Department of Computer Science & Informatics

Masoud Nateghi

CS534 HW5

Professor Joyce Ho

For a hypersphere of radius a in d dimensions, the volume is related to the surface area of a unit hypersphere (S) as

$$V = \frac{S \times a^d}{d}$$

(a) Use this result to show (via a formal mathematical proof) that the fraction of the volume that lies at values of the radius between a − ϵ and a, where 0 < ϵ < a, is given by $f = 1 - \left(1 - \frac{\epsilon}{a}\right)^d$. Hence, show that for any fixed ϵ, no matter how small, this fraction tends to 1 as d ⟶ ∞.

The portion of the volume which lies at the values of the radius between a − ϵ and a, where 0 < ϵ < a can be stated by the formula below:

$$\Delta V = \frac{S \times a^d}{d} - \frac{S \times (a-\epsilon)^d}{d} = \frac{S}{d}\left(a^d - (a-\epsilon)^d\right) = \frac{Sa^d}{d}\left(1 - \left(1 - \frac{\epsilon}{a}\right)^d\right)$$

Therefore, we have

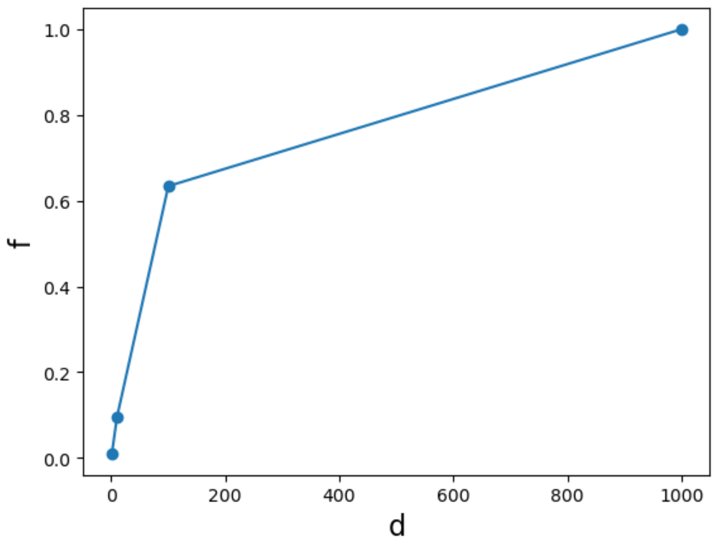
$$f = \frac{\Delta V}{V} = 1 - \left(1 - \frac{\epsilon}{a}\right)^d$$

We know that

$$0 < \epsilon < a \rightarrow 0 < \frac{\epsilon}{a} < 1 \rightarrow 0 < 1 - \frac{\epsilon}{a} < 1$$

If $d \rightarrow \infty$ then $\left(1 - \frac{\epsilon}{d}\right)^d \rightarrow 0$ thus we $\lim_{d\rightarrow\infty} f = \lim_{d\rightarrow\infty} 1 - \left(1 - \frac{\epsilon}{a}\right)^d = 1$

(b) Evaluate the ratio f numerically by plotting the results for different values of ϵ/a = 0.01 and d = 1, 10, 100, and 1000.



(c) What conclusions can you draw from the plot?

As d increases, f tends to be 1 regardless of the value of ϵ as we proved in part (a). If $\frac{\epsilon}{d}$ was less than 0.01, it would be necessary to select larger values for d to ensure that f approaches 1. Furthermore, it's worth noting that ϵ can take any value between 0 and a. Thus, as the dimensionality, d, increases, the fraction, f, tends toward 1 irrespective of the chosen value for ϵ. This observation underscores the concept that in higher dimensions, the volume of the hypersphere is mainly concentrated near its surface.

2. (a) How did you partition the loan data to assess the model performance and choose the hyperparameter (model selection)? This can be repeated from Homework #3 and Homework #4. You might find it useful to read all of this homework to decide if you want to change your assessment strategy.

I initially conducted preprocessing on the data, converting categorical features into numerical features, as outlined in 2b.

Then, I divided the data into separate train and test datasets, reserving 20% of the total data for the test set, selected randomly. For models requiring hyperparameter tuning, such as decision trees, random forests, support vector machines (SVMs), and neural networks, I will utilize 5-fold cross-validation. As for the simple logistic regression model, I will train it using 80% of the training dataset and evaluate its performance on 20% of training dataset for validation and the designated test dataset, reporting the resulting metrics.

