Fait à Manouba le : 04-08-2023

A L'attention du Directeur

de la Société : Laboratoire CRISTAL (ENSI)

Objet : Lettre d'affectation à un stage obligatoire

Monsieur,

Suite à l'offre de stage que vous avez eu l'amabilité d'accorder à **Mohamed Amine Kacem** étudiant(e) à , l'École Nationale des Sciences de l'Informatique , inscrit(e) en premiere année, j'ai le plaisir de confirmer par la présente son affectation à votre honorable établissement et ce, pour un stage **OBLIGATOIRE** du **31-07-2023 au 25-08-2023.**

Je saisis cette occasion pour vous exprimer mes vifs remerciements pour votre précieuse collaboration.

Nous vous signalons, que durant la période de stage, l'étudiant est couvert par la Mutuelle Accident Scolaire et Universitaire-MASU.

Par ailleurs, je me tiens à votre entière disposition pour tout autre renseignement concernant les stages (direction.stages@ensi-uma.tn).

Veuillez croire, Madame, Monsieur, à l'expression de ma haute considération.

La directrice des stages

Raoudha KHCHERIF

Raoudha

CONVENTION DE STAGE D'IMMERSION EN ENTREPRISE

1ème année Ingénieur en Informatique

Entre

L'École Nationale des Sciences de l'Informatique,

L'élève ingénieur Mohamed Amine Kacem

&

L'organisme d'accueil **Laboratoire CRISTAL (ENSI)**

Préambule

Le présent document exprime le souhait des soussignés de développer la coopération scientifique et technologique dans le cadre du **Stage d'été** que tous les élèves ingénieurs qui ont réussi la première année à l'Ecole Nationale des Sciences de l'Informatique (ENSI) doivent effectuer. Ce stage est obligatoire et fait partie intégrante du cursus des élèves ingénieurs de l'ENSI comme indiqué dans l'article 11 du régime des études et des examens applicable à l'ENSI.

A cette fin,
L'École Nationale des Sciences de l'Informatique (ci-après dénommée **ENSI)**
Adresse professionnelle : ENSI- Campus Universitaire Manouba, 2010 Manouba.
Email : direction@ensi-uma.tn
Tél : + 216 71 600 224 / +216 71 600 444 poste 102 - Fax : 216 71 600 449

Représentée par sa directrice de stages **Raoudha KHCHERIF**

L'élève ingénieur (ci-après dénommé **Stagiaire**) : **Mohamed Amine Kacem**
Sexe : F □ M □
Filière : **Diplôme Nationale d'Ingénieur en Informatique** Groupe : ………Tél :**28543086**
Email: **mohamedamine.kacem@ensi-uma.tn**
Et
L'organisme d'accueil (ci-après dénommé **Organisme**) : **Laboratoire CRISTAL (ENSI)**
Représenté par son. Mme/Mr **Mohamed Houcine Elhdhili**
Adresse :**[entreprise_adresse]**
Tél :**71 600 444** Fax : **71 600 449**
Email : **webmaster@ensi.rnu.tn**
Approuvent la présente convention de stage avec les articles suivants :

Article 1 : Objectifs

La présente convention concerne les stages d'été effectués par les étudiants de l'ENSI qui ont réussi la première année des études d'ingénieur en informatique. Ces projets ont pour objets essentiels d'une part, de mettre l'étudiant au contact des réalités du milieu industriel (ou des laboratoires de recherches) et, d'autre part, de mettre en œuvre, dans un cadre réel, les connaissances théoriques acquises durant la première année d'études d'ingénieur en informatique à **l'ENSI**.

Article 2 : Définition et durée du projet

Le stage porte sur le sujet suivant :.Breast cancer detection using AI techniques
Mots clés :..
Il se déroulera à : .Laboratoire CRISTAL (ENSI)
Sous la responsabilité de : ...Mohamed Houcine Elhdhili   à l'Organisme
Durant la période du : 31-07-2023 au : 25-08-2023
L'indemnité reçue lors du stage est de (facultatif) : ....
Dans les conditions normales, le **Stagiaire** ne peut commencer le stage d'été qu'une fois déclaré admis en deuxième année d'ingénieur en informatique à **l'ENSI.** Si le **Stagiaire** entame son stage d'été avant que les résultats des délibérations des sessions principale et de rattrapage n'aient été déclarés, il s'engage à **1)** respecter le calendrier des examens de rattrapage qui lui sera communiqué à la suite de la déclaration des résultats de la session principale **2)** passer tous les examens de rattrapage qui s'imposent et **3)** refaire le stage d'été en cas de redoublement de la première année ingénieur.
Le stage d'immersion en entreprise dure au minimum 4 semaines et au maximum 2 mois. Il doit commencer au plus tôt le lundi 05/06/2023 et se terminer au plus tard le vendredi 25/08/2023.
La convention de stage doit être signée par le **Stagiaire**. Les deux premières pages doivent être paraphées et la troisième page signée par l'**Organisme**. La convention devra ensuite être déposée au secrétariat des départements au plus tard 3 jours après le début du stage et envoyée à l'adresse email suivante : direction.stages@ensi-uma.tn.
L'ENSI ne signe qu'une seule convention de stage d'été par **Stagiaire**. Si l'étudiant et l'Organisme décident d'un commun accord, après la signature de la convention, l'annulation du stage d'été, une attestation rédigée dans ce sens sera envoyée par un responsable de l'Organisme à l'adresse email indiquée précédemment avant que l'étudiant ne soit autorisé à signer une nouvelle convention avec un nouvel organisme d'accueil.

Article 3 : Encadrement scientifique

Durant la période de stage, le **Stagiaire** est encadré par le responsable de l'**Organisme** listé dans l'article 2. Ce dernier, en plus de l'encadrement scientifique, doit fournir les moyens matériels nécessaires au bon déroulement du stage. Si deux ou plusieurs élèves ingénieurs de **l'ENSI** effectuent leurs stages d'été dans le même organisme d'accueil, **le responsable de l'Organisme qui encadre ces élèves ingénieurs doit s'assurer que les tâches affectées à chaque élève ingénieur sont complémentaires et non redondantes.**

Article 4: Financements du projet

L'acquisition du matériel et fournitures nécessaires pour la réalisation du projet est à la charge de l'**Organisme**. En fonction des modalités de déroulement du projet de stage, l'**Organisme** peut accorder une subvention au **Stagiaire** (déplacement et hébergement du **Stagiaire**).

