



SURYA GROUP OF INSTITUTIONS
VIKRAVANDI-605652



NAAN MUDHALVAN PROJECT
AI BASED DIABETES PREDICTION
PHASE 2:INNOVATION

PRESENTED BY

NAME:K.KAVIRAJAN

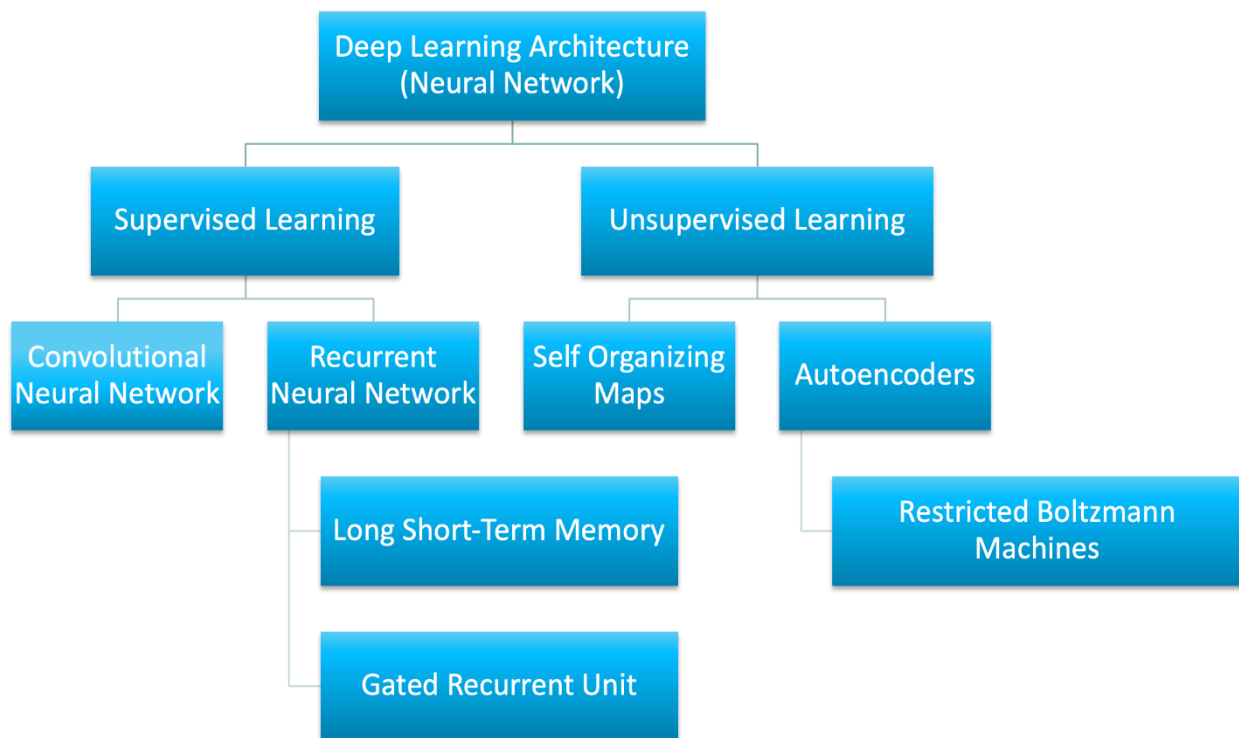
REG.NO:422221106011

DEPARTMENT:ECE 3rd year

INTRODUCTION:

We have implemented various methods or approaches to use our data systematically and in synchronized way for the purpose of the development of our model. Moreover the test plan is according to our model and can be helpful if we wants to make further improvements and developments to our model.

DEEP LEARNING ARCHITECTURE:



Importing libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Importing dataset

```
dataset = pd.read_csv('./input/diabetes-data-set/diabetes.csv')
```

Viewing the dataset, its dimensions, features and statistical Summary

```
dataset.head()
```

	Pregnancies	Glucose	BP	Skin thickness	Insulin	BMI	Diabetes pedigree function	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1

1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

dataset.shape

(768, 9)

dataset.info()

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

RangeIndex: 768 entries, 0 to 767

Data columns (total 9 columns):

Column Non-Null Count Dtype

0 Pregnancies 768 non-null int64

1 Glucose 768 non-null int64

2 BloodPressure 768 non-null int64

3 SkinThickness 768 non-null int64

4 Insulin 768 non-null int64

5 BMI 768 non-null float64

6 DiabetesPedigree

Function 768 non-null float64

7 Age 768 non-null int64

8 Outcome 768 non-null int64

dtypes: float64(2), int64(7)

memory usage: 54.1 KB

dataset.describe().T

count	mean	std	min	25%	50%	75%	max	
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.0000	6.00000	17.00
Glucose	768.0	120.894531	31.972618	0.000	99.00000	117.0000	140.25000	199.00
BloodPressure	768.0	69.105469	19.355807	0.000	62.00000	72.0000	80.00000	122.00
SkinThickness	768.0	20.536458	15.952218	0.000	0.00000	23.0000	32.00000	99.00
Insulin	768.0	79.799479	115.244002	0.000	0.00000	30.5000	127.25000	846.00
BMI	768.0	31.992578	7.884160	0.000	27.30000	32.0000	36.60000	67.10

