UNIVERSITÄT
LEIPZIG

ICDS: Group 5

Dimensionality Reduction

Leipzig, 14.07.2023

Harpreet Int-Veen, Karoline Schuster, Moritz Huhle,
Paula Hagen, Paul Kühnel, Sepideh Kavianpoor Esfahani

Structure

- Recap
- Own PCA Implementation
- Evaluation own PCA
- Comparison with python libraries
- Classification: Decision trees & Cross Validation
- Comparison with T-SNE
- Encountered problems

Recap of the road map

- ✓ Load and clean data set
  a. Investigate data set for missings
  b. Descriptive statistics (distribution of cancer types, age…)
- ✓ Perform PCA using own implementation
- ✓ Evaluate own PCA and compare to results from a library implementation

- ★ Comparison of the most common libraries for PCA
- ★ Performing Classification after dimensionality reduction
- ★ Comparison PCA and T-SNE to evaluate which one fits our case better

Dataset

ICMR Dataset (Indian Council of Medical Research published on Kaggle [1])

- Contains gene samples from people associated with different cancer types
- Which genes or combination of genes are responsible for each cancer type
- Usability: classification and clustering

Descriptive Details

- Input dataset contains 801 samples for the corresponding 801 people with cancer
- Who have been detected with different types of cancer
- Each sample contains expression values of more than 20K genes
- Samples have one of the types of tumours: BRCA, KIRC, COAD, LUAD and PRAD

Dimensionality Reduction:

- Input: complete dataset including all genes (20.531)
- Output: the principal components → genes that are most responsible

Dataset - Distributions of classes

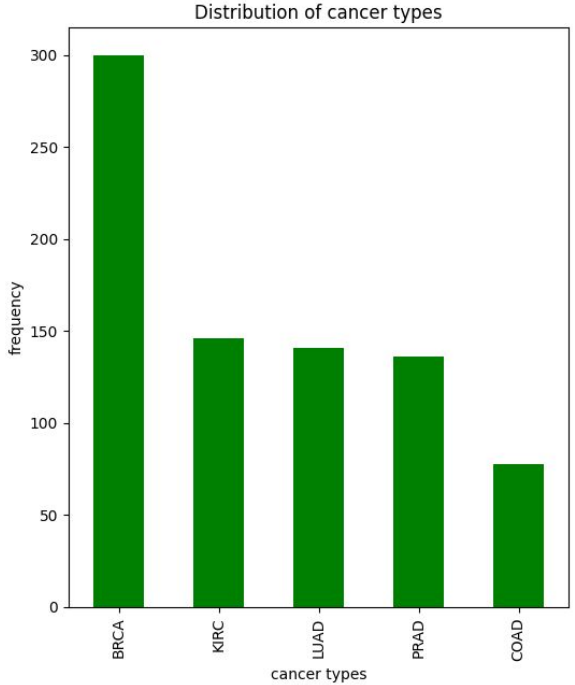


Image sources: Own figure

Eigenvalues and Eigenvectors in PCA

- Eigenvalues and eigenvectors are play a central role in PCA
- Eigenvalues represent the amount of variance explained by each eigenvector or principal component
- Eigenvectors are the directions along which the data exhibits maximum variance

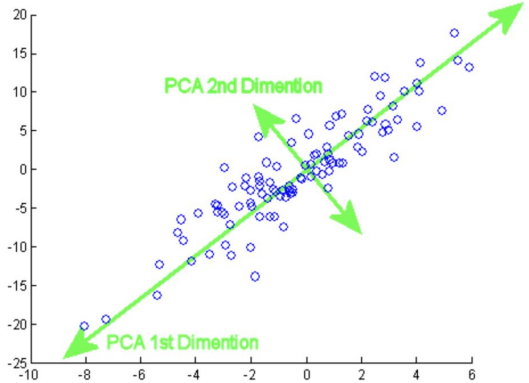


Image source: https://programmathically.com/principal-components-analysis-explained-for-dummies/

