CLASSIFYING CANCER TYPES ON RNA-SEQ GENE EXPRESSION DATA

MACHINE LEARNING

Martin Banchero: 2589681, Davide Bressan: 2652725 , Dennis Dekker: 2656394, Michelle van der Wurff: 2655250

March 2019

ABSTRACT

Nowadays the importance of machine learning in cancer research is increasing. Improved tumor type recognition and classification results in faster diagnosis and treatment application to support the Bio medical fields. Here, different classifiers are trained in the recognition of ten different tumor types using RNA-seq gene expression data. Reduction of the high-dimensionality data is performed by Principal Component Analysis (PCA) and class imbalances by Synthetic Minority Over-sampling Technique (SMOTE), followed by the classification methods: Support Vector Machines (SVM), Random Forest, Naive Bayes and K-Nearest Neighbors (K-NN). After splitting the data available in test and training set, Nested cross-validation has been implemented to tune the hyperparameters and for model selection. Finally, four metrics have been considered to compare the performances on the test set for the different classifiers. A surprising result is that k-NN, the simplest method among the applied, performed better on the unbalanced dataset, whereas Random Forest was best on the dataset obtained after SMOTE over-sampling.

Keywords Machine Learning · RNAseq · Cancer type classification · Gene expression

1 Introduction

For thousands of years cancer is shown to be one of the most devastating diseases across humanity. According to the National Cancer Institute, genomic research became critical to tackle this disease, allowing the development of new technologies and gaining new insights into cancer mechanisms[1]. One of those technologies is RNA sequencing (RNAseq), which can generate expression data for over 20.000 genes per run. In order to analyze such excessive amount of data, a lot of tools are available to determine differential expressed genes (DEGs). While this data can be used to determine crucial genes for each cancer type, this data can also be used for the classification of cancer types. The improvement of computational recognition of tumor types can for example help the biomedical field with early diagnosis. Using a proper classification method of cancer types leads to a more accurate selection of the treatment to be applied, what shows the importance of research in tumor type recognition. The development of experimental and computational methods can be combined to improve the search of the correct treatment. Several studies highlighted the importance of machine learning in this field by applying different methods like neural networks, decision trees, Bayesian networks and support vector machines [2][3]. This study is designed to train the machine in recognition of ten different tumor types using different classification methods based on Machine Learning, which are shown to be able to recognize and predict multiple tumor types. Here, we use the RNA-sequencing expression data of the following tumor types: Breast invasive carcinoma (BRCA), Lung adenocarcinoma (LUAD), Prostate adenocarcinoma (PRAD), Colon adenocarcinoma (COAD), Kidney renal clear cell carcinoma (KIRC), Lung squamous cell carcinoma (LUSC), Ovarian serous cystadenocarcinoma (OV), Rectum adenocarcinoma (READ), Glioblastoma multiforme (GBM), Uterine Corpus Endometrioid Carcinoma (UCEC) and Head and Neck squamous cell carcinoma (HNSC) [4]. For the classification of these cancer types we used support-vector machine (SVM), random forests, naive Bayes and k-nearest neighbors (k-NN). The main question is to determine which method presents the best performance in the classification of the ten different cancer types. As k-NN and naive Bayes are more simple classification methods, SVM and random forest are expected to perform better on this highly dimensional data.

2 Methods

2.1 Data Preprocessing

In the first place, in order to assess the performances of the different machine learning algorithms alongside with the viability of the project, a small data set containing RNA-seq gene expression was used. This data set was downloaded from UCI, the centre for machine learning and intelligent systems [5]. Afterwards, the analysis continue using the data set from the TCGA Pan-cancer Analysis Working Group, an open research project on Synapse [4]. Thousands of samples are present in this repository, each containing expression levels of 20537 mRNA transcripts (RNAseq, Illumina Hiseq 4000). Furthermore, additional information about the patient like age, sex and type of cancer is available. From this big data set, 10 typologies of cancer were selected alongside with patient information. As a result a table was created with a 4 410 x 20 537 dimension. This table contain the samples identifiers on the row and on the columns the gene IDs. Additionally, another table containing the annotation (labels) for each patient was created. The data was transformed using log2 transformation, this is a normalization of the mRNA expression values. The purpose of this was to reduce the outliers and bring the values distribution similar to a normal distribution. In order to test and train the data the cancer type annotation was concatenated to the data set. This leads to 3820 samples, as some samples had replicates in the data.

2.2 Principal Component Analysis

In order to gain insight in the structure of the data, principal component analysis (PCA) was performed. This is a method to reduce high dimensionality, generating principal components in a way that maximizes variance explained by each consecutive principal component. The theory behind this is that variation of the whole data set is explained by the addition of weights to the variables[6]. The variables symbolize the principal components, which are created by linear combinations of the variables to provide the average weights. These weights represent the unit vector weights, creating the opportunity to determine variance by going back and forth in model space. Where the first principal component explains the most variance and the second component the second most variance, etc. Sum-of-squared is applied ($||w|| = 1$) followed by maximization function (eq. (1), retrieved from [6]) to normalize the values and maximize the variance.

$$argmax(t^T t) = argmax(w^T X^T X w) \qquad (1)$$

Where X is the matrix (containing rows and columns data), $X^T X$ the cross-product matrix and t presents the new model data ($t = Xw$ in the equation). The total explained variation is calculated by the regression of the variables together with the new model data using equation eq. (2) retrieved from [6].