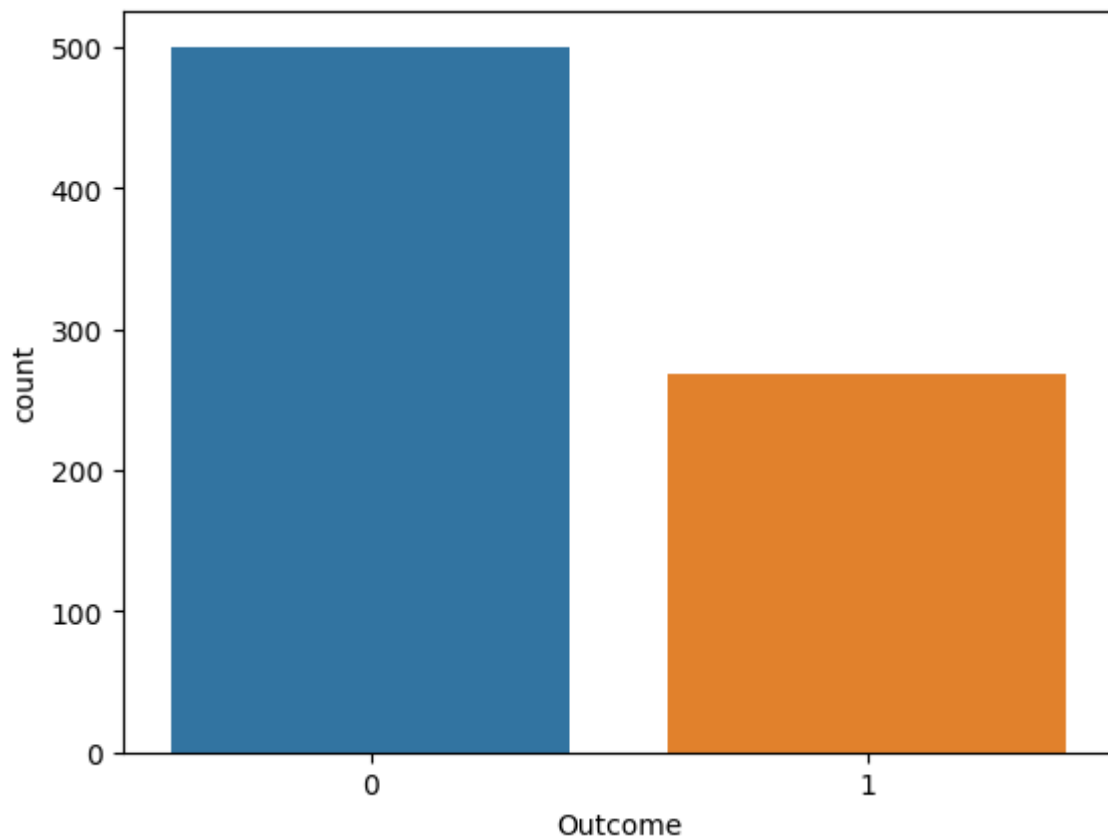
count	mean	std	min	25%	50%	75%	max	
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	0.3725	0.62625	2.42
Age	768.0	33.240885	11.760232	21.000	24.00000	29.0000	41.00000	81.00
Outcome	768.0	0.348958	0.476951	0.000	0.00000	0.0000	1.00000	1.00

Detecting null values

```
dataset.isnull().sum()
Pregnancies      0
Glucose           0
BloodPressure     0
SkinThickness     0
Insulin           0
BMI               0
DiabetesPedigreeFunction  0
Age               0
Outcome           0
dtype: int64
```

