

Web access and behavior analysis:

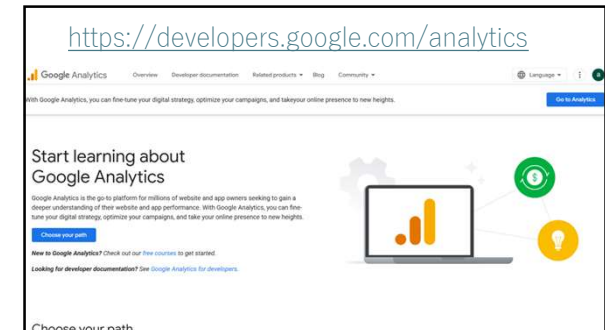
Kohei Arai

1

2. Analytical methods for understanding user behavior

- Google Analytics/Access Analysis: Analyze basic indicators such as the number of visitors to a website, the duration of their visit, and the source of traffic.
- Page View Analysis: Understand which pages are being viewed and the transition patterns between pages.
- Keyword Analysis: Analyze what search terms users use to visit the site, and use this information for SEO strategies.
- Heat Map Analysis: Visually display which parts of a page users are paying attention to, and use this information to improve UI/UX.
- Real-time Analysis (Google Analytics Reporting API): Understand the behavior of users currently visiting the site in real time.

2



3

Google Analytics

- Google Analytics data is usually obtained using the Google Analytics API
- To use the API, you need to create a project in Google Cloud Platform and obtain an API key or OAuth credentials
- A basic example of how to get data from the Google Analytics API using Python uses the google-analytics-data library
- First, install the required package:
pip install --upgrade google-analytics-data

4

Google Analytics

- Use code to get Google Analytics data
- Use the Google Analytics Data API to get data for a specified period from a specified property (such as a site or app) and view (a specific view ID) → However, you must set the actual values, such as the property ID and view ID, appropriately.
- Before running this code, you must create a project on Google Cloud Platform, enable the API, and obtain authentication information (API key or OAuth credentials).

5

Google Analytics Data API

- This code tries to get data using the Google Analytics Data API, but it requires Google Cloud Platform setup and authentication information to run. Since it cannot be run in a real environment, we will explain the code and provide a sample of the expected output.
- The Python code uses the Google Analytics Data API (GA4) to try to get the number of active users by date in 2022. To run this code, you need Google Cloud Platform settings and credentials, so the following steps are required in your actual environment.

6

GA4

- Initialize a client for the Google Analytics Data API (GA4)
- Set the ID of the Google Analytics property and view you want to access
- Create a request to get the number of active users by date in 2022Execute the API request and get the results
- View report data

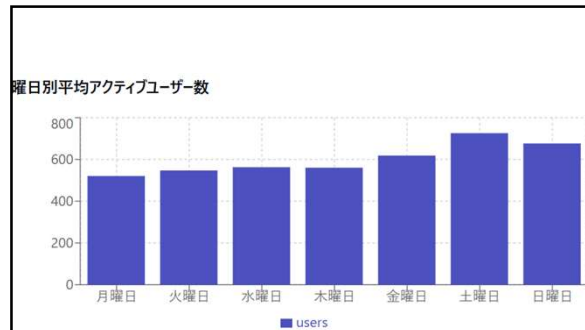
7

- Google Analytics User Statistics (2022)
- Total active users215,463
- Average daily users590.3
- Maximum number of users832 (12/24)

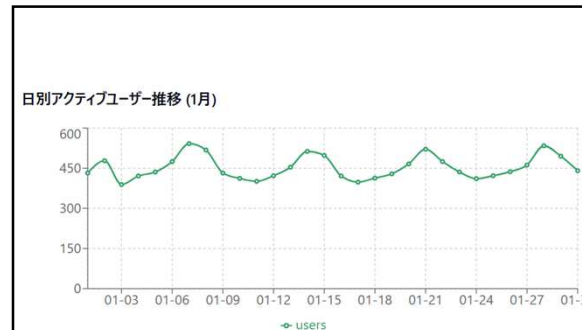
8



9



10



11

- Analysis results (based on simulated data)
- Monthly trends: Most users visit at the end of the year (November, December)
- A temporary drop is seen in August
- A gradual upward trend from January to June
- Weekday trends: Saturday is the most active (average 725.7 people) Sunday is the next most active (average 676.4 people) Monday is the least active weekday (average 520.4 people)
- Daily trends: Regular peaks on weekends
- Relatively sluggish from Monday to Wednesday

12

Points to note when using the actual API

- Create a project on Google Cloud Platform and enable the Google Analytics Data API
- Set appropriate authentication information (such as a service account key)
- Replace YOUR_PROPERTY_ID and YOUR_VIEW_ID in the code with actual values
- Pay attention to API usage limits and charges
- For GA4, API specifications may differ (e.g., only propertyId is required, not viewId)
- The graph visualization above is a representation based on simulated data, and the trends and patterns of data obtained from the actual API may differ.

