tools for stem

Noushin Nabavi

2020-09-09

Contents

Tools and capabilities of data science is changing everyday!

Data can:
- Describe the current state of an organization or process - Detec anomalous events - Diagnose the causes of events and behaviors - Predict future events

Data Science workflows can be developed for:
- Data collection and management - Exploration and visualization - Experimentation and prediction

Applications of data science can include:
- Traditional machine learning: e.g. finding probabilities of events, labeled data, and algorithms - Deep learning: neurons work together for image and natural language recognition but requires more training data - Internet of things (IOT): e.g. smart watch algorithms to detect and analyze motion sensors

Data science teams can consist of:
- Data engineers: SQL, Java, Scala, Python - Data analysts: Dashboards, hy-

pothesis tests and visualization using spreadsheets, SQL, BI (Tableau, power BI, looker) - Machine learning scientists: predictions and extrapolations, classification, etc. and use R or python Data employees can be isolated, embedded, or hybrid.

Data use can come with risks of identification of personal information.
Policies for personally identifiable information may need to consider:
- sensitivity and caution - pseudonymization and anonymization

Preferences can be stated or revealed through the data so questions need to be specific, avoid loaded language, calibrate, require actionable results.

Data storage and retrieval may include:
- parallel storage solutions (e.g. cluster or server) - cloud storage (google, amazon, azure) - types of data: 1) unstructured (email, text, video, audio, web, and social media = document database); 2) structured = relational databases - Data querying: NoSQL and SQL

Communication of data can include:
- Dashboards - Markdowns - BI tools - rshiny or d3.js

Team management around data can use:
- Trello, slack, rocket chat, or JIRA to communicate due data and priority

A/B Testing:
- Control and Variation in samples - 4 steps in A/B testing: pick metric to track, calculate sample size, run the experiment, and check significance

Machine learning (ML) can be used for time series forecasting (investigate seasonality on any time scale), natural language processing (word count, word embeddings to create features that group similar words), neural networks, deep learning, and AI.
Learning can be classified into:
Supervised: labels and features/ Model evaluation on test and train data - recommendation systems - subscription predictions - email subject optimization
Unsupervised: unlabeled data with only features - clustering

Deep learning and AI requirements:
- prediction is more feasible than explanations - lots of very large amount of training data

Chapter 1

Introduction

Tools and capabilities of data science is changing everyday but it is simply making data work for you. Data can: - Describe the current state of an organization or process - Detec anomalous events - Diagnose the causes of events and behaviors - Predict future events

Data Science workflow includes: - Data collection - Exploration and visualization - Experimentation and prediction

Applications of data science: - Traditional machine learning: e.g. finding probabilities of events, labeled data, and algorithms - Deep learning: neurons work together for image and natural language recognition but requires more training data - Internet of things (IOT): e.g. smart watch algorithms to detect and analyze motion sensors

Building a data science team: - Data engineers: SQL, Java, Scala, Python - Data analysts: Dashboards, hypothesis tests and visualization using spreadsheets, SQL, BI (Tableau, power BI, looker) - Machine learning scientists: predictions and extrapolations, classification, etc. and use R or python Data employees can be isolated, embedded, or hybrid.

Data sources and risks [Data collection phase of data science workflow]: (Consider GDPR [General Data Protection Regulation] policies for personally identifiable information which needs to treated with sensitivity and caution through data pseudonymization and anonymization) - web events - customer data (solicited): surveys, reviews, focus groups, in-app questions - logistics data - financial transactions

Data can be qualitative or quantitative. Preferences can be stated or revealed through the data so questions need to be specific, avoid loaded language, calibrate, require actionable results.

Data storage and retrieval: - parallel storage solutions (e.g. cluster or server) -

cloud storage (google, amazon, azure) - types of data: 1) unstructured (email, text, video, audio, web, and social media = document database); 2) structured = relational databases - Data querying: NoSQL and SQL

Communication of data using: - Dashboards: - build using spreadsheets - BI tools - rshiny or d3.js

Ad hoc analysis - can use ticketing system such as Trello or JIRA to communicate due data and priority

A/B Testing: - Control and Variation in samples - 4 steps in A/B testing: pick metric to track, calculate sample size, run the experiment, and check significance

Machine learning:
Supervised: labels and features - recommendation systems - subscription predictions - email subject optimization

Chapter 2

Getting Set-Up with R & RStudio

Chapter 3

Introduction

Curated and organized by Noushin Nabavi, PhD.

Chapter 4

What is in a database?

- This set of exercises explores writing functions and stored procedures in SQL servers.