$$X = tp^T + E \qquad (2)$$

Where t represents the new model data, p regression coefficients and E the residuals. Eventually a PCA model is created to provide information about the data by the observation of scores, residuals and loading. During this project PCA was performed with 20 principal components using the decomposition function from *scikit-learn* package. Where 5 components where selected for the follow up experiments. As it can be observed in results, PCA shown a lot of outliers in the data (see figure S1). In order to remove these outliers all data point with a value higher than 200 on PC1 was removed. This reduced the data further to a final 3425 samples.

```
Consider a sample $A$ (6,4) and let sample $B$ (4,3) be its
    nearest neighbour.
$A$ is the sample for which k − nearest neighbours are
    identified.
$B$ is one of its k − nearest neighbours
Let:
  $A_x \leftarrow 6, B_x \leftarrow 4$
  $A_y \leftarrow 4, B_y \leftarrow 3$
Then:
  $(B_x - A_x, B_y - A_y) \leftarrow (-2, -1)$
The new samples $A'$ and $B'$ will be generated:
  $(A', B') \leftarrow$ (6,4) + rand(0, 1) $* (-2, -1)$
```

Listing 1: Example of SMOTE algorithm to over sample a class. [7].

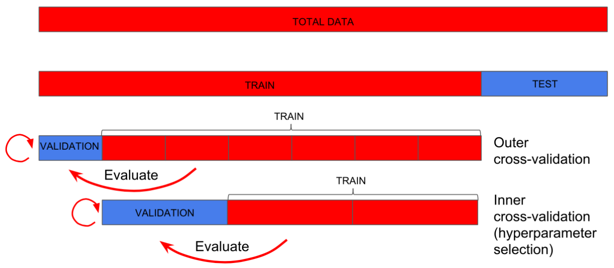


Figure 1: Graphic visualization of nested cross validation. The total data set is split in test and training set. The test set is kept aside and not used for the cross validation. Afterwards, the training set is splitted in 7 folds, and for each iteration (outer loop) one fold is used as validation set. The other 6 folds are again splitted in 3 folds (inner loop), one is used for validation and the other as a training set. The inner loops are necessary to select the model with best performance (best set of hyperparameters). Subsequently, this model is tested on the validation set of the outer loop. This type of cross validation method is good for model selection and at the same time is providing an error estimation.

2.3 Class imbalance

To correct the class imbalance in our data, the Synthetic Minority Over-sampling Technique (SMOTE) was applied[8]. This technique uses a method in under- and over-sampling of the data, where the minority presents the small subset of data and majority the normal subset. The minority class is over-sampled by created synthetic examples using k-nearest neighbors (see 2.5.4) within the class. The difference between the neighbors and the vector presents the synthetic sample by multiplying it with a random number between zero and one. The under-sampling works the other way around by removing random samples for the majority class. The class imbalance was corrected by the implementation of the over_sampling function with package *imblearn*. Eventually visualization of the total amount of samples per class were plotted to check if both minority and majority classes were equally represented . For an example of SMOTE algorithm see listing 1.

```
k ← 7, j ← 3
Randomly split the dataset in k−folds
for each k
  out_val ← k
  out_train ← other (k−1)−folds
  Randomly split out_train in j−folds
  for each j
    in_val ← j
    in_train← other (j−1)−folds
    train a model for each set of hyperparameters on
     in_train
    test each model on in_val.
    Store the accuracy for every set.
  Calculate average accuracy for each set of
   hyperparameters
  Pick the best set, train on out_train and test on out_val
  Save the accuracy for fold k
Calculate the mean and std of accuracy over all the k−folds
```

Listing 2: Nested cross validation pseudocode [7]. For a graphical representation of the procedure see figure 1.

2.4 Cross-validation

To select the best classifier, nested cross-validation was implemented [7]. This generate a distribution of accuracy scores for each method, this allows to select the best one. Furthermore, this type of cross-validation provides insights about which set of hyperparameters are suitable for the classification problem. The implementation of Nested cross-validation was performed using the package *scikit-learn*. In particular, the function *Kfold* that is used to randomly split the training set into 7 folds (see figure 1). Thus, at each iteration the k-fold is used as test set and the rest as a training set. Subsequently, an inner cross-validation loop is performed (using *GridSearchCV*), that split the training set in three folds: two are used to train a model for each combination of input hyperparameters, whereas the last one (validation set) is used to test them. In this way the performance can be evaluated for each combination of hyperparameters and then select the best one. Finally, the inner loop winner model is applied on the test set, as a result the selected accuracy and hyperparameters are stored.

2.5 Classifiers

The performance of the four different classification methods are tested to select which algorithm performs best. To test performance $accuracy$, $F1$ values and $Cohen'sKappa$ (eqs. (3) to (5)) were taken into account:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{3}$$

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{4}$$

$$Cohen's\ Kappa = \frac{P_o - P_e}{1 - P_e} \tag{5}$$

Where (TP) indicates true positives, (TN) true negative, (FP) false positive and (FN) false negative , P_o indicates accuracy, P_e hypothetical probability,