Article 5: Fin de stage –Attestation – Validation

A l'issue du stage, l'**Organisme** délivre au **Stagiaire** une attestation datée, signée et portant le cachet de l'Organisme, mentionnant au minimum le nom du stagiaire, la durée du stage et le sujet du stage.
Une version électronique de cette attestation **devra être envoyée** à la direction des stages de l'**ENSI** à l'adresse email suivante: direction.stages@ensi-uma.tn
Le **Stagiaire** doit deposer en ligne un rapport de stage en format pdf **au plus tard le 02/09/2023.**
La validation du stage d'été aura lieu dans les locaux de l'ENSI au début de l'année universitaire 2023-2024. Le Stagiaire devra défendre son travail lors d'une présentation orale devant un jury constitué d'enseignants de l'ENSI.

Article 6: Engagement de l'étudiant

- Durant le stage, le **Stagiaire** devra se conformer aux usages et règlements de l'**Organisme**. Il est astreint au secret professionnel de l'**Organisme** et s'il est amené à utiliser des informations recueillies dans le cadre du stage, il ne pourra le faire qu'après l'accord de l'**Organisme**.
- En cas de transgression, par le **Stagiaire**, du règlement intérieur ou en cas de constat d'indiscipline, le responsable de l'**Organisme** se réserve le droit de mettre fin au stage, après avoir prévenu la direction de l'ENSI à l'adresse email suivante: direction.stages@ensi-uma.tn

Article 7: Sécurité sociale et assurance

- Durant le stage, le **Stagiaire** est pris en charge par la sécurité sociale étudiante.
- En cas d'accident, l'**Organisme** n'assume aucune responsabilité vis à vis du **Stagiaire**.
- En cas de stage à l'étranger, le **Stagiaire** s'occupera de la prise en charge de la sécurité sociale par ses propres moyens ou, le cas échéant, par l'**Organisme**.

Le Stagiaire: Mohamed Amine **Kacem** **lieu et date:**	[illegible]
Pour l'organisme: **Mr/Mme:** Mohamed Houcine Elhdhili **lieu et date**	cachet et signature
Pour l'ENSI :la directrice des stages: Raoudha KHCHERIF ***Manouba le:04-08-2023***	cachet et signature

Manouba, le 04-08-2023

Objet : Attestation

Je soussignée, Raoudha KHCHERIF , directrice des stages à l'Ecole Nationale des Science de l'Informatique (ENSI), atteste par la présente que l'élève ingénieur **Mohamed Amine Kacem** effectuera son stage d'été obligatoire au sein de l'organisme Laboratoire CRISTAL (ENSI) et ce du 31-07-2023 au 25-08-2023
Je déclare également avoir pris connaissance de la volonté de l'élève ingénieur à effectuer, durant son temps libre et sans aucun chevauchement avec le stage d'iété décrit ci-dessus, un second stage d'été optionnel et volontaire au sein de l'organisme **Laboratoire CRISTAL (ENSI)** et ce du 31-07-2023.au 25-08-2023
Au début de l'année universitaire 2022/2023, l'élève ingénieur doit remettre un rapport de stage relatif **uniquement** au stage d'été obligatoire qu'il doit valider devant un jury composé d'enseignants de l'ENSI. Cela dit, l'ENSI encourage les stages volontaires en entreprise car ils contribuent à parfaire le projet professionnel de l'étudiant et lui permettent d'acquérir plus d'expériences pratiques essentielles à sa formation d'ingénieur en Informatique.

Il est à noter que, durant ce second stage volontaire, l'élève ingénieur s'occupera de la prise en charge de la sécurité sociale par ses propres moyens ou, le cas échéant, par l'organisme d'accueil.

[directeur_satge]

Summer Internship **2022/2023**

University of Manouba
National School of Computer Science

Internship Report

Subject:
Using AI techniques to breast cancer detection

Directed by:
Mohamed Amine Kacem

Supervised by: Mohamed Houcine Elhdhili

Organism: Laboratoire CRISTAL (ENSI)

Fax: 71 600 449

E-mail: webmaster@ensi.run.tn

Introduction:

Cancer is a disease that continues to plague our modern society. Among all types of cancer, breast cancer is now the most common type of cancer occurring in women worldwide. Various factors, including genetics, lifestyle, and the environment, have contributed to the rise in the prevalence of breast cancer among women of all socioeconomic strata. Therefore, proper screening for early diagnosis and treatment becomes a major factor when fighting the disease. Artificial intelligence (AI) continues to revolutionize various spheres of our lives with its numerous applications. Using AI in the existing screening process makes obtaining results even easier and more convenient. Faster, more accurate results are some of the benefits of AI methods in breast cancer screening.

How is breast cancer diagnosed by doctors ?

Removing a sample of breast cells for testing (biopsy):

A biopsy is the only definitive way to make a diagnosis of breast cancer. During a biopsy, doctors use a specialized needle device guided by X-ray or another imaging test to extract a core of tissue from the suspicious area. Often, a small metal marker is left at the site within the breast so the area can be easily identified on future imaging tests.

Biopsy samples are sent to a laboratory for examination by a pathologist under a microscope to determine whether the cells are cancerous. A biopsy sample is also analyzed to determine the type of cells involved in the breast cancer, the aggressiveness (grade) of the cancer, and whether the cancer cells have hormone receptors or other receptors that may influence the treatment options.

Relevant fields :

Before understanding the role of AI in the diagnosis process, we have to go through some important technologies in relevant fields that paved the way for AI to make some significant advancement in the area of cancer detection.

