۱.۴ پیش تخصیص آرایه‌های NumPy

مسئله

شما باید آرایه‌هایی با اندازه معین را با مقداری از قبل تخصیص دهید.

راه‌حل

NumPy دارای توابعی برای تولید بردارها و ماتریس‌هایی با هر اندازه با استفاده از ۰، ۱ یا مقادیر دلخواه شما است:

```
# Load library
import numpy as np


# Generate a vector of shape (1,5) containing all zeros
vector = np.zeros(shape=5)


# View the matrix
print(vector)
array([0., 0., 0., 0., 0.])


# Generate a matrix of shape (3,3) containing all ones
matrix = np.full(shape=(3,3), fill_value=1)


# View the vector
print(matrix)
array([[1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.]])
```

بحث

تولید آرایه‌هایی که از قبل با داده‌ها پر شده‌اند، برای چندین هدف مفید است، مانند عملکرد بهتر کد یا اسـتفاده از داده‌هـای مصنوعی برای آزمایش الگوریتم‌ها. در بسیاری از زبان‌های برنامه نویسی، پیش تخصیص‌یک آرایه از مقادیر پیش فرض (مانند ۰s)یک روش معمول در نظر گرفته می‌شود.

۲.۶ بارگذاری‌یک فایل پارکت[1]

مسئله

شما بایدیک فایل پارکت را بارگذاری کنید.

راه‌حل

تابع read_parquet از کتابخانه‌ی pandas به ما این امکان می‌دهد که فایل‌های Parquet را بخوانیم:

```
# Load library
import pandas as pd


# Create URL
url = 'https://machine-learning-python-cookbook.s3.amazonaws.com/data.parquet'


# Load data
dataframe = pd.read_parquet(url)


# View the first two rows
dataframe.head(2)
```

	دسته بندی	تاریخ و زمان	عدد
۰	۰	۲۰۱۵-۰۱-۰۱ ۰۰:۰۰:۰۰	۵
۱	۰	۲۰۱۵-۰۱-۰۱ ۰۰:۰۰:۰۱	۵

بحث

Parquetیک فرمت محبوب ذخیره سازی داده در فضای بزرگ داده است. اغلب با ابزارهای داده‌های بزرگ مانند Hadoop و Spark مورد استفاده قرار می‌گیرد. در حالی که PySpark خارج از تمرکز این کتاب است، به احتمال زیاد شرکت‌هایی که در مقیاس بزرگ فعالیت می‌کنند ازیک قالب ذخیره‌سازی داده کارآمد مانند پارکت استفاده می‌کنند، و دانستن نحوه خواندن آن دریک دیتافریم و دستکاری آن بسیار ارزشمند است.

همچنین ببینید:

- مستندات پارکت آپاچی[2]

---

[1] - Parquet
[2] - Apache Parquet documentation

۲.۷ بارگیری یک فایل Avro

مسئله

شما بایدیک فایل Avro را در دیتافرِیم pandas بارگذاری کنید.

راه‌حل

از تابع read_avro در کتابخانه‌ی pandavro استفاده کنید:

```
# Load library
import requests
import pandavro as pdx


# Create URL
url = 'https://machine-learning-python-cookbook.s3.amazonaws.com/data.avro'


# Download file
r = requests.get(url)
open('data.avro', 'wb').write(r.content)


# Load data
dataframe = pdx.read_avro('data.avro')


# View the first two rows
dataframe.head(2)
```

	دسته بندی	تاریخ و زمان	عدد
۰	۰	۲۰۱۵-۰۱-۰۱ ۰۰:۰۰:۰۰	۵
۱	۰	۲۰۱۵-۰۱-۰۱ ۰۰:۰۰:۰۱	۵

بحث

Apache Avroیک فرمت داده باینری و منبع باز است که برای ساختار داده بر طرحواره‌ها متکی است. در زمان نوشتن، به اندازه پارکت رایج نیست. با این حال، فرمت‌های داده‌های باینری بزرگ مانند Avro، Thrift و Protocol Buffer به دلیل ماهیت کارآمدشان در حال افزایش محبوبیت هستند. اگر با سیستم‌های داده بزرگ کار می‌کنید، احتمالاً در آینده نزدیک بایکی از این فرمت‌ها مواجه خواهید شد.

همچنین ببینید:

- مستندات پارکت آپاچی[۳]

[3] - Apache Avro documentation

۲.۸ جستجو در پایگاه داده SQLite[4]

مسئله

شما باید داده‌ها را ازیک پایگاه داده با استفاده از زبان SQL بارگیری کنید.

راه‌حل

read_sql_query از کتابخانه‌ی Pandas به ما این امکان را می‌دهد که‌یک دستور کوئری[5] SQL در پایگاه داده ایجاد کرده و آن را بارگذاری کنیم:

```
# Load libraries
import pandas as pd
from sqlalchemy import create_engine


# Create a connection to the database
database_connection = create_engine('sqlite:///sample.db')


# Load data
dataframe = pd.read_sql_query('SELECT * FROM data', database_connection)


# View first two rows
dataframe.head(2
```

	نام	نام خانوادگی	سن	امتیاز پیش‌آزمون
۰	Json	Miller	۴۲	۴
۱	Molly	Jaboson	۵۲	۲۴

بحث

Apache SQL زبانی برای استخراج داده‌ها از پایگاه‌های داده است. در این دستور، ابتدا از create_engine برای تعریف اتصال به موتور پایگاه داده SQL به نام SQLite استفاده می‌کنیم. در مرحله بعد ما از read_sql_query در کتابخانه‌ی Pandas برای کوئری زدن روی پایگاه داده با استفاده از زبان SQL و قرار دادن نتایج دریک DataFrame استفاده می‌کنیم.

SQL به خودی خودیک زبان است و اگرچه فراتر از محدوده این کتاب است، اما مطمئناً برای هر کسی که می‌خواهد در مورد یادگیری ماشینی بیاموزد، ارزش دارد. کوئری SQL ما، SELECT * FROM data، از پایگاه داده می‌خواهد که تمام ستون‌های (*) جدول را به نام عنوان داده به ما بدهد.

---

[4] - Querying a SQLite Database
[5] - query

توجه داشته باشید که این‌یکی از معدود دستور العمل‌های این کتاب است که بدون کد اضافی اجرا نمی‌شود. به طـور خـاص، create_engine('sqlite:///sample.db') فرض می‌کند که‌یک پایگاه داده SQLite از قبل وجود دارد.

- SQLite
- آموزش SQL در وبسایت Schools۳W

۲.۹ کوئری زدن روی‌یک پایگاه داده‌ی SQL از راه دور

مسئله

شما باید به‌یک پایگاه داده SQL از راه دور متصل شوید و داده‌ها را از آن بخوانید.

راه‌حل

یک اتصال با pymysql ایجاد کنید و آن را با Pandas دریک DataFrame بخوانید:

```
# Load library
import requests
import pandavro as pdx


# Create URL
url = 'https://machine-learning-python-cookbook.s3.amazonaws.com/data.avro'


# Download file
r = requests.get(url)
open('data.avro', 'wb').write(r.content)


# Load data
dataframe = pdx.read_avro('data.avro')


# View the first two rows
dataframe.head(2)
```

	عدد	تاریخ و زمان	دسته بندی
۰	۵	۲۰۱۵-۰۱-۰۱ ۰۰:۰۰:۰۰	۰
۱	۵	۲۰۱۵-۰۱-۰۱ ۰۰:۰۰:۰۱	۰

بحث

از بین تمام دستور العمل‌های ارائه شده در این فصل، احتمالاً این دستور العملی است کـه بیشـتر در دنیـای واقعـی اسـتفاده خواهیم کرد. در حالی که اتصال و خواندن ازیک پایگاه داده نمونه‌ی sqlite مفید است، احتمالاً نماینده‌ی جداولی نیست کـه

باید دریک محیط سازمانی به آنها متصل شوید. اکثر نمونه‌های SQL که به آن‌ها متصل می‌شوید، از شما می‌خواهند کـه بـه میزبان و پورت‌یک دستگاه از راه دور متصل شوید، ویک نام کاربری و رمز عبور برای احراز هویت مشخص کنید. این مثـال از شما می‌خواهد که‌یک نمونه SQL در حال اجرا را به صورت محلی راه‌اندازی کنید کـه ازیـک سـرور راه دور در لوکال‌هاسـت تقلید می‌کند تا بتوانید حسی از گردش کار دریافت کنید.

