Zoidberg

A Major Project Report
Submitted in partial fulfillment of the requirements for the degree of
Master of Computer Engineering
in
Artificial Intelligence

by

Gustave JULIEN
Thibault DELERUE
Simon CASTERAN
(Group ID: T-DEV-810-group-37)

Department of Artificial Intelligence
Epitech

1 May 2022

Abstract

This project was done to help doctors to choose the best machine learning algorithm to create an image classifier with the ability to take chest x-rays as inputs and accurately diagnose whether the lungs are infected with pneumonia or not.
Five classifiers were chosen, based on the most commonly used and accepted in the computer vision industry: Nearest Neighbour Classifiers (NNC), Support Vector Machine (SVM), Naive Bayes, Convolutional Neural Networks (CNN) and Transfer Learning (TL).
The first part was to determine which metrics to use to compare the selected machine-learning algorithms. The second part exposes a research on the best parameters to use for each. The third and last part consists in a comparison of the resulting models.

Keywords: Pneumonia, Machine Learning, Artificial Intelligence, Nearest Neighbour Classifier, Support Vector Machine, Naive Bayes, Convolution Neural Networks, Transfer Learning.

Table of Contents

List of Figures

Chapter 1

Introduction

With the development of artificial intelligence, computer vision has advanced to the point where its accuracy, in some situations, can be above that of human vision while taking a much shorter analysis time. This technology opens the door to a wide range of applications, one of which being the analysis of medical images.
At the end of 2019, a new virus, COVID-19, swept the globe infecting billions and killing millions. Its spread rate was so high that tracking and understanding it was almost an insurmountable task. Since the average year has millions of people succumbing to respiratory illnesses, understanding whether one is connected to the new virus is a difficult task. This study was conducted to create an algorithm with the ability to take any number of chest X-rays and determine whether or not each one is infected with pneumonia and of which kind. This constitutes a step towards reaching the goal of tracking the infection rate of patients using only image inputs.

1.1 Background

According to the World Health Organisation (WHO) the pneumonia is a form of acute respiratory infection that affects the lungs. The lungs are made up of small sacs called alveoli, which fill with air when a healthy person breathes. When an individual has pneumonia, the alveoli are filled with pus and fluid, which makes breathing painful and limits oxygen intake.
Pneumonia is the single largest infectious cause of child death worldwide. Pneumonia killed 740,180 children under the age of 5 in 2019, accounting for 14% of all the deaths of children under five years old but 22% of all the deaths of children aged 1 to 5. Pneumonia affects children and families everywhere, but death tolls are highest in South Asia and

sub-Saharan Africa. Children can be protected from pneumonia as it can be prevented with simple interventions and treated with low-cost, low-tech medication and care.

1.2 Pneumonia Detection

The detection of the pneumonia is done via chest x-ray analysis. In a case of pneumonia, the area around the heart in the thoracic cage appears with a white blur depending on the state of the infection, whereas in a normal case only the heart and the beginning of the bronchial tube appear in white.

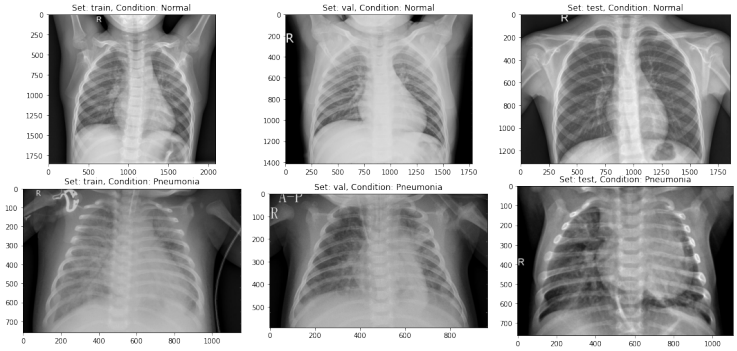


Figure 1.1: Chest X-Ray of people in both normal condition and having pneumonia.

The differentiation between viral and bacterial cases of pneumonia can also be done with the visual analysis of the chest x-ray, although it is much more subtle.

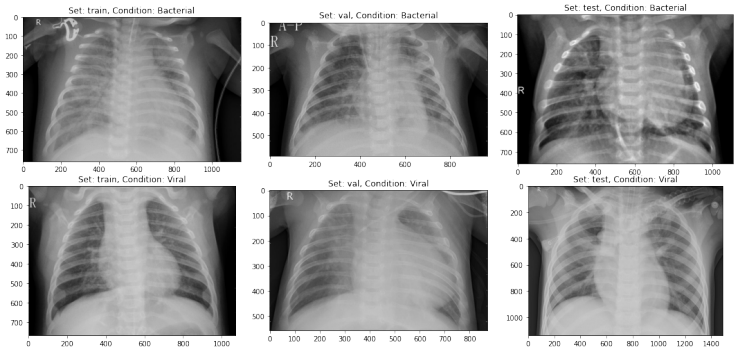


Figure 1.2: Chest X-Ray of people with viral pneumonia and bacterial pneumonia.

1.3 Materials

1.3.1 Datasets

The primary dataset used was provided by the school and consists of a total of 5865 grayscale X-ray images of the chest, 5216 of which are used for training, 624 for testing and 8 for validation. Each of the three sets are further separated into healthy and pneumonic infected lungs. The x-ray images of the pneumonic infected lungs are labelled whether the pneumonia is a viral or a bacterial type.

A second smaller dataset was created from the first dataset with the purpose of testing the performances of the models with a small dataset. It consist of a total of 1440 grayscale X-ray images, 800 of which are used for training, 624 for testing and 16 for validation.

One of the problems encountered was that the number of X-ray images in the primary dataset is imbalanced: we have 1341 images of normal X-ray images and 3875 images of pneumonia X-ray images. We came up with two solutions, according to the algorithm used: the first solution was to artificially augment the number of normal X-ray images by rotating and zooming on the photo in order to have the same number in normal and pneumonia X-ray images; the second solution was to introduce a different weight to the normal and pneumonia images.

All images were provided from the registered doctors included in the project.

1.3.2 Tools

Since this project is founded in artificial intelligence with the goal of being completed and shared as quickly as possible, a Jupyter Notebook was created. This option not only facilitates collaborative work, but it takes advantage of the most commonly used language in data science: Python.
Alongside python, the Numpy, Pandas, sklearn and TensorFlow libraries were incorporated to build on top of pre-built machine learning algorithms.

Chapter 2

Choice of the Metrics

2.1 Confusion matrix

Every model give us a confusion matrix giving the number of True positive (detected pneumonia), False positive (healthy patient flagged as ill), True negative (detected healthy patient, False negative (patient with pneumonia flagged as healthy). Since we are working on a binary classification, our confusion matrix is a 2 by 2 matrix.