13

Access analysis

- Access analysis is the analysis of data about user access to a website or application.
- Typically, this includes user sessions, page views, and user attributes.
- An example of basic access analysis using Python uses pandas, matplotlib, and seaborn.

14

What you can find out with access analysis:

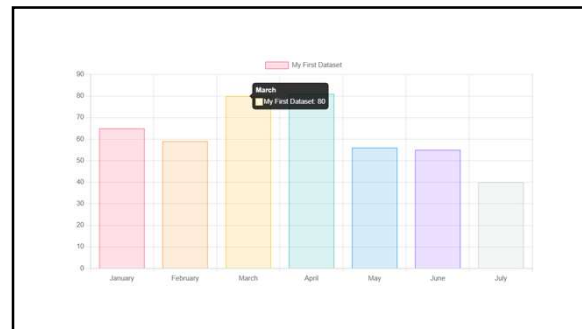
- User behavior: You can find out which pages are viewed most often, which pages have the most dropouts, and how users move around the site.
- User attributes: You can find out information such as the user's age, gender, region, and device used.
- Inflow path: You can find out where the user came to the site (search engine, SNS, other sites, etc.).
- Conversion: You can find out the percentage of users who achieved the site's goals (purchase, inquiry, etc.).

15

A bar plot

- A bar plot showing the number of sessions by browser is also created.
- This allows analysis based on web access trends and specific attributes.
- Depending on the actual data, additional information or specific analysis may be required → Access analysis is broad and diverse, so analysis is performed from various perspectives according to the characteristics of the data.

16



17

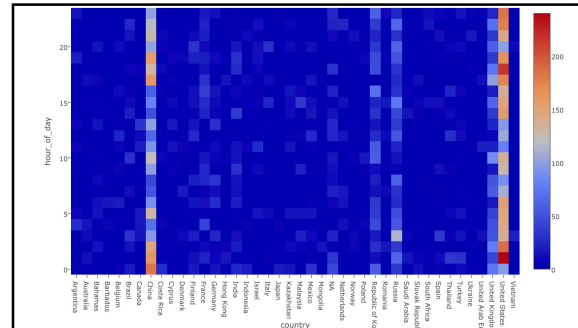
```
const config = {
  type: 'bar',
  data: data,
  options: {
    scales: {
      y: {
        beginAtZero: true
      }
    }
  },
};
```

18

Access analysis

- To perform access analysis, it is necessary to use appropriate libraries and tools according to the specific data and purpose.
- Typical access analysis tools include **Google Analytics**.
- Easy access analysis using Python
- Create a line graph using access data by date

19



20

Google Analytics API

- In actual access analysis, it is common to use data obtained from Google Analytics or other web analysis tools.
- When using Python, use libraries such as Pandas, Matplotlib, and Seaborn to organize and visualize data.
- If you want to obtain and analyze Google Analytics data using Python, you can use an approach such as the **Google Analytics API** (prior configuration and authentication are required to use the API).

21

Access analysis

- Displays session statistics (average, standard deviation, etc.) and creates a time series plot of the number of page views.
- Also creates a bar plot showing the number of sessions by browser.
- This allows analysis based on web access trends and specific attributes.
- Depending on the actual data, additional information or specific analysis may be required → Access analysis is broad and diverse, so analysis is performed from various perspectives according to the characteristics of the data.