Own implementation of the PCA

```
own_eigenvalues, own_eigenvectors = our_pca(train, n_components)
```

Fit the data:

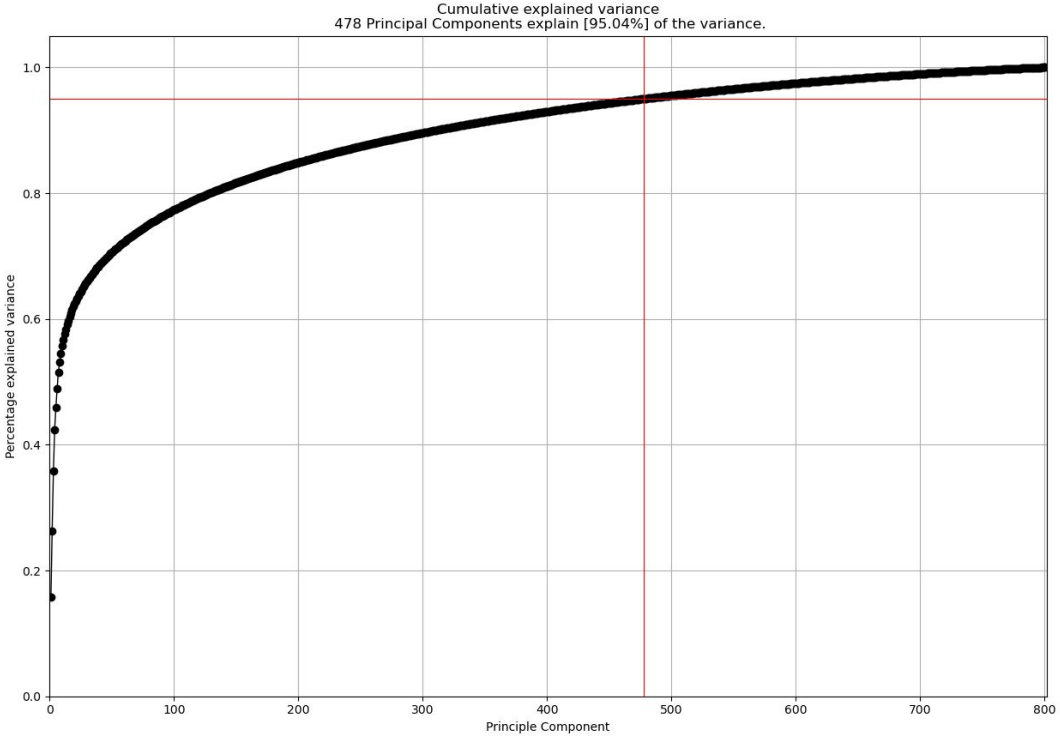
- Input: Cleaned dataset and desired number of principal components
- Output: Calculated eigenvalues and eigenvectors

```
transformed_data = apply_components(centered_data, eigenvectors)
```

