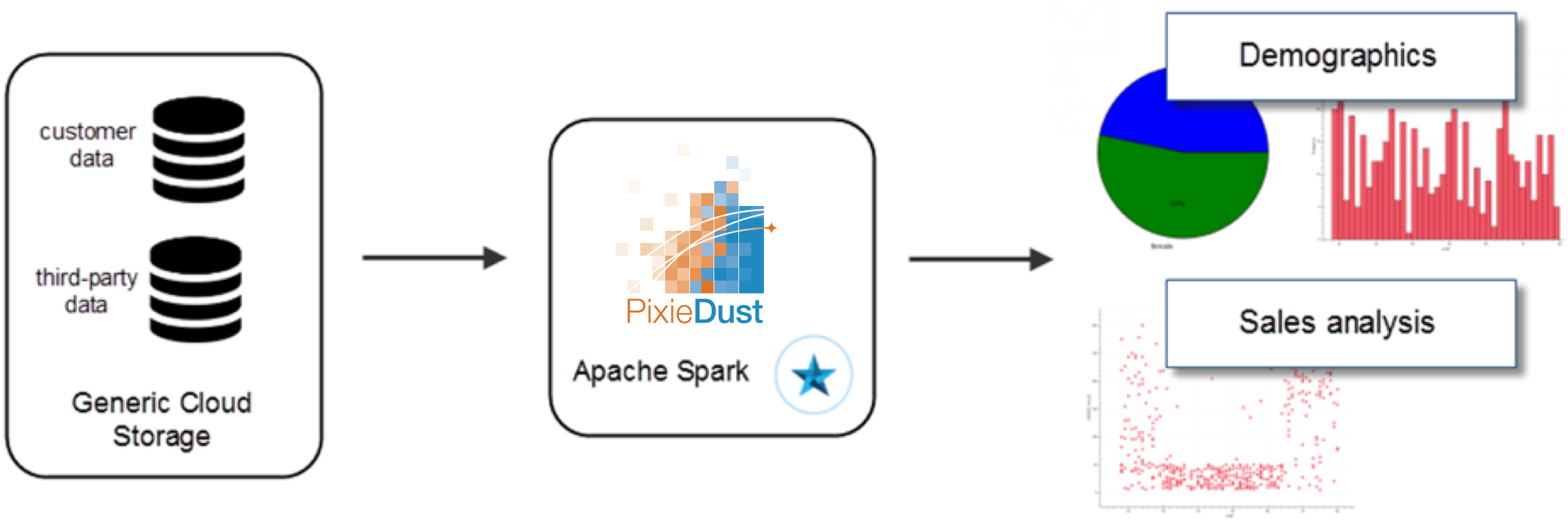
- **Projects**
- **Jupyter notebooks**
- RStudio
- **Python, Spark**, R and Scala
- Connections to data
  - Object Store and many other databases
  - Streams
- Models
  - Natural Language Classifiers
  - Visual Recognition Models
  - **Watson Machine Learning Models**
- Modeler Flows
- Data Flows
- Data Refinery