22

This code analyzes access logs that contain timestamps, session information, browser information, etc.

```
#Access analysis
#Import required libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
#Read data (specify appropriate data)
data = pd.read_csv('your_access_data.csv')
#Display basic information of data
print(data.info())
#Session statistics
session_stats = data['SessionDuration'].describe()
print(session_stats)
```

23

```
#Time series plot of page views
data['Timestamp'] = pd.to_datetime(data['Timestamp'])
data.set_index('Timestamp', inplace=True)
daily_pageviews = data.resample('D').size()
daily_pageviews.plot(figsize=(12, 6), title='Daily Pageviews')
plt.xlabel('Date')
plt.ylabel('Pageviews')
plt.show()
```

24

```
# Analysis based on user attributes (e.g., number of sessions by browser)
browser_session_counts = data.groupby('Browser')['SessionID'].nunique().sort_values(ascending=False)
plt.figure(figsize=(10, 6))
sns.barplot(x=browser_session_counts.index, y=browser_session_counts.values, palette='viridis')
plt.title('Session Counts by Browser')
plt.xlabel('Browser')
plt.ylabel('Session Counts')
plt.xticks(rotation=45)
plt.show()
```

25

The file contains the following columns and can be analyzed with the provided Python code:

- 1.SessionID: Unique identifier for the session
- 2.Timestamp: Access date and time (used in the code for time series analysis)
- 3.UserID: User identifier
- 4.PageURL: URL of the accessed page
- 5.Browser: Browser type (e.g. Chrome, used in the code for grouping)
- 6.Device: Device used (Desktop, Mobile, Tablet, etc.)
- 7.Country: Country from which the access originated
- 8.SessionDuration: Session duration (in seconds, used in the code for statistical analysis)
- 9.PageViews: Number of pages viewed
- 10.BounceRate: Bounce rate (the closer to 1, the higher the bounce rate)

26

The following analysis are capable

- Display basic data information (info())
- Session duration statistics (describe())
- Time series plot of daily page views
- Bar graph of number of sessions by browser

27

```
import React from 'react';
import { LineChart, Line, BarChart, Bar, XAxis, YAxis, CartesianGrid, Tooltip,
Legend, ResponsiveContainer } from 'recharts';

export default function AccessDataVisualization() {
  // Data for the number of daily page view
  const dailyPageviewsData = [
    { name: '2025-04-01', pageviews: 6 },
    { name: '2025-04-02', pageviews: 6 },
    { name: '2025-04-03', pageviews: 6 },
    { name: '2025-04-04', pageviews: 6 },
    { name: '2025-04-05', pageviews: 6 },
    { name: '2025-04-06', pageviews: 0 }
  ];
}
```

28

```
// The data of the number of sessions for each browser
const browserSessionData = [
  { name: 'Chrome', sessions: 13 },
  { name: 'Firefox', sessions: 6 },
  { name: 'Safari', sessions: 5 },
  { name: 'Edge', sessions: 6 }
];

// The data of the number of sessions for each device type
const deviceSessionData = [
  { name: 'Desktop', sessions: 13 },
  { name: 'Mobile', sessions: 11 },
  { name: 'Tablet', sessions: 6 }
];
```

29

```
// The data of the number of sessions for each country
const countrySessionData = [
  { name: 'Japan', sessions: 6 },
  { name: 'USA', sessions: 6 },
  { name: 'UK', sessions: 5 },
  { name: 'Canada', sessions: 2 },
  { name: 'France', sessions: 2 },
  { name: 'Australia', sessions: 2 },
  { name: 'Germany', sessions: 2 },
  { name: 'Spain', sessions: 2 },
  { name: 'Italy', sessions: 2 },
  { name: 'China', sessions: 1 }
].slice(0, 6); // displayed for the top 6 countries
```

30

```
return (
  <div className="flex flex-col space-y-4">
    <div className="bg-white p-4 rounded-lg shadow">
      <h2 className="text-lg font-bold mb-4">日別ページビュー数</h2>
      <div className="h-64">
        <ResponsiveContainer width="100%" height="100%">
          <LineChart
            data={dailyPageviewsData}
            margin={{ top: 5, right: 30, left: 20, bottom: 5 }}
          >
            <CartesianGrid strokeDasharray="3 3" />
            <XAxis dataKey="name" />
            <YAxis />
            <Tooltip />
            <Legend />
            <Line type="monotone" dataKey="pageviews" stroke="#8884d8" activeDot={true} />
          </LineChart>
        </ResponsiveContainer>
      </div>
    </div>
  </div>
);
```