2.5.1 Support Vector Machines

Support vector machine (SVM) uses kernels to train and classify multidimensional data. Multiple margin hyperplanes were selected between the decision boundaries and the training data. Within this margin a small subset of the data was selected as different support vectors ($w^T x + b$) to classify the data as negative(-1) or positive(1) [9]. The size of the margin is calculated by using equation 6:

$$w^T x = ||w||||x||cos\alpha \tag{6}$$

In this equation X and W represent the matrix and the weight of the data respectively. By margin maximization the maximum loss is minimized leading to a separation and classification of the data. The implementation of SVM was performed using the package *sscikit-learn* with the svm function. Different hyperparameters were selected using kernels. The Gaussian kernel γRBF measures distance between two data points, it was calculated by equation eq. (8) .In this equation, γ controls the width of the Gaussian kernel. Gamma values 1e-3, 1e-4, and C (hyperparameter soft margin) values of 1, 10, and 100. The same C values are applied to the linear kernel. The soft margin (C) was calculated by equation 8 to obtain the loss function.

$$k(a,b) = exp(-\gamma||a-b||) \tag{7}$$

$$\frac{1}{2}||w|| + C\sum p_i \tag{8}$$

2.5.2 Random Forest

Random forest is an ensemble of classification trees which uses bootstrap aggregation (bagging). Bagging trains and classifies the models by resampling the dataset. Random forest uses a slightly different approach compared to other classifiers. This algorithm add an extra layer of randomness [10], which result in another split of the selected data on the best (random) predictor . This approach reduces overfitting of the data. This ensemble function was implemented by using the package *scikit-learn*. The following three hyperparameters were selected: Estimators with values: 100, 250, 500, 750 and 1000. Criterion with gini and entropy selection. And the option to add bootstrap.

2.5.3 Naive Bayes

Naive Bayes is an useful classification method when dealing with high-dimensional data. The method treats the features independently given a class and thereby creates independent probabilities to classify [11]. The feature treatments are cleary visible in equations eq. (9) and eq. (10) showing the difference between the theory behind bayes and naive bayes.

$$p(Y|X) = \frac{p(Y|X)p(Y)}{p(X)} \tag{9}$$

$$p(X_1, X_2) = p(X_1|Y)p(X_2|Y) \tag{10}$$

The naive bayes classifier was implemented using the package *scikit-learn* with function *GaussianNB*. The following values are used for hyperparameters selection in smoothing variance: 1e-9, 1e-10 and 1e-7.

2.5.4 K-Nearest Neighbors

The K-nearest neighbors algorithm use a straightforward and simple approach by classifying the nearest neighbours around specific samples [12]. During the selection, the majority class together with the distance weight are taken into account. In the classification different hyperparameters are selected. Also the impact of the class defines how much weight is added. For example, neighbors that are closer to the sample contain more weight, thereby have more impact. Equation eq. (11) shows an example from the Vote method by Cunningham and delany ([12]) which uses the inversion of the distance to determine the weights.

$$Vote(y_j) = \sum \frac{1}{d(q,x)^n} \cdot 1(y_j, y_c) \tag{11}$$

Where q indicates the sample, y_j, y_c the different classes and x neighbors. The k-nearest neighbors classification was implemented using the package *scikit-learn* with function *neighbors*. The following hyperparameters were selected: 7, 9, 15, 17 and 25 as neighbors and uniform or distance weights. The number of neighbors is always an odd number, this is to avoid situations of draws during the assignment of the class to the evaluated sample.

2.6 Experimental approach

To summarize, the data containing a number of instances and their features was splitted in training, validation an test set. Different classifiers were selected to train and validate the data for the recognition of different tumor types. Nested cross-validation was applied in order to compare the different models and estimate their performances. Finally, the models were trained on the train and validation set alongside with the best selected hyperparameters, and least tested on the test set.

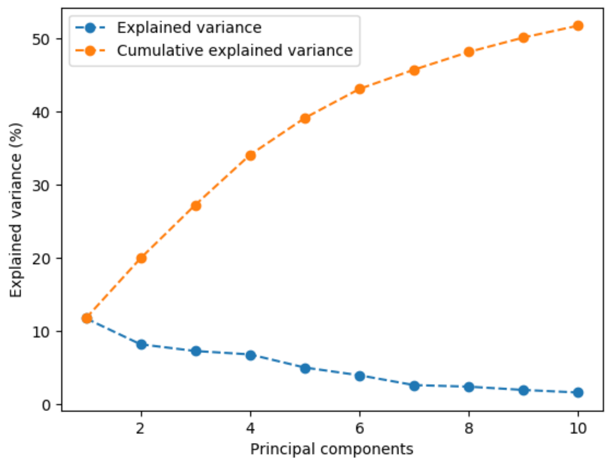


Figure 2: Explained and cumulative explained variance for first 10 principal components

3 Results

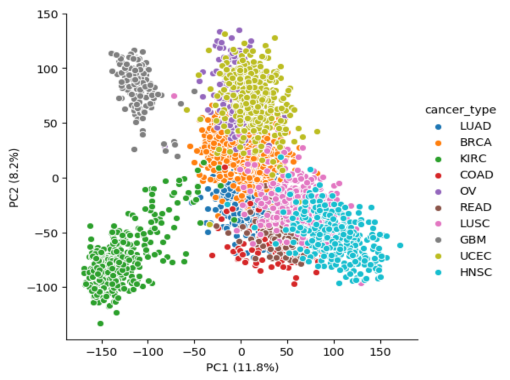
3.1 Data Preprocessing

In our case the PC1 and PC2 were plotted (see figure 3a) to see if any groups can be observed. Following with the analysis the loadings were plotted (see figure 3b), and the top 20 genes which contributed to the most variation in PC1 and PC2 was obtained for further analysis.

