

786: Artificial Intelligence & Machine Learning Fundamentals

Project - Comparing Feature Extraction Approaches

Bilal Zeidan
#400353165
November 16, 2020

Submitted To: Dr. Jeff Fortuna

Data Info

The data we are dealing with is collected from online shoppers in an e-shop website environment. Each row belongs to a unique user and describes a session of browsing through the website. During the session, the user either ends up with purchase or without any purchase. The data has 17 features and 12,330 sessions(rows) collected over a period of 1 year. Some of those features are categorical, ordinal or binary and we have chosen to drop them as they do not fit well with PCA.

The data can be obtained from the UCI Machine Learning Repository through [this link](#). The original authors of this data published a paper describing the use of neural networks, multilayer perceptron and LSTM recurrent neural networks for real time predictions. (Sakar et al., 2019, 1)

Our goal here is not to come up with the best algorithm that solves the problem but to compare the use of features extraction using forward search and dimensionality reduction using PCA.

Descriptive statistics

After loading that csv file, we made sure we don't have any missing data. There was no missing data, Otherwise we would have removed the missing data rows for simplicity. Since we are going to use PCA, it's better to drop the features of categorical, ordinal or binary type. This reduced the data to 10 features.

Data Centering & Scaling

The data needs to be centered for PCA and that requires removing the mean from the input. Also we have opted to scale the data through dividing by the standard deviation. This will help mitigate any effects from outliers on PCA and will put all data in the same range. In Some cases this might not be a good idea, because when we scale by standard deviation unit we kinda squish the data into a small region and this might make it harder for PCA to find components with higher variance.

Data visualizations

Before we start our experiments, we need to have a look at the data and decide on the train/test split. The bar chart below compares the online store visits that resulted in no purchase and purchase. It's clear that a lot more people will end up not purchasing and very few will complete the process till the end. This shows that our data is not very balanced and during splitting we might end up with very few numbers of the True class. The best way to handle this, would be to use cross validation. This ensures a balanced split of classes and gives better results. But, because our focus is on feature extraction and to keep things simple, we went with doing a random split and checked the splits using a bar graph.

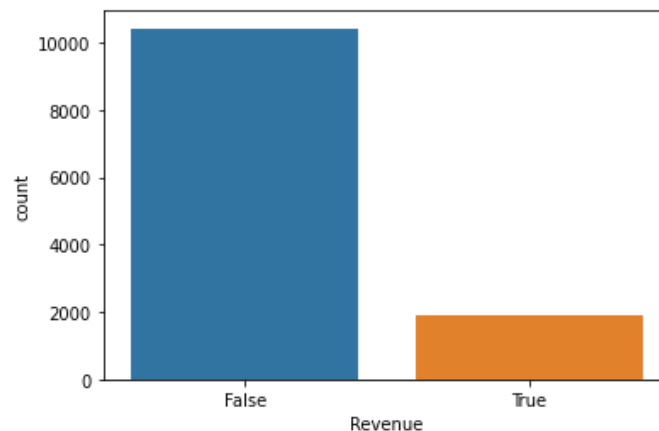



Figure 1 - Data Classes (Trues & Falses)

Experiments

We started by splitting the data as we mentioned earlier by using a random split technique. We decided to allocate 75% of the data for training and 25% for testing. We could have used tools from sklearn that do the splitting, but to be able to understand how this really works we implemented our own splitting function. After splitting the data we ended up with 9248 training sessions and 3082 testing sessions.

PCA

To perform PCA we need to estimate the mean in order to center the data and calculate the covariance. Our assumption would be that the data is normally distributed and therefore we estimate the mean using the equation below.


$$\mu = \sum_i^N \frac{x_i}{N}$$

We can then find the covariance matrix which will be a positive definite matrix and thus we can easily perform an eigendecomposition to diagonalize the our matrix and find its eigenvalues and eigenvectors. This helps us find the principal axes i.e the eigenvectors which are orthonormal vectors. Later we can sort them according to the highest eigenvalues which reflect the variance associated with each principal axes.