2.2 Accuracy

From this confusion matrix, we can calculate the accuracy of the model, the fraction of predictions a model get right:

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

2.3 False negative ratio

Since we are working on pneumonia detection, a false negative can lead to late diagnostic of the pneumonia and possibly to the death of the patient. Therefore accuracy is not enough to accurately rate a model and we need to take better metrics to focus on the false negatives. Accuracy still needs to be high, otherwise it would be useless if it were to flag too many healthy patients.

We can compare the False Negative Ratio of our patients, which is the ratio of ill patients accurately diagnosed as having pneumonia:

$$FNR = \frac{FN}{TP + FN}$$

2.4 ROC curve (receiver operating characteristic curve)

The ROC curve shows the model performance at all classification thresholds (true/false positive/negative).

The AUC (Area Under Curve) is the area under the curve and represents the successfull classification by the model.

The AUC is a number between 0 and 1. An AUC of 1 would have perfectly predicted every positive and negative case. an AUC of 0 would be inverting the two but still be able to differentiate them. An AUC of 0.5 would be unable to differentiate between positive and negative class.[1]

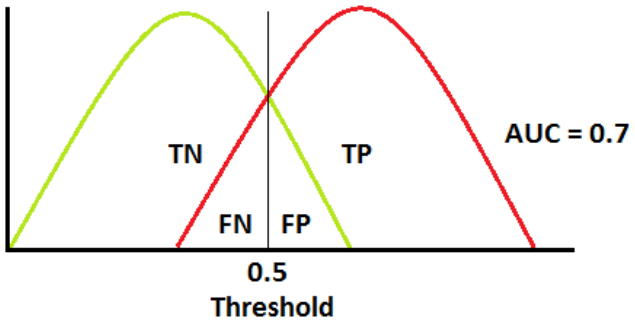


Figure 2.1: AUC figure from towardsdatascience.com

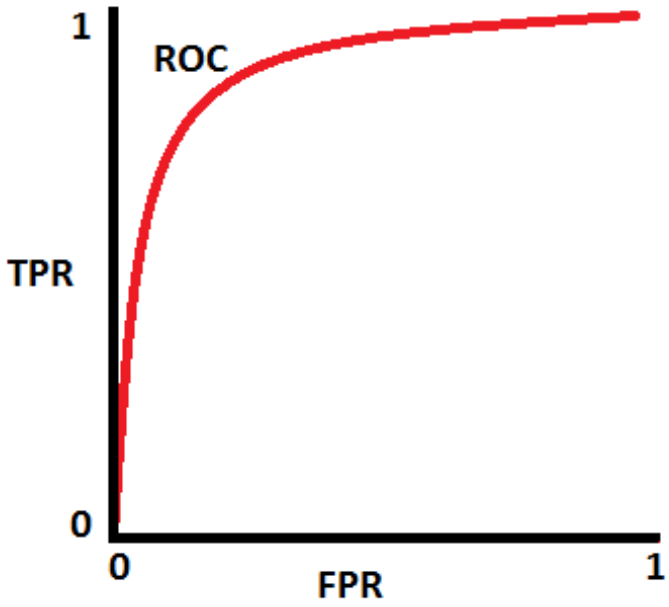


Figure 2.2: ROC curve figure from towardsdatascience.com

Chapter 3

Analysis of the Models

3.1 Nearest Neighbours Classifiers

3.1.1 Algorithm explanation

Nearest Neighbors classifier is a type of simple machine learning algorithm. Its training merely consists in learning all the data set by heart, as it stores the label and all the features of each sample in memory. The testing phase gives it its namesake. It works by mapping the test sample to a position in its training data set and then base final prediction on the closest 'k' number of neighbors. The distance between two samples is computed across all features depends on the metric used.

The K Nearest Neighbors algorithm takes the following main parameters:
- K, which is the number of neighbors to the test sample to take into consideration. This parameter is an integer that ranges from 1 to the total number of sample present in the training data set.
- An algorithm for the model to store and browse the data set. Examples of this include merely brute-forcing all the data set to find the K closest neighbors (brute force), and storing the data set into a tree-based data structure (K-D tree, Ball tree) in order to make the browsing more efficient for large data set with a large number of dimensions.
- A distance metric to consider for measuring the distance between two data sets. It usually is an euclidean distance, with other options including Manhattan and Minkowski distances.

For our research on the model, we ran the scikit learn implementation of the KNN model, using the parameters on auto, with k ranging from 1 to 100.

The model implementation is designed be used in three calls.

The first call serves to instantiate the model and takes the number of nearest neighbors that we want it to base its future predictions on (the value 'k'). We could pass the other aforementioned parameters (the algorithm and the metric) optionally, however if we choose not to do so, the library will decide the best algorithm later during the training period, and the 'minkowski' distance will be used.

The second call trains the model using the data set (images and labels). In that data set, each image must be passed as an array of features. Therefore, a 200*200 pixel image becomes an array of 40 000 features.

The last call to library is to tell the model to make predictions on the testing data set. For each sample in that data set, it will return an array of probabilities for each label, based on a majority vote of the labels of each of the 'k' nearest neighbors to that test sample. In more advanced versions of the KNN model, these probabilities are given more or less weight based on the distance of each of the selected neighbors.

3.1.2 Algorithm experimentation for k values from 1 to 100

At first, we ran the model using the default parameters of the SciKit learn implementation, save for 'n_jobs' parameter. This parameter specifies the number of cores the library might use to execute the model. We set it to 2, which corresponds to the amount of cores on the machine we used to run it on (the default value is 1).

Using these default parameters with k from 1 to 100 yielded the following accuracy curve on the data-set.

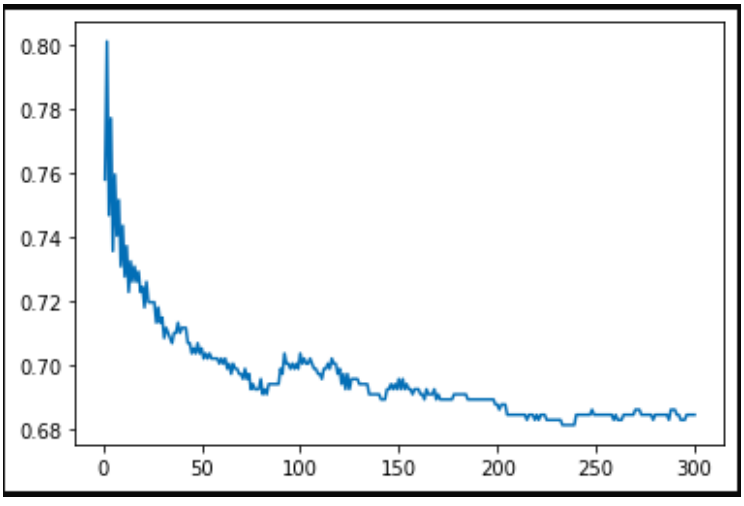


Figure 3.1: Accuracy for k from 1 to 100

Figure 3.1 shows that accuracy only drastically decreases early on from k = 1 to k = 50 dropping from a 80% accuracy to a 70% accuracy. Beyond that, accuracy stabilizes

between 70% and 68%. Given that this measurement shows that the best values of k are found for k between 1 and 10, we decided to limit our runs of k on that interval in order to optimize our time. In the figures below that show k ranging from 1 to 100, we actually ran our measurement by increasing k by 1 on the [1, 10] interval and by increasing k by 10 on the [10, 100] interval. This allows us to get an idea of the resulting trends on the [1, 100] interval by running the model 19 times instead of 100 times.

