School of Computer Science Engineering and Information Systems
Winter Semester –2023-24

B. Tech IT – Capstone Project
2nd Review

Register Number	20BIT0403
Student Name	Abhishek K B
Project Domain (Capstone Project)	Machine Learning
Project Title (Capstone Project)	Systemic Lupus Erythematosus Classification using Machine Learning Techniques
Project Guide Name	Dr. RAGHAVAN R
Project Reviewers	Prof. Hari Ram Vishwakarma Prof. Pradeepa M
Date of Review-2	04th Apr 2024

1. Proposed Methodology & Architecture

The dataset for this machine learning study on Systemic lupus erythematosus classification was gathered from the NCBI portal using the GEO accession number GSE65391.

Before the data is sent into the ML model, it is pre-processed. The mean values of that feature column are used in place of null values. Label Encoder is used to transform categorical data into numerical data, and Standard Scaler is used to standardize the data.

Genetic algorithms are used for feature selection from the set of 88 features. Then, this data was divided into train and test sets.

A multi-layer perceptron is constructed and trained using the features that the PCA has chosen. After that, the model was tested using test data in order to assess its performance using a variety of metrics, including accuracy and F-score.

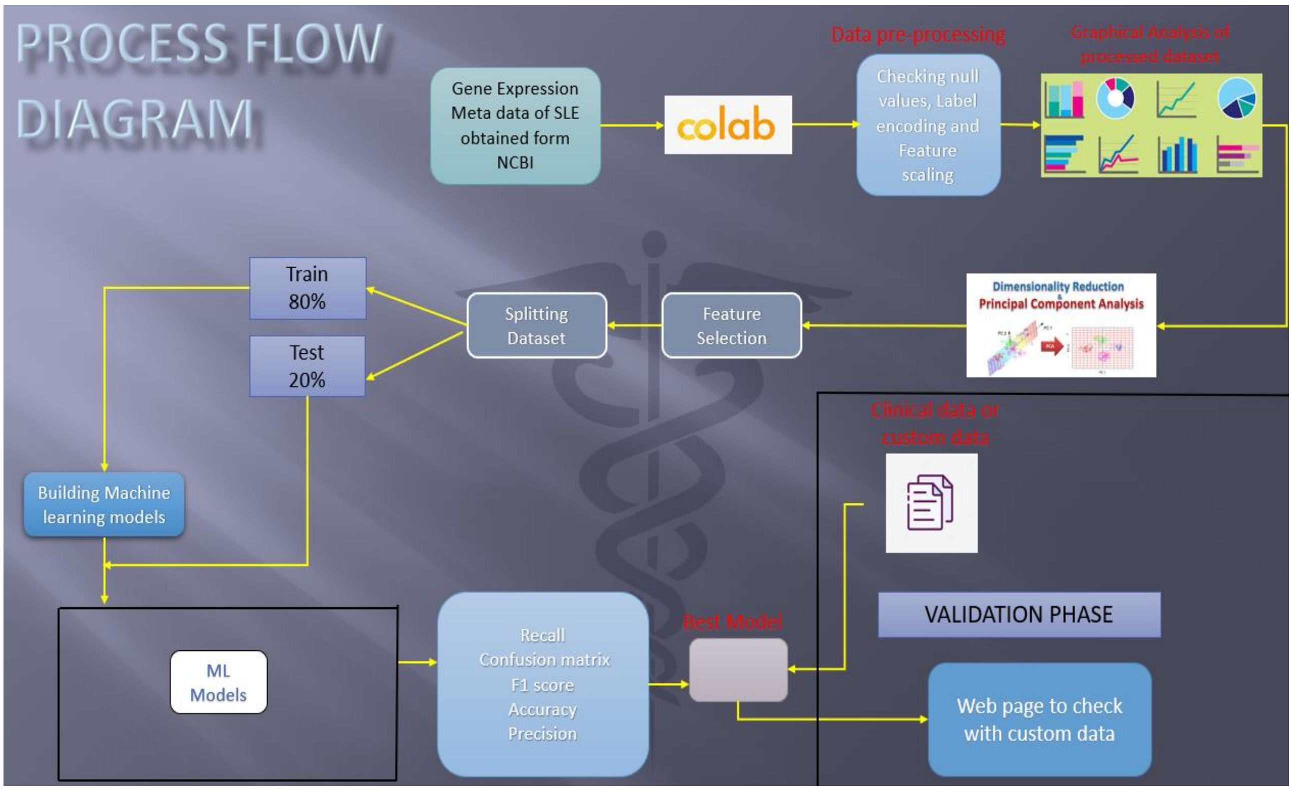

PROCESS FLOW DIAGRAM
Gene Expression Meta data of SLE obtained form NCBI
colab
Data pre-processing
Checking null values, Label encoding and Feature scaling
Graphical Analysis of processed dataset
Dimensionality Reduction Principal Component Analysis
Feature Selection
Splitting Dataset
Train 80%
Test 20%
Building Machine learning models
ML Models
Recall Confusion matrix F1 score Accuracy Precision
Best Model
Clinical data or custom data
VALIDATION PHASE
Web page to check with custom data

Fig1. Process Flow diagram (It is more suitable for this project than Architecture diagram)

2. Complete Design & Module Description

Dataset features:

1. Accession: Unique identifier for each entry.

2. Title: Title associated with the entry.

3. Sourcename: Name of the data source.

4. Batch: Batch number.

5. Batch_replicate: Indicates if the entry is a replicate in the batch.

6. Subject: Identifier for the subject associated with the entry.

7. Visit: Visit number.

8. Set: Indicates the set the entry belongs to.

9. Visit_count: Count of visits.

10. Cumulative_time: Cumulative time associated with the entry.

11. Days_since_diagnosis: Number of days since diagnosis.

12. Days_since_last_visit: Number of days since the last visit.

13. Days_between_diagnosis_and_last_visit: Number of days between diagnosis and last visit.

14. Gender: Gender of the subject.

15. Race: Race of the subject.

16. Age: Age of the subject.

17. Biopsy_history: History of biopsy.

18. Days_since_kidney_biopsy: Number of days since kidney biopsy.

19. Wbc: White blood cell count.

20. Neutrophil_count: Neutrophil count.

21. Lymphocyte_count: Lymphocyte count.

22. Monocyte_count: Monocyte count.

23. Neutrophil_percent: Neutrophil percentage.

24. Lymphocyte_percent: Lymphocyte percentage.

25. Monocyte_percent: Monocyte percentage.

26. Platelet_count: Platelet count.

27. Esr: Erythrocyte sedimentation rate.

28. Hgb: Hemoglobin level.

29. Hct: Hematocrit level.

30. Mcv: Mean corpuscular volume.

31. Mch: Mean corpuscular hemoglobin.

32. Mchc: Mean corpuscular hemoglobin concentration.

33. Rdw: Red cell distribution width.

34. Mpv: Mean platelet volume.

35. Cr: Creatinine level.

36. Alb: Albumin level.

37. Ds_dna: Double-stranded DNA level.

38. C3: Complement C3 level.

39. C4: Complement C4 level.

40. Ast: Aspartate aminotransferase level.

41. Alt: Alanine aminotransferase level.

42. Ald: Aldehyde dehydrogenase level.

43. Ldh: Lactate dehydrogenase level.

44. Steroid_iv_category: Category for intravenous steroid treatment.

45. Cyclophosphamide_category: Category for cyclophosphamide treatment.

46. Oral_steroids_category: Category for oral steroid treatment.

47. Mycophenolate_category: Category for mycophenolate treatment.

48. Hydroxychloroquine_category: Category for hydroxychloroquine treatment.

49. Metotrexate_category: Category for methotrexate treatment.

50. Nsaid_category: Category for nonsteroidal anti-inflammatory drug treatment.

51. Asa_category: Category for acetylsalicylic acid (aspirin) treatment.

52. Treatment: Type of treatment.

53. Treatment_lmm1: Another type of treatment.

54. Sledai: Systemic Lupus Erythematosus Disease Activity Index.

55. Disease_activity: Activity level of the disease.

56. Mdg: Mean disease duration.

57. Seizure: Presence of seizures.

58. Psychosis: Presence of psychosis.

59. Organic_brain_syndrome: Presence of organic brain syndrome.

60. Visual_disturbance: Presence of visual disturbances.

61. Cranial_nerve_disorder: Presence of cranial nerve disorders.

62. Lupus_headache: Presence of lupus headache.

63. Cva: Presence of cerebrovascular accident (stroke).

64. Vasculitis: Presence of vasculitis.

65. Arthritis: Presence of arthritis.

66. Myositis: Presence of myositis.

67. Urinary_casts: Presence of urinary casts.

68. Hematuria: Presence of hematuria.

69. Proteinuria: Presence of proteinuria.

70. Pyuria: Presence of pyuria.

71. New_rash: Presence of new rash.

72. Alopecia: Presence of alopecia.

73. Mucosal_ulcers: Presence of mucosal ulcers.

74. Pleurisy: Presence of pleurisy.

75. Pericarditis: Presence of pericarditis.

76. Low_complement: Presence of low complement levels.

77. Increased_dna_binding: Presence of increased DNA binding.

78. Fever: Presence of fever.

79. Thrombocytopenia: Presence of thrombocytopenia.

80. Leukopenia: Presence of leukopenia.

81. Renal: Presence of renal issues.

82. Musculoskeletal: Presence of musculoskeletal issues.

83. Serology: Presence of serological issues.

84. Sledai_component_class: Classification of SLEDAI components.

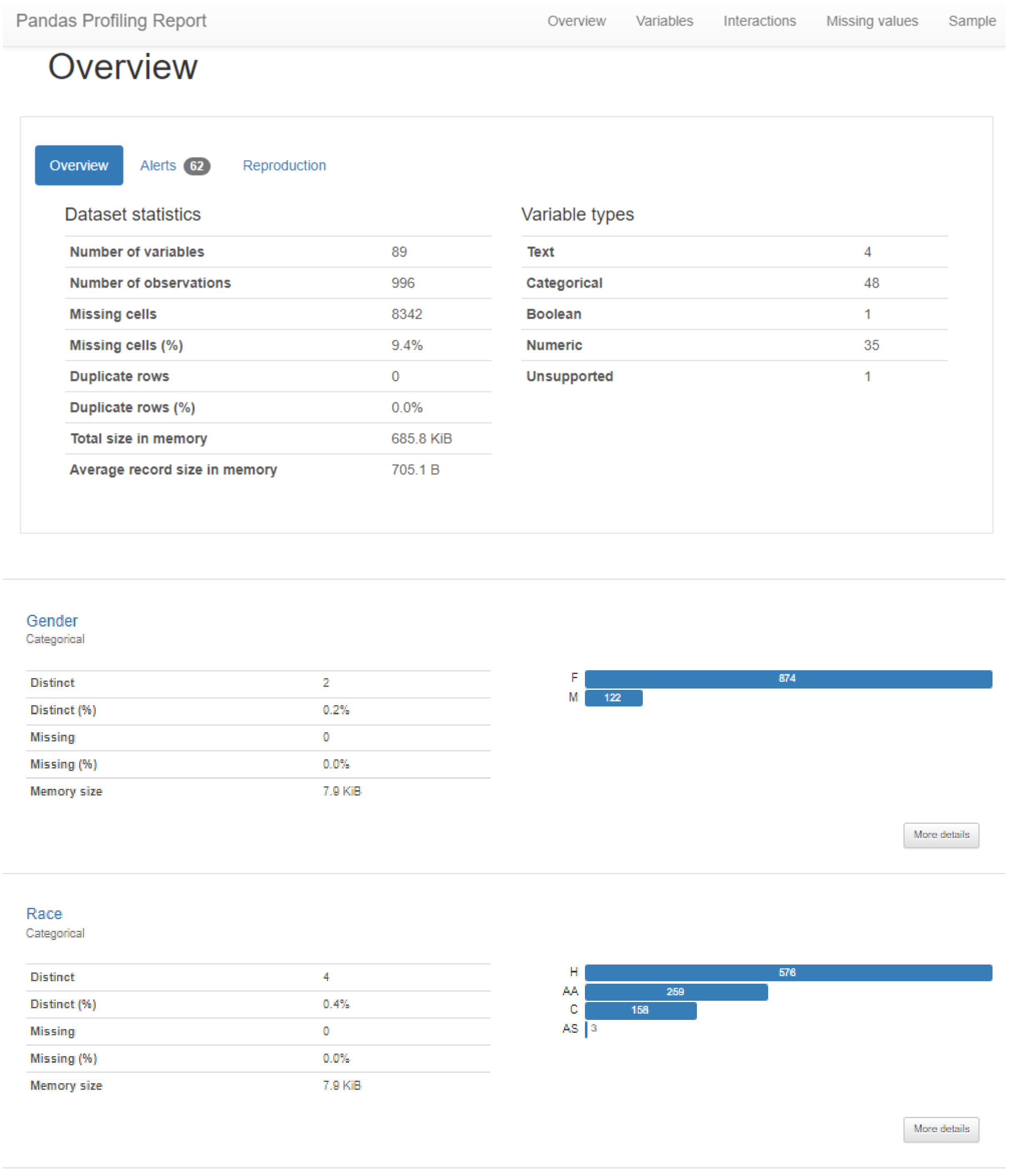
85. Sledaic_lmm2: Another classification for SLEDAI components.

86. Nephritis_class: Classification of nephritis.

87. Neph_treat_lmm3: Another classification for nephritis treatment.

88. Diseasestate: Disease state.

Data Visualizations:



Overview

Dataset statistics

Number of variables	89
Number of observations	996
Missing cells	8342
Missing cells (%)	9.4%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	685.8 KiB
Average record size in memory	705.1 B

Variable types

Text	4
Categorical	48
Boolean	1
Numeric	35
Unsupported	1

Gender
Categorical

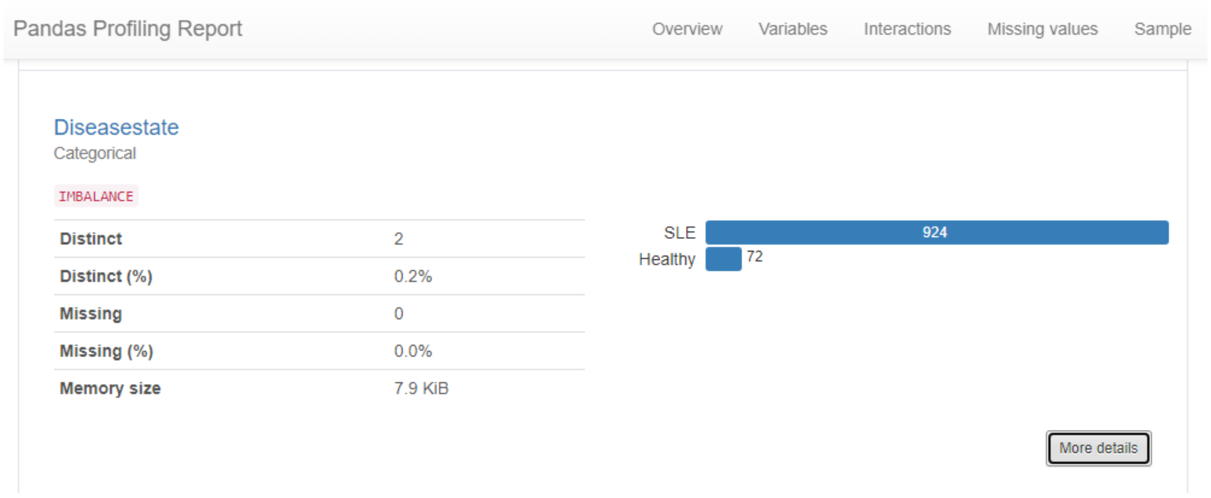
Distinct	2
Distinct (%)	0.2%
Missing	0
Missing (%)	0.0%
Memory size	7.9 KiB