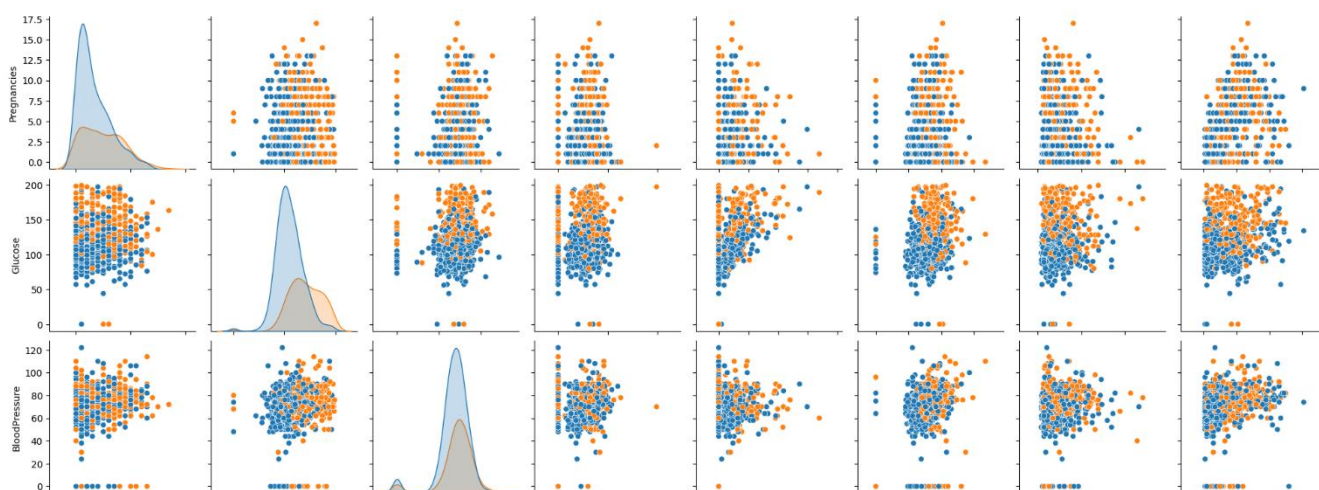
Data Visualization

```
sns.countplot(x = 'Outcome',data = dataset)
<Axes: xlabel='Outcome', ylabel='count'>
```



```
# Pairplot
sns.pairplot(data = dataset, hue = 'Outcome')
plt.show()
```

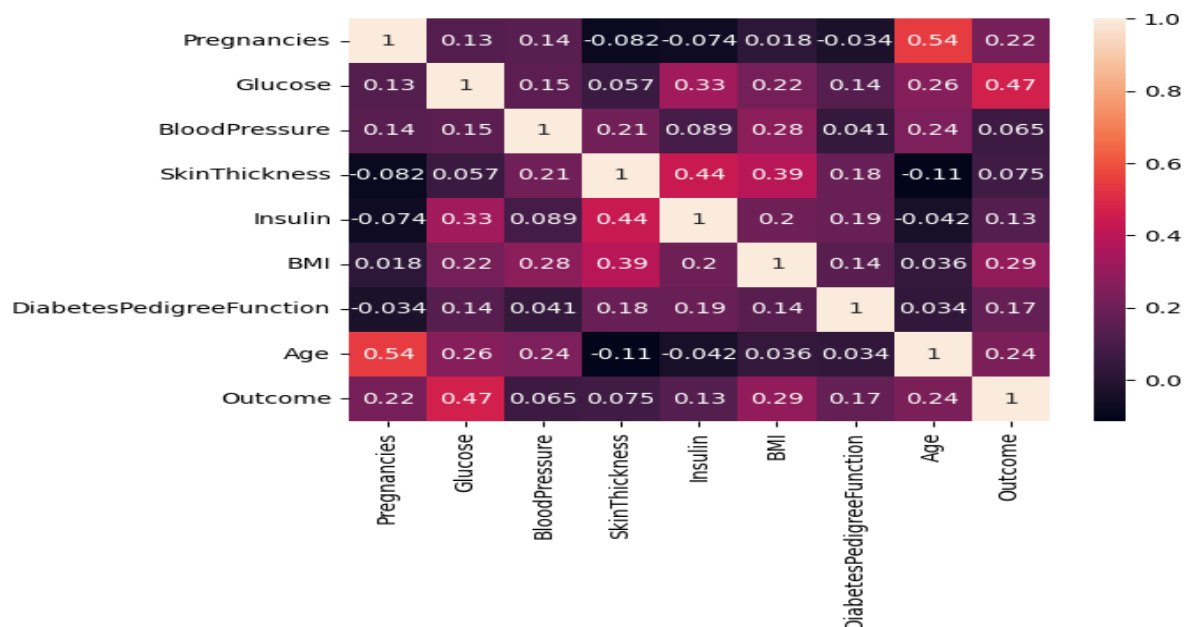
/opt/conda/lib/python3.10/site-packages/seaborn/axisgrid.py:118: UserWarning: The figure layout has changed to tight
self._figure.tight_layout(*args, **kwargs)





Heatmap

```
sns.heatmap(dataset.corr(), annot = True)
plt.show()
```



Processing the Data

Replacing zero values with NaN

```
dataset_new = dataset
```

```
dataset_new[["Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI"]] = dataset_new[["Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI"]].replace(0, np.NaN)
```

```

linkcode
# Count of NaN
dataset_new.isnull().sum()
Pregnancies      0
Glucose           5
BloodPressure     35
SkinThickness     227
Insulin           374
BMI               11
DiabetesPedigree
Function          0
Age              0
Outcome          0
dtype: int64