We measured the time it took to train and test the model for each value of k on our interval on our machine

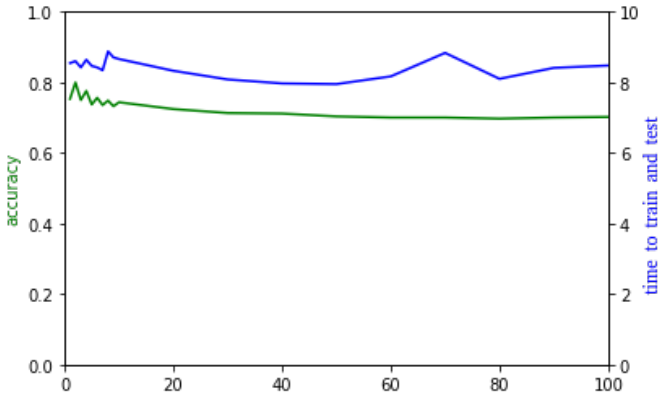


Figure 3.2: Accuracy and Total Training and Testing Time for k from 1 to 100

This shows us that regardless of the value of k, it takes about the same time to fit the model to the training data set and make prediction on the test data set. the various hiccups on the time curve can easily be attributed to other tasks running on the computer at that time.

Given that accuracy does not tell us the whole story, we will delve into the false negatives and false positives to ascertain the value of k that best fits our needs. The following figures show the false negatives and the false positive on the [1, 100] interval for k.

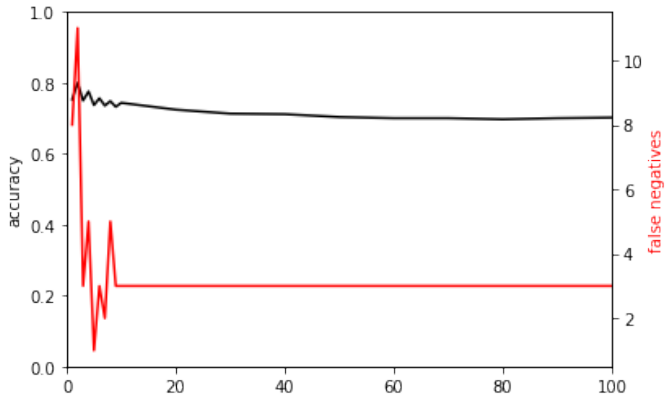


Figure 3.3: Evolution of the Number of False Negatives on the 1 to 100 k interval

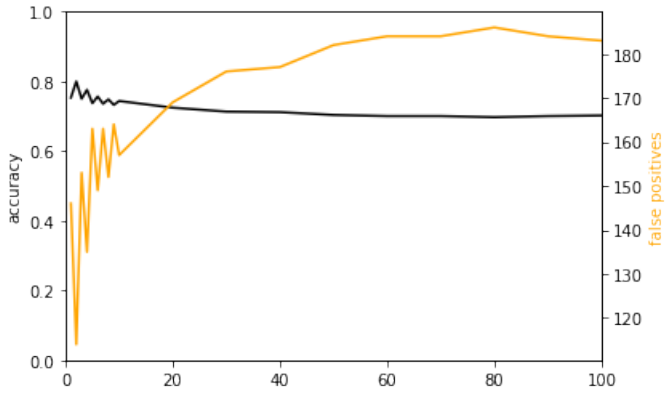


Figure 3.4: Evolution of the Number of False Positives on the 1 to 100 k interval

The confusion matrices for the interval [1, 10] go as follow:

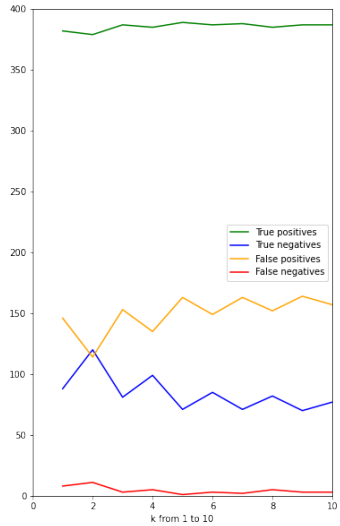


Figure 3.5: Confusion Matrices for k from 1 to 10

This shows our that our lowest number of false positives is reached at 114 at k = 2, with an accuracy close to 80%. Our lowest number number of false negatives is reached at only 1, at k = 5 with an accuracy close to 73%.

3.1.3 Conclusion

KNN is a surprisingly good first approach for classifying pictures given the results. The number of false positives remains quite high for the lowest value of false negatives at k = 5. This might disqualify it in a real medical environment, regardless of the number of false negatives. While time performance is very good, the same cannot be said for the memory usage of the model. Unlike a generalizing algorithm, a KNN model will base its decisions on a copy of a data-set instead of finding and making an intrinsic formula evolve. Therefore, all the data-set must be present in memory at anytime for the model to work. The cost in memory is very steep when speaking about 5216 train images.

3.2 Naive Bayes

Naive Bayes gives weight to parameters depending on its training but does not take into consideration the relations between those parameters. That's why it's called naive. It causes a high bias since int ignores a potential important element.[2]

As seen on our figures, the number of false negatives stays the same after k = 10 while the number of false positives only increases after k = 10. This might be a quirk of our data set,

3.2.1 Algorithms comparison

There are different Naive Bayes, we are going to compare Gaussian Naive Bayes, Multinominal Naive Bayes, Complement Naive Bayes, Bernoulli Naive bayes.
The gaussian model assumes the distribution of features to be gaussian in a binomial distribution, which is fitting since we are working on a 0-1 classification.
Multinomial and complement naive bayes are made for multinomial distribution and are therefore unlikely to outperform the gaussian model for our binomial dataset of sane/pneumonia.
A bernoulli distribution has multiple features of a binary value. We have only one feature (the image squashed in a single array) with a binary value.

