Dimensionality Reduction with PCA and Non-Negative Matrix Factorization

Harold Okai

PCA

Introduction

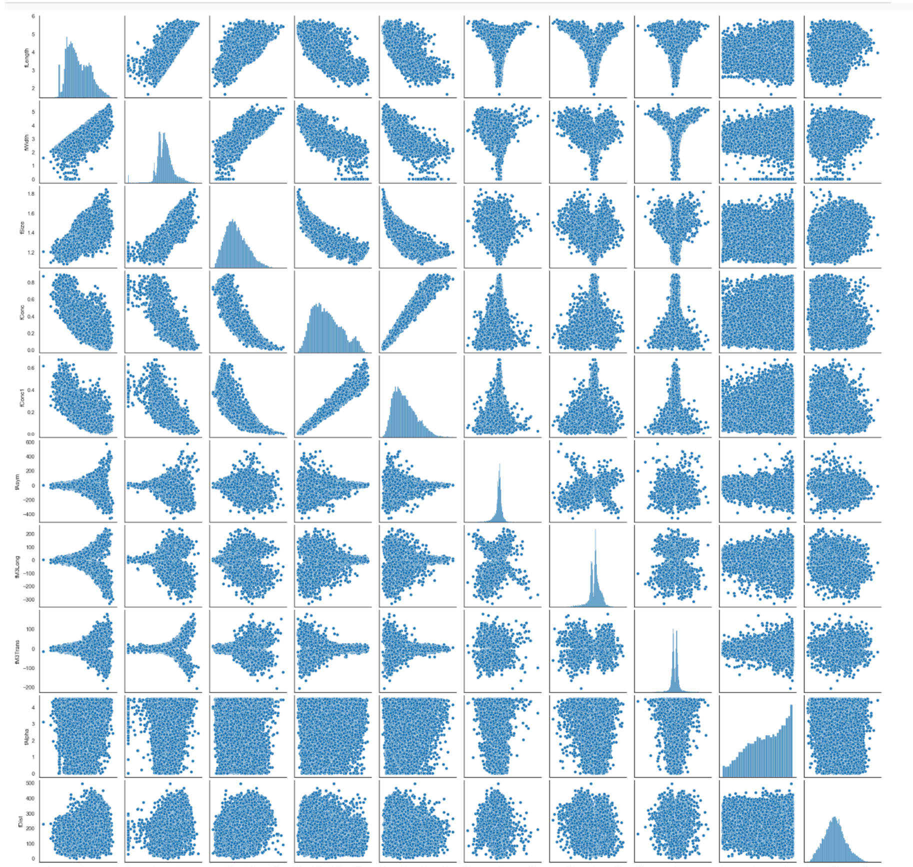
In this project my goal was to improve a Logistic Regression I worked on a couple of weeks ago. The dataset was acquired from the University of California Irvine's Machine Learning Repository. Later on I will also do some unsupervised clustering with NMF, of which dataset I acquired from Kaggle. The goal of this project was to show the techniques used in dimensionality reduction and unsupervised learning methods.

The Datasets

The first dataset involves a binary classification to simulate the classification of primary gammas (signal) and hadronic showers (background). The data had ten features including the binary feature of whether it be signal or background radiation.

```
                                         description  units missing_values
0                              major axis of ellipse     mm             no
1                              minor axis of ellipse     mm             no
2                 10-log of sum of content of all pixels  #phot             no
3        ratio of sum of two highest pixels over fSize   None             no
4                   ratio of highest pixel over fSize   None             no
5    distance from highest pixel to center, project...   None             no
6            3rd root of third moment along major axis     mm             no
7            3rd root of third moment along minor axis     mm             no
8            angle of major axis with vector to origin    deg             no
9            distance from origin to center of ellipse     mm             no
10               gamma (signal), hadron (background)   None             no
```

Here is the data showing the columns with skews with a threshold of over 0.75. Then a pairplot was constructed to show the relationships of the columns with each other.



After that using the Standard Scaler, I checked to see if there were any missing values, but that fortunately returned false. This is the correlation of the features.