F	874
M	122

Race
Categorical

Distinct	4
Distinct (%)	0.4%
Missing	0
Missing (%)	0.0%
Memory size	7.9 KiB

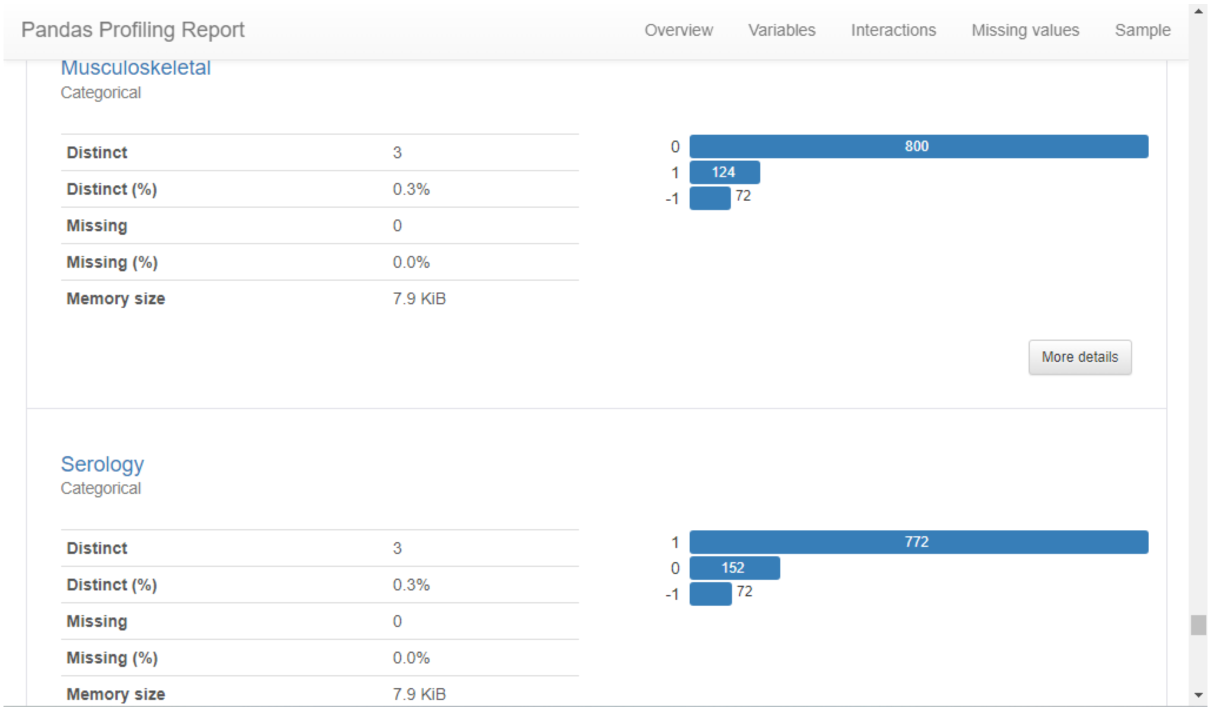
H	576
AA	259
C	158
AS	3



Diseasestate
Categorical

IMBALANCE

Distinct	2
Distinct (%)	0.2%
Missing	0
Missing (%)	0.0%
Memory size	7.9 KiB

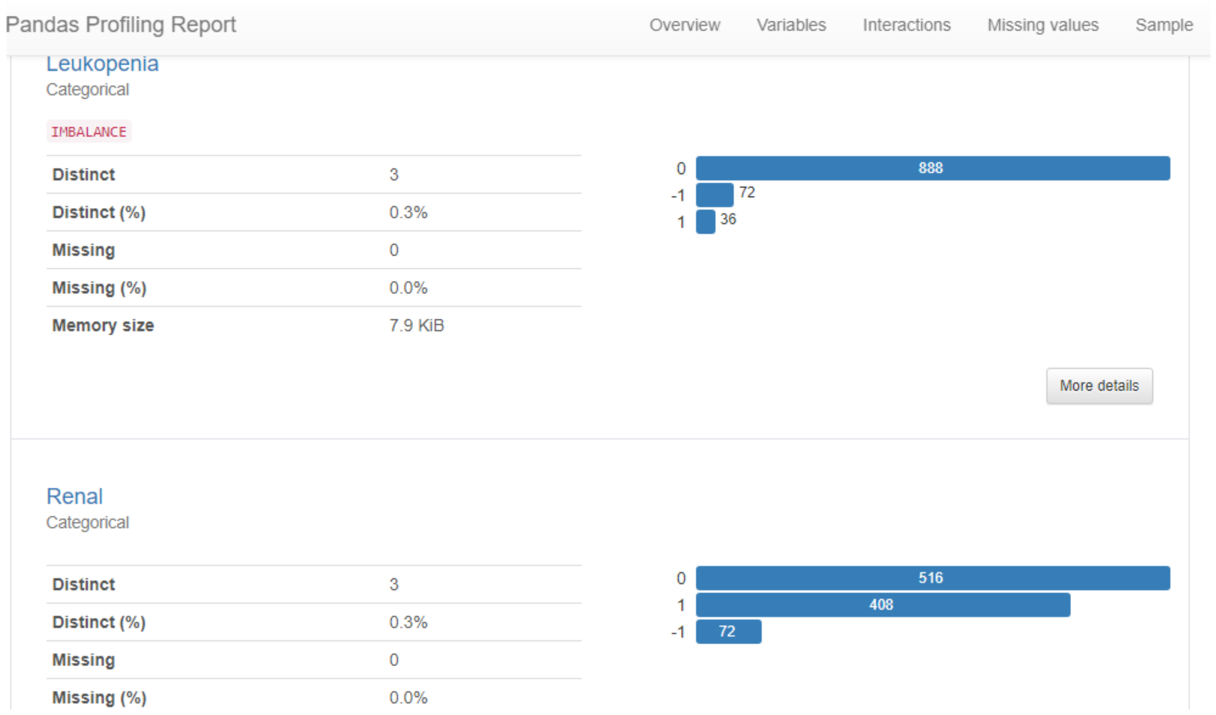


Musculoskeletal
Categorical

Distinct	3
Distinct (%)	0.3%
Missing	0
Missing (%)	0.0%
Memory size	7.9 KiB

Serology
Categorical

Distinct	3
Distinct (%)	0.3%
Missing	0
Missing (%)	0.0%
Memory size	7.9 KiB



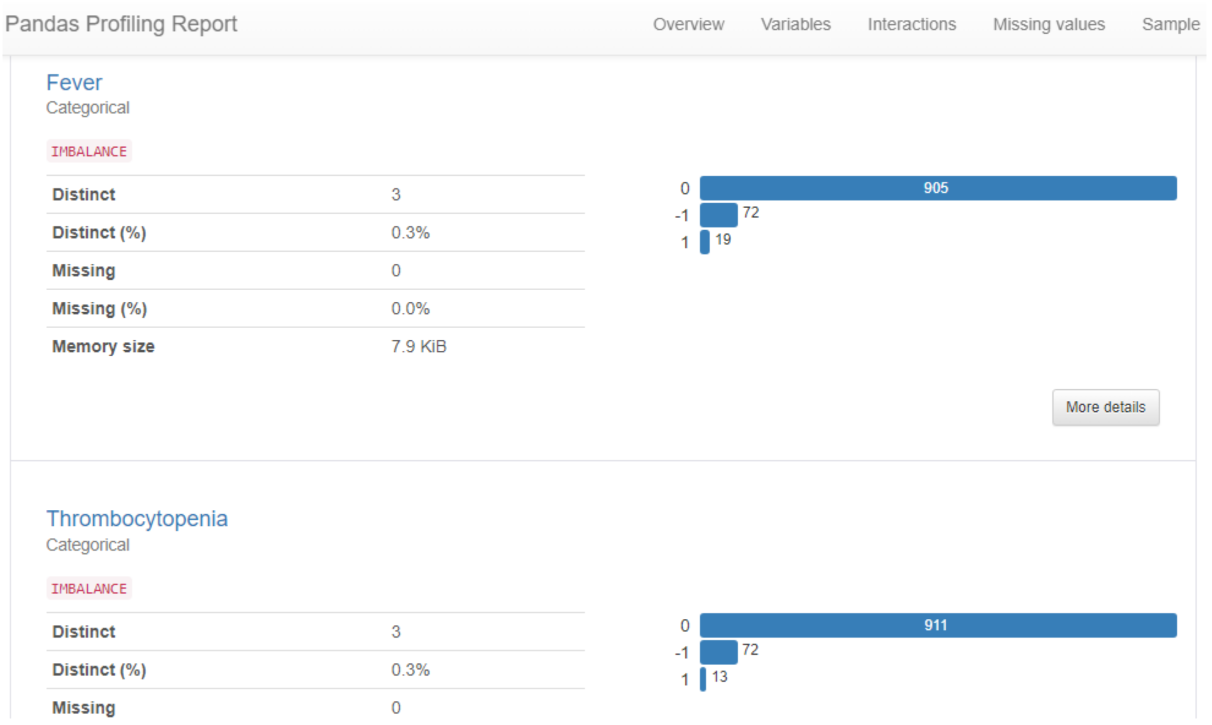
Leukopenia
Categorical

IMBALANCE

Distinct	3
Distinct (%)	0.3%
Missing	0
Missing (%)	0.0%
Memory size	7.9 KiB