همچنین ببینید:

- مستندات PyMySQL
- مستندات Read SQL در کتابخانه‌ی Pandas

۲.۱۰ بارگیری داده‌ها از Google Sheet

مسئله

شما باید داده‌ها را مستقیماً از Google Sheet بخوانید.

راه‌حل

از read_CSV در کتابخانه‌ی pandas استفاده کنید و نشانی اینترنتی ارسال کنید کـه Google Sheet را بـه‌عنوان CSV صادر می‌کند:

```
# Import libraries
import pandas as pd


# Google Sheet URL that downloads the sheet as a CSV
url = "https://docs.google.com/spreadsheets/d/"\
"1ehC-9otcAuitqnmWksqt1mOrTRCL38dv0K9UjhwzTOA/export?format=csv"


# Read the CSV into a dataframe
dataframe = pd.read_csv(url)


# View the first two rows
dataframe.head(2)
```

	عدد	تاریخ و زمان	دسته بندی
۰	۵	۲۰۱۵-۰۱-۰۱ ۰۰:۰۰:۰۰	۰
۱	۵	۲۰۱۵-۰۱-۰۱ ۰۰:۰۰:۰۱	۰

بحث

در حالی که Google Sheets را می‌توان به راحتی دانلود کرد، گاهی اوقات مفید است که بتوانید آنها را مستقیماً در پایتون بدون هیچ مرحله میانی بخوانید. پارامتر کوئری export?format=csv/ در انتهای URL بالایک نقطه پایانی ایجاد می‌کند که می‌توانیم فایل را دانلود کنیم یا در Pandas بخوانیم.

همچنین ببینید:

- Google Sheets API

۲.۱۱ بارگیری داده‌ها از سطل[6] ۳S

مسئله

شما بایدیک فایل CSV را ازیک سطل ۳S که به آن دسترسی دارید بخوانید.

راه‌حل

گزینه‌های ذخیره سازی را به Pandas اضافه کنید تا به شیء[7] ۳S دسترسی داشته باشد:

```
# Import libraries
import pandas as pd


# S3 path to CSV
s3_uri = "s3://machine-learning-python-cookbook/data.csv"


# Set AWS credentials (replace with your own)
ACCESS_KEY_ID = "xxxxxxxxxxxxx"
SECRET_ACCESS_KEY = "xxxxxxxxxxxxxxxx"


# Read the CSV into a dataframe
dataframe = pd.read_csv(s3_uri,storage_options={
                "key": ACCESS_KEY_ID,
                "secret": SECRET_ACCESS_KEY,
        }
)
#View first two rows
dataframe.head(2)
```

	عدد	تاریخ و زمان	دسته بندی
۰	۵	۲۰۱۵-۰۱-۰۱ ۰۰:۰۰:۰۰	۰
۱	۵	۲۰۱۵-۰۱-۰۱ ۰۰:۰۰:۰۱	۰

[6] - S۳ Bucket
[7] - object

بحث

اکنـون بسـیاری از شـرکت‌ها داده‌هـا را در فروشـگاه‌های blob ارائه‌دهنـده ابـری ماننـدAmazon S۳ یـا Google Cloud Storage(GCS) نگهداری می‌کنند. معمولاً متخصصان یادگیری ماشینی برای بازیابی داده‌ها به این منابع متصـل می‌شـوند. اگرچه S۳s) URI ۳(//machine-learning-python-cookbook/data.csv: عمومی‌است، اما همچنان از شما می‌خواهـد که اعتبار دسترسی AWS خود را برای دسترسی به آن ارائـه دهیـد. شـایان ذکـر اسـت کـه اشـیاء عمـومی‌همچنین دارای URLهای HTTP هستند که می‌توانند فایل‌ها را از آن دانلود کنند، مانند این مورد برای فایل CSV.

همچنین ببینید:

- ۳Amazon S
- اعتبارنامه امنیتی AWS

۲.۱۲ بارگذاری داده‌های بدون ساختار

مسئله

شما باید داده‌های بدون ساختار مانند متن یا تصاویر را بارگیری کنید.

راه‌حل

از تابع باز پایتون برای بارگذاری اطلاعات استفاده کنید:

```
# Import libraries
import requests

# URL to download the txt file from
txt_url = "https://machine-learning-python-cookbook.s3.amazonaws.com/text.txt"

# Get the txt file
r = requests.get(txt_url)

# Write it to text.txt locally
with open('text.txt', 'wb') as f:
f.write(r.content)


# Read in the file
with open('text.txt', 'r') as f:
text = f.read()

# Print the content
print(text)
Hello there!
```

	دسته بندی	تاریخ و زمان	عدد
۰	۰	۲۰۱۵-۰۱-۰۱ ۰۰:۰۰:۰۰	۵
۱	۰	۲۰۱۵-۰۱-۰۱ ۰۰:۰۰:۰۱	۵

بحث

در حالی که داده‌های ساختاریافته را می‌توان به راحتی از CSV، JSON یا پایگاه‌های داده مختلف خواند، داده‌های بدون ساختار می‌توانند چالش برانگیزتر باشند و ممکن است نیاز به پردازش سفارشی داشته باشند. گاهی اوقات باز کردن و خواندن فایل‌ها با استفاده از تابع باز اصلی پایتون مفید است. این مورد به ما امکان می‌دهد فایل‌ها را باز کنیم و سپس محتوای آن فایل را بخوانیم.

همچنین ببینید:

- تابع باز پایتون
- مدیران context در پایتون

۳.۲ دریافت اطلاعات در مورد داده ها

مسئله

شما می‌خواهید برخی از ویژگی‌های یک DataFrame را مشاهده کنید.

راه‌حل

یکی از ساده‌ترین کارهایی که می‌توانیم پس از بارگذاری داده‌ها انجام دهیم، مشاهده چند ردیف اول با استفاده از head است:

```
# Load library
import pandas as pd


# Create URL
url =
'https://raw.githubusercontent.com/chrisalbon/sim_data/master/titanic.csv'


# Load data
dataframe = pd.read_csv(url)


# Show two rows
dataframe.head(2)
```

	نام	کلاس P	سن	جنسیت

۰	Allen, Miss Elisabeth Walton	اول	۲۹.۰	زن
۱	Allison, Miss Helen Loraine	اول	۲.۰	زن

همچنین می‌توانیم به تعداد سطرها و ستون‌ها نگاهی بیندازیم:

```
# Show dimensions
dataframe.shape
```

```
(1313, 6)
```

ما می‌توانیم آمار توصیفی برای هر ستون عددی را با استفاده از describe بدست آوریم:

```
# Show statistics
dataframe.describe()
```

	سن	زنده مانده	کد جنسیت
تعداد	۷۵۶.۰۰۰۰۰۰	۱۳۱۳.۰۰۰۰۰۰	۱۳۱۳.۰۰۰۰۰۰
میانگین	۳۰.۳۹۷۹۸۹	۰.۳۴۲۷۲۷	۰.۳۵۱۸۶۶
std	۱۴.۲۵۹۰۴۹	۰.۴۷۴۸۰۲	۰.۴۷۷۷۳۴
مینیمم	۰.۱۷۰۰۰۰	۰.۰۰۰۰۰۰	۰.۰۰۰۰۰۰
۲۵%	۲۱.۰۰۰۰۰۰	۰.۰۰۰۰۰۰	۰.۰۰۰۰۰۰
۵۰%	۲۸.۰۰۰۰۰۰	۰.۰۰۰۰۰۰	۰.۰۰۰۰۰۰
۷۵%	۳۹.۰۰۰۰۰۰	۱.۰۰۰۰۰۰	۱.۰۰۰۰۰۰
ماکزیمم	۷۱.۰۰۰۰۰۰	۱.۰۰۰۰۰۰	۱.۰۰۰۰۰۰

علاوه بر این، روش info می‌تواند اطلاعات مفیدی را نشان دهد:

```
# Show info
dataframe.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1313 entries, 0 to 1312
Data columns (total 6 columns):
Data columns (total 6 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Name 1313 non-null object
1 PClass 1313 non-null object
2 Age 756 non-null float64
3 Sex 1313 non-null object
4 Survived 1313 non-null int64

5 SexCode 1313 non-null int64
dtypes: float64(1), int64(2), object(3)
memory usage: 61.7+ KB
```

بحث

پس از بارگیری برخی از داده ها، ایده خوبی است که بفهمیم ساختار آن چگونه است و چه نوع اطلاعـاتی در آن وجـود دارد. در حالت ایده آل، ما داده‌های کامل را مستقیماً مشاهده می‌کنیم. اما در بیشتر موارد دنیای واقعی، داده‌ها می‌توانند هزاران تا صدها هزار تا میلیون‌ها سطر و ستون داشته باشند. در عوض، برای مشاهده برش‌های کوچک و محاسبه آمار خلاصـه داده‌هـا باید به کشیدن نمونه تکیه کنیم.