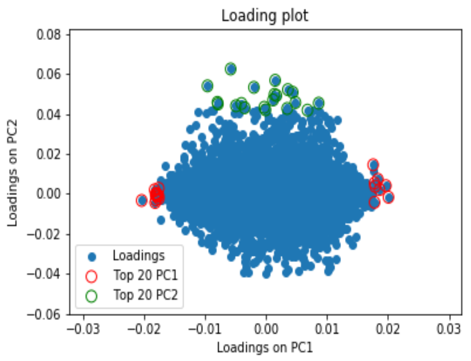
The plot of the PCA shown in figure 3a and each point color by class. The variance explained by PC1 is 11.8% and by PC2 8.2%. Furthermore, the plot of Explained and cumulative variance for number of components can be seen in figure 2. The PCA shows certain level of clustering, mainly for KIRC (red) with the rest also COAD (purple) and LUAD (blue). While the BRCA (yellow) and the PRAD (green) cluster together, it would still be possible to separate them using more principle components. The loadings for PC1 and PC2 were calculated (see figure 3a) and the top 20 genes for each principal component were extracted, in red for those for PC1 and in green for PC2.

3.2 Class imbalance

As mentioned in methods there is some class imbalance present in the data, the majority class is represented by BRCA samples (see figure 4). In particular, BRCA has 815 samples, while READ (the small-



(a) Plot of PC1 vs PC2. Each cancertype is colored differently. PC1 explains 17% of the variantion in the data, PC2 10.9%.



(b) Plot of loadings for PC1 and PC2. In red the top 20 genes contributing to PC1. In green the top 20 genes for PC2.

Figure 3: Principal component analysis. Score plot on the left and loadings plot on the right.

est class) 88. SMOTE has been applied to tackle this problem, over-sampling the minority classes and bring their total amount close to the major class (BRCA). This produce more crowded clusters, as can be seen in figure 5.

3.3 Nested cross validation

Nested cross validation has been applied with four different algorithms, as explained in methods 2. For each of the seven folds an accuracy is stored, an the mean and standard deviation has been calculated, in order to evaluate the expected performances of the different methods. The results are displayed in table 2 for the normal dataset and in table 3 for the dataset obtained after the over-sampling with SMOTE.

After nested cross validation, the sets of hyper-parameters that give the best performance for each algorithms has been selected. On the dataset without SMOTE correction, the results (in order of mean accuracy) are the following:

1. SVM: C=10, gamma=0.0001, kernel rbf.
2. K-NN: number of neighbors=25, weights=distance.
3. Random Forest: bootstrap =True, criterion=gini, number of estimators=250.
4. Naive Bayes:variance smoothing: 1e-09

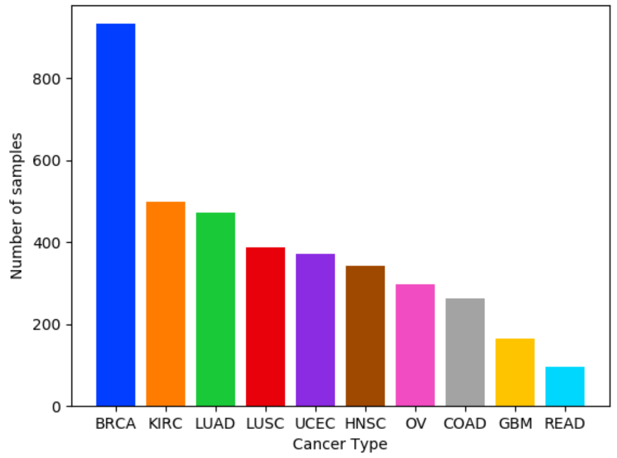
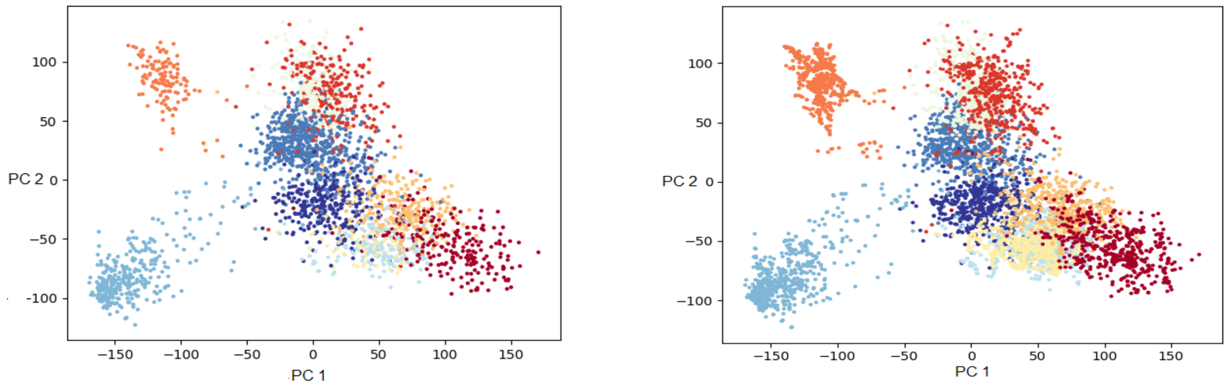


Figure 4: Class imbalance in the data.

Table 1: Number of samples for each tumor type after removing the outliers

Cancer type	Number of samples
BRCA	815
KIRC	426
LUAD	413
LUSC	334
COAD	235
OV	292
READ	88
GBM	158
UCEC	361
HNSC	303



(a) No SMOTE correction

(b) PCA plot. SMOTE corrected.

