***Machine Learning***

***on***

***HEALTH CARE: DIABETES***

***By,***

***Gattu Kamal Yeshodhar Shastry***

**Abstract**

Neural networks or connectionist models for parallel processing are not new. However, a resurgence of interest in the past half decade has occurred. In part, this is related to a better understanding of what are now referred to as hidden nodes. These algorithms are considered to be of marked value in pattern recognition problems. Because of that, we tested the ability of an early neural network model, ADAP, to forecast the onset of diabetes mellitus in a high risk population of Pima Indians. The algorithm's performance was analyzed using standard measures for clinical tests: sensitivity, specificity, and a receiver operating characteristic curve. The crossover point for sensitivity and specificity is 0.76. We are currently further examining these methods by comparing the ADAP results with those obtained from logistic regression and linear perceptron models using precisely the same training and forecasting sets. A description of the algorithm is included.

**INDEX**

|  |  |  |
| --- | --- | --- |
|  | **Content** | *Pg no.* |
| 01 | Introduction – Problem Statement | 1 |
| 02 | Dataset Description | 2 |
| 03 | Architecture for Problem Statement | 3 |
| 04 | Machine Learning model | 4-5 |
| 05 | Results and Evaluation | 6-8 |
| 06 | Reference | 9 |
|  | *Appendix* |  |

**01 – Introduction**

**Problem Statement:**

Build a Machine Learning Model to accurately predict whether or not the patients in the dataset have diabetes or not using below attributes

Pregnancies

Number of times pregnant

Numeric

Glucose

Plasma glucose concentration a 2 hour in an oral glucose tolerance test

Numeric

BloodPressure

Diastolic blood pressure (mm Hg)

Numeric

SkinThickness

Triceps skin fold thickness (mm)

Numeric

Insulin

2-Hour serum insulin (mu U/ml)

Numeric

BMI

Body mass index (weight in kg/ (height in m)^2)

Numeric

DiabetesPedigreeFunction

Diabetes pedigree function

Numeric

Age

Age (years)

Numeric

**02-Dataset Description**

**Attributes:**

* Pregnancies (Number of times pregnant)
* Glucose (Plasma glucose concentration a 2 hour in an oral glucose tolerance test)
* BloodPressure (Diastolic blood pressure (mm Hg))
* SkinThickness (Triceps skin fold thickness (mm))
* Insulin (2-Hour serum insulin (mu U/ml))
* BMI (Body mass index (weight in kg/ (height in m) ^2))
* DiabetesPedigreeFunction (Diabetes pedigree function)
* Age (Age (years))

**03-Architecture of Problem Statement**

**Steps for solving problem:**

* Import required libraries
* Read the data from the database
* Correlate the data
* Split the data for training and testing
* Visualize data
* Transform the data into Normalised form or Standard Scaled Form
* Build Machine Learning model using suitable Regression or Classification techniques
* Evaluate the model and build Confusion matrix and find Classification report

**04-Machine Learning Model**

**Predicted values according to**

* Logistic Regression:

1 0 0 1 0 0 1 1 0 0 1 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0

0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 1 1 1 1 0 0 0 0 0 0 1

1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 1 0 0 0 0 1 0

0 1 0 1 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0

0 0 0 1 0 0

* K Nearest Neighbor:

1 0 0 1 0 0 1 1 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 1 0 0 0

0 0 1 0 0 1 1 1 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 1 1 0 1 1 1 1 0 1 0 0 1 0 1

1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 0 1 0

0 1 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 0

0 0 0 0 0 0

* Support Vector Machine:

1 0 0 1 0 0 1 1 1 0 1 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0

0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 1 1 1 1 0 0 0 0 0 0 1

1 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 1 0 0 0 0 1 0

0 1 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0

0 0 0 0 0 0

* Decision Tree:

1 0 0 1 0 0 1 0 0 1 1 0 0 0 0 1 1 0 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 0 1 0

0 1 1 1 0 0 1 0 0 0 0 1 0 1 1 1 1 0 0 0 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 1

1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 1 0 0 0 1 0

1 0 1 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 1 0 1 1 0 0 1 0 0 1 0 0 0

0 0 0 0 0 0

* Random Forest:

1 0 0 1 0 0 1 1 0 0 1 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 1

0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 1 1 0 1 0 1 0 0 0 0 1

1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 1 0 1 0 0 0

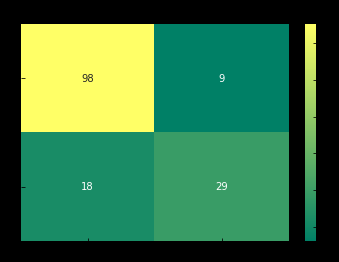
0 1 1 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 1 0 0 1 0 0 1 0 0 0