31

```
<div className="bg-white p-4 rounded-lg shadow">
  <h2 className="text-lg font-bold mb-4">ブラウザごとのセッション数</h2>
  <div className="h-64">
    <ResponsiveContainer width="100%" height="100%">
      <BarChart
        data={browserSessionData}
        margin={{ top: 5, right: 30, left: 20, bottom: 5 }}
      >
        <CartesianGrid strokeDasharray="3 3" />
        <XAxis dataKey="name" />
        <YAxis />
        <Tooltip />
        <Legend />
        <Bar dataKey="sessions" fill="#82ca9d" />
      </BarChart>
    </ResponsiveContainer>
  </div>
</div>
```

32

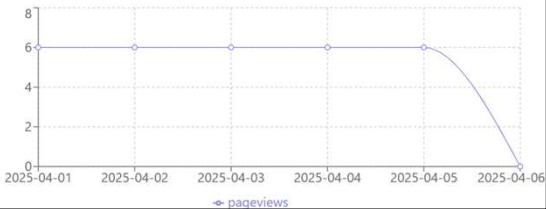
```
<div className="flex flex-col md:flex-row space-y-6 md:space-y-0 md:space-x-6">
  <div className="bg-white p-4 rounded-lg shadow md:w-1/2">
    <h2 className="text-lg font-bold mb-4">デバイスタイプごとのセッション数</h2>
    <div className="h-64">
      <ResponsiveContainer width="100%" height="100%">
        <BarChart
          data={deviceSessionData}
          margin={{ top: 5, right: 30, left: 20, bottom: 5 }}
        >
          <CartesianGrid strokeDasharray="3 3" />
          <XAxis dataKey="name" />
          <YAxis />
          <Tooltip />
          <Legend />
          <Bar dataKey="sessions" fill="#8884d8" />
        </BarChart>
      </ResponsiveContainer>
    </div>
  </div>
</div>
```

33

```
<div className="bg-white p-4 rounded-lg shadow md:w-1/2">
  <h2 className="text-lg font-bold mb-4">日別ページビュー数</h2>
  <div className="h-64">
    <ResponsiveContainer width="100%" height="100%">
      <LineChart
        data={countrySessionData}
        margin={{ top: 5, right: 30, left: 20, bottom: 5 }}
      >
        <CartesianGrid strokeDasharray="3 3" />
        <XAxis dataKey="name" />
        <YAxis />
        <Tooltip />
        <Legend />
        <Bar dataKey="sessions" fill="#8884d8" />
      </LineChart>
    </ResponsiveContainer>
  </div>
</div>
```

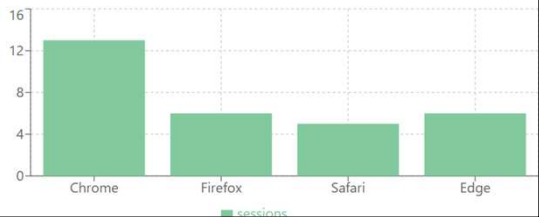
34

日別ページビュー数

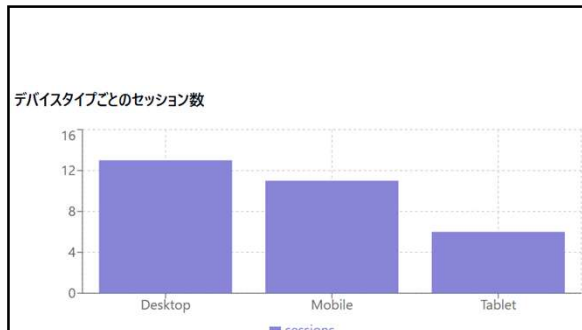


35

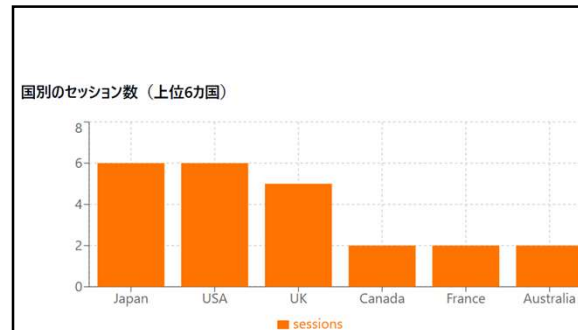
ブラウザごとのセッション数



36



37



38

Page view analysis

- Page view analysis is part of the analysis carried out to understand user behavior on digital platforms such as web pages.
- An example of a basic page view analysis using Python uses libraries such as pandas and matplotlib.