Renal
Categorical

Distinct	3
Distinct (%)	0.3%
Missing	0
Missing (%)	0.0%



Fever
Categorical

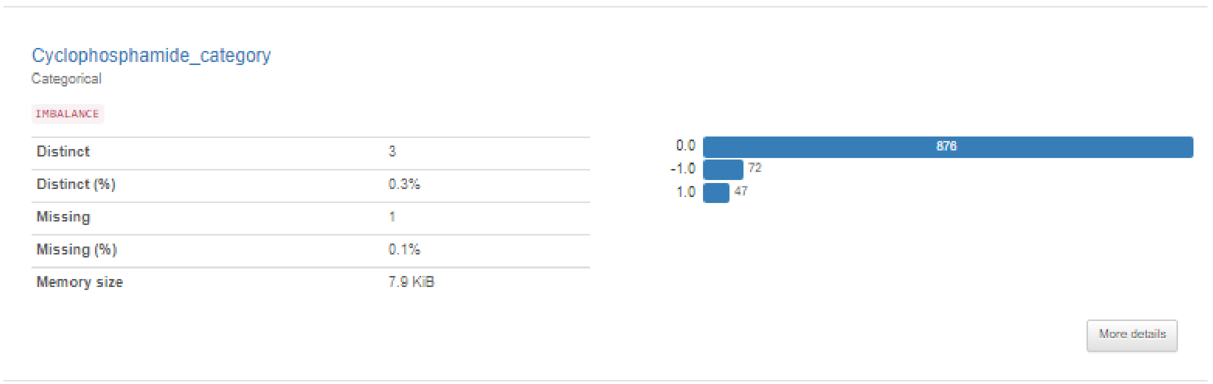
IMBALANCE

Distinct	3
Distinct (%)	0.3%
Missing	0
Missing (%)	0.0%
Memory size	7.9 KiB

Thrombocytopenia
Categorical

IMBALANCE

Distinct	3
Distinct (%)	0.3%
Missing	0

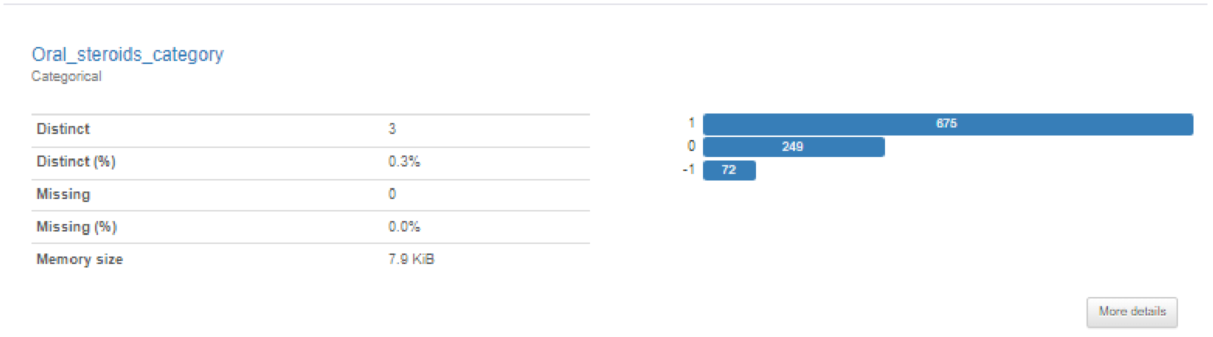


Cyclophosphamide_category
Categorical

IMBALANCE

Distinct	3
Distinct (%)	0.3%
Missing	1
Missing (%)	0.1%
Memory size	7.9 KiB

0.0	876
-1.0	72
1.0	47



Oral_steroids_category
Categorical

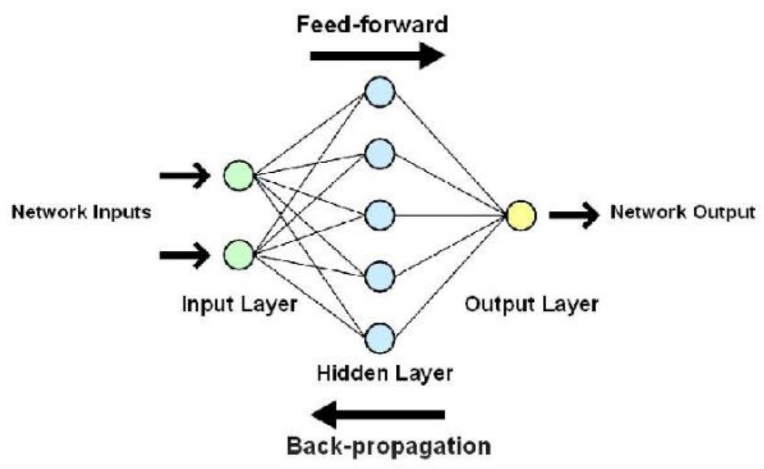
Distinct	3
Distinct (%)	0.3%
Missing	0
Missing (%)	0.0%
Memory size	7.9 KiB

1	675
0	249
-1	72

Module description:

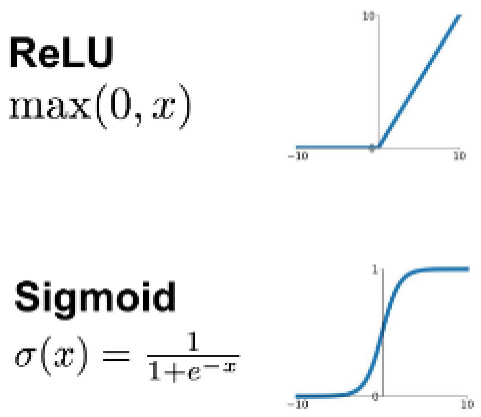
1. **Data Loading and Preprocessing:**
   - Data was loaded from a CSV file using Pandas.
   - Missing values are filled with mean values.
   - Categorical columns are label encoded using Scikit-learn's **LabelEncoder**.
2. **Data Visualization:**
   - Seaborn and Matplotlib are used for data visualization. For example, a joint plot was created to visualize the relationship between age and disease state. This gave the probability distribution of the disease across different age groups.
3. **Dimensionality Reduction (PCA):**
   - Principal Component Analysis (PCA) is applied to reduce the dimensionality of the data.
   - Standardization is performed using Scikit-learn's **StandardScaler**.
   - PCA was used to identify the principal components that explain the most variance in the data.
4. **Feature Selection:**
   - Feature loadings from PCA are analyzed to select the most significant features.
5. **Model Training and Evaluation:**

   By placing the samples in the SLE and Healthy classifications, respectively, they are classified. A individual with the class label "SLE" has the disease; a person with the class label "Healthy" is in good health. To classify, one must use the SLE classification model. Our proposed method uses ANN to classify the data. For the purpose of predicting SLE, a two-layered Multi-Layer Feedforward Neural Network (MLFNN), also known as a Multi-Layer Perceptron (MLP), is chosen.



(image source: analytics vidya)

Adam is the optimization algorithm that is employed. Hyperparameters in artificial neural networks (ANNs), such as the activation function, learning rate, number of hidden layers, and number of neurons in each layer, are crucial and must be adjusted to improve classification performance.

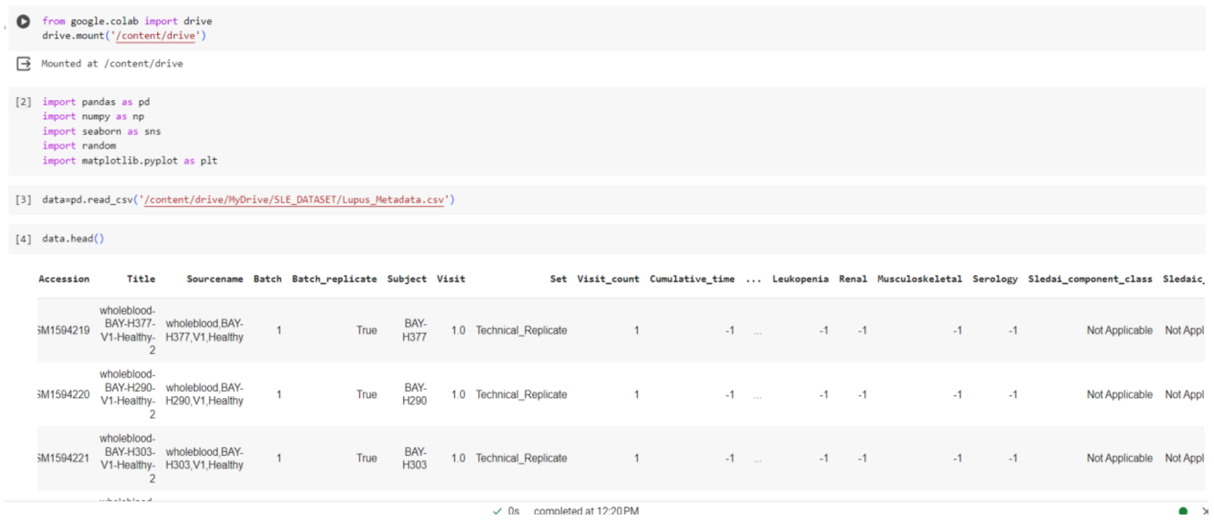


"ReLu" and "Sigmoid" are the activation functions that are employed. The ReLu function is used by the hidden layers, and the sigmoid function is used by the output layer. The loss function utilized is binary_crossentropy because the SLE classification is binary. In order to avoid overfitting, drop-out layers are additionally inserted in between the hidden layers. Dense layers employ an L2 kernel regularizer with a rate of 0.01 to enhance the ANN model's generalization by preventing overfitting.

PCA-ANN Model:

```
 Start
Step 1: Build ANN Model ()
        MLP with two hidden layers, dropout layers, and l2 regularization.
Step 2: Split the data into training and testing parts in the ratio of 80:20
Step 3: Train the neural network using the features selected by PCA.
Loop 1: Until stopping criteria met
Forward Propagation ()
         $h \leftarrow \mathrm{ReLu}(wh \times inp)$
         Dropout ()
         $o \leftarrow \mathrm{Sigmoid}(w0 \times h)$
         Loss = BinaryCrossEntropy(o, true_labels)
Backward Propagation ()
         $\Delta 0 \leftarrow o$ - true_labels
         $\Delta h = f\,1\,(h) \times$ (weight_output_transpose $\times\, \Delta 0$)
         $W_0(new) \leftarrow$ update weight_output
         $W_h(new) \leftarrow$ update weight_hidden
Apply L2 regularization to updated weights
if (stopping criteria met)
Stop training
End Loop
Step 4: Prediction ()
feed the testing data into the trained ANN model
y ← trained_model.predict(x_test,y_test)
if ($y >= 0.5$)
  "SLE"
else
  "Healthy"
Stop
```

3. Implementation:



```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

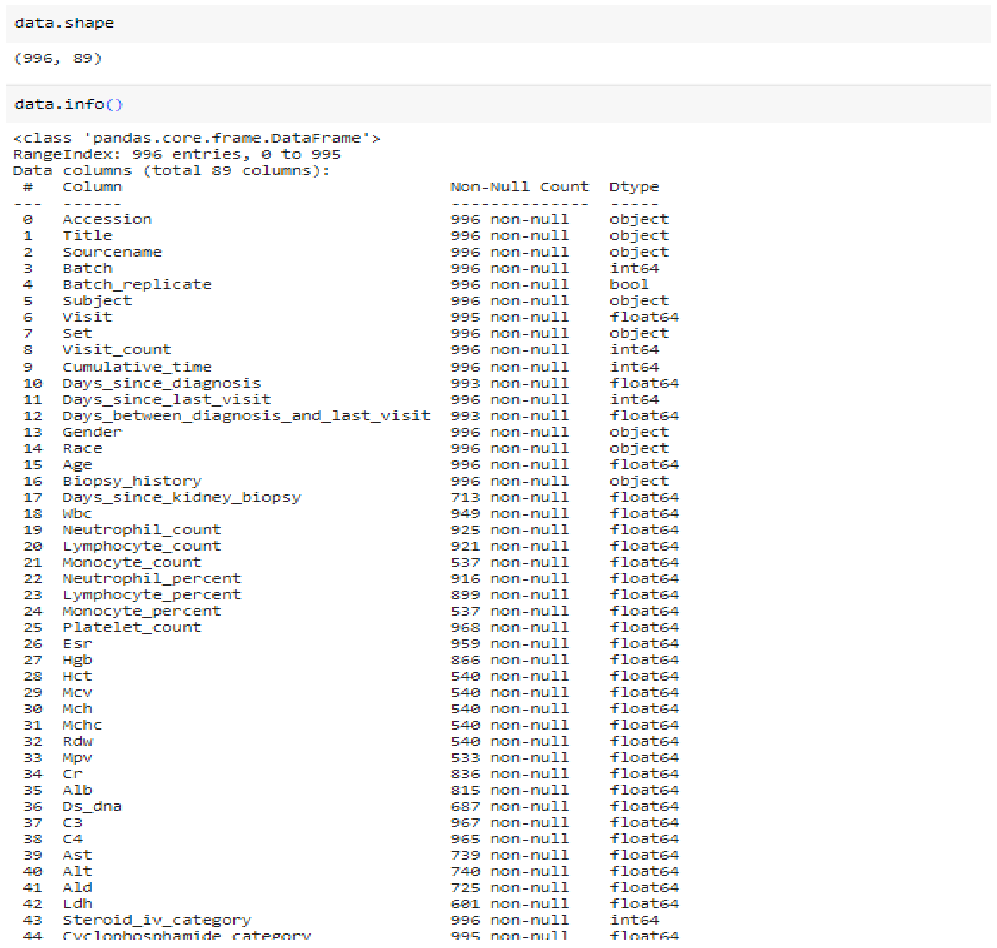
```
[2] import pandas as pd
    import numpy as np
    import seaborn as sns
    import random
    import matplotlib.pyplot as plt
```