در راه حل خود، ما از مجموعه داده اسباب بازی مسافران تایتانیک استفاده می‌کنیم. با استفاده از head، می‌تـوانیم بـه چنـد ردیف اول (پنج به طور پیش فرض) داده‌ها نگاه کنیم. از طرف دیگر، می‌توانیم از tail برای مشاهده چند ردیف آخر اسـتفاده کنیم. با شکل می‌توانیم ببینیم که DataFrame ما شامل چند ردیف و سـتون اسـت. بـا توصـیف می‌تـوانیم برخـی از آمـار توصیفی اولیه برای هر ستون عددی را ببینیم. و در نهایت، اطلاعـات تعـدادی از نقـاط داده مفیـد را در مـورد DataFrame نشان می‌دهد، از جمله انواع داده‌های شاخص و ستون، مقادیر غیر تهی و میزان استفاده از حافظه.

شایان ذکر است که آمار خلاصه، همیشه داستان کامل را بیان نمی‌کند. به عنوان مثال، Pandas با ستون‌های زنده‌مانده و کد جنسیتی به عنوان ستون‌های عددی رفتار می‌کنند زیرا دارای ۱ و ۰ هستند. با این حال، در این مورد مقـادیر عـددی نشـان دهنده دسته‌ها هستند. به عنوان مثال، اگر Survived برابر با ۱ باشد، نشان می‌دهد که مسافر از فاجعه جان سالم به در برده است. به همین دلیل، برخی از آمار خلاصه‌ی ارائه شده منطقی نیستند، مانند انحراف استاندارد ستون کد جنسـیتی (نشـانگر جنسیت مسافر).

۳.۳ برش DataFrame ها

مسئله

شما باید یک زیر مجموعه داده یا برش‌هایی از یک DataFrame را انتخاب کنید.

راه‌حل

از loc یا iloc برای انتخاب یک یا چند ردیف یا مقدار استفاده کنید:

```
# Load library
import pandas as pd


# Create URL
url =
'https://raw.githubusercontent.com/chrisalbon/sim_data/master/titanic.csv'


# Load data
dataframe = pd.read_csv(url)


# Select first row
dataframe.iloc[0]
Name        Allen, Miss Elisabeth Walton
PClass      1st
Age         29
Sex         female
Survived    1
SexCode     1
Name:       0, dtype: object
```

می‌توانیم از : برای تعریف برش ردیف‌هایی که می‌خواهیم، استفاده کنیم. مانند انتخاب ردیف‌های دوم، سوم و چهارم:

```
# Select three rows
dataframe.iloc[1:4]
```

	نام	کلاس P	سن	جنسیت
۱	Allison, Miss Helen Loraine	اول	۲.۰	زن
۲	Allison, Mr Hudson Joshua Creighton	اول	۳۰.۰	مرد
۳	Allison, Mrs Hudson JC (Bessie Waldo Daniels)	اول	۲۵.۰	زن

حتی می‌توانیم از آن برای به دست آوردن تمام ردیف‌ها تا یک نقطه استفاده کنیم. مانند همه ردیف‌ها تا ردیف چهارم:

```
# Select four rows
dataframe.iloc[:4]
```

	نام	کلاس P	سن	جنسیت
۰	Allen, Miss	اول	۲۹.۰	زن

	Elisabeth Walton			
۱	Allison, Miss Helen Loraine	اول	۲.۰	زن
۲	Allison, Mr Hudson Joshua Creighton	اول	۳۰.۰	مرد
۳	Allison, Mrs Hudson JC (Bessie Waldo Daniels)	اول	۲۵.۰	زن

DataFrameها نیازی به ایندکس شدن عددی ندارند. ما می‌توانیم ایندکس یک DataFrame را روی هر مقداری که مقدار آن برای هر ردیف منحصر به فرد باشد، تنظیم کنیم. برای مثال، می‌توانیم فهرست را به‌عنوان نام مسافران تنظیم کنیم و سپس ردیف‌ها را با استفاده از یک نام انتخاب کنیم:

```
# Set index
dataframe = dataframe.set_index(dataframe['Name'])


# Show row
dataframe.loc['Allen, Miss Elisabeth Walton']
```

```
Name        Allen, Miss Elisabeth Walton
PClass      1st
Age         29
Sex         female
Survived    1
SexCode     1
Name:       Allen, Miss Elisabeth Walton, dtype: object
```

بحث

همه سطرها در DataFrame در کتابخانه‌ی Pandas دارای یک مقدار شاخص منحصر به فرد است. به طور پیش فرض، این شاخص یک عدد صحیح است که موقعیت ردیف را در DataFrame نشان می‌دهد. با این حال، لازم نیست که حضور داشته باشد. شاخص‌های DataFrame را می‌توان به صورت رشته‌های الفبایی منحصر به فرد یا شماره مشتری تنظیم کرد. Pandas برای انتخاب ردیف‌ها و تکه‌های ردیف‌ها دو روش ارائه می‌دهد:

- loc زمانی مفید است که نمایه DataFrame یک برچسب (به عنوان مثال، یک رشته) باشد.
- iloc با جستجوی موقعیت در DataFrame کار می‌کند. برای مثال، iloc[0] بدون در نظر گرفتن اینکه شاخص یک عدد صحیح است یا یک برچسب، ردیف اول را برمی‌گرداند.

استفاده زیاد و راحت بودن با loc و iloc مفید است زیرا هنگام پاکسازی داده‌ها زیاد ظاهر می‌شوند.

۳.۵ مرتب سازی مقادیر

مسئله

شما باید یک دیتافریم را بر اساس مقادیر یک ستون مرتب کنید.

راه‌حل

از تابع sort_values در کتابخانه‌ی pandas استفاده کنید:

```
# Load library
import pandas as pd

# Create URL
url =
'https://raw.githubusercontent.com/chrisalbon/sim_data/master/titanic.csv'

# Load data
dataframe = pd.read_csv(url)

# Sort the dataframe by age, show two rows
dataframe.sort_values(by=["Age"]).head(2)
```

	نام	کلاس P	سن	جنسیت
۷۶۳	Dean, Miss Elizabeth Gladys (Millvena)	سوم	۰.۱۷	زن
۷۵۱	Danbom, Master Gilbert Sigvard Emanuel	سوم	۰.۳۳	مرد

بحث

در طول تجزیه و تحلیل و کاوش داده ها، مرتب کردن یک DataFrame بر اساس یک ستون یـا مجموعـه ای از سـتون‌های خاص اغلب مفید است. آرگومان by در sort_values فهرستی از ستون‌ها را می‌گیرد کـه براسـاس آن DataFrame مرتـب می‌شود و این مرتب سازی بر اساس ترتیب نام ستون‌ها در لیست مرتب است.

به‌طور پیش‌فرض، آرگومان accending روی True تنظیم شده است، بنابراین مقادیر پایین‌ترین به بالاترین را مرتب می‌کند. اگر ما مسن‌ترین مسافران را به جای جوان‌ترین مسافران می‌خواستیم، می‌توانیم آن را روی False تنظیم کنیم.

۳.۱۶ تجمیع عملیات و آمار

مسئله

شما باید یک عملیات را روی هر ستون (یا مجموعه ای از ستون ها) در یک دیتافریم جمع کنید.