Close

4.1 explore data tables:

4.1.1 Select the count of the number of rows

```
SELECT count(*)
  FROM tablename;
```

4.1.2 counting missing data:

- Select the count of ticker,
- subtract from the total number of rows,
- and alias as missing

```
SELECT count(*) - count(ticker) AS missing
  FROM fortune500;
```

- Select the count of profits_change,
- subtract from total number of rows, and alias as missing

```
SELECT count(*) - count(profits_change) AS missing
  FROM fortune500;
```

- Select the count of industry,
- subtract from total number of rows, and alias as missing

```
SELECT count(*) - count(industry) AS missing
  FROM fortune500;
```

4.1.3 joining tables:

```
SELECT company.name
-- Table(s) to select from
  FROM company
       INNER JOIN fortune500
       ON company.ticker=fortune500.ticker;
```

4.2 The keys to the database (e.g. foreign vs. primary keys)

- Read an entity relationship diagram

```
-- Count the number of tags with each type
SELECT type, count(*) AS count
  FROM tag_type
 -- To get the count for each type, what do you need to do?
 GROUP BY type
 -- Order the results with the most common
 -- tag types listed first
 ORDER BY count DESC;
```

- or:

```
-- Select the 3 columns desired
SELECT name, tag_type.tag, tag_type.type
  FROM company
      -- Join the tag_company and company tables
       INNER JOIN tag_company
       ON company.id = tag_company.company_id
       -- Join the tag_type and company tables
       INNER JOIN tag_type
       ON tag_company.tag = tag_type.tag
  -- Filter to most common type
  WHERE type='cloud';
```

- coalesce function (to combine columns)

```
-- Use coalesce
SELECT coalesce(industry, sector, 'Unknown') AS industry2,
       -- Don't forget to count!
       count(*)
  FROM fortune500
-- Group by what? (What are you counting by?)
 GROUP BY industry2
-- Order results to see most common first
 ORDER BY count DESC
```

```
-- Limit results to get just the one value you want
 LIMIT 1;
```

- Coalesce with a self-join:

```
SELECT company_original.name, title, rank
  -- Start with original company information
  FROM company AS company_original
       -- Join to another copy of company with parent
       -- company information
   LEFT JOIN company AS company_parent
       ON company_original.parent_id = company_parent.id
       -- Join to fortune500, only keep rows that match
       INNER JOIN fortune500
       -- Use parent ticker if there is one,
       -- otherwise original ticker
       ON coalesce(company_parent.ticker,
                   company_original.ticker) =
            fortune500.ticker
 -- For clarity, order by rank
 ORDER BY rank;
```

4.2.1 Column types and constraints

- Effects of casting
- SELECT CAST(value AS new_type);

```
-- Select the original value
SELECT profits_change,
   -- Cast profits_change
       CAST(profits_change AS integer) AS profits_change_int
  FROM fortune500;
```

- SELECT and divide

```
-- Divide 10 by 3
SELECT 10/3,
       -- Divide 10 cast as numeric by 3
       10::numeric/3;
```

- SELECT value::new_type

```
SELECT '3.2'::numeric,
       '-123'::numeric,
       '1e3'::numeric,
       '1e-3'::numeric,
       '02314'::numeric,
       '0002'::numeric;
```

- Summarize the distribution of numeric values

```
-- Select the count of each value of revenues_change
SELECT revenues_change, count(*)
  FROM fortune500
 GROUP BY revenues_change
 -- order by the values of revenues_change
 ORDER BY revenues_change;
```

- Additional exploration syntax:

```
-- Count rows
SELECT count(*)
  FROM fortune500
 -- Where...
 WHERE revenues_change > 0;
```

4.3 Numeric data types and summary functions

4.3.1 Division

```
-- Select average revenue per employee by sector
SELECT sector,
       avg(revenues/employees::numeric) AS avg_rev_employee
  FROM fortune500
 GROUP BY sector
 -- Use the alias to order the results
 ORDER BY avg_rev_employee;
```

- explore by division:

```
-- Divide unanswered_count by question_count
SELECT unanswered_count/question_count::numeric AS computed_pct,
       -- What are you comparing the above quantity to?
       unanswered_pct
  FROM stackoverflow
 -- eliminate rows where question_count is not 0
 WHERE question_count != 0
 LIMIT 10;
```

Chapter 5

Following SQL functions DATEDIFF( ), DATENAME( ), DATEPART( ), CAST( ), CONVERT( ), GETDATE( ) and DATEADD( ) explore transactions per day

```
SELECT
  -- Select the date portion of StartDate
  CONVERT(DATE, StartDate) as StartDate,
  -- Measure how many records exist for each StartDate
  COUNT(ID) as CountOfRows
FROM CapitalBikeShare
-- Group by the date portion of StartDate
GROUP BY CONVERT(DATE, StartDate)
-- Sort the results by the date portion of StartDate
ORDER BY CONVERT(DATE, StartDate);
```

5.1 seconds or no?