```
[3] data=pd.read_csv('/content/drive/MyDrive/SLE_DATASET/Lupus_Metadata.csv')
```

```
[4] data.head()
```

Accession	Title	Sourcename	Batch	Batch_replicate	Subject	Visit	Set	Visit_count	Cumulative_time	...	Leukopenia	Renal	Musculoskeletal	Serology	Sledai_component_class	Sledaic_
SM1594219	wholeblood-BAY-H377-V1-Healthy-2	wholeblood,BAY-H377,V1,Healthy	1	True	BAY-H377	1.0	Technical_Replicate	1	-1	...	-1	-1	-1	-1	Not Applicable	Not Appl
SM1594220	wholeblood-BAY-H290-V1-Healthy-2	wholeblood,BAY-H290,V1,Healthy	1	True	BAY-H290	1.0	Technical_Replicate	1	-1	...	-1	-1	-1	-1	Not Applicable	Not Appl
SM1594221	wholeblood-BAY-H303-V1-Healthy-2	wholeblood,BAY-H303,V1,Healthy	1	True	BAY-H303	1.0	Technical_Replicate	1	-1	...	-1	-1	-1	-1	Not Applicable	Not Appl

✓ 0s completed at 12:20 PM

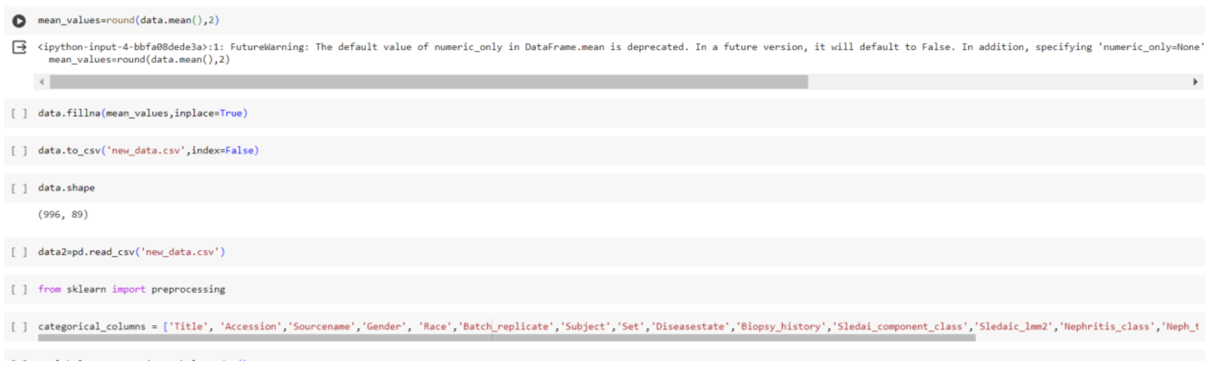


```
data.shape
```

```
(996, 89)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 996 entries, 0 to 995
Data columns (total 89 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   Accession                             996 non-null    object
 1   Title                                 996 non-null    object
 2   Sourcename                            996 non-null    object
 3   Batch                                 996 non-null    int64
 4   Batch_replicate                       996 non-null    bool
 5   Subject                               996 non-null    object
 6   Visit                                 995 non-null    float64
 7   Set                                   996 non-null    object
 8   Visit_count                           996 non-null    int64
 9   Cumulative_time                       996 non-null    int64
 10  Days_since_diagnosis                  993 non-null    float64
 11  Days_since_last_visit                 996 non-null    int64
 12  Days_between_diagnosis_and_last_visit 993 non-null    float64
 13  Gender                                996 non-null    object
 14  Race                                  996 non-null    object
 15  Age                                   996 non-null    float64
 16  Biopsy_history                        996 non-null    object
 17  Days_since_kidney_biopsy              713 non-null    float64
 18  Wbc                                   949 non-null    float64
 19  Neutrophil_count                      925 non-null    float64
 20  Lymphocyte_count                      921 non-null    float64
 21  Monocyte_count                        537 non-null    float64
 22  Neutrophil_percent                    916 non-null    float64
 23  Lymphocyte_percent                    899 non-null    float64
 24  Monocyte_percent                      537 non-null    float64
 25  Platelet_count                        968 non-null    float64
 26  Esr                                   959 non-null    float64
 27  Hgb                                   866 non-null    float64
 28  Hct                                   540 non-null    float64
 29  Mcv                                   540 non-null    float64
 30  Mch                                   540 non-null    float64
 31  Mchc                                  540 non-null    float64
 32  Rdw                                   540 non-null    float64
 33  Mpv                                   533 non-null    float64
 34  Cr                                    836 non-null    float64
 35  Alb                                   815 non-null    float64
 36  Ds_dna                                687 non-null    float64
 37  C3                                    967 non-null    float64
 38  C4                                    965 non-null    float64
 39  Ast                                   739 non-null    float64
 40  Alt                                   740 non-null    float64
 41  Ald                                   725 non-null    float64
 42  Ldh                                   601 non-null    float64
 43  Steroid_iv_category                   996 non-null    int64
 44  Cyclophosphamide_category             995 non-null    float64
```



```
mean_values=round(data.mean(),2)
```

```
<ipython-input-4-bbfa08dede3a>:1: FutureWarning: The default value of numeric_only in DataFrame.mean is deprecated. In a future version, it will default to False. In addition, specifying 'numeric_only=None'
  mean_values=round(data.mean(),2)
```

