ENSEMBLE APPROACH FOR DETECTING DIABETES

ABSTRACT:

Diabetes is a kind of metabolic disease that forms by lack of insulin due to the malfunctioning of the pancreas. Diabetes can push a person into pathological destructionof pancreatic beta cells, coma, cardiovascular dysfunction, renal and retinal failure, joint failure, pathogenic effects on immunity, weight loss, and peripheral vascular diseases. So, for the early detection of diabetes, a robust framework was proposed, where outlier rejection, filling the missing values, data standardization, K-fold validation, and different Machine Learning (ML) classifiers (k-NN, decision trees (DT), random forest (RF), naive Bayes (NB), XGboost and Adaboost were used. To improve the result, the weighted ensembling of different ML models also proposed here. The corresponding Area Under ROC Curve (AUC) of the ML model as the performance metric estimated these weights. Using the grid search technique, the AUC is then maximized during hyperparameter tuning. All experiments were conducted under the same experimental conditions on publicly available Diabetes dataset population near Phoenix, Arizona of 768 femalediabetic patients, where there are 268 diabetic patients (positive) and 500 non-diabetic patients (negative) with eight different attributes.

AIM:

Diabetes is a group of metabolic disorders charaterzied by abnormal metabolism which results most notably in hyperglycemia due to defects in insulin secreation, insulin action or both. Diabets is a serious chronic disease without a cure and it isassociated with significant morbidity and mortality. Diabetes is a serious disesae associated with acute(due to hyperglycemia) and chronic(due to vascular damages) complications.

OBJECTIVE:

Comparing to the existing models our proposed work gives us a better effiency. So, in this analysis, Machine Learning algorithms have been used to classify diabetes.The extensive experiments for the selection of the best performing feature selection methods with selected attribute numbers.

SCOPE:

So, for the early detection of diabetes, a robust framework was proposed, where outlierrejection, filling the missing values, data standardization, K-fold validation, and differentMachine Learning (ML) classifiers (k-NN, random forest (RF),and Xgboost (XB).Adaboost were used.To improve the result, the weighted ensembling of different ML models also proposed here.The corresponding Area Under ROC Curve (AUC) of the ML model as the performance metric estimated these weights. Algorithmsused in are KNN, Random Forest (RA), ADABOOST, XGBOOST.

PROPOSED SYSTEM:

1. **Diabetes dataset:** Reading the data from diabetes dataset(here the datasetis present in csv files)
2. **Preprocessing**: preprocessing is used to process the data weather there isa null value in the data set before implementing the algorithms.
3. **Train data**: The sample of data used to fit the model. The actual dataset that we use to train the model (weights and biases of neuralnetwork) the model *sees* and *learns* from this data.

In train data we use two types:

a. **Grid search**: is the process of scanning the data for a given model. Dependingon the type of model utilized, certain parameters necessary.

b. **Hyper Parameters***:* hyperparameter is a parameter whose value is used tocontrol the learning process.

4. **Test data:** The sample of data used to provide an unbiased evaluation of a final model fit on the training dataset.

Step1: Input the diabetic dataset.

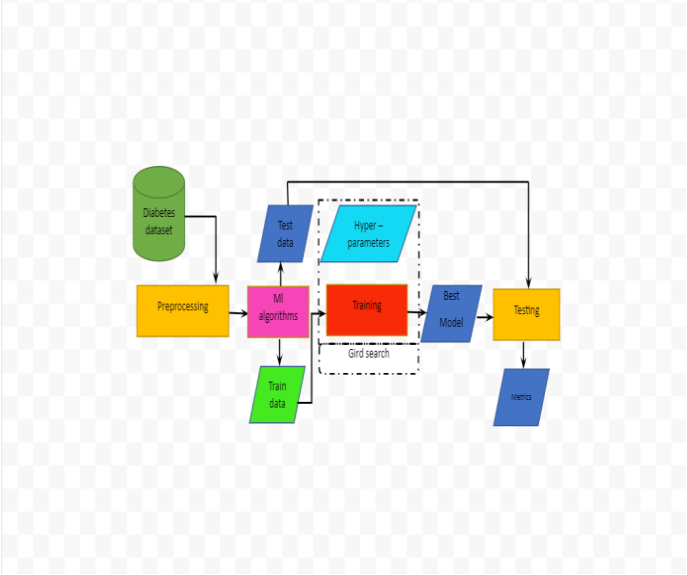
Step2: The dataset has been randomly split into training and testing data.

Step3: Applying the machine learning algorithms to predicting the diabetic disease.

Step4: Finally, the performances of classifiers are evaluated.