راه‌حل

از روش agg در کتابخانه‌ی pandas استفاده کنید. در اینجا، ما به راحتی می‌توانیم حداقل مقدار هر ستون را بدست آوریم:

```
# Load library
import pandas as pd

# Create URL
url =
'https://raw.githubusercontent.com/chrisalbon/sim_data/master/titanic.csv'

# Load data
dataframe = pd.read_csv(url)

# Get the minimum of every column
dataframe = pd.read_csv(url)
Name        Abbing, Mr Anthony
PClass      *
Age         0.17
Sex         female
Survived    0
SexCode     0
dtype:      object
```

گاهی اوقات، ما می‌خواهیم توابع خاصی را به مجموعه‌های خاصی از ستون‌ها اعمال کنیم:

```
# Mean Age, min and max SexCode
dataframe.agg({"Age":["mean"], "SexCode":["min", "max"]})
```

	سن	جنسیت
میانگین	۳۰.۳۹۷۹۸۹	Nan
مینیمم	Nan	۰.۰
ماکزیمم	Nan	۱.۰

همچنین می‌توانیم توابع انبوه را برای گروه‌ها اعمال کنیم تا آمار توصیفی و خاص‌تری به دست آوریم:

```
# Number of people who survived and didn't survive in each class
dataframe.groupby(
        ["PClass","Survived"]).agg({"Survived":["count"]}
  ).reset_index()
```

کلاس P	زنده مانده		تعداد
۰	*	۰	۱
۱	اول	۰	۱۲۹
۲	اول	۱	۱۹۳
۳	دوم	۰	۱۶۰

۴	دوم	۱	۱۱۹
۵	سوم	۰	۵۷۳
۶	سوم	۱	۱۳۸

بحث

توابع انبوه به ویژه در طول تجزیه و تحلیل داده‌های اکتشافی برای یادگیری اطلاعات در مورد زیرجمعیت‌های[8] مختلف داده‌ها و رابطه بین متغیرها مفید هستند. با گروه‌بندی داده‌ها و اعمال آمار انبوه، می‌توانید الگوهایی را در داده‌ها مشاهده کنید که ممکن است در طی فرآیند یادگیری ماشین یا مهندسی ویژگی مفید باشند. در حالی که نمودارهای بصری نیز مفید هستند، اغلب مفید است که چنین آمار توصیفی خاصی نیز به عنوان مرجع برای درک بهتر داده‌ها وجود داشته باشد.

همچنین ببینید:

- مستندات agg در کتابخانه‌ی Pandas

۶.۸ انجام شناسایی موجودیت با نام

مسئله

می‌خواهید شناسایی موجودیت نام‌گذاری شده را در متن (مانند «شخص»، «دولت» و غیره) انجام دهید.

راه‌حل

برای استخراج موجودیت‌ها از متن، از خط لوله[9] و مدل‌های پیش‌فرض شناسایی موجودیت با نام spaCy استفاده کنید:

[8] - subpopulations
[9] - pipeline

```
# Import libraries
import spacy


# Load the spaCy package and use it to parse the text
# make sure you have run "python -m spacy download en"
nlp = spacy.load("en_core_web_sm")
doc = nlp("Elon Musk offered to buy Twitter using $21B of his own money.")


# Print each entity
print(doc.ents)


# For each entity print the text and the entity label
for entity in doc.ents:
   print(entity.text, entity.label_, sep=",")
(Elon Musk, Twitter, 21B)
Elon Musk, PERSON
Twitter, ORG
21B, MONEY
```

بحث

شناسایی موجودیت نامی، فرآیند شناسایی موجودیت‌های خاص از متن است. ابزارهایی مانند spaCy ارائه دهنده‌ی روش‌هایی مانند خطوط لوله از پیش پیکربندی شده و حتی مدل‌های یادگیری ماشینی از پیش آموزش دیده یا تنظیم شده هستند که به راحتی می‌توانند این اشیاء[10] را شناسایی کنند. در این مورد، ما از spaCy برای شناسایی یک شخص ("Elon Musk")، سازمان ("Twitter") و ارزش پولی ("B۲۱") از متن خام استفاده می‌کنیم. با استفاده از این اطلاعات، می‌توانیم اطلاعات ساختاریافته را از داده‌های متنی بدون ساختار استخراج کنیم. سپس این اطلاعات را می‌توان در مدل‌های یادگیری ماشین پایین‌رونده[11] یا تجزیه و تحلیل داده‌ها استفاده کرد.

۶.۱۱ استفاده از بردارهای متن برای محاسبه شباهت متن در یک عبارت جستجو

مسئله

شما می‌خواهید از بردارهای tf-idf برای پیاده سازی یک تابع جستجوی متن در پایتون استفاده کنید.

راه‌حل

شباهت کسینوس بین بردارهای tf-idf را با استفاده از scikit-learn محاسبه کنید:

[10] - entities
[11] - downstream

```
# Load libraries
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel


# Create searchable text data
text_data = np.array(['I love Brazil. Brazil!',
                      'Sweden is best',

                      'Germany beats both'])


# Create the tf-idf feature matrix
tfidf = TfidfVectorizer()
feature_matrix = tfidf.fit_transform(text_data)


# Create a search query and transform it into a tf-idf vector
text = "Brazil is the best"
vector = tfidf.transform([text])


# Calculate the cosine similarities between the input vector and all other
  vectors
cosine_similarities = linear_kernel(vector, feature_matrix).flatten()


# Get the index of the most relevent items in order
related_doc_indicies = cosine_similarities.argsort()[:-10:-1]


# Print the most similar texts to the search query along with the cosine
  similarity
print([(text_data[i], cosine_similarities[i]) for i in related_doc_indicies])
[
   (
      'Sweden is best', 0.6666666666666666),
      ('I love Brazil. Brazil!', 0.5163977794943222),
      ('Germany beats both', 0.0
      )
]
```

بحث

بردارهای متنی برای موارد استفاده از NLP مانند موتورهای جستجو بسیار مفید هستند. پس از محاسبه بردارهای tf-idf برای مجموعه‌ای از جملات یا اسناد، می‌توانیم از همان شیء tfidf برای بردار کردن مجموعه‌های متن آینده استفاده کنیم. سپس، می‌توانیم شباهت کسینوس را بین بردار ورودی و ماتریس بردارهای دیگر محاسبه کرده و بر اساس مرتبط‌ترین اسناد مرتب کنیم.

شباهت کسینوس در محدوده [۰، ۱.۰] است که ۰ کمترین شباهت و ۱ بیشترین شباهت را دارد. از آنجایی که ما از بردارهای tf-idf برای محاسبه شباهت بین بردارها استفاده می‌کنیم، فراوانی وقوع یک کلمه نیز در نظر گرفته می‌شود. با این حال، با یک مجموعه کوچک (مجموعه ای از اسناد) حتی کلمات "متداول" ممکن است اغلب ظاهر نشوند. در این مثال، "سوئد

بهترین است" مرتبط ترین متن با عبارت جستجوی ما "برزیل بهترین است" است. از آنجایی که عبارت کوئری به برزیل اشاره می‌کند، ممکن است انتظار داشته باشیم "من عاشق برزیل هستم. برزیل!» مرتبط ترین مورد باشد. با این حال، "سوئد بهترین است" به دلیل کلمات "است" و "بهترین" شبیه ترین است. با افزایش تعداد اسنادی که به مجموعه خود اضافه می‌کنیم، کلماتی که اهمیت کمتری دارند، وزن کمتری خواهند داشت و تأثیر کمتری بر محاسبات شباهت کسینوس ما خواهند داشت.

همچنین ببینید:

- شباهت کسینوس، GeeksForGeeks

۶.۱۲ استفاده از یک طبقه بندی تحلیل احساسات[۱۲]

مسئله

شما می‌خواهید احساس برخی از متون را برای استفاده به عنوان یک ویژگی یا در تجزیه و تحلیل داده‌های پایین‌رونده[۱۳] طبقه بندی کنید.

راه‌حل

از طبقه‌بندی‌کننده احساسات کتابخانه transformers استفاده کنید.

[12] - Sentiment Analysis
[13] - downstream

```
# Import libraries
from transformers import pipeline


# Create an NLP pipeline that runs sentiment analysis
classifier = pipeline("sentiment-analysis")


# Classify some text
# (this may download some data and models the first time you run it)
sentiment_1 = classifier("I hate machine learning! It's the absolute worst.")
sentiment_2 = classifier(
        "Machine learning is the absolute"
        "bees knees I love it so much!"
)


#Print sentiment output
print(sentiment_1, sentiment_2)
[
        {
                'label': 'NEGATIVE',
                'score': 0.9998020529747009
        }
]

[
        {
                'label': 'POSITIVE',
                'score': 0.9990628957748413
        }
]
```

بحث

کتابخانه Transformers یک کتابخانه‌ی بسیار محبوب برای وظایف NLP است و شامل تعدادی API با استفاده آسان برای مدل‌های آموزشی یا استفاده از مدل‌های از پیش آموزش دیده است. ما در مورد NLP و این کتابخانه در فصل ۲۲ بیشتر صحبت خواهیم کرد، اما این مثال به عنوان مقدمه ای در سطح بالا برای قدرت استفاده از طبقه بندی کننده‌های از پیش آموزش دیده در خطوط لوله یادگیری ماشین شما برای تولید ویژگی ها، طبقه بندی متن یا تجزیه و تحلیل داده‌های بدون ساختار آورده شده است.