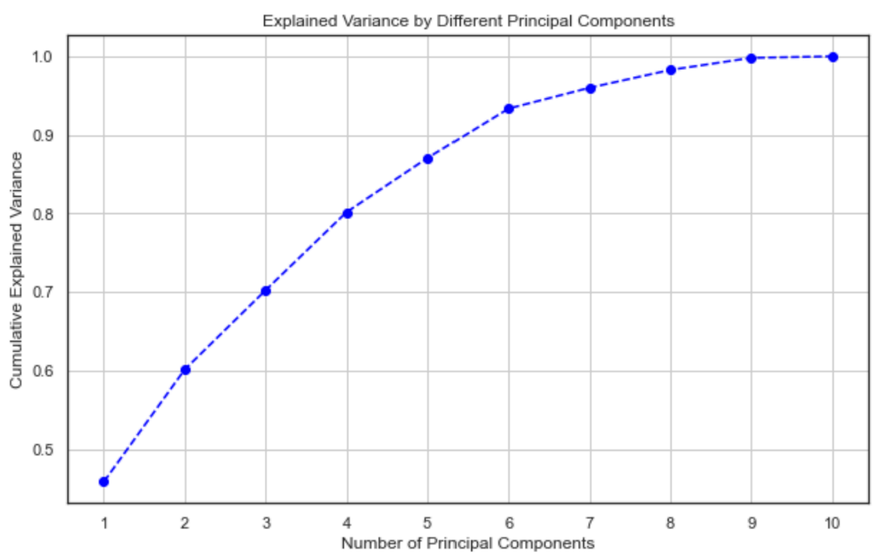
	fLength (scaled)	fWidth (scaled)	fSize (scaled)	fConc (scaled)	fConc1 (scaled)	fAsym (scaled)	fM3Long (scaled)	fM3Trans (scaled)	fAlpha (scaled)	fDist (scaled)
fLength (scaled)	1.000000	0.725438	0.800756	-0.785819	-0.750379	-0.280978	0.015461	0.010772	-0.204598	0.483181
fWidth (scaled)	0.725438	1.000000	0.786416	-0.771972	-0.739520	-0.226068	-0.054786	0.025357	-0.108442	0.369950
fSize (scaled)	0.800756	0.786416	1.000000	-0.870436	-0.826897	-0.155549	0.102162	0.014897	-0.280378	0.437004
fConc (scaled)	-0.785819	-0.771972	-0.870436	1.000000	0.976412	0.112272	-0.121899	-0.011294	0.294386	-0.328332
fConc1 (scaled)	-0.750379	-0.739520	-0.826897	0.976412	1.000000	0.100159	-0.118769	-0.010966	0.286200	-0.304625
fAsym (scaled)	-0.280978	-0.226068	-0.155549	0.112272	0.100159	1.000000	0.274045	0.002553	-0.052532	-0.206730
fM3Long (scaled)	0.015461	-0.054786	0.102162	-0.121899	-0.118769	0.274045	1.000000	-0.017197	-0.216126	0.037025
fM3Trans (scaled)	0.010772	0.025357	0.014897	-0.011294	-0.010966	0.002553	-0.017197	1.000000	0.005968	0.011427

Applying PCA

After applying PCA, here are some of the projections of each component onto others.

	Projection on Component 1	Projection on Component 2	Projection on Component 3	Projection on Component 4	Projection on Component 5	Projection on Component 6	Projection on Component 7	Projec Compo
0	0.962500	0.305616	0.916411	-1.324925	-0.063962	-0.427227	0.002376	-0.12
1	1.608331	0.690461	-1.014410	0.302893	-0.130574	0.309573	0.271516	0.60
2	-4.447499	-0.592512	0.536003	-3.227812	-0.607456	2.911260	0.729073	0.47
3	2.729489	0.230764	-0.347814	-0.114217	-0.668016	-0.101919	0.232733	0.68
4	-2.012905	0.315194	-0.497171	1.842128	0.444696	0.949483	0.293273	0.09