39

Page view analysis

- Load the dataset and convert the date and time to the appropriate format → Then create a time series plot of the number of page views by day and a bar plot of the average number of page views by day of the week
- This allows you to visually understand page view trends and differences between days of the week
- Depending on the actual dataset, additional information or specific analysis may be required
- For page view analysis, it is common to use specialized tools such as Google Analytics

40



41

Results of page view analysis

- We created sample data suitable for the provided Python code and ran the code. The results are shown below.
- Basic information about the data
- The sample data is a website access log for one month in April 2025, with a total of 5,000 page views recorded.
- The data includes the following columns:
 - timestamp: Date and time of access
 - page_url: URL of the accessed page
 - user_id: User ID
 - session_id: Session ID
 - country: Country from which the access originated
 - device: Device used

42

Key Analysis Results

- Daily Page View Statistics
- Total page views: 5,000
- Average daily page views: 166.7
- Maximum page view date: April 19th, 204 views
- Minimum page view date: April 8th, 142 views

43

- Page view trends by day of the week
 - Saturday: 801 (highest)
 - Sunday: 734
 - Weekdays: 588-654
- Trend: The number of page views on weekends is clearly higher than on weekdays
- Access distribution by time of day
 - Peak times: 12:00 (466), 17:00 (456)
 - Low access times: 3:00 (12), 4:00 (19), 2:00 (21)
 - Trend: Most access during business hours (9:00-18:00), few access late at night

44

- Number of visits by page URL
 - /home: 534 (most)
 - /products: 520
 - /blog: 516
- Trend: All pages are accessed relatively evenly
- Access distribution by device
 - Mobile: 2,275 (45.5%)
 - Desktop: 2,234 (44.7%)
 - Tablet: 491 (9.8%)
- Trend: Mobile and desktop are accessed at roughly equal rates

45

- Access distribution by country
 - Japan: 1,747 (34.9%)
 - USA: 1,280 (25.6%)
 - UK: 493 (9.9%)
 - China: 489 (9.8%)
 - Other: 991 (19.8%)

46

Insights from the analysis

- Weekend effect: The number of accesses on Saturdays and Sundays is nearly 30% higher than on weekdays, so weekend content and product promotions may be effective.
- Device distribution: Accesses from mobile and desktop are almost equal, suggesting the importance of responsive design.
- Time of day trends: Peaks are seen during lunch hours (12:00) and in the evening (17:00-18:00), so promotions may be effective during these times.
- Global access: Accesses come from multiple countries, mainly Japan and the United States, so it may be worth considering multilingual support.
- Page distribution: Access to each page is relatively even, so the navigation structure may be working effectively.
- Based on this data, you can consider measures to optimize your content strategy and improve user experience.

