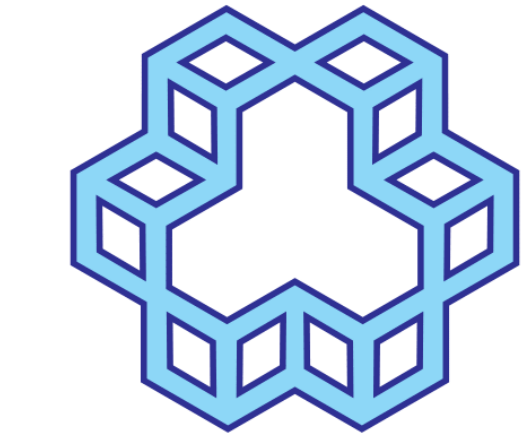


دانشگاه صنعتی خواجه نصیرالدین طوسی

دانشکده مهندسی برق- گروه مهندسی کنترل

درس یادگیری ماشین

گزارش مینی پروژه شماره سه

نام و نام خانوادگی	علیرضا جهانی
شمارهٔ دانشجویی	۴۰۲۲۳۰۴۴
استاد درس	دکتر علیاری
تاریخ	بهمن‌ماه ۱۴۰۲

colab github

فهرست مطالب

۱ سوال اول

۱.۱ بخش اول سوال اول

ابتدا دیتاست را فرا میخوانیم:

```python
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE

import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from mpl_toolkits.mplot3d import Axes3D
import seaborn as sns

dataset = load_iris()
features = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width']
target = 'species'
iris = pd.DataFrame(
    dataset.data,
    columns=features)
iris[target] = dataset.target
iris.head()
```

✓ 13.8s Python

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

این دیتاست شامل ۱۵۰ نمونه از گل‌های زنبق و چهار ویژگی مختلف از هر گل (طول و عرض کاسبرگ و گلبرگ) به همراه نوع گل (یکی از سه گونه Setosa، Versicolor (Virginica است.

Dataset statistics	
Number of variables	5
Number of observations	150
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	1
Duplicate rows (%)	0.7%
Total size in memory	5.4 KiB
Average record size in memory	36.9 B

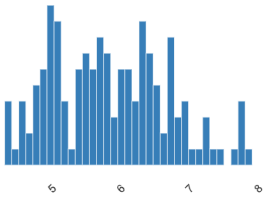
Variable types	
Numeric	4
Categorical	1

دیتاست شامل ۴ ویژگی با مقادیر عددی و یک ستون با مقادیر کاتالوگی مباشد که همان لیبل ها است. در شکل ۱.۱ جزییات دیتاست آورده شده است.

sepal_length
Real number (ℝ)

HIGH CORRELATION

Distinct	35	Minimum	4.3
Distinct (%)	23.3%	Maximum	7.9
Missing	0	Zeros	0
Missing (%)	0.0%	Zeros (%)	0.0%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	5.8433333	Memory size	1.3 KiB



اطلاعات آماری

- Distinct: تعداد مقادیر یکتای موجود در این ویژگی ۳۵ مقدار است.
- Distinct (%): این تعداد مقادیر یکتا معادل ۲۳.۳% از کل داده‌هاست.
- Missing: هیچ مقداری در این ویژگی گم نشده است (Missing: ۰).
- Infinite: هیچ مقداری در این ویژگی به صورت بی‌نهایت وجود ندارد (Infinite: ۰).
- Mean: میانگین مقدار این ویژگی ۵.۸۴۳۳ است.
- Minimum: کمترین مقدار این ویژگی ۳.۴ است.
- Maximum: بیشترین مقدار این ویژگی ۹.۷ است.
- Zeros: هیچ مقداری در این ویژگی صفر نیست (Zeros: ۰).
- Negative: هیچ مقداری در این ویژگی منفی نیست (Negative: ۰).
- Memory size: این ویژگی حجمی معادل ۳.۱ کیلوبایت از حافظه را اشغال می‌کند.

تحلیل هیستوگرام

هیستوگرام نشان‌دهنده توزیع مقادیر ویژگی `sepal_length` است. نکات زیر از هیستوگرام قابل برداشت است:

- مقادیر بین ۵ و ۶ بیشترین فراوانی را دارند.
- توزیع مقادیر تقریباً متقارن است، با کمی تمایل به سمت مقادیر بالاتر (راست‌چوله).
- هیچ مقدار غیرعادی یا نویزی در داده‌ها مشاهده نمی‌شود.

همبستگی بالا

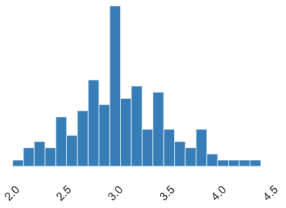
نشانگر همبستگی بالا به این معناست که این ویژگی ممکن است با یکی از ویژگی‌های دیگر یا متغیر هدف همبستگی زیادی داشته باشد، که می‌تواند به پیش‌بینی متغیر هدف کمک کند.

نتیجه‌گیری

ویژگی `sepal_length` به خوبی توزیع شده و اطلاعات زیادی از آن قابل استخراج است. این ویژگی می‌تواند در مدل‌های یادگیری ماشین به عنوان یکی از ورودی‌های مهم استفاده شود. برای تحلیل بیشتر می‌توان این ویژگی را در ترکیب با سایر ویژگی‌ها و متغیر هدف بررسی کرد تا الگوها و ارتباطات بیشتری کشف شود.

sepal_width
Real number (ℝ)

Distinct	23	Minimum	2
Distinct (%)	15.3%	Maximum	4.4
Missing	0	Zeros	0
Missing (%)	0.0%	Zeros (%)	0.0%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	3.0573333	Memory size	1.3 KiB



اطلاعات آماری

- Distinct: تعداد مقادیر یکتای موجود در این ویژگی ۲۳ مقدار است.
- Distinct (%): این تعداد مقادیر یکتا معادل ۱۵.۳% از کل داده‌هاست.
- Missing: هیچ مقداری در این ویژگی گم نشده است (Missing: ۰).
- Infinite: هیچ مقداری در این ویژگی به صورت بی‌نهایت وجود ندارد (Infinite: ۰).
- Mean: میانگین مقدار این ویژگی ۳.۰۵۷۳ است.
- Minimum: کمترین مقدار این ویژگی ۲ است.
- Maximum: بیشترین مقدار این ویژگی ۴.۴ است.
- Zeros: هیچ مقداری در این ویژگی صفر نیست (Zeros: ۰).
- Negative: هیچ مقداری در این ویژگی منفی نیست (Negative: ۰).
- Memory size: این ویژگی حجمی معادل ۳.۱ کیلوبایت از حافظه را اشغال می‌کند.

تحلیل هیستوگرام

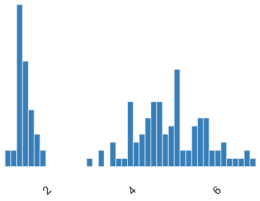
هیستوگرام نشان‌دهنده توزیع مقادیر ویژگی `sepal_width` است. نکات زیر از هیستوگرام قابل برداشت است:

- مقادیر حول ۳ بیشترین فراوانی را دارند.
- توزیع مقادیر به صورت تقریبی نرمال است، با کمی تمرکز در مقادیر میانی.
- هیچ مقدار غیرعادی یا نویزی در داده‌ها مشاهده نمی‌شود.

petal_length
Real number (ℝ)

HIGH CORRELATION

Distinct	43	Minimum	1
Distinct (%)	28.7%	Maximum	6.9
Missing	0	Zeros	0
Missing (%)	0.0%	Zeros (%)	0.0%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	3.758	Memory size	1.3 KiB



اطلاعات آماری

- Distinct: تعداد مقادیر یکتای موجود در این ویژگی ۴۳ مقدار است.
- Distinct (%): این تعداد مقادیر یکتا معادل ۲۸.۷% از کل داده‌هاست.
- Missing: هیچ مقداری در این ویژگی گم نشده است (Missing: ۰).
- Infinite: هیچ مقداری در این ویژگی به صورت بی‌نهایت وجود ندارد (Infinite: ۰).
- Mean: میانگین مقدار این ویژگی ۷۵۸.۳ است.
- Minimum: کمترین مقدار این ویژگی ۱ است.
- Maximum: بیشترین مقدار این ویژگی ۹.۶ است.
- Zeros: هیچ مقداری در این ویژگی صفر نیست (Zeros: ۰).
- Negative: هیچ مقداری در این ویژگی منفی نیست (Negative: ۰).
- Memory size: این ویژگی حجمی معادل ۳.۱ کیلوبایت از حافظه را اشغال می‌کند.

تحلیل هیستوگرام

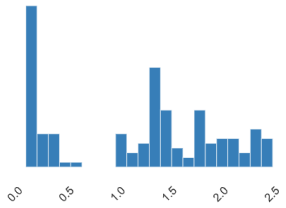
هیستوگرام نشان‌دهنده توزیع مقادیر ویژگی `petal_length` است. نکات زیر از هیستوگرام قابل برداشت است:

- مقادیر به دو بخش عمده تقسیم شده‌اند: یک بخش حول مقدار ۱ و بخش دیگر از ۳ تا ۶.
- توزیع مقادیر به صورت دوپیکه است (Bimodal Distribution) که نشان‌دهنده وجود دو گروه مختلف از داده‌ها است.
- هیچ مقدار غیرعادی یا نویزی در داده‌ها مشاهده نمی‌شود.

petal_width
Real number (ℝ)

HIGH CORRELATION

Distinct	22	Minimum	0.1
Distinct (%)	14.7%	Maximum	2.5
Missing	0	Zeros	0
Missing (%)	0.0%	Zeros (%)	0.0%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	1.1993333	Memory size	1.3 KiB



اطلاعات آماری

- Distinct: تعداد مقادیر یکتای موجود در این ویژگی ۲۲ مقدار است.
- Distinct (%): این تعداد مقادیر یکتا معادل ۱۴.۷% از کل داده‌هاست.
- Missing: هیچ مقداری در این ویژگی گم نشده است (Missing: ۰).
- Infinite: هیچ مقداری در این ویژگی به صورت بی‌نهایت وجود ندارد (Infinite: ۰).
- Mean: میانگین مقدار این ویژگی ۱۹۹۳.۱ است.
- Minimum: کمترین مقدار این ویژگی ۱.۰ است.
- Maximum: بیشترین مقدار این ویژگی ۵.۲ است.
- Zeros: هیچ مقداری در این ویژگی صفر نیست (Zeros: ۰).
- Negative: هیچ مقداری در این ویژگی منفی نیست (Negative: ۰).
- Memory size: این ویژگی حجمی معادل ۳.۱ کیلوبایت از حافظه را اشغال می‌کند.

تحلیل هیستوگرام

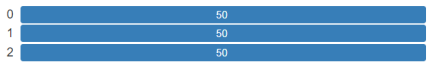
هیستوگرام نشان‌دهنده توزیع مقادیر ویژگی `petal_width` است. نکات زیر از هیستوگرام قابل برداشت است:

- مقادیر به دو بخش عمده تقسیم شده‌اند: یک بخش حول مقدار کمتر از ۵.۰ و بخش دیگر حول مقادیر بین ۱ تا ۲.
- توزیع مقادیر به صورت دوپیکه است (Bimodal Distribution) که نشان‌دهنده وجود دو گروه مختلف از داده‌ها است.
- هیچ مقدار غیرعادی یا نویزی در داده‌ها مشاهده نمی‌شود.

species
Categorical

HIGH CORRELATION UNIFORM

Distinct	3
Distinct (%)	2.0%
Missing	0
Missing (%)	0.0%
Memory size	1.3 KiB



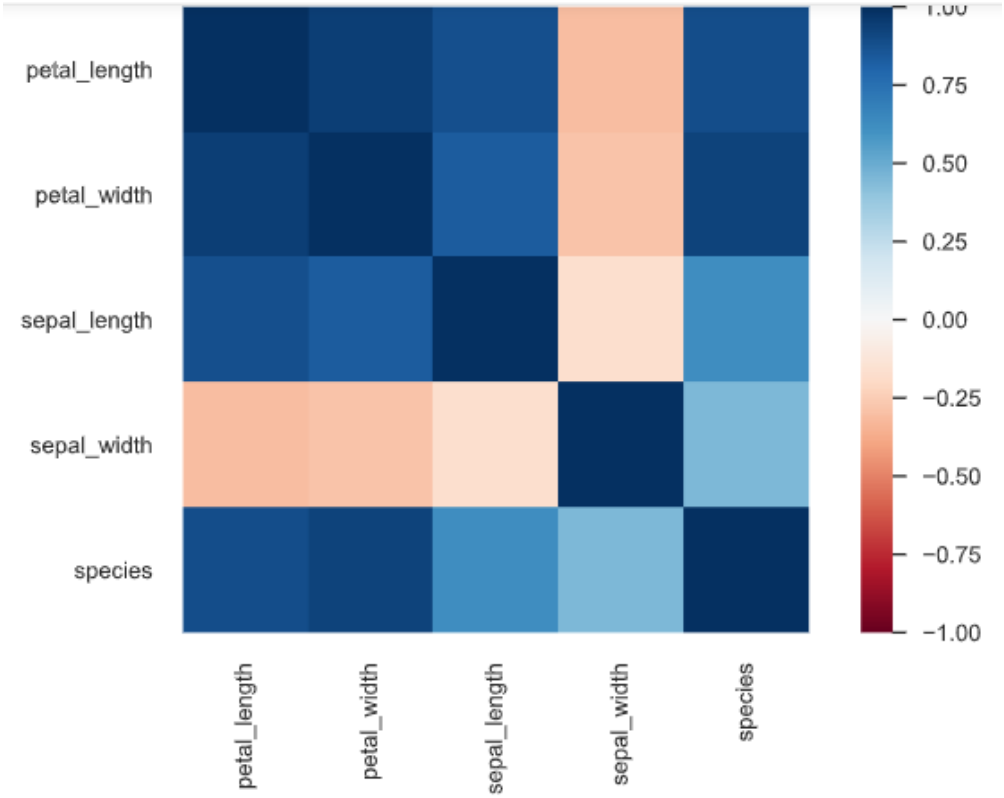
اطلاعات آماری

- Distinct: تعداد مقادیر یکتای موجود در این ویژگی ۳ مقدار است.
- Distinct (%): این تعداد مقادیر یکتا معادل ۲.۰% از کل داده‌هاست.
- Missing: هیچ مقداری در این ویژگی گم نشده است (Missing: ۰).
- size Memory: این ویژگی حجمی معادل ۳.۱ کیلوبایت از حافظه را اشغال می‌کند.