Watson Machine Learning

Train and deploy machine learning models and neural networks

Spark Machine Learning
Scikit-Learn
Tensorflow
Keras
Caffe

Part 1: Static data analysis using Python, Apache Spark and PixieDust



Part 2: Build a product recommendation engine

Recommender Engine

Customers who viewed this item also viewed these products

Dualit Food XL1500 Processor

$560

Add to cart

Kenwood kMix Manual Espresso Machine

★★★★☆

$250

Select options

Weber One Touch Gold Premium Charcoal Grill-57cm

$225

Add to cart

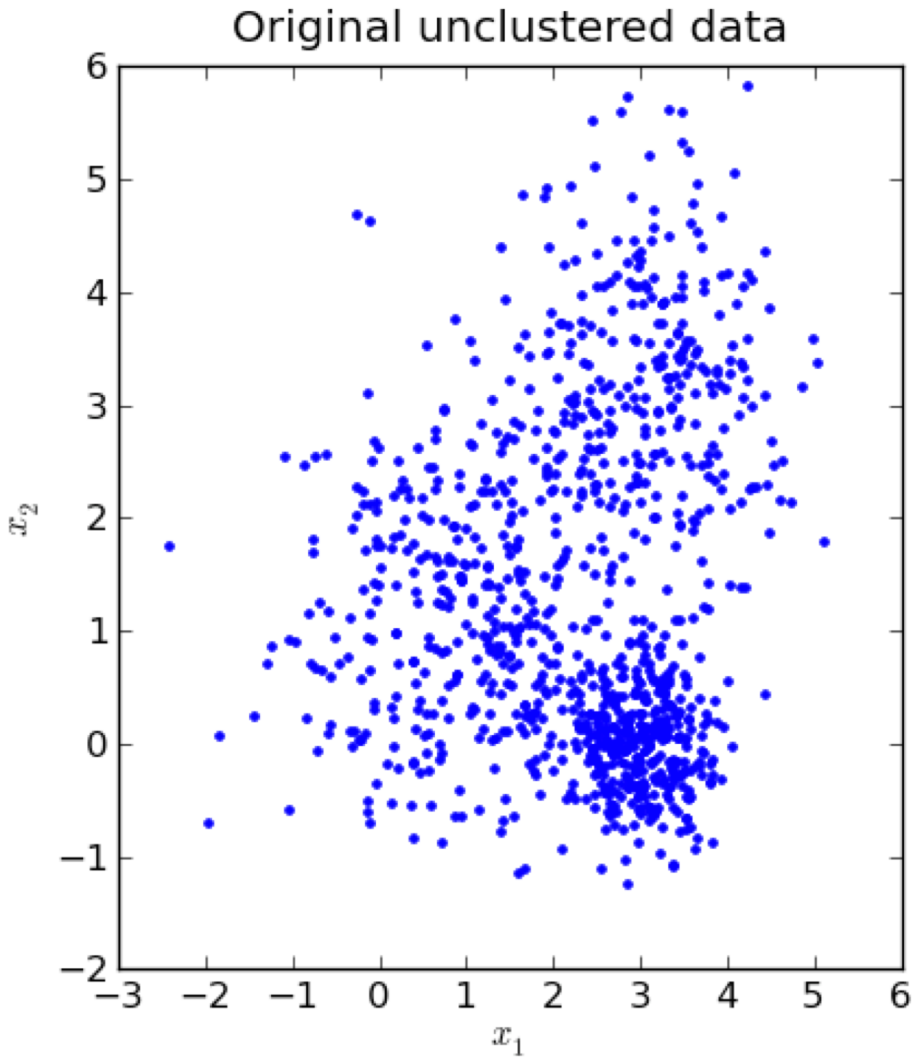
NoMU Salt Pepper and Spice Grinders

$3

View options

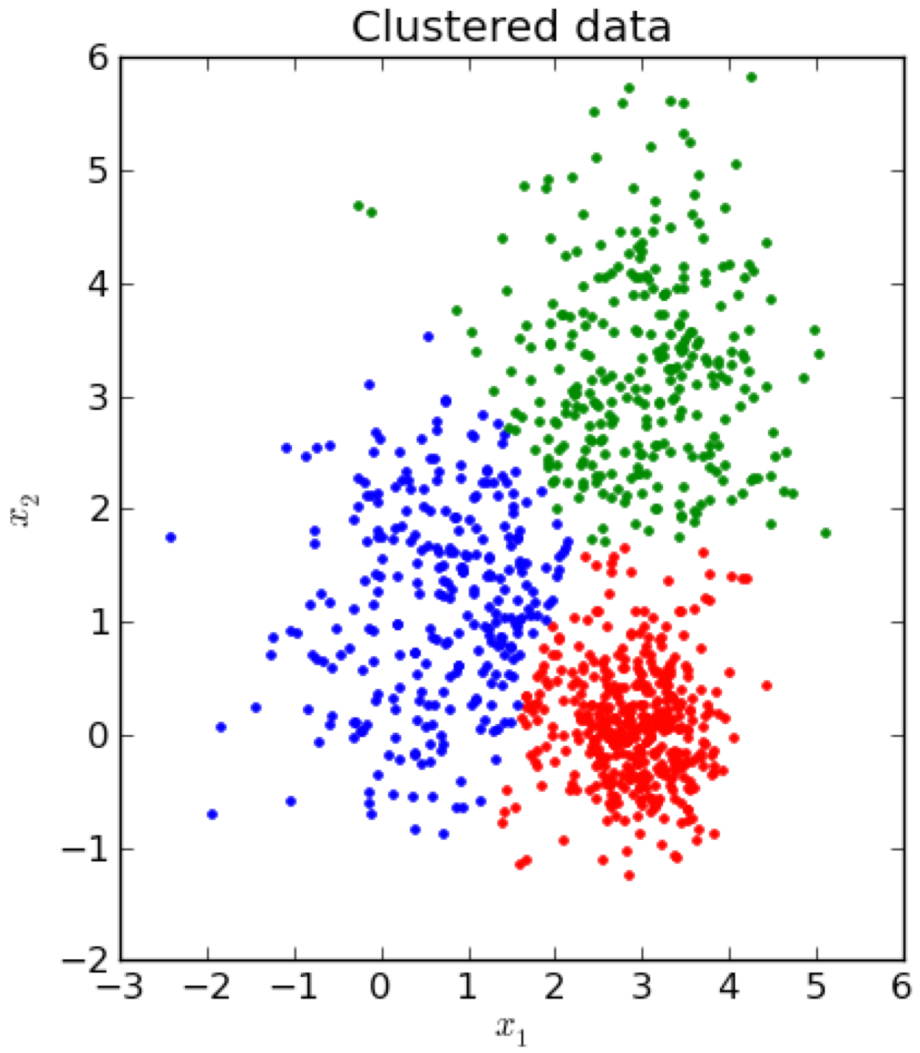
Recommender Engine

Every dot is a customer
The axes are the number of products A and B bought



Recommender Engine

Every dot is a customer
The axes are the number of products A and B bought
You can now use clustering algorithms to group these customers