All algorithms have roughly 70% accuracy, except the Bernoulli at around 60%.

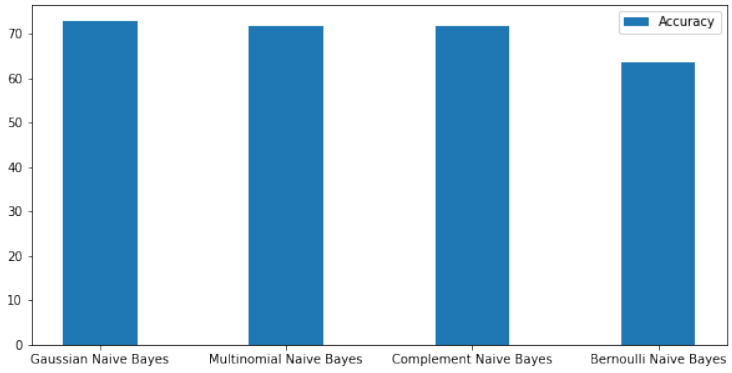


Figure 3.6: Accuracy comparison of different Naive Bayes algorithms

We chose to favor the lowest false negative ratio since a false negative is an undiagnosed pneumonia that can lead to complications and death and kept the Gaussian Naive Bayes algorithm.

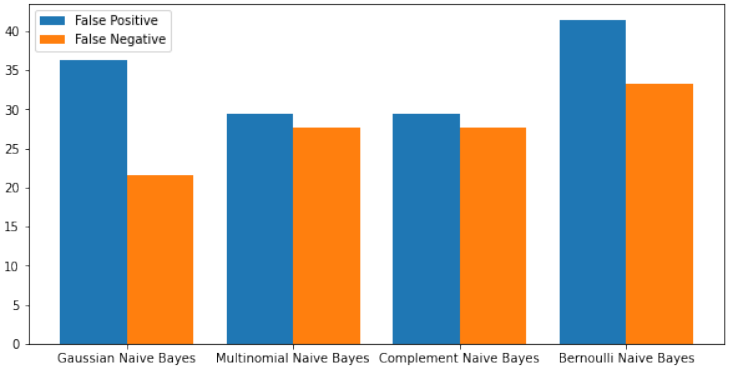


Figure 3.7: False positive and negative ratio comparison of different Naive Bayes algorithms

3.2.2 Image size

The Naive Bayes is a model that has trouble working with data of more dimension than it has training samples. That is why we see almost no difference when working with image size above a few thousands pixels (the size of the training data set being 5k).

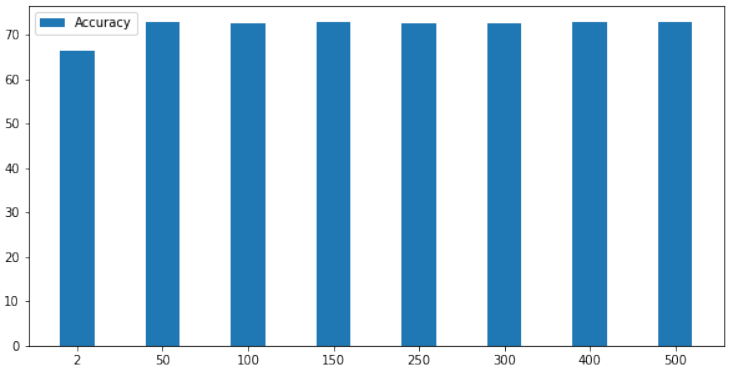


Figure 3.8: Accuracy comparison of image size for the GNB model

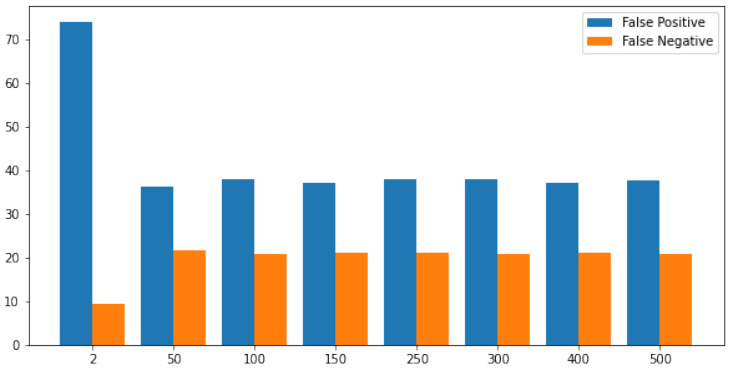


Figure 3.9: False positive and negative ratio comparison of image size for the GNB model

3.2.3 Conclusion

Naives Bayes are used mostly for document classification and spam filtering where they only need a small training data set to estimate the necessary parameters. Due to the decoupling of parameters to estimate them separately, those models run quite faster than other more complex models.

On image classification, the number of parameters is higher, making the model require a bigger training data set. There are attempts to improve it, for example by using intermediate properties [3] or finding salient regions and extracting key points [4] to obtain a smaller vector than the entire image, but even that is not enough to rival the more complex models.

3.3 Support Vector Machine

A Suport Vector Machine (SVM) is a supervised learning algorithm for classification or regression. We are going to compare the classifiers (SVC). Each object it wants to classify is represented as a point in a n-dimensional space. SVCs performs the classification by drawing a line between those points to separate them in categories. In our case, there are two categories (healthy/pneumonia).

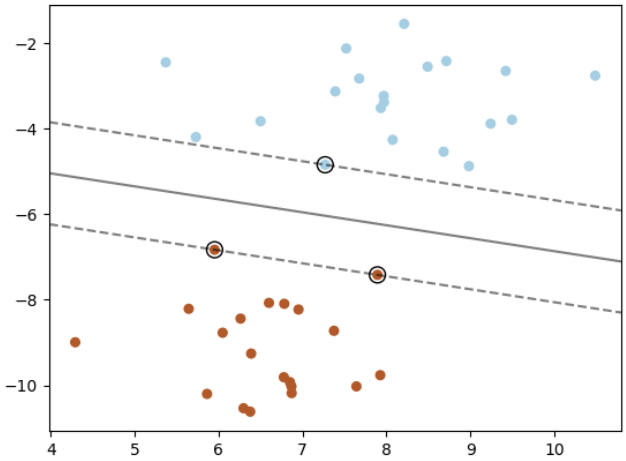


Figure 3.10: svm plot example from the sklearn documentation

3.3.1 Algorithms comparison

SVC only keeps part of the training dataset because its cost function ignores data beyond the margin. The NusSVC is similar with some mathematical differences. LinearSVC is a faster implemention of SVC because it only uses a linear kernel.

SVC, NuSVC and LinearSVC are capable of binary and multi-class classification. They can be used for pneumonia/healthy classification as well as viral/bacterial classification at the same time.