تحلیل

- Correlation High: این ویژگی همبستگی بالایی با متغیرهای دیگر دارد که نشان‌دهنده تأثیر بالای آن بر روی سایر ویژگی‌ها یا برعکس است.
- Uniform: داده‌ها به طور یکنواخت توزیع شده‌اند؛ هر یک از گونه‌ها (۰، ۱، ۲) دارای ۵۰ نمونه هستند.

این ویژگی به دلیل یکنواخت بودن تعداد نمونه‌ها در هر دسته، برای تحلیل‌های دسته‌بندی و یادگیری ماشین مناسب است. هیچ داده گمشده یا نویزی وجود ندارد که دقت مدل را تحت تأثیر قرار دهد.



تحلیل ماتریس همبستگی ویژگی‌های دیتاست Iris

نمودار ارائه شده یک ماتریس همبستگی (Correlation Matrix) را برای ویژگی‌های مختلف دیتاست Iris نشان می‌دهد. این ماتریس همبستگی بین ویژگی‌ها را با استفاده از رنگ‌های مختلف نشان می‌دهد، به طوری که رنگ‌های نزدیک به قرمز نشان‌دهنده همبستگی منفی و رنگ‌های نزدیک به آبی نشان‌دهنده همبستگی مثبت هستند. در زیر تحلیل این ماتریس همبستگی ارائه شده است:

تحلیل همبستگی‌ها

- petal_length و petal_width:
  - همبستگی مثبت قوی با هم دارند (رنگ آبی تیره).
  - نشان‌دهنده این است که با افزایش طول گلبرگ، عرض گلبرگ نیز افزایش می‌یابد.
- sepal_length و petal_length:
  - همبستگی مثبت متوسط دارند (رنگ آبی روشن).
  - نشان‌دهنده این است که با افزایش طول کاسبرگ، طول گلبرگ نیز افزایش می‌یابد.
- sepal_width و petal_length:

- همبستگی منفی دارند (رنگ نارنجی روشن).
- نشان‌دهنده این است که با افزایش طول گلبرگ، عرض کاسبرگ کاهش می‌یابد.

- sepal_width و petal_width:
  - همبستگی منفی دارند (رنگ نارنجی روشن).
  - نشان‌دهنده این است که با افزایش عرض گلبرگ، عرض کاسبرگ کاهش می‌یابد.
- species و petal_length و petal_width:
  - همبستگی مثبت قوی با طول و عرض گلبرگ دارد (رنگ آبی).
  - نشان‌دهنده این است که گونه‌های مختلف گل‌ها تفاوت‌های قابل توجهی در طول و عرض گلبرگ دارند.
- species و sepal_length:
  - همبستگی مثبت متوسط دارد (رنگ آبی روشن).
  - نشان‌دهنده این است که گونه‌های مختلف گل‌ها تفاوت‌های قابل توجهی در طول کاسبرگ دارند.
- species و sepal_width:
  - همبستگی منفی ضعیف دارد (رنگ نارنجی کم‌رنگ).
  - نشان‌دهنده این است که گونه‌های مختلف گل‌ها تفاوت‌های کمی در عرض کاسبرگ دارند.

ابتدا ۲ ویژگی اول را نمایش می‌دهیم:

تحلیل روش‌های کاهش بعد برای دیتاست Iris

با توجه به تحلیل ویژگی‌های دیتاست Iris و ماتریس همبستگی، ما چندین روش کاهش بعد را بررسی خواهیم کرد و مشخص خواهیم کرد که کدام یک از آن‌ها کارآمدتر است. روش‌های کاهش بعد مورد بررسی عبارتند از:

۱. Principal Component Analysis (PCA)

PCA یک روش غیرنظارتی برای کاهش بعد است که سعی می‌کند متغیرهای مرتبط را به یک مجموعه کوچکتر از متغیرهای غیرمرتبط تبدیل کند. PCA با استفاده از تجزیه و تحلیل مقادیر ویژه و بردارهای ویژه ماتریس همبستگی یا کواریانس داده‌ها، مولفه‌های اصلی را استخراج می‌کند.
مزایا:

- کاهش بعد غیرنظارتی.
- شناسایی الگوهای خطی.
- محاسبات نسبتاً ساده.

معایب:

- فقط الگوهای خطی را شناسایی می‌کند.
- بدون در نظر گرفتن متغیرهای هدف.

۲. Linear Discriminant Analysis (LDA)

LDA یک روش نظارتی برای کاهش بعد است که سعی می‌کند فضای ویژگی‌ها را به گونه‌ای تغییر دهد که حداکثر تفکیک بین کلاس‌ها حاصل شود. LDA با توجه به ماتریس پراکندگی بین کلاسی و درون کلاسی، مولفه‌های تفکیک‌کننده را استخراج می‌کند.

مزایا:

- مناسب برای مسائل طبقه‌بندی.
- در نظر گرفتن متغیرهای هدف.
- شناسایی الگوهای خطی.

معایب:

- فقط الگوهای خطی را شناسایی می‌کند.
- نیاز به داشتن اطلاعات کلاس‌ها.

۳. Autoencoders (AE)

AE یک نوع شبکه عصبی عمیق است که سعی می‌کند داده‌ها را به یک فضای ویژگی کمتر با حفظ بیشترین اطلاعات فشرده کند. AE از دو بخش اصلی تشکیل شده است: انکودر و دکودر.

مزایا:

- قابلیت شناسایی الگوهای غیرخطی.
- مناسب برای داده‌های پیچیده.

معایب:

- نیاز به تنظیمات و پارامترهای بیشتر.
- محاسبات پیچیده‌تر و زمان‌برتر.

۴. Variational Autoencoders (VAE)

VAE یک نسخه پیشرفته‌تر از AE است که از رویکرد احتمالاتی برای یادگیری توزیع‌های نهفته استفاده می‌کند. VAE سعی می‌کند داده‌ها را به یک فضای نهفته که توزیع گاوسی دارد، فشرده کند.

مزایا:

- قابلیت شناسایی الگوهای غیرخطی.
- تولید داده‌های جدید مشابه داده‌های اصلی.

معایب:

- نیاز به تنظیمات و پارامترهای بیشتر.
- محاسبات پیچیده‌تر و زمان‌برتر.

مقایسه روش‌ها

- PCA و LDA برای شناسایی الگوهای خطی مفید هستند. با توجه به این که ما یک ماتریس همبستگی با همبستگی‌های قوی و خطی داریم، هر دو روش می‌توانند مفید باشند.
- AE و VAE برای داده‌های پیچیده و غیرخطی مفید هستند. اگر داده‌ها دارای الگوهای غیرخطی باشند، این روش‌ها مناسب‌تر خواهند بود. در این دیتاست، VAE می‌تواند برای تولید داده‌های جدید و افزایش دقت مدل‌ها مفید باشد.

از آنجایی که نیازی نیست مدل را پیچیده کرد از روش PCA استفاده میکنیم.

```
pca = PCA(n_components=2)
points = pca.fit_transform(iris[features])
```

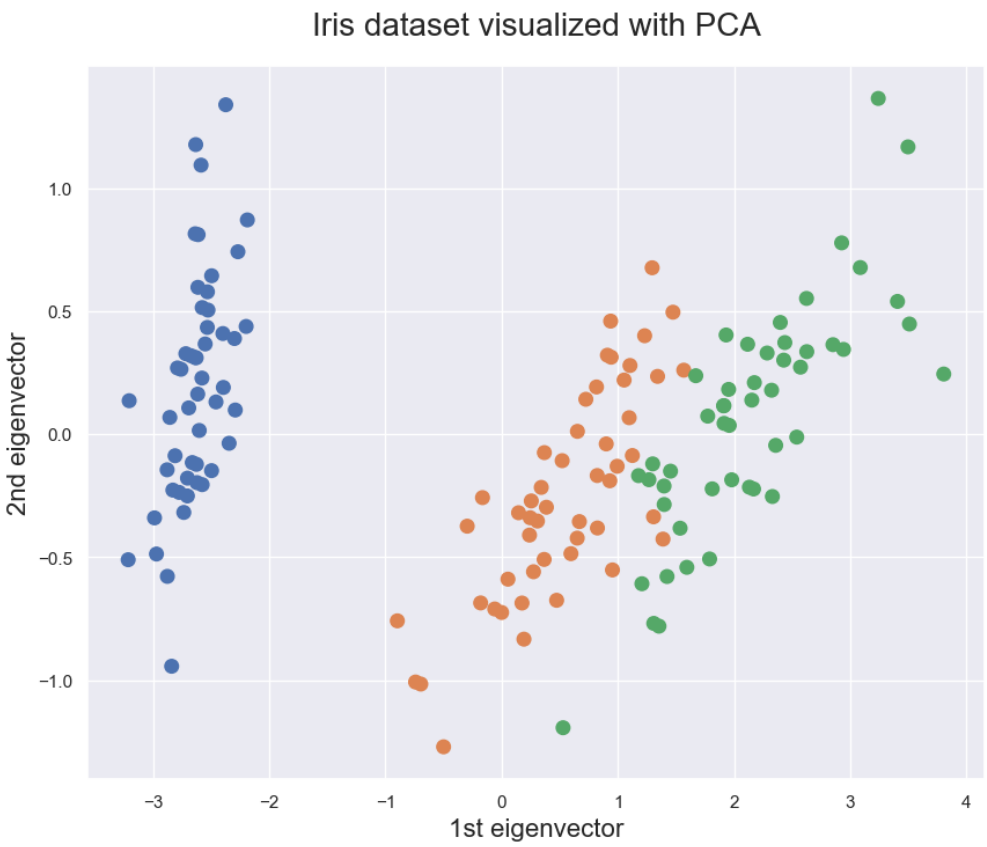
توضیح `n_components` در PCA

پارامتر `n_components` در تحلیل مؤلفه‌های اصلی (PCA) نقش مهمی در تعیین تعداد مؤلفه‌های اصلی که برای تبدیل و کاهش بعد داده‌ها استفاده می‌شود، دارد. این پارامتر مشخص می‌کند که داده‌های اصلی به چند بعد فشرده شوند. در زیر توضیح کامل این پارامتر آورده شده است:

اهمیت `n_components`

- تعیین تعداد مؤلفه‌های اصلی: پارامتر `n_components` تعداد مؤلفه‌های اصلی را که باید استخراج شوند، مشخص می‌کند. این مؤلفه‌ها ترکیبی خطی از ویژگی‌های اصلی هستند که بیشترین واریانس را در داده‌ها توضیح می‌دهند.
- کنترل کاهش بعد: با تنظیم این پارامتر، می‌توان تعداد ابعاد جدیدی که داده‌ها به آن کاهش می‌یابند را کنترل کرد. به عنوان مثال، اگر `n_components = 2` تنظیم شود، داده‌ها به دو بعد فشرده می‌شوند.
- حفظ اطلاعات مهم: انتخاب مناسب تعداد مؤلفه‌ها می‌تواند به حفظ بیشترین اطلاعات ممکن از داده‌های اصلی کمک کند. معمولاً از آزمون و خطا و تحلیل واریانس برای انتخاب بهترین مقدار `n_components` استفاده می‌شود.
- بهبود کارایی الگوریتم‌ها: کاهش بعد با استفاده از PCA و تنظیم مناسب `n_components` می‌تواند به بهبود کارایی الگوریتم‌های یادگیری ماشین و کاهش زمان پردازش کمک کند.

این تصویر نشان‌دهنده تجسم دیتاست Iris پس از اعمال تحلیل مؤلفه‌های اصلی (PCA) است. در این تصویر، داده‌های Iris به دو مؤلفه اصلی (eigenvectors) تبدیل شده‌اند که به عنوان محورهای x و y نمایش داده شده‌اند.

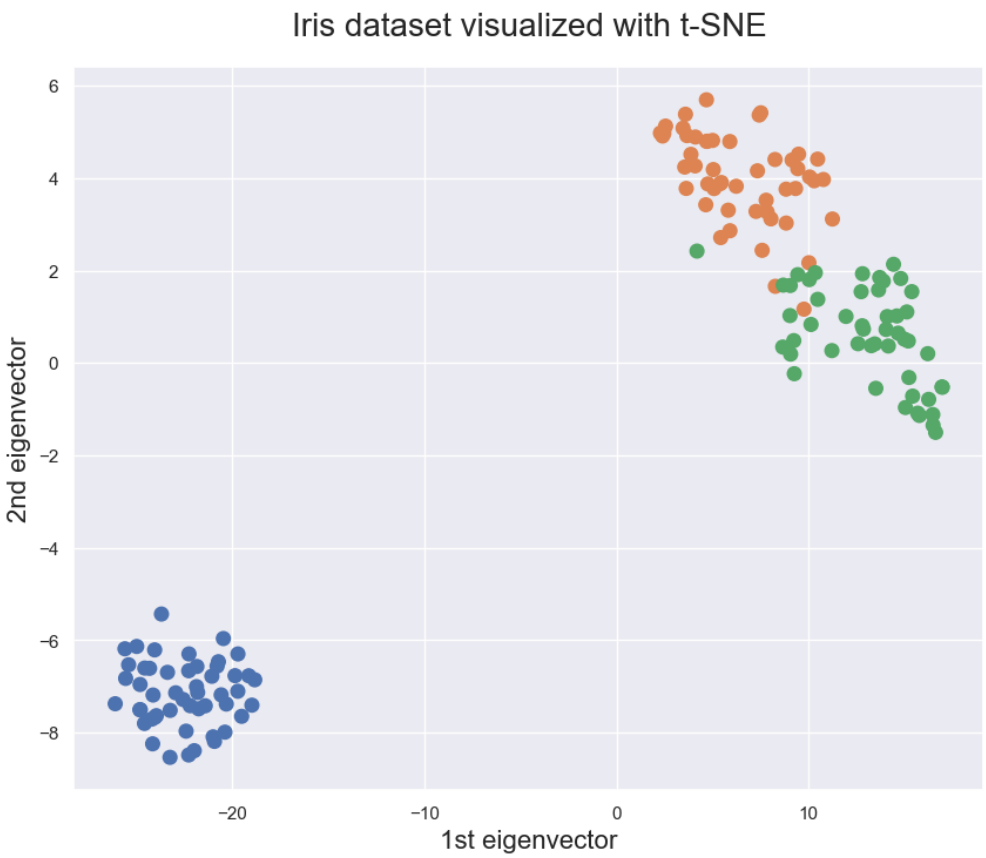


- محورهای اصلی (Eigenvectors):
  - محور x نشان‌دهنده اولین مؤلفه اصلی (۱st eigenvector) است.
  - محور y نشان‌دهنده دومین مؤلفه اصلی (۲nd eigenvector) است.
- تفکیک گونه‌ها:
  - نقاط آبی: نشان‌دهنده گونه .Setosa
  - نقاط نارنجی: نشان‌دهنده گونه .Versicolor
  - نقاط سبز: نشان‌دهنده گونه .Virginica
- مشاهده تفکیک:
  - گونه Setosa (آبی) به وضوح از دو گونه دیگر جدا شده است.
  - گونه‌های Versicolor (نارنجی) و Virginica (سبز) تا حدودی همپوشانی دارند، اما همچنان به خوبی تفکیک شده‌اند.
- پراکندگی داده‌ها:

- داده‌ها در فضای دو بعدی پراکنده شده‌اند و الگوهای تفکیک‌پذیری بین گونه‌ها مشاهده می‌شود.