همچنین ببینید:

- تور سریع Hugging Face Transformers

۸.۱۵ استفاده از جاسازی‌های[۱۴] از پیش آموزش دیده به عنوان ویژگی‌ها

[14] - Embeddings

مسئله

می‌خواهید جاسازی‌های پیش‌آموزش‌شده را از یک مدل موجود در PyTorch بارگیری کنید و از آنها به عنوان ورودی یکی از مدل‌های خود استفاده کنید.

راه‌حل

از torchvision.models برای انتخاب یک مدل و سپس بازیابی یک جاسازی از آن، برای یک تصویر معین، استفاده کنید:

```
# Load libraries
import cv2
import numpy as np
import torch
from torchvision import transforms
import torchvision.models as models

# Load image
image_bgr = cv2.imread("images/plane.jpg", cv2.IMREAD_COLOR)

# Convert to pytorch data type
convert_tensor = transforms.ToTensor()
pytorch_image = convert_tensor(np.array(image_rgb))

# Load the pretrained model
model = models.resnet18(pretrained=True)

# Select the specific layer of the model we want output from
layer = model._modules.get('avgpool')

# Set model to evaluation mode
model.eval()

# Infer the embedding with the no_grad option
with torch.no_grad():
        embedding = model(pytorch_image.unsqueeze(0))

print(embedding.shape)
torch.Size([1, 1000])
```

بحث

در فضای ML، یادگیری انتقال اغلب به عنوان گرفتن اطلاعات آموخته شده از یک کار و استفاده از آن به عنوان ورودی برای کار دیگر تعریف می‌شود. به‌جای شروع از صفر، می‌توانیم از نمایش‌هایی[15] استفاده کنیم که قبلاً از مدل‌های تصویری بزرگ از پیش آموزش دیده (مانند ResNet) آموخته‌ایم تا در مدل‌های یادگیری ماشین خودمان شروع به کار کنیم. به‌طور شهودی‌تر،

[15] - representations

می‌توانید درک کنید که چگونه می‌توانیم از وزن‌های یک مدل آموزش دیده برای تشخیص گربه‌ها به عنوان شروع خوبی برای مدلی که می‌خواهیم برای تشخیص سگ‌ها آموزش دهیم، استفاده کنیم. با به اشتراک گذاشتن اطلاعات از یک مدل به مدل دیگر، می‌توانیم اطلاعاتی را که از سایر مجموعه داده‌ها و معماری‌های مدل به دست می‌آیند، بدون هزینه‌های سربار آموزش یک مدل از ابتدا به کار ببریم.

کل کاربرد یادگیری انتقالی در بینایی کامپیوتر خارج از محدوده این کتاب است. با این حال، راه‌های مختلفی وجود دارد که بتوانیم نمایش‌های مبتنی بر جاسازی‌ها را از تصاویر خارج از PyTorch استخراج کنیم. در TensorFlow، یکی دیگر از کتابخانه‌های رایج برای یادگیری عمیق، می‌توانیم از tensorflow_hub استفاده کنیم:

```
# Load libraries
import cv2
import tensorflow as tf
import tensorflow_hub as hub


# Load image
image_bgr = cv2.imread("images/plane.jpg", cv2.IMREAD_COLOR)
image_rgb = cv2.cvtColor(image_bgr, cv2.COLOR_BGR2RGB)


# Convert to tensorflow data type
tf_image = tf.image.convert_image_dtype([image_rgb], tf.float32)


# Create the model and get embeddings using the inception V1 model
embedding_model = hub.KerasLayer(
        "https://tfhub.dev/google/imagenet/inception_v1/feature_vector/5"
)
embeddings = embedding_model(tf_image)


# Print the shape of the embedding
print(embeddings.shape)
(1, 1024)
```

همچنین ببینید:

- آموزش PyTorch: آموزش انتقال برای بینایی کامپیوتر
- TensorFlow Hub

۸.۱۵ تشخیص اشیا با OpenCV

مسئله

شما می‌خواهید اشیاء را در تصاویر با استفاده از طبقه بندی کننده‌های آبشاری از پیش آموزش دیده شده با OpenCV شناسایی کنید.

راه‌حل

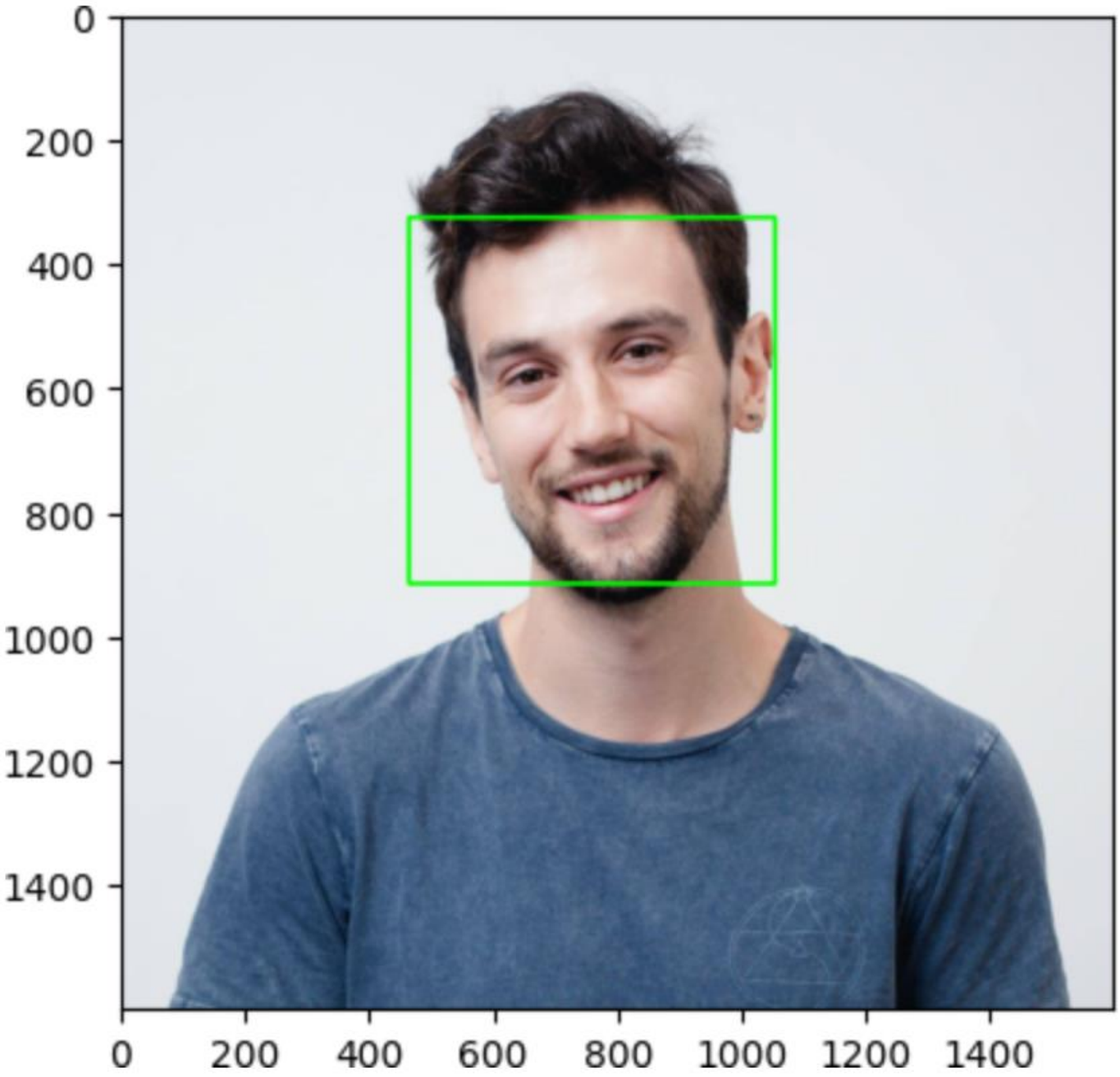
یکی از طبقه‌بندی‌کننده‌های آبشاری Haar OpenCV را دانلود و اجرا کنید. در این مورد، ما از یک مدل تشخیص چهره از پیش آموزش دیده شده برای تشخیص و ترسیم مستطیل دور یک چهره در یک تصویر استفاده می‌کنیم:

```
# Import libraries
import cv2
from matplotlib import pyplot as plt


# first run:
# mkdir models && cd models
# wget https://tinyurl.com/mrc6jwhp
face_cascade = cv2.CascadeClassifier()
face_cascade.load(
        cv2.samples.findFile(
                "models/haarcascade_frontalface_default.xml"
        )
)
# Load image
image_bgr = cv2.imread("images/kyle_pic.jpg", cv2.IMREAD_COLOR)
image_rgb = cv2.cvtColor(image_bgr, cv2.COLOR_BGR2RGB)


# Detect faces and draw a rectangle
faces = face_cascade.detectMultiScale(image_rgb)
for (x,y,w,h) in faces:
        cv2.rectangle(image_rgb, (x, y),
                (x + h, y + w),
                (0, 255, 0), 5)


# Show the image
plt.subplot(1, 1, 1)
plt.imshow(image_rgb)
plt.show()
```



بحث

طبقه‌بندی‌کننده‌های آبشاری Haar مدل‌های یادگیری ماشینی هستند که برای یادگیری مجموعه‌ای از ویژگی‌های تصویر (به‌ویژه ویژگی‌های Haar) استفاده می‌شوند که می‌توانند برای شناسایی اشیاء در تصاویر استفاده شوند. خود ویژگی‌ها، ویژگی‌های مستطیلی ساده ای هستند که با محاسبه تفاوت در مجموع بین مناطق مستطیلی تعیین می‌شوند. پس از آن، یک الگوریتم تقویت گرادیان برای یادگیری مهم ترین ویژگی‌ها و در نهایت ایجاد یک مدل نسبتا قوی با استفاده از طبقه بندی کننده‌های آبشاری اعمال می‌شود.