All three algorithms have a high false positive ratio. While the NuSVC has a lower false negative ratio, its false positive ratio above 80% makes it useless. The linear SVC and regular SVC with a linear kernel are obviously similar. They have a similar false negative ratio compared to the NuSVC but a better false positive ratio. We will keep the linear SVC since it's better than the NuSVC, similar to the regular SVC but faster.[5]

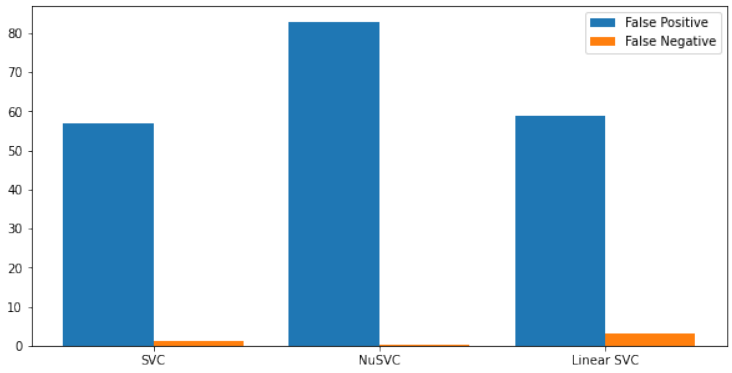


Figure 3.11: False positive and negative ratio comparison of different svc algorithms

3.3.2 Genetic algorithm

We implemented a genetic algorithm that tried to brute force the algorithm configuration by iterating over a parameter grid. The genetic algorithm implemented did not succeed in reducing the noise in a meaningful way and even reduced the model effectiveness in both positive and negative case detection compared to the default parameters.

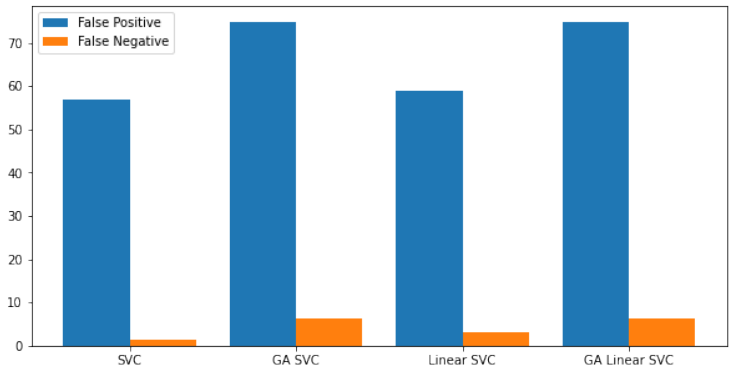


Figure 3.12: False positive and negative ratio comparison of different svc algorithms modified through genetic algorithm

3.3.3 Image size

There is little difference in the false positive and negative ratio above a size of 50x50, like with the Naive Bayes. SVC is a model that has trouble working with data of more dimension than it has training samples. That is why we see almost no difference when working with image size above a few thousands pixels (the size of the training data set being 5k).

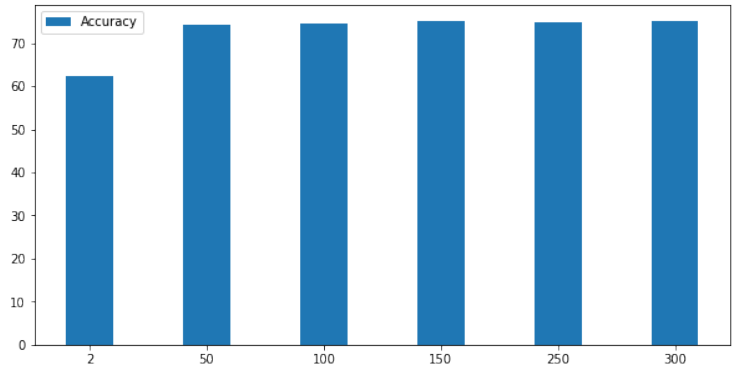


Figure 3.13: Accuracy comparison of image size for the linear svc model

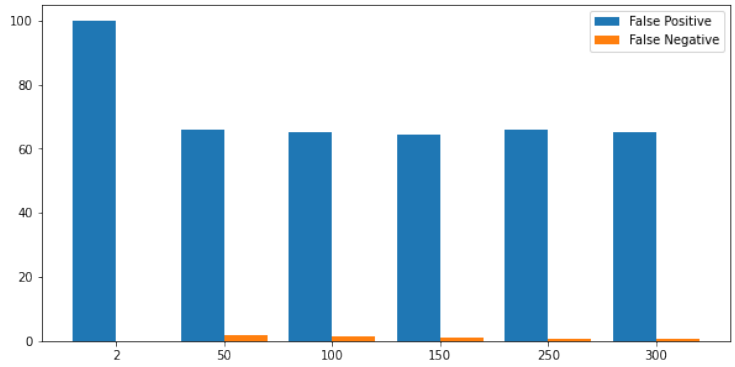


Figure 3.14: False positive and negative ratio comparison of image size for the linear svc model

3.3.4 Conclusion

SVM models are used for classification in bio informatics. They can rival neural networks on some tasks, for example in bio informatics when classifying molecules [6] or genes expression [7] while still being faster. But they quickly fall behind when neural networks get more layers and when the number of parameters increase, like in pictures.

3.4 Convolutionnal Neural Networks

A convolutionnal neural networks (CNN) is a class of artificial neural networks commonly used to analyse visual imagery. It differs from simple neural networks in the fact that it adds to each layer five convolutionnal blocks as well as a Fully Connected layer, it helps find similitude between the images.[8][9]

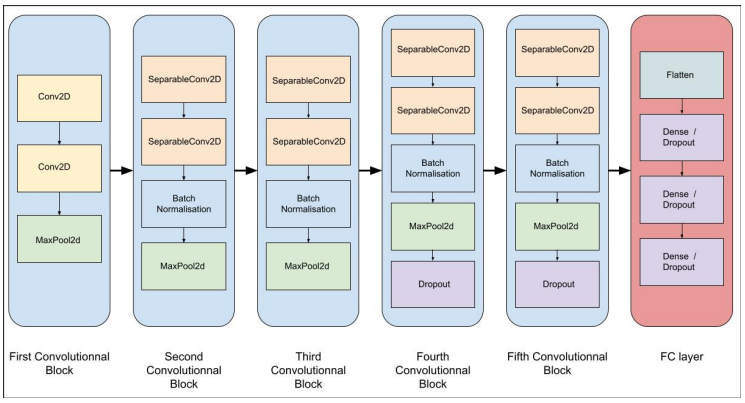


Figure 3.15: Architecture of the CNN model

We decided not to change the convolutionnals block of the model because we did not fully understand the mechanics behind those blocks. We decided to influence three parameters in order to create the best CNN model that we could: the size of the dataset, the quality of the images and the number of epochs of the CNN.[10]
All the testing was done with the algorithm provided in the CNN.ipynb notebook.[11]

3.4.1 Quality of the images

The quality of the images is intrinsically linked to its dimensions. We resize the X-ray images with 5 different dimensions: 100x100, 150x150, 200x200, 400x400, and 600x600.
For performance issues we decided to run the notebook on kaggle.com as it provides free TPU usage and we choose to keep the model with the best ratio between rapidity and performances.