In the feature selection section (part 2d), we exclusively utilize the training data for selecting features. It's important not to incorporate the test data during the feature selection process, as this can lead to an overly optimistic model performance.

2. (b) Pre-process the loan default.csv data from Homework #3 to deal with categorical data. Specify how you are specifically converting the non-numeric data.

I first identified the categorical features as follows:

- term
- grade
- emp_length
- home_ownership
- verification status
- purpose
- earliest_cr_line

Regarding the ‘grade’ feature, the entries were ‘A’, ‘B’, ‘C’, ‘D’, ‘E’, ‘F’, ‘G’, which I mapped to the integers 1, 2, 3, 4, 5, 6, 7, respectively.

In the case of the ‘emp_length’ feature, I utilized the following mapping for this column:

```
emp_length_mapping = {'< 1 year': 1, '1 year': 2, '2 years': 3, '3 years': 4, '4 years': 5, '5 years': 6,
                      '6 years': 7, '7 years': 8, '8 years': 9, '9 years': 10, '10+ years': 11}
```

Furthermore, I noticed that some entries in certain columns were designated as ‘N/A’. To handle these missing values, I replaced them with zeros.

For the ‘home_ownership’, ‘verification_status’, and ‘purpose’, ‘term’ columns, I employed a one-hot encoder to convert the categorical data into a numerical format.

I removed ‘id’, and ‘earliest_cr_line which were not that much informative.

2. (d) Based on your data from 2c, your model assessment strategy in (a), and your functions from Homework #3, perform feature selection to determine which variables you plan to keep. You should provide the rationale for your feature selection process. You may find it useful to save the train/test data as a separate file (e.g., loan fs train.csv, loan fs test.csv) as you will be analyzing its performance below.

In the feature selection section (part 2d), we exclusively utilize the training data for selecting features. It's important not to incorporate the test data during the feature selection process, as this can lead to an overly optimistic model performance.

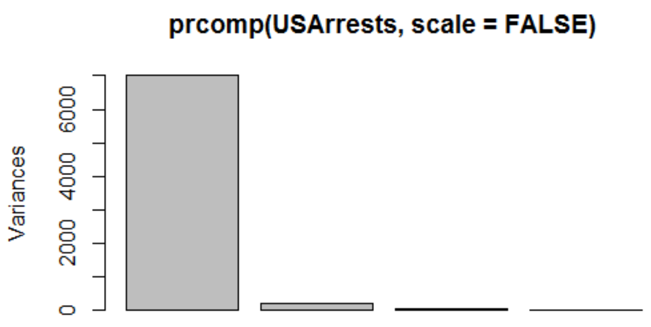
I used the exact same strategy used in HW4 and HW3. Following pre-processing stage, the dataset was divided into training and testing sets. Subsequently, I computed the correlation matrix between the features to identify and remove one of the features from any highly correlated pair. Columns 0, 2, 6, 10, 12, 19, 21, 26, and 28 were consequently removed resulting in a total of 32 features.

Subsequent to this, I computed the correlation between the features and labels to select features displaying strong correlations, whether positive or negative. Such features were retained for the subsequent analysis, as a strong correlation with the label can indicate their informative and practical nature. Ultimately, 25 features with the highest positive or negative correlations were selected, resulting in a total of 25 features after the preprocessing stages.

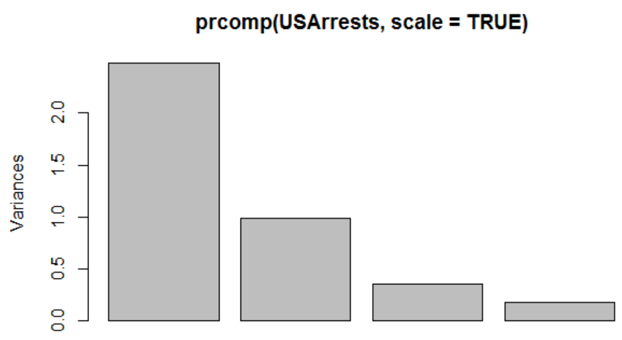
2. (e) Discuss why you should normalize the data before performing PCA.

Adopted from: https://stats.stackexchange.com/questions/69157/why-do-we-need-to-normalize-data-before-principal-component-analysis-pca

Normalization is important in PCA since it is a variance maximizing exercise. It projects original data onto directions which maximize the variance. FOR INSTANCE, the first plot below shows the amount of total variance explained in the different principal components where we have not normalized the data. As you can see, it seems like component one explains most of the variance in the data.



