1 1.1 Exploratory data analysis

REMOVE THESE AUTOMATIC INDICES FROM OVERLEAF CAN BE CONFUSING

1.1 Basic facts

We have a dataset that consists of 5471 (n) samples with 4124 (p) columns. The features of our data are all continuous, in a logarithmic scale and they expression levels for genes. It is worth mentioning that the dataset is sparse, that is, a lot of cells have many gene expressions that are 0. While it is hard to have an objective measure of sparsity we can plot average gene expression levels across all our features, see Figure 1 below, most genes are non zero for a limited number of cells.

Figure 1: Possible figure of average gene expression level across all genes. We can do either histogram or histogram by label.

We have 2 distinct classes of cells, the TREG cells and the CD4+T cells. These are going to be our labels that we want to classify. We have some class imbalance, the ratio is 6/10 in favour of the CD4+T which is the dominant class. This is going to be important for the models we try to tune as some models have options to adjust for class imbalance. We will also try the option of tuning the threshold decision for classifying to one cell or another. By default the probabilistic models in Scikit-learn classify to the positive class if the conditional probability for the given model $\mathbb{P}(y|X) > 0.5$[1]. We will tinker with this threshold to try to optimize the F1-Score given our class imbalance.

1.2 Visualization and dimensionality redcution

Both the fact that $n \approx p$ and the fact that the data is sparse point us towards using towards using regularization and feature selection. The instructions for this problem also make us use PCA with 10 components. To confirm whether the number of components is optimal, we can plot the cumulative sum of the explained variance by each of the components. As you can see below in Figure 2, choosing just the first ten components makes us use a very amount of the total variance. In Section 3 we will try to tune the number of components to get an improved F1-Score.

Figure 2: Scree plot: cumulative sum of PCA components.

For completeness, we add Figure 3 where we use t-SNE to reduce the dimensionality of the data from 4124 to 2[2] with the purpose of visualizing the joint distribution (after the t-SNE transformation) of both cells. Figure 3 shows some separability of the two classes. It also shows some cells of a given clas (TREG) in regions where the density is much higher for the other type of cell (CD4T), maybe we can use this as intuition as to why in later sections we find it difficult to improve the F1-Score beyond 0.95.

Figure 3: Joint distribution of transformed data by t-SNE

2 1.2 Train and evaluate the

This section is split into two parts, the first part describes our code from Python at a high-level, in particular some of scikit-learn procedures to streamline the

Mention reproducibility of our approach.

– Explanation of your approaches 15 points!

- Most tuning is done, VSM give very good results.

[1]see scikit references

[2]The underlying algorithm is stochastic and quite sensitive to how we tune the hyperparameter of perplexity

- Tough to do what shak is doing with pca Example from sklearn
- RESULTS FROM GRIDSEARCH ARE NOT GOING TO BE REPLICABLE, RESULTS FROM FINAL ESTIMATOR BASED ON IT SHOULD BE.

2.1 Tuning the models

2.1.1 Explanation of approaches

2.1.2 Analysis of results

m	$\Re\{\underline{\mathfrak{X}}(m)\}$	$-\Im\{\underline{\mathfrak{X}}(m)\}$	$\mathfrak{X}(m)$	$\frac{\mathfrak{X}(m)}{23}$	A_m	$\varphi(m)$ / °	φ_m / °
1	16.128	8.872	16.128	1.402	1.373	-146.6	-137.6
2	3.442	-2.509	3.442	0.299	0.343	133.2	152.4
3	1.826	-0.363	1.826	0.159	0.119	168.5	-161.1
4	0.993	-0.429	0.993	0.086	0.08	25.6	90
5	1.29	0.099	1.29	0.112	0.097	-175.6	-114.7
6	0.483	-0.183	0.483	0.042	0.063	22.3	122.5
7	0.766	-0.475	0.766	0.067	0.039	141.6	-122
8	0.624	0.365	0.624	0.054	0.04	-35.7	90
9	0.641	-0.466	0.641	0.056	0.045	133.3	-106.3
10	0.45	0.421	0.45	0.039	0.034	-69.4	110.9
11	0.598	-0.597	0.598	0.052	0.025	92.3	-109.3

Model	Acc.	Balanced Acc.	AUC	F1 Score	Confusion Matrix
LDA	**0.965**	0.956	0.993	0.951	[[687, 6], [32, 370]]
LOGIT	0.958	0.954	0.993	0.942	[[672, 21], [25, 377]]
QDA	0.367	0.500	0.500	0.537	[[0, 693], [0, 402]]
KNN	0.788	0.777	0.874	0.718	[[567, 126], [106, 296]]
SVM	0.956	0.953	0.993	0.940	[[669, 24], [24, 378]]
RF	0.943	0.924	0.993	0.917	[[690, 3], [59, 343]]
GBDT	0.963	0.955	0.994	0.949	[[684, 9], [31, 371]]
With PCA					
LDA	0.950	0.936	0.994	0.928	[[685, 8], [47, 355]]
LOGIT	0.966	0.965	0.994	0.954	[[672, 21], [16, 386]]
QDA	0.960	0.958	0.993	0.946	[[669, 24], [20, 382]]
KNN	0.928	0.911	0.956	0.896	[[675, 18], [61, 341]]
SVM	0.967	0.966	0.994	0.956	[[672, 21], [15, 387]]
RF	0.957	0.947	0.989	0.940	[[683, 10], [37, 365]]
GBDT	0.947	0.938	0.990	0.926	[[673, 20], [38, 364]]

3 Our 3 models

– Explanation of your approaches 15 points!

- AdaBoost, just because we have seen it in class and it could be interesting.
- Play around with the PCA optimality. Logit with tuned PCA and everything else.
- Threshold optimization.

4