Here is a summary chart of the performance of each of the dimensions:

	100x100	150x150	200x200	400x400	600x600
loss	1.0014	0.7766	1.1774	0.9999	1.1815
acc	0.7756	0.8125	0.7804	0.7772	0.7596
prec	0.7367	0.7725	0.7437	0.7390	0.7239
rec	0.9974	0.9923	0.9897	0.9949	0.9949

Figure 3.16: Comparison table of the metrics of the different model performances according to the size of the images

We can notice that the best dimension for our model was 150 by 150 as it has the best accuracy and precision and the least loss and recall.

3.4.2 Size of the Dataset

We used two different datasets in order to compare the influence of the size of the dataset on the performance of our model.
The first dataset consist of a total of 5865 gray-scale X-ray images of the chest, 5216 of which are used for training, 624 for testing and 8 for validation. Of the 5216 images of the training set, 1341 of them are of healthy people and the 3875 remaining are of people with pneumonia.
The second dataset consists of a total of 1440 gray-scale X-ray images of the chest, 800 of which are used for training, 624 for testing and 16 for validation. Of the 800 images of the training set, 400 of them are of healthy people and the 400 remaining are of people with pneumonia.

We came with the following results:

	Normal dataset	Small dataset
acc	89.4	62.5
precision	88.0	62.5
recall	96.1	100.0
fn	15.0	0.0
fp	51.0	234.0
tp	375.0	390.0
tn	183.0	0.0

Figure 3.17: Comparison table of the metrics of the different model performances according to the size of the dataset

We can see that for a small dataset of images, the CNN can't differentiate between normal and pneumonia and categorize each image as having pneumonia.
It would have been interesting to generate a bigger dataset since ours is relatively small. Nowadays, it is quite common to have dataset with more than 50 thousands radio-medical image in order to generate AI models.

3.4.3 Conclusion

After all the testing done, we arrived with the following results for our AI model:

```
--------- CONFUSION MATRIX ---------
[[173  61]
 [ 10 380]]

----------- TEST METRICS -----------
Accuracy: 88.62179487179486%
Precision: 86.16780045351474%
Recall: 97.43589743589743%
F1-score: 91.45607701564381

----------- TRAIN METRIC -----------
Train acc: 95.38
```

Figure 3.18: Metrics for the Final CNN Models Generated

The four numbers that appear in the confusion matrix part of the metrics are in order of appearance: predicted normal x-ray that are actually normal, predicted pneumonia x-ray that are actually normal, predicted normal x-ray that are actually pneumonia, predicted pneumonia x-ray that are actually pneumonia. This shows that there is a small amount of sick people that are not diagnosed (10 of 390), it isn't enough to be used in actual environment, but the results are promising.

There are a several points that could improve our model: we didn't test other convolutionnals blocks than those described above; we choose to change the weight of our image in order to re-balance the ratio of normal/pneumonia x-ray, we could choose to do data-augmentation which could improve the model; the dataset provided was too small, a bigger dataset could result in way better results; lastly we choose to use Tensorflow algorithm in order to generate our model, we saw that Tytorch combined with EfficientNet generate model with impressive results.

3.5 Transfer Learning

Transfer Learning is a machine learning method where we reuse a pre-trained model as the starting point for a model on a new task. To put it simply—a model trained on one task is repurposed on a second, related task as an optimization that allows rapid progress when modeling the second task.[12][13]
We choose to re-use the settings we parameters found on the CNN parameters comparison above in order to accelerate the process. We compared two tensorflow pre-trained model: the VGG19 model and the ResNet50V2 model.[14]
The dataset was imbalanced, the training set consisted in 3875 pneumonia X-ray images against 1341 normal X-ray images. In order to re-balance that we used data augmentation to artificially augment the number of normal images.

3.5.1 Comparison of the pre-trained models

We choose to keep the parameters as shown in a Kaggle notebook found [14].
That gave these results for the VGG19 pre-trained model:

```
Train Loss:  0.23332250118255615
Train Accuracy:  0.90625

Test Loss:  0.35419338941574097
Test Accuracy:  0.8461538553237915
```

Figure 3.19: Metrics for the VGG19 pre-trained model

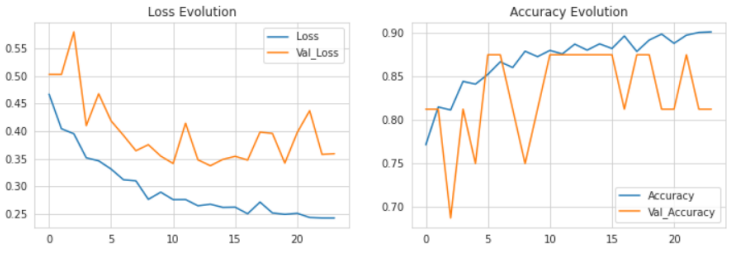


Figure 3.20: Graphics of the evolution of the metrics for the VGG19 pre-trained model

And these results for the ResNet50v2 pre-trained model:

```
Train Loss:  0.13077424466609955
Train Accuracy:  0.9486196041107178

Test Loss:  0.19524818658828735
Test Accuracy:  0.9278846383094788
```

Figure 3.21: Metrics for the ResNet50v2 pre-trained model

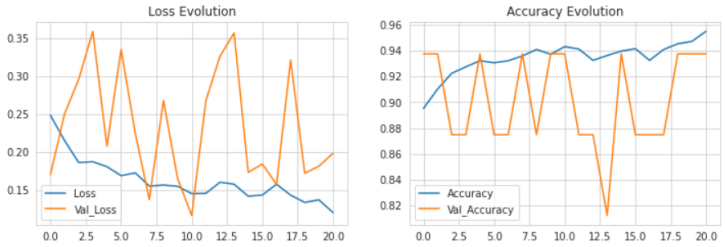


Figure 3.22: Graphics of the evolution of the metrics for the ResNet50v2 pre-trained model

When we compare these two pre-trained model we notice that the ResNet50v2 one has better loss an accuracy, so we decided to keep this pre-trained model for the fine tuning.