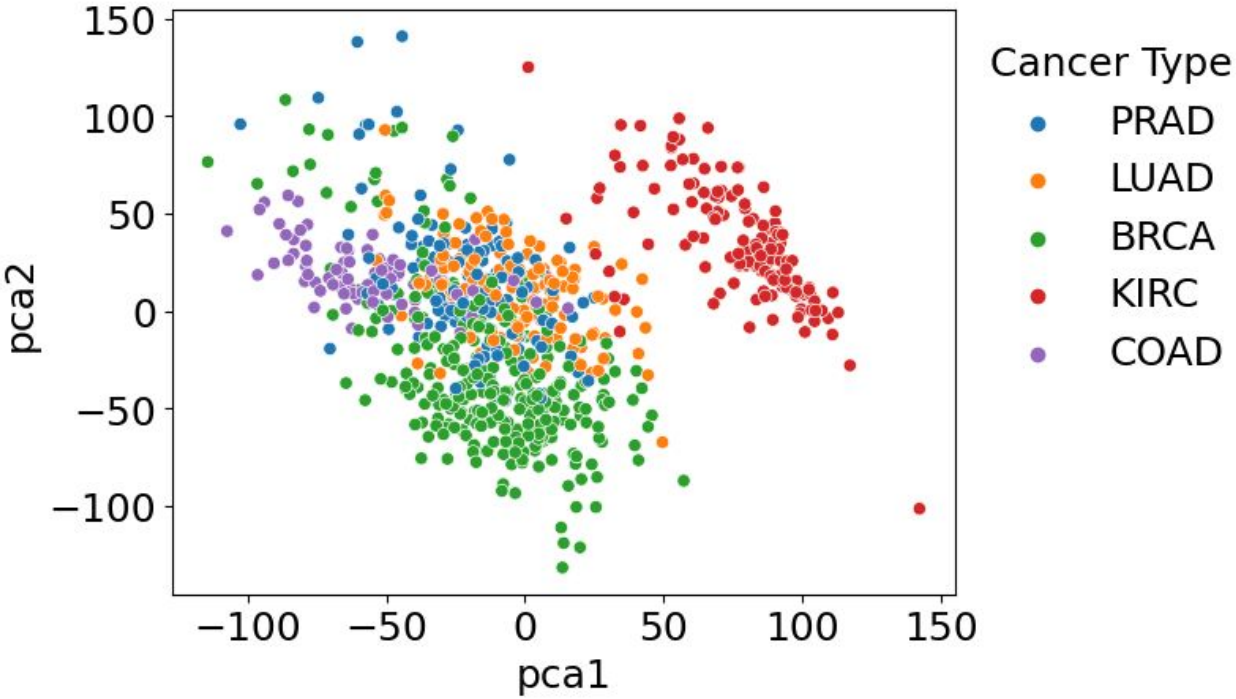
Transform the data:

- Input: Centered data and the previously calculated eigenvectors
- Output: Transformed data set