در حالی که جزئیات این فرآیند خارج از محدوده این کتاب است، قابل توجه است که این مدل‌های از پیش آموزش دیده شده را می‌توان به راحتی از مکان‌هایی مانند OpenCV GitHub به عنوان فایل XML دانلود کرد و بدون آموزش مدلی، روی تصاویر اعمال کرد. این در مواردی مفید است که می‌خواهید ویژگی‌های ساده تصویر باینری مانند contain_face (یا هر شیء دیگری) را به داده‌های خود اضافه کنید.

همچنین ببینید:

- آموزش OpenCV: طبقه بندی آبشاری

۸.۱۵ طبقه بندی تصاویر با Pytorch

مسئله

شما می‌خواهید تصاویر را با استفاده از مدل‌های آموزش عمیق از پیش آموزش دیده شده در Pytorch طبقه بندی کنید.

راه‌حل

از torchvision.models برای انتخاب یک مدل طبقه بندی تصویر از پیش آموزش دیده شده و تغذیه تصـویر از طریـق آن استفاده کنید:

```
# Load libraries
import cv2
import json
import numpy as np
import torch
from torchvision import transforms
from torchvision.models import resnet18
import urllib.request


# Get imagenet classes
with urllib.request.urlopen(
        "https://raw.githubusercontent.com/raghakot/keras-vis/master/resources/"
        ):
        imagenet_class_index = json.load(url)

# Instantiate pretrained model
model = resnet18(pretrained=True)


# Load image
image_bgr = cv2.imread("images/plane.jpg", cv2.IMREAD_COLOR)
image_rgb = cv2.cvtColor(image_bgr, cv2.COLOR_BGR2RGB)


# Convert to pytorch data type
convert_tensor = transforms.ToTensor()
pytorch_image = convert_tensor(np.array(image_rgb))


# Set model to evaluation mode
model.eval()
# Make a prediction
prediction = model(pytorch_image.unsqueeze(0))


# Get the index of the highest predicted probability
_, index = torch.max(prediction, 1)
# Convert that to a percentage value
percentage = torch.nn.functional.softmax(prediction, dim=1)[0] * 100


# Print the name of the item at the index along with the percent confidence
print(imagenet_class_index[str(index.tolist()[0])][1],
        percentage[index.tolist()[0]].item())
airship 6.0569939613342285
```

بحث

بسیاری از مدل‌های آموزش عمیق از پیش آموزش دیده شده، برای طبقه بندی تصاویر به راحتی از طریق PyTorch و TensorFlow در دسترس هستند. در این مثال، ما از ResNet18، یک معماری شبکه عصبی عمیق استفاده کردیم که بر روی مجموعه داده ImageNet با عمق ۱۸ لایه آموزش داده شده است. مدل‌های عمیق‌تر ResNet، مانند ResNet101 و ResNet152 نیز در Pytorch در دسترس هستند - و فراتر از آن، مدل‌های تصویری زیادی برای انتخاب وجود دارند. مدل‌های از پیش آموزش‌دیده شده، بر روی مجموعه داده ImageNet می‌توانند احتمالات پیش‌بینی‌شده را برای همه

کلاس‌های تعریف‌شده در متغیر imagenet_class_index در قطعه کد قبلی، که ما از GitHub دانلود کرده‌ایم، به صورت خروجی نمایش دهند.

به طور مثال تشخیص چهره در OpenCV (نگاه کنید به دستور العمل ۸.۱۶)، که ما می‌توانیم از کلاس‌های تصویر پیش‌بینی‌شده به‌عنوان ویژگی‌های پایین‌رونده[۱۶] برای مدل‌های آینده ML یا تگ‌های فراداده مفیدی که اطلاعات بیشتری را به تصاویر ما اضافه می‌کنند، استفاده کنیم.

همچنین ببینید:

- مستندات PyTorch: مدل‌ها و وزن‌های از پیش آموزش دیده

۱۴.۶ ارزیابی جنگل‌های تصادفی[۱۷] با خطاهای خارج از کیسه[۱۸]

مسئله

شما باید یک مدل جنگل تصادفی را بدون استفاده از اعتبارسنجی متقاطع ارزیابی کنید.

راه‌حل

امتیاز خارج از کیف مدل را محاسبه کنید:

```
# Load libraries
from sklearn.ensemble import RandomForestClassifier
from sklearn import datasets

# Load data
iris = datasets.load_iris()
features = iris.data
target = iris.target

# Create random forest classifier object
randomforest = RandomForestClassifier(
        random_state=0, n_estimators=1000, oob_score=True, n_jobs=-1)

# Train model
model = randomforest.fit(features, target)

# View out-of-bag-error
randomforest.oob_score_
0.9533333333333334
```

بحث

[16] - downstream
[17] - Random Forests
[18] - Out-of-Bag Erros

در جنگل‌های تصادفی، هر درخت تصمیم با استفاده از زیرمجموعه‌ای از مشاهدات راه‌اندازی، آموزش داده می‌شود. این بدان معناست که برای هر درخت یک زیر مجموعه جداگانه از مشاهدات وجود دارد که برای آموزش آن درخت استفاده نمی‌شود. به این مشاهدات خارج از کیسه (OOB) می‌گویند. ما می‌توانیم از مشاهدات OOB به عنوان یک مجموعه تست برای ارزیابی عملکرد جنگل تصادفی خود استفاده کنیم.

برای هر مشاهده، الگوریتم یادگیری، ارزش واقعی مشاهدات را با پیش‌بینی زیرمجموعه‌ای از درختانی که با استفاده از آن مشاهده آموزش ندیده‌اند، مقایسه می‌کند. امتیاز کلی محاسبه می‌شود و یک معیار واحد از عملکرد یک جنگل تصادفی را ارائه می‌دهد. تخمین امتیاز OOB جایگزینی برای اعتبارسنجی متقابل است.

در scikit-learn، می‌توانیم امتیازهای OOB یک جنگل تصادفی را با تنظیم oob_score=True در شیء جنگل تصادفی (یعنی RandomForestClassifier) محاسبه کنیم. امتیاز را می‌توان با استفاده از oob_score_ بازیابی کرد.

۱۴.۱۲ باران مدل XGBoost

مسئله

شما باید یک مدل درختی با قدرت پیش بینی بالا آموزش دهید.

راه‌حل

از کتابخانه xgboost پایتون استفاده کنید:

```
# Load libraries
import xgboost as xgb
from sklearn import datasets, preprocessing
from sklearn.metrics import classification_report
from numpy import argmax

# Load data
iris = datasets.load_iris()
features = iris.data
target = iris.target

# Create dataset
xgb_train = xgb.DMatrix(features, label=target)

# Define parameters
param = {
        'objective': 'multi:softprob',
        'num_class': 3
}

# Train model
gbm = xgb.train(param, xgb_train)

# Get predictions
predictions = argmax(gbm.predict(xgb_train), axis=1)

# Get a classification report
print(classification_report(target, predictions))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        50
           1       1.00      0.96      0.98        50
           2       0.96      1.00      0.98        50

    accuracy                           0.99       150
   macro avg       0.99      0.99      0.99       150
weighted avg       0.99      0.99      0.99       150
```

بحث

XGBoost (که مخفف Extreme Gradient Boosting است) یک الگوریتم تقویت گرادیان بسیار محبوب در فضای یادگیری ماشینی است. اگرچه همیشه یک مدل مبتنی بر درخت نیست، اما اغلب برای مجموعه‌ای از درخت‌های تصمیم استفاده می‌شود. به دلیل موفقیت گسترده در وب سایت مسابقه یادگیری ماشین Kaggle، محبوبیت زیادی به دست آورد و از آن زمان به بعد الگوریتمی‌قابل اعتماد برای بهبود عملکرد فراتر از جنگل‌های تصادفی معمولی یا ماشین‌های تقویت شده گرادیان بوده است.