این تصویر نشان می‌دهد که PCA به خوبی توانسته است داده‌های چندبعدی Iris را به یک فضای دو بعدی فشرده کند که در آن تفکیک گونه‌ها به وضوح قابل مشاهده است.

```
tsne = TSNE(n_components=2, n_iter=1000, random_state=RANDOM_STATE)
points = tsne.fit_transform(iris[features])
```



توضیحات کد

۱. تعریف t-SNE با ۲ مؤلفه: در این خط، یک شیء t-SNE تعریف می‌شود که تعداد مؤلفه‌های اصلی آن ۲ است. همچنین تعداد تکرارهای الگوریتم برای همگرایی (`n_iter`) برابر با ۱۰۰۰ تنظیم شده است. `random_state` نیز برای ایجاد نتایج قابل بازتولید استفاده می‌شود.

۲. اجرای t-SNE و تبدیل داده‌ها: در این خط، t-SNE بر روی داده‌های ویژگی‌های دیتاست Iris اعمال می‌شود و داده‌ها به فضای جدید دو بعدی تبدیل می‌شوند. `fit_transform` ابتدا مدل t-SNE را بر روی داده‌ها آموزش می‌دهد (`fit`) و سپس داده‌ها را به فضای جدید تبدیل می‌کند (`transform`).

تحلیل نمودار t-SNE

این تصویر نشان‌دهنده تجسم دیتاست Iris پس از اعمال الگوریتم t-SNE است. در این تصویر، داده‌های Iris به دو مؤلفه اصلی (eigenvectors) تبدیل شده‌اند که به عنوان محورهای x و y نمایش داده شده‌اند.

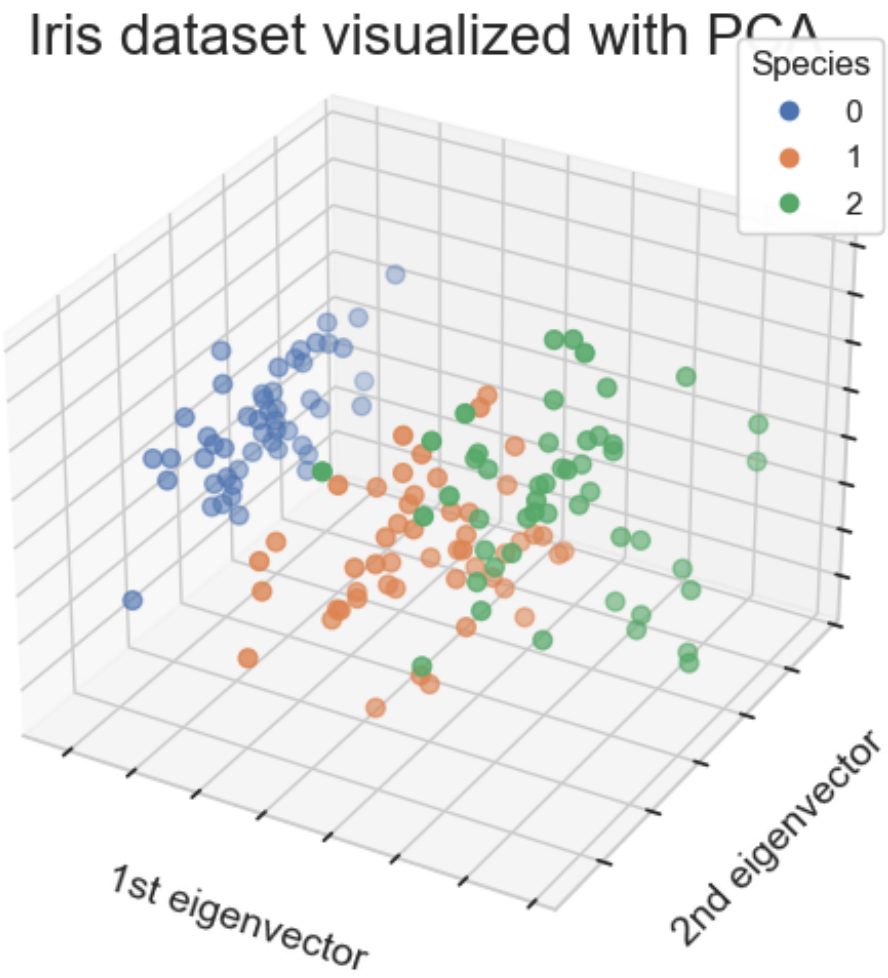
- محورهای اصلی (Eigenvectors):

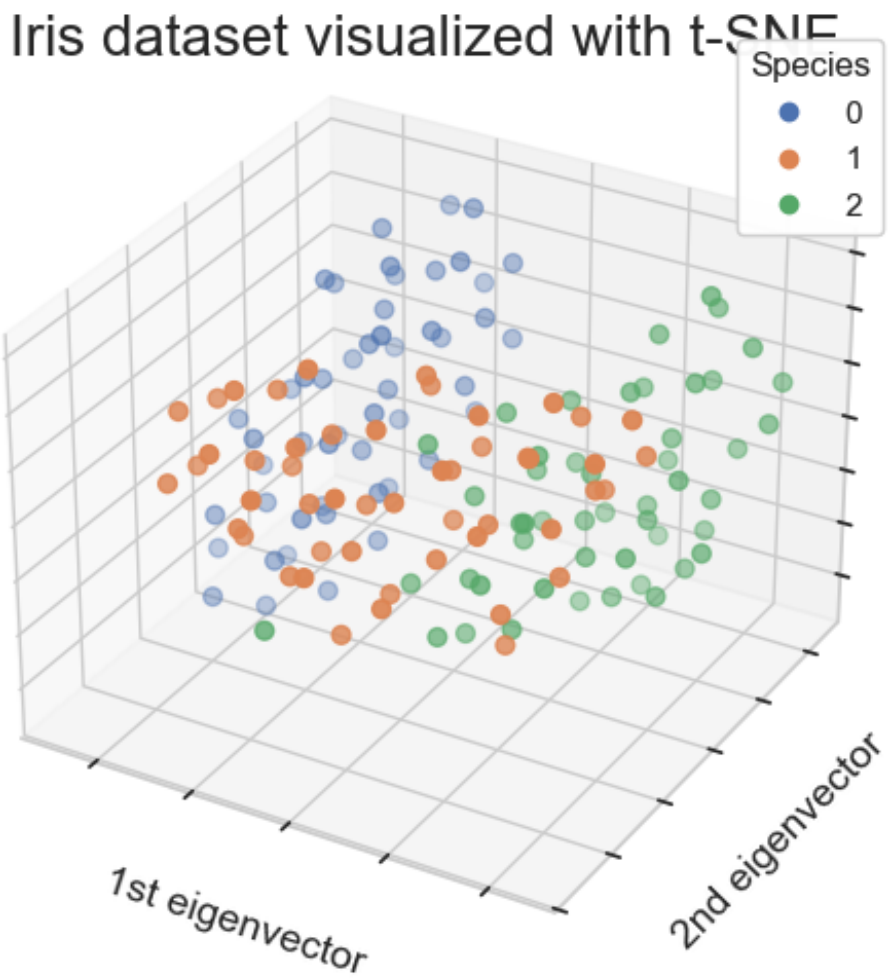
- محور x نشان‌دهنده اولین مؤلفه اصلی (1st eigenvector) است.
- محور y نشان‌دهنده دومین مؤلفه اصلی (2nd eigenvector) است.

- تفکیک گونه‌ها:
  - نقاط آبی: نشان‌دهنده گونه .Setosa
  - نقاط نارنجی: نشان‌دهنده گونه .Versicolor
  - نقاط سبز: نشان‌دهنده گونه .Virginica
- مشاهده تفکیک:
  - گونه Setosa (آبی) به وضوح از دو گونه دیگر جدا شده است.
  - گونه‌های Versicolor (نارنجی) و Virginica (سبز) نیز به خوبی تفکیک شده‌اند، اما برخی نقاط همپوشانی دارند.
- پراکندگی داده‌ها:
  - داده‌ها در فضای دو بعدی پراکنده شده‌اند و الگوهای تفکیک‌پذیری بین گونه‌ها مشاهده می‌شود.

این تصویر نشان می‌دهد که t-SNE به خوبی توانسته است داده‌های چندبعدی Iris را به یک فضای دو بعدی فشرده کند که در آن تفکیک گونه‌ها به وضوح قابل مشاهده است.

در ادامه کاهش ابعاد به ۳ بعد را با روش های PCA و t-SNE داریم:





چون ۳ بعدی هستند تحلیل های پیچیده تری دارند ولی برای جلوگیری از پیچیدگی تحلیل و محاسباتی از همان PCA دو بعدی استفاده میشود.

۲.۱ بخش دوم سوال اول

ابتدا دیتاست را بصورت ۸۰/۲۰ برای تمرین و تست تقسیم میکنیم. سپس به کم standard-scaler دیتا ها را scale میکنیم. نکته قابل توجه scale کردن روی دیتاست train و تعمیم آن به test است. حال به کمک SVM با هسته خطی مدل را ایجاد میکنیم و فیت میکنیم سپس با مدل فیت شده دیتاست test را پیشبینی میکنیم. نتایج به شرح زیر میباشد:

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

svm_linear = SVC(kernel='linear', random_state=44)
svm_linear.fit(X_train, y_train)
y_pred = svm_linear.predict(X_test)
```

در این مرحله، یک مدل SVM با هسته خطی و حالت تصادفی ثابت تعریف می‌شود. سپس مدل با استفاده از داده‌های آموزشی آموزش داده می‌شود. در نهایت، مدل برای پیش‌بینی برچسب‌های داده‌های آزمایشی استفاده می‌شود.

وزن ها و بایاس مدل تمرین داده شده برای بردار بصورت زیر میباشد:

```
weights :  [-0.17011404  0.48114532 -0.62212763 -0.72763817] bias :  -0.7924655222232578
```

نتایج حاصل از گزارش طبقه‌بندی مدل SVM به شرح زیر است:

- دقت (precision):
  - کلاس ۰: 1.00
  - کلاس ۱: 0.90
  - کلاس ۲: 1.00
- بازخوانی (recall):
  - کلاس ۰: 1.00
  - کلاس ۱: 1.00
  - کلاس ۲: 0.92
- امتیاز F۱ (f۱-score):
  - کلاس ۰: 1.00
  - کلاس ۱: 0.95
  - کلاس ۲: 0.96
- تعداد نمونه‌ها (support):
  - کلاس ۰: 9
  - کلاس ۱: 9
  - کلاس ۲: 12

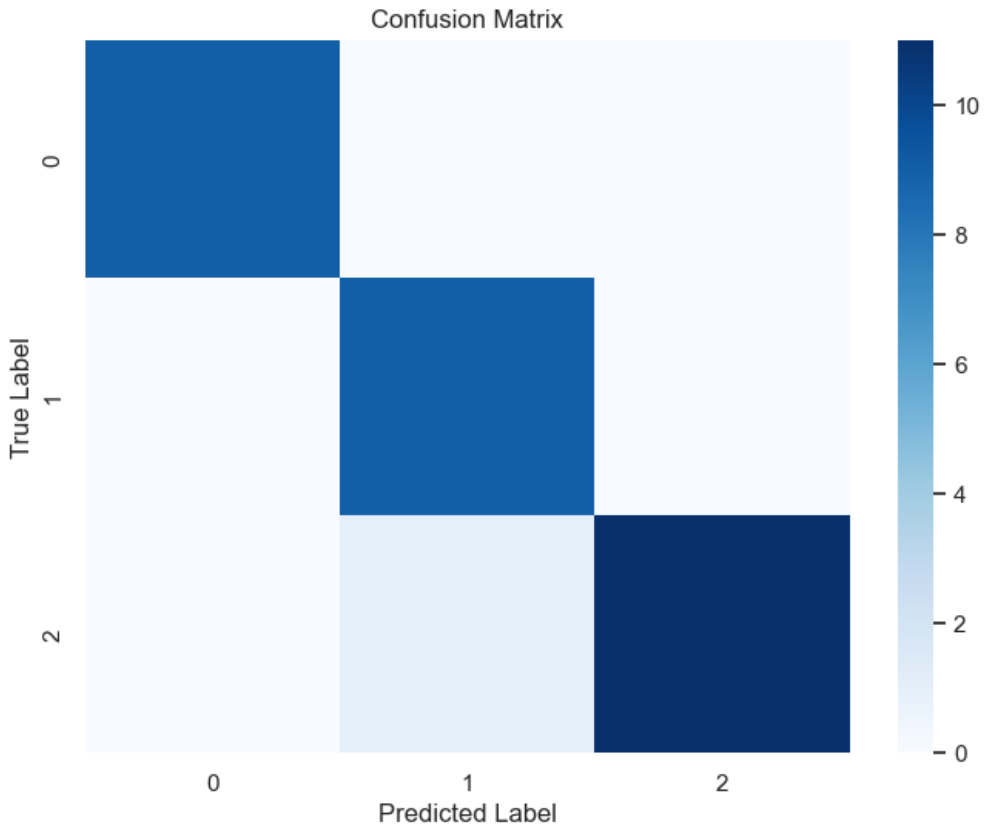
شاخص‌های کلی:

- دقت کل (accuracy): 0.97
- میانگین ماکرو (macro avg):
  - دقت: 0.97
  - : 0.97
  - امتیاز F۱: 0.97

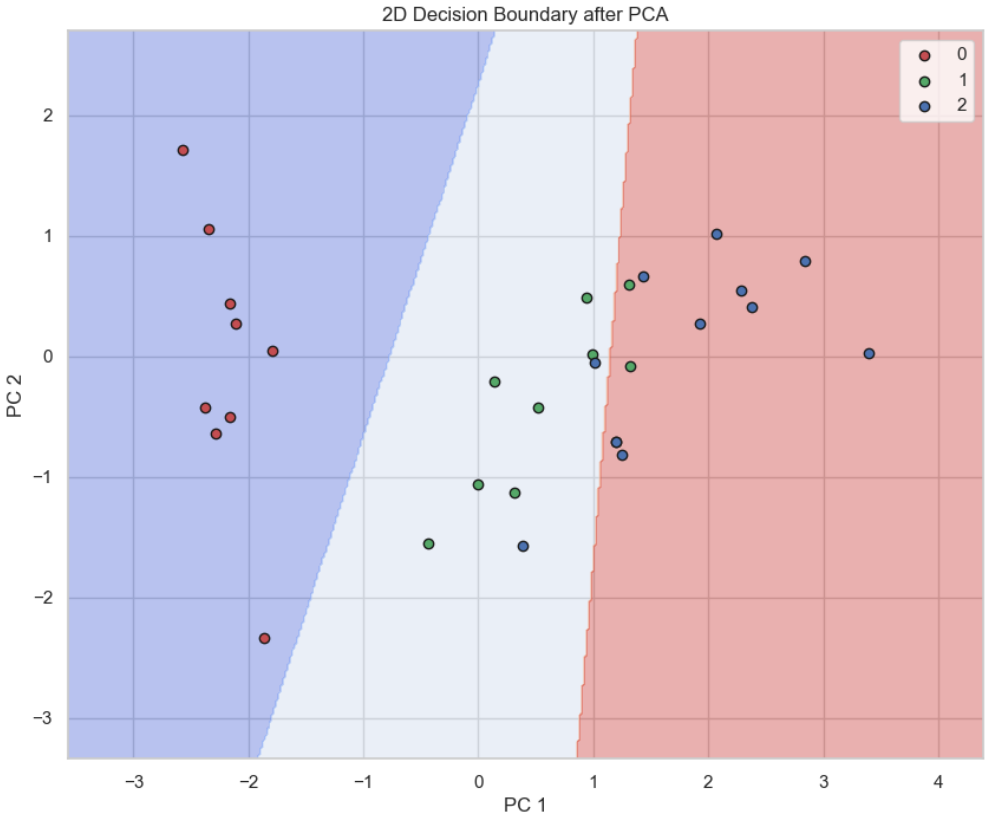
- تعداد نمونه‌ها: 30

- میانگین وزنی (weighted avg):
  - دقت: 0.97
  - بازخوانی: 0.97
  - امتیاز F۱: 0.97
  - تعداد نمونه‌ها: 30

	precision	recall	f1-score	support
0	1.00	1.00	1.00	9
1	0.90	1.00	0.95	9
2	1.00	0.92	0.96	12
accuracy			0.97	30
macro avg	0.97	0.97	0.97	30
weighted avg	0.97	0.97	0.97	30



نتایج نشان می‌دهد که مدل در کل عملکرد بسیار خوبی داشته است. دقت، بازخوانی و امتیاز F۱ برای کلاس‌های مختلف در سطح بالایی قرار دارد، به خصوص کلاس‌های ۰ و ۲ که دقت و بازخوانی آنها 1.00 است. کلاس ۱ نیز با وجود دقت 0.90، بازخوانی 1.00 و امتیاز F۱ برابر 0.95 عملکرد خوبی داشته است. میانگین‌های ماکرو و وزنی نیز نشان‌دهنده تعادل خوب مدل در پیش‌بینی کلاس‌های مختلف است.



در این نمودار، مرز تصمیم‌گیری مدل SVM پس از کاهش ابعاد با استفاده از PCA (تحلیل مولفه‌های اصلی) نشان داده شده است. محور افقی نشان‌دهنده مولفه اصلی اول (PC ۱) و محور عمودی نشان‌دهنده مولفه اصلی دوم (PC ۲) است. هر نقطه در نمودار نمایانگر نمونه‌ای از داده‌ها است که با توجه به کلاس مربوطه‌اش رنگ‌بندی شده است.

- کلاس ۰: با نقاط قرمز نشان داده شده است.
- کلاس ۱: با نقاط سبز نشان داده شده است.
- کلاس ۲: با نقاط آبی نشان داده شده است.

نواحی رنگی در پس‌زمینه نشان‌دهنده مرزهای تصمیم‌گیری مدل SVM برای تفکیک کلاس‌ها هستند. ناحیه آبی مربوط به کلاس ۰، ناحیه سفید مربوط به کلاس ۱ و ناحیه قرمز مربوط به کلاس ۲ است.

تحلیل نمودار:

- نقاطی که در ناحیه آبی قرار دارند، توسط مدل به درستی به عنوان کلاس ۰ شناسایی شده‌اند.
- نقاطی که در ناحیه سفید قرار دارند، به عنوان کلاس ۱ توسط مدل شناسایی شده‌اند.
- نقاطی که در ناحیه قرمز قرار دارند، به عنوان کلاس ۲ توسط مدل شناسایی شده‌اند.

مرزهای تصمیم‌گیری بین کلاس‌ها به خوبی مشخص هستند و مدل توانسته است با دقت بالایی نمونه‌ها را به کلاس‌های مربوطه تخصیص دهد. این نمودار نشان می‌دهد که مدل SVM با استفاده از PCA توانسته است داده‌ها را به خوبی تفکیک کند و مرزهای تصمیم‌گیری دقیقی برای هر کلاس ترسیم نماید.

۳.۱ بخش سوم سوال اول

کد زیر مدل‌های مختلف SVM با هسته چندجمله‌ای (polynomial) و درجات مختلف (از ۱ تا ۱۰) را آموزش داده و دقت هر یک را محاسبه می‌کند. ابتدا یک لیست از مدل‌های SVM با هسته چندجمله‌ای و درجات مختلف ساخته می‌شود. سپس لیست‌هایی برای ذخیره دقت‌ها و ماتریس‌های درهم‌ریختگی ایجاد می‌شود. سپس

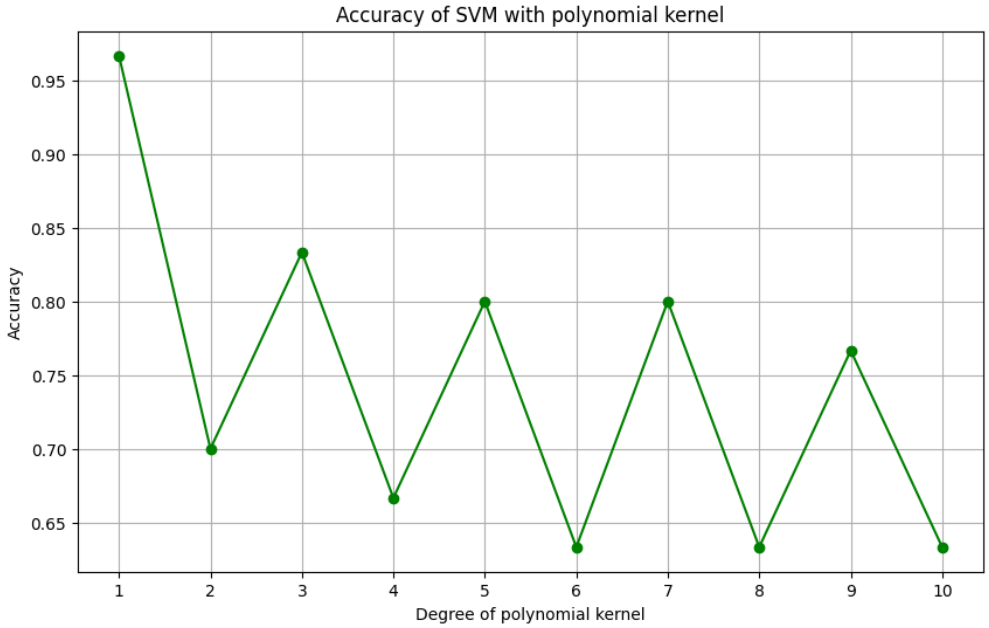
- مدل با داده‌های آموزشی آموزش داده می‌شود.
- پیش‌بینی‌ها بر روی داده‌های آزمایشی انجام می‌شود.
- دقت مدل محاسبه و در لیست دقت‌ها ذخیره می‌شود.
- ماتریس درهم‌ریختگی محاسبه و در لیست مربوطه ذخیره می‌شود.

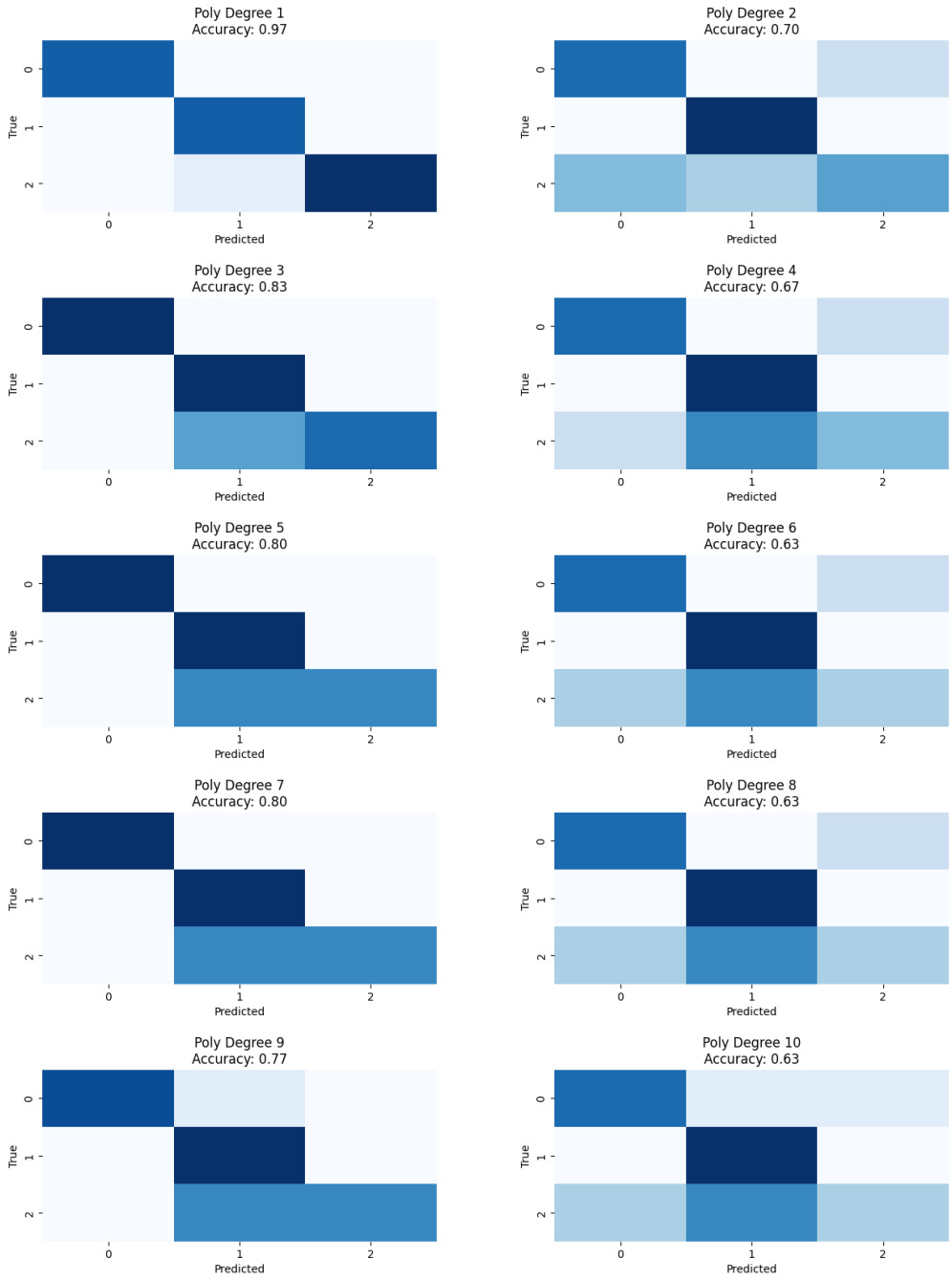
در نهایت، دقت هر مدل با درجه مربوطه چاپ می‌شود.