0 0 0 0 0 0

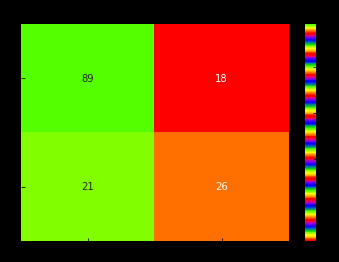
**05-Results and Evaluation**

**Confusion Matrix:**

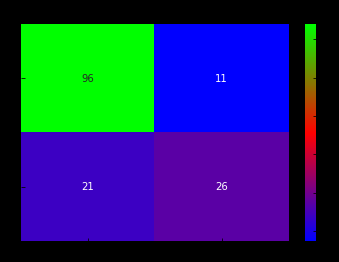
Logistic Regression:



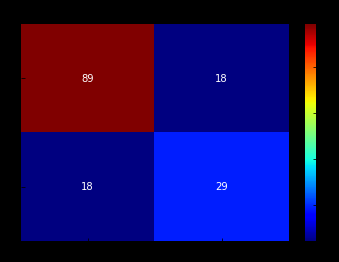
K nearest Neighbor



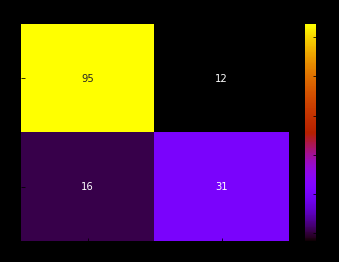
Support Vector Machine



Decision Tree



Random Forest Tree



**Classification Report:**

**Logistic Regression**

precision recall f1-score support

0 0.84 0.92 0.88 107

1 0.76 0.62 0.68 47

avg / total 0.82 0.82 0.82 154

**K Nearest Neighbor**

precision recall f1-score support

0 0.81 0.83 0.82 107

1 0.59 0.55 0.57 47

avg / total 0.74 0.75 0.74 154

**Support Vector Machine**

precision recall f1-score support

0 0.82 0.90 0.86 107

1 0.70 0.55 0.62 47

avg / total 0.78 0.79 0.78 154

**Decision tree**

precision recall f1-score support

0 0.83 0.83 0.83 107

1 0.62 0.62 0.62 47

avg / total 0.77 0.77 0.77 154

**Random Forest**

precision recall f1-score support

0 0.86 0.89 0.87 107

1 0.72 0.66 0.69 47

avg / total 0.81 0.82 0.82 154

**URL:**

<https://ibm-watson-ml.eu-gb.bluemix.net/v3/wml_instances/fc7e6a05-cdd3-465e-baac-000fe7bed127/published_models/a05a9a20-e747-4cdb-af70-ed0177f1191f/deployments/d0c3a6bb-eb39-4097-9fa2-fac8814a2d3d/online>

**WML Credentials:**

"username": "8ca38a68-61d6-446b-855d-5dd787f30be2",

"password": "e5050cc5-6071-447e-befc-7642e62ff10e",

"instance\_id": "fc7e6a05-cdd3-465e-baac-000fe7bed127",

"url": "https://ibm-watson-ml.eu-gb.bluemix.net"**06-Reference**

**Link:**

[**https://www.kaggle.com/ratnesh88/indian-diabetes-prediction/data**](https://www.kaggle.com/ratnesh88/indian-diabetes-prediction/data)

***Appendix***

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

get\_ipython().magic(u'matplotlib inline')

import seaborn as sns

plt.style.use('bmh')

import sys

import types

import pandas as pd

from botocore.client import Config

import ibm\_boto3

def \_\_iter\_\_(self): return 0

# @hidden\_cell

# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.

# You might want to remove those credentials before you share your notebook.

client\_09c657f29b02424fa7490dd3b3e67651 = ibm\_boto3.client(service\_name='s3',

ibm\_api\_key\_id='W4VUcZtgE-Ec0Ub6jkoreyGspLNLwSXL\_Qfrlb\_JN5aX',

ibm\_auth\_endpoint="https://iam.eu-gb.bluemix.net/oidc/token",

config=Config(signature\_version='oauth'),

endpoint\_url='https://s3.eu-geo.objectstorage.service.networklayer.com')

body = client\_09c657f29b02424fa7490dd3b3e67651.get\_object(Bucket='teamb304kamal-donotdelete-pr-wou516kxtnupfb',Key='diabetes.csv')['Body']