If we look at the second picture, we have normalized the data first. Here it is clear that the other components contribute as well. The reason for this is because PCA seeks to maximize the variance of each component.



Thus, we can see that if we do not normalize the data, features with a high variance will have a higher weight compared to features with a small variance distorting their relevance when computing the PCA.

2. (g) Using the PCA function you wrote above in 2f, report how many components were needed to capture 95% of the variance in the normalized data. Discuss what characterizes the first 3 principal components (i.e., which original features are important).

We determined that 31 principal components (PCs) are required to account for 95% of the original data's variance. In examining the significance of the features, we observed the loading entries and sorted them in descending order based on their absolute values. We can infer that features associated with larger loading values hold more importance. This conclusion is supported by the prioritization of features with larger coefficients in the linear combination of all the features, indicating their greater contribution to explaining the data's variability. Specifically, for each PC, we identified the first 5 critical features are as follows:

PC1: ‘int_rate’, ‘grade’, ‘term’, ‘home_ownership_RENT’, ‘verification_status_Not Verified’

PC2: ‘home_ownership_RENT’, ‘home_ownership_MORTGAGE’, ‘purpose_home_improvement’, ‘total_rev_hi_lim’, ‘revol_bal’

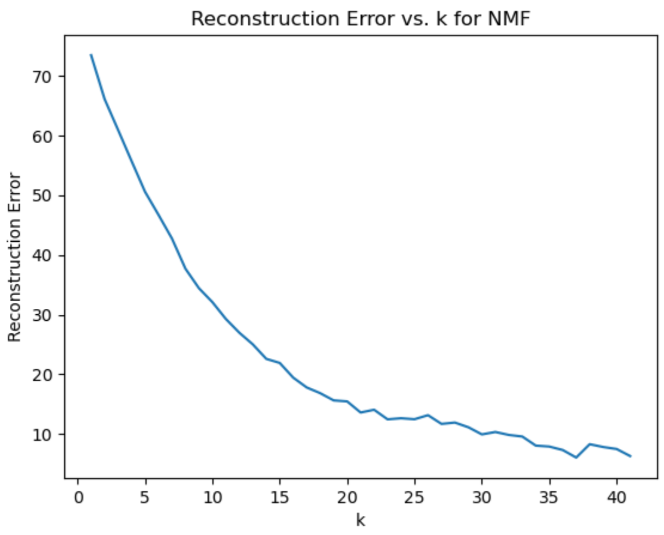
PC3: ‘purpose_debt_consolidation’, ‘purpose_credit_card’, ‘total_acc’, ‘open_acc’, ‘dti’

2. (h) Using the PCA function you wrote above in 2f, transform your train and test data using PCA. You might find it useful to save the train/test data as a separate file (e.g., loan pca train.csv, loan pca test.csv)

Done!

2. (j) Using the function above, 2i, determine the optimal number of components. You should justify your choice via a plot of the reconstruction error as a function of k. Discuss what characterizes the first 3 factors (i.e., which original features are important).

We first normalized the input data for the function run_nmf using minmax scaling to scale data into 0 to 1. Then, we utilized varying numbers of components for performing NMF and calculating the reconstruction error.



Based on the figure above, it appears that 30 components would suffice for a successful reconstruction of the original data.

To analyze the significance of the features, we examined the loading entries, arranging them in descending order based on their absolute values. It can be assumed that features associated with larger entry values are more critical. We observed that for each factor, the first 5 significant features are:

F1: 'verification_status_Verified', 'int_rate', 'purpose_house', 'recoveries', 'annual_inc'

F2: 'purpose_credit_card', 'dti', 'open_acc', 'loan_amnt', 'installment'

F3: 'home_ownership_MORTGAGE', 'open_acc', 'tot_cur_bal', 'loan_amnt', 'installment'

2. (k) Using the NMF function you wrote above in 2i, transform your train and test data using NMF and the optimal k you found from the previous question. You might find it useful to save the train/test data as a separate file (e.g., loan nmf train.csv, loan nmf test.csv)

Done!

Datasets for parts b, d, h, and k are saved in, trainx_b/testx_b, trainx_d/testx_d, trainx_h/testx_h, and trainx_k/testx_k respectively.

3. (f) For each of these 5 models above, build a model for each of the 4 datasets (2b, 2d, 2h, 2k) and report the results in a Table (you should have 20 model+dataset combinations across the 3 metrics). Note that for the original dataset, you may find it useful to preprocess it using Standardization or MinMaxScaling.