```
[ ] data.fillna(mean_values,inplace=True)
```

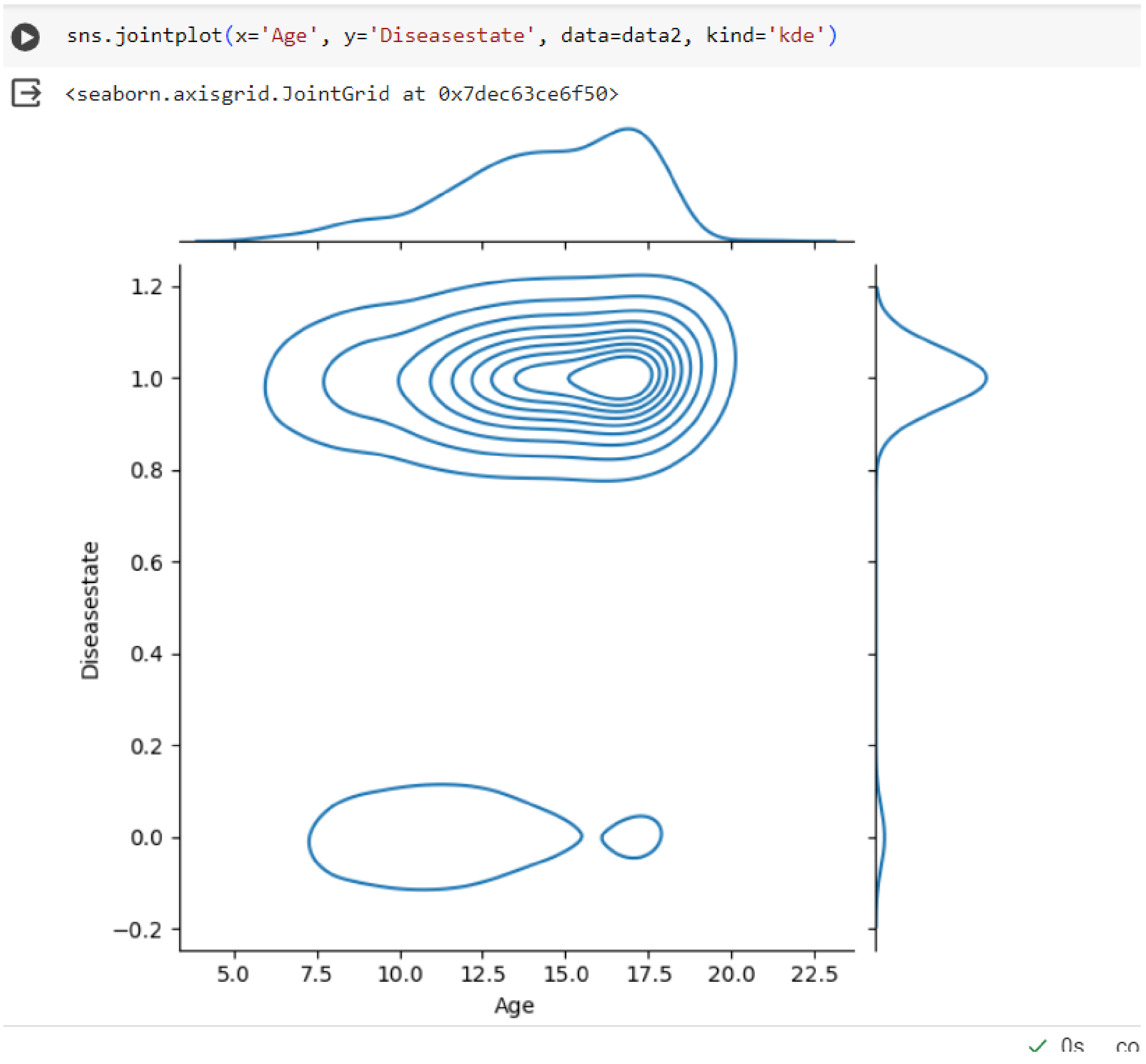
```
[ ] data.to_csv('new_data.csv',index=False)
```

```
[ ] data.shape

(996, 89)
```

```
[ ] data2=pd.read_csv('new_data.csv')
```

```
[ ] from sklearn import preprocessing
```

```
[ ] categorical_columns = ['Title', 'Accession','Sourcename','Gender', 'Race','Batch_replicate','Subject','Set','Diseasestate','Biopsy_history','Sledai_component_class','Sledaic_lmm2','Nephritis_class','Neph_t
```



```
sns.jointplot(x='Age', y='Diseasestate', data=data2, kind='kde')
```

```
<seaborn.axisgrid.JointGrid at 0x7dec63ce6f50>
```

```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
```

```
[ ] df = pd.read_csv('f2_data.csv')
```

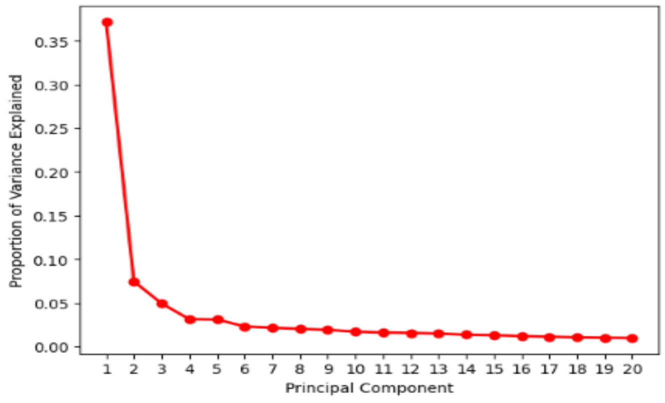
```
[ ] X = df.iloc[:,0:87]
    y = df.iloc[:,88]
```

```
[ ] scaler = StandardScaler()
    X_std = scaler.fit_transform(X)
    column_names = X.columns
    X_std_df = pd.DataFrame(X_std, columns=column_names)
    X_std_df.to_csv('standard.csv',index=False)
```

```
[ ] pca = PCA(n_components=20)
    pca.fit(X_std)
```

```
▾         PCA
PCA(n_components=20)
```

```
plt.plot(np.arange(1, pca.n_components_ + 1), pca.explained_variance_ratio_, 'ro-', linewidth=2)
plt.xlabel('Principal Component')
plt.ylabel('Proportion of Variance Explained')
plt.xticks(np.arange(1, pca.n_components_ + 1))
plt.show()
print(pca.components_)
```



```
[[-0.05439598 -0.08641829 -0.08464649 ...  0.10917065  0.00322186
   0.04207031]
 [-0.02254204  0.12264548  0.12377491 ... -0.23330977  0.13397865
   0.27647091]
 [ 0.09475205  0.01682018  0.01633956 ...  0.12207792 -0.23488731
  -0.06185049]
 ...
 [ 0.05512229 -0.009322   -0.00917967 ...  0.08068881  0.06982159
  -0.11739019]
 [ 0.09731092 -0.01101508 -0.01111658 ...  0.04650007 -0.18134051
  -0.0220272 ]
 [ 0.04365192 -0.00943034 -0.00987857 ... -0.0323098   0.02964416
   0.05217209]]
```

```
pca = PCA(n_components=6)
```

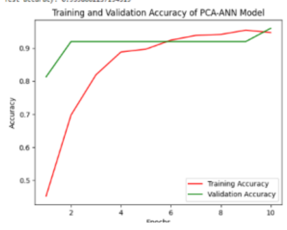
```
pca.fit(X_std)
```

```python
from sklearn import model_selection
df=pd.read_csv('f2_data.csv')
c1=['Ds_dna','Days_between_diagnosis_and_last_visit','Days_since_kidney_biopsy','Title','Cumulative_time','Accession'] #PCA
X_pca = np.array(X_std_df[c1])
y = np.array(df['Diseasestate'])
t_train,t_test,T_train,T_test=model_selection.train_test_split(X_pca,y,test_size=0.3,random_state=42)
```

```python
from keras.models import Sequential, load_model
from keras.utils import plot_model

from keras.layers import Dense, Dropout
from keras.regularizers import l2
from keras.optimizers import Adam
from keras.callbacks import EarlyStopping

def create_binary_model():
    model = Sequential()
    model.add(Dense(32, input_dim=6, activation='relu', kernel_regularizer=l2(0.01)))
    model.add(Dropout(0.4))
    model.add(Dense(16, activation='relu', kernel_regularizer=l2(0.01)))
    model.add(Dropout(0.4))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(loss='binary_crossentropy', optimizer=Adam(lr=0.0001), metrics=['accuracy'])
    return model

pca_model = create_binary_model()
plot_model(pca_model, to_file='model_architecture.png', show_shapes=True)


# add early stopping callback
early_stopping = EarlyStopping(monitor='val_loss', patience=5)

history = pca_model.fit(t_train, T_train, epochs=10, batch_size=32, validation_data=(t_test, T_test), callbacks=[early_stopping])
pca_model.save("Trained_model.h5")
# evaluate the model on the test set
loss, accuracy = pca_model.evaluate(t_test, T_test)
print("Test accuracy:", accuracy)