After performing PCA to find the principal components, we can now get rid of the least important components(ones that correspond to the least eigenvalues). We can do this by dropping the corresponding eigenvectors.

After reducing each component we measured the reconstruction error for PCA. Then, we used two classifiers LDA and Decision Tree to perform classification on the reduced matrix. During each iteration we stored the confusion matrix and the computational time which we will later discuss further.

Forward Search

Forward search attempts to find the best features in a dataset and extract them. Those features are the best features found that represent the data and can be used to slice the training and testing data.

During forward search, each feature/group of features is trained against a classifier of choice. We are using an LDA classifier, although we have tried to use a Decision Tree just for testing it out. That resulted in different features being selected. And that is expected as the forward search attempts to calculate the classification errors and that's how the best features are found and extracted.

Computational Times

PCA

We have calculated the computational time of every part of our process. However the initial computation of the covariance matrix and the eigendecomposition is not taken into account in this table as this is only done once(a separate timer was used for that) and is not part of the loop.

As more features are removed during PCA the data training becomes faster. However the testing did not get affected that much and that could be due to the small size of the test data. In general the times in the table are very small and that also could be the small amount of features in total. Some datasets have hundreds of features and the time difference can be spotted better. However, it's obvious that with less features the computational time of training and testing will be reduced.

Removed	PCA	LDA Training + PCA	LDA Testing	DT Training + PCA	DT Testing
0 Features	0.0062	0.0273	0.0016	0.0512	0.0018
1 Features	0.0016	0.0227	0.0016	0.0471	0.0017
2 Features	0.0016	0.021	0.0017	0.0466	0.0018
3 Features	0.0015	0.019	0.0016	0.0369	0.0017
4 Features	0.0014	0.0167	0.0016	0.0318	0.0016
5 Features	0.0015	0.0155	0.0015	0.027	0.0017
6 Features	0.0014	0.0128	0.0013	0.0221	0.0018
7 Features	0.0013	0.0111	0.0016	0.0244	0.0017

Table 1 - PCA Computational Time Table

From the graph below we can see that PCA itself did not get affected by how many components we removed. But the classifiers computational time indeed did get reduced and almost linearly while using LDA. (Note that the x-axis is nothing but the an index of the rows in the table above with 0 corresponding to the first row)

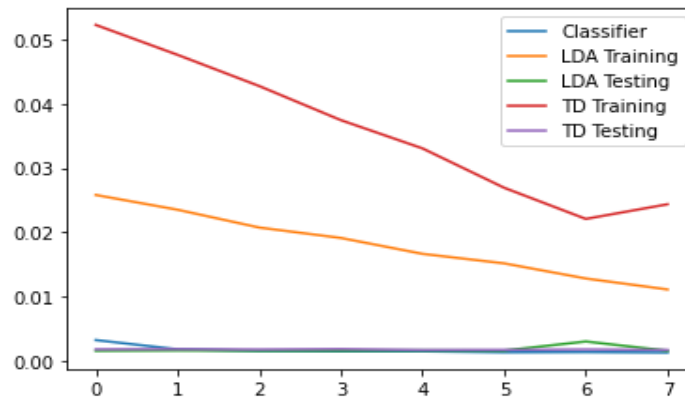


Figure 2 - PCA Computational Time Graph

Forward Search

The table below represents the times captured during forward search feature extraction. Forward search time decreases almost linearly with less features being extracted.