```

```

# Replacing NaN with mean values
dataset_new["Glucose"].fillna(dataset_new["Glucose"].mean(), inplace = True)
dataset_new["BloodPressure"].fillna(dataset_new["BloodPressure"].mean(), inplace = True)
dataset_new["SkinThickness"].fillna(dataset_new["SkinThickness"].mean(), inplace = True)
dataset_new["Insulin"].fillna(dataset_new["Insulin"].mean(), inplace = True)
dataset_new["BMI"].fillna(dataset_new["BMI"].mean(), inplace = True)

```

```

dataset_new.isnull().sum()
Pregnancies      0
Glucose           0
BloodPressure     0
SkinThickness     0
Insulin           0
BMI               0
DiabetesPedigree
Function          0
Age              0
Outcome          0
dtype: int64

```

Logistic Regression

```

y = dataset_new['Outcome']
X = dataset_new.drop('Outcome', axis=1)

```

```

# Splitting X and Y
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size = 0.20, random_state = 42, stratify = dataset_new['Outcome'])

```

```

from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, Y_train)
y_predict = model.predict(X_test)

```

```

from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, Y_train)

```

```
y_predict = model.predict(X_test)
```

/opt/conda/lib/python3.10/site-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

```
n_iter_i = _check_optimize_result(
```

```
y_predict
```

```
array([1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1,  
       0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0,  
       0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0,  
       1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
       0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1,  
       0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1,  
       0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0])
```

```
# Confusion matrix
```

```
from sklearn.metrics import confusion_matrix
```

```
cm = confusion_matrix(Y_test, y_predict)
```

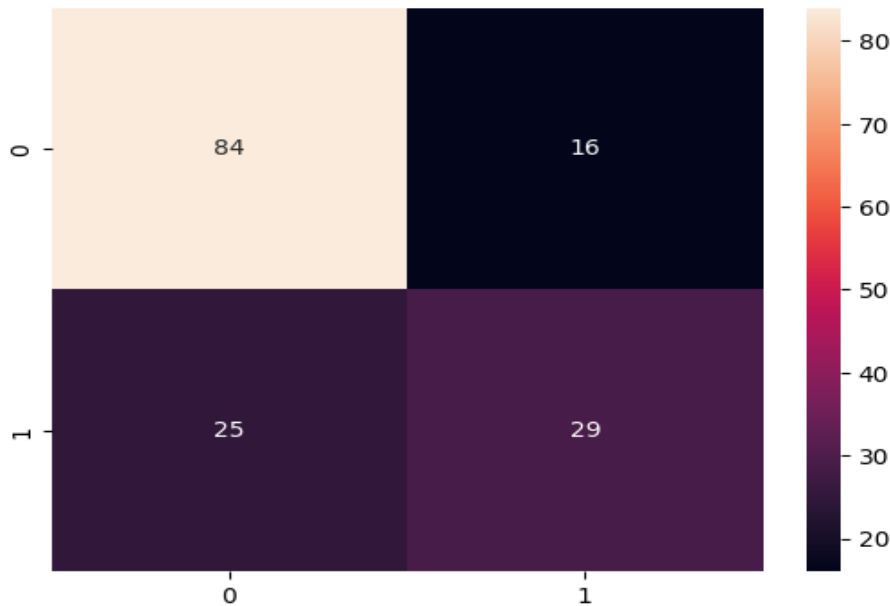
```
cm
```

```
array([[84, 16],  
       [25, 29]])
```

```
# Heatmap of Confusion matrix
```

```
sns.heatmap(pd.DataFrame(cm), annot=True)
```

```
<Axes: >
```

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(Y_test, y_predict)
accuracy
```

```
0.7337662337662337
```

```
y_predict = model.predict([[1,148,72,35,79.799,33.6,0.627,50]])
```

```
print(y_predict)
```

```
if y_predict==1:
```

```
    print("Diabetic")
```

```
else:
```

```
    print("Non Diabetic")
```

```
[1]
```

```
Diabetic
```

```
/opt/conda/lib/python3.10/site-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LogisticRegression was fitted with feature names
```

```
warnings.warn(
```

CONCLUSION:

In conclusion ,implementing innovative techniques such as machine learning algorithms ,big data analytics, and continuous model refinement are pivotal in enhancing prediction system accuracy. Embracing advancements in artificial intelligence, leveraging diverse data sources and fostering a culture of ongoing research and development can significantly contribute to the precision and reliability of prediction systems, ensuring their effectiveness in various domains and industries.