Figure 5: PCA plots with and without class imbalance correction.

On the dataset with SMOTE correction, the same set of selected hyperparameters occurs as best for each of the seven folds in nested cross validation:

1. SVM: C=100, gamma=0.001, kernel rbf.
2. K-NN: number of neighbors=7, weights=distance.
3. Random Forest: bootstrap =False, criterion=entropy, number of estimators=750.
4. Naive Bayes:variance smoothing: 1e-09

Table 2: Nested cross validation results for the four algorithm tested on the original dataset. The mean and standard deviation of the accuracy scores are visualized.

Method	Mean	STD
SVM	0.911	0.01
k-NN	0.901	0.01
Random forest	0.900	0.02
Naive Bayes	0.898	0.02

Table 3: Nested cross validation results for the four algorithm tested on the dataset after applying SMOTE oversampling method. The mean and standard deviation of the accuracy scores are visualized.

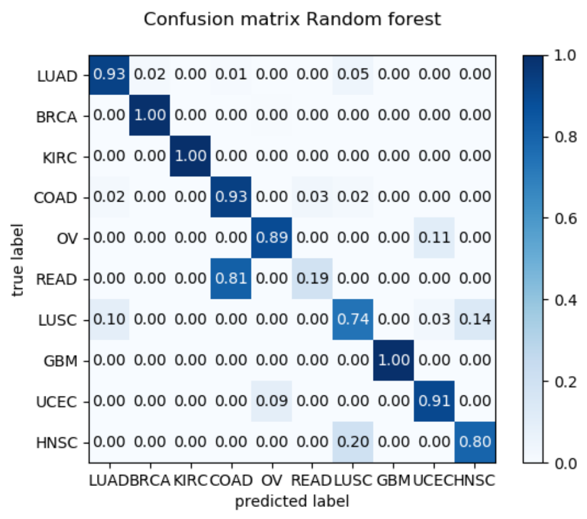
Method	Mean	STD
SVM	0.964	0.005
k-NN	0.964	0.005
Random forest	0.958	0.006
Naive Bayes	0.8434	0.01

3.4 Final testing

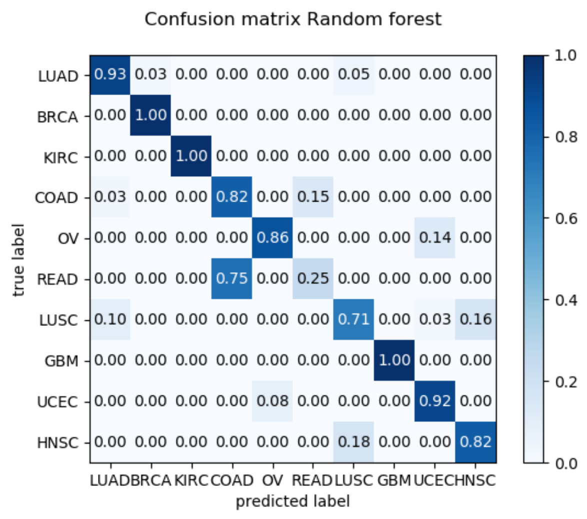
Finally, after tuning the hyperparameters and evaluating the performance of the different algorithms, a model has been trained on the whole train and validation set together, this for each of the methods mentioned before (using both datasets, with and without SMOTE oversampling). The test was performed using the data left apart at the beginning of the project (857 samples). The results are visualized in table 4 and 5. Furthermore, the confusion matrices are presented in figure S3 for SVM classifier, in figure S4 for K-NN, in figure 6 for random forest and in figure S2 for the Naive Bayesian classifier. The baseline accuracy obtained with a dummy classifier (sklearn package) is 0.145, this for the dataset without SMOTE oversampling, whereas including SMOTE is 0.098 . The strategy selected is "stratified", that respects the proportion of the different classes in the training set.

Table 4: Accuracies, Cohen's Kappa and F1 score of each model trained on unbalanced data.

Method	Accuracy	Cohen's kappa	F1 score
SVM	0.911	0.896	0.903
k-aNN	0.917	0.903	0.908
Random forests	0.910	0.895	0.906
Naive Bayes	0.903	0.887	0.895



(a) Model trained on the dataset without SMOTE oversampling.



(b) Model trained on the dataset with SMOTE oversampling.

Figure 6: Random forest confusion matrix

Table 5: Accuracies, Cohen's Kappa and F1 score of each method trained on SMOTE corrected data.

Method	Accuracy	Cohen's kappa	F1 score
SVM	0.885	0.867	0.889
k-NN	0.890	0.872	0.893
Random forests	0.901	0.886	0.900
Naive Bayes	0.869	0.848	0.871

4 Discussion

The data in the PCA plot (see figure 3) already showed a promising clustering of the cancer types. Even though some overlapping is present between classes, selecting 5 principal components should yield highly accurate results, as this explains $\approx 40\%$ of the variance (see figure 2. However, before using the different methods some issues with the data, such as class imbalance and determining a acceptable cross-validation, should be addressed.

In order to amend the class imbalance, SMOTE correction was used. The biggest part of the class imbalance was a result of the large amount of samples for the BRCA data set. Using SMOTE to produce more samples of the other cancer types could lead to a better classification of underrepresented data, however there are also some downsides of using this method. For example, the method of oversampling data is based on k-NN (see section 2.3), what is also one of the methods used for classification.