1- Digital pathology:

Digital pathology creates high-resolution images from tissue specimens mounted on glass slides and is of significance to the diagnostic field . It transforms histology glass slides into digital images using computerized technology. It stores histologic information, which can later be used for the analysis of any other slide by allowing doctors to detect any variation from the normal and thereby evaluate the disease and its progression. This is extremely helpful in diagnosing various diseases, especially cancer.

The role of digital pathology :

Digital pathology plays a vital role in modern medical practice. The availability of faster and cheaper options has made it easier for clinicians and pathologists to store and handle information. Advances in machine learning have enabled the combination of digital pathology and AI, offering image-based diagnosis possibilities. Oncology especially is one of the fields that has benefited a lot from this advancement. The complex signaling and transcription pathways, connecting cancer, stromal, and immune cells can be displayed as images using this technology. The opportunity of viewing whole-sized images of tissue specimens has helped in the viewing of subvisual morphometric phenotypes and has made way for better patient care.

Data collection with digital pathology :

Analysis of data on pathological samples has become easier with digital pathology and has led to a more in-depth understanding of the data gathered. Digital methods allow data to be gathered, integrated, and analyzed to greater extents than what is achievable using traditional techniques, often with more efficiency than the latter. It has great potential of achieving more reliable, precise data of larger volume as outputs with a sizeable scale of summary and comparison.

For AI algorithms, it is necessary to have good-quality training images. It is the work of the pathologist to manually identify and annotate the regions showing abnormalities or of any pathological significance. Under ideal conditions, it is performed by experts in the field. Qualitative evaluation can quickly identify cell types accurately and give an insight into histological, morphological, and biologically relevant patterns. Qualitative analysis allows accessing data from tissue sides that are not possible to be evaluated manually. Manual methods have a higher margin of error in comparison to digital methods considering the errors in the accuracy of data collected and processed by humans may not be as precise as a computer run by AI.

2- Radiomics :

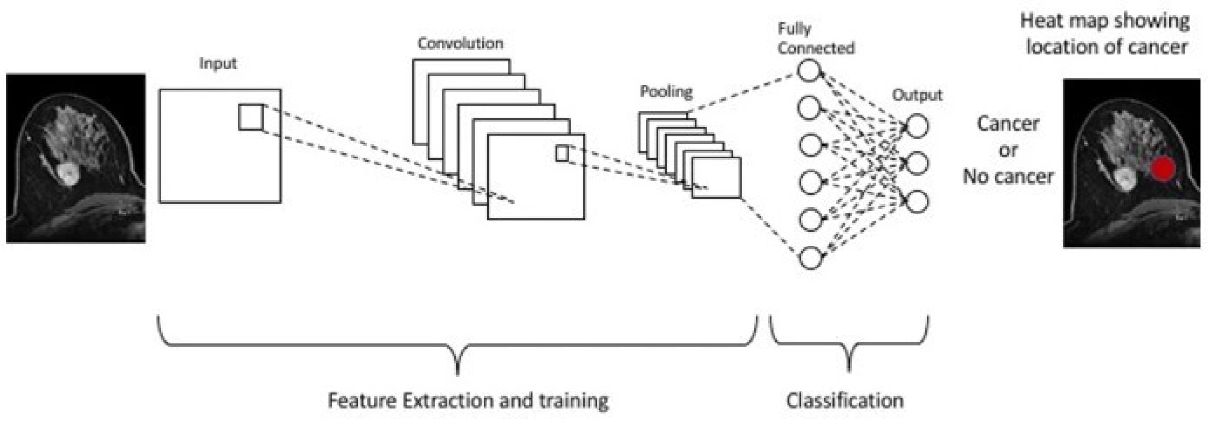
Radiomics is a technique widely used in AI systems. It extracts quantitative aspects from an image called a feature. This usually occurs by pattern recognition algorithms that recognize images and provide as its outcome a set of numbers that represent a quantitative feature of the part of the image under view. Radiomics is based on the idea that extracted features represent various activities happening at the genetic and molecular levels. Machine learning consists of computational algorithms that make use of image features extracted by employing radiomics to help understand disease outcomes. The dataset that I trained my models on contains features extracted by employing radiomics.

Artificial intelligence and screening of breast cancer :

1- End to end deep learning model :

Figure 1 shows a graphic representation of a deep learning workflow. The input layer represents the breast cancer image that serves as input to the CNN. The multiple convolutional layers are stacked on top of the input layer. Each convolutional layer applies filters or kernels to extract specific features from the input image. These filters learn to detect patterns such as edges, textures, or other relevant features related to breast cancer. After each convolutional layer, activation functions like rectified linear unit (ReLU) are typically applied to introduce nonlinearity into the network. Following some of the convolutional layers, pooling layers are used to downsample the spatial dimensions of the feature maps. Common pooling techniques include max-pooling or average pooling. Pooling helps reduce the computational complexity and extract the most salient features. After the convolutional and pooling layers, fully connected layers are employed. These layers connect all the neurons from the previous layers to the subsequent layers. Fully connected layers enable the network to learn complex relationships between features. The final layer is the output layer, which provides the classification or prediction. In the case of breast cancer detection, it might output the probability or prediction of malignancy or benignity.

Fig. 1



2- Breast cancer detection using quantitative features :

This project focuses on building machine and deep learning models to take as input features that are either extracted by employing radiomics or extracted from a CNN after applying a Flatten layer.

About the dataset (from Kaggle):

The features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.

Attribute Information:

1) ID number
2) Diagnosis (M = malignant, B = benign)
3-32)

Ten real-valued features are computed for each cell nucleus:

a) radius (mean of distances from center to points on the perimeter)
b) texture (standard deviation of gray-scale values)
c) perimeter
d) area
e) smoothness (local variation in radius lengths)
f) compactness (perimeter^2 / area - 1.0)
g) concavity (severity of concave portions of the contour)
h) concave points (number of concave portions of the contour)
i) symmetry
j) fractal dimension ("coastline approximation" - 1)

The mean, standard error and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features. For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius.