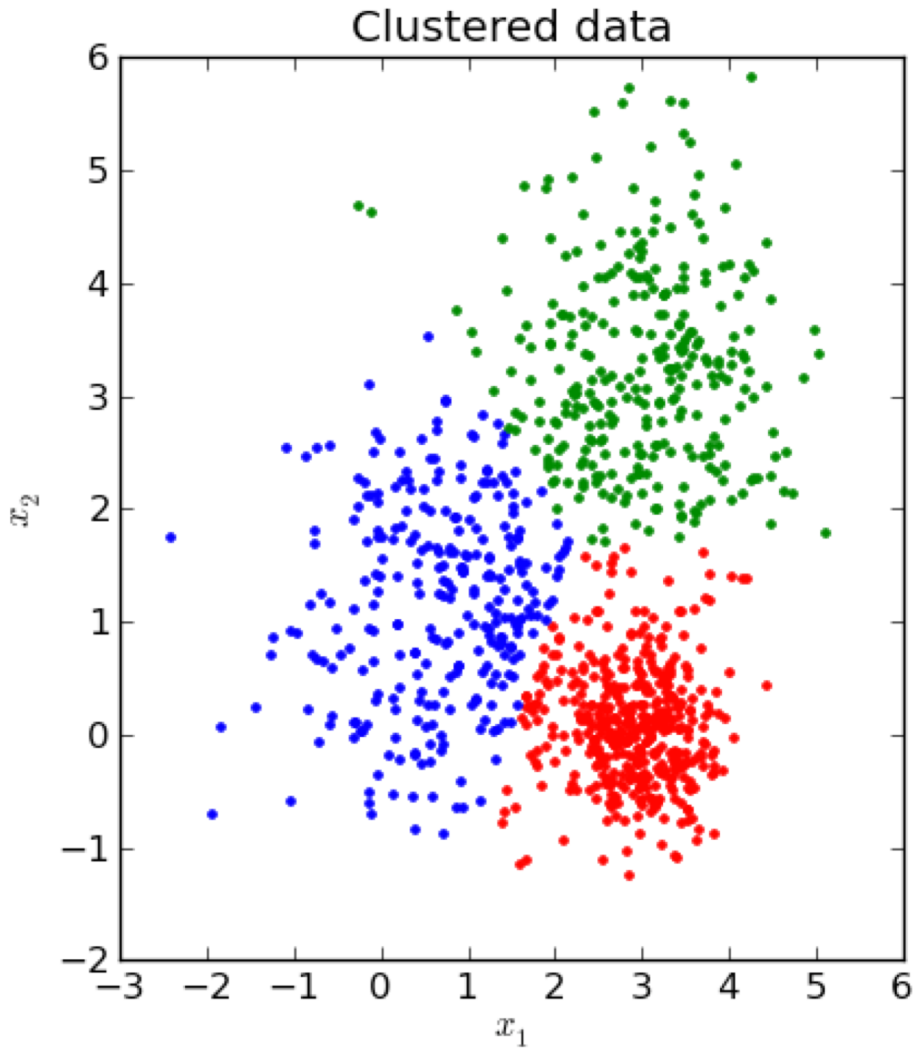
Recommender Engine

Every dot is a customer
The axes are the number of products A and B bought
You can now use clustering algorithms to group these customers

Imagine not 2 but thousands of products in a thousands dimensional space

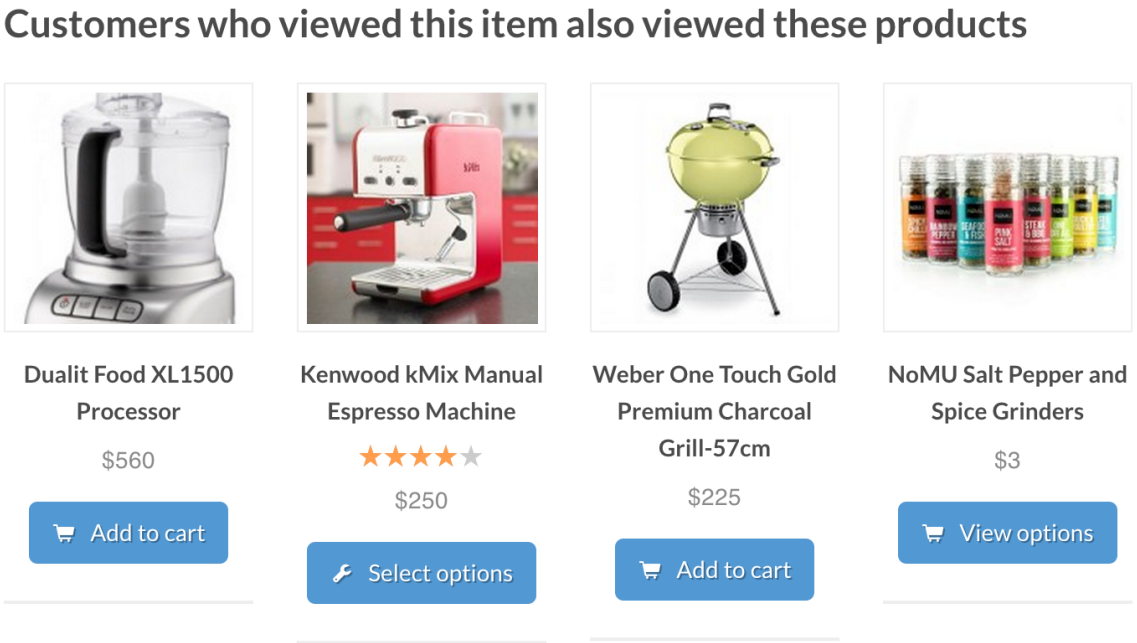
With machine learning algorithms you can still find clusters of customers who are similar