	Partition	Train			Validation			Test		
Model	Dataset	AUC	F1	F2	AUC	F1	F2	AUC	F1	F2
Logistic Regression	B	0.7270	0.6245	0.5096	0.6990	0.5699	0.4575	**0.7369**	**0.6428**	**0.5314**
	D	0.7270	0.6245	0.5096	0.7022	0.5759	0.4590	**0.7382**	**0.6454**	**0.5321**
	H	0.7344	0.6369	0.5363	0.6844	0.5427	0.4448	0.7257	0.6206	0.5252
	K	0.5627	0.2752	0.2090	0.5722	0.3101	0.2436	0.5527	0.2577	0.1972
Decision Tree	B	0.8195	0.6164	0.5019	0.8148	0.6151	0.5006	**0.7382**	**0.6454**	**0.5321**
	D	0.8186	0.6149	0.4996	0.8092	0.6136	0.4991	**0.7382**	**0.6454**	**0.5321**
	H	0.7965	0.5236	0.4320	0.7027	0.4156	0.3429	0.5922	0.3544	0.2811
	K	0.8263	0.6022	0.5075	0.7716	0.5555	0.4676	0.7177	0.6060	0.5167
Random Forest	B	0.9019	0.6238	0.5092	0.8352	0.6105	0.4972	**0.7382**	**0.6454**	**0.5321**
	D	0.9146	0.6520	0.5400	0.8340	0.6164	0.5065	**0.7374**	**0.6431**	**0.5359**
	H	0.8856	0.4285	0.3212	0.7717	0.3653	0.2682	0.6016	0.3398	0.2444
	K	0.9616	0.6945	0.5904	0.8018	0.5730	0.4769	0.7093	0.5905	0.4908
SVM	B	0.7705	0.5004	0.3851	0.7641	0.4993	0.3847	**0.6794**	**0.5281**	**0.4116**
	D	0.8766	0.0123	0.0077	0.7760	0.0057	0.0036	0.5058	0.0232	0.0146
	H	0.8095	0.4953	0.3803	0.7990	0.49421	0.3800	**0.6794**	**0.5281**	**0.4116**
	K	0.8766	0.0123	0.0077	0.7760	0.0057	0.0036	0.5058	0.0232	0.0146
Neural Network	B	0.8394	0.6091	0.5117	0.8165	0.5844	0.4890	**0.7299**	**0.6283**	**0.5317**
	D	0.8378	0.6149	0.5177	0.8221	0.5992	0.5018	**0.7329**	**0.6323**	**0.5498**
	H	0.8439	0.6213	0.5214	0.8207	0.5991	0.5007	0.7261	0.6212	0.5297
	K	0.8284	0.5969	0.4935	0.8118	0.5780	0.4762	0.7235	0.6178	0.5026

3. (g) Using the results from 3f, compare and contrast the various models with respect to interpretability, computation time, and predictive performance. If you were the loan officer using this model, which one would you use to guide whether or not to give a loan?

For each model, we have highlighted the two best performances on different datasets. The table below provides a summary of the runtime and performance of each model on the specified datasets.

	Dataset	Runtime (sec)	AUC
Logistic Regression	B	0.0508	0.7369
	D	0.0449	**0.7382**
Decision Tree	B	**0.0109**	**0.7382**
	D	**0.0089**	**0.7382**
Random Forest	B	0.3234	**0.7382**
	D	0.0838	0.7374
SVM	B	0.1355	0.6794
	H	0.1439	0.6794
Neural Network	H	1.5998	0.7299
	K	1.5911	0.7329

From the table above, it is evident that the top performances were achieved by the logistic regression, decision tree, and random forest models, with an AUC of 0.7382. Notably, the decision tree models exhibited the fastest runtimes, averaging around 10 milliseconds. The runtime refers to the time required for training each model and making predictions on the test dataset. Considering interpretability, the decision tree model stands out as it conducts classification based on provided features, closely resembling the decision-making process undertaken by human beings.

Decision Tree > Random Forest > Logistic Regression > SVM > NN (interpretability)

Decision Tree > Random Forest > Logistic Regression > NN > SVM (Performance)

Decision Tree > Logistic Regression > SVM > Random Forest > NN (runtime)

From the desired (left) to the undesired (right), we can compare the various aspects of the models as illustrated above.

Neural Network was the only model that had a robust performance on each of the datasets. It is apparent that across all the aspects, the decision tree outperforms the other models. As a loan officer, I would have opted for the decision tree model due to its superior interpretability, strong performance, and reasonable runtime.