Next we find the explained variance with a threshold of 95%, we find that the number of principal components in the model is about 5-7.




```
threshold = 0.95
```

```
components = np.cumsum(pca.explained_variance_ratio_) < threshold
components.sum()
```

```
6
```

The fitted PCA data was then trained on a Logistic, with L2 penalty. After that, these are the metric we got

```
{'accuracy': 0.827900455660708,
 'recall': array([0.91108108, 0.67447657]),
 'precision': array([0.83772366, 0.80439952]),
 'f1score': array([0.8728638 , 0.73373102])}
```

Now we do a side by side comparison of the Logistic Regression with and without PCA.

```
{'accuracy': 0.7956536978618998,
 'recall': array([0.90108108, 0.60119641]),
 'precision': array([0.80648283, 0.76717557]),
 'f1score': array([0.8511616 , 0.67411962])}
```

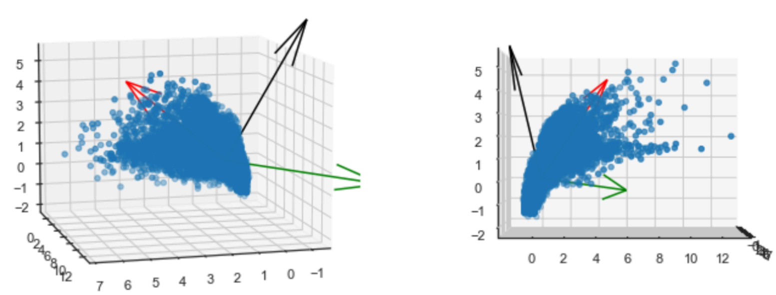
Before Applying PCA

```
{'accuracy': 0.827900455660708,
 'recall': array([0.91108108, 0.67447657]),
 'precision': array([0.83772366, 0.80439952]),
 'f1score': array([0.8728638 , 0.73373102])}
```

After Applying PCA

We can see that some of the scores in each iteration of the Logistic Regression had some scores increase and some of the scores decreased. For example the F1 score was much better before applying PCA, but the recall score went up from 90 to 91. This is because like I stated in the beginning, this dataset is not a prime candidate for PCA.

So even from a 3D view, no matter which angle you look at it from, there is no real improvement visually, since from all angles it is clear that the 3D plot still looks the same way. And there are no two real dominant directions of variance with respect to the

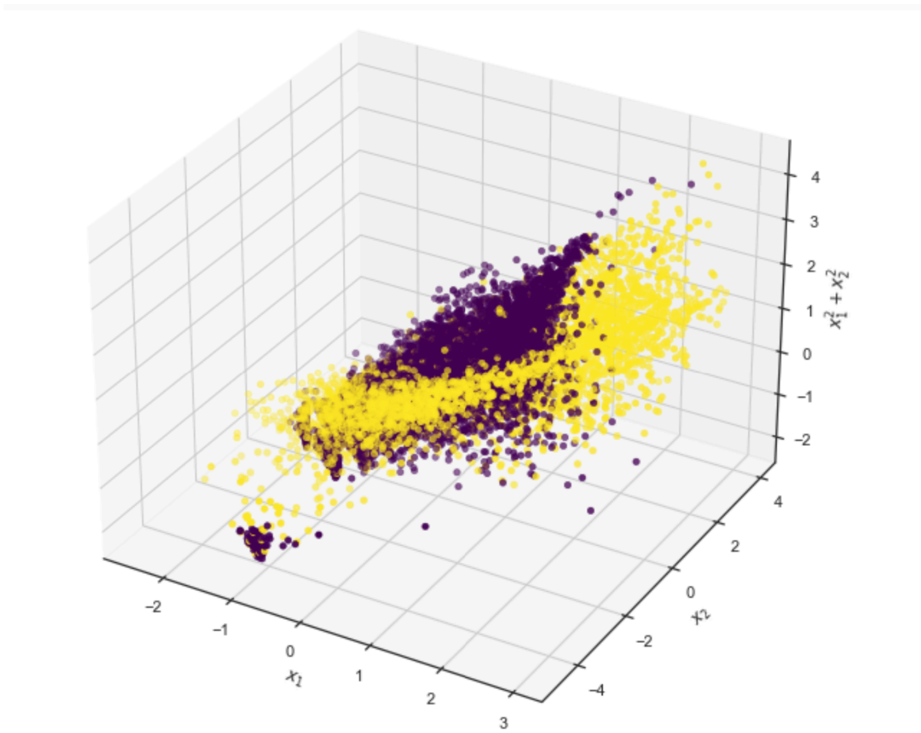



```
red component accounts for 82.02% of explained variance
black component accounts for 10.36% of explained variance
green component accounts for 7.62% of explained variance
```