- DATEDIFF() can be used to calculate the trip time by finding the difference between Start and End time
- Here, we will use DATEPART() to see how many transactions have seconds greater than zero and how many have them equal to zero

```
SELECT
-- Count the number of IDs
COUNT(ID) AS Count,
    -- Use DATEPART() to evaluate the SECOND part of StartDate
    "StartDate" = CASE WHEN DATEPART(SECOND, StartDate) = 0 THEN 'SECONDS = 0'
   WHEN DATEPART(SECOND, StartDate) > 0 THEN 'SECONDS > 0' END
FROM CapitalBikeShare
GROUP BY
    -- Complete the CASE statement
CASE WHEN DATEPART(SECOND, StartDate) = 0 THEN 'SECONDS = 0'
 WHEN DATEPART(SECOND, StartDate) > 0 THEN 'SECONDS > 0' END
```

- Which day of week is busiest?

```
SELECT
    -- Select the day of week value for StartDate
DATENAME(weekday, StartDate) as DayOfWeek,
    -- Calculate TotalTripHours
SUM(DATEDIFF(second, StartDate, EndDate))/ 3600 as TotalTripHours
FROM CapitalBikeShare
-- Group by the day of week
GROUP BY DATENAME(weekday, StartDate)
-- Order TotalTripHours in descending order
ORDER BY TotalTripHours DESC
```

- finding the outliers:

```
SELECT
-- Calculate TotalRideHours using SUM() and DATEDIFF()
   SUM(DATEDIFF(SECOND, StartDate, EndDate))/ 3600 AS TotalRideHours,
    -- Select the DATE portion of StartDate
   CONVERT(DATE, StartDate) AS DateOnly,
    -- Select the WEEKDAY
   DATENAME(WEEKDAY, CONVERT(DATE, StartDate)) AS DayOfWeek
FROM CapitalBikeShare
-- Only include Saturday
WHERE DATENAME(WEEKDAY, StartDate) = 'Saturday'
GROUP BY CONVERT(DATE, StartDate);
```

5.1.1 Variables for datetime data: storing data in variables

5.1.2 DECLARE & CAST

- use CapitalBikeShare table as starting point

```
-- Create @ShiftStartTime
DECLARE @ShiftStartTime AS time = '08:00 AM'

-- Create @StartDate
DECLARE @StartDate AS date

-- Set StartDate to the first StartDate from CapitalBikeShare
SET
@StartDate = (
     SELECT TOP 1 StartDate
     FROM CapitalBikeShare
     ORDER BY StartDate ASC
)

-- Create ShiftStartDateTime
DECLARE @ShiftStartDateTime AS datetime

-- Cast StartDate and ShiftStartTime to datetime data types
SET @ShiftStartDateTime = CAST(@StartDate AS datetime) + CAST(@ShiftStartTime AS datetime)

SELECT @ShiftStartDateTime
```

- DECLARE a TABLE:

```
-- Create @Shifts
DECLARE @Shifts TABLE(
    -- Create StartDateTime column
StartDateTime datetime,
    -- Create EndDateTime column
EndDateTime datetime)
-- Populate @Shifts
INSERT INTO @Shifts (StartDateTime, EndDateTime)
SELECT '3/1/2018 8:00 AM', '3/1/2018 4:00 PM'
SELECT *
FROM @Shifts
```

- INSERT INTO **?** based on CapitalBikeShare table:

```
-- Create @RideDates
DECLARE @RideDates TABLE(
    -- Create RideStart
RideStart date,
    -- Create RideEnd
```

```
RideEnd date)
-- Populate @RideDates
INSERT INTO @RideDates(RideStart, RideEnd)
-- Select the unique date values of StartDate and EndDate
SELECT DISTINCT
    -- Cast StartDate as date
CAST(StartDate as date),
    -- Cast EndDate as date
CAST(EndDate as date)
FROM CapitalBikeShare
SELECT *
FROM @RideDates;
```

5.1.3 Date manipulation

- First day of month:

```
-- Find the first day of the current month
SELECT DATEADD(month, DATEDIFF(month, 0, GETDATE()), 0)
-- Or
SELECT DATEDIFF(month, 0, GETDATE()), 0)
-- Or
SELECT DATEDIFF(year, '12/31/2017', '1/1/2019')
-- Or for yesterday use -1
WHERE CAST(year as date) = DATEADD (d, -1, GETDATE())
```