For example:
https://en.wikipedia.org/wiki/K-means_clustering
https://en.wikipedia.org/wiki/Cluster_analysis



Recommender Engine

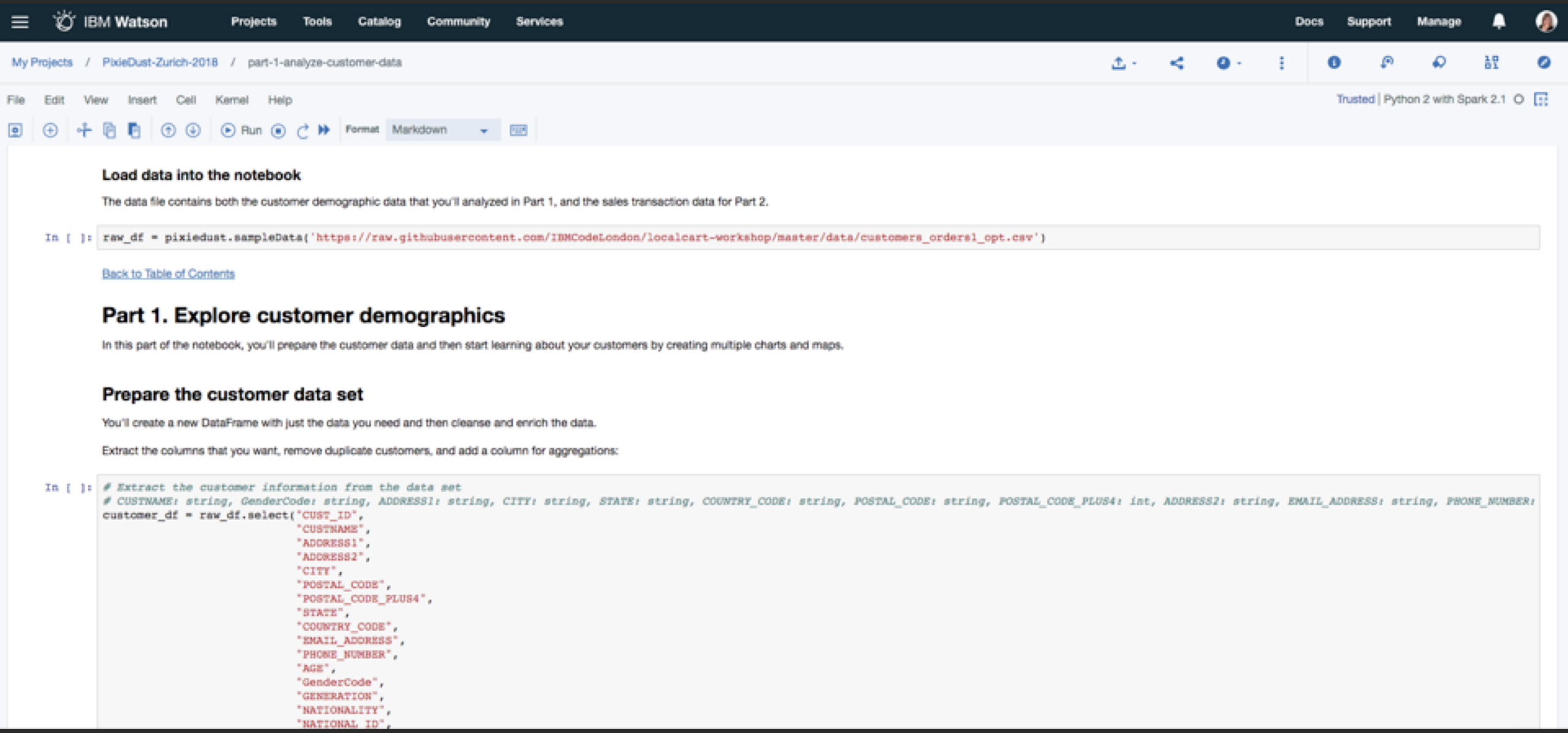
But how do you get from the clusters to a recommendation?