To compare the performance of the different classifiers and at the same time tune the hyperparameters, nested cross validation has been applied. The results can be seen in table 2 for the normal dataset and in table 3 for the dataset with SMOTE oversampling. The mean accuracy is smaller in the former table, while the standard deviation is higher when compared with the latter. This can be a signal of over-fitting of the models trained on the dataset with SMOTE oversampling. Furthermore, SVM performed slightly better than k-NN on the original dataset, while they have the same mean accuracy on the SMOTE dataset. As mentioned before, the oversampling is based on k-NN, bringing to more data points within original instances on the features space. Thus, k-NN has an advantage due to the clustering of data points after SMOTE, what is in line with the expectations that k-NN should perform worse as it is one of the simplest classification method.

Afterwards, four models have been trained on the training set (with and without class imbalance) and tested on the test set left apart. The results are shown in table 4 and 5. k-NN (F1 score of 0.908) displayed the best performance for the unbalanced and

Random forests for the balanced dataset (F1 score of 0.9). These results were not expected, as SVM was thought to be the best classifier. For Naive Bayes the performance is lower than the other classifiers.

The resulting confusion matrices can be seen in figure 6 and in the supplemental (figure S2, S3, S4). The tumor class READ has the least samples (see figure 4), and all the four classifiers have troubles to classify it correctly. In particular, both k-NN and SVM are completely miss-classifying READ as COAD, whereas Random forests miss-classifies around $\approx$ 80% of the samples. This result is unexpected, because SVM and k-NN were assumed to be the best methods. Rectum and colon are part of the same organ, thus the two tumor types shown similar gene expressions and therefore distinguishing them is an hard task for the classifiers. Another cancer type that is not being properly classified is HNSC, because all the methods are miss-classifying it as LUSC ($\approx$ 20%). In this case both the tumor types derived from squamous cells, explaining the ambiguity in their classification.

Furthermore, comparing the confusion matrices in figure 6, the model trained on the bigger dataset (i.e. with SMOTE over-sampling) performs better if compared with the model trained on the normal dataset, as can also be seen in table 5. When the dataset does not contain class imbalances, COAD is slightly predicted better, as well as HNSC.

5 Conclusion

Although PCA shows promising clustering of the cancer types, some methods had to be applied to deal with outliers and class imbalance in order to get more accurate results.
As a result was shown that all the classifiers present good accuracy. Nevertheless Naive Bayes classifier showed lower performance in comparison to the rest, the accuracy is still above expectations. One possible explanation for the good performances of the classifiers is the differential expression of genes for the different types of cancer. This difference might led to a more easy task for the classifiers. However, some types of cancer it seems hard to separate during the classification, for example all the classifiers shown problems to predict READ correctly, due to its similarity with COAD. In the other hand READ is the smallest class, making difficult their classification even corrected by class imbalance.
As a further research it would be good to try a different approach using neural networks or deep learning. In order to conclude this project it is clear the importance of computational methods, in this case machine learning, to develop a better understanding of the disease and help to improve treatments related to it.

References

[1] National Cancer Institute. *Milestones in Cancer Research and Discovery - National Cancer Institute*. 2015. URL: https://www.cancer.gov/research/progress/250-years-milestones (visited on 03/18/2019).

[2] Joseph A. Cruz and David S. Wishart. *Applications of machine learning in cancer prediction and prognosis*. Feb. 2006. DOI: 10.1177/117693510600200030. URL: http://www.ncbi.nlm.nih.gov/pubmed/19458758%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC2675494.

[3] Themis P. Exarchos et al. "Machine learning applications in cancer prognosis and prediction". In: *Computational and Structural Biotechnology Journal* 13 (2014), pp. 8–17. ISSN: 20010370. DOI: 10.1016/j.csbj.2014.11.005. URL: http://www.ncbi.nlm.nih.gov/pubmed/25750696%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4348437%20https://linkinghub.elsevier.com/retrieve/pii/S2001037014000464.

[4] Cancer Genome Atlas Research Network et al. "The Cancer Genome Atlas Pan-Cancer analysis project." In: *Nature genetics* 45.10 (Oct. 2013), pp. 1113–20. ISSN: 1546-1718. DOI: 10.1038/ng.2764. URL: http://www.ncbi.nlm.nih.gov/pubmed/24071849%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3919969%20http://www.nature.com/articles/ng.2764.

[5] C. Dua, D. and Graff. *UCI Machine Learning Repository*. 2019. URL: http://archive.ics.uci.edu/ml (visited on 03/21/2019).

[6] Rasmus Bro and Age K. Smilde. "Principal component analysis". In: *Anal. Methods* 6.9 (Apr. 2014), pp. 2812–2831. ISSN: 1759-9660. DOI: 10.1039/C3AY41907J. arXiv: arXiv:1011.1669v3. URL: http://xlink.rsc.org/?DOI=C3AY41907J.

[7] Jacques Wainer and Gavin Cawley. "Nested cross-validation when selecting classifiers is overzealous for most practical applications". In: (2018). arXiv: 1809.09446. URL: https://arxiv.org/pdf/1809.09446.pdf%20http://arxiv.org/abs/1809.09446.

[8] Nitesh V. Chawla et al. "SMOTE: Synthetic minority over-sampling technique". In: *Journal of Artificial Intelligence Research* 16 (June 2002), pp. 321–357. DOI: 10.1613/jair.953. URL: https://jair.org/index.php/jair/article/view/10302.