Removed	FS	LDA Training + FS	LDA Testing	DT Training + FS	DT Testing
7 Features	0.2843	0.0106	0.0016	0.0512	0.0018
6 Features	0.122	0.0138	0.0016	0.0144	0.0018
5 Features	0.1173	0.0157	0.0017	0.0193	0.0018
4 Features	0.1176	0.0171	0.0016	0.021	0.0018
3 Features	0.0992	0.0199	0.0017	0.022	0.0018
2 Features	0.0851	0.0216	0.0016	0.0232	0.0018
1 Features	0.0566	0.024	0.0017	0.0328	0.0018
0 Features	0.0305	0.0271	0.0031	0.0381	0.0018

Table 2 - Forward Search Computational Time Table

From the graph below we can notice that forward search algorithms which should be read opposite will show a linear increase in time and maybe even a quadratic one as more features need to be extracted (Note that the x-axis is nothing but the an index of the rows in the table

above with 0 corresponding to the first row). Also, the training and testing times slightly increase as more features are present in the data.

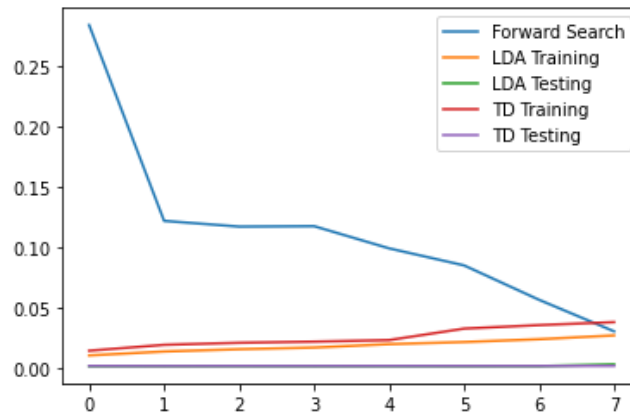


Figure 4 - Forward Search Computational Time Graph

Results

PCA

The x-axis in Figure 5 represents how many features are retained. We can see that as more features are removed reconstruction error increases. This is a great representation of how much data we are losing with every principle component reduction.

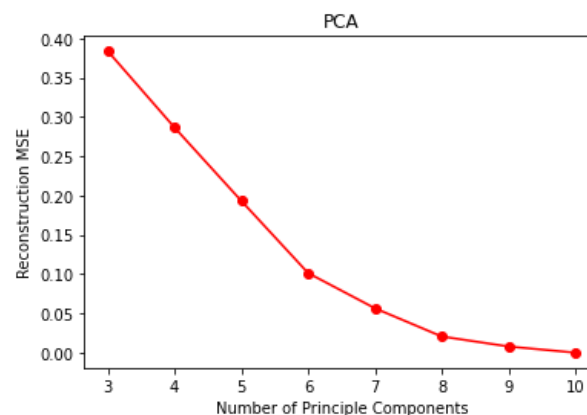


Figure 5 - PCA Reconstruction Error

After using the two classifiers we can notice that the classification error did not get affected much till around feature 5 or 6. That means that features 5 up till 10, had little effect on the classification error.

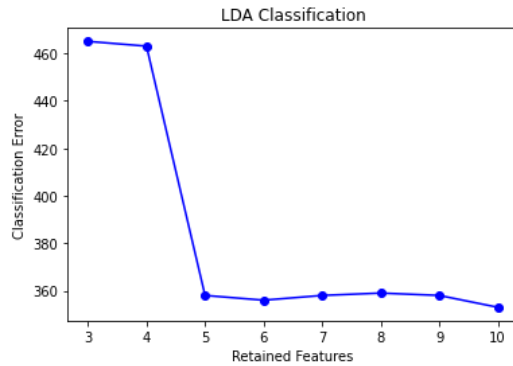


Figure 6 - LDA

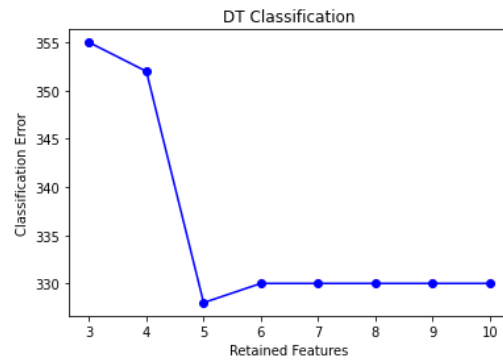


Figure 7 - Decision Tree

Confusion matrices below are ordered from 0 feature extraction at the top left up to 7 feature extraction at the bottom right. The scores are not changing much for the true positives but the true negatives are. In fact with full feature selection we got better true positives and true negatives than any other during LDA.