Evaluation – PCA libraries



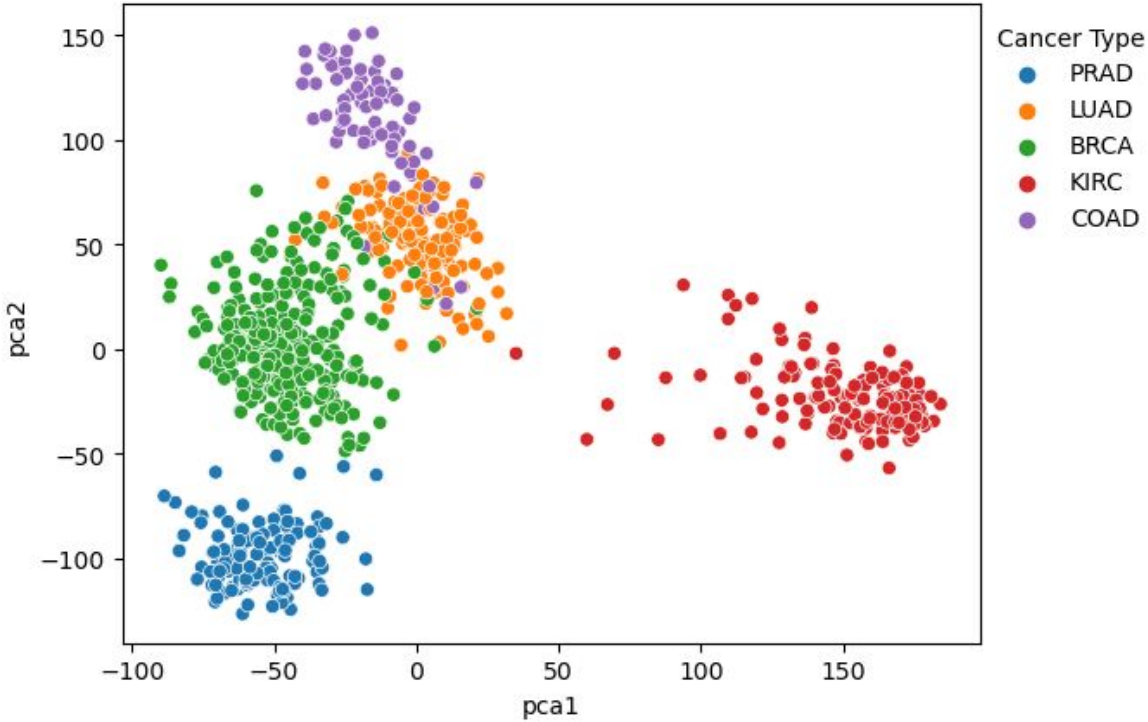
Evaluation – standardized data



sklearn PCA plotting first two components

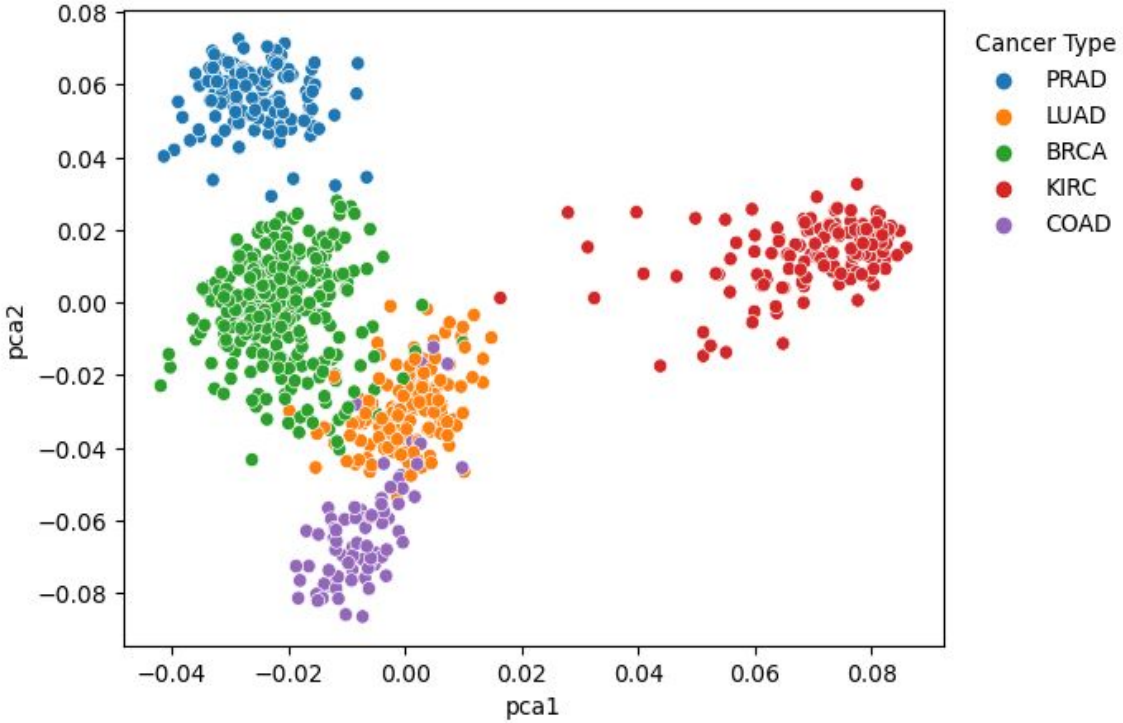
Evaluation – PCA libraries: sklearn

- Variance explained by first and second component: [0.1583855 0.1050396]
- Total variance explained with 2 components: 0.2634251
- Total variance explained with 129 components: 0.8002672