[9] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. "A training algorithm for optimal margin classifiers". In: *Proceedings of the fifth annual workshop on Computational learning theory - COLT '92*. New York, New York, USA: ACM Press, 2004, pp. 144–152. ISBN: 089791497X. DOI: 10.1145/130385.130401. arXiv: arXiv:1011.1669v3. URL: http://portal.acm.org/citation.cfm?doid=130385.130401.

[10] Andy Liaw and Matthew Wiener. "Classification and Regression by randomForest". In: *R News* 2 (2002), pp. 18–22. URL: http://cran.r-project.org/doc/Rnews/.

[11] I Rish. *An empirical study of the naive Bayes classifier*. Vol. 3. 2001, pp. 41–46.

[12] Pádraig Cunningham and Sarah Jane Delany. *k-Nearest Neighbour Classifiers*. Tech. rep. 2007, pp. 1–17. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.100.1131%7B%5C&%7Drep=rep1%7B%5C&%7Dtype=pdf.

6 Supplemental

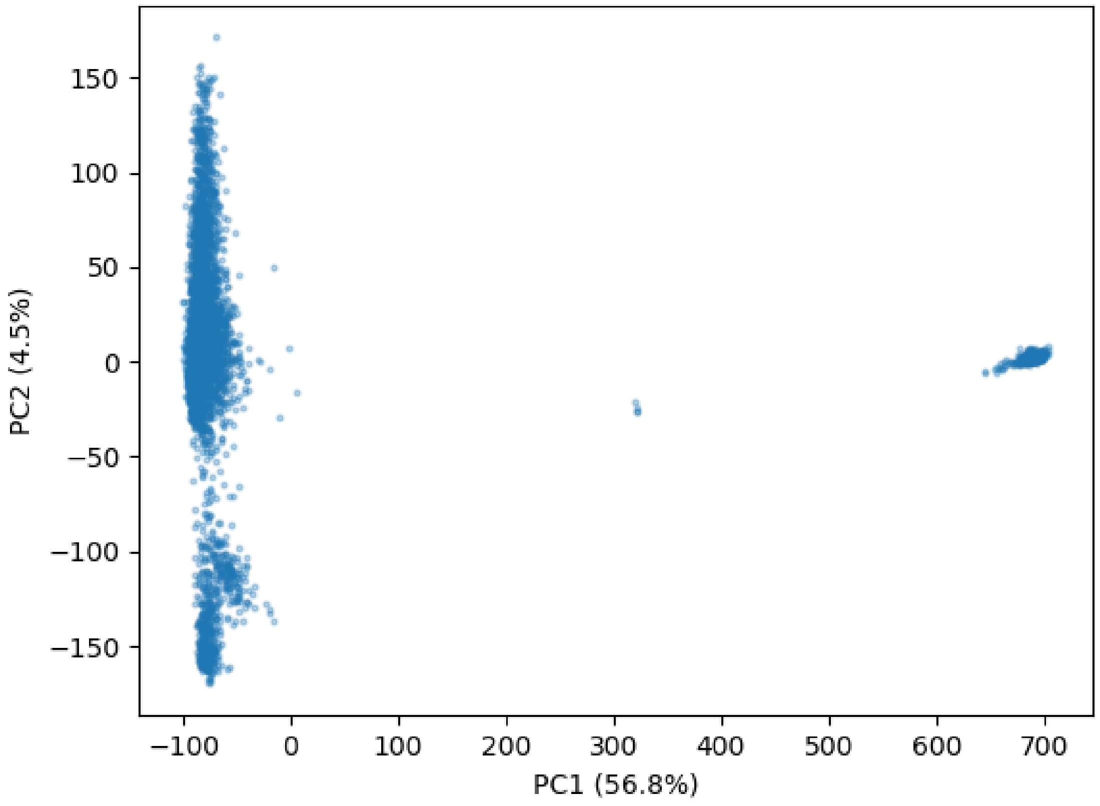
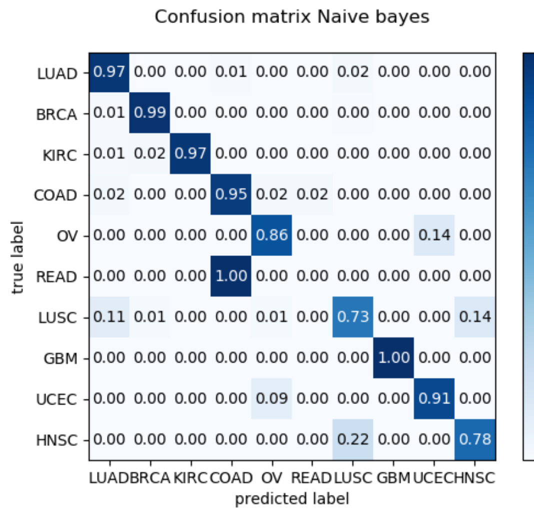
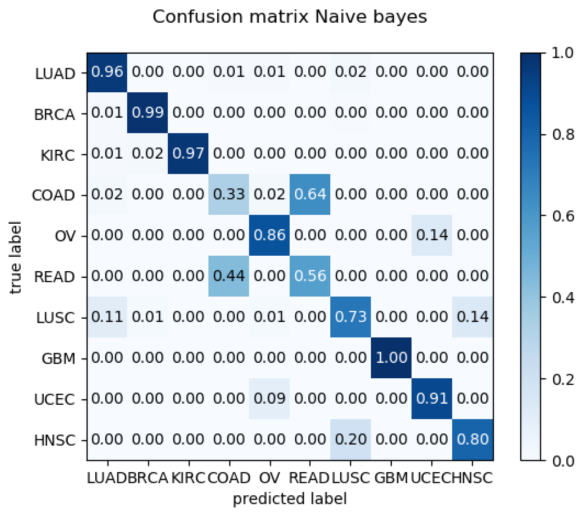


Figure S1: Outliers fist PCA. Removed ouliers based on PC1 (PC1 > 200).