Recommender Engine

Recommend the most popular items in the cluster
But filter out the products already bought

```
# This function takes a cluster and the quantity of every product already purchased or in the user's cart
from pyspark.sql.functions import desc
def get_recommendations_by_cluster(cluster, purchased_quantities):
    # Existing customer products
    print('PRODUCTS ALREADY PURCHASED/IN CART:')
    customer_products = []
    for i in range(0, len(product_cols)):
        if purchased_quantities[i] > 0:
            customer_products.append((product_cols[i], purchased_quantities[i]))
    df_customer_products = sc.parallelize(customer_products).toDF(["PRODUCT","COUNT"])
    df_customer_products.show()
    # Get popular products in the cluster
    print('POPULAR PRODUCTS IN CLUSTER:')
    cluster_products = get_popular_products_in_cluster(cluster)
    df_cluster_products = sc.parallelize(cluster_products).toDF(["PRODUCT","COUNT"])
    df_cluster_products.show()
    # Filter out products the user has already purchased
    print('RECOMMENDED PRODUCTS:')
    df_recommended_products = df_cluster_products.alias('cl').join(df_customer_products.alias('cu'), df_clus
    df_recommended_products = df_recommended_products.filter('cu.PRODUCT IS NULL').select('cl.PRODUCT','cl.C
    df_recommended_products.show(10)
```


IBM Watson
Projects
Tools
Catalog
Community
Services
Docs
Support
Manage
My Projects / PixieDust-Zurich-2018 / part-1-analyze-customer-data
File Edit View Insert Cell Kernel Help
Trusted | Python 2 with Spark 2.1
Run
Format
Markdown
Load data into the notebook
The data file contains both the customer demographic data that you'll analyzed in Part 1, and the sales transaction data for Part 2.
In [ ]:
raw_df = pixiedust.sampleData('https://raw.githubusercontent.com/IBMCodeLondon/localcart-workshop/master/data/customers_orders1_opt.csv')
Back to Table of Contents
Part 1. Explore customer demographics
In this part of the notebook, you'll prepare the customer data and then start learning about your customers by creating multiple charts and maps.
Prepare the customer data set
You'll create a new DataFrame with just the data you need and then cleanse and enrich the data.
Extract the columns that you want, remove duplicate customers, and add a column for aggregations:
In [ ]:
# Extract the customer information from the data set
# CUSTNAME: string, GenderCode: string, ADDRESS1: string, CITY: string, STATE: string, COUNTRY_CODE: string, POSTAL_CODE: string, POSTAL_CODE_PLUS4: int, ADDRESS2: string, EMAIL_ADDRESS: string, PHONE_NUMBER:
customer_df = raw_df.select("CUST_ID",
"CUSTNAME",
"ADDRESS1",
"ADDRESS2",
"CITY",
"POSTAL_CODE",
"POSTAL_CODE_PLUS4",
"STATE",
"COUNTRY_CODE",
"EMAIL_ADDRESS",
"PHONE_NUMBER",
"AGE",
"GenderCode",
"GENERATION",
"NATIONALITY",
"NATIONAL_ID",

ibm.biz/localcart

Part 1 – PixieDust

- Load data
- Clean data
- Visualize data

Part 2 – Recommender

- Load data
- Clean data
- Cluster customer with a k-means model
- Deploy this model to WML
- Test the API
- Build an interactive PixieApp

Links

ibm.biz/localcart

Find us on Twitter:
@MargrietGr
@IBMCodeLondon

More examples:
https://developer.ibm.com/code/