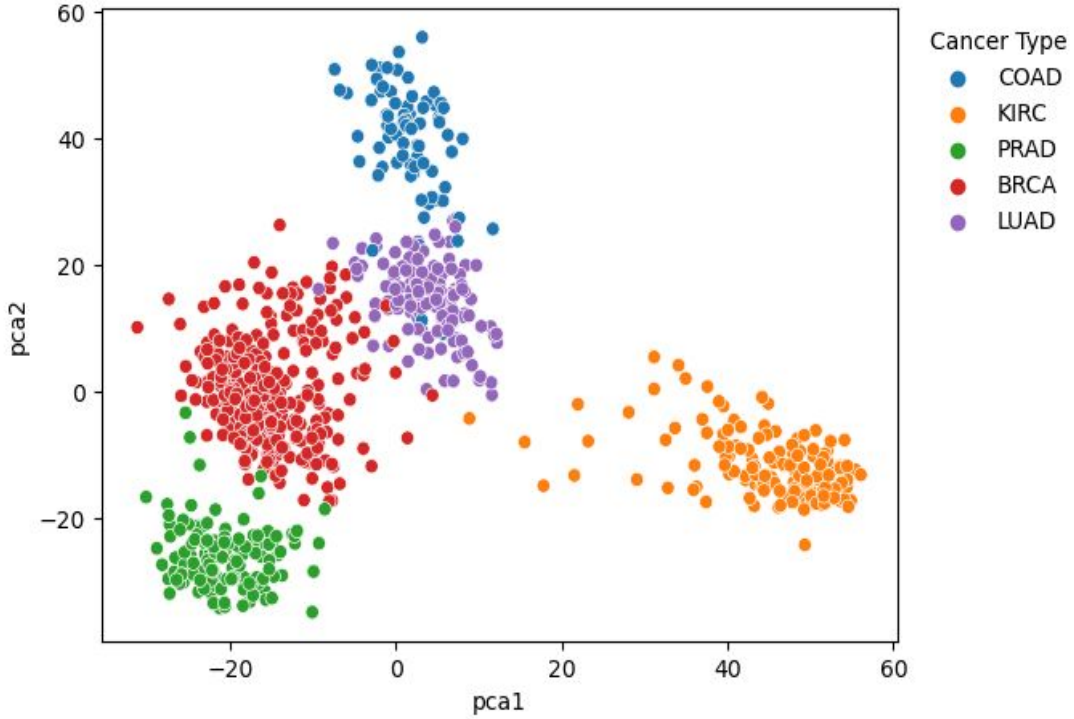
Evaluation – PCA libraries: statsmodels

- Variance explained by first and second component: [0.1583855 0.1050396]
- Total variance explained with 2 components: 0.2634251
- Total variance explained with 129 components: 0.8000135



Evaluation – own PCA (on data sample)

- Variance explained by first and second component: [0.1617865 0.1080658]
- Total variance explained with 2 components: 0.2698523
- Total variance explained with 92 components: 0.8008600



Evaluation – Comparison of own PCA and statsmodels

	eigenvec_0	eigenvec_1
0	-0.002904	-0.012407
1	0.011636	0.005962
2	0.003792	0.002665
3	-0.009405	-0.004983
4	-0.002216	-0.001000
...	...	...
1995	0.015478	0.000973
1996	-0.002956	0.003602
1997	0.001680	-0.001715
1998	-0.023535	-0.011840
1999	0.000709	0.006357

eigenvectors of own PCA

\-

	eigenvec_0	eigenvec_1
0	-0.002904	-0.012407
1	0.011636	0.005962
2	0.003792	0.002665
3	-0.009405	-0.004983
4	-0.002216	-0.001000
...	...	...
1995	0.015478	0.000973
1996	-0.002956	0.003602
1997	0.001680	-0.001715
1998	-0.023535	-0.011840
1999	0.000709	0.006357

eigenvectors of statsmodels

=

	eigenvec_0	eigenvec_1
0	7.329207e-17	1.613293e-16
1	1.994932e-16	6.938894e-17
2	1.847481e-16	6.418477e-17
3	1.578598e-16	3.512815e-16
4	1.301043e-17	1.173107e-16
...	...	...
1995	5.204170e-18	4.488597e-17
1996	5.637851e-18	3.035766e-18
1997	7.806256e-18	4.293441e-17
1998	0.000000e+00	1.353084e-16
1999	1.084202e-18	8.066464e-17

```
is_close_to_zero = np.isclose(check_df, 0.0,
                              atol=1e-10)
np.all(is_close_to_zero) -> True
```