اگرچه XGBoost به دلیل محاسباتی فشرده شناخته شده است، بهینه سازی عملکرد محاسباتی (مانند پشتیبانی از GPU) در چند سال گذشته، تکرار سریع با XGBoost را به طور قابل توجهی آسان کرده است، و زمانی که عملکرد آماری الزامی‌است، به عنوان یک الگوریتم رایج انتخاب می‌شود.

همچنین ببینید:

- مستندات XGBoost

۱۴.۶ بهبود عملکرد بلادرنگ با LightGBM

مسئله

شما باید یک مدل مبتنی بر درخت تقویت شده با گرادیان را آموزش دهید که از نظر محاسباتی بهینه شده باشد.

راه‌حل

از کتابخانه ماشینی تقویت شده با گرادیان lightgbm استفاده کنید:

```
# Load libraries
import lightgbm as lgb
from sklearn import datasets, preprocessing
from sklearn.metrics import classification_report
from numpy import argmax


# Load data
iris = datasets.load_iris()
features = iris.data
target = iris.target


# Create dataset
lgb_train = lgb.Dataset(features, target)


# Define parameters
params = {
        'objective': 'multiclass',
        'num_class': 3,
        'verbose': -1,
}



# Train model
gbm = lgb.train(params, lgb_train)


# Get predictions
predictions = argmax(gbm.predict(features), axis=1)


# Get a classification report
print(classification_report(target, predictions))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	50
1	1.00	1.00	1.00	50
2	1.00	1.00	1.00	50
accuracy			1.00	150
macro avg	1.00	1.00	1.00	150
weighted avg	1.00	1.00	1.00	150

بحث

کتابخانه lightgbm برای ماشین‌های تقویت‌شده‌ی گرادیان استفاده می‌شود و برای زمان آموزش، استنتاج و پشتیبانی GPU بسیار بهینه شده است. به عنوان یک نتیجه از کارایی محاسباتی آن، اغلب در تولید و در تنظیمات در مقیاس بزرگ استفاده

می‌شود. اگرچه استفاده از مدل‌های Sikit-Learn معمولا آسان‌تر است، برخی از کتابخانه‌ها، مانند lightgbm، می‌توانند زمانی مفید باشند که داده‌های بزرگ داشته باشیم یا از لحاظ زمان‌های آموزشی دقیق برای مدل، محدود شده‌اید.

همچنین ببینید:

- مستندات LightGBM
- مستندات CatBoost (یکی دیگر از کتابخانه‌های بهینه شده برای GBM)

۱۵.۵ پیدا کردن نزدیکترین همسایه‌های تقریبی (ANN)

مسئله

می‌خواهید نزدیک‌ترین همسایگان را برای داده‌های بزرگ با تأخیر کم را واکِشی[19] کنید:

راه‌حل

از جستجوی تقریبی نزدیکترین همسایگان (ANN) با کتابخانه faiss فیس بوک استفاده کنید:

19 - fetch

```
# Load libraries
import faiss
import numpy as np
from sklearn import datasets
from sklearn.neighbors import NearestNeighbors
from sklearn.preprocessing import StandardScaler