```python
classifiers = [SVC(kernel='poly', degree=d, C=0.7) for d in range(1, 11)]

accuracies = []
degrees = range(1, 11)
conf_matrices = []

for clf in classifiers:
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    accuracies.append(accuracy)
    conf_matrices.append(confusion_matrix(y_test, y_pred))

for degree, accuracy in zip(degrees, accuracies):
    print(f'degree {degree}, Accuracy = {accuracy:.2f}')
```

```
degree 1, Accuracy = 0.97
degree 2, Accuracy = 0.70
degree 3, Accuracy = 0.83
degree 4, Accuracy = 0.67
degree 5, Accuracy = 0.80
degree 6, Accuracy = 0.63
degree 7, Accuracy = 0.80
degree 8, Accuracy = 0.63
degree 9, Accuracy = 0.77
degree 10, Accuracy = 0.63
```

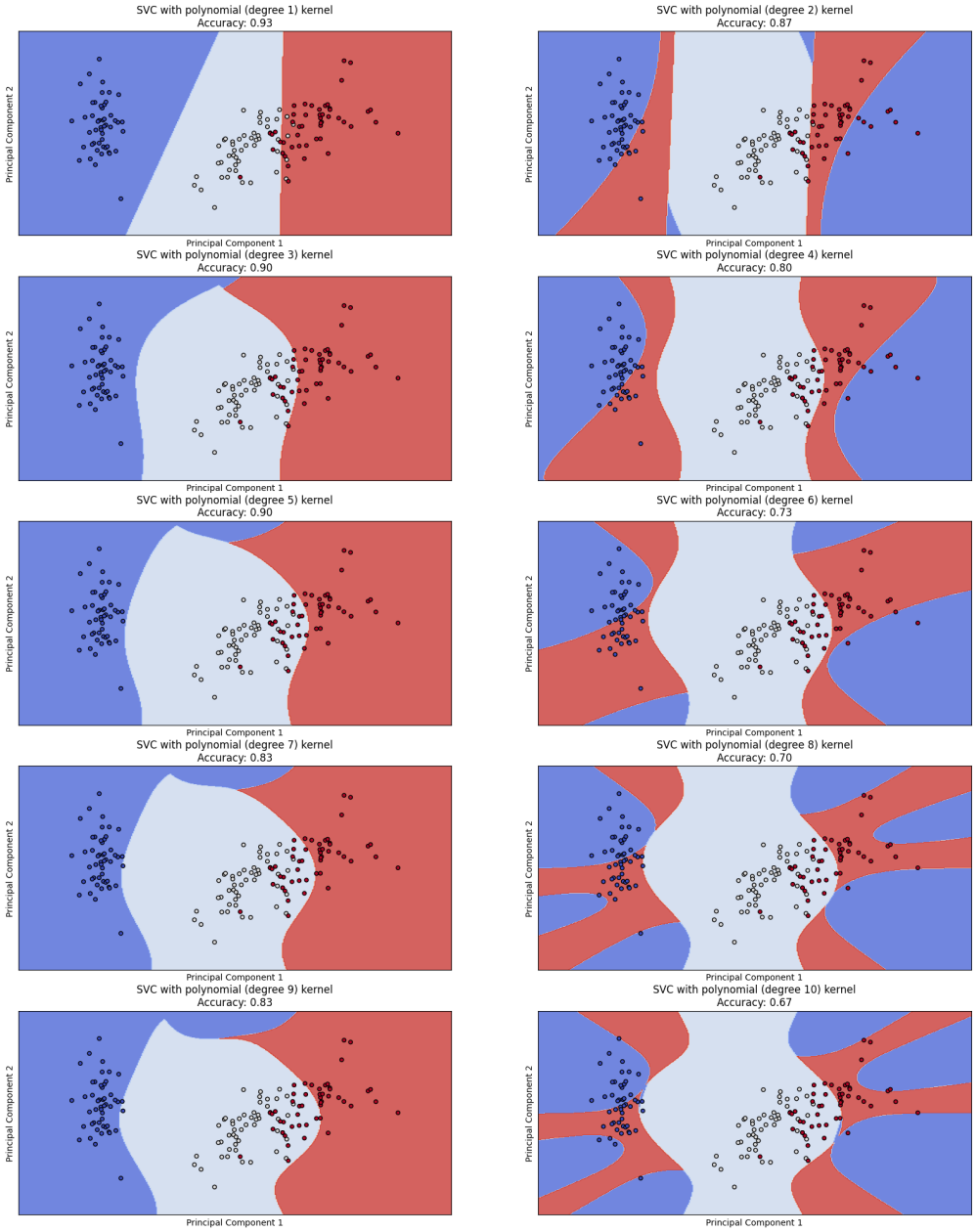
نمودار زیر دقت مدل‌های SVM با هسته چندجمله‌ای و درجات مختلف (از ۱ تا ۱۰) را نشان می‌دهد:





تحلیل نمودار:

- دقت مدل با درجه ۱ بالاترین مقدار، یعنی حدود 0.95، را دارد.
- دقت مدل با درجه ۲ به شدت کاهش یافته و به حدود 0.65 رسیده است.
- دقت مدل با درجه ۳ دوباره افزایش یافته و به حدود 0.80 رسیده است.
- دقت مدل‌ها با درجات ۴، ۶، ۷، و ۹ نوساناتی بین 0.70 تا 0.80 دارند.



- دقت مدل‌ها با درجات ۵، ۸، و ۱۰ به شدت کاهش یافته و به حدود 0.65 رسیده است.

این نوسانات نشان می‌دهد که انتخاب درجه مناسب برای هسته چندجمله‌ای تأثیر زیادی بر عملکرد مدل SVM دارد. درجات پایین‌تر (به خصوص درجه ۱) و درجات میانی (مانند درجه ۳ و ۷) عملکرد بهتری داشته‌اند، در حالی که درجات بسیار بالا یا بسیار پایین (مانند درجه ۲ و ۱۰) عملکرد ضعیف‌تری داشته‌اند.

- برای داده‌های مورد بررسی، درجه ۱ بهترین عملکرد را دارد.
- انتخاب درجه مناسب برای هسته چندجمله‌ای نیازمند آزمون و خطا و بررسی دقیق دقت مدل‌ها است.

دسترسی به گیف ایجاد شده اینجا کلیک کنید.

۴.۱ بخش چهارم سوال اول

کتابخانه‌ها و توابع کمکی

ابتدا کتابخانه‌های مورد نیاز و چندین تابع کمکی برای محاسبه هسته‌ها و اجرای SVM تعریف می‌شود.

- کتابخانه‌ها: شامل `cvxopt`، `sklearn`، `pandas`، `numpy`، `matplotlib` و `seaborn` است.
- توابع هسته‌ای: شامل `linear_kernel`، `polynomial_kernel`، `gaussian_kernel` و `sigmoid_kernel`.

تابع SVM۱

این تابع یک SVM دوکلاسه را با استفاده از برنامه‌ریزی درجه دوم حل می‌کند:

- پارامترها: شامل داده‌های ورودی `X`، برچسب‌های `y`، پارامتر `C` و نوع هسته (`kernel_type`).
- محاسبه ماتریس گرام: با توجه به نوع هسته، ماتریس گرام محاسبه می‌شود.
- حل مسئله :QP با استفاده از `cvxopt`، مسئله QP حل و ضرایب لاگرانژ به دست می‌آیند.
- پیش‌بینی: برای داده‌های آزمایشی با استفاده از وکتور پشتیبان و ضرایب لاگرانژ، پیش‌بینی انجام می‌شود.

تابع multiclass_svm

این تابع یک SVM چندکلاسه را با رویکرد یک‌درمقابل‌بقیه پیاده‌سازی می‌کند:

- پارامترها: شامل داده‌های ورودی `X`، داده‌های آزمایشی `X_t`، برچسب‌های `y`، پارامتر `C` و نوع هسته (`kernel_type`).
- طبقه‌بندی‌کننده‌ها: برای هر کلاس، یک SVM دوکلاسه آموزش داده می‌شود.
- تابع تصمیم‌گیری: برای پیش‌بینی کلاس داده‌های آزمایشی بر اساس نمرات تصمیم‌گیری هر SVM استفاده می‌شود.

تابع visualize_multiclass_classification۱ این تابع نتایج SVM چندکلاسه را به صورت تصویری نمایش می‌دهد:

- پارامترها: شامل داده‌های آموزشی `X_train`، برچسب‌ها `y_train1`، نوع هسته (`kernel_type`)، و پارامترهای طبقه‌بندی‌کننده.
- نمایش نقاط داده: داده‌های آموزشی به همراه مرزهای تصمیم‌گیری برای هر کلاس نمایش داده می‌شود.

بخش اصلی کد

این بخش شامل مراحل زیر است:

۱. بارگذاری و آماده‌سازی داده‌ها: مجموعه داده ایرس بارگذاری شده و با استفاده از PCA به دو بعد کاهش می‌یابد.

۲. تقسیم داده‌ها: داده‌ها به مجموعه‌های آموزشی و آزمایشی تقسیم می‌شوند.

۳. آموزش و ارزیابی :SVM یک SVM چندکلاسه با استفاده از هسته چندجمله‌ای برای درجات مختلف آموزش داده شده و دقت آن ارزیابی می‌شود.

۴. نمایش نتایج: نتایج به صورت گرافیکی نمایش داده می‌شوند.

نتایج به شرح زیر است:

```
Training with polynomial degree 1
Degree: 1, Accuracy: 0.8333333333333334
Training with polynomial degree 2
Degree: 2, Accuracy: 0.9666666666666667
Training with polynomial degree 3
Degree: 3, Accuracy: 0.9666666666666667
Training with polynomial degree 4
Degree: 4, Accuracy: 0.9666666666666667
Training with polynomial degree 5
Degree: 5, Accuracy: 0.9666666666666667
Training with polynomial degree 6
Degree: 6, Accuracy: 0.9666666666666667
Training with polynomial degree 7
Degree: 7, Accuracy: 0.9666666666666667
Training with polynomial degree 8
Degree: 8, Accuracy: 0.9666666666666667
Training with polynomial degree 9
Degree: 9, Accuracy: 0.9666666666666667
Training with polynomial degree 10
Degree: 10, Accuracy: 0.9666666666666667
```

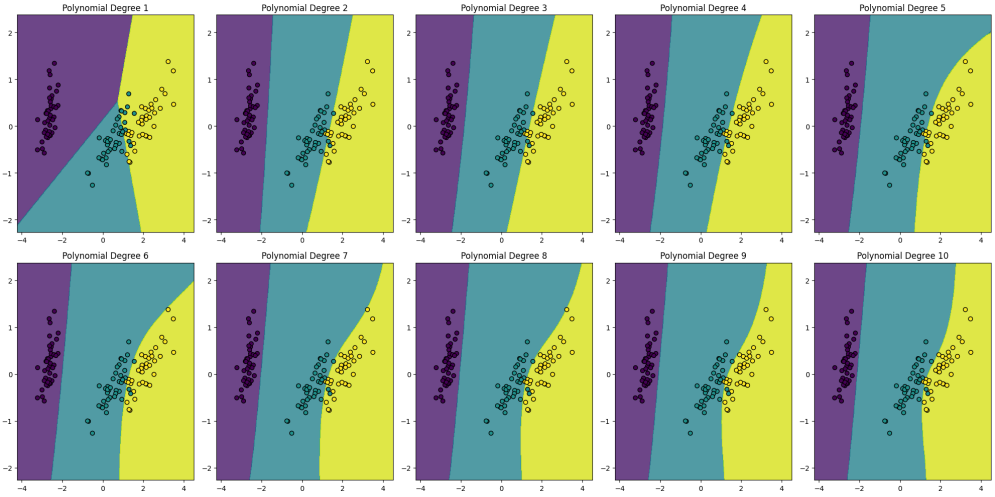
نتایج آموزش یک مدل SVM چندکلاسه با استفاده از هسته چندجمله‌ای برای درجات مختلف بر روی داده‌های ایرس به شرح زیر است. نتایج به صورت گرافیکی نمایش داده شده و دقت هر مدل محاسبه شده است.

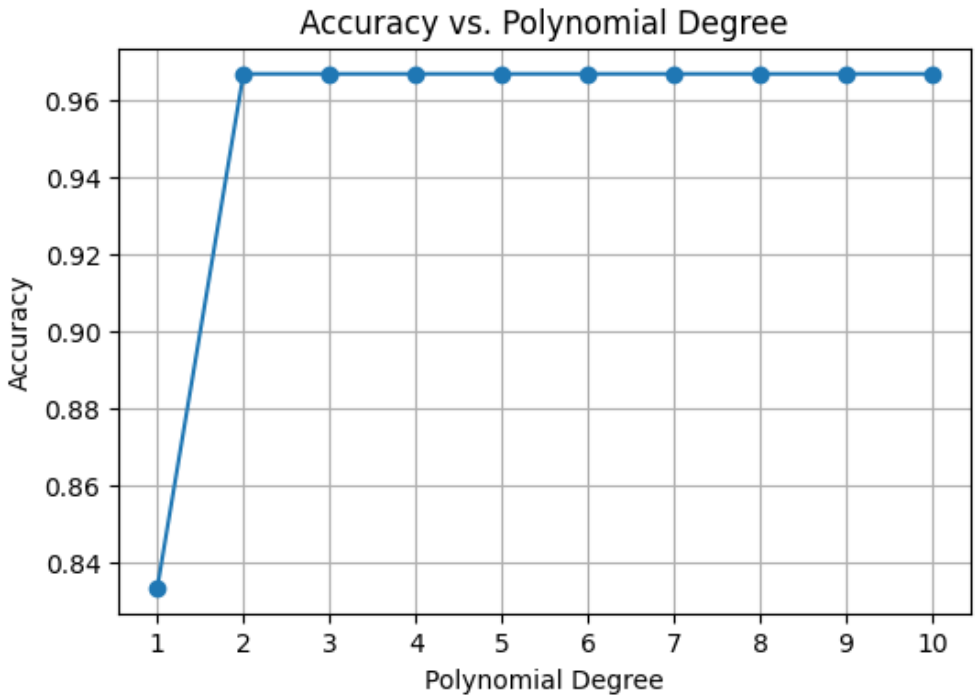
درجه ۱ Degree (Polynomial ۱)

- دقت: 83.33%
- مرزهای تصمیم‌گیری به صورت خطی بوده و نمی‌توانند به خوبی داده‌ها را جدا کنند.

درجات ۲ تا ۱۰ Degrees (Polynomial ۲ to ۱۰)

- دقت: 96.67%





- مرزهای تصمیم‌گیری با افزایش درجه بهبود یافته و به خوبی داده‌ها را جدا می‌کنند. از درجه ۲ به بعد، دقت مدل ثابت مانده و بهبود بیشتری نداشته است.

این نشان می‌دهد که استفاده از هسته چندجمله‌ای با درجه بالاتر از ۲ برای این مجموعه داده خاص، بهبود قابل توجهی در دقت مدل ندارد و درجه ۲ بهترین انتخاب برای این مدل است.

نتایج بصورت گیف در اینجا آپلود شده است.

۲ سوال سوم

۱.۲ بخش اول سوال سوم

داده‌های نامتوازن و تشخیص تقلب کارت اعتباری

تقلب کارت اعتباری یک تهدید رو به رشد با عواقب گسترده در صنعت مالی، شرکت‌ها و دولت‌ها است. تقلب را می‌توان به عنوان فریب مجرمانه با هدف کسب سود مالی تعریف کرد. با توجه به اینکه کارت اعتباری به محبوب‌ترین روش پرداخت برای تراکنش‌های آنلاین و آفلاین تبدیل شده است، نرخ تقلب نیز افزایش یافته است. دلایل اصلی تقلب به دلیل عدم امنیت کافی است که شامل استفاده از کارت‌های اعتباری دزدیده شده برای دریافت وجه نقد از بانک‌ها از طریق دسترسی قانونی می‌باشد. این مسئله منجر به دشواری بالای جلوگیری از تقلب کارت اعتباری می‌شود.

اهمیت تشخیص تقلب

تشخیص تقلب کارت اعتباری بسیار مهم است. تحقیقات زیادی برای تشخیص این نوع تقلب انجام شده است، که اکثریت تقلب‌های کارت اعتباری را تشکیل می‌دهند. استفاده از روش‌های سنتی برای تشخیص تقلب به دلیل حجم بزرگ داده‌ها غیرممکن است. با این حال، مؤسسات مالی توجه خود را به روش‌های محاسباتی جدید برای حل مشکل تقلب کارت اعتباری معطوف کرده‌اند.

چالش‌های داده‌های نامتوازن

مشکل طبقه‌بندی یکی از موضوعات کلیدی تحقیق در زمینه یادگیری ماشین است. روش‌های طبقه‌بندی موجود تنها می‌توانند عملکرد مطلوبی در مجموعه داده‌های متوازن داشته باشند. با این حال، در کاربردهای عملی، تعداد زیادی مجموعه داده نامتوازن وجود دارد. برای مشکل تقلب، کلاس اقلیت که همان تراکنش‌های غیرعادی هستند، اهمیت بیشتری دارند. به عنوان مثال، هنگامی که کلاس اقلیت کمتر از ۱ درصد از کل مجموعه داده‌ها را تشکیل می‌دهد، دقت کلی به بیش از ٪۹۹ می‌رسد حتی اگر همه کلاس‌های اقلیت به اشتباه طبقه‌بندی شده باشند.

روش‌های نمونه‌برداری از کلاس اقلیت

نمونه‌برداری از کلاس اقلیت یک روش رایج برای مقابله با مشکل طبقه‌بندی داده‌های نامتوازن است. هدف اصلی از نمونه‌برداری بیش از حد افزایش تعداد نمونه‌های کلاس اقلیت است تا اطلاعات طبقه‌بندی اصلی بهتر حفظ شود. بنابراین، در زمینه‌هایی که دقت طبقه‌بندی بالاتری مورد نیاز است، الگوریتم‌های نمونه‌برداری بیش از حد به طور کلی انتخاب می‌شوند.

استفاده از شبکه عصبی autoencoder و پیشنمونه‌گیری

این مقاله به پیاده‌سازی تشخیص تقلب کارت اعتباری با استفاده از autoencoder حذف نویز (Denoising AutoEncoder) و پیشنمونه‌گیری می‌پردازد. برای داده‌های نامتوازن، ما تصمیم گرفتیم از روش‌های فوق برای دستیابی به مدل مناسب استفاده کنیم.

روش‌های استفاده شده برای حل چالش‌ها

- پیشنمونه‌گیری یا SMOTE: مجموعه داده را با تولید نمونه‌های کلاس اقلیت متوازن می‌کند.
- شبکه عصبی autoencoder نویزگیر (DAE): به طور همزمان داده‌ها را نویزگیری و طبقه‌بندی می‌کند و دقت تشخیص کلاس اقلیت را بهبود می‌بخشد.
- تحلیل مولفه‌های اصلی (PCA): ابعاد داده را کاهش می‌دهد و ویژگی‌های مربوطه را انتخاب می‌کند.

مدل پیشنهادی ترکیبی از پیشنمونه‌گیری و هایautoencoder نویزگیر است که بهبودهای قابل توجهی در تشخیص تراکنش‌های تقلبی نسبت به روش‌های سنتی نشان داده است. نتایج ارزیابی نشان می‌دهد که مدل‌های مبتنی بر یادگیری عمیق مانند DAE در ترکیب با روش‌های پیشنمونه‌گیری و تحلیل مولفه‌های اصلی قادر به شناسایی تقلبات با دقت بالا هستند. این امر به کمک شبکه عصبی autoencoder و تحلیل‌های خاص در توسعه مدل‌های تشخیص تقلب متمرکز و دقیق کمک می‌کند، به ویژه در زمینه مجموعه داده‌های نامتوازن و نویزی.

۲.۲ بخش دوم سوال سوم

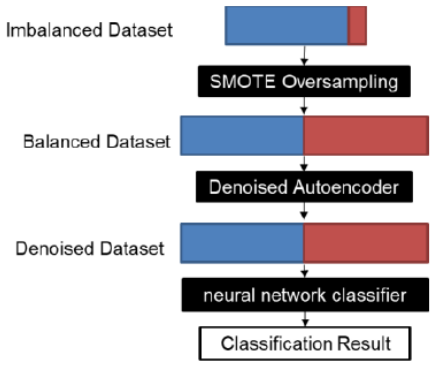


Fig. 5 Flowchart of the porcess

شکل ۱: رفرنس: https://arxiv.org/pdf/۱۹۰۸.۱۱۵۵۳

معماری شبکه برای تشخیص تقلب کارت اعتباری

این معماری شامل دو قسمت اصلی است: autoencoder حذف نویز (Denoising AutoEncoder) و طبقه‌بند شبکه عصبی.

فرآیند کلی تشخیص

فرآیند تشخیص تقلب شامل مراحل زیر است:

۱. مجموعه داده نامتوازن: داده‌های اولیه شامل تعداد زیادی تراکنش عادی و تعداد کمی تراکنش تقلبی است.

۲. پیشنمونه‌گیری با استفاده از :SMOTE برای متوازن کردن مجموعه داده، از تکنیک SMOTE استفاده می‌شود تا تعداد نمونه‌های کلاس اقلیت افزایش یابد. به عنوان مثال، قبل از پیشنمونه‌گیری، مجموعه داده آموزشی شامل ۲۲۶۵۲ رکورد تراکنش بود که ۲۲۵۳۸ نمونه در کلاس عادی و ۱۱۴ نمونه در کلاس غیرعادی قرار داشت. پس از پیشنمونه‌گیری، هر دو کلاس دارای ۲۲۵۳۸ نمونه شدند.

۳. autoencoder حذف نویز: داده‌های متوازن شده به autoencoder حذف نویز وارد می‌شود تا نویزهای موجود در داده‌ها حذف شوند و داده‌ها به صورت پاکسازی شده به دست آیند.

۴. طبقه‌بند شبکه عصبی: داده‌های بدون نویز به طبقه‌بند شبکه عصبی وارد می‌شوند تا نتیجه طبقه‌بندی نهایی حاصل شود. طبقه‌بند شبکه عصبی با استفاده از لایه‌های کاملاً متصل و تابع خطای آنتروپی متقاطع SoftMax بهینه‌سازی می‌شود.

استفاده از autoencoder حذف نویز

autoencoder حذف نویز برای بهبود دقت طبقه‌بندی طراحی شده است. این الگوریتم با یادگیری حذف نویز و بازسازی ورودی بدون اختلال به مدل کمک می‌کند تا در برابر داده‌های خراب مقاوم‌تر باشد و دقت تشخیص را بهبود بخشد.

استفاده از پیشنمونه‌گیری

نمونه‌برداری از کلاس اقلیت یک روش رایج برای مقابله با مشکل طبقه‌بندی داده‌های نامتوازن است. هدف اصلی از نمونه‌برداری بیش از حد افزایش تعداد نمونه‌های کلاس اقلیت است تا اطلاعات طبقه‌بندی اصلی بهتر حفظ شود. در این مورد، از تکنیک SMOTE برای افزایش نمونه‌های کلاس اقلیت استفاده شده است. تعداد لایه ها برای دسته بندی و autoencoder در شکل زیر آورده شده است.

Table 3. Model design for classifier

Denoised Dataset (29)
Fully-Connected-Layer (22)
Fully-Connected-Layer (15)
Fully-Connected-Layer (10)
Fully-Connected-Layer (5)
Fully-Connected-Layer (2)
SoftMax Cross Entropy Loss Function

شکل ۲: رفرنس: https://arxiv.org/pdf/۱۹۰۸.۱۱۵۵۳

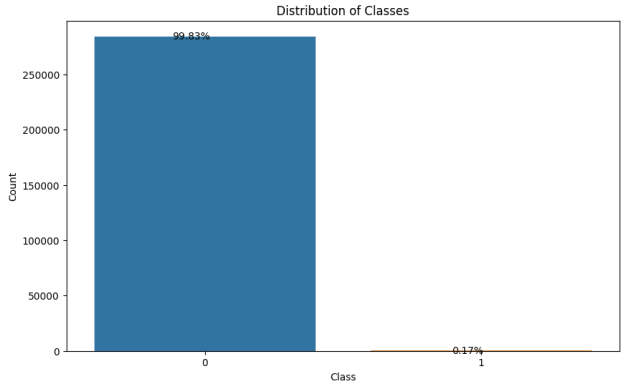
Table 2. Model design for denoised autoencoder

Dataset with noise (29)
Fully-Connected-Layer (22)
Fully-Connected-Layer (15)
Fully-Connected-Layer (10)
Fully-Connected-Layer (15)
Fully-Connected-Layer (22)
Fully-Connected-Layer (29)
Square Loss Function

شکل ۳: رفرنس: https://arxiv.org/pdf/۱۹۰۸.۱۱۵۵۳

۳.۲ بخش سوم سوال سوم

ابتدا دیتا ها را میخوانیم سپس نحوه پخش آنها را انجام میدهیم نتایج به شرح زیر است:



همانطور که مشهود است مدل به شدن دچار عدم بالانسی در تعداد کلاس های مختلف است. دیتا ست ها را بصورت زیر به ۳ دسته ،train test و eval تقسیم میکنیم.

```
(205060, 2) (51266, 2) (28481, 2)
```

توزیع در دیتاست های تمرین و ارزیابی به شرح زیر دیتسربیوشن دارند:

```
Training set classes:  [204706.    354.]
Testing set classes:  [51177.    89.]
Validation set classes:  [28432.    49.]
```

- `SMOTE(sampling_strategy='minority', random_state=44)`: ایجاد یک شیء SMOTE با استراتژی نمونه‌برداری 'minority' و حالت تصادفی ۴۴.
- `np.argmax(y_train, smote.fit_resample(X_train, axis=1))`: اعمال روش SMOTE بر روی داده‌های ورودی `X_train` و برچسب‌های `y_train` پس از تبدیل برچسب‌ها به مقادیر عددی با استفاده از `np.argmax`.
- `to_categorical(y_train_res, num_classes=2)`: تبدیل برچسب‌های خروجی به قالب دسته‌بندی شده (one-hot) با دو کلاس.
- `np.sum(y_train_res, axis=0)`: محاسبه مجموع عناصر هر کلاس در مجموعه داده‌های جدید برای بررسی تعادل کلاس‌ها.

خروجی این کد نشان می‌دهد که بعد از اعمال SMOTE، تعداد نمونه‌ها برای هر دو کلاس به مقدار 204706 رسیده است، که نشان‌دهنده تعادل کامل بین کلاس‌ها است.

```
smote = SMOTE(sampling_strategy='minority', random_state=44)
X_train_res, y_train_res = smote.fit_resample(X_train, np.argmax(y_train, axis=1))
y_train_res = to_categorical(y_train_res, num_classes=2)
np.sum(y_train_res, axis=0)
```

✓ 0.5s

```
array([204706., 204706.])
```

```python
def add_noise(data, noise_factor=0.2):
    noisy_data = data + noise_factor * np.random.normal(loc=0.0, scale=1.0, size=data.shape)
    noisy_data = np.clip(noisy_data, 0., 1.)
    return noisy_data

X_train_noisy = add_noise(X_train_res)
X_test_noisy = add_noise(X_test)
input_dim = X_train_res.shape[1]
encoding_dim = 10

input_layer = Input(shape=(input_dim,))
encoder = Dense(encoding_dim, activation="relu")(input_layer)
encoder = Dense(22, activation="relu")(encoder)
encoder = Dense(15, activation="relu")(encoder)
encoder = Dense(encoding_dim, activation="relu")(encoder)
encoder = Dense(15, activation="relu")(encoder)
encoder = Dense(22, activation="relu")(encoder)
decoder = Dense(input_dim, activation='sigmoid')(encoder)

autoencoder = Model(inputs=input_layer, outputs=decoder)
autoencoder.compile(optimizer='adam', loss='mean_squared_error')

checkpoint = ModelCheckpoint('best_autoencoder.keras', monitor='val_loss', save_best_only=True, mode='min', verbose=1)

autoencoder.fit(X_train_noisy, X_train_res,
                epochs=20,
                batch_size=256,
                shuffle=True,
                validation_data=(X_test_noisy, X_test),
                callbacks=[checkpoint],
                verbose=1)
```

- ابتدا، نویز گاوسی به داده‌های آموزشی و تست اضافه می‌شود تا داده‌ها به صورت نویزی درآیند.
- سپس، یک مدل autoencoder با چندین لایه چگال (Dense) ساخته می‌شود که هدف آن بازسازی داده‌های ورودی است.
- مدل با استفاده از تابع خطای `mean_squared_error` و بهینه‌ساز `adam` کامپایل می‌شود.
- یک چک‌پوینت برای ذخیره بهترین مدل بر اساس کمترین مقدار خطای اعتبارسنجی (`val_loss`) تعریف می‌شود.
- در نهایت، مدل autoencoder با داده‌های نویزی و داده‌های اصلی به عنوان هدف آموزش داده می‌شود.