- What was yesterday? Creating a function that returns yesterday's date

```
-- Create GetYesterday()
CREATE FUNCTION GetYesterday()
-- Specify return data type
RETURNS date
AS
BEGIN
-- Calculate yesterday's date value
RETURN(SELECT DATEADD(day, -1, GETDATE()))
END
```

- 1 input/output
- Create a function named SumRideHrsSingleDay() which returns the total ride time in hours for the **?** parameter passed.

```
-- Create SumRideHrsSingleDay
CREATE FUNCTION SumRideHrsSingleDay (@DateParm date)
-- Specify return data type
RETURNS numeric
AS
```

```
-- Begin
BEGIN
RETURN
-- Add the difference between StartDate and EndDate
(SELECT SUM(DATEDIFF(second, StartDate, EndDate))/3600
FROM CapitalBikeShare
 -- Only include transactions where StartDate = @DateParm
WHERE CAST(StartDate AS date) = @DateParm)
-- End
END
```

- Multiple inputs/outputs
- Create a function that accepts both StartDate and EndDate then returns the total ride hours for all transactions that occur within the parameter values.

```
-- Create the function
CREATE FUNCTION SumRideHrsDateRange (@StartDateParm datetime, @EndDateParm datetime)
-- Specify return data type
RETURNS numeric
AS
BEGIN
RETURN
-- Sum the difference between StartDate and EndDate
(SELECT SUM(DATEDIFF(second, StartDate, EndDate))/3600
FROM CapitalBikeShare
-- Include only the relevant transactions
WHERE StartDate > @StartDateParm and StartDate <@EndDateParm)
END
```

5.1.4 User defined functions: inline (faster) and multi-statement value functions (slower)

- Inline value function:

```
-- Create the function
CREATE FUNCTION SumStationStats(@StartDate AS datetime)
-- Specify return data type
RETURNS TABLE
AS
RETURN
SELECT
StartStation,
    -- Use COUNT() to select RideCount
COUNT(ID) as RideCount,
    -- Use SUM() to calculate TotalDuration
```

```
    SUM(DURATION) as TotalDuration
FROM CapitalBikeShare
WHERE CAST(StartDate as Date) = @StartDate
-- Group by StartStation
GROUP BY StartStation;
```

- Multi statement value function

```
-- Create the function
CREATE FUNCTION CountTripAvgDuration (@Month CHAR(2), @Year CHAR(4))
-- Specify return variable
RETURNS @DailyTripStats TABLE(
TripDate date,
TripCount int,
AvgDuration numeric)
AS
BEGIN
-- Insert query results into @DailyTripStats
INSERT @DailyTripStats
SELECT
    -- Cast StartDate as a date
CAST(StartDate AS date),
    COUNT(ID),
    AVG(Duration)
FROM CapitalBikeShare
WHERE
DATEPART(month, StartDate) = @Month AND
    DATEPART(year, StartDate) = @Year
-- Group by StartDate as a date
GROUP BY CAST(StartDate AS date)
-- Return
RETURN
END
```

5.1.5 User defined functions in action: i.e. execute functions

- can use SELECT to execute scalar functions

```
-- Create @BeginDate
DECLARE @BeginDate AS date = '3/1/2018'
-- Create @EndDate
DECLARE @EndDate AS date = '3/10/2018'
SELECT
  -- Select @BeginDate
```

```
  @BeginDate AS BeginDate,
  -- Select @EndDate
  @EndDate AS EndDate,
  -- Execute SumRideHrsDateRange()
  dbo.SumRideHrsDateRange(@BeginDate, @EndDate) AS TotalRideHrs
```

- anotehr example:

```
-- Create @RideHrs
DECLARE @RideHrs AS numeric
-- Execute SumRideHrsSingleDay()
EXEC @RideHrs = dbo.SumRideHrsSingleDay @DateParm = '3/5/2018'
SELECT
  'Total Ride Hours for 3/5/2018:',
  @RideHrs
```

- Execute TVF into variable:

```
-- Create @StationStats
DECLARE @StationStats TABLE(
StartStation nvarchar(100),
RideCount int,
TotalDuration numeric)
-- Populate @StationStats with the results of the function
INSERT INTO @StationStats
SELECT TOP 10 *
-- Execute SumStationStats with 3/15/2018
FROM dbo.SumStationStats ('3/15/2018')
ORDER BY RideCount DESC
-- Select all the records from @StationStats
SELECT *
FROM @StationStats
```

Chapter 6

Final Words

We have finished a nice book.