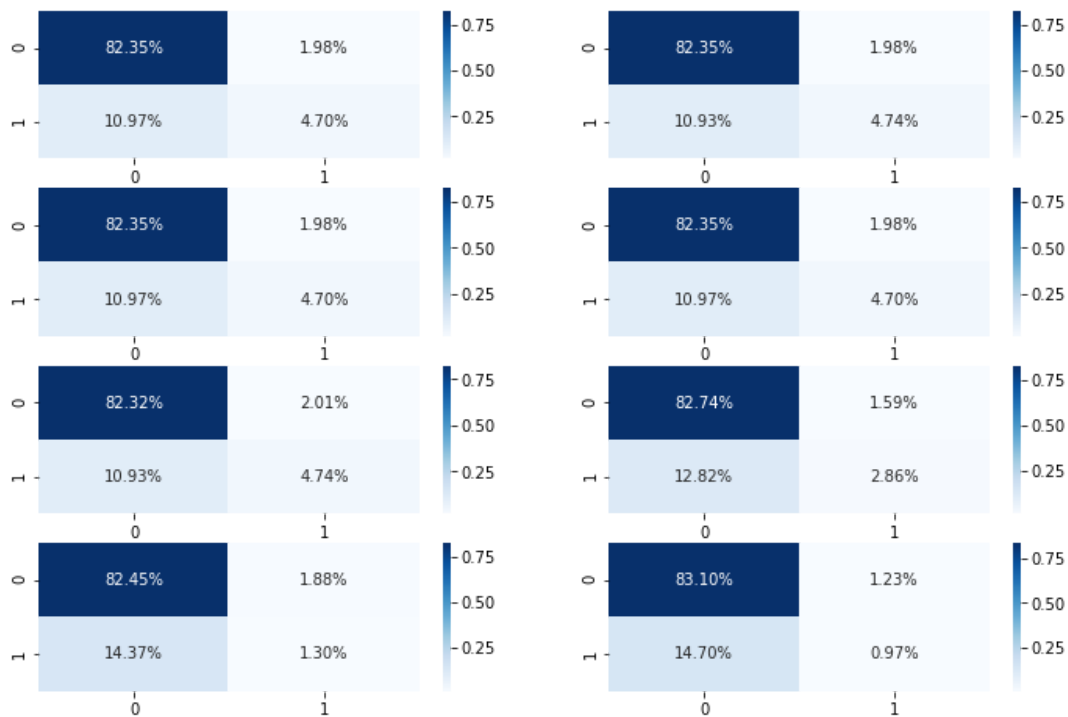


Figure 7 - Forward Search LDA Confusion Matrices

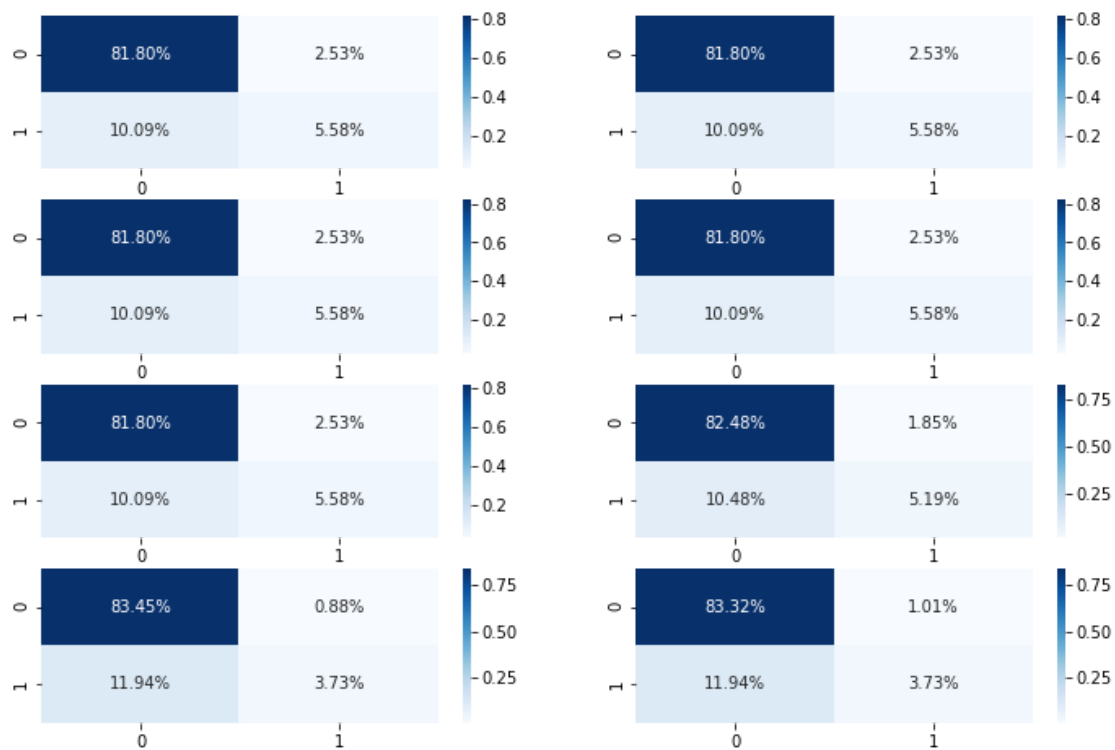


Figure 9 - Figure 7 - Forward Search DT Confusion Matrices

Forward Search

Applying LDA after the forward search did not affect the results at all. However in the Decision tree the results did vary and the more features are removed the worse the classification error was getting.

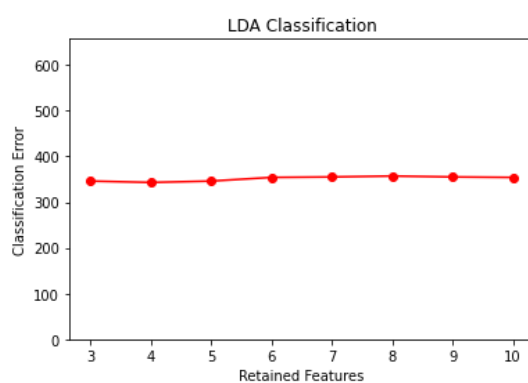


Figure 10 - LDA

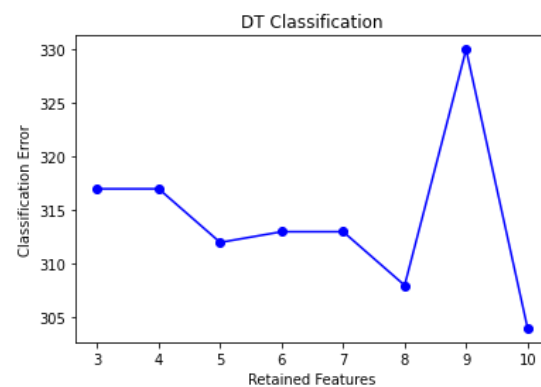


Figure 11 - Decision Tree



Further Discussion

First the computational time of PCA is obviously less than performing the forward search. However during training of the data the forward search performed better and that makes sense as we are completely removing features with forward search.

On the other side, during PCA the number of features in the reconstructed matrix will stay the same but the representation of the features is different as PCA attempts to find a linear transformation between the features.

If we look again at the classification errors and compare them we can find the results were better with all features. However, reducing some features did not affect the error much and we were able to use less computational time.

In general we can say that in both PCA and Forward Search there is always some sort of loss of information that can not be mitigated. A feature selection technique can be beneficial for very large dimensional datasets. But if we have a lower number of features and we can include all the independent features in our training then that should be better.