Importing the dependencies:

First thing, we need to import our dependencies which consist of frameworks, libraries, functions and modules we are going to use in the process of building the model.

```python
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
import numpy as np
import seaborn as sns
from sklearn.decomposition import PCA
import pandas as pd
from sklearn.preprocessing import StandardScaler
import sklearn.datasets as ds
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, f1_score
from sklearn.model_selection import cross_val_score
import matplotlib.pyplot as plt
import xgboost as xgb
```

Loading the dataset :

The breast cancer dataset is also available in the sklearn.datasets module provided by scikit-learn, so all we need to do is load from there.

```python
breast_cancer_dataset=ds.load_breast_cancer()
print(type(breast_cancer_dataset))
```

```
<class 'sklearn.utils._bunch.Bunch'>
```

As we can see, the type of the dataset loaded directly from the sklearn.datasets module is a <class 'sklearn.utils._bunch.Bunch'>. So what is this class ?
This object is a simple container or dictionary-like data structure that holds various attributes related to the dataset. We can work directly with the bunch object to access its attributes, which typically include ‘data’, ‘target’ and ‘feature_names’.

Transforming the data into a pandas DataFrame :

The reason we opted for DataFrame structure from pandas is because we are dealing with tabular data and by definition, a DataFrame is a two-dimensional data structure, which means it has rows and columns. If we were dealing with an unstructured dataset a DataFrame will no longer be the best choice, in which case we might opt to a dataset object from tf.Data.dataset.
Below is the syntax to convert that object we got to a DataFrame:

```
ds=pd.DataFrame(data=breast_cancer_dataset.data, columns=breast_cancer_dataset.feature_names)
ds['Diagnosis']=breast_cancer_dataset.target
print(type(ds))
```

```
<class 'pandas.core.frame.DataFrame'>
```

Exploratory data analysis (EDA):

The next step is to do some exploratory data analysis (EDA) in order to gain some insights about our dataset:

```
print('the shape is: ',ds.shape)
print("general infos: \n",ds.info())
```

```
the shape is:  (569, 31)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   mean radius              569 non-null    float64
 1   mean texture             569 non-null    float64
 2   mean perimeter           569 non-null    float64
 3   mean area                569 non-null    float64
 4   mean smoothness          569 non-null    float64
 5   mean compactness         569 non-null    float64
 6   mean concavity           569 non-null    float64
 7   mean concave points      569 non-null    float64
 8   mean symmetry            569 non-null    float64
 9   mean fractal dimension   569 non-null    float64
 10  radius error             569 non-null    float64
 11  texture error            569 non-null    float64
 12  perimeter error          569 non-null    float64
 13  area error               569 non-null    float64
 14  smoothness error         569 non-null    float64
 15  compactness error        569 non-null    float64
 16  concavity error          569 non-null    float64
 17  concave points error     569 non-null    float64
 18  symmetry error           569 non-null    float64
 19  fractal dimension error  569 non-null    float64
 20  worst radius             569 non-null    float64
 21  worst texture            569 non-null    float64
 22  worst perimeter          569 non-null    float64
 23  worst area               569 non-null    float64
 24  worst smoothness         569 non-null    float64
 25  worst compactness        569 non-null    float64
 26  worst concavity          569 non-null    float64
 27  worst concave points     569 non-null    float64
 28  worst symmetry           569 non-null    float64
 29  worst fractal dimension  569 non-null    float64
 30  Diagnosis                569 non-null    int64
dtypes: float64(30), int64(1)
memory usage: 137.9 KB
general infos:
 None
```

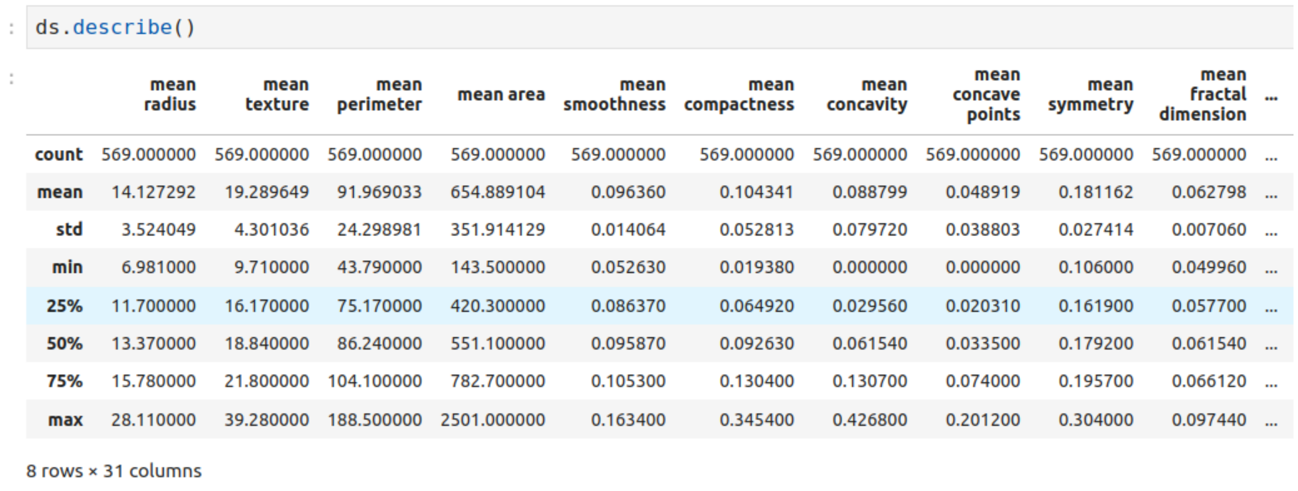
We can also check for missing values in the dataset:

```
print(ds.isnull().sum())
```

```
mean radius                0
mean texture               0
mean perimeter             0
mean area                  0
mean smoothness            0
mean compactness           0
mean concavity             0
mean concave points        0
mean symmetry              0
mean fractal dimension     0
radius error               0
texture error              0
perimeter error            0
area error                 0
smoothness error           0
compactness error          0
concavity error            0
concave points error       0
symmetry error             0
fractal dimension error    0
worst radius               0
worst texture              0
worst perimeter          0
worst area               0
worst smoothness         0
worst compactness        0
worst concavity          0
worst concave points     0
worst symmetry           0
worst fractal dimension  0
Diagnosis                0
dtype: int64
```

Fortunately our dataset doesn’t contain any missing values.