# Retrieve accuracy and loss values from the history object
accuracy = history.history['accuracy']
val_accuracy = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

# Plot accuracy
```

```python
# Display the loss plot
plt.show()
```

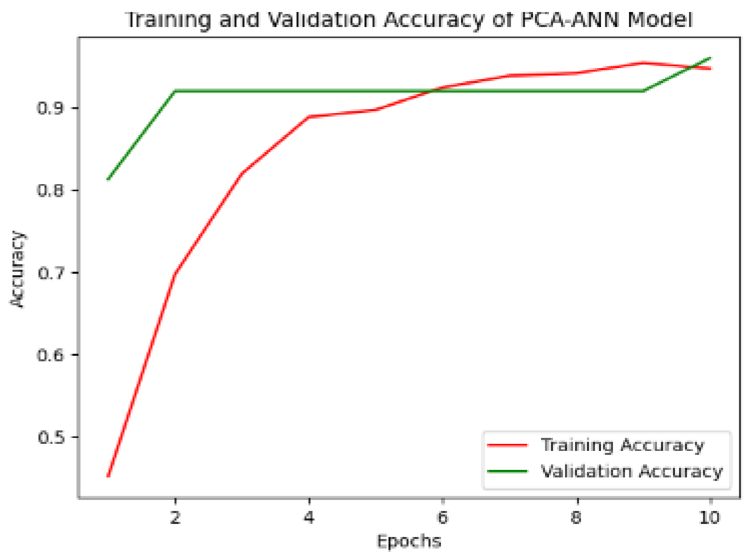
```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g.,tf.keras.optimizers.legacy.Adam.
Epoch 1/10
22/22 [==============================] - 2s 21ms/step - loss: 1.1143 - accuracy: 0.4519 - val_loss: 0.9230 - val_accuracy: 0.8127
Epoch 2/10
22/22 [==============================] - 0s 10ms/step - loss: 0.8992 - accuracy: 0.6973 - val_loss: 0.7710 - val_accuracy: 0.9197
Epoch 3/10
22/22 [==============================] - 0s 9ms/step - loss: 0.7644 - accuracy: 0.8192 - val_loss: 0.6698 - val_accuracy: 0.9197
Epoch 4/10
22/22 [==============================] - 0s 8ms/step - loss: 0.6912 - accuracy: 0.8881 - val_loss: 0.5869 - val_accuracy: 0.9197
Epoch 5/10
22/22 [==============================] - 0s 8ms/step - loss: 0.6183 - accuracy: 0.8967 - val_loss: 0.5186 - val_accuracy: 0.9197
Epoch 6/10
22/22 [==============================] - 0s 9ms/step - loss: 0.5426 - accuracy: 0.9240 - val_loss: 0.4633 - val_accuracy: 0.9197
Epoch 7/10
22/22 [==============================] - 0s 9ms/step - loss: 0.4833 - accuracy: 0.9383 - val_loss: 0.4172 - val_accuracy: 0.9197
Epoch 8/10
22/22 [==============================] - 0s 9ms/step - loss: 0.4599 - accuracy: 0.9412 - val_loss: 0.3790 - val_accuracy: 0.9197
Epoch 9/10
22/22 [==============================] - 0s 9ms/step - loss: 0.4114 - accuracy: 0.9541 - val_loss: 0.3482 - val_accuracy: 0.9197
Epoch 10/10
22/22 [==============================] - 0s 8ms/step - loss: 0.3870 - accuracy: 0.9469 - val_loss: 0.3213 - val_accuracy: 0.9599
 1/10 [==>...........................] - ETA: 0s - loss: 0.3011 - accuracy: 0.9688/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native
  saving_api.save_model(
10/10 [==============================] - 0s 4ms/step - loss: 0.3213 - accuracy: 0.9599
Test accuracy: 0.9598662257194519
```



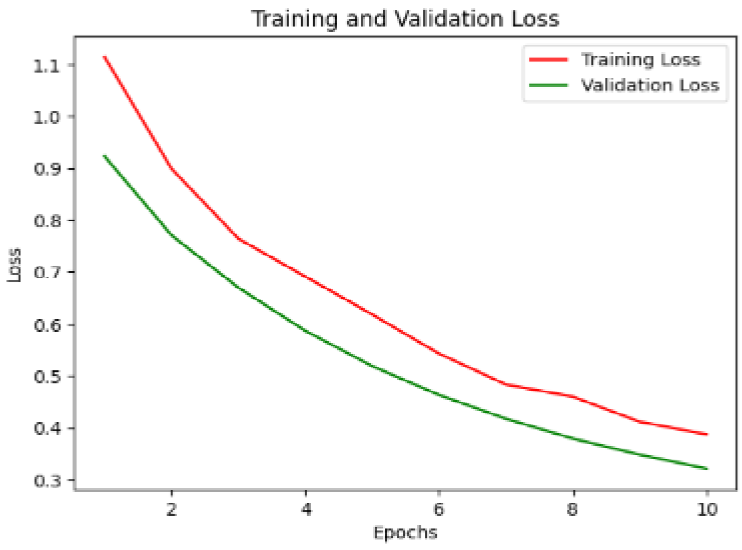
4. Testing, Performance metrics

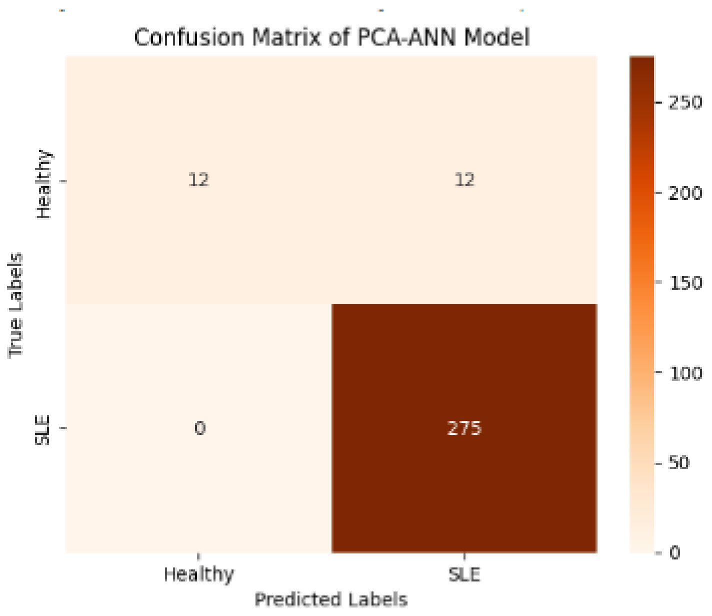
PCA-ANN, PCA was used to reduce the dimensionality of the complex dataset while retaining its essential information. As a result, 5 crucial features have been selected from the scree plot. Then, these features are utilized to train and test the ANN. This model achieved an accuracy of 96%.

Below diagrams depicts the PCA-ANN model's training and validation accuracy for each epoch. This graph shows the model's ability to generalize to new data. The hold-out validation dataset's validation curve provides a primary indication of how well the model generalizes. The number of training iterations is indicated by the epochs on the X-axis.



At the beginning of training, typically there is a rapid decrease in the training loss. This is because the ANN is learning to fit the data and reduce the error.





Algorithm	Accuracy	Precision	F1-Score
PCA-ANN	96%	95.81%	97.8%

Work in progress:

- Working on Genetic Algorithm +ANN.
- The current model is having target column with only two classes (Healthy and SLE). Working on feature construction to make this model to have some more classes which will help to identify sub diseases of SLE like lupus-nephritis etc.