```
autoencoder.load_weights('best_autoencoder.keras')
X_train_denoised = autoencoder.predict(X_train_noisy)
X_test_denoised = autoencoder.predict(X_test_noisy)
classifier_input = Input(shape=(input_dim,))
classifier_layer = Dense(encoding_dim, activation="relu")(classifier_input)
classifier_layer = Dense(22, activation="relu")(classifier_layer)
classifier_layer = Dense(15, activation="relu")(classifier_layer)
classifier_layer = Dense(10, activation="relu")(classifier_layer)
classifier_layer = Dense(5, activation="relu")(classifier_layer)
classifier_layer = Dense(2, activation='softmax')(classifier_layer)
classifier = Model(inputs=classifier_input, outputs=classifier_layer)
classifier.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
classifier_checkpoint = ModelCheckpoint('best_classifier.keras', monitor='val_loss', save_best_only=True, mode='min', verbose=1)
classifier.fit(X_train_denoised, y_train_res,
               epochs=20,
               batch_size=256,
               shuffle=True,
               validation_data=(X_test_denoised, y_test),
               callbacks=[classifier_checkpoint],
               verbose=1)
```

۱. بارگذاری وزن‌های (Autoencoder):

- مدل autoencoder از قبل آموزش داده شده و وزن‌های آن ذخیره شده‌اند. این وزن‌ها بارگذاری می‌شوند تا مدل بتواند داده‌ها را دنویز کند.

۲. پیش‌بینی داده‌های دنویز شده:

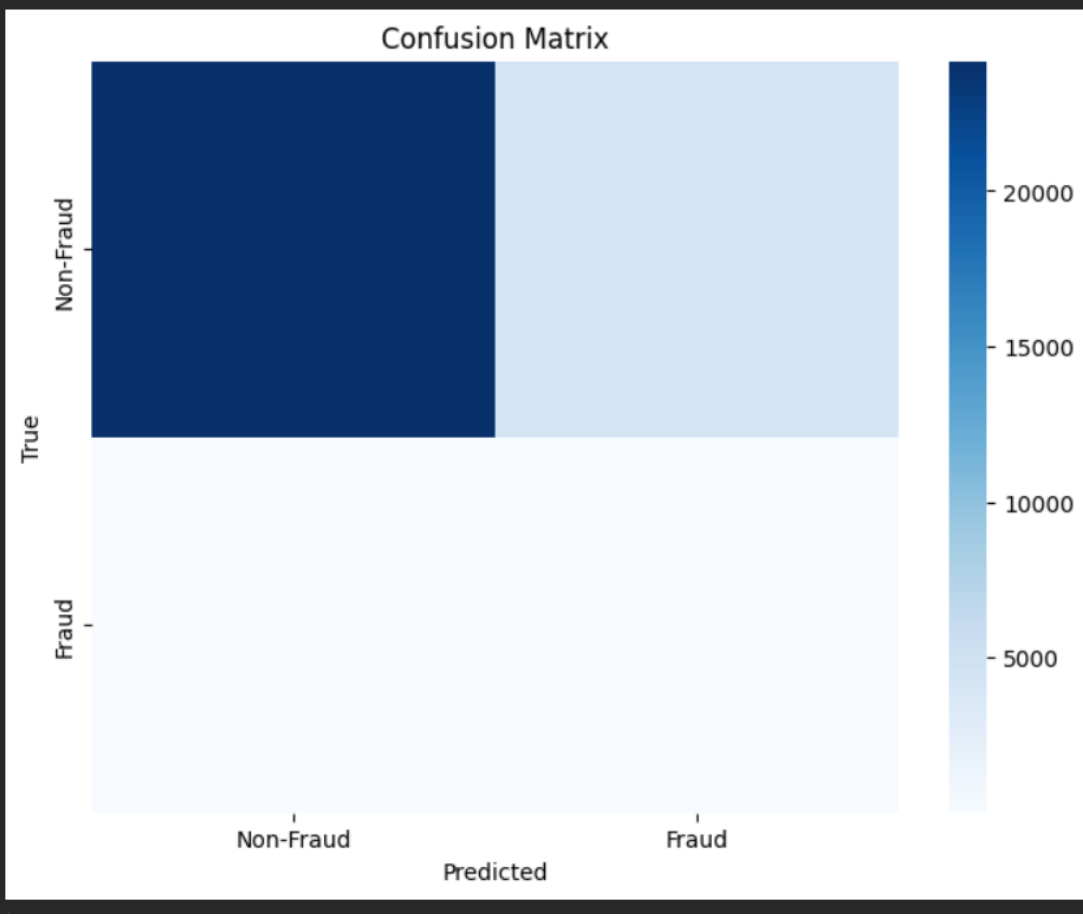
- داده‌های نویزی ورودی به مدل autoencoder داده می‌شوند تا داده‌های دنویز شده پیش‌بینی شوند. این کار برای داده‌های آموزشی و تست انجام می‌شود.

۳. ساخت و آموزش مدل طبقه‌بندی:

- یک مدل طبقه‌بندی با چندین لایه چگال (Dense) ساخته می‌شود. این مدل با داده‌های دنویز شده آموزش داده می‌شود.
- مدل با استفاده از تابع خطای `categorical_crossentropy` و بهینه‌ساز `adam` کامپایل می‌شود.
- یک چک‌پوینت برای ذخیره بهترین مدل طبقه‌بندی کننده بر اساس کمترین مقدار خطای اعتبارسنجی (`val_loss`) تعریف می‌شود.
- در نهایت، مدل طبقه‌بندی کننده با داده‌های دنویز شده و برچسب‌های آموزشی به مدت ۲۰ دوره آموزشی (epoch) آموزش داده می‌شود.

۴.۲ بخش چهارم سوال سوم

```
Recall: 0.6938775510204082
[[24166  4266]
 [   15    34]]
```



	precision	recall	f1-score	support
Non-Fraud	1.00	0.85	0.92	28432
Fraud	0.01	0.69	0.02	49
accuracy			0.85	28481
macro avg	0.50	0.77	0.47	28481
weighted avg	1.00	0.85	0.92	28481

- دقت (Precision):

- برای کلاس عدم تقلب: 1.00
- برای کلاس تقلب: 0.01
- دقت بالا برای کلاس عدم تقلب و دقت بسیار پایین برای کلاس تقلب نشان می‌دهد که مدل بیشتر نمونه‌های تقلب را به اشتباه به عنوان عدم تقلب پیش‌بینی می‌کند.

- بازخوانی (Recall):
  - برای کلاس عدم تقلب: 0.85
  - برای کلاس تقلب: 0.69
  - بازخوانی نشان می‌دهد که مدل توانایی بهتری در شناسایی نمونه‌های تقلب دارد (0.69) اما هنوز نیاز به بهبود دارد.
- امتیاز F۱ (F۱-Score):
  - برای کلاس عدم تقلب: 0.92
  - برای کلاس تقلب: 0.02
  - امتیاز F۱ ترکیبی از دقت و بازخوانی است. امتیاز بسیار پایین برای کلاس تقلب نشان می‌دهد که مدل به خوبی در شناسایی تقلب عمل نمی‌کند.
- دقت کلی (Accuracy):
  - مدل دارای دقت کلی 0.85 است که نشان‌دهنده عملکرد خوب در شناسایی عدم تقلب است اما دقت کلی نمی‌تواند نشان‌دهنده عملکرد خوب مدل در کلاس‌های نامتوازن باشد.
- میانگین کل Macro) (Avg:
  - میانگین کل دقت: 0.50
  - میانگین کل بازخوانی: 0.77
  - میانگین کل F۱-Score: 0.47
  - میانگین کل برای هر کلاس محاسبه می‌شود و نشان‌دهنده عملکرد متوازن مدل نیست.
- میانگین وزن‌دار Weighted) (Avg:
  - میانگین وزن‌دار دقت: 1.00
  - میانگین وزن‌دار بازخوانی: 0.85
  - میانگین وزن‌دار F۱-Score: 0.92
  - میانگین وزن‌دار بر اساس تعداد نمونه‌ها محاسبه می‌شود و به دلیل تعداد زیاد نمونه‌های عدم تقلب، به مدل امتیاز بالاتری می‌دهد.

این نتایج نشان می‌دهند که مدل در شناسایی نمونه‌های تقلب به خوبی عمل نمی‌کند و نیاز به بهبود دارد.

در مسائل با توزیع نامتوازن برچسب‌ها، معیار Accuracy (دقت کلی) نمی‌تواند به تنهایی عملکرد مدل را به درستی نشان دهد. این به این دلیل است که دقت کلی تعداد نمونه‌های صحیح پیش‌بینی شده را نسبت به کل نمونه‌ها محاسبه می‌کند، بدون توجه به توزیع نابرابر کلاس‌ها. در این شرایط، مدل ممکن است عملکرد بسیار خوبی برای کلاس غالب داشته باشد اما نتواند کلاس نادر را به خوبی تشخیص دهد.

توزیع نامتوازن برچسب‌ها

در مسائل با توزیع نامتوازن، تعداد نمونه‌های یک کلاس (مثلاً عدم تقلب) بسیار بیشتر از تعداد نمونه‌های کلاس دیگر (مثلاً تقلب) است. در چنین شرایطی، مدل می‌تواند با پیش‌بینی همیشگی کلاس غالب به دقت کلی بالایی دست یابد، حتی اگر نتواند نمونه‌های کلاس نادر را به درستی تشخیص دهد.

معیار Accuracy

دقت کلی (Accuracy) تعداد پیش‌بینی‌های صحیح را تقسیم بر کل پیش‌بینی‌ها می‌کند. در مسائل با توزیع نامتوازن، این معیار ممکن است گمراه‌کننده باشد زیرا نسبت بالای نمونه‌های کلاس غالب می‌تواند دقت کلی را به طور مصنوعی بالا ببرد.

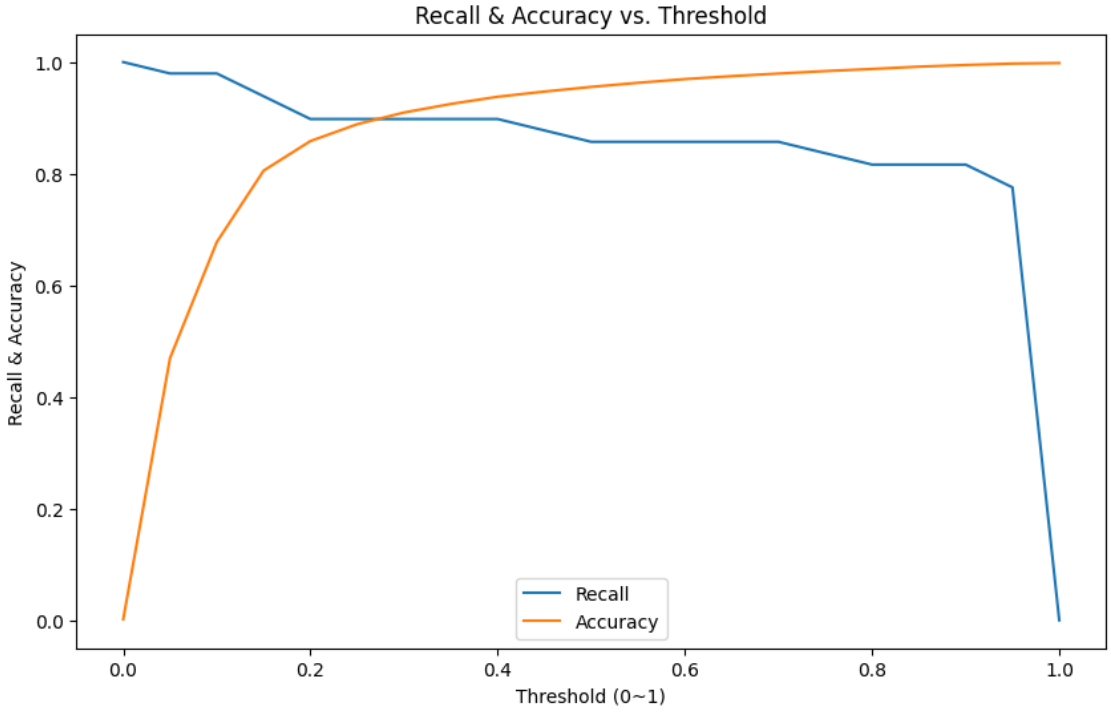
نیاز به معیارهای دیگر

برای ارزیابی بهتر مدل در مسائل با توزیع نامتوازن، استفاده از معیارهای دیگر مانند Precision (دقت)، Recall (بازخوانی)، و F۱-Score ضروری است.

- Precision: نسبت نمونه‌های مثبت صحیح پیش‌بینی شده به کل نمونه‌های مثبت پیش‌بینی شده. این معیار میزان پیش‌بینی‌های درست در بین پیش‌بینی‌های مثبت را نشان می‌دهد.
- Recall: نسبت نمونه‌های مثبت صحیح پیش‌بینی شده به کل نمونه‌های مثبت واقعی. این معیار توانایی مدل در شناسایی نمونه‌های مثبت را نشان می‌دهد.
- F۱-Score: میانگین هارمونیک دقت و بازخوانی. این معیار تعادلی بین دقت و بازخوانی ایجاد می‌کند و برای مسائل با توزیع نامتوازن مناسب است.
- ROC-AUC: مساحت زیر منحنی ،ROC که عملکرد مدل در تفکیک بین کلاس‌ها را نشان می‌دهد.

این معیارها به ارزیابی دقیق‌تر عملکرد مدل در شرایطی که توزیع برچسب‌ها نامتوازن است کمک می‌کنند و نشان می‌دهند که آیا مدل توانایی شناسایی کلاس نادر را دارد یا خیر.