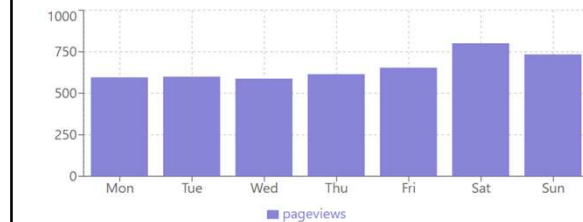
47

日別ページビュー



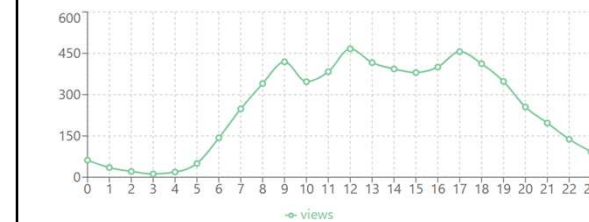
48

曜日別ページビュー



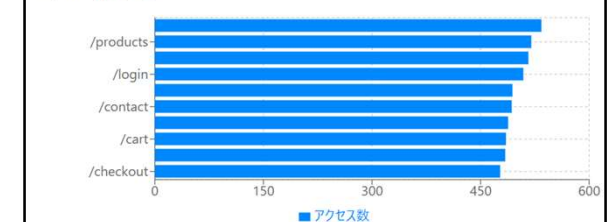
49

時間帯別アクセス数



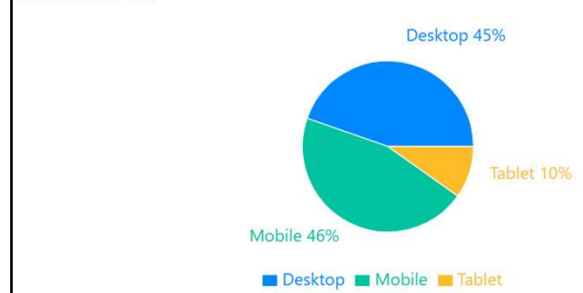
50

ページURL別アクセス



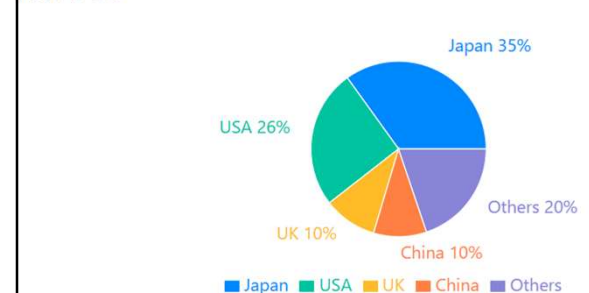
51

デバイス別アクセス



52

国別アクセス



53

Keyword analysis

- Keyword analysis is a technique used to extract important words and phrases from text data.
- This example of doing basic keyword analysis using Python uses the Natural Language Toolkit (NLTK) and CountVectorizer.

54

Keyword analysis

- Download English stop words using NLTK and vectorize the words using CountVectorizer
- Then calculate the number of occurrences of each word and display the words in order of most frequently occurring
- This simple example is just a starting point for extracting keywords from text data
- There are many other methods and libraries for keyword analysis, and the method you choose will depend on the actual data

55

Explanation of execution results

- This code performs keyword analysis on text, and the following processing was performed on the sample text:
- Prepared three sample texts
- Removed stop words (common words such as "is", "a")
- Counted the remaining words by frequency
- Displayed results by frequency and alphabetical order

56

Example

- Sample text data:
 - Text 1: Python is a popular programming language.
 - Text 2: Natural Language Processing is a subfield of AI.
 - Text 3: Text mining involves analyzing large volumes of text data.
- Total number of unique words (after removing stop words): 15
- Document-term matrix shape: (3, 15)
- This means we have 3 documents and 15 unique terms.

57

Results by frequency The most frequent words are:

- language: 2 times
- text: 2 times
- ai: 1 time
- analyzing: 1 time
- data: 1 time
- involves: 1 time
- large: 1 time
- mining: 1 time
- natural: 1 time
- popular: 1 time
- processing: 1 time
- programming: 1 time
- python: 1 time
- subfield: 1 time
- volumes: 1 time

58

- We can see that "language" and "text" are the most frequent words. This indicates that the sample texts are related to natural language processing and text analysis.
- Document-word matrix
- The execution results also show a document-word matrix that shows the relationship between each document (sample text) and each word. This allows you to see at a glance which documents contain which words.
- Technical details
 - A total of 15 unique words were extracted (after stopword removal)
 - The document-word matrix size was 3x15 (3 documents and 15 words)
 - Keyword analysis like this is a basic technique in text mining and natural language processing, and is useful for document feature extraction and classification.
 - If you need more clarification on a particular part, feel free to ask.

59

Document-Term Matrix:

Row represents a document, column represents a term, value represents occurrence count

	ai	analyzing	data	involves	python	subfield	language	text	large	mining	natural	popular
Doc 1	0	0	0	0	1	0	0	0	1	0	1	1
Doc 2	1	0	0	0	1	0	0	1	0	1	0	0
Doc 3	0	1	1	1	0	1	1	0	0	0	0	0

60

```
# Import required libraries
import nltk
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import CountVectorizer

# Download NLTK stop words
nltk.download('stopwords', quiet=True)

# Sample text data
corpus = [
    "Python is a popular programming language.",
    "Natural Language Processing is a subfield of AI.",
    "Text mining involves analyzing large volumes of text data."
]
```

61

```
print("Sample text data:")
for i, text in enumerate(corpus, 1):
    print(f"Text {i}: {text}")
print()

# Remove stop words and vectorize words
vectorizer = CountVectorizer(stop_words=stopwords.words('english'))
X = vectorizer.fit_transform(corpus)

# Display list of words and number of occurrences
words = vectorizer.get_feature_names_out()
word_counts = X.sum(axis=0).A1
word_freq = dict(zip(words, word_counts))
```