# add missing \_\_iter\_\_ method, so pandas accepts body as file-like object

if not hasattr(body, "\_\_iter\_\_"): body.\_\_iter\_\_ = types.MethodType( \_\_iter\_\_, body )

df\_data\_1 = pd.read\_csv(body)

df\_data\_1.head()

df\_data\_1.info()

corr=df\_data\_1.corr()

plt.figure(figsize=(10,4))

sns.heatmap(corr,annot=True,cmap='summer')

plt.show()

x=df\_data\_1.iloc[:,:-1].values # Independant variables

y=df\_data\_1.iloc[:,-1].values #dependant variables

x.shape,y.shape

plt.figure(figsize=(15,6))

plt.boxplot(x,vert =False,labels=['Pregnancies','Glucose','Blood Pressure','Skin Thickness','Insulin','BMI','DPF','Age'],

patch\_artist=True)

plt.show()

from sklearn.preprocessing import StandardScaler,MinMaxScaler

sc=StandardScaler() #z-score

mms=MinMaxScaler() #(0-1)->normalisation

x\_sc =sc.fit\_transform(x)

x\_norm=mms.fit\_transform(x)

fig=plt.figure(figsize=(15,6))

plt.style.use('bmh')

# Without scaling

plt.boxplot(x,vert=False,labels=['Pregnancies','Glucose','Blood Pressure','Skin Thickness','Insulin','BMI','DPF','Age'],patch\_artist=True)

plt.title('Without Scaling')

plt.show()

# Normalisation

fig=plt.figure(figsize=(15,6))

plt.boxplot(x\_norm,vert=False,labels=['Pregnancies','Glucose','Blood Pressure','Skin Thickness','Insulin','BMI','DPF','Age'],patch\_artist=True)

plt.title('Normalisation(0-1)')

plt.show()

# Standard scaling

fig=plt.figure(figsize=(15,6))

plt.boxplot(x\_sc,vert=False,labels=['Pregnancies','Glucose','Blood Pressure','Skin Thickness','Insulin','BMI','DPF','Age'],patch\_artist=True)

plt.title('Standard Scaling(Z-score)')

plt.show()

from sklearn.cross\_validation import train\_test\_split

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x\_sc,y,test\_size=0.2,random\_state=0)

x\_train.shape,y\_train.shape,x\_test.shape,y\_test.shape

from sklearn.linear\_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

model\_log= LogisticRegression(C=10.0) # class

model\_knn= KNeighborsClassifier(n\_neighbors=3)

model\_svm= SVC(kernel='rbf')

model\_dt= DecisionTreeClassifier()

model\_rf= RandomForestClassifier(n\_estimators=100)

model\_log.fit(x\_train,y\_train)

model\_knn.fit(x\_train,y\_train)

model\_svm.fit(x\_train,y\_train)

model\_dt.fit(x\_train,y\_train)

model\_rf.fit(x\_train,y\_train)

print('Model trained successfully')

y\_pred\_log=model\_log.predict(x\_test)

y\_pred\_knn=model\_knn.predict(x\_test)

y\_pred\_svm=model\_svm.predict(x\_test)

y\_pred\_dt=model\_dt.predict(x\_test)

y\_pred\_rf=model\_rf.predict(x\_test)

from sklearn.metrics import confusion\_matrix,classification\_report

cm\_log= confusion\_matrix(y\_test,y\_pred\_log)

cm\_knn= confusion\_matrix(y\_test,y\_pred\_knn)

cm\_svm= confusion\_matrix(y\_test,y\_pred\_svm)

cm\_dt= confusion\_matrix(y\_test,y\_pred\_dt)

cm\_rf= confusion\_matrix(y\_test,y\_pred\_rf)

fig=plt.figure(figsize=(30,18))

plt.subplot(2,3,1)

sns.heatmap(cm\_log,annot=True,cmap='summer')

plt.title('Logistic Regression')

plt.subplot(2,3,2)

sns.heatmap(cm\_knn,annot=True,cmap='prism')

plt.title('K Nearest Neighbor ')

plt.subplot(2,3,3)

sns.heatmap(cm\_svm,annot=True,cmap='brg',)

plt.title('Support Vector Machine')

plt.subplot(2,3,4)

sns.heatmap(cm\_dt,annot=True,cmap='jet',)

plt.title('Decision Tree')

plt.subplot(2,3,5)

sns.heatmap(cm\_rf,annot=True,cmap='gnuplot',)

plt.title('Random Forest Tree')

plt.show()

cr\_log=classification\_report(y\_test,y\_pred\_log)