SYSTEM ARCHITECTURE:

In the PIMA Indians Diabetes Data Collection, we suggest a new diabetes prediction pipeline. Preprocessing is the heart of obtaining the most advanced outcome in the proposed pipeline, which is the outer refuse, Filling missed values, standard info, collection of features and cross-validation of K-fold. We see the average value in the missing attribute place rather than median value, since it is more central to the average of thedistribution of the attribute. Cross- validation of the dataset is carried out with cautionto retain the percentage of the class proportion in the same manner as in the initial dataset. Various ML (k-nearest Neighbor (kNN) grouping modules is RF, DT, NB, AB and XGBoast (XB)). Built in our pipeline suggested.



Architecture

SOURCE CODE:

```
import numpy as npimport pandas as pd
import matplotlib.pyplot as plt
%matplotlib    inlineimport seaborn as sns
# sns.set(style="whitegrid")import warnings
from sklearn.svm import SVC, NuSVCwarnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_splitfrom sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifierfrom sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifierfrom sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifierfrom sklearn.neural_network import
MLPClassifier
import xgboost as xgbfrom scipy import stats
from scipy.stats import uniform, randintfrom sklearn.metrics import f1_score
from sklearn.model_selection import KFold, StratifiedKFold, RepeatedStratifiedKFoldfrom
sklearn.metrics import roc_curve, auc, accuracy_score
# from tflearn.data_utils import to_categoricalfrom sklearn import preprocessing
 from sklearn.model_selection import GridSearchCV, RandomizedSearchCVfrom sklearn.metrics
import classification_report
rom scipy import interp
from sklearn.metrics import confusion_matrixfrom sklearn.decomposition import PCA from
sklearn.decomposition import FastICA from keras.utils import to_categorical
Renamed_feature= []                        #list of names that will rename to feature columnall_clf_res=[]
#every classifier auc values are stored in it random_initializer=100            #random initializer
n_dots=50
for i in range(8):
#for renaming dataset of columns features F1 -- F8Renamed_feature.append('F'+str(i+1))
Parameters :
Input -
data is the pandas type variable iqr_Mean
```

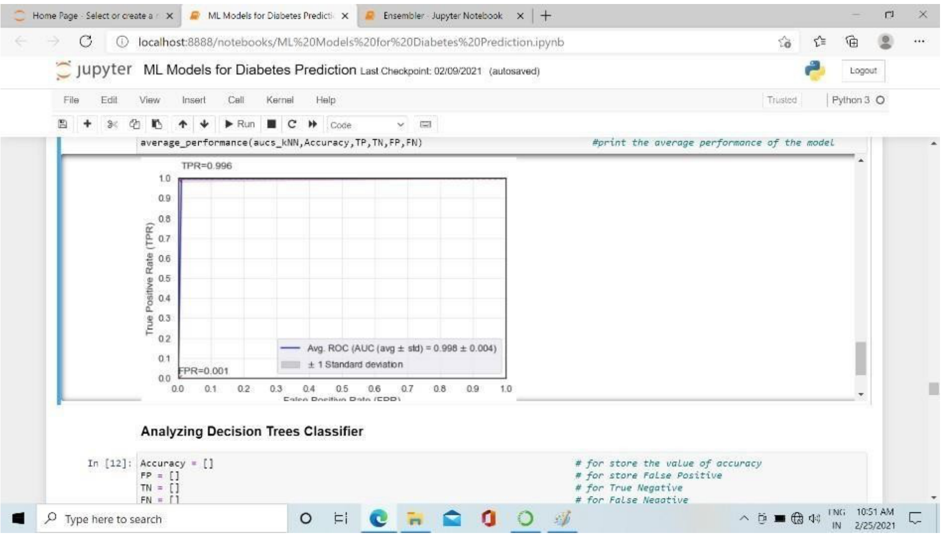
- for outleir rejection with Mean

iqr_Medain- for outleir rejection with Medainiqr- for drop the outleir

manual -for manual rejection

Return - dataframe with outleir rejectionfilled with Input parameter

OUTPUTS:



ML Models for Diabetes Prediction Last Checkpoint: 02/09/2021 (autosaved)