Evaluation – Comparison of own PCA and sklearn

	eigenvec_0	eigenvec_1
0	-0.002904	-0.012407
1	0.011636	0.005962
2	0.003792	0.002665
3	-0.009405	-0.004983
4	-0.002216	-0.001000
...	...	...
1995	0.015478	0.000973
1996	-0.002956	0.003602
1997	0.001680	-0.001715
1998	-0.023535	-0.011840
1999	0.000709	0.006357

eigenvectors of own PCA

-

	eigenvec_0	eigenvec_1
0	0.002904	-0.012407
1	-0.011636	0.005962
2	-0.003792	0.002665
3	0.009405	-0.004983
4	0.002216	-0.001000
...	...	...
1995	-0.015478	0.000973
1996	0.002956	0.003602
1997	-0.001680	-0.001715
1998	0.023535	-0.011840
1999	-0.000709	0.006357

eigenvectors of sklearn

=

	eigenvec_0	eigenvec_1
0	0.005808	5.292219e-11
1	0.023273	3.901491e-11
2	0.007584	1.111204e-10
3	0.018810	2.558294e-11
4	0.004432	9.895155e-11
...	...	...
1995	0.030955	8.259762e-11
1996	0.005912	2.090068e-11
1997	0.003360	4.778626e-12
1998	0.047070	1.534545e-11
1999	0.001417	4.864997e-13

```
is_close_to_zero = np.isclose(check_df, 0.0,
                              atol=1e-10)
np.all(is_close_to_zero) -> False
```

Classification – own PCA

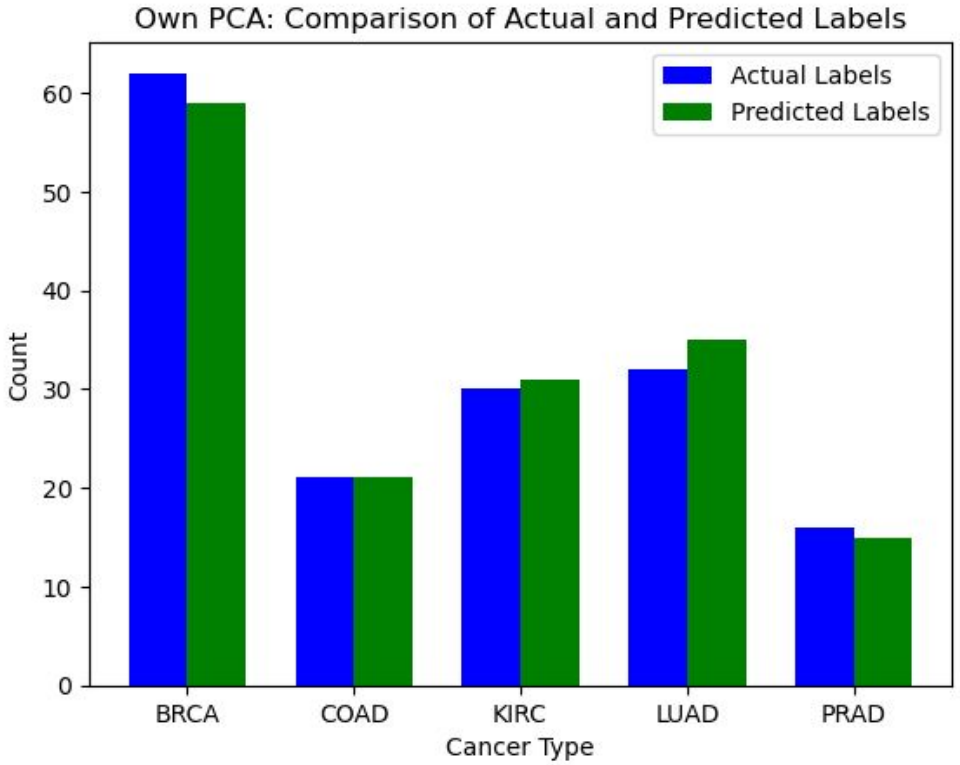
- Can the extracted components be used for classification into cancer types?
- Method: **Decision Trees** with **Cross-Validation**
  - without, own PC leads to only ~0.8-0.9 accuracy (same for sklearn)