original data, except for the red component.

Transforming the Dataset into a Higher Dimension

How about if we transformed the dataset into a higher dimension? After transforming the data into a higher dimension, we can see that there is still real improvement in terms of a linearly separable solution.



After applying PCA, we end up with a mean accuracy score that is less than what we started with.

```
lr= LogisticRegression().fit(PHI_train, y_train)
print(str.format("Test set  mean accuracy score for for Kernal PCA: {}"
```

```
Test set  mean accuracy score for for Kernal PCA: 0.831230283911672
```

Non Negative Matrix Factorization

The Dataset

The dataset used in this unsupervised clustering was acquired on Kaggle. It is a collection of news articles collected from the Huffington Post with corresponding topics and dates of publication etc.

	link	headline	category	short_description	authors	date
0	https://www.huffpost.com/entry/covid-boosters-...	Over 4 Million Americans Roll Up Sleeves For O...	U.S. NEWS	Health experts said it is too early to predict...	Carla K. Johnson, AP	2022-09-23
1	https://www.huffpost.com/entry/american-airlin...	American Airlines Flyer Charged, Banned For Li...	U.S. NEWS	He was subdued by passengers and crew when he ...	Mary Papenfuss	2022-09-23
2	https://www.huffpost.com/entry/funniest-tweets...	23 Of The Funniest Tweets About Cats And Dogs ...	COMEDY	"Until you have a dog you don't understand wha...	Elyse Wanshel	2022-09-23
3	https://www.huffpost.com/entry/funniest-parent...	The Funniest Tweets From Parents This Week (Se...	PARENTING	"Accidentally put grown-up toothpaste on my to...	Caroline Bologna	2022-09-23
4	https://www.huffpost.com/entry/amy-cooper-lose...	Woman Who Called Cops On Black Bird-Watcher Lo...	U.S. NEWS	Amy Cooper accused investment firm Franklin Te...	Nina Golgowski	2022-09-22

The analysis was only limited to only 10,000 articles, because as we can see the entire data is quite large.

```
boston_doj_pressrelease_df.shape
```

```
(209527, 6)
```

Applying NMF

There were 29 unique topics dataset, that is according to the 10,000 index cutoff.

```
# Surpress warnings from using older version of sklearn:
def warn(*args, **kwargs):
    pass
import warnings
warnings.warn = warn

from sklearn.decomposition import NMF
model = NMF(n_components=29, init='random', random_state=818)
doc_topic = model.fit_transform(tfidf_mat)

doc_topic.shape
# we should have 10000 observations (articles) and five latent features
```

```
(10000, 29)
```

```
model.components_.shape
```

```
(29, 2000)
```

These are dataframe showing how each word contributes to the various clustered topics.

	abc	able	abortion	abortions	absolutely	abuse	academy	access	according	account
topic_1	0.004	0.001	0.000	0.0	0.002	0.000	0.000	0.001	0.000	0.002
topic_2	0.000	0.000	0.000	0.0	0.000	0.000	0.000	0.000	0.009	0.002
topic_3	0.003	0.000	0.015	0.0	0.000	0.000	0.000	0.004	0.001	0.012
topic_4	0.003	0.000	0.000	0.0	0.000	0.009	0.001	0.001	0.028	0.000
topic_5	0.000	0.000	0.000	0.0	0.003	0.000	0.000	0.003	0.000	0.000