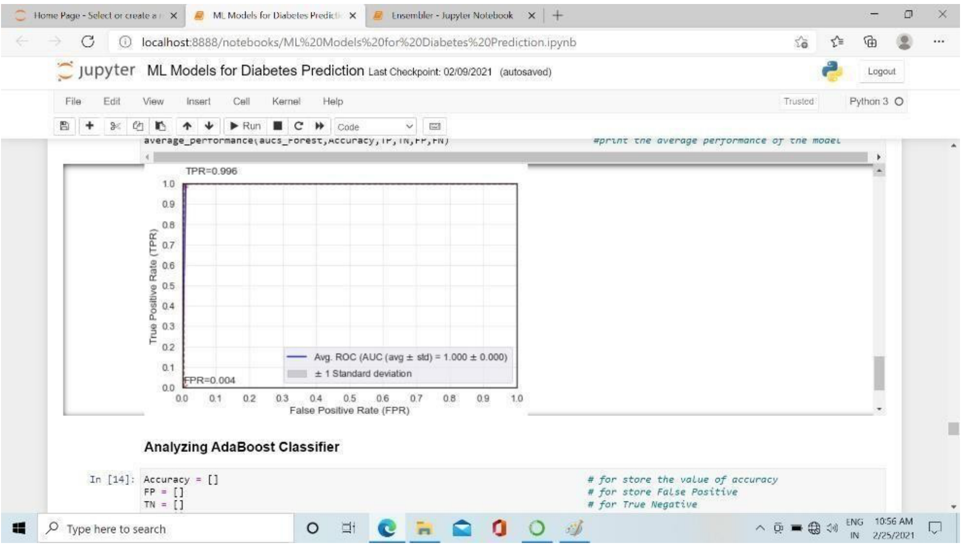
```
average_performance(aucs_kNN,Accuracy,TP,TN,FP,FN)                    #print the average performance of the model
```

Analyzing Decision Trees Classifier

```
In [12]: Accuracy = []                                                # for store the value of accuracy
         FP = []                                                      # for store False Positive
         TN = []                                                      # for True Negative
         FN = []                                                      # for False Negative
```

GRAPH FOR KNN

Here the KNN gets an accuracy of 99%


Home Page - Select or create a
ML Models for Diabetes Predicti
Ensembler - Jupyter Notebook
localhost:8888/notebooks/ML%20Models%20for%20Diabetes%20Prediction.ipynb
jupyter
ML Models for Diabetes Prediction Last Checkpoint: 02/09/2021 (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Help
Trusted
Python 3
Run
Code
TPR=0.996
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
True Positive Rate (TPR)
Avg. ROC (AUC (avg ± std) = 1.000 ± 0.000)
± 1 Standard deviation
FPR=0.004
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
False Positive Rate (FPR)
Type here to search
ENG IN
10:56 AM 2/25/2021

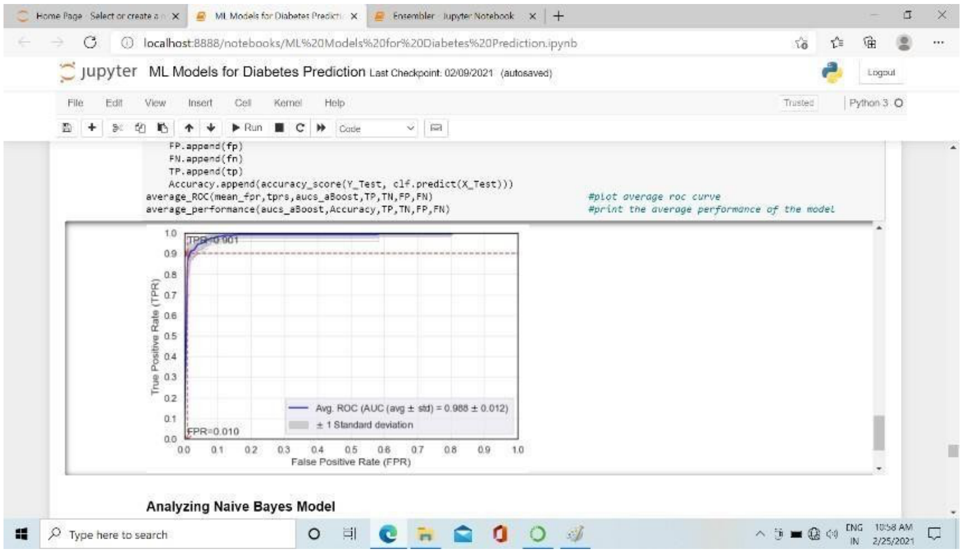
```
average_performance(aucs_Forest,Accuracy,TP,TN,FP,FN)                    #print the average performance of the model
```

Analyzing AdaBoost Classifier

```
In [14]: Accuracy = []                                                   # for store the value of accuracy
         FP = []                                                         # for store False Positive
         TN = []                                                         # for True Negative
```