```python
thresholds = np.arange(0.0, 1.05, 0.05)
recalls = []
accuracies = []
for threshold in thresholds:
    y_pred = (y_pred_prob[:, 1] >= threshold).astype(int)
    recall = recall_score(np.argmax(y_valid, axis=1), y_pred)
    accuracy = accuracy_score(np.argmax(y_valid, axis=1), y_pred)
    recalls.append(recall)
    accuracies.append(accuracy)
plt.figure(figsize=(10, 6))
plt.plot(thresholds, recalls, label='Recall')
plt.plot(thresholds, accuracies, label='Accuracy')
plt.xlabel('Threshold (0~1)')
plt.ylabel('Recall & Accuracy')
plt.title('Recall & Accuracy vs. Threshold')
plt.legend()
plt.show()
```



تحلیل معیارهای Recall و Accuracy بر اساس Threshold

نموداری که ارائه شده نشان‌دهنده تغییرات دو معیار بازخوانی (Recall) و دقت کلی (Accuracy) بر اساس تغییرات آستانه (Threshold) در مدل طبقه‌بندی است.

بازخوانی (Recall)

بازخوانی با کاهش آستانه افزایش می‌یابد. این به این دلیل است که با کاهش آستانه، مدل نمونه‌های بیشتری را به عنوان مثبت پیش‌بینی می‌کند، که منجر به شناسایی تعداد بیشتری از نمونه‌های مثبت واقعی می‌شود.

- در آستانه‌های پایین، بازخوانی بسیار بالاست اما با افزایش آستانه، بازخوانی کاهش می‌یابد. این به این دلیل است که مدل با آستانه‌های بالا تعداد کمتری از نمونه‌های مثبت را شناسایی می‌کند.

دقت کلی (Accuracy)

دقت کلی در آستانه‌های پایین بسیار کم است زیرا مدل تعداد زیادی از نمونه‌های منفی را به اشتباه به عنوان مثبت پیش‌بینی می‌کند.

- با افزایش آستانه، دقت کلی افزایش می‌یابد و در آستانه‌های بالاتر به مقدار بیشتری می‌رسد.
- در آستانه‌های بسیار بالا، دقت کلی مجدداً کاهش می‌یابد زیرا مدل تعداد کمی از نمونه‌های مثبت را شناسایی می‌کند.

در مسائل با توزیع نامتوازن برچسب‌ها، تعیین آستانه مناسب بسیار مهم است. انتخاب آستانه‌ای که تعادلی بین بازخوانی و دقت کلی ایجاد کند، می‌تواند به عملکرد بهتر مدل کمک کند.

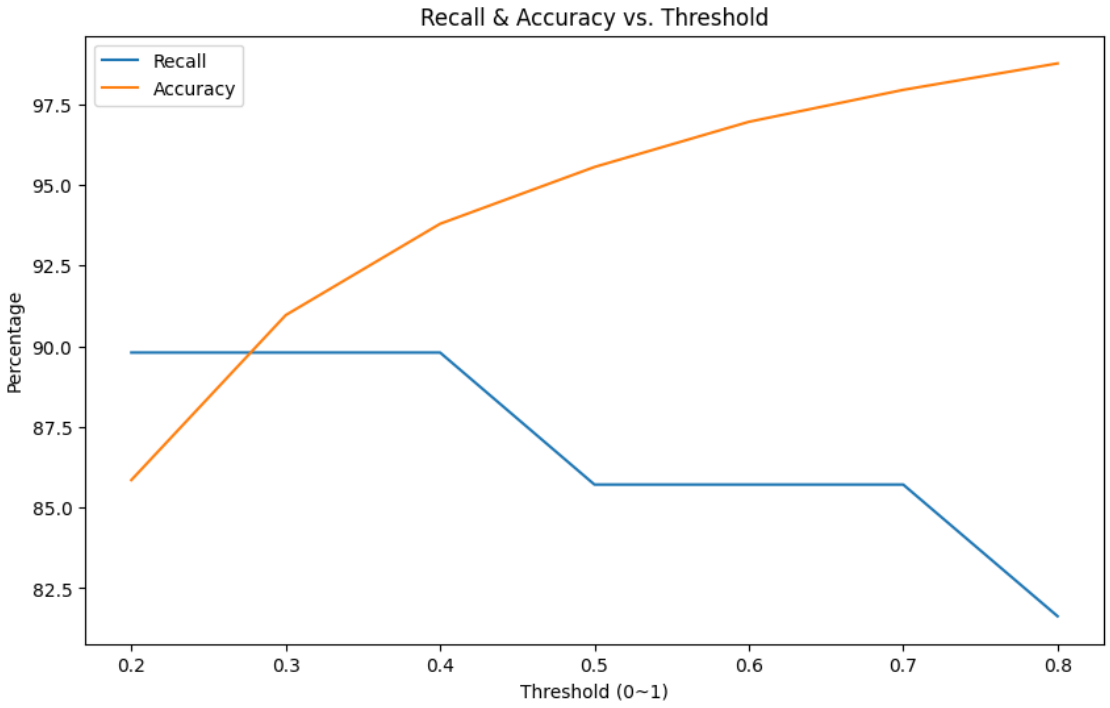
- اگر هدف شناسایی حداکثری نمونه‌های مثبت باشد (مثل شناسایی تقلب)، بازخوانی بالا مهم‌تر است.
- اما اگر هدف کاهش تعداد پیش‌بینی‌های نادرست باشد، دقت کلی اهمیت بیشتری پیدا می‌کند.

نتایج به شرح زیر تحلیل می‌شود:

نرخ بازخوانی (Recall) (Rate)

- نرخ بازخوانی در آستانه‌های 0.2 تا 0.4 ثابت و برابر با 89.80% است.
- از آستانه 0.5 به بعد، نرخ بازخوانی به تدریج کاهش می‌یابد.
- در آستانه 0.8، نرخ بازخوانی به 81.63% می‌رسد.
- کاهش نرخ بازخوانی با افزایش آستانه نشان می‌دهد که مدل با افزایش آستانه، تعداد کمتری از نمونه‌های مثبت را شناسایی می‌کند.

Threshold	Recall Rate	Accuracy
0.2	89.80%	85.85%
0.3	89.80%	90.96%
0.4	89.80%	93.79%
0.5	85.71%	95.55%
0.6	85.71%	96.95%
0.7	85.71%	97.94%
0.8	81.63%	98.76%



دقت کلی (Accuracy)

- دقت کلی با افزایش آستانه به طور مداوم افزایش می‌یابد.
- از آستانه 0.2 تا 0.8، دقت کلی از 85.85% به 98.76% می‌رسد.
- افزایش دقت کلی با افزایش آستانه نشان می‌دهد که مدل با افزایش آستانه، تعداد کمتری از نمونه‌های منفی را به اشتباه به عنوان مثبت پیش‌بینی می‌کند.

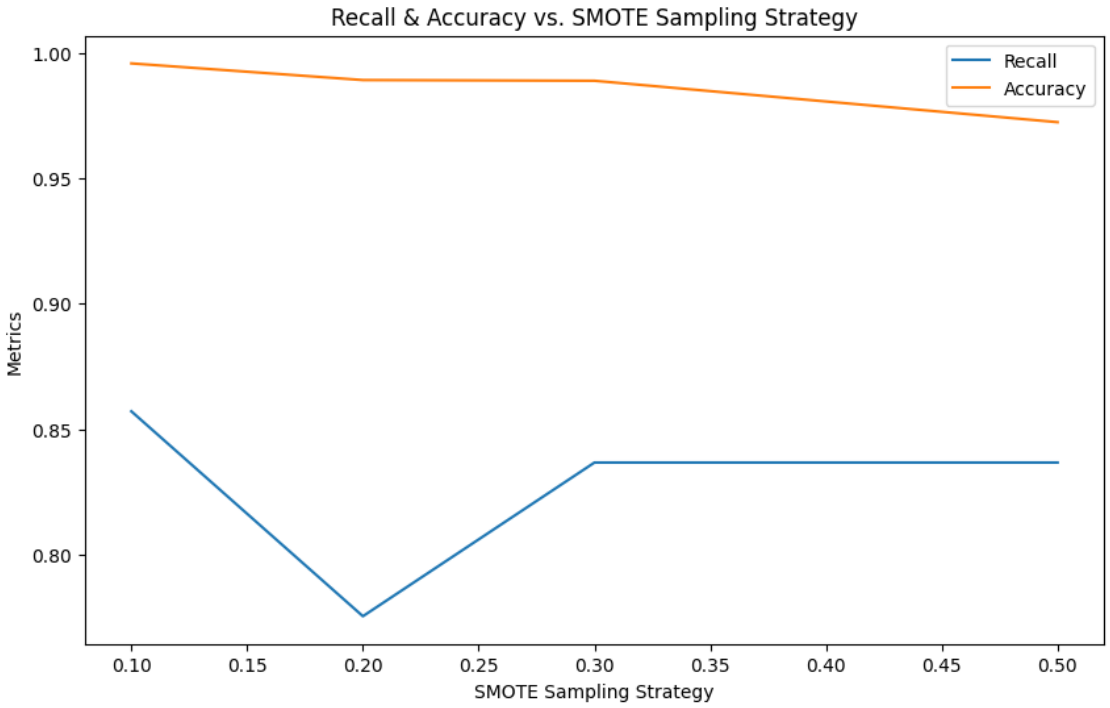
نتیجه‌گیری

- در آستانه‌های پایین، نرخ بازخوانی بالا و دقت کلی پایین است. این به این دلیل است که مدل در شناسایی نمونه‌های مثبت موفق است اما تعداد زیادی از نمونه‌های منفی را به اشتباه به عنوان مثبت پیش‌بینی می‌کند.
- در آستانه‌های بالا، دقت کلی بالا و نرخ بازخوانی پایین است. این به این دلیل است که مدل تعداد کمتری از نمونه‌های مثبت را شناسایی می‌کند و بیشتر نمونه‌های منفی را درست پیش‌بینی می‌کند.
- انتخاب آستانه مناسب بستگی به هدف مدل دارد. اگر هدف شناسایی حداکثری نمونه‌های مثبت باشد، باید آستانه پایین‌تری انتخاب شود. اگر هدف کاهش تعداد پیش‌بینی‌های نادرست باشد، باید آستانه بالاتری انتخاب شود.

۵.۲ بخش پنجم سوال سوم

در این بخش به پیاده سازی با تغییر آستانه oversampling میپردازیم که ۴ آستانه انتخاب شده است ونتایج به شرح زیر است.

```
0.1     0.857143  0.995752
0.2     0.775510  0.989116
0.3     0.836735  0.988835
0.5     0.836735  0.972332
```



نرخ بازخوانی (Recall) Rate

- نرخ بازخوانی در آستانه 0.1 بالاست و برابر با 0.857143 است.
- در آستانه 0.2، نرخ بازخوانی به 0.775510 کاهش می‌یابد.
- با افزایش آستانه به 0.3، نرخ بازخوانی مجدداً افزایش یافته و به 0.836735 می‌رسد.
- در آستانه 0.5، نرخ بازخوانی ثابت می‌ماند و تغییری نمی‌کند.
- این تغییرات نشان می‌دهد که با تغییر آستانه، مدل توانایی متفاوتی در شناسایی نمونه‌های مثبت دارد.

دقت کلی (Accuracy)

- دقت کلی در آستانه 0.1 بسیار بالاست و برابر با 0.995752 است.
- در آستانه 0.2، دقت کلی به 0.989116 کاهش می‌یابد.
- در آستانه‌های 0.3 و 0.5، دقت کلی به ترتیب به 0.988835 و 0.972332 کاهش می‌یابد.
- دقت کلی نشان می‌دهد که مدل با تغییر آستانه، توانایی متفاوتی در پیش‌بینی صحیح نمونه‌ها دارد، اما دقت کلی همواره بالاست.

نتیجه‌گیری

- آستانه 0.1 بهترین ترکیب از بازخوانی بالا و دقت کلی بالا را فراهم می‌کند.
- آستانه 0.2 به طور کلی منجر به کاهش نرخ بازخوانی و دقت کلی می‌شود.
- انتخاب آستانه مناسب بستگی به هدف مدل دارد؛ اگر شناسایی حداکثری نمونه‌های مثبت هدف باشد، آستانه‌های پایین‌تر بهتر هستند.
- در آستانه‌های بالاتر، دقت کلی کاهش می‌یابد اما همچنان قابل قبول است.

۶.۲ بخش ششم سوال سوم

در این قسمت به تحلیل نتایج بدون استفاده از oversampling و denoising-autoencoder پرداخته میشود.

```
              precision    recall  f1-score   support

   Non-Fraud       1.00      1.00      1.00     56863
       Fraud       0.89      0.72      0.79        99

    accuracy                           1.00     56962
   macro avg       0.94      0.86      0.90     56962
weighted avg       1.00      1.00      1.00     56962

Confusion Matrix:
 [[56854     9]
 [   28    71]]
Accuracy: 1.00
Recall: 0.72
```

Precision (دقت)

- برای کلاس عدم تقلب: 1.00
- برای کلاس تقلب: 0.89
- دقت بالا برای هر دو کلاس نشان می‌دهد که مدل تعداد کمی از نمونه‌های مثبت پیش‌بینی شده را به اشتباه به عنوان منفی پیش‌بینی کرده است.

Recall (بازخوانی)

- برای کلاس عدم تقلب: 1.00
- برای کلاس تقلب: 0.72
- بازخوانی بالا برای کلاس عدم تقلب و نسبتاً پایین‌تر برای کلاس تقلب نشان می‌دهد که مدل توانایی بهتری در شناسایی نمونه‌های عدم تقلب دارد.

F۱-Score

- برای کلاس عدم تقلب: 1.00
- برای کلاس تقلب: 0.79
- امتیاز F۱ ترکیبی از دقت و بازخوانی است. امتیاز بالای F۱ برای هر دو کلاس نشان‌دهنده عملکرد خوب مدل است، هرچند که برای کلاس تقلب کمی پایین‌تر است.

Accuracy (دقت کلی)

- دقت کلی: 1.00
- دقت کلی بالا نشان می‌دهد که مدل تعداد زیادی از نمونه‌ها را به درستی طبقه‌بندی کرده است.

Avg Macro (میانگین کل)

- میانگین دقت: 0.94
- میانگین بازخوانی: 0.86
- میانگین F۱-Score: 0.90
- میانگین کل برای هر کلاس محاسبه شده و نشان‌دهنده عملکرد کلی مدل است.

Avg Weighted (میانگین وزن‌دار)

- میانگین دقت: 1.00
- میانگین بازخوانی: 1.00
- میانگین F۱-Score: 1.00
- میانگین وزن‌دار بر اساس تعداد نمونه‌ها محاسبه شده و به دلیل تعداد زیاد نمونه‌های عدم تقلب، به مدل امتیاز بالاتری می‌دهد.

ماتریس درهم ریختگی (Confusion Matrix)

- مدل ۵۶۸۵۴ نمونه عدم تقلب را به درستی و ۹ نمونه عدم تقلب را به اشتباه به عنوان تقلب پیش‌بینی کرده است.
- مدل ۷۱ نمونه تقلب را به درستی و ۲۸ نمونه تقلب را به اشتباه به عنوان عدم تقلب پیش‌بینی کرده است.