5 rows × 2000 columns

The dataframe below also shows the topics and which category it most relates to.

```
topic_1             CRIME
topic_2             MEDIA
topic_3         EDUCATION
topic_4      LATINO VOICES
topic_5          POLITICS
topic_6          POLITICS
topic_7             WOMEN
topic_8      ENTERTAINMENT
topic_9            COMEDY
topic_10        PARENTING
topic_11            WOMEN
topic_12            CRIME
topic_13         POLITICS
topic_14         POLITICS
topic_15             TECH
topic_16           COMEDY
topic_17         POLITICS
topic_18            MEDIA
topic_19          SCIENCE
topic_20         BUSINESS
topic_21           TRAVEL
topic_22         WELLNESS
topic_23            CRIME
topic_24        EDUCATION
topic_25         WELLNESS
topic_26     ENTERTAINMENT
topic_27            CRIME
topic_28           IMPACT
topic_29             TECH
dtype: object
```

Topic 1 mostly relates to Crime, topic 2 mostly to Media and so on.

```
topics_word.T.sort_values(by='topic_1', ascending=False)
```

	topic_1	topic_2	topic_3	topic_4	topic_5	topic_6	topic_7
said	0.651	0.000	0.000	0.000	0.000	0.000	0.000
officials	0.030	0.001	0.094	0.019	0.001	0.085	0.000
official	0.024	0.015	0.000	0.005	0.000	0.009	0.011
authorities	0.022	0.000	0.053	0.000	0.000	0.000	0.000

```
topics_word.T.sort_values(by='topic_2', ascending=False
```

	topic_1	topic_2	topic_3	topic_4	topic_5	topic_6
trump	0.000	0.790	0.001	0.000	0.000	0.000
donald	0.000	0.359	0.000	0.000	0.022	0.000
administration	0.001	0.062	0.008	0.016	0.000	0.001
campaign	0.000	0.061	0.000	0.020	0.006	0.000
election	0.000	0.039	0.001	0.000	0.000	0.033

```
topics_word.T.sort_values(by='topic_29', ascending=False)
```

	topic_1	topic_2	topic_3	topic_4	topic_5	topic_6	topic_7
media	0.000	0.00	0.000	0.000	0.000	0.000	0.000
social	0.000	0.00	0.000	0.000	0.000	0.000	0.000
called	0.000	0.01	0.000	0.008	0.029	0.005	0.000
right	0.001	0.00	0.007	0.000	0.002	0.000	0.064
twitter	0.000	0.00	0.001	0.000	0.037	0.000	0.000

Finally we look at how each of the words relates to the topics and the category. We can see that with topics 1 which mostly relates to crime we have words like, 'said', 'officials', 'authorities' etc. With topic 2 which is related to Media we have words like 'trump', 'administration', 'election' etc. For Tech we have 'media', 'social', 'twitter' etc.

Conclusion

In conclusion our machine learning could benefit from a more 'apt' dataset when it comes to Logistic Regression. Probably a dataset that was more "cyclic" and prone to inaccuracies when it comes to prediction. The dataset was okay without PCA, but we cannot inject our own sentiments to a dataset as data comes as it is collected. With NMF we could have done better with a much larger dataset. That is including the whole dataset in the analysis, but for the lack of computational resources we had to limit the data. Also a better approach would have been to use more deep learning methods as they are more suited to Natural Language Processing (NLP).

References

Wolberg,William, Mangasarian,Olvi, Street,Nick, and Street,W.. (1995). Breast Cancer Wisconsin (Diagnostic). UCI Machine Learning Repository. https://doi.org/10.24432/C5DW2B.

D. Heck et al., CORSIKA, A Monte Carlo code to simulate extensive air showers, Forschungszentrum Karlsruhe FZKA 6019 (1998). http://rexa.info/paper?id=ac6e674e9af20979b23d3ed4521f1570765e8d68

Ismael Mousa. (2024). Raw Google News [Data set]. Kaggle. https://doi.org/10.34740/KAGGLE/DS/5094567