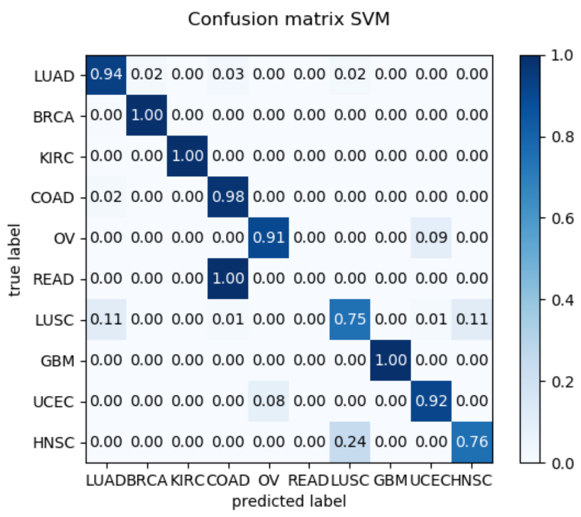


(a) Model trained on the dataset without SMOTE oversampling.

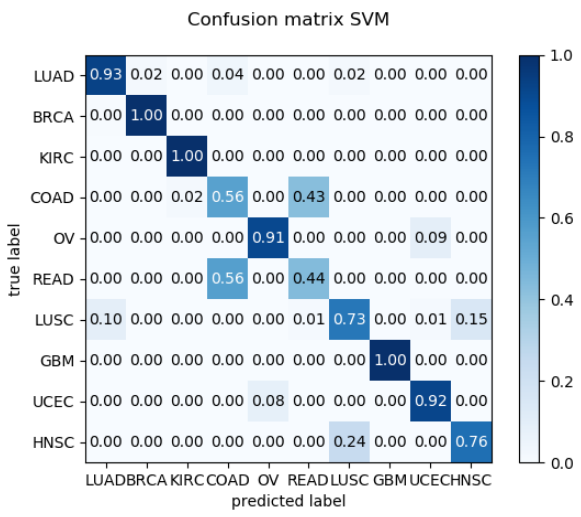


(b) Model trained on the dataset with SMOTE oversampling.

Figure S2: Naive Bayes confusion matrix

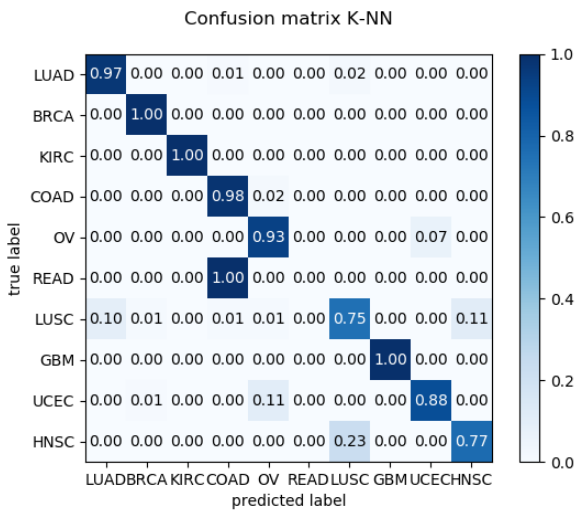


(a) Model trained on the dataset without SMOTE oversampling.

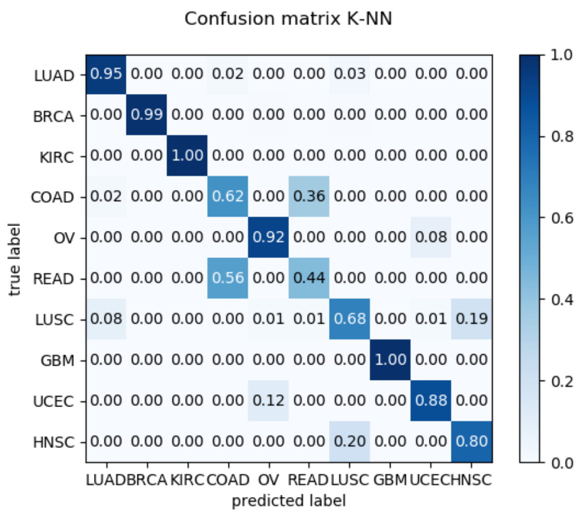


(b) Model trained on the dataset with SMOTE oversampling.

Figure S3: SVM confusion matrix



(a) Model trained on the dataset without SMOTE oversampling.



(b) Model trained on the dataset with SMOTE oversampling.

Figure S4: KNN confusion matrix

7 Contribution

The full project, including data, plot and scripts, is publicly available on GitHub: `https://github.com/Dennis-Dekker/Machine_learning`

Special thanks to us, a lovely group of human beings: Davide Bressan, Michelle van der Wurff, Martin Banchero, Dennis Dekker.