	VGG19 Metrics	ResNet50v2 Metrics
loss	0.3542	0.1952
accuracy	0.8462	0.9279
precision	0.9016	0.9367
recall	0.8462	0.9487
fn	60.0000	20.0000
fp	36.0000	25.0000
tn	198.0000	209.0000
tp	330.0000	370.0000
AUC	0.9285	0.9750

Figure 3.23: Comparison table of the metrics of the ResNet50v2 and VGG19 pre-trained model

3.5.2 Conclusion

After fine tuning our model we came to these results:

```
Train Loss:  0.06438476592302322
Train Accuracy:  0.9743098020553589


Test loss:  0.14678733050823212
Test Accuracy:  0.9439102411270142
```

Figure 3.24: Metrics for the ResNet50v2 pre-trained model fine tuned

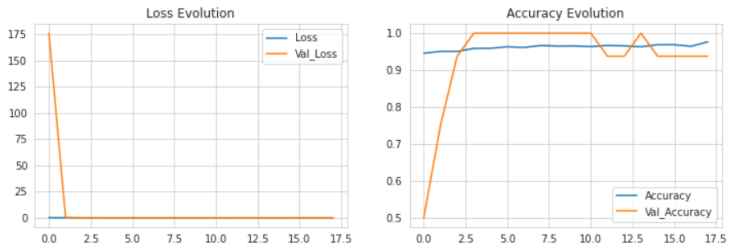


Figure 3.25: Graphics of the evolution of the metrics for the ResNet50v2 pre-trained model fine tuned

	ResNet50v2 Fine Tuned
AUC	0.987136
Accuracy	0.943910
FN	15.000000
FP	20.000000
Loss	0.146787
Precision	0.949367
Recall	0.961538
TN	214.000000
TP	375.000000

Figure 3.26: Table of the metrics of the ResNet50v2 fine tuned pre-trained model

We can conclude that the model generated is very accurate and has a small portion of false negatives. Maybe with some better fine tuning and with a bigger dataset it could be use in medical examens.

Chapter 4

Comparison of the Models

4.0.1 Metrics comparison

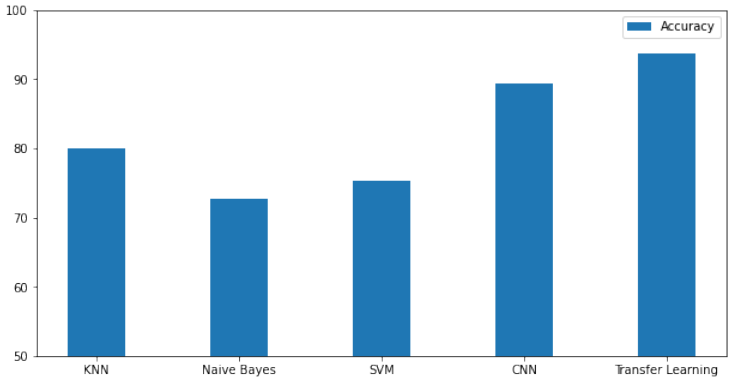


Figure 4.1: Accuracy of our best models

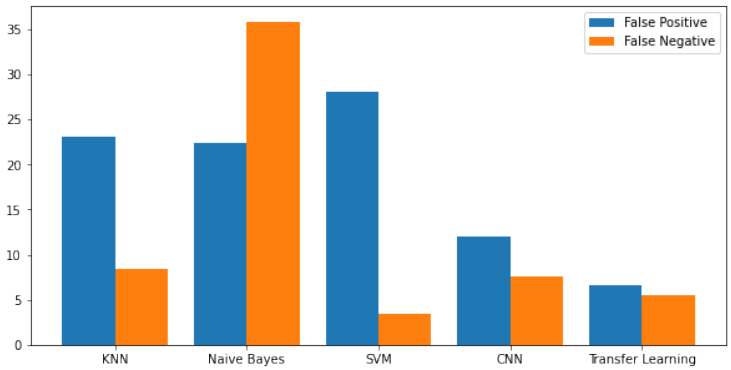


Figure 4.2: false ratios of our best models

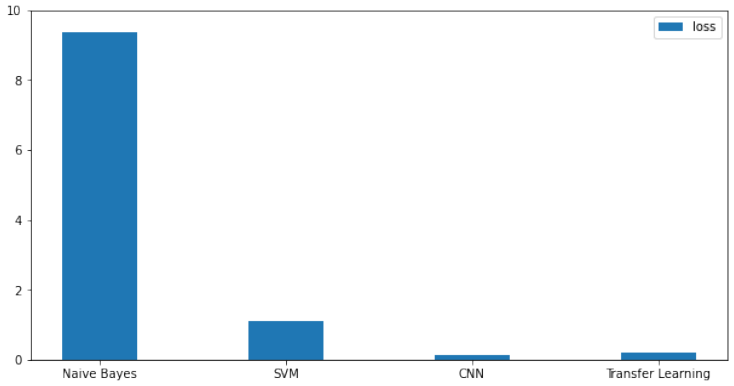


Figure 4.3: Loss of our best models

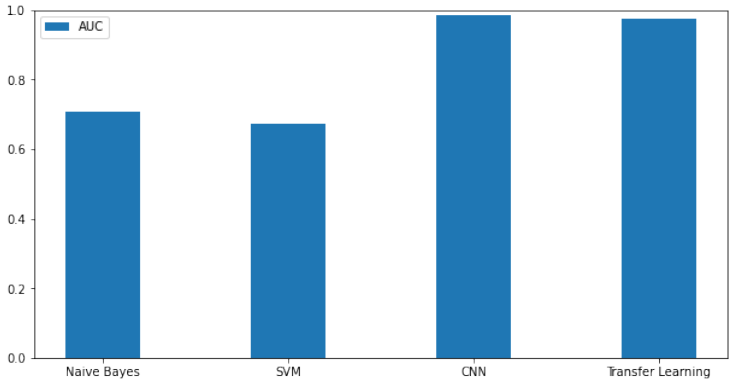


Figure 4.4: AUC of our best models

We can see a huge difference between the neural network and the non neural networks models. While the Naive Bayes reached 35% false negative, the other models all had below 10% false negative. The main difference between them is the False Positive rate. The two neural networks (a CNN and a CNN retrained trough transfer learning) are the only one with above 90% accuracy

4.0.2 Non neural network models limitations

The Gaussian Naive Bayes and Linear SVC have an accuracy of around 70% caused by a really high false positive ratio, meaning that the algorithms will label a lot of healthy patients as ill, necessitating all data to be reviewed again. They could be used as a tool to complement an human judgement but not be used as a reliable tool in a real environment.

Those models suffer from the same flaws, they become unreliable when the number of parameters is higher than the data set size.