Cross-validation results:

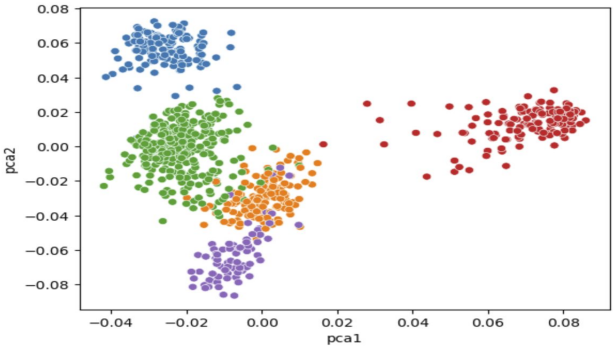
	original data set		reduced data set (2000 features)	
	original data set	2 sklearn components	2 sklearn components	2 own components
Accuracy	0.976	0.974	0.966	0.969
Precision	0.977	0.975	0.964	0.970
Recall	0.974	0.974	0.961	0.970
F1	0.976	0.974	0.962	0.969

Classification – own PCA

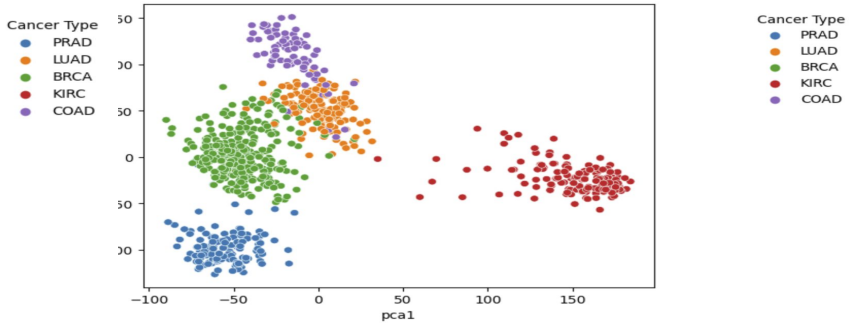
Results on reduced data set



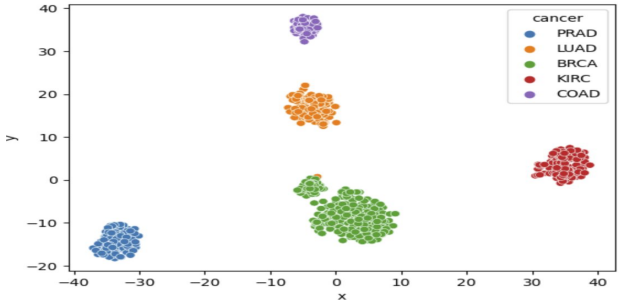
"Comparison of Computational Efficiency between t-SNE and PCA"



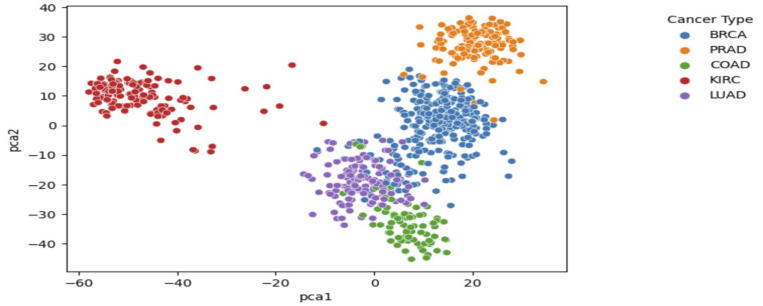
statsmodels(Time:109.23175406455994)



sklearn:(Time: 3.9830238819122314)