# Load data
iris = datasets.load_iris()
features = iris.data

# Create standardizer
standardizer = StandardScaler()

# Standardize features
features_standardized = standardizer.fit_transform(features)

# Set faiss parameters
n_features = features_standardized.shape[1]
nlist = 3
k = 2

# Create an IVF index
quantizer = faiss.IndexFlatIP(n_features)
index = faiss.IndexIVFFlat(quantizer, n_features, nlist)

# Train the index and add feature vectors
index.train(features_standardized)
index.add(features_standardized)

# Create an observation
new_observation = np.array([[ 1, 1, 1, 1]])

# Search the index for the 2 nearest neighbors
distances, indices = index.search(new_observation, k)

# Show the feature vectors for the two nearest neighbors
np.array([list(features_standardized[i]) for i in indices[0]])
array([[1.03800476, 0.55861082, 1.10378283, 1.18556721],
       [0.79566902, 0.32841405, 0.76275827, 1.05393502]])
```

بحث

KNN یک رویکرد عالی برای یافتن مشابه ترین مشاهدات در مجموعه ای از داده‌های کوچک است. با این حال، با افزایش اندازه داده‌های ما، زمان محاسبه فاصله بین هر مشاهده و سایر نقاط مجموعه داده‌ی ما نیز افزایش می‌یابد. سیستم‌های ML در مقیاس بزرگ مانند موتورهای جستجو یا توصیه، اغلب از نوعی معیار تشابه برداری برای بازیابی مشاهدات مشابه استفاده

می‌کنند. اما در مقیاس بزرگ در زمان واقعی، جایی کـه مـا بـه نتـایج کمتـر از ۱۰۰ میلی‌ثانیـه نیـاز داریـم، اجـرای KNN غیرممکن می‌شود.

ANN به ما کمک می‌کند تا با قربانی کردن مقداری از کیفیتِ جستجوی دقیقِ نزدیکترین همسایگان به نفع سرعت، بر ایـن مشکل غلبه کنیم. این بدان معناست که اگرچه ترتیب و موارد در ۱۰ همسایه اول یک جستجوی ANN ممکن اسـت بـا ۱۰ نتیجه اول از یک جستجوی دقیق KNN مطابقت نداشته باشد، ما آن ۱۰ همسایه اول را بسیار سریعتر دریافت می‌کنیم.

در این مثال، ما از یک رویکرد ANN به نام شاخص فایل معکوس (IVF) استفاده می‌کنیم. این رویکرد با اسـتفاده از خوشـه بندی برای محدود کردن دامنه فضای جستجو برای جستجوی نزدیکترین همسـایگان مـا کـار می‌کنـد. IVF از تسسـلات[۲۰] Voronoi برای تقسیم فضای جستجوی ما به تعـدادی منطقـه (یـا خوشـه) مجـزا اسـتفاده می‌کنـد. و وقتـی بـرای یـافتن نزدیک‌ترین همسایگان می‌رویم، از تعداد محدودی از خوشه‌ها بازدید می‌کنیم تا مشاهدات مشـابهی را پیـدا کنـیم، بـرخلاف مقایسه بین هر نقطه از مجموعه داده‌هایمان با همدیگر.

نحوه ایجاد مجموعه‌های Voronoi از داده‌ها به بهترین وجه با استفاده از داده‌های ساده، قابل مشاهده است. همانطور کـه در شکل ۱۵-۱ نشان داده شده است، یک نمودار پراکنده از داده‌های تصادفی که در دو بعد مشاهده می‌شود، در نظر بگیرید.

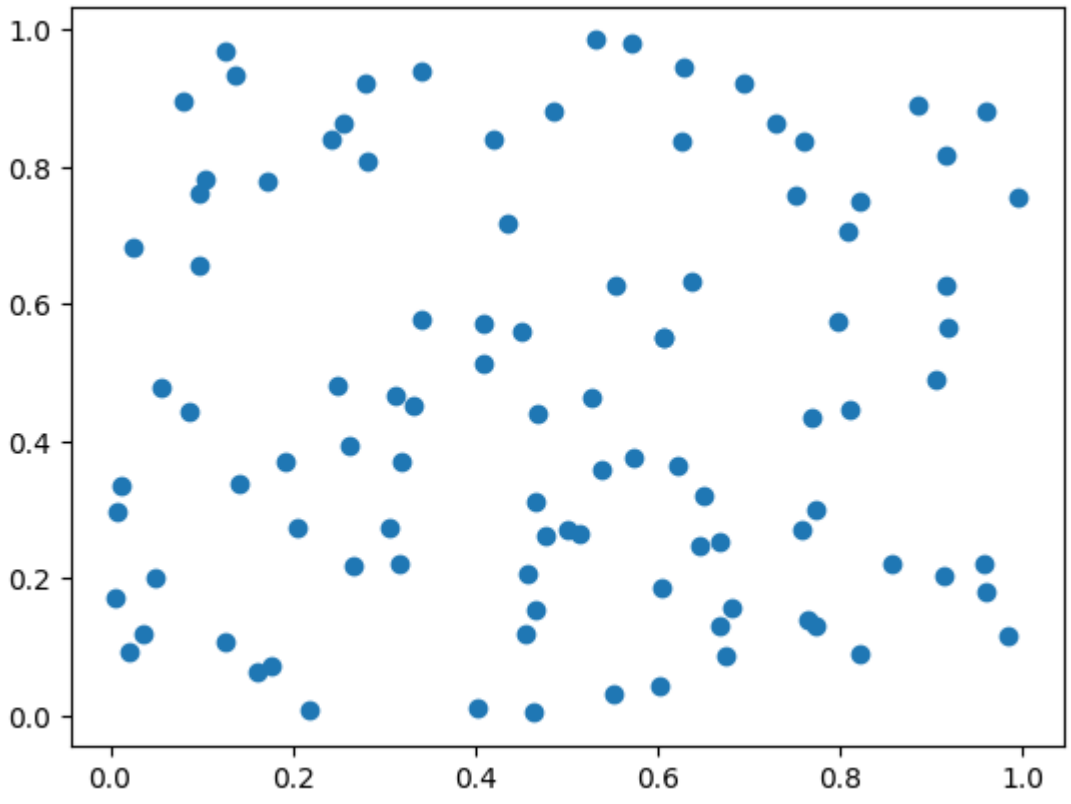


Figure 15-1. A scatter plot of randomly generated two-dimensional data

با استفاده از Tessellations Voronoi، می‌توانیم تعدادی زیرفضا ایجاد کنیم که هر کدام از آنها فقط شامل یک زیرمجموعه کوچک از کل مشاهداتی است که می‌خواهیم در آن جستجو کنیم، همانطور که در شکل ۱۵-۲ نشان داده شده است.

[20] - tessellations

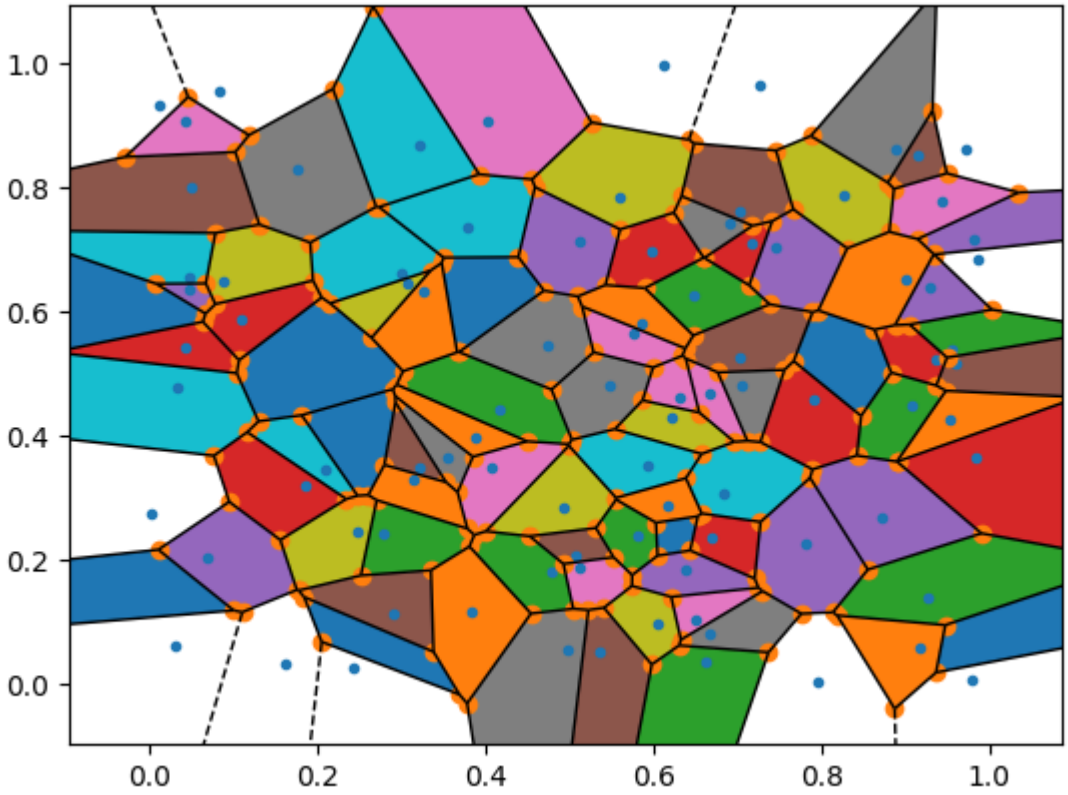


Figure 15-2. Randomly generated two-dimensional data separated into a number of different subspaces

پارامتر nlist در کتابخانه Faiss به ما امکان می‌دهد تعداد خوشه‌هایی را که می‌خواهیم ایجاد کنیم تعریف کنیم. یک پارامتر اضافی، nprobe، می‌تواند در زمان پرس و جو برای تعریف تعداد خوشه‌هایی که می‌خواهیم جستجو کنیم تا نزدیک‌ترین همسایه‌ها را برای یک مشاهده مشخص بازیابی کنیم، استفاده شود. افزایش هر دو nlist و nprobe می‌تواند منجر به همسایگان با کیفیت بالاتر به قیمت تلاش محاسباتی بزرگتر و در نتیجه زمان اجرای طولانی تر برای شاخص‌های IVF شود. کاهش هر یک از این پارامترها، اثر معکوس خواهد داشت و کد شما سریعتر اجرا می‌شود اما همچنین خطر بازگشت نتایج با کیفیت پایین تر را خواهید داشت.

توجه داشته باشید که این مثال دقیقا همان خروجی دستور اول در این فصل را برمی‌گرداند. این به این دلیل است که ما با داده‌های بسیار کوچک کار می‌کنیم و فقط از سه خوشه استفاده می‌کنیم، که باعث می‌شود نتایج ANN ما به طور قابل‌توجهی با نتایج KNN ما متفاوت نباشد.

همچنین ببینید:

- نزدیکترین نمایه‌های همسایه برای جستجوی شباهت (انواع شاخص ANN مختلف)

۱۵.۶ ارزیابی تقریبی نزدیکترین همسایگان

مسئله

می‌خواهید ببینید ANN شما چگونه با نزدیک‌ترین همسایگان (KNN) مقایسه می‌شود:

راه‌حل

فراخوان[۲۱] k@ نزدیکترین همسایه ANN را در مقایسه با KNN محاسبه کنید:

[21] - recall

```
# Load libraries
import faiss
import numpy as np
from sklearn import datasets
from sklearn.neighbors import NearestNeighbors
from sklearn.preprocessing import StandardScaler

# Number of nearest neighbors
k = 10

# Load data
iris = datasets.load_iris()
features = iris.data

# Create standardizer
standardizer = StandardScaler()

# Standardize features
features_standardized = standardizer.fit_transform(features)

# Create KNN with 10 NN
nearest_neighbors = NearestNeighbors(n_neighbors=k).fit(features_standardized)

# Set faiss parameters
n_features = features_standardized.shape[1]
nlist = 3

# Create an IVF index
quantizer = faiss.IndexFlatIP(n_features)
index = faiss.IndexIVFFlat(quantizer, n_features, nlist)

# Train the index and add feature vectors
index.train(features_standardized)
index.add(features_standardized)
index.nprobe = 1

# Create an observation
new_observation = np.array([[ 1, 1, 1, 1]])

# Find distances and indices of the observation's exact nearest neighbors
knn_distances, knn_indices = nearest_neighbors.kneighbors(new_observation)

# Search the index for the two nearest neighbors
ivf_distances, ivf_indices = index.search(new_observation, k)
```

بحث

Recall @k به سادگی به عنوان تعداد آیتم‌هایی است که توسط ANN در برخی از k نزدیک‌ترین همسایه‌هایی که در نزدیک‌ترین همسایه‌ها در همان k، تقسیم بر k ظاهر می‌شوند؛ بازگردانده می‌شود. در این مثال، در ۱۰ نزدیکترین همسایه، ما ۱۰۰٪ فراخوانی داریم، به این معنی که ANN ما، همان شاخص های KNN ما را در k=10 برمی گرداند (البته نه لزوماً به همان ترتیب).

یادآوری یا فراخوان[۲۲] یک معیار رایج برای استفاده در هنگام ارزیابی شبکه های عصبی مصنوعی در برابر نزدیکترین همسایگان است.

همچنین ببینید:

- یادداشت Google در ANN برای سرویس موتور تطبیق Vertex

[22] - recall