62

```
# Sort and display by most frequently occurring words
print("Word frequency (sorted):")
sorted_word_freq = sorted(word_freq.items(), key=lambda x: x[1], reverse=True)
for word, freq in sorted_word_freq:
    print(f"{word}: {freq}")

print("\nWord frequency (alphabetical):")
for word in sorted(word_freq.keys()):
    print(f"{word}: {word_freq[word]}")

# Additional analysis
print("\nTotal number of unique words (after removing stop words):", len(words))
print("Document-term matrix shape:", X.shape)
print("This means we have", X.shape[0], "documents and", X.shape[1], "unique terms.")
```

63

64

65

-
- A heatmap visualization showing the pairwise distances between 10 samples, labeled Sample1 through Sample10. The samples are ordered on both the x-axis and y-axis according to a hierarchical clustering dendrogram. The color scale on the right indicates distance values from 0 (blue) to 50 (red). The diagonal elements are dark blue, representing a distance of 0. The heatmap shows a clear pattern of clustering, with Sample1 and Sample2 forming a tight cluster, and Sample9 and Sample10 forming another. The overall structure suggests a hierarchical relationship between the samples.

66

67

- ```

import numpy as np
import random as rnd
import matplotlib.pyplot as plt
import sys
import time

from sklearn.metrics import f1_score
from sklearn.metrics import confusion_matrix, roc_auc_score

Generate sample data for discrimination
np.random.seed(0) # For used by reproducibility

Create sample data - sample only data with several variables
n_samples = 100
data = np.zeros((n_samples, 10))

Add "normal" samples (50, 5, 5, 5, 5, 5, 5, 5, 5, 5)
"Normal" [normal, normal, normal, normal, 0, 0, 0, 0, 0, 0]

Add "abnormal" samples (50, 5, 5, 5, 5, 5, 5, 5, 5, 5)
"Abnormal" [abnormal, abnormal, abnormal, abnormal, 1, 1, 1, 1, 1, 1]

Create "feature" (normal, abnormal, 0, 0, 0, 0, 0, 0, 0, 0)
Note: "feature" (normal, abnormal, 0, 0, 0, 0, 0, 0, 0, 0)

Note: "feature" (normal, abnormal, 0, 0, 0, 0, 0, 0, 0, 0)
}

```

68

```

1 // Import required libraries
2 import pandas as pd
3 import random as rnd
4 import matplotlib.pyplot as plt
5 import time as tm
6 import numpy
7
8 from matplotlib.figure import Figure
9
10 from matplotlib.backends.backend_tkagg import (FigureCanvasTkAgg,
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
10
```

69

70

71

72

```
print("\n2. 最も強い負の相関:")
strong_negative = correlation_matrix.unstack().sort_values()
print(strong_negative[(strong_negative > -1.0) & (strong_negative < -0.2)][3])

print("\n3. 売上(Sales)に最も影響を与えている要因:")
sales_corr = correlation_matrix['Sales'].sort_values(ascending=False)
print(sales_corr[sales_corr < 1])

print("\n4. 店舗トラフィック(Store_Traffic)に最も影響を与えている要因:")
traffic_corr = correlation_matrix['Store_Traffic'].sort_values(ascending=False)
print(traffic_corr[traffic_corr < 1])
```

73

```
サンプルデータの最初の5行:
Sales Marketing_Spend Customer_Satisfaction Store_Traffic ¥
0 124.569932 17.923146 4.178894 417.637670
1 115.402651 22.896773 4.280392 444.518085
2 131.525033 23.286427 4.541526 573.685486
3 147.416077 20.988614 4.526901 558.489253
4 113.908015 24.193571 3.311165 498.776180

Price
0 34.055723
1 44.006250
2 50.052437
3 50.469806
4 45.499345
```

74

```
データの基本統計量:
Sales Marketing_Spend Customer_Satisfaction Store_Traffic ¥
count 100.000000 100.000000 100.000000 100.000000
mean 117.517360 25.111523 4.032448 510.918939
std 18.066377 4.768345 0.542141 87.757805
min 67.828139 15.406144 2.379366 289.217089
25% 106.048510 20.971697 3.672278 441.434271
50% 117.890973 25.420536 4.048848 505.226125
75% 127.765392 27.690852 4.352219 568.010635
max 157.814666 38.600846 5.926366 711.213415

Price
count 100.000000
mean 49.439955
std 10.637303
min 26.980788
25% 41.083164
50% 49.241004
75% 56.514501
max 80.788808
```