GRAPH OF RF

Here the RF gets an accuracy of 99%

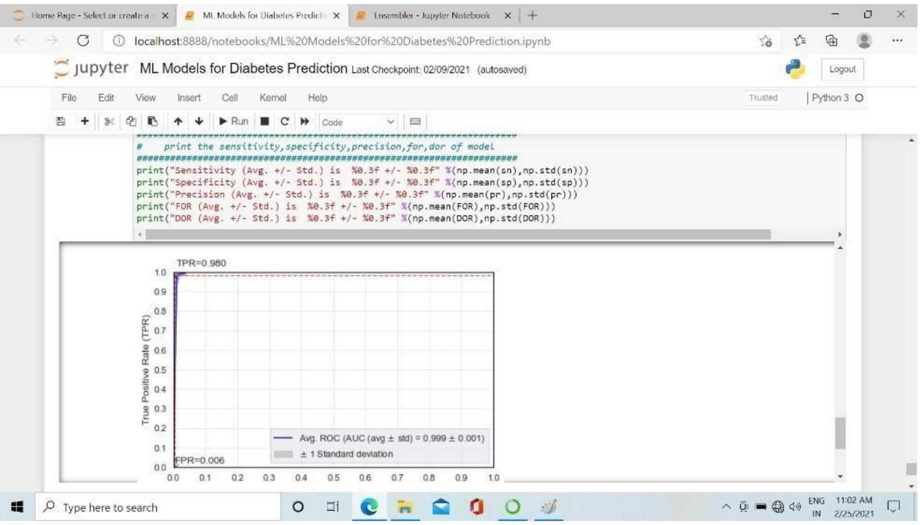

Home Page Select or create a
ML Models for Diabetes Predicti
Ensembler Jupyter Notebook
localhost:8888/notebooks/ML%20Models%20for%20Diabetes%20Prediction.ipynb
jupyter
ML Models for Diabetes Prediction Last Checkpoint: 02/09/2021 (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Help
Trusted
Python 3
Run
Code
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
TPR=0.901
True Positive Rate (TPR)
Avg. ROC (AUC (avg ± std) = 0.988 ± 0.012)
± 1 Standard deviation
FPR=0.010
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
False Positive Rate (FPR)
Type here to search
ENG IN
10:58 AM 2/25/2021

```
    FP.append(fp)
    FN.append(fn)
    TP.append(tp)
    Accuracy.append(accuracy_score(Y_Test, clf.predict(X_Test)))
average_ROC(mean_fpr,tprs,aucs_aBoost,TP,TN,FP,FN)                       #plot average roc curve
average_performance(aucs_aBoost,Accuracy,TP,TN,FP,FN)                    #print the average performance of the model
```

Analyzing Naive Bayes Model

GRAPH FOR ADABOOST

Here the ADABOOST gives an accuracy of 98%


Home Page - Select or create a
ML Models for Diabetes Predicti
Ensembler - Jupyter Notebook
localhost:8888/notebooks/ML%20Models%20for%20Diabetes%20Prediction.ipynb
jupyter ML Models for Diabetes Prediction Last Checkpoint: 02/09/2021 (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Trusted
Python 3
Run
Code
TPR=0.980
True Positive Rate (TPR)
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
FPR=0.006
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
Avg. ROC (AUC (avg ± std) = 0.999 ± 0.001)
± 1 Standard deviation
Type here to search
ENG IN
11:02 AM
2/25/2021

```
###########################################################################
#    print the sensitivity,specificity,precision,for,dor of model
###########################################################################
print("Sensitivity (Avg. +/- Std.) is  %0.3f +/- %0.3f" %(np.mean(sn),np.std(sn)))
print("Specificity (Avg. +/- Std.) is  %0.3f +/- %0.3f" %(np.mean(sp),np.std(sp)))
print("Precision (Avg. +/- Std.) is  %0.3f +/- %0.3f" %(np.mean(pr),np.std(pr)))
print("FOR (Avg. +/- Std.) is  %0.3f +/- %0.3f" %(np.mean(FOR),np.std(FOR)))
print("DOR (Avg. +/- Std.) is  %0.3f +/- %0.3f" %(np.mean(DOR),np.std(DOR)))
```

GRAPH FOR XGBOOST

Here the XGBOOST gives an accuracy of 99%

SUMMARY AND CONCLUSION

Diabetes mellitus (DM), is a series of metabolic disorders in human body due to high level of blood sugar in body. If left untreated, it can cause many complications in the long run. Diabetes is majorly caused due to dis-functioning of pancreas leading to failure in production of required insulin. From the above survey its understood thatitis always better to use more than one classifier for predicting the output as its accuracyis higher than accuracy of single classifier. Bagging ensemble classifier should be usedin the predicting System so that can obtain better and efficient results. Diabetes mellitus(DM), is a series of metabolic disorders in human body due to high level of blood sugarin body. If left untreated, it can cause many complications in the long run. Diabetes is majorly caused due to dis-functioning of pancreas leading to failure in production of required insulin. From the above survey its understood thatitis always better to use more than one classifier for predicting the output as its accuracy is higher than accuracyof single classifier. Bagging ensemble classifier should be used in the predicting System so that can obtain better and efficient results.