The next step could be to get some statistics about the dataset:



```
ds.describe()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	...
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0.181162	0.062798	...
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.027414	0.007060	...
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000	0.049960	...
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900	0.057700	...
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200	0.061540	...
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195700	0.066120	...
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000	0.097440	...

8 rows × 31 columns

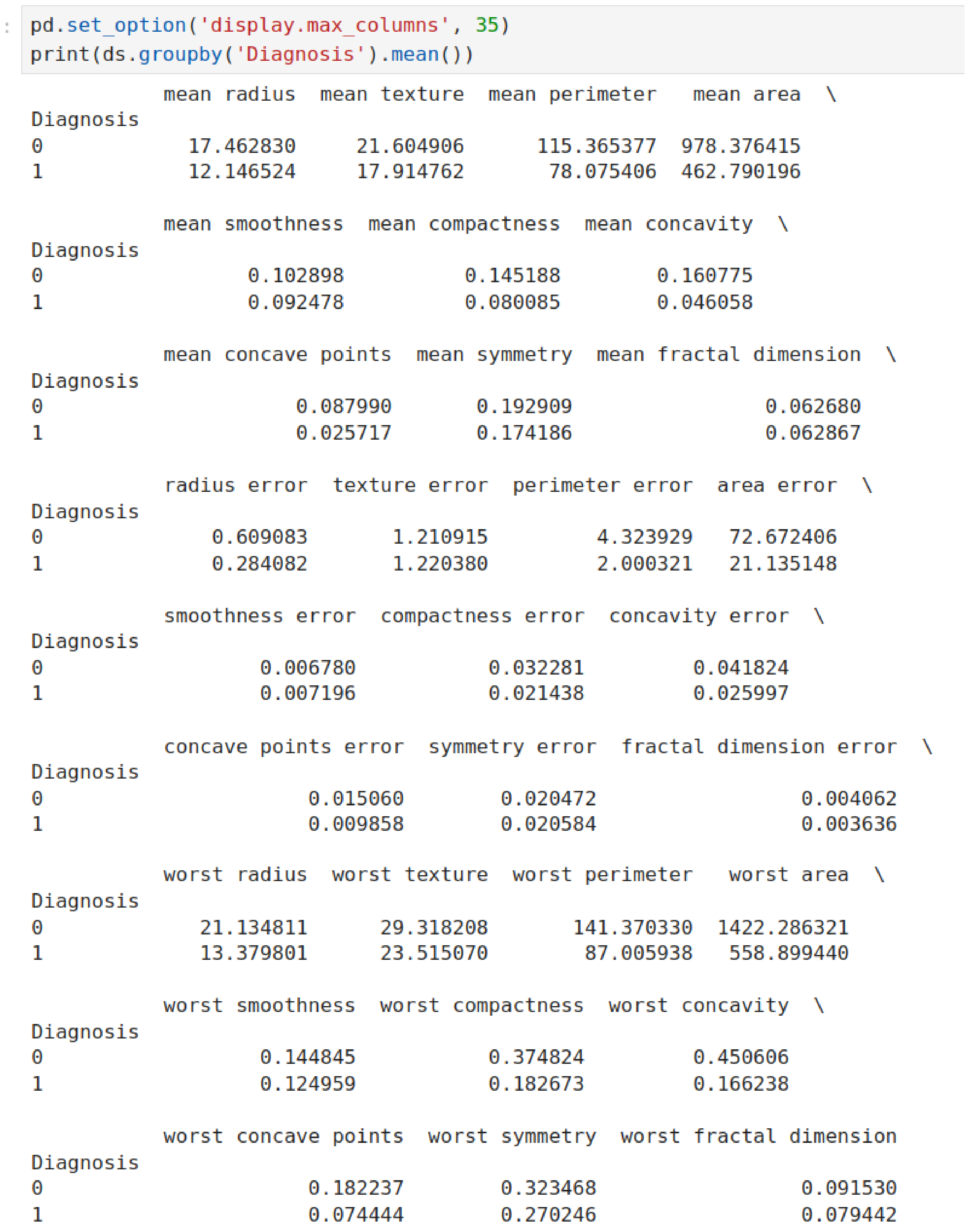
Distribution of the targets :

```
print("Note: 1 means Benign and 0 means Malignant\n")
ds['Diagnosis'].value_counts()
```

```
Note: 1 means Benign and 0 means Malignant

Diagnosis
1    357
0    212
Name: count, dtype: int64
```

Grouping the two targets:

This step is very important as it enables us to see if there are any features that present a large difference in their mean values. If such features are present we can then plot them and see the data distribution to get more insight about the data. In addition if two features present similar mean values for each class then we can omit them in the training process.



```
pd.set_option('display.max_columns', 35)
print(ds.groupby('Diagnosis').mean())
```

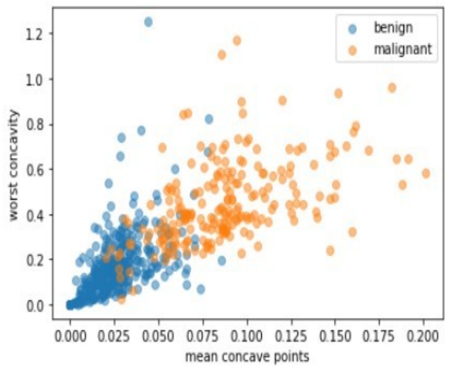
```
           mean radius  mean texture  mean perimeter   mean area  \
Diagnosis
0            17.462830     21.604906      115.365377  978.376415
1            12.146524     17.914762       78.075406  462.790196