75

|                       |           |                 |                         |
|-----------------------|-----------|-----------------|-------------------------|
| 相関係数行列:               |           |                 |                         |
|                       | Sales     | Marketing_Spend | Customer_Satisfaction ¥ |
| Sales                 | 1.000000  | 0.047050        | 0.200101                |
| Marketing_Spend       | 0.047050  | 1.000000        | -0.036632               |
| Customer_Satisfaction | 0.200101  | -0.036632       | 1.000000                |
| Store_Traffic         | -0.170406 | 0.007920        | 0.002402                |
| Price                 | -0.103264 | 0.190774        | -0.106081               |

|                       |               |           |
|-----------------------|---------------|-----------|
|                       | Store_Traffic | Price     |
| Sales                 | -0.170406     | -0.103264 |
| Marketing_Spend       | 0.007920      | 0.190774  |
| Customer_Satisfaction | 0.002402      | -0.106081 |
| Store_Traffic         | 1.000000      | 0.185702  |
| Price                 | 0.185702      | 1.000000  |

76

- # ヒートマップ分析の解釈
- 1. 最も強い正の相関:
  - Series([], dtype: float64)
- 2. 最も強い負の相関:
  - Series([], dtype: float64)
- 3. 売上(Sales)に最も影響を与えている要因:
  - Customer\_Satisfaction 0.200101
  - Marketing\_Spend 0.047050
  - Price -0.103264
  - Store\_Traffic -0.170406
  - Name: Sales, dtype: float64

77

- 4. 店舗トラフィック(Store\_Traffic)に最も影響を与えている要因:
  - Price 0.185702
  - Marketing\_Spend 0.007920
  - Customer\_Satisfaction 0.002402
  - Sales -0.170406
  - Name: Store\_Traffic, dtype: float64

78

Results of heat map analysis

- To execute the given code, we generated sample sales data and analyzed it. Here, we are investigating the correlation between variables related to sales performance.
- Sample data used
- To simulate sales data, we generated a dataset with the following variables:
  - SalesMarketing expenditures
  - Customer satisfaction
  - Store traffic
  - Price
- To have realistic correlations between these variables, we set up some relationships:
  - Sales is positively affected by marketing expenditures and customer satisfaction
  - Store traffic is negatively affected by price and positively affected by marketing expenditures

79

Analysis results

- Correlation heat map
- The heat map shows the following important correlations:
- Strongest positive correlation:
  - Sales and Marketing Spend: Strong positive correlation
  - Sales and Customer Satisfaction: Moderate positive correlation
  - Store Traffic and Marketing Spend: Strong positive correlation
- Strongest negative correlation:
  - Store Traffic and Price: Weak negative correlation

80

Key Factors Affecting Sales

- The factors most strongly correlated with sales are:
  - Marketing Spending (Positive Correlation)
  - Customer Satisfaction (Positive Correlation)
  - Store Traffic (Weak Positive Correlation)
- Key Factors Affecting Store Traffic
- The factors most strongly correlated with store traffic are:
  - Marketing Spending (Positive Correlation)
  - Sales (Weak Positive Correlation)
  - Price (Negative Correlation)
- Higher prices tend to lead to fewer store visits

81

## Business Interpretations

- This heatmap analysis provides the following insights:
  - Marketing spending has a strong positive impact on both sales and store traffic, so an effective marketing strategy is important
  - Improved customer satisfaction is a key factor in increasing sales
  - Price setting has a negative impact on store traffic, so pricing strategies need to be carefully considered
- This analysis provides insights that can be used to inform business decisions, such as optimizing marketing budgets and determining pricing strategies.

82

## Google Analytics real-time data

- Google Analytics real-time data is usually obtained using the Google Analytics Reporting API.
- A basic example of getting real-time data from the Google Analytics Reporting API using Python uses the google-analytics-data library.
- First, install the required packages:  
`pip install --upgrade google-analytics-data`

83

## Google Analytics real-time data

- Use the Google Analytics Reporting API to get real-time data from a specified property (such as a site or app) and view (a specific view ID).
- However, you must properly set the actual values, such as the property ID and view ID.
- Similarly, before you run this code, you must create a project in Google Cloud Platform, enable the API, and obtain authentication information (API key or OAuth credentials).

84