The KNN starts at 80% accuracy, it is more suitable for our small data set than the Naive Bayes and SVC. Besides, when data set gets larger, KNN becomes much slower since it needs to keep the entire training set in mind to compare with the testing data set.

The SVC advantage is its low false negative ratio which is our priority since an undetected pneumonia can lead to death.

Chapter 5

Conclusion and Possible Improvements

5.1 Conclusion

Our findings match with the scientific publications and the published notebooks on the dataset page on kaggle where the Naive Bayes, SVC and KNN have low results while the CNN and transfer learning have the bets metrics.
In the current state, our models can't be used in a medical environment but, specially the transfer learning model, show promising results. With a larger dataset and better fine tuning we could probably arrive at a model which can be used by actual doctors.

5.2 Possible Improvements

As shown in the CNN.ipynb notebook, it is possible to determine the origin of the pneumonia, whether it is viral, bacterial or fungal. We implement an CNN AI to distinguish whether on our dataset the pneumonia is viral or bacterial of origin, but it is possible to generate a model that can automatically detect if the person has pneumonia and its origin. The same thing can be done for the other models.
Another path that has not been explored is unsupervised neural network. Some studies has shown near perfect results in detecting pneumonia and classifying its origin, but it poses some ethical problems. After analysing the results of an unsupervised neural network created in order to detect pneumonia, researchers found that the model was capable to find whether the person was Black, White or Asian with X-ray images of 160*160 pixels with more than 95% of accuracy. Even blurring the characteristic that researchers think can determine the race of a person (size of the bones, size of the heart, contours of the lungs, quality of the image...), researchers weren't able to determine how the model

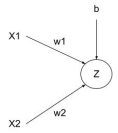
was capable of that.[15] These AI are very powerful but pose some ethical question with their utilisation.

Appendix A

Appendix

A.1 Machine learnia Deep learning

The principle behind is relatively simple: if we take an image of dimension 200x200 for example, after flattening it we obtain a vector v of dimension (1,4000). We want to create a function "f(v)=x" which will return a number between 0 and 1, the closer the number is to 1 the more likely the person has a pneumonia.[16]
For the sake of having simple calculus we will consider the vector v to be of dimension (1,2) : v =[x1,x2].
We can represent a neuron as a function z that will take as parameters our vector v as shown in the graph and the function below :



$$z(x_1, x_2) = w_1 x_1 + w_2 x_2 + b$$

We can then create the function sigmoid, which is the probability of truth :

$$\sigma(z) : \frac{1}{1 + \exp -z}$$

Loi de Bernouilli:

$$P(Y = y) = a(z)^y \alpha (1 - a(z))^{1-y}$$

Vraisemblance:

$$L = \Pi_{i=1}^{m}(a_i^{y_i}(1-a_i)^{1-y_i})$$

LogLoss:

$$\begin{aligned}
LogLoss &= \log(\Pi_{i=1}^{m}(a_i^{y_i}(1-a_i)^{1-y_i})) && \text{(A.1)}\\
&= \sum_{i=1}^{m}\log(a_i^{y_i}(1-a_i)^{1-y_i}) && \text{(A.2)}\\
&= \sum_{i=1}^{m}(\log(a_i^{y_i}) + \log(1-a_i)^{1-y_i}) && \text{(A.3)}\\
&= \sum_{i=1}^{m}(\log(a_i^{y_i}) + (1-y_i)\log(1-a_i)) && \text{(A.4)}\\
&= -\frac{1}{m}\sum_{i=1}^{m} y_i(\log(a_i) + (1-y_i)\log(1-a_i)) && \text{(A.5)}\\
& && \text{(A.6)}
\end{aligned}$$

References

[1] S. Narkhede, "Understanding auc roc curve," Accessed on 26 Jun 2018. [Online]. Available: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

[2] Scikit-Learn, "Naive bayes." [Online]. Available: https://scikit-learn.org/stable/modules/naive_bayes.html

[3] P. P. Li Fei-Fei, "A bayesian hierarchical model for learning natural scene categories." [Online]. Available: https://journal.iis.sinica.edu.tw/paper/1/160230-2.pdf?cd=B61CB3F3C7E72FF7E

[4] I.-C. C. Shih-Chung Hsu and C.-L. Huang, "Image classification using naive bayes classifier with pairwise local observations." [Online]. Available: https://journal.iis.sinica.edu.tw/paper/1/160230-2.pdf?cd=B61CB3F3C7E72FF7E

[5] Scikit-Learn, "Naive bayes svc." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[6] J. S. Evgeny Byvatov, Uli Fechner and G. Schneider, "Comparison of support vector machine and artificial neural network systems for drug/nondrug classification." [Online]. Available: https://pubs.acs.org/doi/full/10.1021/ci0341161

[7] Z. F. Ying Peng, Cheng Peng and G. Chen, "Bioinformatics analysis identifies molecular markers regulating development and progression of endometriosis and potential therapeutic drugs." [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8372410/

[8] Z. W. G. G.-T. H. R. S. Rohit Kundu, Ritacheta Das, "Pneumonia detection in chest x-ray images using an ensemble of deep learning models." [Online]. Available: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0256630

[9] J. G. E. D. V. M. D. B. T. J. A. Lei Rigi Baltazar, Mojhune Gabriel Manzanillo, "Artificial intelligence on covid-19 pneumonia detection using chest xray images."

[Online]. Available: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0257884

[10] 3Blue1Brown, "Neural networks." [Online]. Available: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi

[11] A. Sagar, "Deep learning for detecting pneumonia from x-ray images." [Online]. Available: https://towardsdatascience.com/deep-learning-for-detecting-pneumonia-from-x-ray-images-fc9a3d9fdba8

[12] A. l. Mohammad Rahimzadeha, "A modified deep convolutional neural network for detecting covid-19 and pneumonia from chest x-ray images based on the concatenation of xception and resnet50v2." [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352914820302537

[13] S. T. D. J. E. F. T. Yusuf Brima, Marcellin Atemkeng, "Transfer learning for the detection and diagnosis of types of pneumonia including pneumonia induced by covid-19 from chest x-ray images." [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8394302/

[14] K. Shingde, "Pneumonia detection - transfer learning(95% acc)." [Online]. Available: https://www.kaggle.com/code/karan842/pneumonia-detection-transfer-learning-95-acc?scriptVersionId=91935490

[15] A. lot of scientists, "Reading race: Ai recognizes patient's racial identity in medical images." [Online]. Available: https://arxiv.org/ftp/arxiv/papers/2107/2107.10356.pdf

[16] MachineLearnia, "Formation deep learning." [Online]. Available: https://www.youtube.com/playlist?list=PLO_fdPEVlfKoanjvTJbIbd9V5d9Pzp8Rw