           mean smoothness  mean compactness  mean concavity  \
Diagnosis
0                 0.102898          0.145188        0.160775
1                 0.092478          0.080085        0.046058

           mean concave points  mean symmetry  mean fractal dimension  \
Diagnosis
0                     0.087990       0.192909                0.062680
1                     0.025717       0.174186                0.062867

           radius error  texture error  perimeter error  area error  \
Diagnosis
0              0.609083       1.210915         4.323929   72.672406
1              0.284082       1.220380         2.000321   21.135148

           smoothness error  compactness error  concavity error  \
Diagnosis
0                  0.006780           0.032281         0.041824
1                  0.007196           0.021438         0.025997

           concave points error  symmetry error  fractal dimension error  \
Diagnosis
0                      0.015060        0.020472                 0.004062
1                      0.009858        0.020584                 0.003636

           worst radius  worst texture  worst perimeter   worst area  \
Diagnosis
0             21.134811      29.318208       141.370330  1422.286321
1             13.379801      23.515070        87.005938   558.899440

           worst smoothness  worst compactness  worst concavity  \
Diagnosis
0                  0.144845           0.374824         0.450606
1                  0.124959           0.182673         0.166238

           worst concave points  worst symmetry  worst fractal dimension
Diagnosis
0                      0.182237        0.323468                 0.091530
1                      0.074444        0.270246                 0.079442
```

Following an intuitive approach, we can spot that ‘mean concave points’ and ‘worst concavity’ present a large difference between the two targets so we can plot them to get a better understanding about the data. In addition ‘mean fractal dimension’, ‘texture error’, ‘symmetry error’, ‘smoothness error’ and ‘fractal dimension error’ seem to be useless features as they have approximately the same mean value for both classes.

Data Visualization :

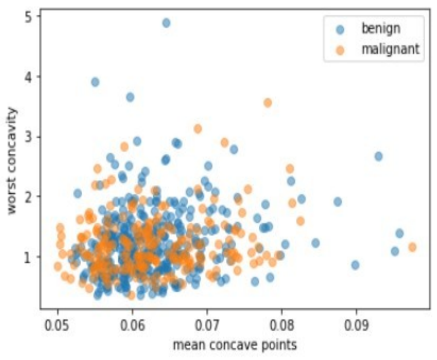
Following the intuitive approach we discussed above we could plot the two features that represent the largest difference in their mean values. In our case the features are ‘mean concave points’ and ‘worst concavity’

```
# seperating the benign data from the malignant data
ds_benign = ds[ds['Diagnosis']==1]
ds_malignanat = ds[ds['Diagnosis']==0]
# plotting the two features with the most variance in the mean value (an intuitive approach)
plt.scatter(ds_benign['mean concave points'], ds_benign['worst concavity'], label='benign', alpha=0.5)
plt.scatter(ds_malignanat['mean concave points'], ds_malignanat['worst concavity'], label='malignant', alpha=0.5)
plt.xlabel('mean concave points')
plt.ylabel('worst concavity')
plt.legend()
plt.show()
```



As we expected, the plot clearly shows two clusters of data points. Now let’s try and plot two features that have negligent difference between their mean values corresponding to every class :

```
# seperating the benign data from the malignant data
ds_benign = ds[ds['Diagnosis']==1]
ds_malignanat = ds[ds['Diagnosis']==0]
# plotting two features that represent similar mean values for both classes
plt.scatter(ds_benign['mean fractal dimension'], ds_benign['texture error'], label='benign', alpha=0.5)
plt.scatter(ds_malignanat['mean fractal dimension'], ds_malignanat['texture error'], label='malignant', alpha=0.5)
plt.xlabel('mean concave points')
plt.ylabel('worst concavity')
plt.legend()
plt.show()
```



The data points are very randomly distributed and show no clear clusters.

Note:
We could also plot all possible pairs of features using the following line of code “sns.pairplot(ds, hue='Diagnosis')” but that will result in a 30x30 grid with 900 cells which will very laborious to go over one by one.

Data Preprocessing :

Splitting the data into train/development/test sets:

The primary purpose of splitting data is to evaluate the performance of a machine learning model. Each subset serves a specific role in this process:

- **Training Set:** This subset is used to train the machine learning model. The model learns patterns and relationships in the data from this portion.
- **Development (Validation)** Set: This is used to tune hyperparameters, assess model performance during training, and prevent overfitting. It helps in selecting the best model among different choices.
- **Test Set:** This subset is used to assess the model's performance on unseen data. It provides an estimate of how well the model is likely to perform in a real-world scenario.

```
X = ds.drop(columns=['Diagnosis','mean fractal dimension', 'texture error', 'symmetry error', 'smoothness error',\
                    'fractal dimension error'] , axis=1)
y = ds['Diagnosis']
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.4, random_state=42)
X_dev,X_test,y_dev,y_test = train_test_split(X_train,y_train,test_size=0.5, random_state=42)
```

Data Normalization:

Normalizing or scaling data is a crucial preprocessing step in many machine learning algorithms and data analysis tasks. It involves transforming the data to have a consistent scale or distribution which will help the gradient descent algorithm to converge faster.

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_dev = scaler.transform(X_dev)
X_test = scaler.transform(X_test)
print("X_train.shape = ",X_train.shape)
print("X_test.shape = ",X_test.shape)
print("X_dev.shape = ",X_dev.shape)
```