cr\_knn=classification\_report(y\_test,y\_pred\_knn)

cr\_svm=classification\_report(y\_test,y\_pred\_svm)

cr\_dt=classification\_report(y\_test,y\_pred\_dt)

cr\_rf=classification\_report(y\_test,y\_pred\_rf)

print("\*"\*20+'Logistic Regression'+"\*"\*20)

print(cr\_log)

print("\*"\*20+'K Nearest Neighbor'+"\*"\*20)

print(cr\_knn)

print("\*"\*20+'Support Vector Machine'+"\*"\*20)

print(cr\_svm)

print("\*"\*20+'Decision tree'+"\*"\*20)

print(cr\_dt)

print("\*"\*20+'Random Forest'+"\*"\*20)

print(cr\_rf)

from watson\_machine\_learning\_client import WatsonMachineLearningAPIClient

wml\_credentials ={

"username": "8ca38a68-61d6-446b-855d-5dd787f30be2",

"password": "e5050cc5-6071-447e-befc-7642e62ff10e",

"instance\_id": "fc7e6a05-cdd3-465e-baac-000fe7bed127",

"url": "https://ibm-watson-ml.eu-gb.bluemix.net"

}

client = WatsonMachineLearningAPIClient(wml\_credentials)

import json

instance\_details = client.service\_instance.get\_details()

published\_model = client.repository.store\_model(model=model\_log, meta\_props={'name':'Diabetes'},

training\_data=x\_train, training\_target=y\_train)

published\_model\_uid = client.repository.get\_model\_uid(published\_model)

model\_details = client.repository.get\_details(published\_model\_uid)

print(json.dumps(model\_details, indent=2))

models\_details = client.repository.list\_models()

loaded\_model = client.repository.load(published\_model\_uid)

test\_predictions = loaded\_model.predict(x\_test[:5])

print(test\_predictions)

# client.repository.delete(published\_model\_uid)

created\_deployment = client.deployments.create(published\_model\_uid, "Health")

scoring\_endpoint = client.deployments.get\_scoring\_url(created\_deployment)

print(scoring\_endpoint)

sc.mean\_

sc.var\_

x\_sc

import urllib3, requests, json

# retrieve your wml\_service\_credentials\_username, wml\_service\_credentials\_password, and wml\_service\_credentials\_url from the

# Service credentials associated with your IBM Cloud Watson Machine Learning Service instance

wml\_credentials ={

"username": "8ca38a68-61d6-446b-855d-5dd787f30be2",

"password": "e5050cc5-6071-447e-befc-7642e62ff10e",

"instance\_id": "fc7e6a05-cdd3-465e-baac-000fe7bed127",

"url": "https://ibm-watson-ml.eu-gb.bluemix.net"

}

import numpy as np

mean=np.array([ 3.84505208, 120.89453125, 69.10546875, 20.53645833,

79.79947917, 31.99257812, 0.4718763 , 33.24088542])

std=np.array([ 1.13392724e+01, 1.02091726e+03, 3.74159449e+02,

2.54141900e+02, 1.32638869e+04, 6.20790465e+01,

1.09635697e-01, 1.38122964e+02])

test=np.array([6,148,72,35,0,33,0.67,50])

z=(test-mean)/std

input\_values=list(z)

url\_model='https://ibm-watson-ml.eu-gb.bluemix.net/v3/wml\_instances/fc7e6a05-cdd3-465e-baac-000fe7bed127/published\_models/a05a9a20-e747-4cdb-af70-ed0177f1191f/deployments/d0c3a6bb-eb39-4097-9fa2-fac8814a2d3d/online'

headers = urllib3.util.make\_headers(basic\_auth='{username}:{password}'.format(username=wml\_credentials['username'], password=wml\_credentials['password']))

url = '{}/v3/identity/token'.format(wml\_credentials['url'])

response = requests.get(url, headers=headers)

mltoken = json.loads(response.text).get('token')

header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}

# NOTE: manually define and pass the array(s) of values to be scored in the next line

payload\_scoring = {"fields": ["f0", "f1", "f2", "f3","f4","f4","f5","f6","f7"], "values": [input\_values]}

response\_scoring = requests.post('https://ibm-watson-ml.eu-gb.bluemix.net/v3/wml\_instances/fc7e6a05-cdd3-465e-baac-000fe7bed127/published\_models/a05a9a20-e747-4cdb-af70-ed0177f1191f/deployments/d0c3a6bb-eb39-4097-9fa2-fac8814a2d3d/online', json=payload\_scoring, headers=header)

print("Scoring response")

print(json.loads(response\_scoring.text))