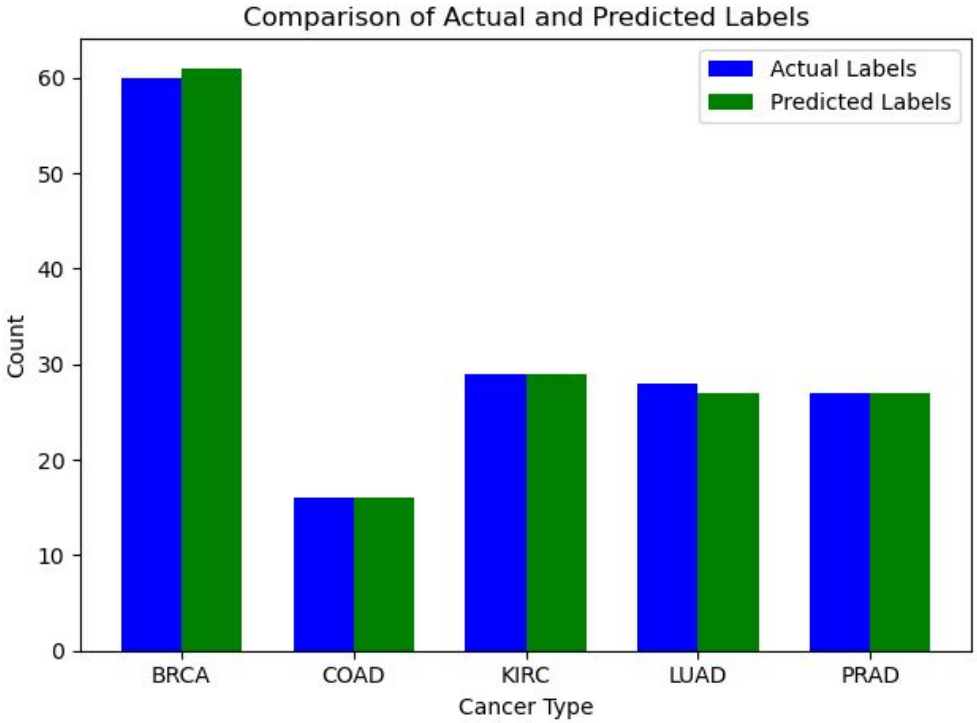
Tsne(Time: 11.370056867599487)



Own Pca:(Time: 5.4523210525512695)

Comparison with Actual and Predicted Labels

- Average performance metrics with cross-validation
- Average Accuracy: 0.9962500000000001
- Average Precision: 0.997249024199844
- Average Recall: 0.9964761904761904
- Average F1: 0.9968469106650926



Challenges we encountered

- Running time of our PCA (25 min. on Uni-Leipzig cluster - 32 GB, 8 × CPU)

Time Complexity of PCA

Let **m** be the number of objects (rows), **n** be the number of attributes (columns)

1. Centering the data: $O(mn)$
2. Calculating the covariance matrix: $O(m^3)$ naive and $O(m^{2.37188})$ optimally [2]
3. Calculating an eigenvector: $O(m^3)$

Ideas on how to improve our PCA

1. Use heuristic to compute eigenvectors (like power-iteration $O(m^2)$)
2. Only calculate the necessary eigenvectors
3. Work on sparse matrix

References

[1] ICMR Dataset. https://www.kaggle.com/datasets/shibumohapatra/icmr-data (last access: 05.07.2023)

[2] Duan, Ran, Hongxun Wu, and Renfei Zhou. "Faster matrix multiplication via asymmetric hashing." arXiv preprint arXiv:2210.10173 (2022).

UNIVERSITÄT
LEIPZIG

Thank You!

Introductory Computing for Data Science - Dimensionality reduction

Paula Hagen, Moritz Huhle, Sepideh Kaviapoor Esfahani, Paul Kühnel,
Karoline Schuster, Harpreet Int-Veen