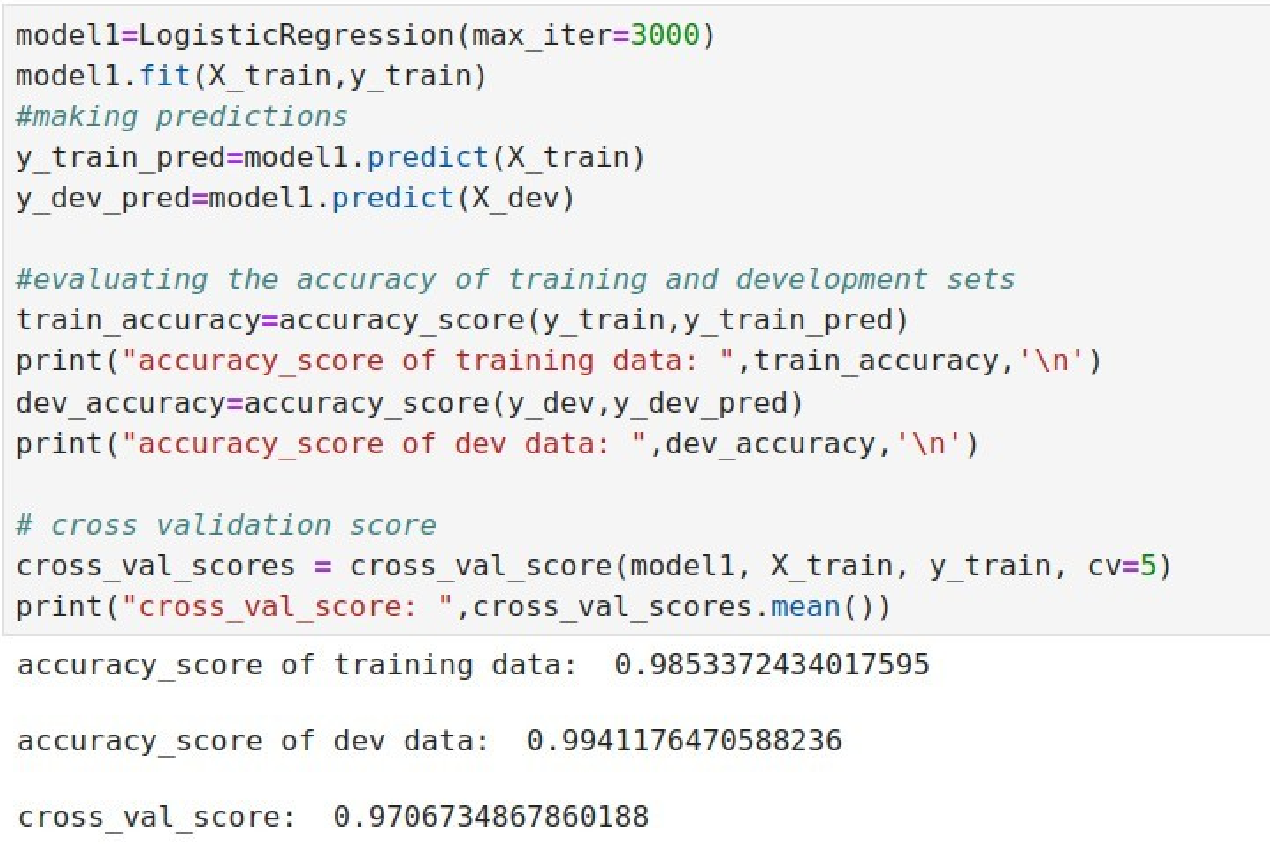
```
X_train.shape =  (341, 25)
X_test.shape =  (171, 25)
X_dev.shape =  (170, 25)
```

PS: fitting the scaler to the train data only is good practice to avoid data leakage

Model Selection :

In order to select the best model we have to try a few models and evaluate them on the dev set, the model with the highest accuracy will be evaluated on test set to verify its accuracy.

Logistic Regression:



```
model1=LogisticRegression(max_iter=3000)
model1.fit(X_train,y_train)
#making predictions
y_train_pred=model1.predict(X_train)
y_dev_pred=model1.predict(X_dev)

#evaluating the accuracy of training and development sets
train_accuracy=accuracy_score(y_train,y_train_pred)
print("accuracy_score of training data: ",train_accuracy,'\n')
dev_accuracy=accuracy_score(y_dev,y_dev_pred)
print("accuracy_score of dev data: ",dev_accuracy,'\n')

# cross validation score
cross_val_scores = cross_val_score(model1, X_train, y_train, cv=5)
print("cross_val_score: ",cross_val_scores.mean())
```

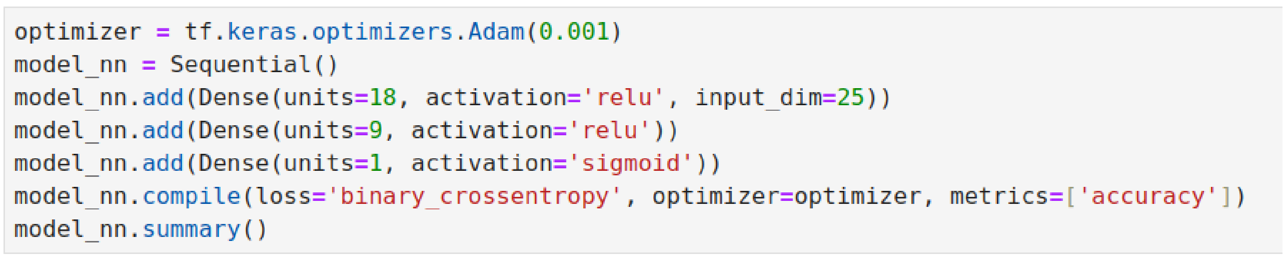
```
accuracy_score of training data:  0.9853372434017595

accuracy_score of dev data:  0.9941176470588236

cross_val_score:  0.9706734867860188
```

The model shows very promising results.

Deep Learning model :

Deep learning models are often associated with more complex and unstructured data like images so we expect the model to have a very high accuracy given the simplicity of our data.

We will build a three-layer model with a number of units deceasing as we go deeper and a ReLU activation. The output layer will have one unit with a sigmoid activation as have a binary classification problem.



```
optimizer = tf.keras.optimizers.Adam(0.001)
model_nn = Sequential()
model_nn.add(Dense(units=18, activation='relu', input_dim=25))
model_nn.add(Dense(units=9, activation='relu'))
model_nn.add(Dense(units=1, activation='sigmoid'))
model_nn.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=['accuracy'])
model_nn.summary()
```

```
Model: "sequential_17"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_51 (Dense)            (None, 18)                468

 dense_52 (Dense)            (None, 9)                 171

 dense_53 (Dense)            (None, 1)                 10

=================================================================
Total params: 649 (2.54 KB)
Trainable params: 649 (2.54 KB)
Non-trainable params: 0 (0.00 Byte)
```

After building the model, it is time o fit our training data to it:

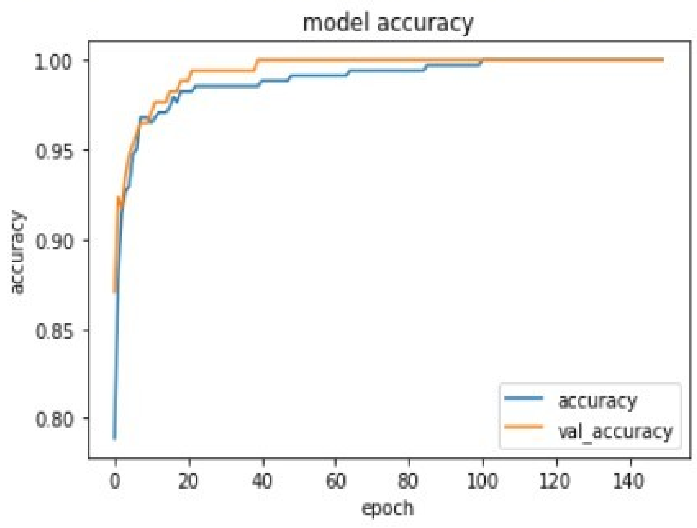
```
history = model_nn.fit(X_train, y_train, epochs=150, validation_data=(X_dev,y_dev))
```

```
Epoch 143/150
11/11 [==============================] - 0s 2ms/step - loss: 0.0041 - accuracy: 1.0000 - val_loss: 0.0024 - val_accuracy: 1.0000
Epoch 144/150
11/11 [==============================] - 0s 2ms/step - loss: 0.0040 - accuracy: 1.0000 - val_loss: 0.0023 - val_accuracy: 1.0000
Epoch 145/150
11/11 [==============================] - 0s 2ms/step - loss: 0.0039 - accuracy: 1.0000 - val_loss: 0.0022 - val_accuracy: 1.0000
Epoch 146/150
11/11 [==============================] - 0s 2ms/step - loss: 0.0038 - accuracy: 1.0000 - val_loss: 0.0021 - val_accuracy: 1.0000
Epoch 147/150
11/11 [==============================] - 0s 2ms/step - loss: 0.0038 - accuracy: 1.0000 - val_loss: 0.0024 - val_accuracy: 1.0000
Epoch 148/150
11/11 [==============================] - 0s 3ms/step - loss: 0.0036 - accuracy: 1.0000 - val_loss: 0.0022 - val_accuracy: 1.0000
Epoch 149/150
11/11 [==============================] - 0s 3ms/step - loss: 0.0036 - accuracy: 1.0000 - val_loss: 0.0021 - val_accuracy: 1.0000
Epoch 150/150
11/11 [==============================] - 0s 2ms/step - loss: 0.0035 - accuracy: 1.0000 - val_loss: 0.0020 - val_accuracy: 1.0000
```

Let’s plot the accuracy and loss of the training and validation sets:

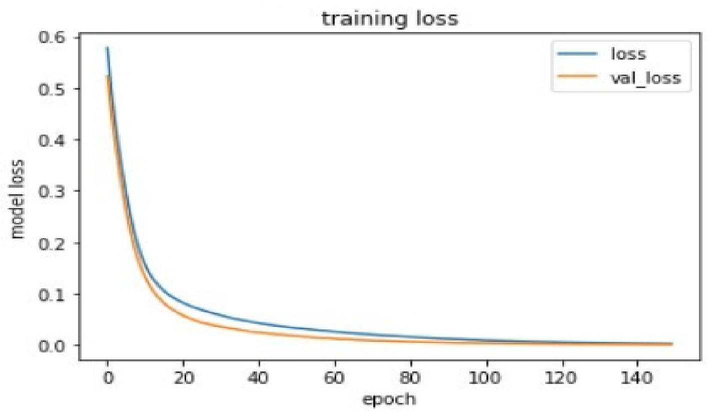
```
print(history.history.keys())
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['accuracy','val_accuracy'], loc='lower right')
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
<matplotlib.legend.Legend at 0x7f8bbba73430>
```



```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.xlabel('epoch')
plt.ylabel('model loss')
plt.title('training loss')
plt.legend(['loss','val_loss'], loc='upper right')
```

```
<matplotlib.legend.Legend at 0x7f8bbbad20e0>
```



The results could not be better! We have our perfect model, but for the sake of curiosity let’s try one more additional model.

XGBoost:

```
model_xgb = xgb.XGBClassifier(n_estimators = 15)
model_xgb.fit(X_train, y_train)
train_pred_xgb = model_xgb.predict(X_train)
dev_pred_xgb = model_xgb.predict(X_dev)
train_accuracy_xgb = accuracy_score(y_train, train_pred_xgb)
dev_accuracy_xgb = accuracy_score(y_dev, dev_pred_xgb)
print("accuracy_score of train set = ",train_accuracy_xgb)
print ("accuracy_score of dev set = ",dev_accuracy_xgb)
```

```
accuracy_score of train set =  1.0
accuracy_score of dev set =  1.0
```

Similar to the NN model, the results we got with XGBoost are oustanding.

Final Decision:

The candidates for the best model are the NN model and the XGBoost model. One final step would be to evaluate them on the test set:

```
test_pred_xgb = model_xgb.predict(X_test)
test_accuracy_xgb = accuracy_score(y_test, test_pred_xgb)
print('accuracy_score of test set on xgb model = ',test_accuracy_xgb)
model_nn.evaluate(X_test, y_test)
```

```
accuracy_score of test set on xgb model =  1.0
6/6 [==============================] - 0s 727us/step - loss: 0.0047 - accuracy: 1.0000
[0.004694061353802681, 1.0]
```

It seems like there is no tie break, so both models will do fine.